UNIVERSIDAD POLITÉCNICA DE VALENCIA

UNIVERSIDAD
POLITECNICA
DE VALENCIA

Heuristics and metaheuristics for heavily constrained hybrid flowshop problems

Thijs Urlings

Submitted in fulfillment of the requirements of
the degree of
DOCTOR OF PHILOSOPHY

Supervised by:
Rubén Ruiz García

Valencia, 2010

Es ist nicht genug zu wissen, man muss auch anwenden.
Es ist nicht genug zu wollen, man muss auch tun.

Knowing is not enough, we must apply.
Willing is not enough, we must do.

Johann Wolfgang von Goethe,
Wilhelm Meisters Wanderjahre, 1821.

ABSTRACT

HEURISTICS AND METAHEURISTICS FOR HEAVILY CONSTRAINED HYBRID FLOWSHOP PROBLEMS

Due to the current trends in business as the necessity to have a large catalogue of products, orders that increase in frequency but not in size, globalisation and a market that is increasingly competitive, the production sector faces an ever harder economical environment. All this raises the need for production scheduling with maximum efficiency and effectiveness.

The first scientific publications on production scheduling appeared more than half a century ago. However, many authors have recognised a gap between the literature and the industrial problems. Most of the research concentrates on optimisation problems that are actually a very simplified version of reality. This allows for the use of sophisticated approaches and guarantees in many cases that optimal solutions are obtained. Yet, the exclusion of real-world restrictions harms the applicability of those methods. What the industry needs are systems for optimised production scheduling that adjust exactly to the conditions in the production plant and that generates good solutions in very little time. This is exactly the objective in this thesis, that is, to treat more realistic scheduling problems and to help closing the gap between the literature and practice.

The considered scheduling problem is called the hybrid flowshop problem, which consists in a set of jobs that flow through a number of production stages. At each of the stages, one of the machines that belong to the stage is visited.

A series of restriction is considered that include the possibility to skip stages, non-eligible machines, precedence constraints, positive and negative time lags and sequence dependent setup times. In the literature, such a large number of restrictions has not been considered simultaneously before. Briefly, in this thesis a very realistic production scheduling problem is studied.

Various optimisation methods are presented for the described scheduling problem. A mixed integer programming model is proposed, in order to obtain optimal solutions for limited cases and in order to analyse the complexity of each of the problem restrictions. In continuation, seven constructive heuristics are presented with the purpose of obtaining fast solutions to the problem in more general cases. Advanced metaheuristic methods are studied in detail, starting with five genetic algorithms that allow studying the effect of the solution representation. Three local search based algorithms are proposed, which is very novel if the elevated complexity of the problem is taken into account. In addition, novel methods are presented that shift the solution representation during the search process in order to acquire near-optimal solutions. The obtained results endorse the use of these new shifting techniques.

In the literature, hardly any publications appear that treat multi-objective optimisation for the hybrid flowshop problem. In this Ph.D. thesis, two metaheuristics that produce Pareto fronts for this problem are presented. It is shown that it is not obvious to measure the results, and a methodology in order to do so is proposed, using state-of-the-art techniques. Finally, practical applications are commented in the scope of technology transfer towards companies.

RESUMEN

HEURÍSTICAS Y METAHEURÍSTICAS PARA PROBLEMAS DE TALLER DE FLUJO HÍBRIDO ALTAMENTE RESTRINGIDOS

Debido a las actuales tendencias empresariales como la necesidad de tener un catálogo de productos amplio, pedidos que aumentan en frecuencia pero no en tamaño, la globalización y un mercado donde la competitividad aumenta, el sector de la producción encara un entorno económico cada vez más duro. Todo esto requiere de una programación de la producción con la máxima eficiencia y eficacia.

Las primeras publicaciones científicas sobre la programación de la producción aparecieron hace más de medio siglo. Sin embargo, muchos autores han reconocido una brecha entre la literatura y la problemática industrial. La mayoría de la investigación se concentra en problemas de optimización que no son más que una versión muy simplificada de la realidad. Esto permite el uso de métodos sofisticados y garantiza la obtención de soluciones óptimas en muchos casos. No obstante, la exclusión de restricciones existentes en el mundo real complica la aplicabilidad de dichos métodos. Lo que necesita la industria son sistemas de programación de la producción optimizada que se ajusten exactamente a la situación de la planta y que den buenas soluciones en muy poco tiempo. El objetivo de esta tesis doctoral es precisamente este, el de tratar problemas de programación más realistas y el de ayudar a cerrar la brecha entre la literatura y la práctica.

El problema de producción tratado es conocido como taller de flujo híbrido, que consiste en un conjunto de trabajos que pasan por varias etapas productivas. En cada etapa se visita una de las máquinas que pertenecen a la etapa. Se consideran una serie de restricciones que incluyen la posibilidad de saltar etapas, máquinas no elegibles, relaciones de precedencia, solapes y esperas y tiempos de cambio dependientes de la secuencia. Hasta la fecha, en la literatura no se ha considerado tal cantidad de restricciones simultáneamente. En conclusión, en esta tesis se estudia un problema muy realista de programación de la producción.

Para este problema, se presentan varios métodos de optimización. Se propone un modelo matemático para obtener soluciones exactas en casos limitados y para analizar la complejidad de cada una de las restricciones. Se proponen siete heurísticas con el fin de obtener soluciones rápidas en casos generales. Diversos métodos metaheurísticos avanzados se estudian en detalle, empezando con cinco algoritmos genéticos que permiten el estudio del efecto de la representación de la solución. Se proponen tres métodos basados en búsqueda local, algo muy novedoso si se tiene en cuenta la enorme dificultad del problema estudiado. Adicionalmente, se estudian métodos novedosos que cambian de representación de solución durante el proceso de búsqueda para así obtener soluciones de muy alta calidad. Los resultados conseguidos avalan el uso de estas nuevas técnicas cambiantes.

En la literatura apenas hay publicaciones que tratan sobre la optimización multi-objetivo del taller de flujo híbrido. En esta tesis doctoral se presentan dos metaheurísticas que producen como resultado fronteras de Pareto para este problema. Se demuestra que no es obvia la manera de medir los resultados y se propone una metodología para ello, usando técnicas consideradas como estado del arte. Finalmente, se comentan aplicaciones prácticas en el ámbito de la transferencia tecnológica hacia empresas.

RESUM

HEURÍSTIQUES I METAHEURÍSTIQUES PER A PROBLEMES DE TALLER DE FLUX HÍBRID ALTAMENT RESTRINGITS

A causa de les actuals tendències empresarials com ara la necessitat de tenir un catàleg de productes ampli, comandes que augmenten en freqüència però no en grandària, la globalització i un mercat on la competitivitat augmenta, el sector de la producció afronta un entorn econòmic cada vegada més dur. Tot això requereix d'una programació de la producció amb la màxima eficiència i eficàcia.

Les primeres publicacions científiques sobre la programació de la producció van aparèixer fa més de mig segle. No obstant això, molts autors han reconegut una bretxa entre la literatura i la problemàtica industrial. La majoria de la investigació es concentra en problemes d'optimització que no són més que una versió molt simplificada de la realitat. Això permet l'ús de mètodes sofisticats i garanteix l'obtenció de solucions òptimes en molts casos. No obstant això, l'exclusió de restriccions existents en el món real complica l'aplicabilitat d'aquests mètodes. El que necessita la indústria són sistemes de programació de la producció optimitzada que s'ajusten exactament a la situació de la planta i que donen bones solucions en molt poc temps. Aquest és precisament l'objectiu d'aquesta tesi doctoral: tractar problemes de programació més realistes i ajudar a tancar la bretxa entre la literatura i la pràctica.

El problema de producció tractat s'anomena taller de flux híbrid, i consisteix en un conjunt de treballs que passen per diverses etapes productives. En cada etapa es visita una de les màquines que pertanyen a l'etapa. S'hi consideren una sèrie de restriccions que comprenen la possibilitat de saltar etapes, màquines no elegibles, relacions de precedència, solapamentes i esperes i temps de canvi depenents de la seqüència. Fins a la data, en la literatura no s'ha considerat tal quantitat de restriccions simultàniament. En conclusió, en aquesta tesi s'estudia un problema molt realista de programació de la producció.

Per a aquest problema, es presenten diversos mètodes d'optimització. Es proposa un model matemàtic per a obtenir solucions exactes en casos limitats i per a analitzar la complexitat de cadascuna de les restriccions. Es proposen set heurístics amb la finalitat d'obtenir solucions ràpides en casos generals. Diversos mètodes metaheurístics avançats s'estudien en detall, tot començant amb cinc algorismes genètics que permeten l'estudi de l'efecte de la representació de la solució. Es proposen tres mètodes basats en recerca local, una cosa molt nova si es té en compte l'enorme dificultat del problema estudiat. Addicionalment, s'estudien mètodes nous que canvien la representació de la solució durant el procés de recerca per a obtenir així solucions de molt alta qualitat. Els resultats obtinguts avalen l'ús d'aquestes noves tècniques canviants.

En la literatura a penes hi ha publicacions que tracten sobre l'optimització multi-objectiu del taller de flux híbrid. En aquesta tesi doctoral es presenten dues metaheurístiques que produïxen com resultat fronteres de Pareto per a aquest problema. Es demostra que no és òbvia la manera de mesurar els resultats i es proposa una metodologia per a això, mitjançant l'ús de tècniques considerades com estat de l'art. Finalment, es comenten aplicacions pràctiques en l'àmbit de la transferència tecnològica cap a empreses.

Acknowledgements

Although it might seem different for those who are not involved, working on science is not an individual thing. Apart from the fact that one always has to be conscious of the related literature and the work of others, nobody manages to finish a Ph.D. thesis without help and support of his professional and private surroundings. Many are those that have played a direct or indirect role in making this thesis possible. These pages are dedicated to them, in order to express my thankfulness. It is a hard task to sum up all, but there are some that I would like to mention explicitly.

First of all, Dr. Rubén Ruiz deserves my absolute thankfulness and admiration. His dedication and conviction and the capability of always coming up with new ideas, as well as his successful coordination our team, are a great example to me. His speed and precision when revising drafts of articles or of this thesis makes the cooperation with him a privilege.

My colleagues of the Sistemas de Optimización Aplicada group have also played an important role in introducing me in the world of research and bringing this thesis to a good end. The continuous exchange of ideas, knowledge sharing and teaching and learning among each other, has been a fruitful environment for developing the needed both technical and scientific skills. I am especially grateful to Gerardo Minella. Working side-to-side with him for four years and a half has been a pleasure and a luxury because of his constant willingness to help me solve my programming and other informatics problems. The multi-objective advances in this thesis are thanks to the fruitful cooperation with him, in which his expertise and efficiency have become clear once again. Furthermore I want to mention Dr. Michele Ciavotta, whose ideas for multi-objective scheduling in flowshop problems were useful for the research done in this thesis. Thankful am I to Dr. Eva Vallada for her continuous attention and for helping me with many administrative and practical details. For the table conversion from StatGraphics to LaTeX format, the idea and initial code by Alejandro Rodriguez have been of great value. I want to name the remaining members of the group for their fellowship: Ketrina, Kostanca, Carlos, Pablo, Catalin, Javier and Luis.

I owe special thanks to Dr. Thomas Stützle, whose guidance, fresh ideas and structured way of thinking and working turned my stay in the Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle into a valuable and important period of my Ph.D. Moreover, he proved his merit as a coauthor, supplying me with detailed reviews. The assistance of my coauthor Prof. Dr. Funda Sivrikaya Şerifoğlu has been of key importance for the literature review and for the different solution representations. Worth mentioning has been the influence of Dr. Tjark Vredeveld, of the University of Maastricht. As the corrector of my Masters thesis, that was the basis for this Ph.D. thesis, he served me with his advice and gave some interesting indications.

This Ph.D. thesis would not have been possible without the support of the Instituto Technológico de Informática, where I have been working since 2007 and where I have been able to dedicate part of my time to the research that has resulted in this thesis.

Thanks to all.

To my mother and my father, to Sanne and to Ivo,
for giving me a solid basis of thrust, curiosity and
persistence.

To Carolina, for inspiring and motivating me and
for creating the optimal conditions to reach my
goals.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION AND OBJECTIVES

In recent times, markets have become more and more international. Internet and other technological developments have increased the speed of communication, the reach of marketing and the possibilities for distribution. Globalisation causes greater competition, since the physical distance of competitors loses importance. These developments demand for a further customisation of the products a company aims to sell. Clients want to be able to decide on the features of the product they buy and to choose the configuration they like. This asks for a wide range of products and makes mass production of a single unique product a weak business model.

Within the decision making process we can distinguish between long, medium and short term decisions. Long term or strategic decisions, are for example how many machines to buy, whether to invest in machinery or to rent it, or what products to offer to the market. Examples of medium term decisions are client order acceptance, personnel planning or lot sizing. Scheduling however, purely involves short time decisions, where the main question is: Which task to process at what moment on which machine? An overview of the categorisation of decisions to be taken in a production company, can be found in Figure 1.1.

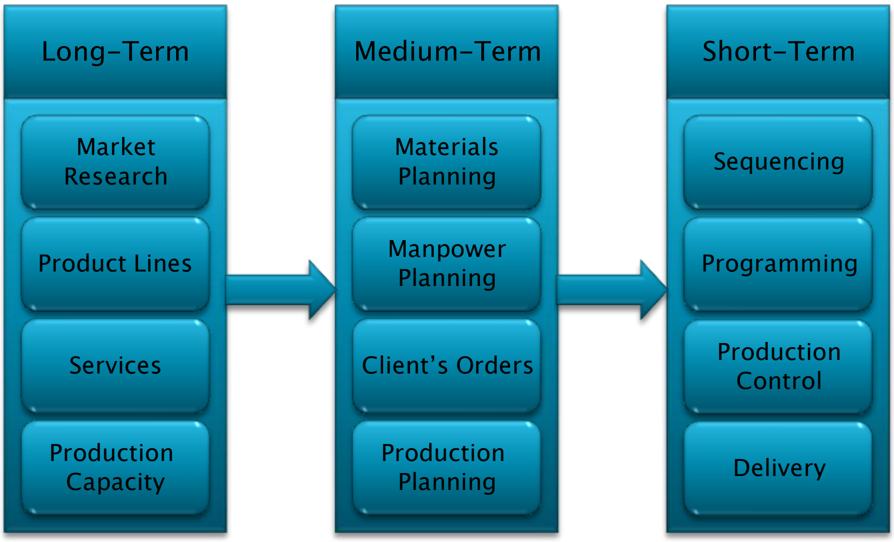


Figure 1.1: Division of decisions in terms of time horizon. Constructed from Ruiz (2003).

Scheduling can be defined as the assignment of start and finish times to events. In the case of production scheduling, these events are processing tasks. The constraints are mainly determined by the availability of resources, which depends on the production environment. Scheduling has become more and more important in the last decades, due to the simultaneous increase in the number of products, reduction in the size of orders and shortening delivery times, as stated by Botta-Genoulaz (1997). Proth (2007) agrees on this and describes how scheduling evolved from manual to automated and from static to dynamic. The necessity of new strategies such as customisation of products because of the more international and more competitive markets, causes the organisation of the production to be more and more complex. This is where scheduling becomes highly important. Optimisation of the production schedule allows for increased production capacity or client satisfaction without machinery investments, since the set of machines are typically assumed to be fixed. The recent paper on parallel machines by Li and Yang (2009) is just an example. Other resources are usually neglected or assumed unlimited. The work by Chen (2004) or Ruiz and Andrés (2007) are examples of exceptions to this assumption.

1.1. Motivation for this Ph.D. thesis

Since the first studies on scheduling by Salveson (1952) and others, a rich body of literature has been built including a wide range of problems with various characteristics. Nevertheless, many researchers (Ledbetter and Cox, 1977; Ford et al., 1987; McKay et al., 1988; Olhager and Rapp, 1995) have noted in their papers that there has always been a so-called gap between the theory and practice of scheduling. Dudek et al. (1992); MacCarthy and Liu (1993), and also McKay et al. (2002) have criticised scheduling research in general and the flowshop scheduling literature in particular regarding this gap. Similar conclusions can be found in literature reviews (see Graves, 1981; Allahverdi et al., 1999; Linn and Zhang, 1999; Vignier et al., 1999; Ruiz and Vázquez Rodríguez, 2010; Ribas et al., 2010) on different scheduling environments. Reisman et al. (1997) conducted a statistical review on flowshop sequencing/scheduling research between years 1952-1994. They discuss the exponentially growing body of literature on this subject and conclude that from a total of 170 reviewed papers, only 5 (i.e., 3%) dealt with true applications. Thirty-four papers or 20% dealt with "applications" that were not grounded in real-world settings.

The paper by Schutten (1998) tries to fill the gap between the operations research literature, in which high level algorithms are developed but side constraints that occur in practice are not considered, and the production literature in which myopic algorithms such as priority rules are used to solve practical problems. The paper illustrates how the shifting bottleneck procedure for the classical job shop can be extended to deal with practical features such as transportation times, setups, downtimes, multiple resources and convergent job routings.
Allaoui and Artiba (2004) also conjecture that there is a large gap between the literature of scheduling and the real life industry. The paper deals with a practical and stochastic hybrid flow shop scheduling problem under maintenance

constraints to optimise several objectives based on flow time and due dates. Also, setup, cleaning and transportation times are taken into account. The paper aims to show how to integrate simulation and optimisation to tackle this practical problem, and to illustrate by an experimentation study that the performance of heuristics applied to this problem can be affected by the percentage of the breakdown times.

Another paper involving realistic considerations is provided by Low (2005) who considers a flowshop with multiple unrelated machines. Some practical processing restrictions such as independent setup and dependent removal times are taken into account, and the objective is to minimise the total flow time in the system. A simulated annealing (SA)-based metaheuristic is proposed to solve the addressed problem. The initial solution is generated by priority rules like SPT and LPT. Multiple insertion technique is used when scheduling from a given job sequence at a stage, and two different strategies are tried to obtain the priority list for stages other than the first one; first come first serve (FCFS) and FIX, which uses the same job listing as in the first stage for all the other stages.

The time needed to adapt a machine after processing a certain task to its next task, is usually referred to as a setup time. The existence of setup times is a common phenomenon, both in industry and in the literature. Botta-Genoulaz (2000) propose several heuristics for a flowshop with parallel identical machines in each stage, positive time lags between the stages and precedence constraints between jobs as well as sequence-independent setup and removal times. Scheduling problems are harder if the setup times are sequence dependent, i.e. if the duration of the setup depends both on the previous and on the next job. According to Allahverdi et al. (2008), the amount of papers that take sequence dependent setup times into account is growing rapidly with an average of more than 40 papers in the last ten years.

In some industrial flowshop environments, once a job has been started at the first stage processing at the other stages cannot be delayed. This restriction is known as no-wait and can be found for example in the steel industry, where the material is not allowed to cool down between stages as this can cause defects in the composition of the steel. This problem, in combination with setup times, is studied by Ruiz and Allahverdi (2007b) for the total completion time.

The consumption of (half) products in order to complete other jobs demands precedence relationships between the jobs, since a job cannot be started before the job that it consumes is completed. In general, a job cannot be started in the first machine before all predecessors are finished at the last machine. Gladky et al. (2004) consider the less restricted case in which a job can be started at a certain stage when all predecessors have been processed at that stage. The precedence relationships are given per machine in the presented paper.
Another realistic situation is modelled by Naderi and Ruiz (2010), namely the existence of multiple flowshops for the production of a given set of jobs. In this problem, denoted as a distributed permutation flowshop, each job is assigned to one of the identical factories, and for each factory a processing order for the assigned jobs is to be established. This occurs in basically all sectors where companies possess more than one production plant or multiple production lines in one plant.

Although there is a recent trend towards more realistic formulations of scheduling problems such as the ones reviewed above, there are still not many research efforts to jointly consider realistic constraints prevailing in real-world manufacturing environments. One important drawback is that the solutions of such complex problems are rather difficult to obtain. Indeed, heuristic and metaheuristic solution approaches are needed to obtain good solutions in reasonable computational times. Yet, a wealth of such solution approaches may be developed with different degrees of "blindness" to problem specific knowledge representing interesting tradeoffs.

1.2. Classification of scheduling problems

Graham et al. (1979) introduced a three-field notation, in order to define different production scheduling problems in a schematic way. The first field is used to denote the machine settings, the second field contains the restrictions on the problem and the third field represents the objective function. The most basic machine setting, in fact trivial for the maximum completion time objective, is the one-machine problem, where a set N of n jobs have to be processed by one machine. This setting is denoted with a "1" in the first field. The problem is to find an optimal processing order for the jobs on this machine. Reeves (1995a) developed several heuristics for the total completion time problem with unequal job release dates; a problem that is proven to be $\mathcal{NP}$-Hard by Rinnooy Kan (1976).

The most studied setting at the moment, is the flowshop problem, represented by the letter F: n jobs visit a set M of m machines and each job visits all machines in the same order. Although the permutation of jobs can be different for each machine in the most general version of the problem, most research is done for the permutation flowshop problem. This restricts the permutation of jobs to be equal for all machines, which reduces the complexity of the problem, as described in Rad et al. (2009). Johnson (1954) was the first to consider the flowshop problem. Gupta and Stafford (2006) give a short overview of research done on the regular flowshop problem. Several reviews are available, comparing heuristics for the permutation flowshop problem (Ruiz and Maroto, 2005; Hejazi and Saghafian, 2005; Framinan et al., 2004).

Another possible machine setting is the production environment with parallel machines. In this case, each of the n jobs visits only one of the m machines, that are arranged in parallel. Considering processing speeds, the machines can either be identical, uniform or unrelated, denoted respectively by P, Q and R. The hybrid flowshop is the combination of the last two configurations: a set of n jobs visits the set of stages $M = \{1, \ldots, m\}$, all in the same order. Within each stage i, $1 \leq i \leq m$, a set of parallel machines $M_i = \{1, \ldots, m_i\}$ is present. Each job is processed by one of the m_i available machines at this stage. The first field for this machine setting is HF. The flow of the jobs through the hybrid shop is shown schematically in Figure 1.2. In 1999, the two first reviews on hybrid flexible flowline problems appeared, one by Vignier et al. (1999) and one by Linn and Zhang (1999). The latter conclude that there exists a gap between theory and practice and that there is need of future research in this direction. Quadt and Kuhn (2007) categorise the papers on hybrid flowshop problems in a taxonomy. The most recent reviews are the ones by Ruiz and Vázquez Rodríguez (2010) and Ribas et al. (2010).

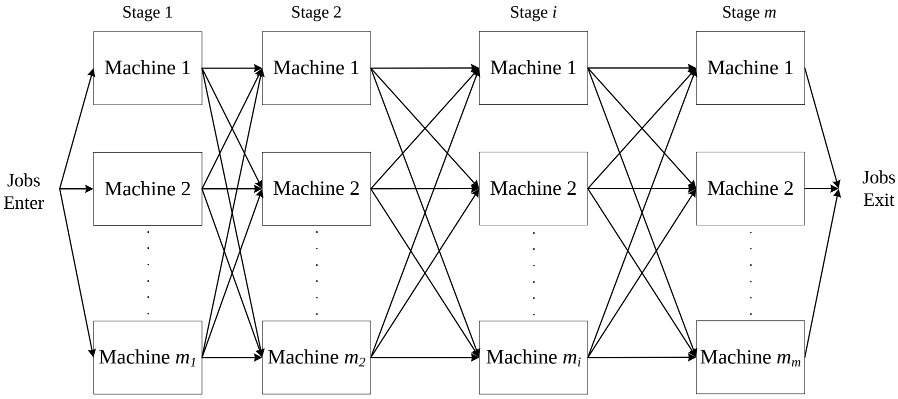


Figure 1.2: Hybrid flowshop environment. Source: Ruiz (2003).

In jobshop problems, the machines of the set M are neither parallel, nor structured in stages. The set of operations $N = \{1, \ldots, n\}$ is interrelated by a set of precedence constraints A, determining the order for machine visiting. An important heuristic contribution for the jobshop is done by Adams et al. (1988), who determine the bottleneck in each iteration of their constructive heuristic, and locally reoptimise considering this bottleneck.

1.3. Objectives

In this Ph.D. thesis, the main objective is the close study of a scheduling problem that is generally applicable to real-world situations. Doing so, we aim to diminish the gap between the necessities of the industries and their planning problems on the one hand, and the scheduling literature with its mainly theoretical advances on the other hand. In order to reach this goal, we need to develop advances and effective methods, capable of finding good solutions for problem instances of a realistic size, within reasonable computing time. In a nutshell, the algorithms should be able to find solutions that professional schedulers cannot easily improve manually, within a time span that allows the scheduler to try different scenarios and "play" with the parameters and settings. It can be very interesting, for example, to schedule a certain production plan and see how the objective values change when an additional order is added, or when the quantity of some product is increased. Other useful scenarios that require running the scheduling algorithm again could be the effect of adding a machine in one of the stages or reducing the setup times by assigning more human resources. In order to compare all such different situation, a certain flexibility is required, that can only be achieved by short algorithm running times. In colloquial business language, we could say that the algorithm should give a result “within the time of going for a coffee”.

The rest of this thesis is structured as follows: Chapter 2 introduces in detail the hybrid flexible flowline problem that forms the basis for this research. We highlight a practical application and we give a review on the literature on realistic scheduling. In Chapter 3 we present a mathematical model for the problem. The model is used to solve small problem instances and to analyse the complexity of each of the problem restrictions. Chapter 4 opens the possibility to solve large problem instances as well, using heuristics. First, we list a number of machine assignment rules ranging from the straightforward first available machine rule to advanced look-ahead rules that make extensive use of the problem data. Then, we give various possible solution representations for this problem, together with the cardinality of the solution space of each representation. The chapter is concluded with a group of fast dispatching rules and an adaptation of the famous NEH heuristic. In Chapter 5, for each of the earlier given solution representations, a genetic algorithm is developed. Comparison of the algorithms gives an indication of the effectiveness of each of the representations. We show some more advanced and modern local search algorithms in Chapter 6. The algorithms use the philosophy of state-of-the-art algorithms for the regular flowshop problem, but have been especially designed for the hybrid flexible flowline that lies closer to problems faced in reality. Chapter 7 introduces a completely new and highly effective algorithm that makes use of the problem characteristics in a clever way and shifts from one solution representation to another. The performance of the metaheuristic is compared to the performance of the best algorithms presented in earlier chapters. In Chapter 8 we change our focus from the problem restrictions to the optimisation criterion. We present two algorithms that allow for the optimisation

of multiple objectives at the same time, working with Pareto frontiers. The conclusions of this thesis are given in the final chapter, Chapter 9.
The tables that are not strictly needed to understanding the concepts and the results presented in this Ph.D. thesis are given in the appendices, in order to give the reader the opportunity to consult all data without interrupting the main thread unnecessarily. Appendix A contains tables with means and interactions that correspond to the experiments described in the text and to the figures showing the results graphically. The analysis of variance tables related to those experiments are given in Appendix B. In Appendix C, the best found solution values are given for the hybrid flexible flowline benchmark instances. The instances themselves would occupy too much space when printed. They can therefore be downloaded from `http://soa.iti.es/problem-instances`.

CHAPTER 2

THE HYBRID FLEXIBLE FLOW LINE PROBLEM

As an introduction to this chapter that defines the problem for this present thesis, we describe the production process for ceramic tiles. The ceramic tile sector has a big economical influence on the Castellón region, in the north of the Valencian Community (Spain). Segura et al. (2004) interviewed 81 ceramic tile companies on their strategic functioning. The results allow for a division of the producers in three groups. The first group, formed by 23 enterprises, is characterised by a main focus on the diversification of their production. Group 2, containing 30 companies, is mainly concerned with the costs of their production. The remaining 18 companies do not recognise a significant difference among the earlier mentioned priorities.

Another survey by Vallada et al. (2005), counting with the cooperation of the same 81 companies, shed more light on present issues regarding production scheduling. The authors conclude that the machines can be grouped in production stages that are visited in the same order. In the small and in some medium-sized production companies, two stages can be distinguished, namely the pressing, drying and glazing machines in the first stage, and the firing, classifying and packaging machines in the second stage. In other medium-sized and large factories, the machines can be divided in three stages: the pressing, drying and glazing machines in stage 1; the kiln firing machines in stage 2; and the classifying and packaging machines in stage 3. Since most companies have more than one production line, each of the stages consists in a set of unrelated parallel machines. Moreover, between different batches of tiles time for machine adjustments is needed. The time the adjustments take depends on the processing sequence, since the molds are changed less often if batches of the same size are grouped. The resulting problem is a hard combinatorial optimisation problem, known as the hybrid flexible flow line problem with sequence dependent setup times. Since, according to the authors, not even the largest ceramic tile producers use optimisation methods that solve this problem in an adequate way, and since no software is available to do so, the development of methods that allow for flexible production scheduling is required.

For a better understanding of the production process in the ceramic tile sector, Figure 2.1 demonstrates schematically the different operations that have to be performed in order to get to the final product. Since some of the subsequent machines are directly connected by conveyor belts, when modelling the problem analytically, these can be joined into one stage. Some important additional properties have to be taken into account when one aims to make feasible schedules for a ceramic tile factory. Not only the processing time can differ between parallel machines within a stage, also the physical possibility to process a certain tile model depends on the machine. Some tiles of large size, for instance, can only be processed on special machines. Other tiles that have a specific kind of decoration require another type of machine. The consumption of auxiliary products in order to complete a more complex article, obliges the scheduler to adopt another restriction in his model. In order to fabricate a corner profile, for example, first two flat ceramic surfaces need to be made, which will then be processed together to form the final corner structure. All auxiliary products that will be consumed need to be completed before the processing of the complex structure can start. When creating auxiliary products, but also for some exceptional simplified final products, stages might be skipped. Although the processing at a next stage in theory usually starts when processing is finished at the previous stage, this is not the case when producing ceramic tiles. Since a job represents a large batch of small products, the first products of the batch can go to the next stage, while the last products are still being processed at the previous stage. After the kiln firing stage, however, the contrary may happen. Since the tiles have to cool down before entering in the classification stage, waiting times should be taken into account. Regarding the setup times, two possible situations have to be considered. Mostly, the setup can be performed as soon as the machine is empty. However, in some cases, setup can only be performed if the job is at the stage. To give an example, for a correct calibration of the kiln firing machine, some of the tiles need to be present in order to see if the result is as expected. Another scheduling property that is sometimes forgotten in theoretical models, is the fact that the machines are usually processing previous work at the moment of designing a schedule for new jobs. No jobs can be assigned to a machine until it finishes all jobs that belong to earlier production plans.

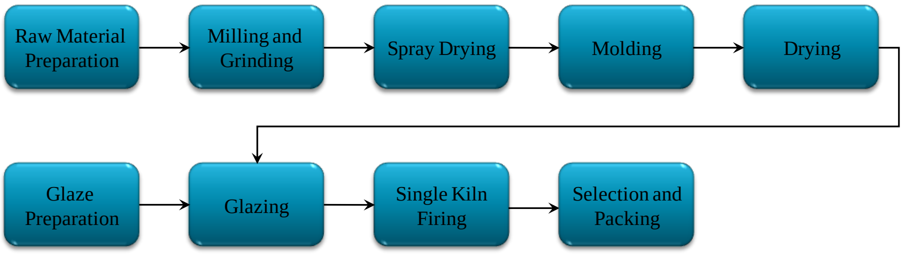


Figure 2.1: Graphical view of the steps in the ceramic tile production.

The production scheduling problem faced by the ceramic tile sector has served as an example and direct inspiration and motivation for the combinatorial problem that we consider in this Ph.D. thesis. In fact, the problem we treat, including the restriction, is identical to the one described for the production of ceramic tiles. As stated above, this problem is known as a hybrid flexible flow line (HFFL), where flexible means that each job $j \in N$ visits a subset $F_j \subseteq M$ of the stages and skips the remaining ones. This happens in most industries, as many products might be finished without adding certain options. Some examples are windshield rain sensors in car manufacturing, painting in furniture production or the glazing of ceramic tiles.

The processing time for job j on machine l at stage i is denoted by p_{ilj}. These times depend on the job and the machine, such that machines are unrelated, and are zero for all the machines at stages that the job does not visit (i.e., $p_{ilj} = 0, \forall i \notin F_j$).

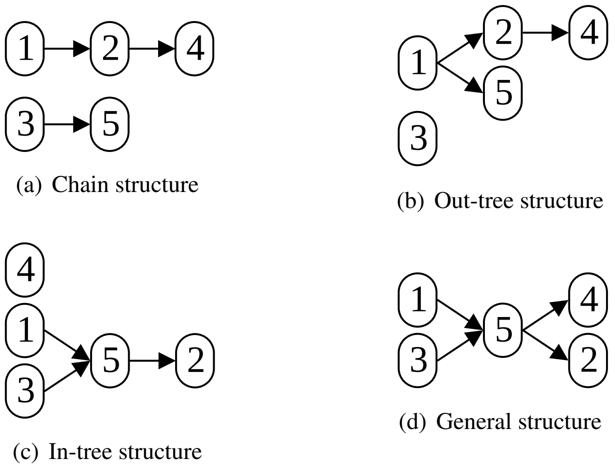
An example to demonstrate the need of unrelated machines in order to model a problem comes from the ceramic tile production. We compare two molding machines with different sizes. Machine A has a width of 40 inch, while machine B is of width 24 inch. If both process tiles of 12 inch, machine A is able to mold 3 tiles at a time and machine B is able to do 2. For this job, machine A is 50% faster than machine B. However, when processing tiles of 10 inch, machine A can process 4 tiles simultaneously, while machine B still handles 2 tiles. For this job machine A is 100% faster than machine B. This situation can only be modelled with unrelated parallel machines at each stage.

Furthermore, the following constraints are considered in this hybrid flexible flow line:

- $E_{ij} \subseteq M_i$ is the set of eligible machines for job j in stage i. This means that not all machines at a given stage might process a job j that visits such stage. Consider for example a stage with a small and a large machine. Small products can be processed on either of the two machines whereas large products can only be processed on the large one. Note that p_{ilj} is irrelevant if $l \notin E_{ij}$ and that necessarily $\mid E_{ij} \mid > 0$ if $i \in F_j$.

- rm_{il} expresses the release date for machine l in stage i. No operation can be started at machine l before rm_{il}. This allows us to model machines that did not finish the previous scheduled jobs yet.

- $P_j \subseteq N \setminus \{j\}$ gives a set of predecessors of job j. Job j cannot start until all jobs in P_j have finished. This is the case if auxiliary products are needed to start the processing of the final product. In Figure 2.2, different types of structures are shown for the precedence constraint graph. For the most simple type, each job has either zero or one predecessor and either zero or one successor, so that the relationships form chains. When jobs have only one predecessor but possibly various successors, is called an out-tree structure. In the opposite case, when a job can have various predecessors but only one successor, we speak about an in-tree. In this Ph.D. thesis the most general case is considered, where various predecessors and various successors are allowed. This is what best reflects the industrial situation, where on the one hand more than one auxiliary products can be necessary in order to finish a final product and on the other hand an auxiliary product can be used for the production of more than one final product.



(a) Chain structure

(b) Out-tree structure

(c) In-tree structure

(d) General structure

Figure 2.2: Different types of structures for the graph representing the precedence relationships.

- lag_{ilj} models the time lag for job j between stage i and the next stage to be visited, when job j is processed on machine l at stage i. A job in reality often consists of a large quantity of products with the same specifications, like a batch of ceramic tiles or a batch of bolts and nuts. If so, the first products can in many cases be processed at the next stage before finishing the whole job. In other cases, the start at a next stage might be delayed because of products that have to dry or cool down. Negative time lags model the former cases whereas positive time lags model the latter ones. In case of negative time lags, some conditions have to be fulfilled: $\mid lag_{ilj} \mid \leq p_{ilj}$ and $\mid lag_{ilj} \mid \leq p_{i+1,l',j} \; \forall l' \in E_{i+1,j}$, where $i + 1$ is the next visited stage by job j (not necessarily the next physical stage in the shop) and l' an eligible machine in that stage. The first condition avoids that the job to starts in the next stage before starting in the current stage; the second condition avoids that the job finishes in the next stage before finishing in the current stage. A graphical example of a valid negative time lag is given in Figure 2.3.

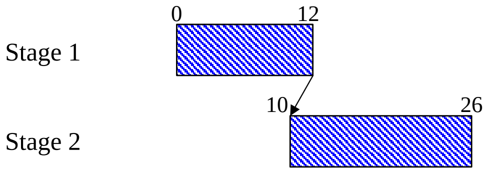


Figure 2.3: Graphical example of a negative time lag or overlap.

- S_{iljk} denotes the setup time between the processing of job j and job k on machine l inside stage i. Setup time is the time needed to reconfigure, clean or adjust a machine between two jobs. The setup time between painting a black product and a white one is usually larger than the time needed if the white product is processed before the black one, as remnants of black paint in the white paint are more evident than remnants of white paint in the black paint. We therefore treat sequence dependent setup times. These setup times are assumed separable from the processing time.

- A_{iljk} is a binary parameter that indicates whether the corresponding setup is anticipatory (one) or not (zero). Most machine setups can be performed before the product enters the stage, but in some cases (to attach the product to the machine, for example) setup has to be postponed until the product arrives at the machine. Figure 2.4 shows a graphical example of both.

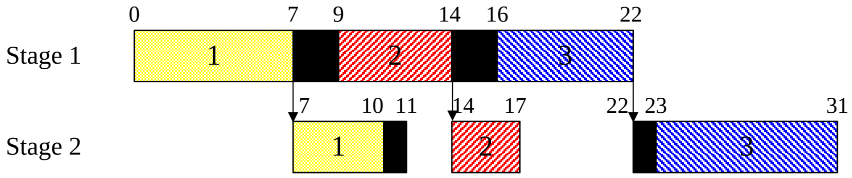


Figure 2.4: Graphical example of an anticipatory setup (between jobs 1 and 2 at stage 2) and a non anticipatory setup (between jobs 2 and 3 at stage 2).

For furniture manufacturing the same problem characteristics arise. Some of the products can only be manufactured on specialised machines. Production lines are seldom encountered empty, so machine availability from the start cannot be assumed. A drying time has to be taken into account after the painting stage. This can be modelled as a positive time lag. Attaching pieces of wood to the machines constitute non-anticipatory setups. Anticipatory setups occur as well, if the colour of paint has to be changed, for example.

The usual main objective in any company is to maximise either profit, or the shareholder value, depending on its legal structure. However, with the data we have it is not possible to deduce the influence of a certain production schedule on those financial quality measures. Moreover, the goal of a company as a whole is often not equal to the goal of some department. Just as the commercial department might have incentives related to the amount of sales rather than to the costs, the production department is asked to optimise the production rather than the financial value of the company. We therefore limit ourselves to the more tangible goals.

Among such goals we can find the minimisation of makespan, flowtime, setup time, lateness, tardiness, earliness, number of late jobs. More optimisation objectives can be defined, but these cover the most important ones. Makespan or maximum completion time is the moment in which the last task is finished. Flowtime measures the time that the jobs remain in the production plant, i.e., the difference between the completion time and the release date for a job. Total flowtime is directly related to work in progress. Minimisation of setup times is especially important if setup cost is extremely high. Lateness measures the difference between the moment of finishing a job and the moment that it is due to be finished. Lateness can be either positive or negative. Tardiness is closely related to lateness. It measures only the difference if a job is completed late; otherwise tardiness is zero. If late jobs are valueless, no matter how late they are, the number of late jobs is typically minimised. Especially when stocking costs are high, just in time management is important. For those cases earliness should also be minimised. Similar to tardiness, it measures only the difference between the moment of completing a job and its due date if the job is completed early. Otherwise earliness is zero. Earliness is usually minimised in combination with tardiness.

The goal that we aim to optimise for the presented HFFL is makespan or maximum completion time minimisation. It is the most generic objective, and it does not depend on the data on due dates. This data is highly important for lateness related goals, since the problem looses its interest if the due dates are too tight or too loose. Since choosing adequate due dates is difficult for a complex problem as the one we consider, we prefer to avoid the need of these data. Moreover, in practice the producer might have the possibility to negotiate the due date after scheduling the order. In this case the main interest is not to meet the due dates set by clients, but to have an efficient schedule and to adapt the due dates to it. Makespan is mainly production oriented, assuring efficiency by giving priority to compact schedules. For a more formal definition of makespan C_{ilj} is used to express the completion time of job j at stage i, where the job has been assigned to machine l. If we denote $LS_j = \max_{i \in F_j} i$ the last stage visited by job j, we can define the makespan as $C_{\max} = \max_{j \in N, l \in E_{LS_j,j}} C_{LS_j,l,j}$. Using the three field notation by Vignier et al. (1999) and using some extensions of our own, we can define this HFFL problem as:

$$HFFLm, ((RM^{(i)})_{i=1}^{(m)})/M_j, rm, prec, S_{iljk}, A_{iljk}, lag/C_{\max}$$

Although the number of feasible solutions is reduced by machine eligibility, stage skipping and precedence constraints, many simplifications of this problem have been proven to be $\mathcal{NP}$-Hard. Actually, the standard hybrid flow shop problem is just a special case of this HFFL problem. Lee and Vairaktarakis (1994) showed $\mathcal{NP}$-hardness of hybrid flow shop problems in general. That precedence relationships do not simplify the problem was concluded by Ullman (1975), who proved that the two parallel machine problem with precedence constraints is already $\mathcal{NP}$-Hard. The same holds for setup times, as Gupta (1986) classified the regular flow shop with sequence dependent setup times as $\mathcal{NP}$-Complete. From the previous discussion, and by reduction to simpler problems, the considered HFFL problem is obviously $\mathcal{NP}$.

2.1. Example instance

To illustrate the problem, we introduce example instance 1. This example describes an instance with two stages, where each stage contains three unrelated parallel machines. Five jobs have to be processed, and Job 4 is a predecessor of Job 1. Job 4 is not processed in the second stage and Job 5 skips the first stage. Release times for the machines in Stage 1 are 73, 125 and 98, respectively, and 113, 135 and 45 for the machines in Stage 2. The remaining data is given in Tables 2.1 and 2.2, where “-” means the machine is not able to process the job.

Stage	1			2		
Machine	1	2	3	4	5	6
Job						
1	-	16(0)	53(69)	-	-	38
2	-	-	11(98)	-	41	-
3	98(-3)	-	-	9	19	62
4	70(0)	-	-	-	-	-
5	-	-	-	-	97	80

Table 2.1: Example instance 1. Processing times of each job on each eligible machine. In brackets the time lag (if applicable).

Job	1	2	3	4	5	1	2	3	4	5
	Machine 1					Machine 4				
1	-	-	-	-		-	-	-	-	-
2	-	-	-	-		-	-	-	-	-
3	-	-	-	102(1)		-	-	-	-	-
4	-	-	119(0)	-		-	-	-	-	-
5	-	-	-	-		-	-	-	-	-
	Machine 2					Machine 5				
1	-	-	-	-		-	-	-	-	-
2	-	-	-	-		-	-	117(0)	-	104(0)
3	-	-	-	-		-	122(0)	-	-	114(0)
4	-	-	-	-		-	-	-	-	-
5	-	-	-	-		-	110(0)	106(0)	-	-
	Machine 3					Machine 6				
1	-	115(0)	-	-		-	-	124(1)	-	107(0)
2	113(1)	-	-	-		-	-	-	-	-
3	-	-	-	-		114(0)	-	-	-	119(1)
4	-	-	-	-		-	-	-	-	-
5	-	-	-	-		83(0)	-	88(0)	-	-

Table 2.2: Example instance 1. Setup times between pairs of jobs at each machine. A “1” in brackets indicates that setup times are anticipatory, a “0” that they are not.

The optimal makespan value for example instance 1 is 366. In Figure 2.5 one of the optimal solutions is shown in a Gantt chart.

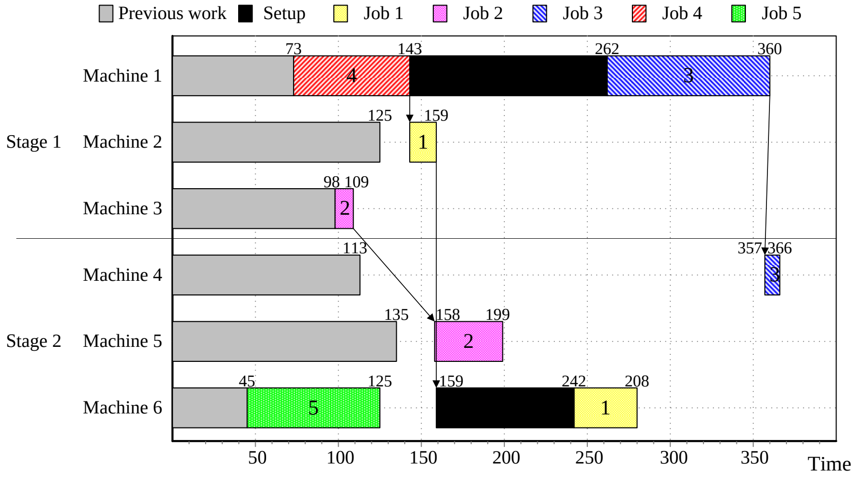


Figure 2.5: Gantt diagram with an optimal solution for example instance 1.

2.2. Literature review

The review of the relevant literature in this section is organised in the following way: In Subsection 2.2.1, we describe applications of genetic algorithms for scheduling problems that are closely related to real-world production situations. Subsection 2.2.2 is more focused, in the sense that only papers containing genetic algorithms for hybrid flexible flow line problems are cited. Finally, an overview of different solution representations for scheduling problems in the literature, is given in Subsection 2.2.3.

2.2.1. Genetic algorithm applications in realistic scheduling

Genetic algorithms (GAs) are a popular tool used for solving a range of optimisation problems including realistic scheduling problems. Oduguwa et al. (2005) provide a survey on evolutionary computation applications to real-world problems in metal forming industry, paper industry and chemical industry, and in scheduling and process planning, engineering design optimisation and related manufacturing applications. The survey is on the applications of the core methodologies of evolutionary computation which are listed as the genetic algorithms, evolutionary programming, evolution strategies and genetic programming. The results show that the majority of papers reviewed employ variants of GAs such as simple GAs, micro GAs, multiple-objective GAs and GAs with advanced operators.

Ruiz and Maroto (2006) propose the adaptation of a genetic algorithm metaheuristic, which performed well in regular flowshops in an earlier study presented in Ruiz et al. (2006), to a much more realistic version of the problem with sequence dependent setup times, unrelated parallel machines at each production stage, and machine eligibility. Such a problem is common in the production of textiles and ceramic tiles. The proposed algorithm incorporates four new crossover operators which identify and maintain building blocks in the form of similar job occurrences in both parents. To avoid premature convergence in the population, the researchers have implemented two methods, named restart and generational schemes. Whenever the lowest makespan in the population does not change for more than G_r generations, the restart procedure replaces 80% of the worst individuals of the population with both new good chromosomes and new random genetic material. The proposed generational scheme does not allow for clones in the population, i.e., a new individual will only replace the worst individual in the population if its makespan is better than that of the worst and if its sequence is not already in the population. Parameters and operators of the GA are determined using an extensive calibration by means of experimental designs. The proposed algorithm is tested against several adaptations of other well-known and recent metaheuristics to the problem using several experiments with a set of 1320 random instances as well as with real data taken from companies of the ceramic tile manufacturing sector. A statistical analysis shows that the proposed algorithm is between 53% and 135% more effective than the second best method, the genetic algorithm by Reeves (1995b).

An industrial application is given by Bertel and Billaut (2004) on a three-stage hybrid flowshop scheduling problem with recirculation. The problem is to perform jobs between a release date and a due date, in order to minimise the weighted number of tardy jobs. An integer linear programming formulation of the problem and a lower bound are proposed. A greedy algorithm and a genetic algorithm are presented as approximate methods and evaluated on instances like industrial ones. The representation in the GA is such that each position in the chromosome corresponds to one operation to schedule, and it contains a job number. Each job appears recurrently, and the number of recurrences is equal to the number of operations. Cyclic crossover (Bierwirth, 1995) and swap mutation are used together with a so-called truncated selection scheme in which the number of identical chromosomes allowed in the population increases with increasing iteration number.

In another application, Tanev et al. (2004) hybridise priority/dispatching rules and GAs by incorporating several such rules in the chromosome representation of a GA designed to solve the problem of scheduling the customers' orders in factories of plastic injection machines (FPIM). The problem is a multiobjective, real-world, flexible job shop scheduling problem. The chromosomes are in the form of strings of priority rules like FIFO, SPT, LPT, order due time, and their variations for selecting the next order for the currently becoming free machine. Performance evaluations are conducted for evolving a schedule of 400 customers' orders on an experimental model of FPIM.

Lohl et al. (1998) present an application of a genetic algorithm to a real-world scheduling problem in polymer industry. The problem is highly constrained. The quality of the results and the numerical performance is discussed in comparison with a mathematical programming algorithm. When designing the chromosome representation, the polymerisation stage is regarded as the stage where the crucial decisions are made. A linear genome type with the batches as genes is chosen. The length of the genome is determined by the maximum number of polymerisation which can be scheduled. The actually scheduled batches are determined by the schedule builder.

Dorn et al. (1996) describe an experimental comparison of four iterative improvement techniques for schedule optimisation including iterative deepening, random search, tabu search and genetic algorithms. They apply these techniques on the data of a steel making plant in Austria. To cope with the contradictory and over-constrained problem, the researchers have developed a model to describe the gradual satisfaction of given constraints considered explicitly.

Gilkinson et al. (1995) present a GA application to solve the real-world scheduling problem of a company that produces laminated paper and foil products. The manufacturing system is composed of workcell groups (stages with one or more parallel machines). Jobs may skip some stages. For certain products, it is possible to process multiple jobs on a single machine. The objective is a weighted combination of three objectives: minimisation of the number of late jobs, unbalanced machines and work in process time.

Ruiz and Allahverdi (2007a) address a permutation flowshop with no-wait condition. This means that no buffers exist between the subsequent machines, i.e., once a job is finished at one machine, processing should directly start at the next machine. The optimisation criterion is maximum lateness, which means that the largest difference between completion time and due date is minimised. The authors present a dominance rule for the three-machine case, as well as several heuristics. Four variants of a genetic algorithms are implemented and compared with the results of two state-of-the-art metaheuristics. The distinct GA variants either use an elitism approach or a steady-state structure and in each case appliance of local search can be done or not.

Vallada and Ruiz (2009) present different versions of a genetic algorithm for the unrelated parallel machine problem with sequence dependent and machine dependent setup times. They minimise makespan for the given problem. The difference between the algorithm versions is the use of a local search technique in the crossover operator and the use of a separate local search operator.

2.2.2. Genetic algorithms for hybrid flowshop problems

GAs are also popular tools to apply to the hybrid flowshop problems. Leon and Ramamoorthy (1997) explore problem-space-based neighbourhoods for industrial and randomly generated problems in the context of hybrid flowshop scheduling. The search is conducted in neighbourhoods generated by perturbing the problem data and not solutions; hence the name. The performance measures are the makespan and the mean tardiness. Three simple local search heuristics are proposed.

Lee et al. (1997) compare a GA to tabu search, simulated annealing and pair-wise exchange improvement for the lot sizing and scheduling in hybrid flowshops with variable lot sizes. The computational results show the superiority of the GA for these problems.

Jin et al. (2002) model a real-world application of a printed circuit board manufacturing system as a three-stage hybrid flowshop problem. The objective they aim to optimise is the makespan value. They present three subproblem approaches: a flowshop simplification and two parallel machine models. One parallel machine method uses both ready and tail times, while the other employs only ready times. A heuristic is applied to each of the subproblems. Furthermore, a compound approach is presented in the form of a genetic algorithm. The genetic algorithm is initially seeded with the solutions given by the subproblem approaches. The genetic algorithm improves those good initial solutions with 16%.

Kurz and Askin (2003, 2004) examine scheduling in hybrid flowshops with sequence-dependent setup times to minimise makespan. This type of manufacturing environment is found in industries such as printed circuit board and automobile manufacture. An integer program is formulated and discussed. Because of the difficulty in solving the integer program directly, several heuristics are developed, including a random keys genetic algorithm which is found to be very effective for the problems examined.

Sivrikaya Şerifoğlu and Ulusoy (2004) present a GA for makespan minimisation in hybrid flowshops. They apply roulette selection, exchange mutation and uniform order-based crossover. The initial population is seeded with three heuristic rules: shortest processing time (SPT), longest processing time (LPT) and shortest total processing time (STPT). The results are compared to a lower bound and to the results of heuristic rules. As the optima are unknown for the larger instances, a statistical method is used to estimate the optimal solution values.

Oguz and Ercan (2005) present a new crossover operator (NXO) for genetic algorithms in list scheduling. They compare it to the partially matched crossover (PMX). Furthermore, swap mutation and insertion mutation are evaluated. The best combination turns out to be an algorithm using the new crossover operator and insertion mutation. The genetic algorithm is shown to outperform a tabu search algorithm, implemented in order to solve the same problem.

Torabi et al. (2006) investigate the lot and delivery scheduling problem in a simple supply chain where a single supplier produces multiple components on a hybrid flowshop and delivers them directly to an assembly facility. The objective is to minimise the average of holding, setup, and transportation costs per unit time. They develop a mixed integer nonlinear program, an optimal enumeration method to solve the problem, and a hybrid genetic algorithm which incorporates a neighbourhood search into a basic genetic algorithm that enables the algorithm to perform genetic search over the subspace of local optima.

More recently, Jenabi et al. (2007) apply a genetic algorithm with a local improvement procedure to the economic lot sizing and scheduling problem in hybrid flowshops. The results are compared to those of a simulated annealing approach. The GA outperforms the SA in solution quality, but requires more computation time.

Simulation can be used when stochastic hybrid flowshop problems are concerned, as in Yang et al. (2007). In this paper a genetic algorithm is presented for a multi-layer ceramic capacitor application. The genetic algorithm outperforms the dispatching rules it is compared to with 33% to 61%.

Jungwattanakit et al. (2008) consider a hybrid flowshop problem with unrelated machines in each stage. Moreover, sequence dependent setup times are taken into account. The objective is to minimise a linear combination of two criteria: makespan on the one hand, and the number of tardy jobs on the other hand. The authors formulate a mixed integer program and implement a number of constructive heuristics and several dispatching rules for the problem. Finally a genetic algorithm is presented. In Jungwattanakit et al. (2009), the authors compare the genetic algorithm to a tabu search and a simulated annealing approach. They conclude that simulated annealing leads to the best results among all methods.

Tavakkoli-Moghaddam et al. (2009) tackle a slightly different problem with processor blocking. In the hybrid flowshop they treat, a machine is occupied by a job as soon as processing of the job starts, and only becomes available again when processing of the job starts at the next stage. Different from the common scheduling problems, the machine does not necessarily become available when processing of the job ends. The authors present a combination of a genetic algorithm with a nested variable neighbourhood search, referred to as a memetic algorithm. A series of experiments shows that the memetic algorithm performs better than a classic genetic algorithm, without local search.

Approaches different from GAs are also used, see for example the tabu search by Nowicki and Smutnicki (1998), in this case for simpler problems. Kochhar et al. (1988) provide a local search approach for a realistic hybrid flowshop problem with buffer capacities, blocking starvation, breakdowns and downtimes as well as setup times. Jin et al. (2006) propose some new lower bounds and implement a simulated annealing and a variable-depth search algorithm for the hybrid flowshop. Sequence dependent setup times are added to the problem by Naderi et al. (2010), who tackle a flexible hybrid flowshop scheduling problem with makespan objective. The authors point out the excessive simplicity of the regular flowshop and present two algorithms for this more realistic flowshop. The algorithms, a modified dynamic dispatching rule heuristic and an iterated local search metaheuristic, outperform seven algorithms from the literature.

What we can conclude from this subsection is that genetic algorithm applications for hybrid flowshop problems are not as common in the literature, as similar applications for simpler problems like the regular flowshop problem or the parallel machines problem. It is harder to obtain good results for a hybrid flowshop problem than for less complex problems. In our opinion, however, this is no reason to neglect this machine setting, especially since it is at least as frequent in industry as the other mentioned settings.

2.2.3. Representation schemes for GA applications in scheduling

The choice of a representation scheme is an important decision in the design of a GA which affects other design choices like the crossover and mutation operators, and eventually the performance of the algorithm. In fact, an inappropriate representation may lead to the failure of the GA itself. The representation schemes used in the GA approaches to scheduling problems are various. Simple permutations of tasks (jobs, operations) are most popular.

One of the first publications based on this idea, though used in combination with a tabu search algorithm, was the paper by Voss (1993). Chromosomes representing priority values (Dhodhi et al., 2002), execution times (Nossal, 1998), and machine assignments (Woo et al., 1997) for tasks are also used. A compound representation is provided by França et al. (2005) who consider the problem of scheduling part families and jobs within each part family in a flowshop manufacturing cell with sequence dependent family setup times to minimise the makespan. A genetic algorithm and a memetic algorithm with local search are proposed. The chromosome is a concatenation of $K + 1$ strings where K is the number of part families. The first string gives the order in which the families are scheduled on different machines. The rest of the strings each give the order in which the jobs of family f are processed for $f = 1, \ldots, K$. A variant of order crossover and two swap mutation operators are used. The population structure consists of several clusters, each one having a leader solution and three supporter solutions.

Some other approaches to scheduling problems use multiple-array chromosomes representing more than one dimension of the problem such as the ones presented by Ghedjati (1999) and Gonçalves et al. (2005). Both these last papers address problems involving precedence constraints.

The design decisions become more important for applications where the problem involves precedence constraints. Usually, topological ordering of tasks is used in the chromosomes. Ramachandra and Elmaghraby (2006) try to minimise the weighted sum of the completion times of a set of precedence-related jobs on two parallel identical machines. They test the results obtained by a GA approach against that obtained by a binary integer programming model. The chromosome representation is based on topological orderings of jobs, and schedules are obtained by using the first available machine rule for machine assignments. The initial population is seeded with heuristically obtained permutations. The researchers use one point order crossover which respects precedence constraints. A controlled swap mutation operator swaps two nodes that are interchangeable with respect to the precedence constraints. Kwok and Ahmad (1997) try to schedule arbitrary task graphs onto multiprocessors, where the task graphs represent parallel programs. They also use topological ordering type of representation in their genetic algorithm. The nodes of the graph are topologically ordered in the chromosome, and they are assigned to the processors to minimise the overall execution time of the program. Again, single point order crossover and controlled swap mutation are employed.

Ge (1999) addresses a similar problem, namely multiprocessor scheduling of graphs representing data-flow programs. The researcher employs a systematic approach to generate feasible permutations of nodes. The chromosome is a compound of sub-strings. Using the precedence diagram, the distance of each node z from source s is computed by taking only edge distance into account. Nodes with the same distance value are grouped in the same cluster. In the chromosome representation, nodes (jobs) within the same cluster are sequenced randomly and clusters are concatenated starting from the one with the smallest distance value.

Another compound type of representation scheme employed for problems involves priority listings for tasks. Cavory et al. (2004) consider the cyclic job shop scheduling problem with linear precedence constraints. The chromosome representation of the GA approach is a compound of distinct sub-chromosomes, each one related to a machine. Each sub-chromosome indicates a preference list, corresponding to an order of priority for the processing of the tasks on this machine. Crossover is partially-mapped crossover. Mutation is a simple swap operator that exchanges two alleles of a sub-chromosome.

Gonçalves et al. (2005) present a hybrid genetic algorithm for a job shop scheduling problem. The chromosome representation of the problem is based on random keys. It includes $2n$ genes where n is the number of operations. The first n genes give operation priorities. The second set includes factors to be used in the computation of delay times for the operations. Parameterized uniform crossovers are employed. As the mutation operator, one or more new members of the population are randomly generated from the same distribution as the original population.

Ghedjati (1999) also uses priority information in the chromosome structure, this time in a two-dimensional representation scheme. The paper addresses job-shop scheduling problems with several unrelated parallel machines and precedence constraints between the operations of the jobs (with either linear or non-linear process routings). A chromosome consists of two parts. The first part contains indices of priority rules to be used for operation assignment, the second part indices corresponding to one of the seven heuristics for machine assignment. One point crossover and swap mutation are used as the operators. Similarly, Wang et al. (1997) also use a chromosome structure consisting of two parts in their application to the matching and scheduling of interdependent subtasks of an application task in a heterogenous computing environment. The matching string represents the subtask-to-machine assignments, and the scheduling string gives the execution ordering of the subtasks assigned to the same machine.

Representation schemes other than task orderings and priority listings are also used although not as often. Nossal (1998), for example, presents a genetic algorithm for multiprocessor scheduling of dependent, periodic tasks. In this application, the scheduling problem is encoded by deriving execution intervals for the tasks, which determine the temporal boundaries for the execution points in time. The genetic algorithm selects the actual start time for each task from within the corresponding interval. The scheduler builds and then assesses the associated schedule with regard to the fulfillment of the deadlines of the tasks and the inter-task relations.

CHAPTER 3

MATHEMATICAL MODEL

3.1. Introduction

The classic solution to solve combinatorial problems, is to define a mathematical model in order to obtain optimal solutions. This is especially suited for small or relatively easy problems, since large and complex problems cannot be solved this way, due to time and memory limits. In this chapter we present a mixed integer programming (MIP) formulation for the hybrid flexible flow line problem introduced in Chapter 2. The MIP model is tested against a comprehensive benchmark and the results evaluated by advanced statistical tools that make use of decision trees. The results allow us to identify the constraints that increase the difficulty.

There are several well-known branch-and-bound approaches developed for the relatively easier problem of hybrid flow shop scheduling, for example Brah and Hunsucker (1991); Rajendran and Chaudhuri (1992); Santos et al. (1995). Although, to the best of our knowledge, there is not any branch-and-bound approach developed for the hybrid flexible flow line problem with the same or similar characteristics as considered here yet, some researchers provide MIP formulations for simpler problems. Sawik (2000) presents MIP formulations for scheduling of a flexible flow line with blocking. The machines are assumed to be identical. The basic MIP formulation is enhanced to model reentrant shops, where jobs visit a set of stages more than once, and to incorporate alternative processing routes for jobs. Kurz and Askin (2003) consider a hybrid flexible flow line environment with identical parallel machines and non-anticipatory sequence-dependent setup times. Their objective is to minimize the makespan. They provide a MIP formulation for the problem and propose some lower bounds. In the survey on exact methods for the hybrid flowshop, Kis and Pesch (2005) stress the progress of those methods, due to the development of new tight lower bounds. However, the addition of restrictions such as sequence dependent setup times make lower bounds such as the one recently proposed by Haouari and Hidri (2008) inapplicable for the case considered here.

3.2. The MIP model formulation

In the following, we provide a MIP formulation for the HFFL problem defined in Chapter 2. We first need some additional notation in order to simplify the exposition of the model:

- G_i is the set of jobs that visit stage i, ($G_i \subseteq N$ and $G_i = \{j|i \in F_j\}$),
- $G_{il} \subseteq G_i$ is the set of jobs that can be processed on machine l inside stage i, i.e., $G_{il} = \{j|i \in F_j \wedge l \in E_{ij}\}$,
- S_k gives the complete and unchained set of successors of job k, i.e., $S_k = \{j|k \in P_j\}$
- FS_k (LS_k) is the first (last) stage that job k visits.

The model involves the following decision variables:

$$
\begin{aligned}
X_{iljk} &= \begin{cases} 1, & \text{if job } j \text{ precedes job } k \text{ on machine } l \text{ at stage } i \\ 0, & \text{otherwise} \end{cases} \\
C_{ij} &= \text{Completion time of job } j \text{ at stage } i \\
C_{\max} &= \text{Maximum completion time}
\end{aligned}
$$

The objective function is:

$$\min C_{\max} \tag{3.1}$$

And the constraints are:

$$\sum_{\substack{j\in\{G_i,0\}\\ j\neq k, j\notin S_k}} \sum_{l\in E_{ij}\cap E_{ik}} X_{iljk} = 1, \qquad k\in N,\ i\in F_k \tag{3.2}$$

$$\sum_{\substack{j\in G_i\\ j\neq k, j\notin P_k}} \sum_{l\in E_{ij}\cap E_{ik}} X_{ilkj} \leq 1, \qquad k\in N,\ i\in F_k \tag{3.3}$$

$$\begin{gathered}\textstyle\sum_{\substack{h\in\{G_{il},0\}\\ h\neq k, h\neq j\\ h\notin S_j}} X_{ilhj} \geq X_{iljk}, \qquad j,k\in N,\ j\neq k,\ j\notin S_k,\\ i\in F_j\cap F_k,\ l\in E_{ij}\cap E_{ik}\end{gathered} \tag{3.4}$$

$$\textstyle\sum_{l\in E_{ij}\cap E_{ik}} (X_{iljk}+X_{ilkj}) \leq 1, \quad \begin{array}{l} j\in N, k=j+1,\ldots,n, j\neq k,\\ j\notin P_k, k\notin P_j, i\in F_j\cap F_k\end{array} \tag{3.5}$$

$$\sum_{k\in G_{il}} X_{il0k} \leq 1, \quad i\in M,\ l\in M_i \tag{3.6}$$

$$C_{i0} = 0, \qquad i\in M \tag{3.7}$$

$$\begin{gathered} C_{ik} + V(1-X_{iljk}) \geq \max\Big\{\max_{p\in P_k} C_{LS_p,p}, rm_{il}, C_{ij} + A_{iljk}\cdot S_{iljk}\Big\}\\ +(1-A_{iljk})\cdot S_{iljk} + p_{ilk},\\ k\in N,\ i=FS_k,\ l\in E_{ik},\ j\in\{G_{il},0\},\ j\neq k,\ j\notin S_k \end{gathered} \tag{3.8}$$

$$\begin{gathered} C_{ik} + V(1-X_{iljk}) \geq \max\Big\{C_{i-1,k}+\\ \sum_{\substack{h\in\{G_{i-1},0\}\\ h\neq k, h\notin S_k}} \sum_{l'\in E_{i-1,h}\cap E_{i-1,k}} \Big(lag_{i-1,l',k}\cdot X_{i-1,l',h,k}\Big),\\ rm_{il}, C_{ij} + A_{iljk}\cdot S_{iljk}\Big\} + (1-A_{iljk})\cdot S_{iljk} + p_{ilk},\\ k\in N, i\in\{F_k\setminus FS_k\}, l\in E_{ik}, j\in\{G_{il},0\}, j\neq k, j\notin S_k \end{gathered} \tag{3.9}$$

$$C_{\max} \geq C_{LS_j,j}, \qquad j\in N \tag{3.10}$$

$$\begin{gathered} X_{iljk} \in \{0,1\} \\ j \in \{N,0\}, k \in N, j \neq k, k \notin P_j, i \in F_j \cap F_k, l \in E_{ij} \cap E_{ik} \end{gathered} \tag{3.11}$$

$$C_{ij} \geq 0, \quad j \in N, \; i \in F_j \tag{3.12}$$

The set of constraints (3.2) assures that every job should be preceded by exactly one job on only one machine at each stage. Here only the possible variables are considered. Note that for every stage and machine we introduce a dummy job 0, which precedes the first job at each machine. This also allows for the consideration of initial setup times. Constraint set (3.3) is similar in the way that every job should have at most one successor. Constraint set (3.4) forces that if a job is processed on a given machine at a stage, then it should have a predecessor on the same machine. This is a way of forcing that assignments are consistent in the machines. Constraint set (3.5) avoids the occurrence of cross-precedences. Note again that only the possible alternatives are considered. With constraint set (3.6) we enforce that dummy job 0 can only be predecessor of at most one job on each machine at each stage. Constraint set (3.7) simply ensures that dummy job 0 is completed at time 0 in all stages. Constraint set (3.8) controls the completion time of jobs at the first stage they start processing by considering all eligible machines. The value V represents a big number so to make the constraint redundant if the assignment variable is zero. Notice that precedence relationships are considered by accounting for the completion of all the predecessors of a given job. Note also that both types of sequence dependent setup times (anticipatory and non-anticipatory) are also taken into account. Constraint set (3.9) gives the completion time on subsequent stages. Here the completion time of the same job in the previous stage along with the lag time is considered. Constraint set (3.10) defines the maximum completion time. Finally, (3.11) and (3.12) define just the decision variables.

3.3. Computational Evaluation

We define a complete set of instances to test the MIP model and to investigate the effect of realistic considerations on problem difficulty. Due to the complexity of the problem and the number of different characteristics considered, a total of 10 factors are combined at the levels given in Table 3.1 below.

Factor	**Symbol**	**Values**
Number of jobs	n	5, 7, 9, 11, 13, 15
Number of stages	m	2, 3
Number of unrelated parallel machines per stage	m_i	1, 3
Distribution of the release dates for the machines	rm_{il}	0, $U[1, 200]$
Probability for a job to skip a stage	PF_j	0%, 50%
Probability for a machine to be eligible	PE_{ij}	50%, 100%
Distribution of the setup times as a percentage of the processing times	DS_{iljk}	$U[25, 74]$, $U[75, 125]$
Probability for the setup time to be anticipatory	PA_{iljk}	$U[0, 50]\%$, $U[50, 100]\%$
Distribution of the lag times	$Dlag_{ilj}$	$U[1, 99]$, $U[-99, 99]$
Number of directly preceding jobs	NP_j	0, $U[1, 3]$

Table 3.1: Factors considered in the design of the initial test bed.

The number of directly preceding jobs needs some further explanation. It is the number of "direct" predecessors, i.e., predecessors that are directly connected in the predecessor graph, without intermediate job. If we consider job 1 to be a predecessor of job 2 and job a predecessor of job 3, then job 1 is an indirect predecessor of job 3. In an example instance of 15 jobs and a $U[1, 3]$ distribution for the number of directly preceding jobs, the total number of predecessor relationships (both direct and indirect) is 35. The highest number of predecessors for one job is six in the same instance.

The distribution of the processing times is fixed to $U[1, 99]$. The total number of combinations is $6 \cdot 2^9$ =3,072. There are three replicates per combination, so in total there are 9,216 instances. It is important to remark that when generating the instances all restrictions affecting the data (see Chapter 2 for details) were considered. For example, every job must visit at least one stage and at least one machine on every visited stage must be eligible (and thus the factors PF_j and PE_{ij} must be controlled). Additionally, special care must be given to the generation of the precedences among jobs. We will use this set of instances to test the MIP model. A subset of the instances is available at `http://soa.iti.es/problem-instances`.

3.3.1. MIP model evaluation

For every problem instance, a file containing the model in .LP-format is constructed and then solved with CPLEX 9.1 on a Pentium IV 3.2 GHz computer with 1 Gbyte of RAM memory. It could be argued that an ad-hoc branch and bound algorithm would perform better than the best regarded commercial solver available. However, we refrained from developing such a method mainly due to the fact that obtaining a tight lower bound for the HFFL problem considered is a very daunting task. As a matter of fact, considering only the sequence-dependent setup times already defeats most possible lower bounds since the amount of setups depends on the sequence. Using commercial solvers for flowshop problems with setups has been pursued in the literature. For example, Stafford and Tseng (2002) solved instances of up to 9 jobs and 9 machines for a $F/S_{ijk}/C_{\max}$ problem with LINDO commercial solver. The authors needed about 6,622 and 300 seconds CPU time in a Pentium III 800 Mhz computer for each one of the two models they proposed, respectively. According to the review and evaluation of heuristics for the same problem in Ruiz et al. (2005), most exact methods proposed for the $F/S_{ijk}/C_{\max}$ problem are very limited and the bounds proposed not tight. For all the above reasons, it seems plausible that an efficient solver using linear relaxations of variables as bounds would perform reasonably well.

Due to the large number of instances, we impose a time limit for every model of 300 seconds. For each model, we record a categorical variable called “type of outcome” with three possible values 0, 1 and 2. Outcome 0 means that an optimal solution was found, in which case we record the time needed and the optimal $C_{\max}$ value. Outcome 1 means that the 300 seconds time limit was reached and a feasible integer solution was found. In this case we record the solution found and the gap between this solution and the best MIP bound. Lastly, the outcome value 2 indicates that no feasible integer solution could be found within the time limit.

Table 3.2 shows the results for all the controlled factors in the case of $n = 7$, $m = 3$, $m_i = 3$ and $rm_{il} = U[1, 200]$. Each cell gives the average of the 3 replicates. In the table, the percentage of instances for which an optimal solution can be found within the time limit (%Opt) and the average time needed to reach this optimal solution (Av time) are displayed. The percentage of instances for which an integer feasible solution is found within the time limit (%Limit) is also displayed in the table. The percentage of instances for which no solution could be found (type of outcome 2) can be easily obtained by subtracting these two percentages from 100 (i.e., $100-$%Opt$-$%Limit) but there are none in this case.

			PA_{iljk}	$U[0,50]\%$				$U[50,100]\%$			
			$Dlag_{ilj}$	$U[1,99]$		$U[-99,99]$		$U[1,99]$		$U[-99,99]$	
PF_j	PE_{ij}	DS_{iljk}	NP_j	0	$U[1,3]$	0	$U[1,3]$	0	$U[1,3]$	0	$U[1,3]$
0%	50%	$U[25,74]$	%Opt	66.67	66.67	66.67	100	0	66.67	33.33	100
			Av time	48.64	33.56	169.77	71.79	0	117.49	27.47	83.72
			% Limit	33.33	33.33	33.33	0	100	33.33	66.67	0
		$U[75,125]$	%Opt	66.67	100	33.33	100	66.67	100	33.33	100
			Av time	69.32	28.01	41.36	5.06	80.17	45.37	57.58	60.7
			% Limit	33.33	0	66.67	0	33.33	0	66.67	0
	100%	$U[25,74]$	%Opt	0	0	0	0	0	0	0	0
			Av time	0	0	0	0	0	0	0	0
			% Limit	100	100	100	100	100	100	100	100
		$U[75,125]$	%Opt	0	0	0	0	0	0	0	0
			Av time	0	0	0	0	0	0	0	0
			% Limit	100	100	100	100	100	100	100	100
50%	50%	$U[25,74]$	%Opt	100	100	100	100	100	100	100	100
			Av time	2.03	0.06	9	0.2	0.04	0.04	0.74	0.31
			% Limit	0	0	0	0	0	0	0	0
		$U[75,125]$	%Opt	100	100	100	100	100	100	100	100
			Av time	0.07	0.04	0.21	0.5	0.23	0.08	0.13	0.16
			% Limit	0	0	0	0	0	0	0	0
	100%	$U[25,74]$	%Opt	100	33.33	100	100	100	66.67	66.67	100
			Av time	4.27	100.83	0.78	74.54	67.9	8	0.31	4.86
			% Limit	0	66.67	0	0	0	33.33	33.33	0
		$U[75,125]$	%Opt	100	100	66.67	66.67	100	100	100	100
			Av time	14.81	5.87	103.05	2.22	80.78	87.48	9.43	21.7
			% Limit	0	0	33.33	33.33	0	0	0	0

Table 3.2: MIP model results for $n = 7$, $m = 3$, $m_i = 3$ and $rm_{il} = U[1, 200]$ with a CPU time limit of 300 seconds.

From Table 3.2, it follows for the combination $n = 7$, $m = 3$, $m_i = 3$ and $rm_{il} = U[1, 200]$ that when all stages are visited ($PF_j = 0\%$) the models are

more difficult to solve. The same applies to the case when all machines inside a stage are eligible (PE_{ij}=100%). The combination $PF_j = 0\%$ and PE_{ij}=100%, i.e., every stage is visited and every machine is eligible, is especially difficult: In no instance an optimal solution could be found within the time limit, regardless of other parameter values. These results confirm what is expected, with more stages and more eligible machines, more feasible solutions and therefore more time is needed for obtaining the optimal solution. As regards the MIP model, the factors that affect the distribution of the data in the instance (rm_{il}, DS_{iljk}, PA_{iljk} and $Dlag_{ilj}$) do not seem to have a clear significant effect on the difficulty. The aggregated results for all the values of n, m, m_i and averaged over the other parameters are shown in Table 3.3. As it has been pointed out, the total number of variables and constraints depends on many factors and ultimately, on all the data in a given instance. We show the average number of variables and constraints for the MIP models in Table 3.3 as well.
It can be observed in Table 3.3 that the previous findings are confirmed: increasing n, m and m_i results in harder problems. However, there is an interesting result. Increasing the number of unrelated parallel machines m_i for the larger values of n (13 and 15) seems to have a positive impact, although small, on the percentage of instances with integer optimal solutions. For example, for $n = 15$, $m = 2$ and $m_i = 1$ we find that only 0.26% of the instances end up with optimal solutions but for $m_i = 3$ this percentage increases up to 8.85%. Initially this result might seem counter-intuitive since with more unrelated parallel machines per stage more variables are needed in the model. The explanation to this behaviour comes from the fact that n is, by far, the most influential factor. With $n = 15$ the number of variables is very large. Having more unrelated parallel machines at each stage means that the assignment of jobs to machines at each stage becomes more important. With only one machine per stage there is no assignment and solutions are solely influenced by the permutation of the jobs. In other words, more unrelated parallel machines per stage helps lessening the sheer effect of the number of jobs on the difficulty of the instances.

We can say that the overall performance of the proposed MIP model, given

	m	2		3	
n	m_i	1	3	1	3
5	%Opt	100.00	100.00	100.00	83.07
	Av Time	0.30	1.47	11.03	20.63
	%Limit	0.00	0.00	0.00	16.93
	Variables	34.22	65.49	48.29	93.21
	Constraints	121.68	225.27	172.15	322.00
7	%Opt	79.17	77.60	74.48	63.54
	Av Time	11.78	17.78	8.93	34.67
	%Limit	20.83	22.40	25.52	36.46
	Variables	61.21	126.46	83.79	177.46
	Constraints	227.64	473.41	313.89	660.57
9	%Opt	53.39	59.38	46.35	39.06
	Av Time	30.42	42.73	29.31	33.35
	%Limit	46.61	40.62	37.24	58.85
	Variables	109.70	213.54	153.79	294.41
	Constraints	430.13	832.98	598.00	1136.39
11	%Opt	32.29	29.95	22.92	24.22
	Av Time	56.30	31.26	49.36	53.68
	%Limit	50.00	69.01	51.04	62.24
	Variables	168.55	319.74	236.46	444.31
	Constraints	682.27	1274.33	946.11	1755.09
13	%Opt	8.07	17.45	5.73	12.76
	Av Time	63.94	48.88	113.05	53.07
	%Limit	67.45	71.87	65.63	65.10
	Variables	236.02	445.58	330.86	444.31
	Constraints	975.39	1802.58	1352.49	1755.09
15	%Opt	0.26	8.85	0.78	3.12
	Av Time	81.00	83.13	43.59	92.85
	%Limit	72.40	71.61	63.80	70.31
	Variables	315.40	598.59	441.51	840.52
	Constraints	1319.54	2426.40	1828.30	3386.19

Table 3.3: Aggregated MIP model results for a CPU time limit of 300 seconds.

its complexity and number of variables, is good. In Table 3.3 we have that the most complex case is given by $n = 15$, $m = 3$ and $m_i = 3$, and only 3.12% of the problems could be solved to optimality and in another 70.31% of the cases a feasible integer solution was obtained before the time limit was reached. Therefore in 26.57% of the problems no solution could be found. As shown, in this case the average number of variables and constraints is more than 840 and 3386 respectively. The average gap between the feasible integer solutions and the best bounds found by CPLEX is 71.48% which is deemed as large. However, we are only allowing for a total of 300 seconds of CPU time per instance, which, given the complexity of the problem to be solved, is quite short.

3.3.2. MIP model statistical analysis

Most valuable statistical tools suited for analysing the effect of the 10 considered factors on the performance of the MIP model are nullified by the fact that the response variable considered (type of outcome) is categorical. Under this circumstance, ANOVA technique, for example, cannot be applied. Non-parametric statistical tests like the well known Kuskal-Wallis or Wilcoxon signed-rank tests that can take categorical response variables are also not suitable. Since, with these tools, the choices are limited to mostly paired tests and with 10 factors and all the possible interactions not too much information could be obtained. Therefore, we propose the application of an advanced technique called Automatic Interaction Detection (AID).

AID recursively bisects experimental data according to one factor into mutually exclusive and exhaustive sets that describe the response variable in the best possible and statistically significant way. AID works on an interval scaled or purely categorical response variable and maximises the sum of squares between groups by means of a given statistic. The original AID technique was proposed by Morgan and Sonquist (1963). Kass (1980) developed an improved version called Chi-squared Automatic Interaction Detection (CHAID) by including statistical significance testing in the partition process and by allowing multi-way splits of the data. Later, Biggs et al. (1991) further improved CHAID method and created what is known as Exhaustive CHAID algorithm that does a more thorough job when examining all possible partitions for each factor. These techniques are of common use in the fields of Education, Population Studies, Market Research as well as many others. We use Exhaustive CHAID for analysing our experimental data. The method starts with all data classified into a first (root) node. Then, all factors are considered for splitting the node and the best multi-way split according to the levels of each factor is calculated. To this end, a statistical significance test is carried out so to rank the factors on how well they split the node. A Chi-squared (χ^2) test is used for categorical factors. After the node has been split, the same procedure is applied to all sub-nodes until no more significant partitions can be found or until a given stopping criterion is met. Usually, a classification or decision tree is obtained as a result of the application of the method. The resulting tree enables a careful study of the effect of the different factors, and what is more important, the interactions between them.

We use SPSS DecisionTree 3.0 software which implements Exhaustive CHAID algorithms. All 10 factors as well as the response variable are deemed as categorical (nominal). We choose a minimum number of cases (data) for each node before splitting of 192. Nodes with fewer cases are not split. Furthermore, if splitting a parent node results in a child node with less than 96 cases, the node will not be split. These values are chosen after a close examination of initial test trees and to avoid splits in the trees on the basis of small data samples. Furthermore, the values 192 and 96 ensure that we still have a large number of cases per parent and child nodes. We set a confidence level for splitting of 99.9% and a Bonferroni adjustment for multi-way splits that compensates the statistical bias in multi-way paired tests. The first three levels of the resulting tree are shown in Figure 3.1.

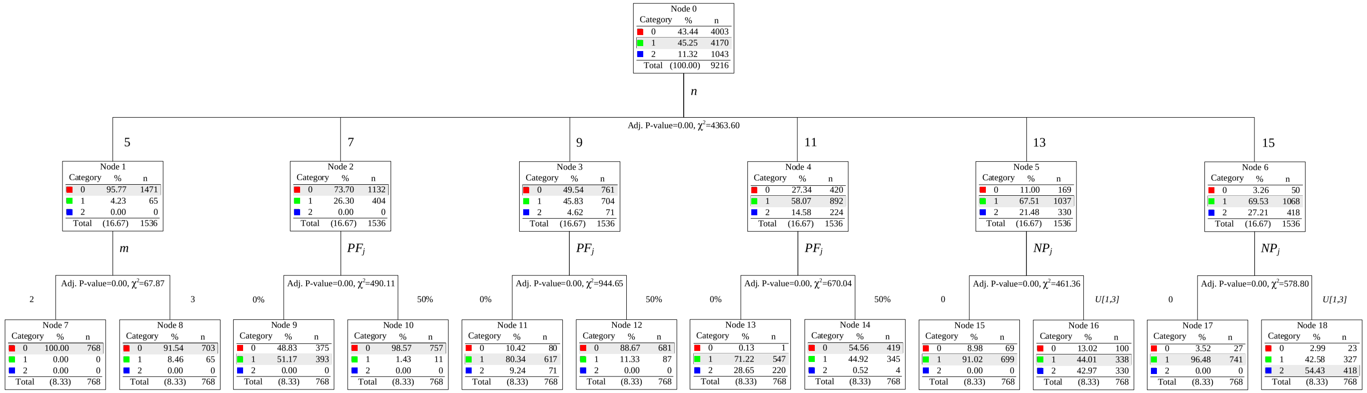


Figure 3.1: Decision tree with the first three levels shown in detail, time limit=300 seconds.

In Figure 3.1, the root node contains all data of the experiment and at that level, the most significant factor is the number of jobs or n. Therefore, the next level is composed of one node for every possible value of n. Moreover, this split is done with a very high level of confidence since the p-value is very close to 0 and the result of the χ^2 statistic is very high, i.e., n is the most influential factor on the response variable with a statistically significant effect. Within the resulting six nodes, as the value of n increases the number of cases for which no solution is found increases. In Node 6, where $n = 15$, few instances were solved to optimality.
After this first multi-way split, each node is split in two according to different factors. For $n = 5$, m is the most influential factor whereas for $5 \leq n \leq 11$ the factor PF_j (probability for a job j to skip a stage) is more important. As it has been mentioned, a 0% probability for a stage to be skipped results in more difficult instances for all the values of n. Surprisingly, for $n = 13$ and 15 the factor NP_j (Number of preceding jobs for job j) is the most discriminating. In the child nodes of nodes 5 and 6, there is an interesting observation. For the instances where there are no precedence relations (nodes 15 and 17 with $NP_j = 0$) either an optimal solution or an integer feasible solution is found within the time limit of 300 seconds. In nodes 16 and 18 with $NP_j = U[1, 3]$, about half of the instances remain unsolved. This outcome is again counter-intuitive and a careful analysis is needed. While adding predecessors results in fewer variables since a job cannot be scheduled before one of its predecessors, it greatly complicates some of the constraints of the model, more precisely, constraint set (3.8). This affects the branch and bound algorithm used by CPLEX and results in instances being more difficult to solve. For reasons of space the full tree cannot be shown in detail. Instead we have constructed a simplified tree shown in Figure 3.2.

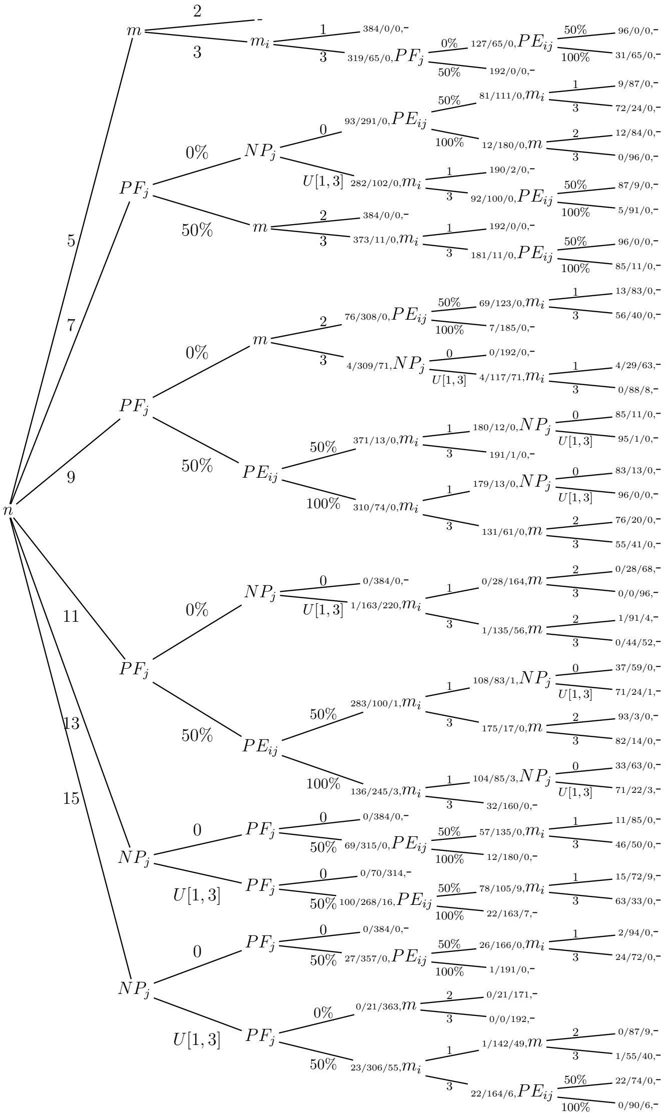


Figure 3.2: Full simplified decision tree, time limit=300 seconds.

In this tree we omit the values of the three types of outcome from the first three levels, since they can be seen in Figure 3.1. From the fourth level until the last significant level we show at the edges the factors according to which parent node is split into child nodes. At a given node we show the absolute values of the three types of outcome. Also shown is the factor that results in further child node division or "-" if no further statistically significant divisions are found or if the stopping criterion for branching is met.
As can be seen, apart from the already mentioned factors n, m and PF_j, there are other factors which determine differences on the three levels of the response variable. These are m_i, NP_j and PE_{ij} (Probability for a machine in stage i to be eligible for job j). It is interesting that all other factors which affect mainly the distributions of setup times, anticipatory setups, lags and release dates for machines do not appear to be significant. Although not shown here, extending the previous tree by allowing parent and children nodes to have any number of data results in very little variations. Therefore, the proposed MIP model does not seem to be affected by the factors rm_{il}, DS_{iljk}, PA_{iljk} or $Dlag_{ilj}$.

As it has been mentioned before, the average gap obtained for the type of outcome 1 is more than 70% which makes us think that allowing for more time would not change the results significantly. In order to test this hypothesis, we ran all the experiments once more with the only difference that the allowed CPU time was increased from 300 to 900 seconds. The aggregated results for all the values of n, m and m_i are shown in Table 3.4.
It can be observed that in all situations the percentage of instances with optimal solutions (%Opt) increases. However, this increase is rather small especially if we consider that the maximum allowed CPU time has tripled. For $n = 15$, $m = 3$ and $m_i = 3$ we see that the percentage of optimal solutions has increased from 3.12 to 5.21 and the average time from 92.85 to 261.60 seconds. The total CPU time necessary for solving all instances with 900 seconds stopping time has been 1,294 hours (almost 54 days). Allowing for more CPU time seems to have a small effect on the number of optimal solutions obtained. Carrying out the exhaustive CHAID analysis yields the tree depicted in Figure 3.3 (only the first three levels shown).

	m	2		3	
n	m_i	1	3	1	3
5	%Opt	100.00	100.00	100.00	90.36
	Av Time	0.32	2.06	10.47	73.14
	%Limit	0.00	0.00	0.00	9.64
7	%Opt	83.85	85.16	75.26	69.27
	Av Time	60.58	99.33	18.31	75.81
	%Limit	16.15	14.84	24.74	30.73
9	%Opt	60.16	65.36	48.44	41.41
	Av Time	124.30	89.95	51.38	65.79
	%Limit	39.84	34.64	38.54	58.33
11	%Opt	35.68	34.11	28.91	26.56
	Av Time	106.81	125.49	140.87	124.99
	%Limit	51.56	65.89	45.31	61.98
13	%Opt	14.06	20.31	8.85	16.93
	Av Time	254.17	146.95	230.03	209.46
	%Limit	61.98	73.44	63.54	61.46
15	%Opt	1.82	12.24	1.56	5.21
	Av Time	492.76	176.77	246.60	261.60
	%Limit	71.61	72.40	67.45	69.79

Table 3.4: Aggregated MIP model results for a CPU time limit of 900 seconds.

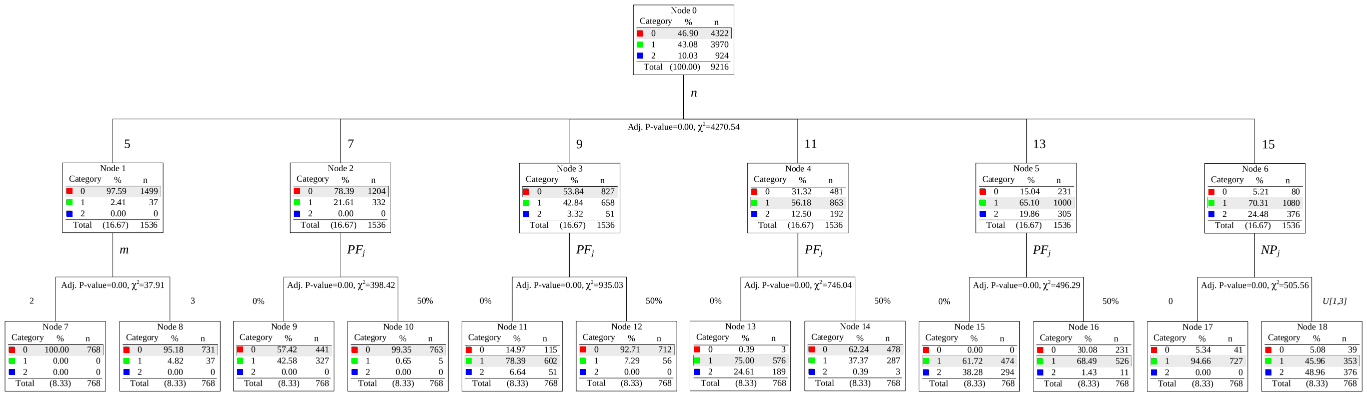


Figure 3.3: Decision tree with the first three levels shown in detail, time limit=900 seconds.

The two trees of Figures 3.1 and 3.3 have little differences as regards the most influential factors. As expected, the number of optimal solutions and integer solutions (types of outcome 1 and 2) increase in general in Figure 3.3, while the number of cases for which no solutions are found decreases. At the root node 46.90% of instances are solved to optimality (from the original 43.44% obtained with 300 seconds maximum CPU time), and the percentage of unsolved instances has decreased from 11.32 to 10.03. One interesting test that we can carry out is to see if this observed 3.46% of additional instances that are solved to optimality when 900 seconds of CPU time are allowed is statistically significant. Since the two sets of results represent dependent samples, we have carried out a McNemar (1947) test on paired proportions. The results of this test are sound. There is a statistically significant difference between the percentages of instances solved to optimality when increasing the CPU time with a χ^2 value of 313.08 and a p-value close to 0. The 95% confidence interval of the percentage increase on solutions solved to optimality is $[3.35, 3.50]\%$. Although there is a statistically significant difference, this is really small. Obtaining a maximum of 3.50% additional instances solved to optimality does not compensate the tripled CPU time. The full simplified tree for the later case in which 15 minutes CPU time are allowed is given in Figure 3.4.

As can be seen in Figure 3.4, there are fewer levels in the full simplified tree. This is also an expected outcome since the factors that had a weak effect in the case with 300 seconds maximum CPU time are nullified when more CPU time is allowed, i.e., only the main factors n, m, m_i, PF_j, NP_j and PE_{ij} are statistically affecting the difficulty of the MIP model instances.

3.4. Conclusions

In this chapter we have shown a complete formulation as well as a mixed integer programming mathematical model for the hybrid flexible flowshop problem defined in Chapter 2. This model allows for the consideration of realistic scheduling environments. In order to clearly identify the effect of each considered characteristic on the proposed mathematical model, we have solved a comprehensive benchmark and carried out an extensive statistical analysis by means of decision trees. This tool, to the best of our knowledge, has not been applied to the analysis of MIP model performance before. The analysis has allowed us to identify some interesting and counter-intuitive interactions between the many different characteristics of the realistic problem considered. The results establish a sound basis for further analyses of such a complex problem and for the development of heuristics and/or metaheuristics, which are needed to solve larger sized problems in tolerable times. Although considering instances of up to 15 jobs is not practically relevant, our aim here is not to solve practically sized problems using MIP models and CPLEX, but to investigate the effect of the realistic characteristics included in the model on the problem difficulty. The research in this chapter has lead to the publication of Ruiz et al. (2008).

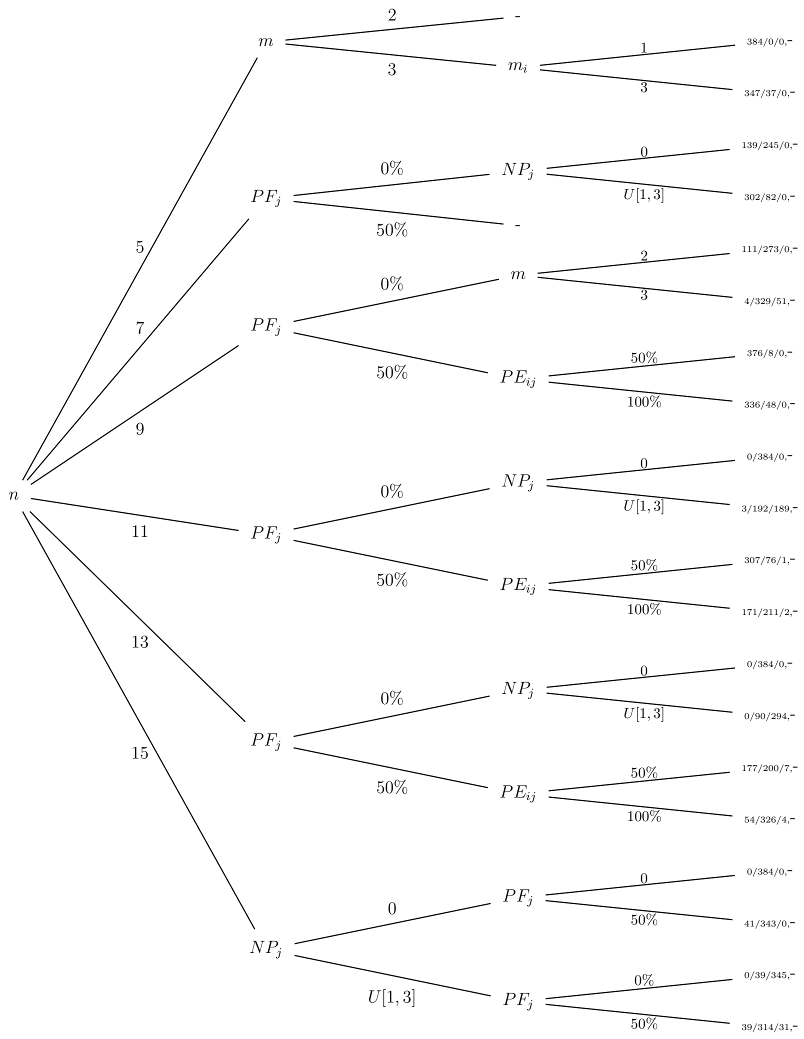


Figure 3.4: Full simplified decision tree, time limit=900 seconds.

CHAPTER 4

HEURISTICS

4.1. Introduction

A first effort on solving the presented HFFL problem consists in the mixed integer programming (MIP) model presented in Chapter 3. The model achieves optimal solutions, but only for a limited problem size. In a set of 9216 instances with $5 \leq n \leq 15$, only 4,003 are solved to optimality within a five minutes limit. The CPLEX 9.1 solver found a feasible solution without optimality guarantee in 4,170 cases. For the remaining 1,043 instances, not even a feasible solution was obtained within 5 minutes. For $n = 15$ the respective numbers of cases are 50, 1,068 and 418 out of 1,536 instances. In Figure 4.1 one can see that allowing CPLEX to run three times more time results in little changes in the number of instances solved to optimality.

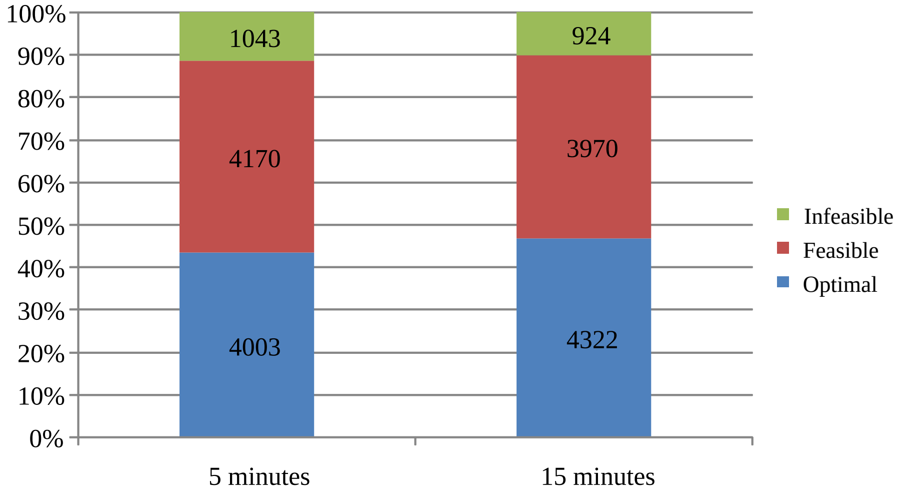


Figure 4.1: Number of problem instances solved by CPLEX within 5 and 15 minutes, respectively.

In practice, the number of jobs tends to be much higher than 15; depending on the sector and on the size of the company, schedules of about 100 jobs are much more common. Besides, CPU times of 15 minutes or more are very inconvenient as time pressure causes the necessity of almost instant schedules. Heuristic methods are a solution to both complications; they use to be both faster and able to manage huge instance sizes.

In order to test, compare and analyse the heuristics, we use a different benchmark of large instances. The data set is based on six factors with a two levels for each factor. Four other factors have fixed values, since it is shown in Chapter 3 that those factors do not have a significant influence on the hardness of the instances. The levels we use for the data set are given in Table 4.1. For each factor combination, there are three instances, resulting in a total of 192 instances. These instances are available at `http://soa.iti.es/problem-instances`. Note again that the total number of predecessors can be higher than the number of direct predecessors. In an example instance of 100 jobs and a $U[1,5]$ distribution for NP_j, the total number of predecessor relationships is 320 and the maximum number of predecessors for one job 26.

Factor	**Symbol**	**Values**
Number of jobs	n	50, 100
Number of stages	m	4, 8
Number of unrelated parallel machines per stage	m_i	2, 4
Distribution of the release dates for the machines	rm_{il}	$U[1, 200]$
Probability for a job to skip a stage	PF_j	0%, 50%
Probability for a machine to be eligible	PE_{ij}	50%, 100%
Distribution of the setup times as a percentage of the processing times	DS_{iljk}	$U[75, 125]$
Probability for the setup time to be anticipatory	PA_{iljk}	$U[50, 100]\%$
Distribution of the lag times	$Dlag_{ilj}$	$U[-99, 99]$
Number of directly preceding jobs	NP_j	0, $U[1, 5]$

Table 4.1: Factors and levels used in the benchmark.

4.2. Machine assignment rules

As has been discussed in Chapter 2, Subsection 2.2.3, there are many possible solution presentations for the HFFL problem. Representations as simple as job permutations are possible, as well as complex job-machine multiple arrays or even schemes with the starting time for each task. This allows for both semi-active and non semi-active schedules, where a schedule is defined as semi-active by Pinedo (2008) if no task can be completed earlier without changing the processing order on any of the machines. If the objective function includes earliness, the optimal solution can be a non semi-active schedule. However, for the makespan objective, each non semi-active schedule is dominated by a semi-active schedule with the same machine assignments and the same job order. The solution space is much smaller with a permutation representation. However, only a simple job permutation does not suffice. Given a certain job permutation, jobs have to be assigned to an eligible machine at each stage. Therefore, we implemented some existing and some new machine assignment rules.

Given a certain job permutation, decisions have to be taken on the machine assignments at each stage. For those decisions nine machine assignment rules have been developed. One of the rules is applied to all the stages a job visits before starting the assignments of the next job in the permutation. All rules calculate a value for each eligible machine using static information on the problem instance and dynamic information on the partial schedule established so far. The machine with the minimal value is chosen.

To describe the machine assignment rules some additional notation needs to be defined. The machine assigned to job j at stage i is denoted by T_{ij} or by l in brief. The previous job that was processed at machine l is denoted by $k(l)$. Let stage $i-1$ be the last stage visited by job j before stage i, stage $i+1$ the next stage to be visited, and stages FS_j and LS_j the first and last stages job j visits, respectively. Let furthermore $A_{i,l,k(l),j} = S_{i,l,k(l),j} = 0$ for $i \notin F_j$ or $i \in F_j$ but $l \notin E_{ij}$ and $A_{i,l,k(l),j} = S_{i,l,k(l),j} = C_{i,k(l)} = 0$ when no preceding job $k(l)$ exists. Completion times for job j at all visited stages can now be calculated with the following expressions:

$$\begin{aligned} C_{FS_j,j} &= \max\{rm_{FS_j,l}; \max_{p\in P_j} C_{LS_p,p}; C_{FS_j,k(l)} + A_{FS_j,l,k(l),j} \cdot S_{FS_j,l,k(l),j}\} \\ &\quad +(1 - A_{FS_j,l,k(l),j}) \cdot S_{FS_j,l,k(l),j} + p_{FS_j,l,j}, \qquad j \in N \end{aligned} \tag{4.1}$$

$$\begin{aligned} C_{ij} &= \max\{rm_{il}; C_{i,k(l)} + A_{i,l,k(l),j} \cdot S_{i,l,k(l),j}; C_{i-1,j} + lag_{i-1,T_{i-1,j},j}\} \\ &\quad +(1 - A_{i,l,k(l),j}) \cdot S_{i,l,k(l),j} + p_{ilj}, \qquad j \in N, i > FS_j \end{aligned} \tag{4.2}$$

The calculations should be made job-by-job to obtain the completion times of all tasks. For each job, the completion time for the first stage is calculated with Equation (4.1), considering availability of the machine, completion times of the predecessors, setup and its own processing time. For the other stages Equation (4.2) is applied, considering availability of the machine, availability of the job (including lag), setup and its processing time.

If job j is assigned to machine l inside stage i, the time at which machine l completes job j is denoted as L_{ilj}. Following our notation, $L_{ilj} = C_{ij}$ given $T_{ij} = l$. Furthermore, we refer to the job visiting stage i after job j as job q and to an eligible machine at the next stage for job j as $l' \in E_{i+1,j}$.

Suppose now that we are scheduling job j in stage i, $i \in F_j$. We have to consider all machines $l \in E_{ij}$ for assignment. The proposed assignment rules are the following:

4.2.1. Rules based on current job, current stage

1. First Available Machine (FAM): Assigns the job to the first available eligible machine. This is the machine with the minimum liberation time from its last scheduled job, or lowest release date if no job is scheduled at the machine yet, i.e. $T_{ij} = l$ such that $\min\limits_{l \in E_{ij}} \{\max(L_{ilk}; rm_{il})\}$.

2. Earliest Starting Time (EST): Chooses the machine that is able to start job j at the earliest time. Therefore we also have to take the availability of the job and setup times into account. The decision value can be described as follows:
$\min\limits_{l \in E_{ij}} \{\max\{rm_{il}; L_{ilk} + A_{ilkj} \cdot S_{ilkj}; \max\limits_{h \in P_j} C_{LS_h h}\} + (1 - A_{ilkj}) \cdot S_{ilkj}\}$
if $i = FS_j$, else
$\min\limits_{l \in E_{ij}} \{\max\{rm_{il}; L_{ilk} + A_{ilkj} S_{ilkj}; C_{i-1,j} + lag_{i-1,l',j}\} + (1 - A_{ilkj}) S_{ilkj}\}$.

3. Earliest Completion Time (ECT): Takes the eligible machine capable of completing job j at the earliest possible time. Thus the difference with the previous rule is that this rule includes processing times. Job j is assigned to machine l such that $\min\limits_{l \in E_{ij}} L_{ilj}$. We refer to Equations 4.1 and 4.2 for the calculation of this value.

4. Earliest Preparation Next Stage (EPNS): The machine able to prepare the job at the earliest time for the next stage to be visited is chosen. Therefore time lags between the current and the next stage are taken into account by assigning job j to machine l with $\min\limits_{l \in E_{ij}} \{L_{ilj} + lag_{ilj}\}$. The rule uses more information about the continuation of the job, without directly focusing on the machines in the next stage. If $i = LS_j$ this rule reduces to ECT.

4.2.2. Look-ahead rules

The rules proposed in Subsection 4.2.1 only use information on the current job and the machines in the current stage. The number of cases that has to be compared is therefore $|E_{ij}|$ in rules 1 to 4. By making assumptions on future decisions, or simply by using averages, we can also use information on jobs yet to schedule and/or machines in later stages. Rules taking into account this type of information are usually called look-ahead rules. The complexity of those rules is generally higher.

5. Earliest Completion Next Stage (ECNS): The availability of machines in the next stage to be visited and the corresponding processing times are considered as well, since we assign job j to the machine in stage i that can make the job be finished earliest in the next visited stage $i+1$. Note that we are assigning only to stage i. Formally, the decision value for machine l at stage i, can be written as: $\min\limits_{l\in E_{ij}, l'\in E_{i+1,j}} \{L_{i+1,l',j}|T_{ij}=l\} = \min\limits_{l'\in M_{i+1}} (\max\{rm_{i+1,l'}; C_{i+1,k(l')} + A_{i+1,l',k(l'),j} \cdot S_{i+1,l',k(l'),j}; L_{ilj} + lag_{ilj}\} + (1-A_{i+1,l',k(l'),j}) \cdot S_{i+1,l',k(l'),j} + p_{i+1,l',j})$. The consideration of the machines in the next stage implies a somewhat longer calculation. The completion time has to be evaluated for each combination of one machine at stage i and one machine at stage $i+1$. This means a total of $|E_{ij}| \cdot |E_{i+1,j}|$ completion time evaluations. The rule reduces to ECT if no single minimum is found, or if $i = LS_j$.

6. Forbidden Machine (FM): Excludes machine l^* that is able to finish job q earliest. ECT is applied to the remaining eligible machines for job j. While the foregoing rules are greedy, worse results might be expected for later jobs. This rule is supposed to obtain better results for later jobs, as it reserves the machine able to finish the next job earliest. Mathematically, we choose machine l considering $\min\limits_{l\in E_{ij}}\{L_{ilj} - |l - l^*| \cdot I\}$ where I is a high positive number and l^* given by $\min\limits_{l^*\in E_{iq}} \{L_{i,l^*,f} + S_{i,l^*,f,q} + p_{i,l^*,q}\}$, job f being the last job scheduled at l^*. The number of calculations that has to be made in order to apply this rule is $|E_{iq}| + |E_{ij}| - 1$ if $l^* \in E_{ij}$ and $|E_{iq}| + |E_{ij}|$ otherwise. Note that job j is assigned to machine l^* if

this is the only eligible machine. ECT is applied if $j \in P_q$ as job j has to be finished as early as possible in this case, or if job j is the last job at stage i.

7. Next Job Same Machine (NJSM): The assumption is made that job q is assigned to the same machine as job j. Assigned machine T_{ij} is chosen such that job q is finished earliest. So machine l is chosen by optimising $\min\limits_{l \in E_{ij}} L_{ilj} + S_{iljq} + p_{ilq}$. Note that only job j is assigned. The rule is especially useful if setups are relatively large, as the foregoing rules do not take the setup between job j and job q into account. The number of cases to be compared is $|E_{ij}|$, just like in the non look-ahead rules. The difference is that each calculation requires a constant additional amount of time. Reduces to ECT if job j is the last at this stage.

8. Sum Completion Times (SCT): Completion times of job j and job q are calculated for all eligible machine combinations at stage i. The number of combinations is $|E_{ij}| \cdot |E_{iq}|$. Machine l is chosen such that the sum of both completion times is the smallest: $\min\limits_{l \in E_{ij}, l^* \in E_{iq}} \{L_{ilj} + L_{i,l^*,q}\}$. Similar to NJSM, but without the assumption that job q is assigned to the same machine. Reduces to ECT if job j is the last at stage i.

9. Anticipatory Based (AB): Concentrates on possibilities for future anticipatory setups. Non-anticipatory setups might cause important delays. Therefore this rule tries to avoid this type of setups. Anticipation factor $AF_l = \sum\limits_{h \in H} A_{iljh} \cdot S_{iljh} / |E_{ih}|$ expresses the expected advantage caused by the anticipatory setups, H being the set of jobs sequenced after job j. The factor is subtracted from the EPNS value and the result $\min\limits_{l \in E_{ij}} \{L_{ilj} + lag_{ilj} - AF_l\}$ gives the machine l to which to assign job j. The complexity of the calculation is in $O(n \cdot m_i)$. Reduces to EPNS if job j is the last job at this stage.

Especially for the first five assignment rules, the growing amount of information used represents a tradeoff between the probability on good schedules on the one hand, and valuable computation time on the other hand. The remaining four rules are designed for alternative assignments, concentrating on drawbacks of the earlier rules. These rules are original and exploit specific information of this problem. Since the problem is very complex, more complex machine assignment rules are needed than mostly used in the literature.

In the following pages, all machine assignment rules are applied to example instance 2, starting from the same job permutation. As a result, nine different makespan values are obtained. Table 4.2 gives the processing times and the time lags for example instance 2 and Table 4.3 gives the anticipatory and non anticipatory setup times. In example instance 2 no stages are skipped and all machines are eligible for all jobs. No precedence relationships among jobs exist in this example. The machine release dates are 149, 85 and 188 for the machines in stage 1, 127, 55 and 160 for the machines in stage 2, and 184, 104 and 180 for the machines in the third stage.

Stage	1			2			3		
Machine	1	2	3	4	5	6	7	8	9
Job									
1	70(25)	89(-54)	77(-12)	79(95)	98(-3)	54(71)	67	3	48
2	59(-7)	81(-33)	79(93)	41(19)	52(-20)	84(49)	23	23	71
3	47(43)	14(-14)	27(56)	76(38)	67(89)	76(73)	92	71	6
4	18(39)	24(80)	92(-32)	93(-9)	63(59)	54(96)	90	48	53
5	70(-1)	16(11)	83(3)	32(-13)	95(42)	95(-13)	24	13	37

Table 4.2: Example instance 2. Processing times of each job on each eligible machine. In brackets the time lag (if applicable).

Stage	Machine	Job	1	2	3	4	5
1	1	1	-	92(0)	119(1)	82(1)	104(0)
		2	83(1)	-	79(1)	86(1)	102(1)
		3	122(1)	79(0)	-	100(0)	101(1)
		4	123(1)	120(1)	92(1)	-	85(1)
		5	100(1)	124(0)	85(1)	87(1)	-
	2	1	-	102(1)	123(1)	86(1)	121(0)
		2	78(1)	-	108(0)	89(0)	85(1)
		3	112(1)	84(0)	-	95(0)	86(0)
		4	92(1)	76(0)	76(1)	-	120(0)
		5	105(0)	93(1)	122(1)	91(0)	-
	3	1	-	75(1)	87(0)	100(0)	109(0)
		2	114(1)	-	78(1)	109(1)	77(0)
		3	93(1)	113(1)	-	75(0)	112(0)
		4	119(1)	92(1)	91(1)	-	82(1)
		5	87(1)	125(0)	85(1)	125(1)	-
2	1	1	-	93(1)	116(0)	83(0)	101(0)
		2	87(1)	-	104(1)	83(0)	82(0)
		3	114(0)	102(1)	-	121(0)	100(0)
		4	88(0)	75(0)	83(1)	-	117(1)
		5	78(1)	81(1)	125(1)	90(1)	-
	2	1	-	97(0)	79(0)	92(0)	109(0)
		2	96(1)	-	115(0)	123(1)	91(0)
		3	86(0)	112(0)	-	80(1)	102(1)
		4	95(0)	104(1)	104(0)	-	113(0)
		5	124(1)	100(1)	95(1)	119(0)	-
	3	1	-	85(0)	101(1)	109(0)	80(1)
		2	107(0)	-	99(1)	116(1)	123(1)
		3	103(0)	77(1)	-	123(0)	119(0)
		4	76(0)	76(1)	95(0)	-	92(0)
		5	102(0)	83(1)	95(0)	79(1)	-
3	1	1	-	80(0)	115(0)	95(1)	93(1)
		2	78(0)	-	82(0)	115(1)	86(0)
		3	113(1)	96(1)	-	114(1)	106(0)
		4	100(1)	108(1)	113(0)	-	122(0)
		5	121(1)	121(1)	98(1)	101(0)	-
	2	1	-	101(0)	109(1)	112(1)	120(0)
		2	117(0)	-	90(0)	103(1)	78(1)
		3	103(0)	114(0)	-	118(1)	118(1)
		4	117(1)	85(1)	86(1)	-	98(1)
		5	86(0)	77(0)	98(0)	98(1)	-
	3	1	-	86(1)	108(0)	101(1)	121(1)
		2	89(0)	-	108(0)	101(0)	99(0)
		3	110(0)	107(0)	-	117(1)	102(0)
		4	119(0)	113(0)	122(0)	-	117(1)
		5	117(1)	86(0)	94(0)	81(1)	-

Table 4.3: Example instance 2. Setup times between pairs of jobs at each machine. A “1” in brackets indicates that setup times are anticipatory, a “0” that they are not.

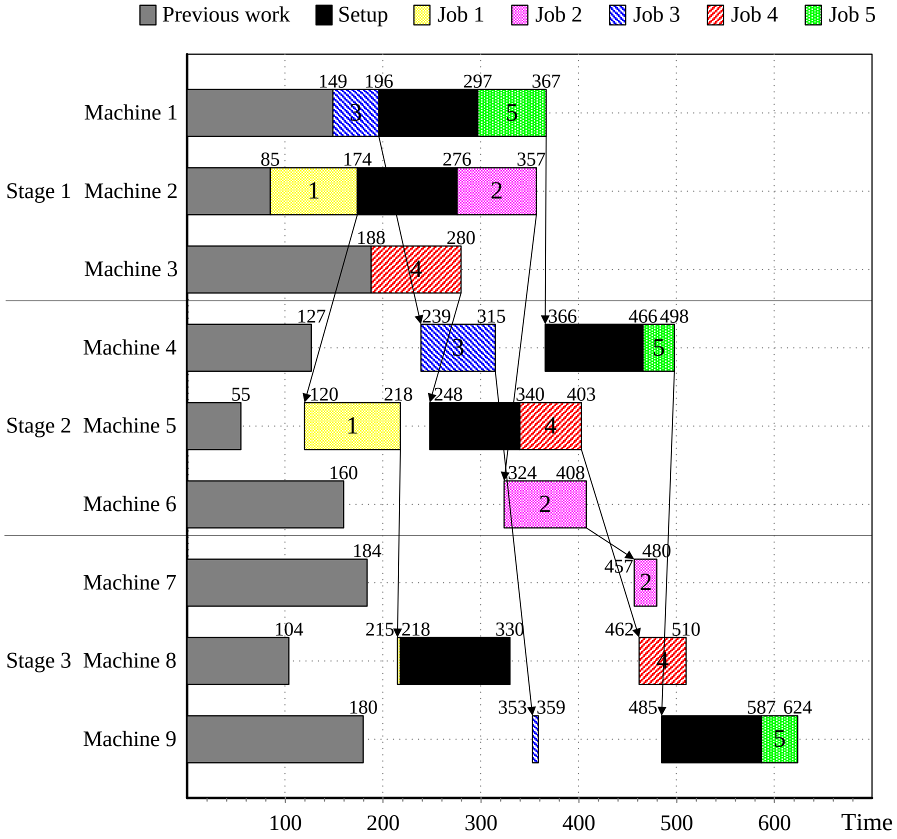


Figure 4.2: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 1. Makespan value: 624.

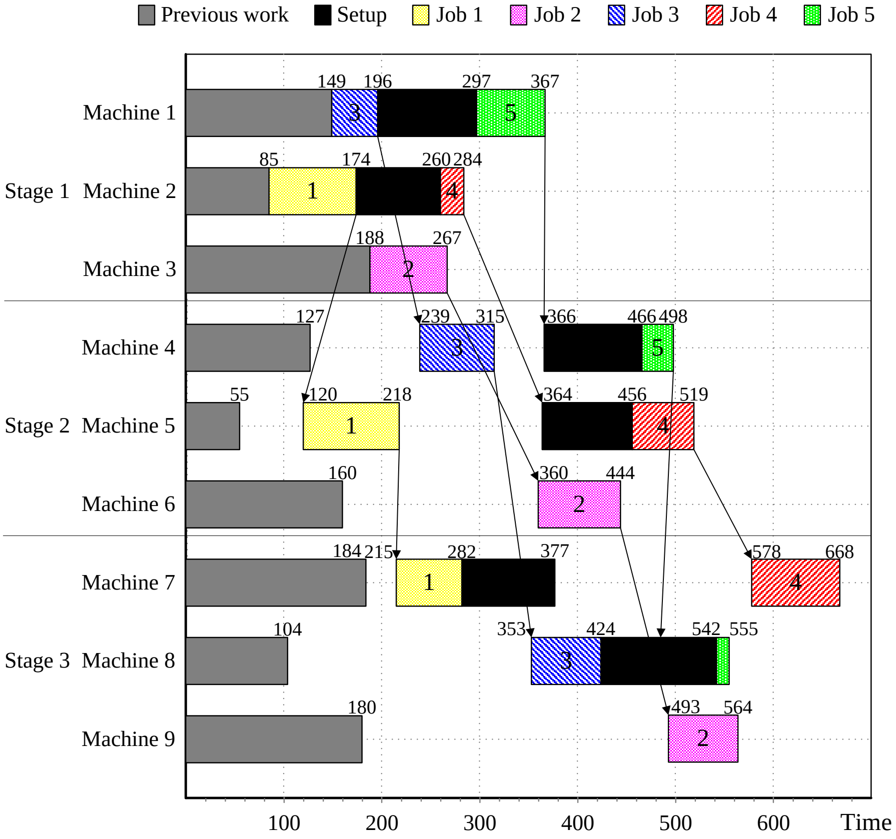


Figure 4.3: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 2. Makespan value: 668.

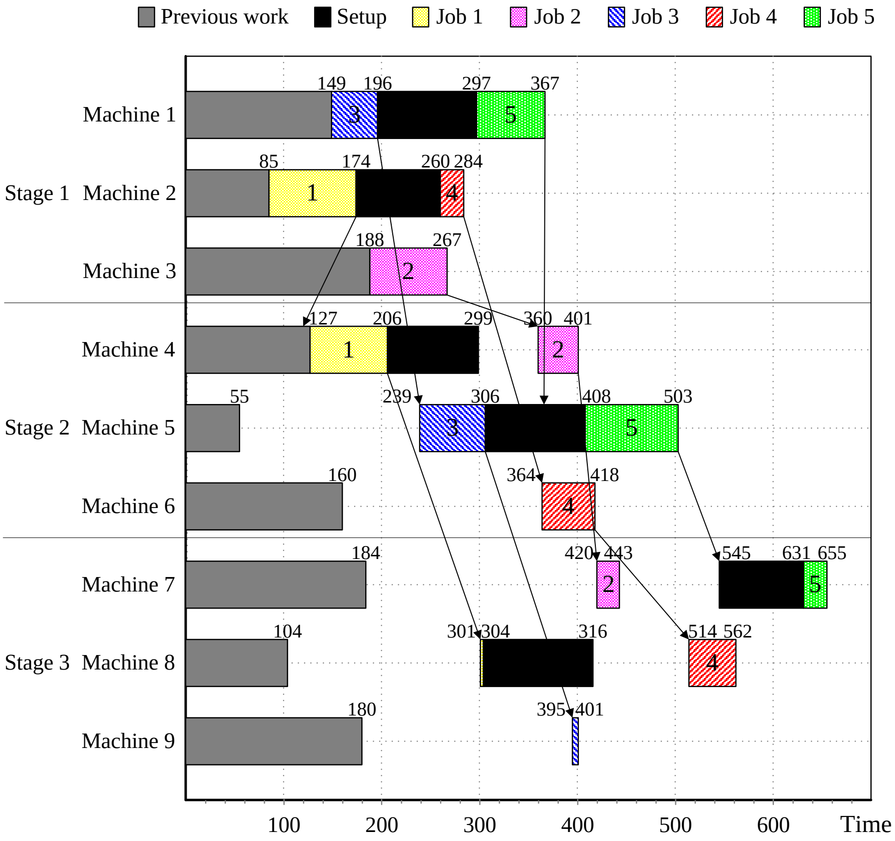


Figure 4.4: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 3. Makespan value: 655.

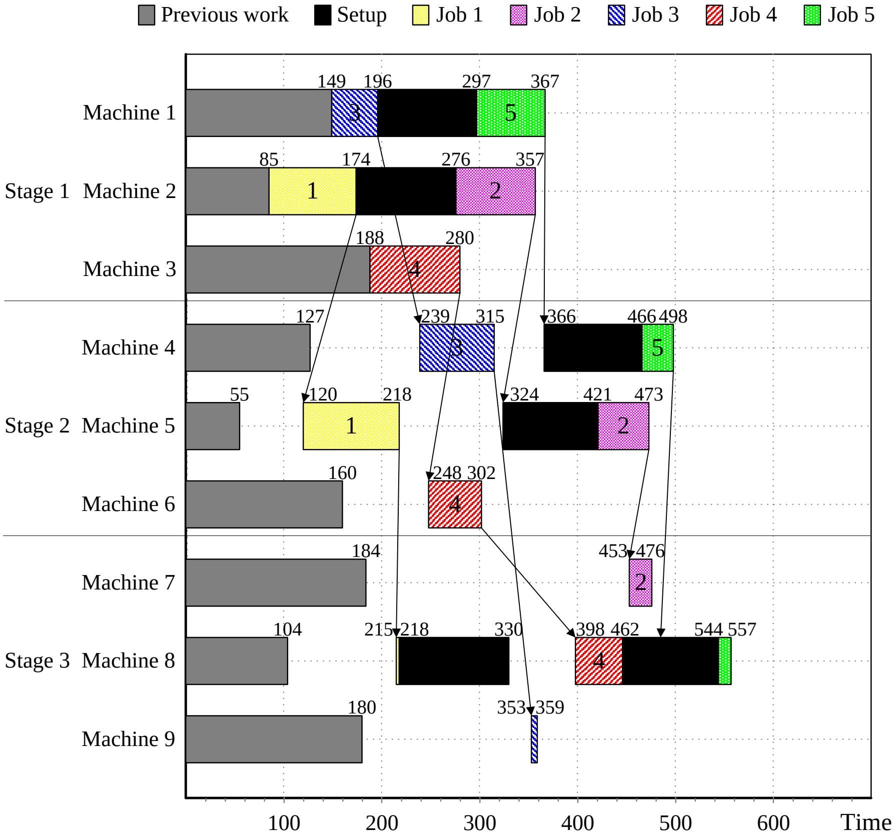


Figure 4.5: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 4. Makespan value: 557.

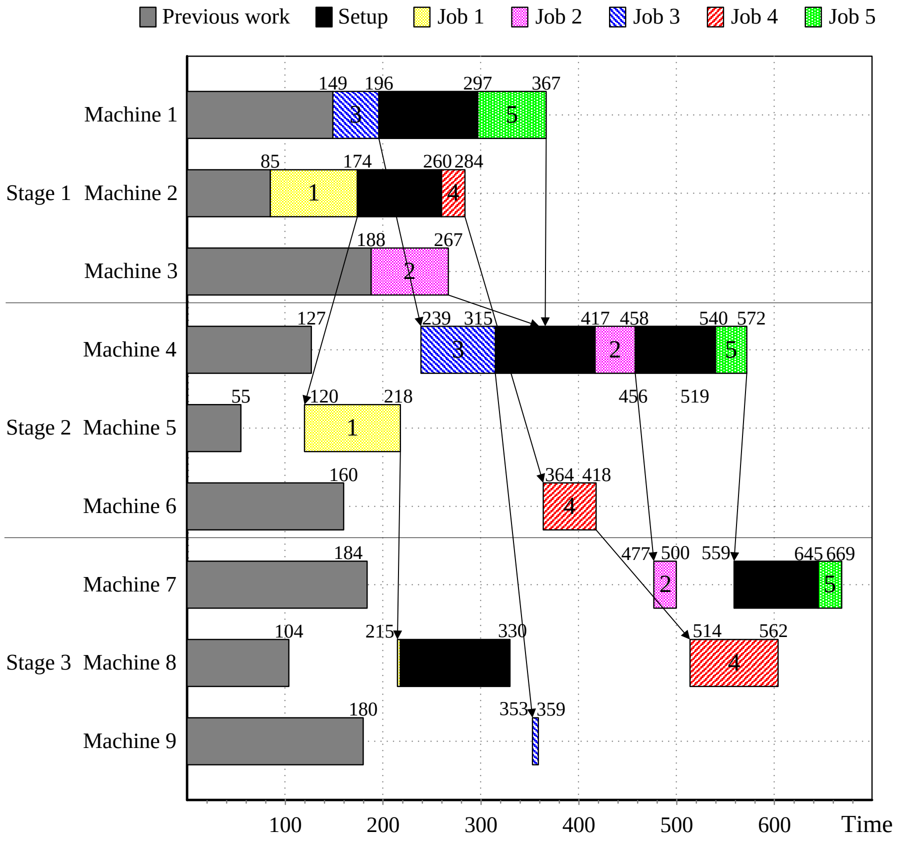


Figure 4.6: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 5. Makespan value: 669.

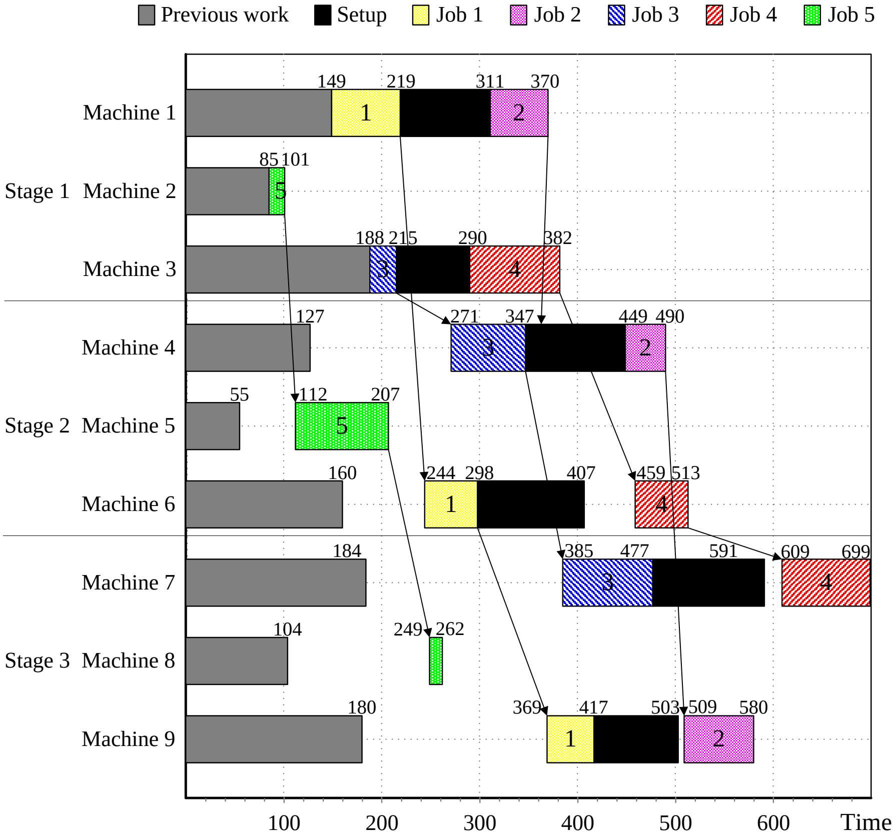


Figure 4.7: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 6. Makespan value: 699.

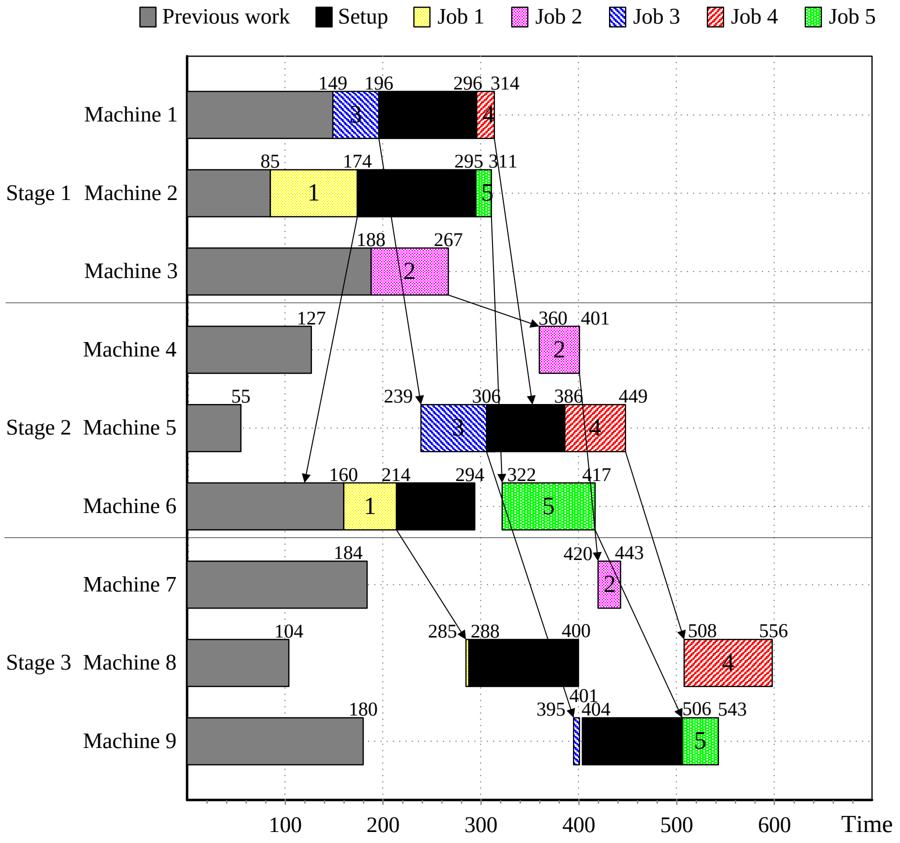


Figure 4.8: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 7. Makespan value: 556.

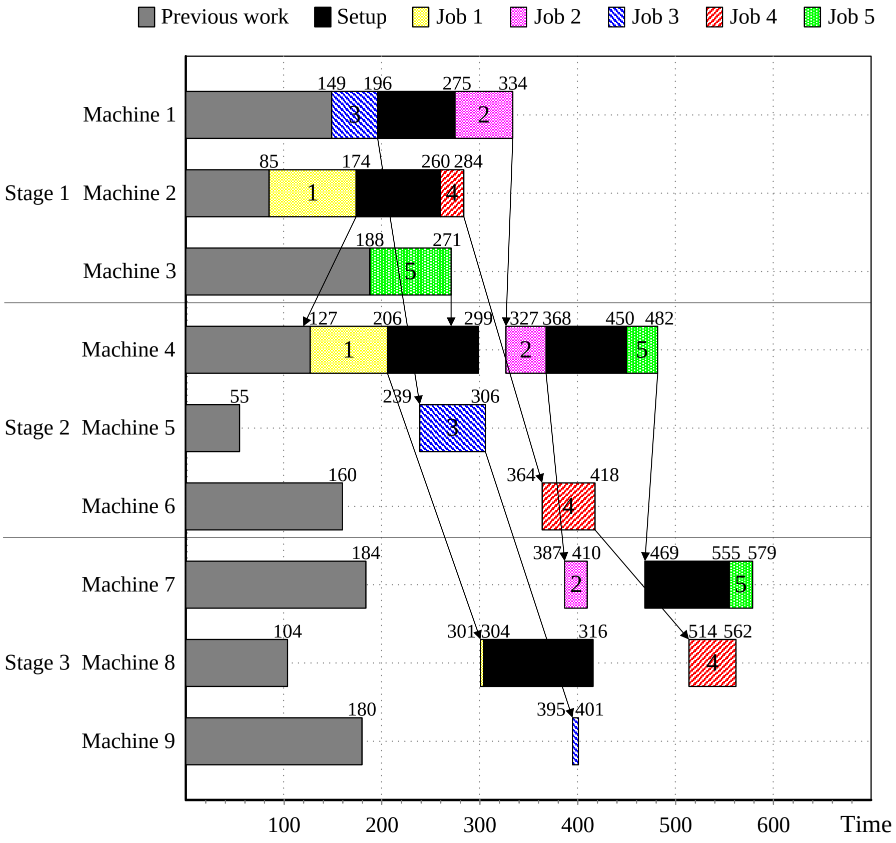


Figure 4.9: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 8. Makespan value: 579.

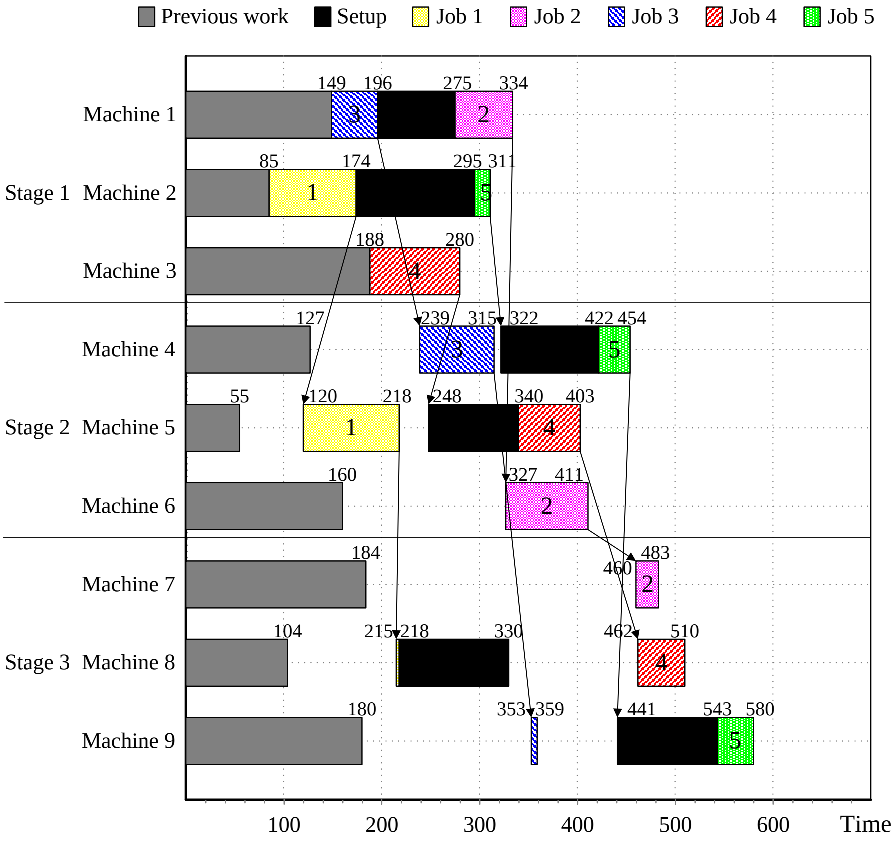


Figure 4.10: Gantt of solution obtained applying job permutation (1,3,2,4,5) and machine assignment rule 9. Makespan value: 580.

4.3. Solution representations

Since the hybrid flowshop problem has multiple dimensions, different solution representations are possible. Several possibilities that can be found in literature are given in Subsection 2.2.3. The choice of the representation is of crucial importance for the results obtained by heuristics or metaheuristics. One should take into account the tradeoff: A too verbose representation results in an inefficient algorithm and a too compact representation might exclude important solutions. In order to illustrate this second point, we use example instance 3 in this section. The instance consists in five jobs that have to be processed at three stages, where each stage has three parallel unrelated machines. It is a special case of the considered problem, while all release dates are assumed to be zero, no stages are skipped, all machines are eligible and no precedence relationships exist. Table 8.1 gives the processing times and the time lags and Table 4.5 gives the anticipatory and non anticipatory setup times.

In the remaining of this section, four distinct solution representations are considered in detail.

Stage	1			2			3		
Machine	1	2	3	4	5	6	7	8	9
Job									
1	18(29)	62(-24)	23(67)	61(3)	36(70)	24(60)	90	13	86
2	65(92)	29(-18)	98(61)	76(61)	37(18)	18(-2)	82	93	17
3	6(45)	53(-28)	39(-28)	28(-28)	30(82)	78(66)	70	58	52
4	56(-22)	69(-31)	34(-31)	81(38)	95(-37)	31(68)	47	48	71
5	25(-22)	38(96)	57(37)	22(-2)	99(9)	68(5)	14	2	15

Table 4.4: Example instance 3. Processing times of each job on each eligible machine. In brackets the time lag (if applicable).

4.3.1. Permutation with a single rule for machine assignment

The most compact representation consists of a job sequence and the machine assignment rule used for all jobs. This can be seen in Figure 4.11, where S^A, the best possible solution with this representation, is represented. The makespan for S^A is 191, as can be seen in the Gantt diagram in Figure 4.12. The chromosome size is $n + 1$ and the number of possible solutions is $n! \cdot\ r$, where r is the number of machine assignment rules. In Figure 4.13 is shown how the number of chromosomes grows for increasing n, given that $r = 9$. Note that some chromosomes might represent the same solution, as distinct rules might lead to the same choice of machine assignments. Additionally, permutations can be infeasible because of the precedence constraints. In case of one precedence

Stage	Machine	Job	1	2	3	4	5
1	1	1	-	38(0)	34(0)	32(0)	71(1)
		2	42(1)	-	51(1)	65(0)	65(0)
		3	63(0)	47(0)	-	72(1)	25(0)
		4	39(1)	39(0)	29(0)	-	26(1)
		5	37(0)	70(1)	59(0)	30(0)	-
	2	1	-	33(1)	45(1)	35(0)	72(1)
		2	36(1)	-	40(1)	68(0)	49(1)
		3	45(0)	60(0)	-	26(1)	40(1)
		4	42(1)	45(0)	50(0)	-	59(0)
		5	32(1)	56(0)	67(0)	71(0)	-
	3	1	-	66(0)	70(1)	27(1)	30(1)
		2	73(0)	-	40(1)	54(1)	25(0)
		3	27(1)	27(1)	-	26(1)	61(0)
		4	28(0)	32(1)	49(0)	-	58(1)
		5	62(1)	58(1)	60(1)	58(0)	-
2	1	1	-	55(0)	69(1)	46(1)	39(1)
		2	34(0)	-	72(0)	42(1)	73(0)
		3	53(0)	44(0)	-	37(0)	29(0)
		4	70(1)	50(0)	45(1)	-	47(1)
		5	59(0)	69(0)	32(0)	60(1)	-
	2	1	-	57(0)	43(1)	51(0)	53(1)
		2	69(0)	-	63(0)	64(1)	70(0)
		3	54(1)	34(1)	-	36(0)	39(1)
		4	73(0)	73(0)	34(1)	-	36(0)
		5	74(1)	36(0)	58(1)	34(0)	-
	3	1	-	52(0)	73(0)	54(1)	61(0)
		2	29(1)	-	67(1)	69(1)	46(0)
		3	43(1)	38(1)	-	67(0)	25(0)
		4	38(0)	49(0)	33(0)	-	70(0)
		5	74(0)	50(0)	26(0)	68(1)	-
3	1	1	-	52(0)	43(1)	25(0)	47(1)
		2	55(1)	-	25(0)	46(1)	49(1)
		3	52(1)	54(0)	-	71(0)	28(0)
		4	43(1)	31(1)	59(1)	-	41(1)
		5	49(1)	69(0)	36(1)	36(1)	-
	2	1	-	68(1)	66(0)	50(0)	37(0)
		2	62(0)	-	70(0)	45(1)	40(1)
		3	29(1)	35(0)	-	45(0)	61(0)
		4	53(1)	32(0)	49(0)	-	71(0)
		5	50(1)	37(0)	27(1)	74(0)	-
	3	1	-	35(1)	36(1)	26(0)	50(1)
		2	57(1)	-	39(0)	53(0)	40(0)
		3	39(1)	25(1)	-	31(1)	66(0)
		4	64(0)	30(1)	56(1)	-	44(1)
		5	49(0)	28(0)	32(1)	56(1)	-

Table 4.5: Example instance 3. Setup times between pairs of jobs at each machine. A “1” in brackets indicates that setup times are anticipatory, a “0” that they are not.

relationship, the number of feasible chromosomes is half of the total number of chromosomes. This is only a quarter if two precedence constraints exist between four distinct jobs, and one third if one job has two precedence relationships.

Note that not all possible solutions are reachable with this representation. For example, the solution given in Figure 2.5 is not reachable since the jobs do not visit the machines in the same order (non-permutation solutions).

Assignment rule	Job permutation				
4	5	1	3	4	2

Figure 4.11: Permutation with single machine assignment rule.

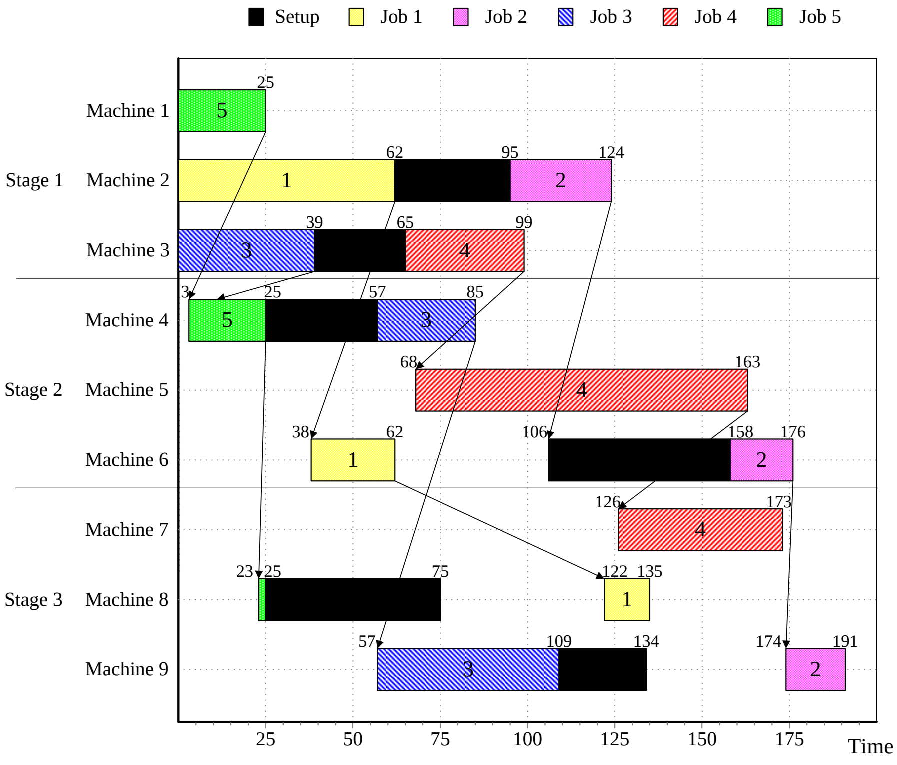


Figure 4.12: Gantt of solution S^A.

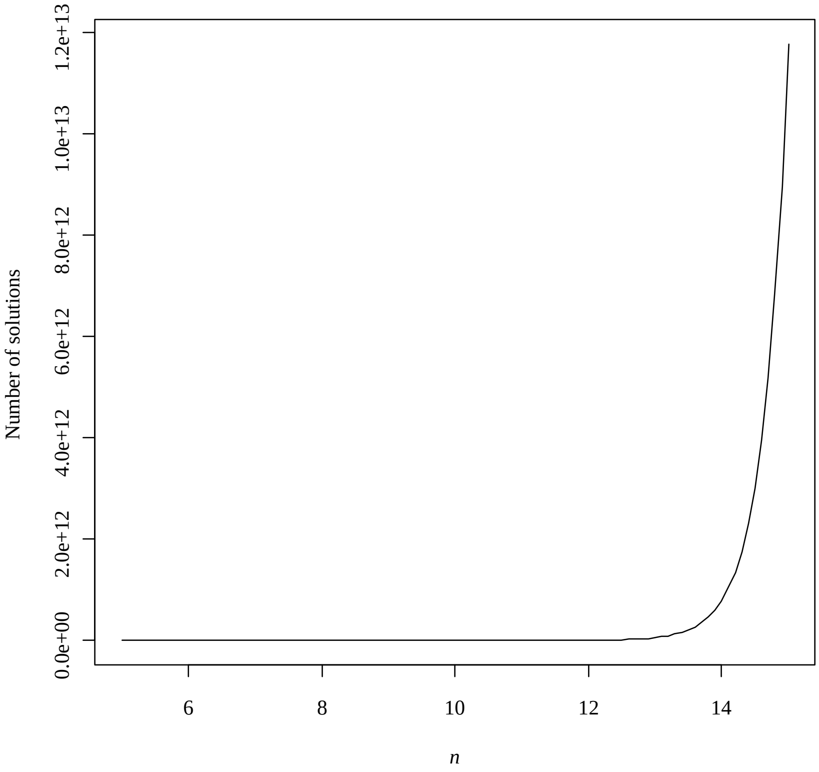


Figure 4.13: Number of possible solutions for different numbers of jobs; permutation with a single rule for machine assignment. $r = 9$.

Number of jobs	Number of solutions
5	1,080
7	45,360
9	3,265,920
11	359,251,200
13	56,043,187,200
15	1.17691E+13

Table 4.6: Number of possible solutions for a permutation with a single rule for machine assignment. $r = 9$.

4.3.2. Permutation with a machine assignment rule for each job

Allowing independent machine assignment rules for every job in the sequence yields a more flexible representation. More machine assignment combinations are possible and as a result more good solutions can be represented. The best solution for the example problem instance of this section is solution S^B, shown in Figures 4.14 and 4.15. The makespan of this solution has a value of 185 and is indeed better than any solution with a single machine assignment rule. This chromosome structure has a size of $2 \cdot n$ and leads to $n! \cdot r^n$ different chromosomes.

Job permutation	2	4	1	3	5
Assignment rules	3	5	3	5	4

Figure 4.14: Permutation with a machine assignment rule for each job.

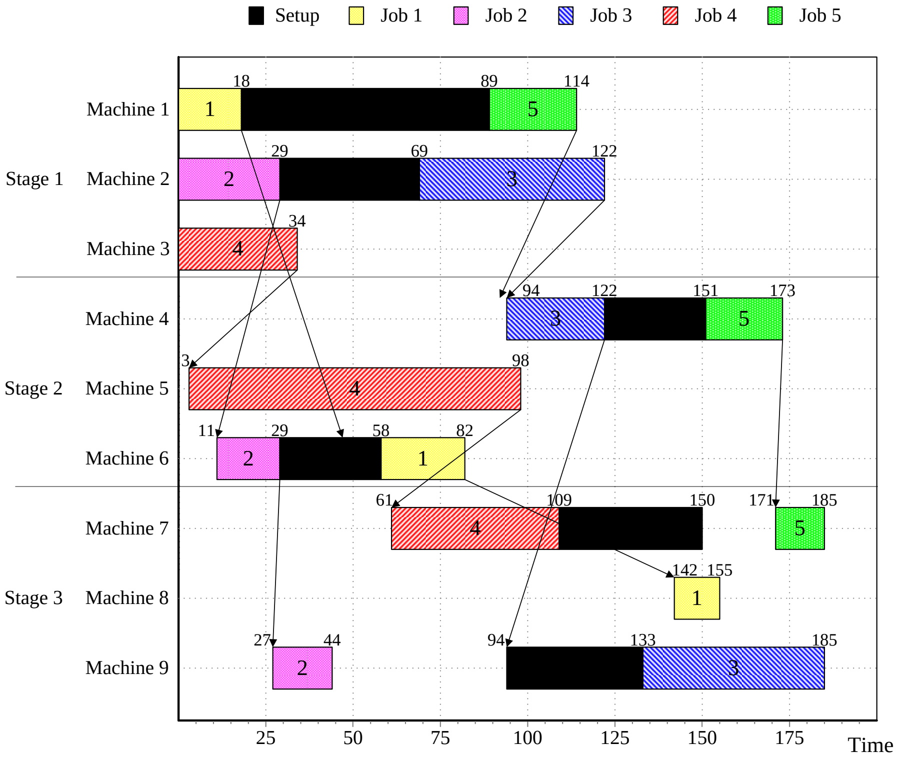


Figure 4.15: Gantt of solution S^B.

In Figure 4.16, the number of possible chromosomes is shown for an increasing number of jobs, given that r is nine.

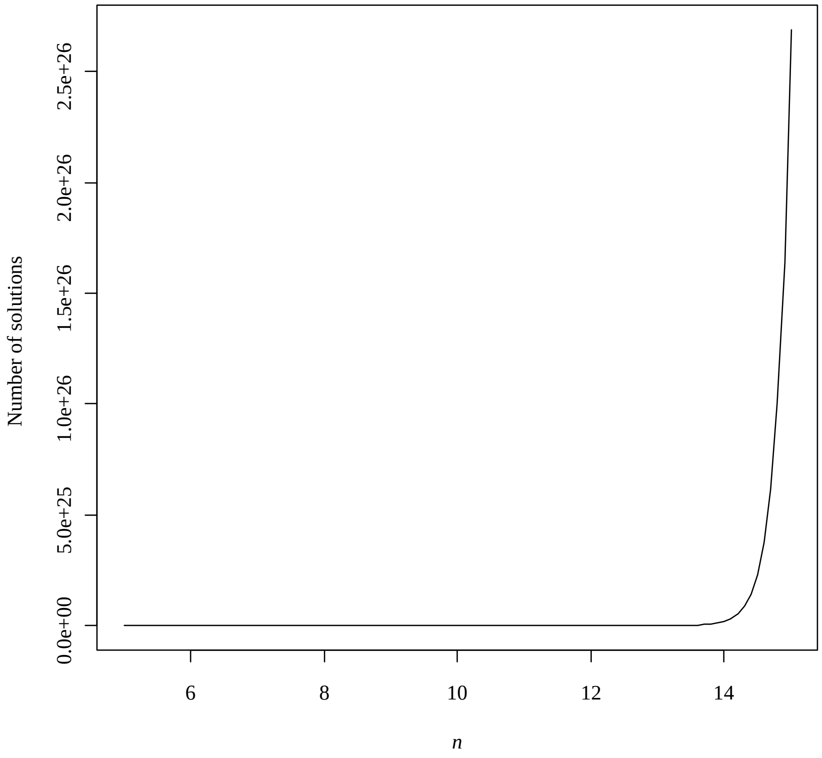


Figure 4.16: Number of possible solutions for different numbers of jobs; permutation with a machine assignment rule for each job. $r = 9$.

Number of jobs	Number of solutions
5	7,085,880
7	24,106,163,760
9	1.40587E+14
11	1.25263E+18
13	1.58283E+22
15	2.69239E+26

Table 4.7: Number of possible solutions for a permutation with a machine assignment rule for each job. $r = 9$.

4.3.3. Permutation with the machine assignments for each job

In both foregoing representations, machine assignment rules take the decisions which machine to use for which job. Instead of taking these decisions with the help of a rule, the algorithm itself can also work on the assignments. By incorporating the machine assignments in the representation, the job-machine combinations can be optimised by the algorithm. This means that there are, apart from the job sequence, n arrays giving machine assignments associated with m stages respectively. The best solution for this representation of size $(1+m)n$, solution S^C, is demonstrated in Figures 4.17 and 4.18. In the latter we can see that makespan has decreased to a value of 183 with the increase of representation directness. It is easy to verify that the number of possible solutions is $n! \cdot \prod_{j \in N} \prod_{i \in F_j} |E_{ij}|$. The increase of the number solutions for increasing instance size is shown graphically in Figure 4.19. Note that the number of stages is irrelevant when each stage contains only one machine.

Job permutation	1	3	2	4	5	
	1	3	2	3	1	Stage 1
Machine assignments	6	4	6	5	4	Stage 2
	8	9	9	7	8	Stage 3

Figure 4.17: Permutation with all machine assignments in the representation.

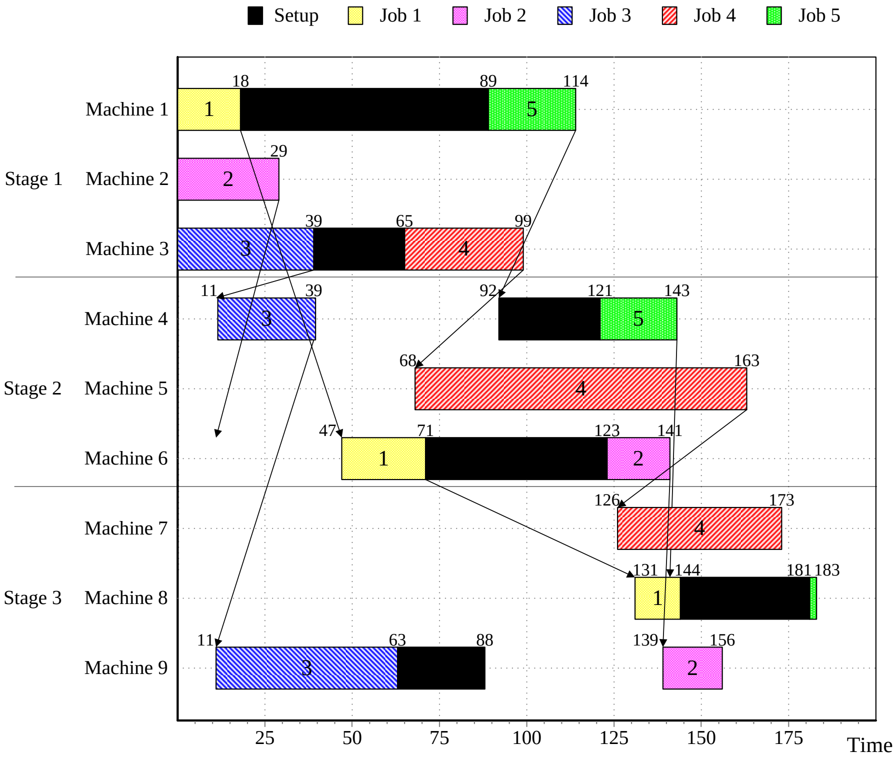


Figure 4.18: Gantt of solution S^C.

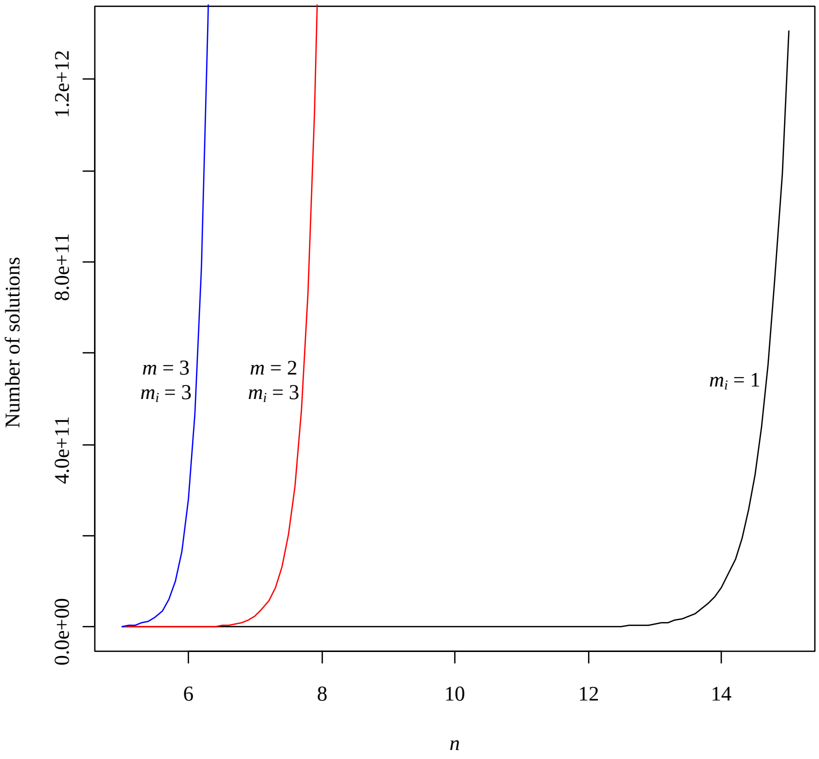


Figure 4.19: Number of possible solutions for different instance sizes; permutation with the machine assignments for each job.

m	m_i	n	**Number of solutions**
-	1	5	120
		7	5,040
		9	362,880
		11	39,916,800
		13	6,227,020,800
		15	1.30767E+12
2	3	5	7,085,880
		7	24,106,163,760
		9	1.40587E+14
		11	1.25263E+18
		13	1.58283E+22
		15	2.69239E+26
3	3	5	1,721,868,840
		7	5.27202E+13
		9	2.76718E+18
		11	2.219E+23
		13	2.52353E+28
		15	3.86328E+33

Table 4.8: Number of possible solutions for a permutation with the machine assignments for each job.

4.3.4. Ordered list of tasks for each machine

A representation is called direct if each chromosome corresponds to a solution and vice versa. In the case of the HFFL also non-permutation solutions should be taken into account. This can be achieved with a list of tasks in processing order for each machine. The size of this representation is equal to the number of tasks, which is equal to $\sum_{j \in N} |F_j|$. In Figures 4.20 and 4.21, S^D, the best solution for the complete solution representation is shown. By definition, this solution is the optimal solution for this problem instance. As can be observed, the makespan of 182 is lower than all previous makespans for this problem instance.

Note that there is no reason to delay any tasks, so all tasks are started at the earliest possible moment, given by Equations (4.1) and (4.2) in Chapter 4.2.

Stage	Machine	Jobs	
1	1	3	5
	2	1	2
	3	4	
2	4	5	3
	5	2	
	6	4	1
3	7	4	
	8	5	1
	9	3	2

Figure 4.20: Ordered lists of tasks to process for each machine.

If the objective would have been earliness-tardiness or something similar, the start times of all tasks would have mattered and a direct representation would request this information within the representation.
The number of possible solutions is equal to the number of possible solutions for the HFFL problem, without considering precedence constraints and machine eligibility. To derive the number of solutions we first concentrate a moment on the problem of non-identical parallel machines, which is actually a HFFL with only one stage. Let m be the number of parallel machines and let the variable $k_l \leq n$ be the number of jobs assigned to machine l. We denote the solution space of the (sub-)problem with m machines Φ_m. Let's analyse the solution space for some small values of m:

$$card(\Phi_1) = n! \tag{4.3}$$

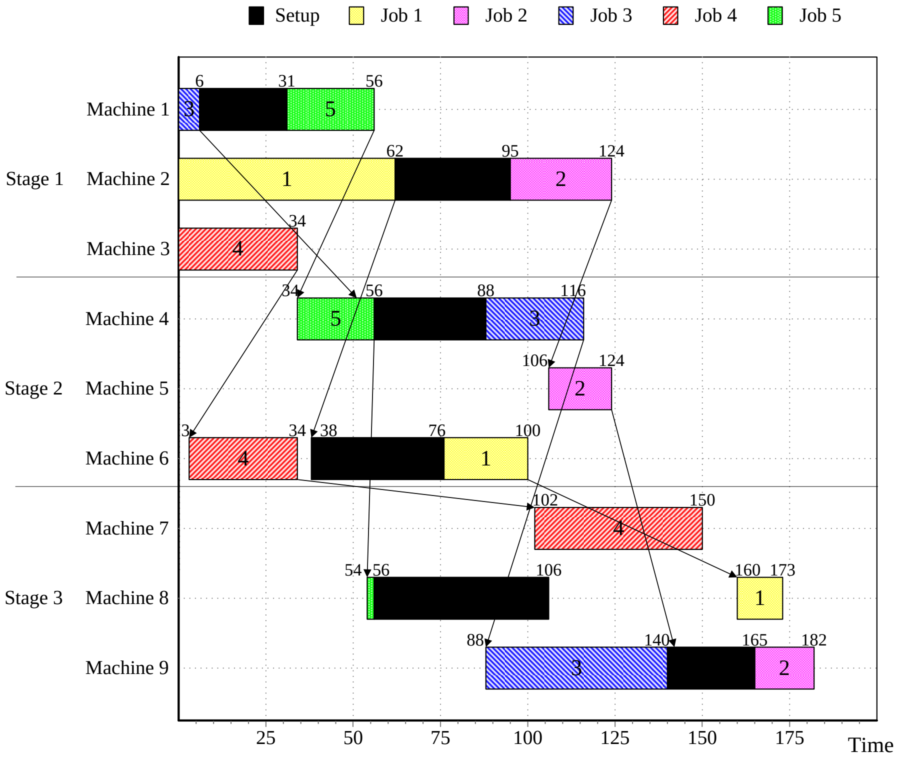


Figure 4.21: Gantt of solution S^D.

For one machine (Equation 4.3) the solution space is straightforward, since only one permutation is involved.

$$
\begin{gathered}
card(\Phi_2) = \sum_{k_1=0}^{n} \left(\binom{n}{k_1} k_1!(n-k_1)! \right) = \\
\sum_{k_1=0}^{n} \left(\frac{n!}{k_1!(n-k_1)!} k_1!(n-k_1)! \right) = \sum_{k_1=0}^{n} n! = n! \sum_{k_1=0}^{n} 1 = \\
n!(n+1)
\end{gathered}
\tag{4.4}
$$

The expression $\binom{n}{k_1}$ in Equation 4.4 gives the number of possible combinations to assign k_1 jobs in machine 1 and $n - k_1$ jobs in machine 2. One should multiply by $k_1!$ to take into account the order of the jobs in machine 1 and by $(n-k_1)!$ for the order in machine 2. The sum over k_1 from 0 to n gives the full amount of possibilities, since any number of jobs in this range can be assigned to machine 1.

The simplification can also be explained intuitively. Any permutation, cut in two parts in any position, can be used as a solution. The first part of the permutation (length 0 to n) is processed in the given order by machine 1, the second part in the given order by machine 2.

If we add a third machine, then we can divide the jobs previously assigned to machine 2 among the machines 2 and 3. There are $\binom{n-k_1}{k_2}$ ways to divide the jobs among these machines, for a given k_2. One has to sum over the possible k_2's. For the number of permutations, we have to substitute the permutations in machine 2, $(n-k_1)!$, by the permutations in the machines 2 and 3: $(k_2)!(n-k_1-k_2)!$. This leads to Equation 4.5:

$$\begin{gathered} card(\Phi_3) = \sum_{k_1=0}^{n} \Big(\tbinom{n}{k_1} k_1! \sum_{k_2=0}^{n-k_1} \Big(\tbinom{n-k_1}{k_2} k_2!(n-k_1-k_2)! \Big) \Big) = \\ \sum_{k_1=0}^{n} \Big(\sum_{k_2=0}^{n-k_1} \Big(\tbinom{n}{k_1} k_1! \tbinom{n-k_1}{k_2} k_2!(n-k_1-k_2)! \Big) \Big) = \\ \sum_{k_1=0}^{n} \Big(\sum_{k_2=0}^{n-k_1} \Big(\frac{n!}{k_1!(n-k_1)!} k_1! \frac{(n-k_1)!}{k_2!(n-k_1-k_2)!} k_2!(n-k_1-k_2)! \Big) \Big) = \\ \sum_{k_1=0}^{n} \sum_{k_2=0}^{n-k_1} n! = n! \sum_{k_1=0}^{n} \sum_{k_2=0}^{n-k_1} 1 = \\ n!(n+1)(n+2)/2 \end{gathered} \tag{4.5}$$

This gives a good indication what the formula for m machines should look like. The number of possible solutions is given in Theorem 4.1, where we define the sum of zero elements to be zero (i.e. $\sum_{a=1}^{0} k_a = 0$):

Theorem 4.1. (Parallel machines)

$$card(\phi_m) = n! \sum_{k_1=0}^{n} \sum_{k_2=0}^{n-k_1} \cdots \sum_{k_{m-1}=0}^{n-\sum_{a=1}^{m-2} k_a} 1$$

Proof by induction: For $m = 1$, no sum is involved, such that Theorem 4.1 coincides with Equation 4.3. It is easy to verify with Equations 4.4 and 4.5 respectively that Theorem 4.1 holds for 2 and 3 machines as well.
Let us now assume that Theorem 4.1 is true for m machines. If we add one machine, then we can divide the jobs assigned to machine m among the machines m and $m + 1$. We can cut at any position of any job sequence on machine m

$$card(\phi_{m+1}) = n! \sum_{k_1=0}^{n} \sum_{k_2=0}^{n-k_1} \cdots \sum_{k_{m-1}=0}^{n-\sum_{a=1}^{m-2} k_a} \Bigg(\sum_{k_m=0}^{n-\sum_{a=1}^{m-1} k_a} 1 \Bigg) \tag{4.6}$$

which also follows from Theorem 4.1. This finishes the proof. □

A closer look learns that the expression in Theorem 4.1 can also be written in a more compact way, without the sums. A solution to the problem of assignment and task ordering at stage i can be represented as a string for each machine with the jobs assigned to that machine in the order of processing. All strings can be concatenated, where a zero is introduced between the strings of each two following machines. The result is a "long" string of length $n_i + m_i - 1$, with n_i jobs and $m_i - 1$ zeros. The string for the first stage of the solution in Figure 2.5, for example, is "4 3 0 1 0 2". The number of different long strings that can be made is the number of permutations of all elements, divided by the number of permutation of the zeros, since interchanging two zeros does not yield two different strings. It is easy to verify that the number of partial solutions for stage i is therefore $\frac{(n_i+m_i-1)!}{(m_i-1)!}$.

Theorem 4.2. (Parallel machines, a more compact result)

$$card(\phi_m) = n! \sum_{k_1=0}^{n} \sum_{k_2=0}^{n-k_1} \cdots \sum_{k_{m-1}=0}^{n-\sum_{a=1}^{m-2} k_a} 1 = \frac{(n+m-1)!}{(m-1)!}$$

To formulate the number of solutions for the HFFS we need to introduce some new notation. The number of jobs visiting stage i is denoted $n_i = \sum_{j \in N | i \in F_j} 1$ and the number of jobs assigned to machine l in stage i is represented by the variable k_{il}. We will call the solution space of the HFFL without machine eligibility and precedence constraints Ω. The number of chromosomes, or $card(\Omega)$, given in Theorem 4.3, is easily deducted from Theorem 4.1. The proof is therefore omitted.

Theorem 4.3. (Hybrid flexible flow line)

$$card(\Omega) = \prod_{i=1}^{m} \Big(n_i! \sum_{k_{i1}=0}^{n_i} \cdots \sum_{k_{im_i}=0}^{n_i - \sum_{a=1}^{m_i-2} k_{ia}} 1 \Big) = \prod_{i=1}^{m} \frac{(n_i + m_i - 1)!}{(m_i - 1)!}$$

The number of possible solutions for the example in Figure 4.20, with just five jobs, three stages and five machines, is thus 69,120. For the largest instances we will use, of 100 jobs and 8 stages, each consisting of 4 parallel machines, the number of chromosomes is $(103!/3!)^8$, a number too large to calculate for Microsoft Excel 2003. In Figure 4.22, the increase of the number of possible chromosomes is shown for increasing values of n. Giving the number of feasible solutions is extremely difficult and instance-specific, due to precedence constraints and machine eligibility. To give an idea: precedence relationships cut down the number of feasible solutions faster than in a permutation flowshop. A relationship between two jobs that visit all stages eliminates at least $(1 - 2^m) \cdot 100\%$ of the feasible solutions, as only half of the permutations is possible in each stage.

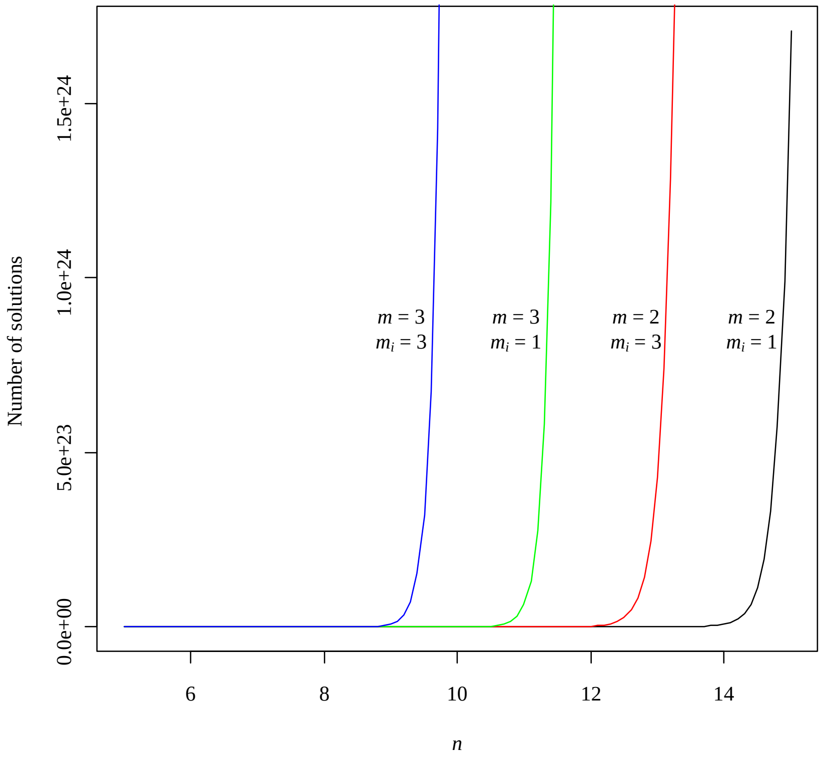


Figure 4.22: Number of possible solutions for different instance sizes; ordered list of tasks for each machine.

m	m_i	n	**Number of solutions**
2	1	5	14,400
		7	25,401,600
		9	1.31682E+11
		11	1.59335E+15
		13	3.87758E+19
		15	1.71001E+24
2	3	5	6,350,400
		7	32,920,473,600
		9	3.98338E+14
		11	9.69395E+18
		13	4.27503E+23
		15	3.16284E+28
3	1	5	1,728,000
		7	1.28024E+11
		9	4.77847E+16
		11	6.36015E+22
		13	2.41458E+29
		15	2.23614E+36
3	3	5	16,003,008,000
		7	5.97309E+15
		9	7.95018E+21
		11	3.01822E+28
		13	2.79517E+35
		15	5.62491E+42

Table 4.9: Number of possible solutions for ordered list of tasks for each machine.

4.4. Dispatching rules

The idea of dispatching rules is one of the oldest in the scheduling field, with the first contributions in the late nineteenth century. We adapt some well known dispatching rules to the HFFL. The jobs are scheduled on a stage-by-stage basis rather than scheduling each job in all stages. This has the advantage that non-permutation solutions (as shown in Figure 2.5) can be obtained. The rules are applied at each moment to the jobs that can be scheduled according to the precedence relationships (i.e. those jobs that are "eligible"). The eligible jobs set is denoted by R. The details of the heuristics are the following:

- Shortest Processing Time (SPT): For each stage, and among eligible jobs that are processed in that stage, the job with the smallest average processing time in the stage is scheduled. The average processing time (APT) for a given job j in a stage $i \in F_j$ is calculated as follows:

  $$APT_{ij} = \frac{\sum\limits_{l \in E_{ij}} p_{ilj}}{|E_{ij}|}$$

  We have chosen to use the average processing time rather than the minimum processing time, as the machine assignment rules schedule the job in many cases to a machine with a processing time higher than the minimum, due to the impact of machine availability, setup times, time lags, etcetera.

- Longest Processing Time (LPT): Same as SPT but the job scheduled is the one with the largest APT.

- Least Work Remaining (LWR): For each stage, and among eligible jobs to be processed in that stage, the job with the smallest sum of average processing times in the remaining stages (including the present stage) is scheduled. Note that we only consider the remaining stages in which the job is processed. Remaining work can be calculated by the following expression: $WR_{ij} = \sum_{a=i}^{m} APT_{aj}$, where $APT_{aj} = 0, \ \forall a \notin F_j$.

- Most Work Remaining (MWR): Same as LWR but the job scheduled is the one with the largest sum of average processing times in the remaining stages.

- Most Work Remaining with Average Setup Times (MWR-AST): It is a refinement of MWR in which we also consider an average of the pending setup times. This average setup is calculated for the present job and all others in R on the remaining stages and eligible machines, resulting in scheduling the job with the highest APT/AST calculation as follows: $APT/AST_{ij} =$

$$\sum_{k=i,k\in F_j}^{m}\left(APT_{kj}+\frac{\sum_{h\in R,h\notin P_j,k\in F_h}\sum_{l\in(E_{kj}\cap E_{kh})}S_{kljh}}{|H_{kj}|}\right)$$

Where H_{kj} is the set containing all possible setups, i.e., $H_{kj} = \{(h,l \in R\times E_{kj})|k\in F_h, h\notin P_j, l\in E_{kh}\}$.

The dispatching rules provide the next job to be processed in the current stage. In addition to that, we need a rule for assigning that specific job to one of its eligible machines at a given stage. As all dispatching rules are very fast, each rule is applied for all machine assignment rules described in Section 4.2, and the best solution is chosen.

4.5. NEH heuristic

The NEH algorithm owes its name to its inventors Nawaz et al. (1983), who proposed the algorithm for the regular flowshop problem. The algorithm is profusely used in the scheduling literature and is well known for its efficiency. Contrary to the previous dispatching rules, the NEH works with a permutation of jobs that are scheduled one by one in all stages, so the schedule is obtained in a job-by-job basis. We adapted the algorithm for the HFFL, but for better understanding we will first explain the original algorithm.

In the first step of the original NEH, jobs are sorted in decreasing total processing time. In this initial order, precedence constraints may be violated, as it only indicates the order of insertion in the final permutation.

The NEH algorithm starts by taking the first two jobs of the initial order, and the schedules associated with the two possible sequences are calculated. The best sequence from the two is used as a basis for inserting the third job from the initial order. This third job is inserted in the three possible positions of the sequence containing the first two jobs, and the best sequence among the three is kept for inserting the fourth job. The process continues until all jobs have been

considered.
We have modified this insertion step of the NEH method in order to take into account the precedence constraints. When a job is to be scheduled, we look for the earliest and latest possible insertion position in the incumbent sequence, i.e., the job cannot be placed before any of its predecessors and no later than any of its successors. To adapt the algorithm to hybrid flowshops, total processing time to determine the initial order has to be replaced by total average processing time ($TAPT_j = \sum_{i \in F_j} APT_{ij}$).

To compare different algorithm settings, we run the NEH algorithm on the set of large instances. We execute the heuristic once for each instance, with each of the machine assignment rules presented in Chapter 4.2, to compare their effectiveness. Note that running the algorithm more than once on an instance makes no sense, as it is a deterministic algorithm and the same solution value would be obtained. As a measure for the results we calculate the relative deviation of the best known solution value in percent, and take the average over the considered set of instances. The best found objective values are given in Appendix C. In Figure 4.23, the results are shown by means of an Analysis of Variance (ANOVA). We first concentrate on the first nine methods that refer to each of the machine assignment rules. The 99% Tukey confidence intervals serve to determine whether two values are statistically significantly different. As quite some intervals do not overlap, the choice of a right machine assignment rule clearly has its importance. We managed to improve over 25% on the most used FAM rule. We can conclude that ECT, EPNS, ECNS and NJSM yield better results on average than the other remaining rules, although the other rules give better results in some occasions.
As the heuristic is quite fast (more details in Section 4.6), all machine assignment rules can be used together in order to get a better solution. The last three methods represent the following implementations, respectively:

- *All*: We execute the heuristic once for each machine assignment rule; the best found solution is stored and determines the final solution value.

- *Job1*: When inserting a job in the partial permutation, all machine assignment rules are applied for this job. The best combination of assignment rule and insertion position determines the new partial schedule. Once the machine assignment rule is chosen for a job, it will not change any more for that job.

- *Job3*: When a job is inserted in the partial permutation, all machine assignment rules are tried for the new job, and for the previous job and the next job in the partial permutation. The best combination determines the new partial schedule.

Although *Job1* and *Job3* seem to be more advanced and more flexible, *All* obtains the best makespan values. Apparently, the best machine assignment rule in a partial permutation is often not adequate in the global solution. Moreover, it is faster than *Job3* and equally fast as *Job1*. We therefore adopt the simple *All* machine assignment rule implementation in the rest of this Ph.D. thesis.

Framinan et al. (2003) showed in their paper on the permutation flowshop

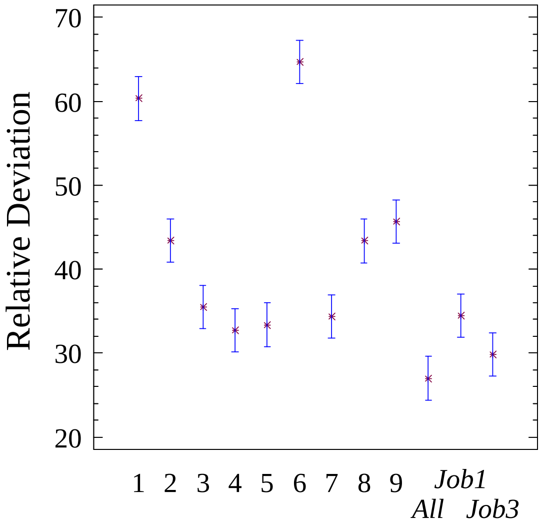


Figure 4.23: Factor means and 99% Tukey confidence intervals for the machine assignment method in NEH; large instances.

problem that changing the initial order can lead to considerable improvements. We have therefore tested various different initial orders for the HFFL. It seems logical that avoiding infeasible partial solutions would improve the results. This can be achieved by inserting the job with highest $TAPT$, among those that have all their predecessors scheduled already; the ready jobs. The heuristic is executed once for each instance with the initial order of the original NEH implementation ($orig$) and once for the initial order respecting the precedence constraints ($ready$). Another ANOVA in Figure 4.24 shows that giving priority to the ready jobs is counterproductive. $ready$ is clearly worse than the $orig$ implementation. This can be explained as follows: Jobs with many predecessors are kept for the end, when the partial solution is closest to the final solution. However, because of the large number of predecessors of these jobs only the last position(s) are feasible and least freedom of choice is given when most information is available.

The opposite is therefore more effective: jobs with many predecessors have little freedom and should be ordered in an early stage of the algorithm. The same holds for jobs with many successors. Less constrained jobs can better be scheduled later on, when the partial solution is more similar to the final one. The best initial order is therefore sorting the jobs by decreasing sum of number of predecessors and successors. In case the sum is equal for various jobs, these

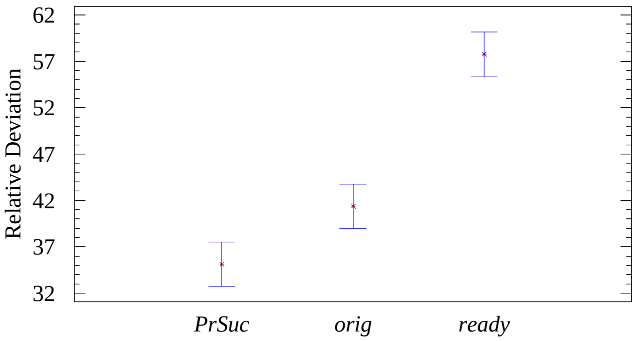


Figure 4.24: Factor means and 99% Tukey confidence intervals for different initial orders in NEH; large instances.

will be ordered among by decreasing $TAPT$. We can join both conditions by calculating the value $PrSuc_j = \sum_{k \in P_j} 1 + \sum_{k \in N | j \in P_k} 1 + \frac{TAPT_j}{\max_{k \in N} TAPT_k}$ for each job and ordering the jobs by decreasing $PrSuc_j$. Figure 4.24 confirms that this initial order is more effective than both other implementations. Note that the three initial orders are equal for instances without precedence constraints; only instances with precedence constraints are therefore regarded in Figure 4.24. The pseudocode for this implementation is given in Algorithm 1.

Algorithm 1: HFFL Adaptation of the NEH Heuristic

```
Input: instance data, assignment rule
Output: permutation $\pi$
begin
    foreach job $j$ in $N$ do
        //calculate total average processing time (TAPT)
        set $index_j$ to $\sum_{i \in F_j} \sum_{l \in E_{ij}} p_{ilj} / |E_{ij}|$;
    set MaxProc to $\max_{j \in N} index_j$;
    foreach job $j$ in $N$ do
        //decimal part takes care of TAPT ranking
        set $index_j$ to $index_j$/MaxProc;
        foreach job $k$ in $N$ do
            derive set of successors $Suc_k$ from the sets of predecessors $P_q$,
            $\forall q \in N \setminus k$;
        //integer part takes care of precedence constraint ranking
        set $index_j$ to $index_j + |P_j| + |Suc_j|$;
    put jobs in array $InsertOrd$ in decreasing $index$ order;
    set $\pi$ to $(InsertOrd(1))$;
    for $k = 2$ to $n$ do
        set $BestMak$ to a high number, e.g., $\sum_{j \in N} \sum_{i \in F_j} \max_{l \in E_{ij}} p_{ilj}$;
        for $q = 1$ to $k$ do
            insert job $InsertOrd[k]$ in position $q$ of $\pi$;
            if no precedence relationships violated in $\pi$ then
                set $PartMak$ to makespan of $\pi$ using assignment rule;
                if $PartMak < BestMak$ then
                    set $BestMak$ to $PartMak$;
                    set $BestPos$ to $q$;
            erase job $InsertOrd[k]$ from position $q$ of $\pi$;
        insert job $InsertOrd[k]$ in position $BestPos$ of $\pi$;
    generate schedule with makespan $BestMak$ from $\pi$ and assignment rule;
    return permutation $\pi$;
end
```

4.6. Conclusions

In this chapter, we have presented several well-known dispatching rules, adapted to the hybrid flexible flow line problem. Furthermore, the NEH heuristic is applied to this particular problem. Various initial permutations for the NEH heuristic are tested and compared. The best one, not common in the literature, first schedules the jobs with many predecessor relationships and leaves the more flexible jobs, with little or no relationships, for the end.

The performance of all presented heuristics is analysed both for the small instances of Chapter 3, of which the optimum is known, and for the large instances described in Section 4.1. In order to get an idea of the needed CPU times, the averages for the large instances are given in Table 4.10. For the NEH heuristic, n full solutions and $2 + 3 + \cdots + (n-1) = (n(n-1)/2) - 1$ partial solutions need to be evaluated for each machine assignment rule, if we do not take into account the precedence constraints. With precedence constraints these numbers decrease. For the dispatching rules, however, only one solution per machine assignment rule needs to be evaluated. Therefore, the NEH heuristic obviously needs much more calculation time than the dispatching rules. Note furthermore that the calculation of the average setup times in the MWRST rule, makes the rule 3 to 4 times slower than the other dispatching rules.

Algorithm	**Average CPU (ms)**
SPT	6.67
LPT	6.18
LWR	6.67
MWR	6.98
MWRST	22.30
NEH	3,925

Table 4.10: Average CPU times for the large instances.

The exact calculation time is not such a big issue for these heuristics, however. All give practically instant solutions for the largest instances. We do not present the calculation times for the small instances, since the times are so short that they can hardly be measured.

The more important issue for these heuristics, is the solution quality. The results for the large instances are shown graphically in Figure 4.25. An important conclusion that can be drawn, is that none of the heuristics closely approximates the best known solutions. Another important point is the fact that the NEH heuristic justifies the extra CPU time by being with distance the best performing heuristic.

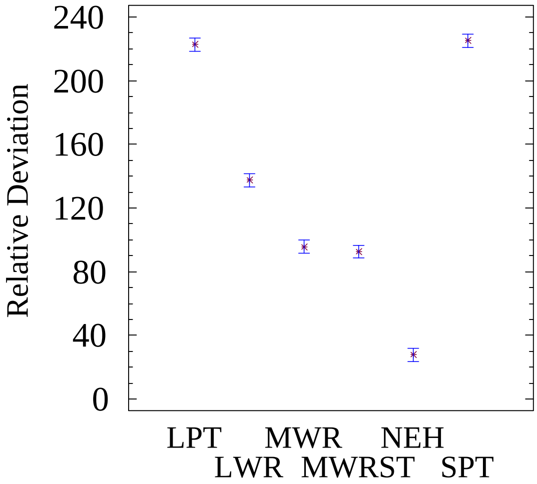


Figure 4.25: Factor means and 99% Tukey confidence intervals for all heuristics; large instances; deviation of best known solution value.

For the large instances, we have no known optimal solutions to compare to. We therefore also give the results for the small instances for which the optimal solution is found in Chapter 3. Recall that the optimum is found for 4,322 instances; 46.9% of the total of 9,216 instances in the set. The average deviation from the optimum is given for all heuristics in Figure 4.26. The conclusions for the large instances still hold. The NEH heuristic has the best average performance and find the optimum in most (2,008) cases. Although all heuristics find the optimal solution in some cases, the average deviation from the optimum is considerable. This indicates the necessity of metaheuristics, that need more time, but in change obtain solutions closer to the optimum. The underlying data for the figures can be found in Appendix A, Tables A.2 and A.3.

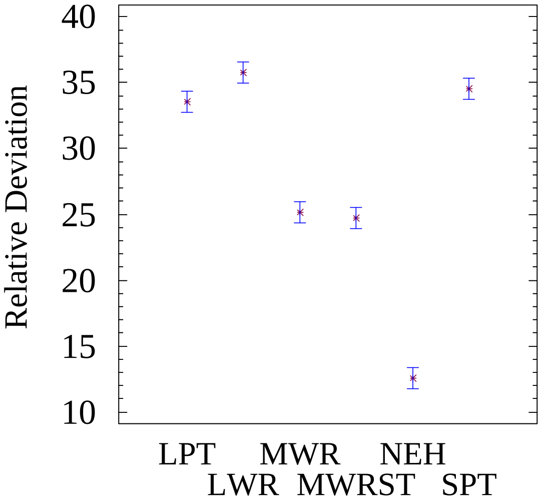


Figure 4.26: Factor means and 99% Tukey confidence intervals for all heuristics; small instances; deviation of the optimum.

CHAPTER 5

GENETIC ALGORITHMS

As pointed out in Section 2.2, Genetic Algorithms are very common in the field of production scheduling. The main advantage of GAs is that the problem characteristics hardly influence the logic of the algorithm, which makes them very flexible.

The idea of genetic algorithms is based on the analogy between the development of good solutions and the development of a species according to Darwin's theories. In this analogy, a solution is an individual and the solution representation is the individual's DNA or chromosome. A set of solutions is a population of individuals and if a population is replaced by a new one, we speak about a next generation. Survival of the fittest and natural selection play a central role in the evolution of a species, a so does it for the solution search.

The main structure of GAs is the following: A set (or population) of initial solutions (or individuals) is generated. Solutions are selected to be combined (or crossed) and changed (or mutated) until a population of the same size is obtained. The new population replaces the old one and the process is repeated, until a stopping criterion is met. See Figure 5.1 for a schematic view.

The book by Holland (1992) is the new version of the famous 1975 work that initiated the use of genetic algorithms in combinatorial optimisation. Another

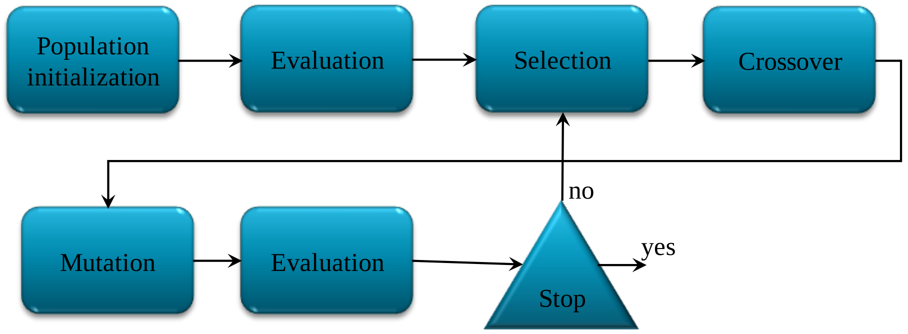


Figure 5.1: A schematic view of a Genetic Algorithm. Constructed from Ruiz (2003).

good reference for more details on genetic algorithms, is the book by Goldberg (1989).

In the following of this chapter, five different genetic algorithms are presented. The first two algorithms employ distinct ways to renew the population for one and the same solution representation. Of the remaining three algorithms, each one has its own solution representation. New operators and procedures to assure feasibility are presented. A computational evaluation is used to compare the genetic algorithms among each other and with methods earlier presented in this Ph.D. thesis.

5.1. BGA

The Basic Genetic Algorithm (BGA) uses the standard solution representation of a single permutation with one machine assignment rule, as explained in Section 4.3.1. An elitism approach is applied to avoid losing the best individuals. The best two individuals of a population are copied directly to the new population, without neither crossover nor mutation.

The different selection methods are the same for all presented GAs. Roulette selection uses a mapping from makespan value to fitness value. We assign to each individual a fitness value $F_x = \max\limits_{y \in Pop} C_{\max}(y) - C_{\max}(x) + 1$, where

Pop is the population of solutions in the GA. An individual x is chosen with probability $F_x/\sum_y F_y$. Note that the addition of 1 in the fitness calculation is necessary to avoid the risk of division by zero in the selection probability. Another advantage is that, although selection of good individuals is more probable, none of the individuals is totally excluded from selection. This avoids early convergence. Random selection is straightforward; all individuals have equal probability. Tournament selection randomly takes a number of individuals. The individual with lowest makespan among them is chosen for crossover.
For the permutation representation many crossover operators are already defined. One-Point Order Crossover (OP) is one of the most basic crossover operators. Two chromosomes are cut each in two parts by choosing one random point. The first part of each chromosome (and the machine assignment rule) is left unchanged for the offspring, whereas the second part is filled with the missing jobs in the order they appear in the other solution. See Figure 5.2.

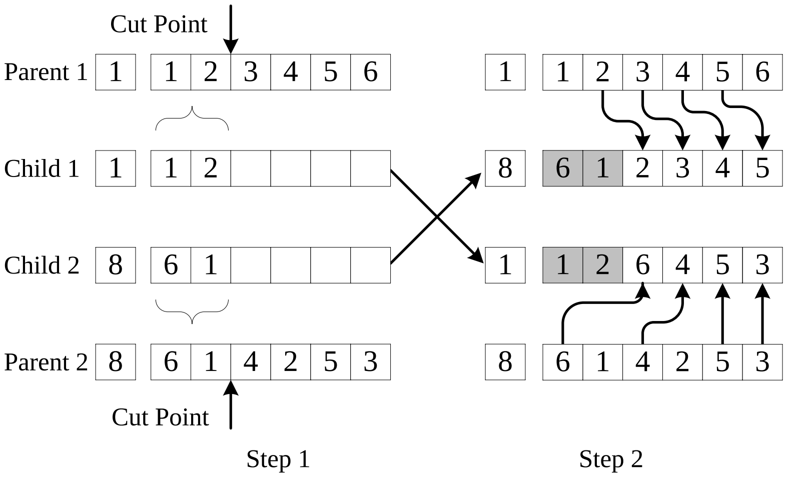


Figure 5.2: One-Point Order crossover operator.

Two-Point Order Crossover (TP) is quite similar. The chromosomes are divided into three parts by two random points. The first and the last part (and the machine assignment rule) remain unchanged, while the middle part is filled with the missing jobs in the order the occur in the other solution. This is illustrated in Figure 5.3.

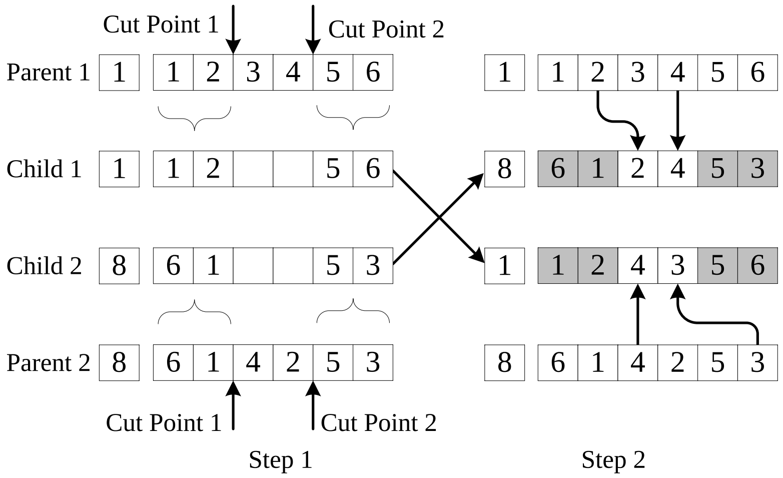


Figure 5.3: Two-Point Order crossover operator.

When applying Uniform Order Based Crossover (UOBX, Figure 5.4), one selects for each location in the child permutation the first unscheduled job of one of the parent solutions, each with equal probability. The machine assignment rule is also take from either of the two parents with equal probability.

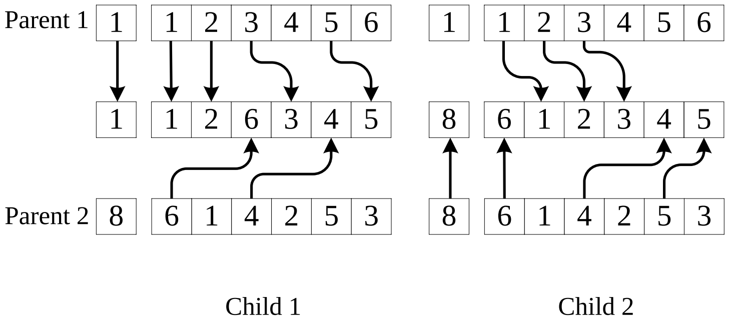
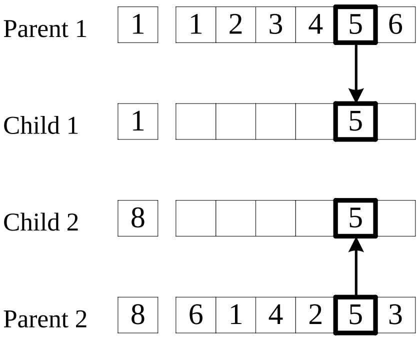


Figure 5.4: Uniform Order Based crossover operator.

Note that feasibility regarding precedence constraints is maintained in all three crossover operators. This is not straightforward. More advanced operators as the Similar Job Order Crossover (SJOX) by Ruiz et al. (2006) might obtain unfeasible offspring from two feasible parents. SJOX copies in Step 1 (Figure 5.5) the jobs that are in the same position in both parents directly to the offspring. Step 2 and 3 in Figure 5.6 fill the rest off the offspring's chromosomes as One-Point Crossover does. In this example, job 4 is a predecessor of job 5, which is respected in both parents but violated in one of the new individuals.



Step 1

Figure 5.5: Similar Job Order crossover operator; Step 1.

Shift Mutation (SM) does not maintain the precedence feasibility by default. A job is excluded from the permutation and inserted in a random position of the partial sequence (see Figure 5.7). To maintain feasibility, the insertion step has to be modified: insertion of the job is done in a random position not before its last predecessor and not after its first successor.

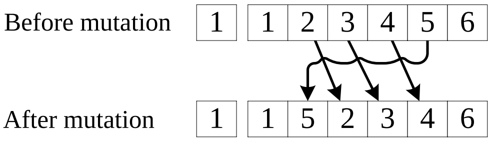


Figure 5.7: Shift Mutation operator.

Position Mutation (PM) consists in exchanging two neighbouring jobs, as done in Figure 5.8. Under precedence constraints this is only done if there is no

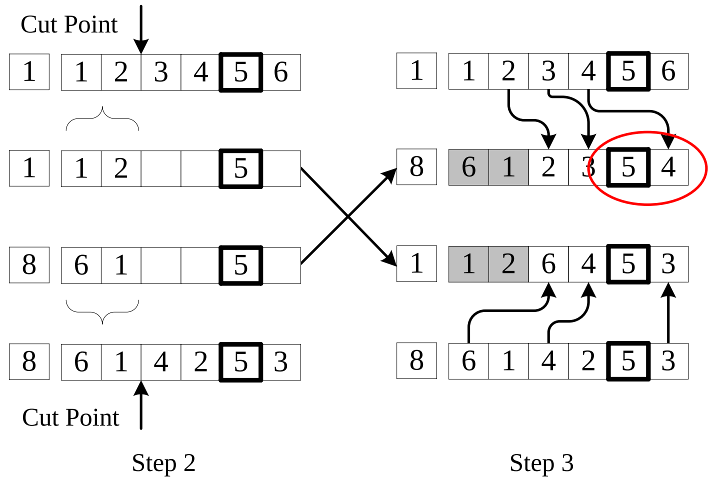


Figure 5.6: Similar Job Order crossover operator; Steps 2 and 3.

precedence relationship among the two. Note that the set of solutions that can be obtained with this mutation is actually a subset of the set by SM, as insertion in the position next to the previous position results in the same change.

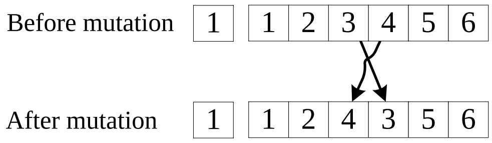


Figure 5.8: Position Mutation operator.

Apart from the mentioned job mutations, the machine assignment rule might also be mutated under a certain probability. In this case one of the other rules is chosen randomly.

The population is initialised with three types of solutions. The solutions of the first type, are generated by using the NEH heuristic explained in Section 4.5 of Chapter 4. Since the resulting solution depends on the used machine assignment rule, for each machine assignment rule that is used in BGA, one NEH solution is inserted in the initial population. The second type of solutions, are NEH solutions that are submitted to a number of random mutations. The solutions of type 1 and type 2 together form 25% of the initial population. The remaining 75% of the individuals of type 3, are generated by randomly sequencing eligible jobs; those jobs whose predecessors have already been sequenced. After the generation of the job sequences, a machine assignment rule is assigned to each individual, to complete the population initialisation.

5.2. SGA

The structure of the Steady-state Genetic Algorithm (SGA) is radically different from the standard structure. New individuals do not fill a new population, but directly replace the worst solutions in the current population. Replacement is only done if the new individual is better than the individual it replaces and if the individual does not exist in the population yet. This replacement scheme results in a higher pressure; elitism is not necessary as only bad individuals are replaced. Ruiz and Maroto (2006) show the efficiency of this generational scheme, compared to regular GA implementations, for hybrid flowshop problems. As the solution representation is the same as for BGA, the same operators can be used.

5.3. SGAR

The steady-state structure described in the foregoing subsection is also used for the Steady-state Genetic Algorithm with multiple Rules (SGAR). This algorithm works with a permutation with a machine assignment rule for each job, as described in Section 4.3.2. The NEH initialisation does not change; we apply the same rule for all jobs. For the remaining initial solutions, different rules in one chromosome are allowed. During crossover and job mutation, the machine assignment rule sticks to the job it belongs to. After job mutation, the assignment rule for each job is changed to any other rule with probability P_{ma}.

5.4. SGAM

The Steady-state Genetic Algorithm with Machine assignments (SGAM) has a more direct solution representation with the machines each job is assigned to in the chromosome. For more details about its representation, we refer to Section 4.3.3.

As in SGAR, the NEH solutions are generated with a single machine assignment rule. A transformation of the final NEH solutions is made, in order to represent them correctly for this algorithm. The machine assignments chosen by the rule are stored in the chromosomes. In other words, the information on the used machine assignment rule is replaced by the information on job-machine combinations. Machine assignments stick to the jobs and only change during the machine assignment mutation, when each job can have one assignment changed with a certain probability.

In some situations, the machine assignment improvement might “stay behind” due to permutation improvements. Machine assignment advantageous for a certain permutation might be less appropriate for another permutation. To detect and correct this phenomenon, we can temporarily make use of the machine assignment rules. With certain (low) probability, the solution is compared to a solution generated with the same permutation, but with the machine assignments at all visited stages generated by a machine assignment rule. If the solution with machine assignment is strictly better, the machines assigned by the rule replace those in the chromosome.

5.5. EGA

While none of the previous genetic algorithms explores the full search space, the Exact representation Genetic Algorithm (EGA) does. Although the chromosome structure is not based on a job permutation, which makes the use of the main operators introduced before impossible, the main structure of the genetic algorithm does not differ from the previously presented genetic algorithms. EGA is also a steady-state GA, which determines the way of introduction of new solutions to be equal to the introduction described for SGA.

With each machine assignment rule, one NEH solution is generated, after which the lists of tasks for each machine are stored in the chromosomes. Earlier mentioned crossover and mutation operators are not applicable for this solution representation.

5.5.1. Specific crossover operators

For crossover, two operators are proposed. Guaranteed Feasibility Crossover (GFX) maintains a list of all available tasks for the assignment to the offspring: tasks whose start times can directly be derived. Among the tasks that are not scheduled yet, the ones that are available and not preceded by other unscheduled tasks in their machine, are stored in a list for this parent. At each iteration, either one of the two parents is chosen and a random task from this list is assigned to the same machine in the child's chromosome. When a complete schedule is obtained for the first child, the process is repeated for the second child. An example is given in Figure 5.9, where job 4 is a predecessor of job 5. At the start (iteration 1) the tasks of job 1, job 3 and job 4 in stage 1 are available and the tasks of job 2 in stage 2. Suppose the toss is won by parent 2. Job 4 at stage 1 and job 2 at stage 2 are candidates. Suppose job 4 wins the toss; the task is scheduled as the first job at machine 1 for the child. At iteration 2, job 1 and job 3 are available in stage 1, job 2 in stage 2 and job 4 in stage 3. Note that job 5 is no candidate for the second position in the first stage, as job 4 has to be scheduled in all stages for job 5 to be available. This procedure is continued until all tasks are scheduled.

This is a new and novel crossover operator, resolving the infeasibility issues caused by the precedence relationships in multiple stages. The implementation requires more than 100 lines of code and the computational complexity can be given by $O(n^2m^2)$, as we have to check the availability of each (unscheduled) task in each iteration, and the number of iterations might be as large as the number of tasks.

Fast Crossover (FX) is similar to GFX, but no list of available tasks is maintained. For both parents only a list is maintained with the first unscheduled task in each machine. One of these tasks is scheduled at the same machine for the child. Note that feasibility of the offspring is not guaranteed. If the offspring

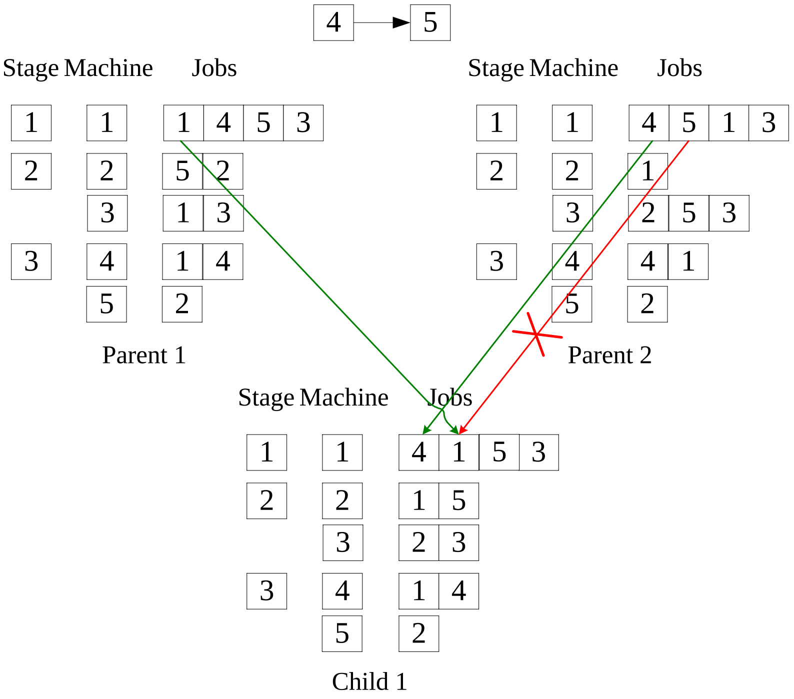


Figure 5.9: Guaranteed Feasibility Crossover operator (GFX).

results to be unfeasible, the unfeasible solution is replaced by an exact copy of the parent. The advantage is that FX crossover is much faster than GFX.

5.5.2. Specific mutation operators

Similarly, two mutations can be applied to the machine assignment arrays in the chromosomes. Both place a task at a new position in any of the eligible machines in the same stage. Fast Mutation (FM) checks the precedence relationships within the new machine. The new position is chosen randomly between the minimum and maximum position, according to the direct precedence relations. This, however does not guarantee global feasibility. We use Figure 5.10 as an example. If we apply mutation on job 5 in the first stage, direct precedence constraints do not impose any position after job 4 (its predecessor). However, placing this task before job 1 leads to an infeasible solution. In this solution, job 1 cannot be started before termination of job 5 in stage 1. But job 4 cannot be finished before finishing job 1. As job 5 cannot be started before job 4 is completed, none of these tasks can be started. This example demonstrates the complexity of the problem we consider.

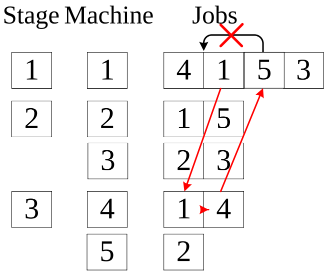


Figure 5.10: Fast Mutation operator.

Guaranteed Feasibility Mutation (GFM) assures that the new schedule is feasible. In case of precedence constraints this implies not only direct predecessors and successors, but also indirect relationships. A task is defined to be indirectly preceding the task that is going to be inserted either if it is followed in its machine by a direct predecessor or an indirect predecessor, or if a task of the same job in a later stage is an indirect predecessor. Analogously, the definition of an indirect successor task is a task that either follows a direct or an indirect successor in the same machine, or if a task of the same job in an earlier stage is an indirect successor. At the moment of inserting a task in a new position, one has to make sure that it is not inserted after any direct or indirect predecessor and not before any direct or indirect successor.

Obviously this implies a larger cost in running time than the Fast Mutation, but no solutions have to be discarded. Due to the recursive character it is very hard to determine the computational complexity.

5.6. Computational Evaluation

For the empirical evaluation of the genetic algorithms, both the set of large instances introduced in Chapter 4 and a subset of the small instances introduced in Chapter 3, are used. We only use a subset of the latter benchmark, since various levels for the controlled factors do not have a significant influence on the hardness of the instances, as shown in Chapter 3. Therefore we do not consider these factors for the remaining experiments; we fix each of these factors at the most realistic level. The distribution of the release dates for machines is fixed at $U[1, 200]$. The distribution of setup times as a percentage of the processing times is $U[75, 125]$ and the probability of the setup to be anticipatory is distributed $U[50, 100]\%$. Time lags are distributed $U[-99, 99]$. In total there are 576 small and 192 large instances. The factor levels are summarised in Table 5.1, and all instances are available at `http://soa.iti.es/problem-instances`.

Factor	**Small instances**	**Large instances**
Number of jobs	5, 7, 9, 11, 13, 15	50, 100
Number of stages	2, 3	4, 8
Number of unrelated parallel machines per stage	1, 3	2, 4
Distribution of the release dates for the machines	$U[1, 200]$	$U[1, 200]$
Probability for a job to skip a stage	0%, 50%	0%, 50%
Probability for a machine to be eligible	50%, 100%	50%, 100%
Distribution of the setup times as a percentage of the processing times	$U[75, 125]$	$U[75, 125]$
Probability for the setup time to be anticipatory	$U[50, 100]\%$	$U[50, 100]\%$
Distribution of the lag times	$U[-99, 99]$	$U[-99, 99]$
Number of directly preceding jobs	0, $U[1, 3]$	0, $U[1, 5]$

Table 5.1: Factors and levels used in the benchmark.

The stopping criterion for all metaheuristics is given by a time limit depending on the size of the instance. The algorithms are stopped after a CPU running time of $n \sum_i m_i \cdot t$ milliseconds, where t is an input parameter. Giving more time to larger instances is a natural way of decoupling the results from the lurking “total CPU time” variable. Otherwise, if worse results are obtained for large instances, it would not be possible to tell if it is because of the limited CPU time or because of the instance size. Constructive heuristics also need more time for larger instances. Besides, with a constant time limit, the allowed CPU time for small instances would be relatively high, which makes it very easy for the algorithms. We are mainly interested in short CPU times, as required in practice as well. The same formula to calculate the allowed processing time is used in Urlings et al. (2010a,b).

All experiments are executed on a Pentium IV computer with a single 3.0 GHz processor and 1 GB of RAM memory. The algorithm is implemented in Delphi with the 2007 compiler and under Windows XP Professional operating system.

5.6.1. Calibrations

In the literature, the values of the algorithm parameters are generally fixed after some quick tests which are not further commented (as Almada-Lobo and James (2010) do for their tabu search), or even fixed without tests (Hasija and Rajendran (2004) is one of many examples that can be given), either random or following other authors. In other cases, as in the paper of Haq et al. (2004), the parameter values are just not stated. The reasons are usually that a large number of parameters is involved, which makes calibration of the algorithm a time- and resource-consuming task. However, we agree with Hooker (1995) that it is an important or even necessary step in the process of algorithm development, since it improves the performance of the algorithms and it helps to understand the functioning of the algorithms and which parts determine their success.

Before calibration of BGA, the algorithm is subjected to some preliminary tests to reduce the number of parameter levels to be tested in the fine-tuning process. Shift Mutation performed clearly better than Position Mutation in preliminary experiments. Two-Point crossover is not outperformed by any of the other crossover operators. This fixes the crossover and the mutation operator.

The comparison of the machine assignment rules for the NEH algorithm in Section 4.5 showed the superiority of ECT, EPNS, ECNS and NJSM. Since a high efficiency is important for the GA, only these four rules are used. The corresponding four NEH solutions (each one only using one rule) seed the initial GA population.

The probability of the mutation changing the machine assignment rule is fixed at 5% per individual.

Crossover probabilities (P_c) of 40% and 60% are tested. Job mutation probabilities (P_{mut}) are either 1% or 2% per job. As commented, the four best assignment rules are used. To test the necessity of various machine assignment rules, a level is added where only EPNS is applied, which implies that the assignment rule mutation probability is 0. Population sizes of 50, 80 and 200 are compared as are the three selection methods roulette, random and tournament among five individuals.

The aforementioned setting results in a total number of parameter combinations of 72. An overview of the tested parameter levels is shown in Table 5.2. Each algorithm is tested five independent times on each given parameter setting and instance. As for t in the stopping time formula, we test t =5 and 25 milliseconds.

Parameter	**BGA & SGA**	**SGAR & SGAM**	**EGA**
Num. rules	4, 1	4, 1	N/A
Pop. size	50, 80, 200	50, 80, 200	50, 80, 200
Select type	Roulette, Random, Tourn.	Roulette, Random, Tourn.	Roulette, Random, Tourn.
Pc	40%, 60%	40%, 60%	40%, 60%
$Pmut$	1%, 2%	1%, 2%	1%, 5%
P_{ma}	5%	1%	N/A
Crossover	Two-Point Order	Two-Point Order	GFX
Mutation	Shift Mutation	Shift Mutation	FM, GFM

Table 5.2: Test values for the algorithm parameters.

For experiment with the smallest number of instances, i.e., for the 192 large instances, this results in 72 parameter settings × 2 t-values × 5 replicates × 192 problem instances = 138,240 data. Because of the high quantity of results, the power of the test is very high and the three ANOVA hypotheses of normality, homoscedasticity and independence of residuals are easily fulfilled. The high power also allows us to use the high 99% Tukey confidence intervals for the ANOVAs. The final parameters for BGA for all instance sets are listed in Table 5.3.

For a more detailed description of the methodology used for the calibration, SGA is used as an example. We will explain the calibration of SGA for the set of large instances. The results of the calibration of other algorithms and instance sets are obtained by applying the same procedure. All ANOVA tables are given in Appendix B. The tested parameter levels are identical to the levels tested for BGA. The algorithm parameter with the highest F-Ratio is the mutation probability, with a value of 2049. This value is much higher than the F-Ratio of the interaction with time parameter t (214), which indicates robustness and makes a separated analysis of the factor unnecessary. Investigation of the factor means plot (see Figure 5.11) shows that a probability of 2% is most suitable, so we fix the parameter at this value. Running a new ANOVA for the remaining factors with the fixed mutation probability, the population size appears to be the most influencing factor with an F-Ratio of 576; again higher than the interaction F-Ratio. A small population size of 50 is worse than populations of 80 or 200 individuals, so we force the population size to larger than 50. In the next ANOVA, the interaction between the population size and the selection type is stronger than any of the factor means. Figure 5.12 shows this interaction. We fix selection at the most advantageous combination: random selection and a population size of 200. Of the remaining factors, only crossover probability is

$m_i = 1$	$m_i = 3$	Large
Roulette Selection Pop. size = 200 $P_c = 60\%^1$ $P_{mut} = 2\%$	Random Selection Pop. size = 200^1 $P_c = 60\%^1$ $P_{mut} = 2\%$ 4 rules^1	Tournament selection Pop. size = 200 $P_c = 60\%$ $P_{mut} = 2\%$ 4 rules^1

[1] not significant in ANOVA

Table 5.3: Final values for the BGA parameters after calibration.

significant (without interaction) and therefore fixed at 60%. Either usage of the four machine assignment rules, or only applying EPNS stays unfixed for the moment due to experiments being inconclusive. After calibration of all GAs for all instance sets we will fix the insignificant parameters at the most convenient levels.

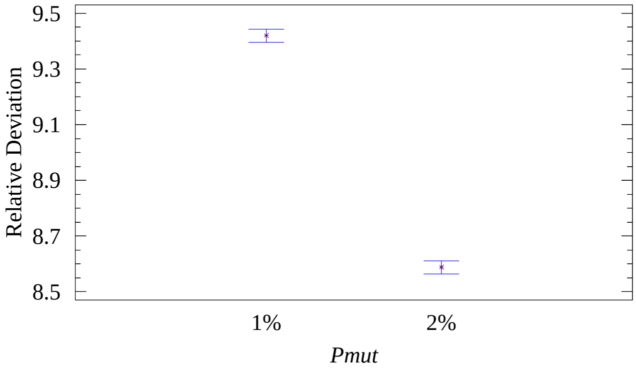


Figure 5.11: Factor means and 99% Tukey confidence intervals for the mutation probability in SGA; large instances.

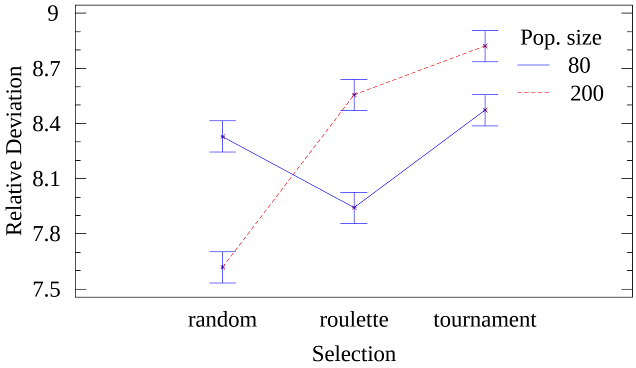


Figure 5.12: Interaction and 99% Tukey confidence intervals between the population size and the selection type in SGA; large instances.

In Table 5.4, the parameter values that are fixed during the calibration, are given.

$m_i = 1$	$m_i = 3$	Large
Random Selection[1] Pop. size = 200 $P_c = 60\%^1$ $P_{mut} = 2\%$	Random Selection Pop. size = 200 $P_c = 60\%^1$ $P_{mut} = 2\%^1$ 4 rules[1]	Random selection Pop. size = 200 $P_c = 60\%$ $P_{mut} = 2\%^2$ 4 rules[1]

[1] not significant in ANOVA, [2] strong interaction with t

Table 5.4: Final values for the SGA algorithm parameters after calibration.

For SGAR, the probability of the mutation changing the machine assignment rule is fixed at 1% per job. All other parameter levels are identical to the levels tested for BGA and SGA. Note that SGAR and SGAM are not calibrated for instances with a single machine per stage, as the algorithms only differ from SGA in machine assignment decisions. Machine assignments are trivial for these instances. The fixed parameter values for the remaining instances are given in Table 5.5.

$m_i = 3$	Large
Random Selection Pop. size = 200[1] $P_c = 60\%^1$ $P_{mut} = 2\%^1$ 4 rules	Random Selection Pop. size = 200 $P_c = 60\%$ $P_{mut} = 2\%$ 4 rules[1]

[1] not significant in ANOVA

Table 5.5: Final values for the SGAR algorithm parameters after calibration.

For SGAM, the probability of the mutation changing the machine assignment rule is fixed at 1% per individual. Recall that the machine assignment rule is only used to compare between the current makespan and the makespan obtained by applying this rule on the same job sequence. Comparison will be done with a probability of 1% per individual.

$m_i = 3$	Large
Random Selection	Roulette Selection
Pop. size = 200	Pop. size = 80
$P_c = 60\%^1$	$P_c = 60\%^1$
$P_{mut} = 2\%^1$	$P_{mut} = 2\%$
4 rules	4 rules

[1] not significant in ANOVA

Table 5.6: Final values for the SGAM algorithm parameters after calibration.

Preliminary tests demonstrate that the Fast Crossover operator for EGA yields worse results than GFX. This is due to the fact that whenever precedence constraints are present, many of the crossed individuals appear to be infeasible. The larger the problem instance, the larger the probability that some restriction is violated. Therefore only a very small part of the generated individuals for the largest problems is used in the continuation of the algorithm. The rest of the crossover actions is simply a waste of time. Fast Crossover is therefore not regarded in the calibration process. Job mutation probabilities (P_{mut}) are either 1% or 5% per machine in the calibration experiments. The remaining levels are left unchanged with respect to the other genetic algorithms in this Chapter. The final parameter settings for the EGA can be found in Table 5.7. Note that this is the only algorithm where a (low) crossover probability of 40% is preferred to a higher probability of 60%. This is easily explained by the more time consuming Guaranteed Feasibility Crossover operator.

5.6.2. Comparison among genetic algorithms

The calibrated GAs are tested with more time as well. Apart from the data obtained with $t = 5$ and $t = 25$, the calibrated algorithms are executed with $t = 125$ milliseconds for the CPU time limit. This results in 1.25 seconds for the smallest and 400 seconds for the largest instances. We first compare the solution quality of the various calibrated genetic algorithms among themselves,

$m_i = 1$	$m_i = 3$	Large
Random Selection	Random Selection	Roulette Selection
Pop. size = 200	Pop. size = 200	Pop. size = 80
$P_c = 40\%$	$P_c = 40\%$	$P_c = 40\%$
$P_{mut} = 5\%$	$P_{mut} = 5\%$	$P_{mut} = 5\%$
GF mutation[1]	GF mutation	GF mutation[2]

[1] not significant in ANOVA, [2] strong interaction with t

Table 5.7: Final values for the EGA algorithm parameters after calibration.

to compare the various solution representations.

We first present an ANOVA for the smallest instances ($m_i = 1$) in the set of instances described at the beginning of this section. The set contains 288 instances, that are actually regular flowshop problems, since parallel machines are found in none of the stages. Recall that only BGA, SGA and EGA are evaluated for this set, as SGAR and SGAM reduce to SGA when no machine assignment decisions have to be made. Each algorithm is executed five times for three different values of t. The ANOVA in Table B.3 shows that stage skipping PF_j is the most important factor over the algorithms, all the instance properties and running time. Stage skipping makes the problem easier, as less tasks are involved. There is no interaction with the algorithms however, which form the second most important factor. Although the exact solution used in EGA is complete (i.e., the optimal solution is reachable), this algorithm is on average far worse than BGA and SGA. The steady state structure gives the SGA a slight, but statistically significant, advantage compared to BGA. Figure 5.13 shows the strongest interaction, which is between the algorithms and the existence of precedence relationships. EGA appears to perform better than the other two algorithms under the restriction of precedence constraints, mainly because the solution space is smaller and EGA.

An ANOVA only for the instances with one machine per stage for which the optimum is guaranteed by the MIP model also confirms the inefficiency of EGA. In this case, since the problem instances are very easy, no difference can be found between BGA and SGA. The data are given in Table A.5.

For $m_i = 3$ the percentage of eligible machines PE_{ij} is the most important

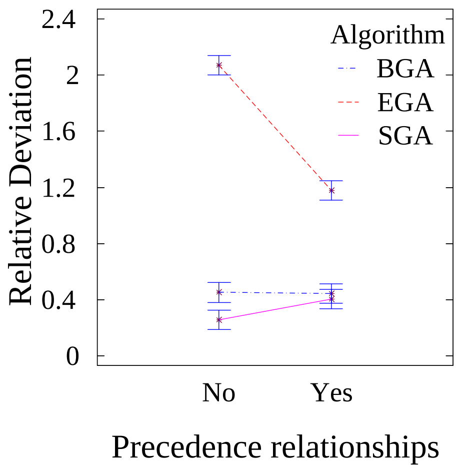


Figure 5.13: Interaction and 99% Tukey confidence intervals for precedence relationships and the algorithm; instances with one machine per stage.

factor. The percentage of eligible machines logically influences the importance of machine assignments. The interaction between the algorithm and PE_{ij} is shown in Figure 5.14. If only half of the machines is eligible the differences in performance are small, but if all machines can process all jobs, the machine assignment rules are proven to be more efficient than incorporating the assignments into the representation. This is a counter-intuitive result since one would expect an exact machine assignment representation to perform better. However, the proposed machine assignment rules outperform the exact representations. In Figure 5.15, one can see that the difference decreases for larger running times.

If we limit the ANOVA to the instances with three machines per stage for which the optimum is known, the results change in a surprising way. As shown in Figure 5.16, the EGA obtains the best results for this instance set. These problem instances are relatively easy, and only the EGA algorithm is able to search the full search space.

For the large instances the most important factor is NP_j, i.e., the number of predecessors. These constraints make the problem harder to solve for the GAs. The interaction of this factor and the GAs can be found in Figure 5.17.

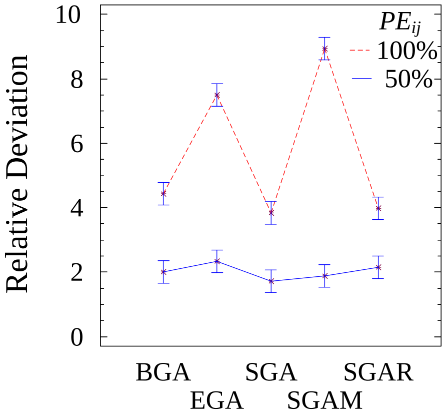


Figure 5.14: Interaction and 99% Tukey confidence intervals for machine eligibility and the algorithm; instances with three machines per stage.

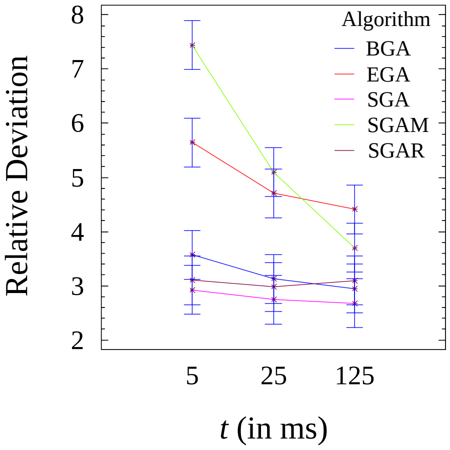


Figure 5.15: Interaction and 99% Tukey confidence intervals for the allowed running time and the algorithm; instances with three machines per stage.

Again, we find here other counter-intuitive results. Presumably, with precedence constraints, less job permutations are feasible and therefore the search space

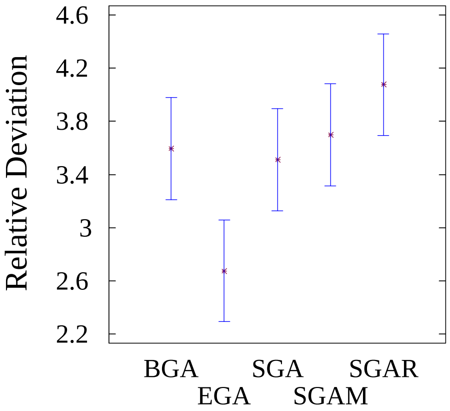


Figure 5.16: Means and 99% Tukey confidence intervals for the genetic algorithms; instances with three machines per stage for which the optimum is known.

becomes smaller. However, the operators of the GAs are much more time consuming when precedence relations are present in order to preserve feasibility and hence the worse results. The influence of this factor is especially large for EGA and SGAM. The complicated EGA operators are especially slow under precedence constraints and SGAM spends more running time on machine assignments and has therefore less time to concentrate on the job sequence, which is more important in the case of precedence constraints.

Also interesting is the performance of each algorithm for the different allowed running times, shown in Figure 5.18 for the large instances. For EGA we see the behavior one would expect; increasing the allowed running time leads to significantly better results. The same holds, in a weaker sense, for SGAM. There is a clear improvement when increasing t from 5ms to 25ms, but increasing further does not pay off. BGA, SGA and SGAR do obtain better solutions with longer running times, but the difference is quite small. This proves that the algorithms with less direct solution representations and therefore smaller search spaces, need less time to explore a large part of the solution space than algorithms based on more verbose representations. Note that not only the relative, but also the absolute differences are large.

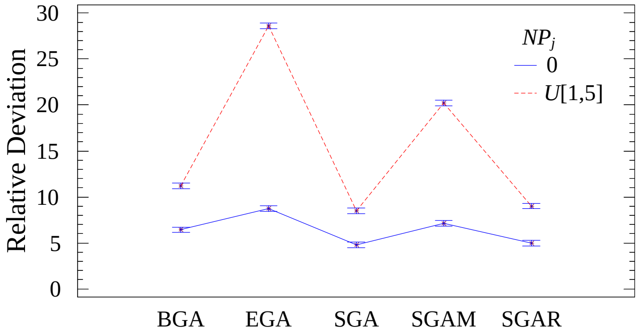


Figure 5.17: Interaction and 99% Tukey confidence intervals for the number of predecessors and the algorithm; large instances.

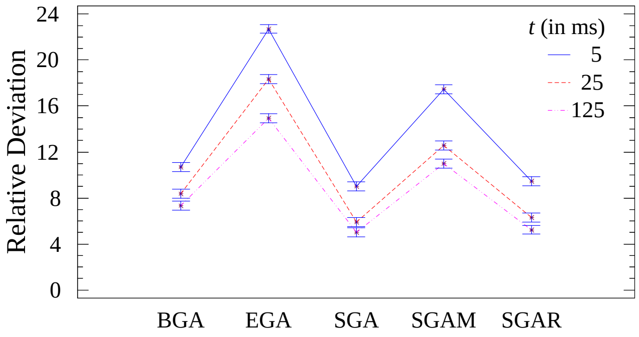


Figure 5.18: Interaction and 99% Tukey confidence intervals for the allowed running time and the algorithm; large instances.

5.6.3. Comparison with other methods

To test the quality of the proposed GAs, we compare them with several other methods. Boyer and Hura (2005) presented a random scheduling (RS) algorithm for sequencing all tasks in a distributed heterogeneous computing environment. They show that their algorithm is less complex than evolutionary algorithms, computes schedules in less time and requires less memory and fewer parameter fine-tuning. We implement an RS algorithm as a benchmark; a minimum all algorithms have to achieve to be considered effective and efficient. If a given algorithm is outperformed by RS, it is either fundamentally wrong or very time consuming. The RS algorithm just produces random solutions until a given termination criterion is met. The solutions are represented by a random feasible job sequence and a machine assignment rule (see Figure 4.11). Note that Random Scheduling (RS) is not calibrated as it does not have any parameters. The machine assignment rule in RS is randomly chosen among ECT, EPNS, ECNS and NJSM at each iteration.

For small instances, the MIP results in Ruiz et al. (2008) are used as a comparison. For both small and larger instances, we also use the results of the dispatching rules and the specific adaptation of the NEH heuristic, as described in Chapter 4. Since those heuristic are very fast, they are run once for each machine assignment rule. The minimum of these results is the final solution value.

In Figure 5.19 the results for the small instances with three machines per stage are plotted for all implemented methods. A more detailed plot of the best methods in Figure 5.20. Note that the Tukey intervals for all GAs and RS are narrower than those of the MIP and heuristics methods. The reason is that for each instance, there is only one result for the MIP and heuristics, as these are deterministic methods. For the GAs and RS there are 15 results (five replicates and three t values). Note furthermore that we used the MIP results with the time limit of 15 minutes; the best results obtained in Ruiz et al. (2008). Some instances were solved to optimality within this time limit, some ended up with a non-optimal solution and in some cases no feasible solution was found within the 15 minutes bound. The shown relative deviation is the average for all cases where an optimal or feasible solution was obtained. The hardest instances are therefore not included in the average MIP deviation. It is clear that the dispatching rules, although improved with the variety of machine assignment rules, do not even approach the performance of any other method. However, one has to take into account that the computation time for the dispatching rules is extremely short (for these instances, less than a millisecond on average).

An even more important result is that all remaining methods give better results

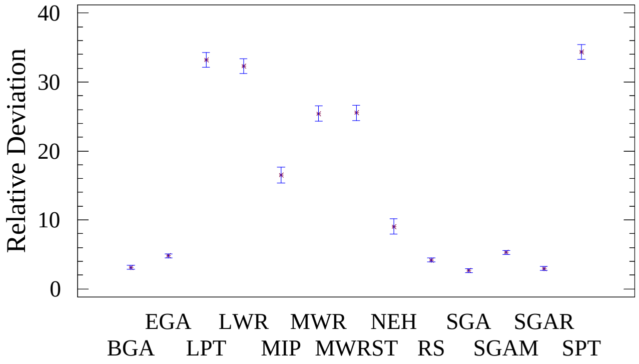


Figure 5.19: Means and 99% Tukey confidence intervals for GAs, MIP and heuristics; small instances with three machines per stage.

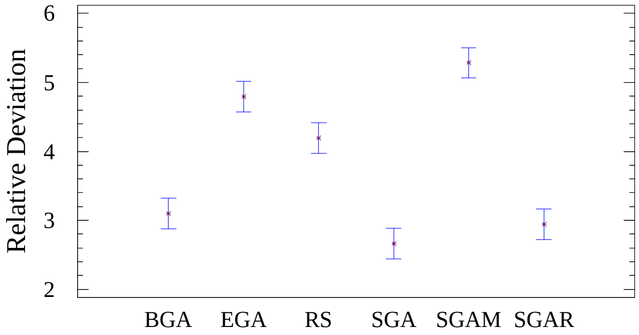


Figure 5.20: Means and 99% Tukey confidence intervals for GAs and RS; small instances with three machines per stage.

than the MIP in less computation time. Not only the needed computation time is problematic for the model; with longer running times the memory capacity becomes a problem, too.

NEH does not reach the solution quality of the GAs and RS, but we have to take into account that this is a fast heuristic, compared to algorithms with longer running times.

Surprisingly, the performance of RS is even better than the performance of

EGA and SGAM. Apparently these two algorithms do not profit of the more verbose solution representations; at least not for the tested running times. The simplicity and speed of RS seems to be an advantage for solving small instances of a complex problem as the one addressed in this Ph.D. thesis.
BGA, SGA and SGAR are the best implemented methods. The steady state structure seems to be advantageous, but the difference is not significant. Introducing for each job an assignment rule does not consume too much running time, but machine assignments are not improved much either.

The results for the small instances with a single machine per stage (Figure 5.21) are similar. Only SGAR and SGAM are left out since no machine assignment is needed in this case. Note that the dispatching rules obtain worse results than for three machines per stage. Since each machine assignment rule yields the same solution, we do not take the best out of several solutions in this case.

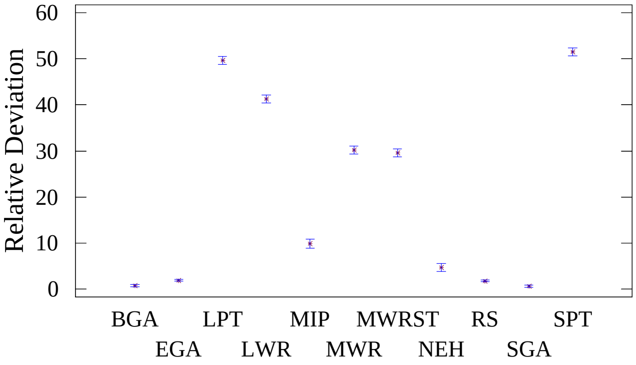


Figure 5.21: Means and 99% Tukey confidence intervals for GAs, MIP and heuristics; small instances with one machine per stage.

Concentrating on the large instances (see Figure 5.22) the differences in average relative percentage deviation are huge and some slight changes in the ranking are noticed. Contrary to the results of the small instances, EGA and SGAM are significantly better than RS. The lack of structure in the solution search of RS starts to play a role when the search space is larger. This method is therefore also outperformed by NEH, which only needs a few seconds for these instances (Table 4.10). As already mentioned in the GA comparison, the operators in EGA are very time consuming. For large instances this is even worse than for the small ones. EGA thus finishes as the worst GA, but still improves the initial NEH solutions significantly, as seen in Figure 5.23.

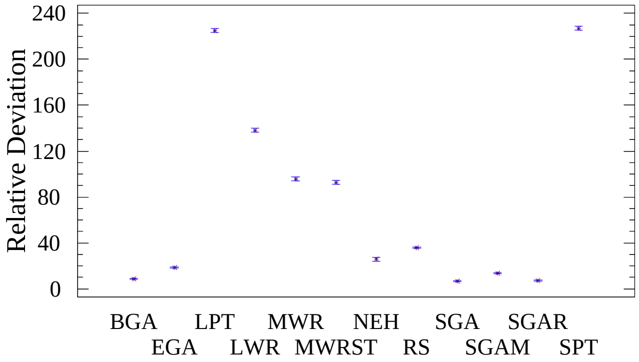


Figure 5.22: Means and 99% Tukey confidence intervals for GAs and heuristics; large instances.

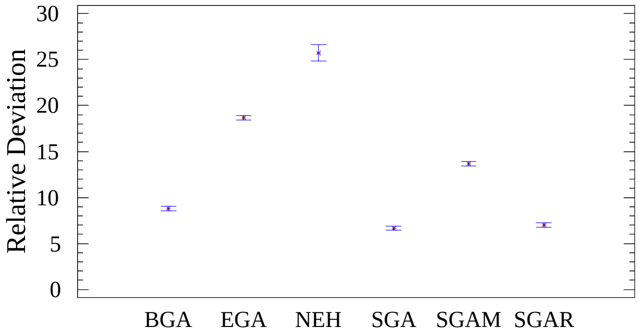


Figure 5.23: Means and 99% Tukey confidence intervals for GAs and NEH; large instances.

5.7. Conclusions

Five genetic algorithms employing several different solution representation schemes have been presented in this Chapter. They are subjected to a extensive calibration. The calibration results in a total CPU time of 4,255 hours and 12 minutes, which means slightly more than 22 days in a cluster of 8 parallel computers. The time investment is worth it, since it teaches us about the behaviour of the algorithms and performance is stronger when the parameter settings are calibrated. Despite, many publications can be found in literature, where no calibration is done or the calibration is not mentioned.

Considering the algorithm parameters, the instance characteristics and the time limit, the algorithm parameters appear least important. This indicates that the algorithms are robust as regards instance characteristics and CPU time.

The research presented in this chapter has lead to the publication of Urlings et al. (2010a).

CHAPTER 6

LOCAL SEARCH ALGORITHMS

To solve combinatorial problems that are too big to be solved to optimality, often a local search method is called into action. In local search we start with a certain solution and try to change this solution in such a way that we get a new solution. The changes we use define the neighbourhood: a function connecting each solution to a subset of the solution set. From every new solution we go to one of its neighbours. A possible definition of local search could thus be a metaheuristic method that aims to optimise an objective function, going from one solution to another solution in its neighbourhood, describing a path through the search space doing so.

The simplest local search algorithm is iterative improvement: we always pick the best neighbour until we get to a local optimum; a solution better than all its neighbours. However, this local optimum might be very far away from the global optimum. The challenge of a local search algorithm is therefore to converge towards better solutions, without getting stuck in a local optimum worse than the global optimum.

6.1. Introduction

Whereas GAs are famous for their flexibility, Local Search (LS) algorithms often obtain better results, especially for relatively simple problems. According to Hoos and Stützle (2004), "...LS methods are surprisingly simple, and the respective algorithms are rather easy to understand, communicate and implement. Yet, these algorithms can often solve computationally hard problems very efficiently and robustly."

A condition for their good performance, however, is the use of speedups or accelerations, which highly depend on the problem. When we consider non-permutation representations, as for example the solution representations used for EGA, the implementation of accelerations is highly complicated and its effectiveness minimal. This is due to the large amount of interconnections between the tasks within a production schedule. Changing a certain task of position does not only change the start and finish time of the tasks of the same job in the following stages and the start and finish times of the tasks after the moved task; the precedence relationships cause changes in possibly all machines. As a result of these drawbacks of direct representations for local search algorithms, all algorithms in this Section work with a single permutation and one machine assignment rule.

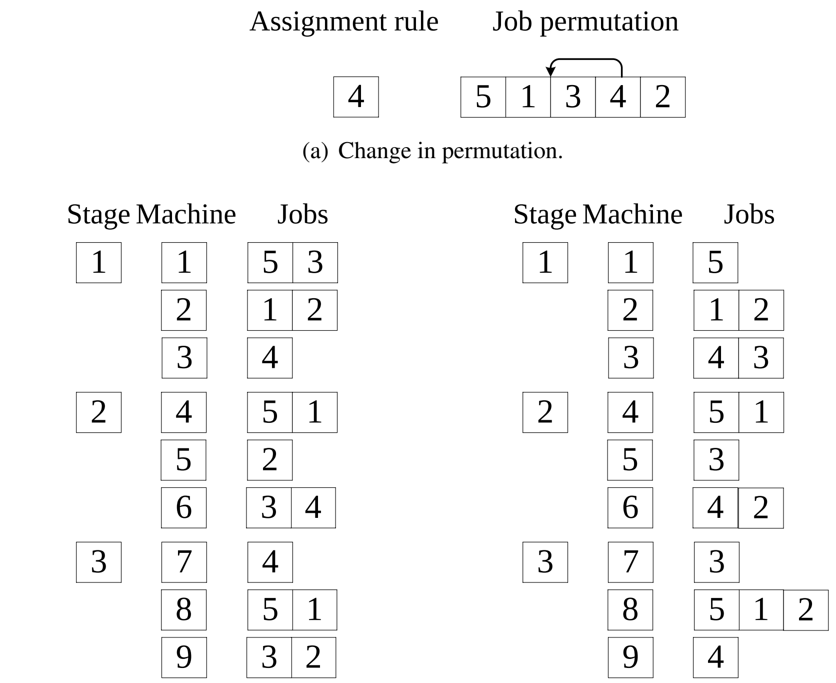
Probably the most important decision in local search design is the definition of the neighbourhood. Large neighbourhoods are powerful, but time consuming. The smallest neighbourhood in our problem is Adjacent Interchange (AI), which consists in interchanging pairs of adjacent jobs in the job permutation. The pair of adjacent jobs whose interchange causes the largest decrease of the makespan, is interchanged. This neighbourhood for permutations is quite standard and also used in Dannenbring (1977). In the case of our problem, because of precedence relationships, the number of neighbours is always $\leq n - 1$.

The Insertion neighbourhood is much more extensive. The neighbourhood is defined as the set of solutions that can be reached by excluding one job from the job sequence and inserting this job in another position within the same sequence. For precedence constraints, the number of reinsertion positions

per job is generally lower than $n - 1$; the job can only be inserted after its last predecessor and before its first successor. Considering the n jobs in a sequence, and disregarding for the moment the precedence constraints, there is a maximum number of neighbours of $n(n - 1) - (n - 1) = (n - 1)^2$, as insertion of the job in position i at position $i - 1$ gives the same neighbour as inserting the job in position $i - 1$ at position i.
The complete search of the latter neighbourhood results in strong local optima. However, too much valuable CPU time is used by the LS and few iterations can be done. To limit the neighbourhood size, one can insert a job a maximum number of positions (b) towards the beginning of the sequence or a maximum number (e) towards the end. We will denote this limited search Insert(b,e). Note that Insert(1,0), Insert(0,1), Insert(1,1) and AI are actually the same neighbourhood and that Insert(n,n) is the unlimited search. Another way to decrease the needed time, is not continuing until reaching a local optimum, but stopping LS after a given number of neighbourhood scans.

As many similar permutations have to be compared, it seems straightforward to implement some accelerations. The faster a local search is, the more searches can be done (or generations made by the GA) per unit of time. However, the complexity of the problem we consider limits the possibilities to accelerate. The accelerations by Taillard (1990), for example, are not applicable to our problem. This type of accelerations does not take into account the case of hybrid flowshops where machine assignments can change due to a change in the job permutation.
What can be done is using the part of the permutation that is unaffected by the movements. Suppose that the jobs in the positions j and $j + 1$ are interchanged. Then, the tasks of the jobs until position $j - 1$ remain unaffected (for $j > 1$). An example is given in Figure 6.1. Job 4 is placed before job 3. This does not affect the start nor the finish times of the previous jobs; job 5 and 1. The rest of the jobs, however, can possibly be assigned to other machines. Note that the stability of the previous jobs only holds for non look-ahead machine assignment rules. If instead of rule 1 (earliest available machine), rule 7 (next job same machine) is applied, then the machine assignments of job 1 are directly

influenced by the exchange between job 3 and 4. In this example job 5 is not affected, however, if stage skipping occurs the assignments of all jobs might change. If jobs skip stages, the next job used for the assignment rule is the next job visiting the stage, which is not necessarily the next job in the permutation.



(a) Change in permutation.

(b) Task order and assignment before job exchange.

(c) Task order and assignment after job exchange.

Figure 6.1: The results of a change in the job permutation.

For calculation of the importance of the accelerations, precedence constraints among jobs complicate the matter considerably. Forgetting about precedence constraints for the moment, it is easy to see that the number of job calculations without accelerations for AI is $(n-1)n$. $n-1$ neighbours should be considered and each of the n jobs is assigned to a machine at all stages it visits. Now we will consider the same case with accelerations. In

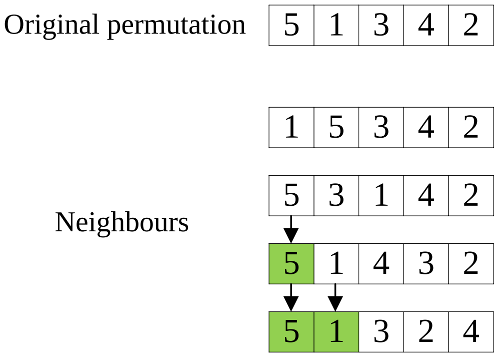


Figure 6.2: Example for $n = 5$ of the adjacent interchange (AI) neighbourhood. Using accelerations, the jobs in green do not have to be recalculated.

order to obtain the first neighbour, we interchange the positions of the first two jobs, assign all n jobs to the machines and calculate their finish times. To get the second neighbour, we put the first job back in its original position and interchange the second and the third job. Since the first job in this neighbour is distinct from the first job in the previous neighbour, all n jobs should be assigned and calculated again. The third neighbour has the same job in the first position as the second neighbour and can use its completion times. A graphical example is given in Figure 6.2. The total number of job calculations is consequently $n + n + (n - 1) + \cdots + 3 = n(n + 1)/2 + n - 3$. For large numbers of n this gets close to 50%. For the insertion neighbourhood comparable results are to be found. The precedence constraints, however, reduce the time advantage. Suppose that, in an example with n jobs (n divisible by 3 without loss of generality), only the job at position $n/3$ and at position $n/2$ can be interchanged with their neighbours. Then $n + (1 + 2n/3) = 5n/3 + 1$ job calculations have to be made, which is a time advantage of only about $1/6$ compared to the regular $2n$ calculations.

To study the influence of the accelerations and the effectiveness of the local search techniques in practice, we run 800 generations of the SGA introduced in Chapter 5, and then apply local search to the population. Local search is repeated until the solutions are not improved anymore. For the 800 generations,

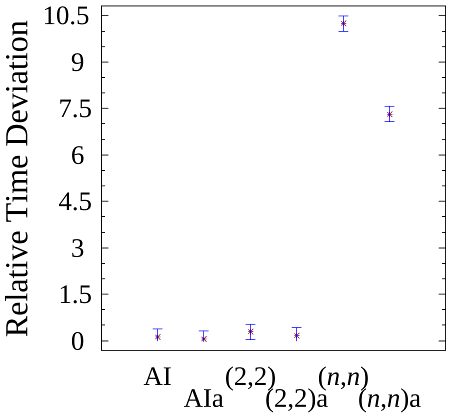


Figure 6.3: Comparison of increment in time (×100%) between local search with (-a) and without accelerations.

computation times range from 30 seconds for the smaller instances, till 15 minutes for the largest instances of 100 jobs and 32 machines (distributed over 8 stages). For each neighbourhood search, three executions are done with accelerations and three without accelerations.

For each execution the relative increase in time is measured. An ANOVA is used to analyse the results. In Figure 6.3, the means and confidence intervals are shown for each neighbourhood with and without accelerations. If the 99% Tukey intervals do not overlap, we can assume a difference with an error probability of 1%. The impact of the accelerations depends on the size of the neighbourhood. For Insert(n,n), an average time saving of about 28.6% is measured. The values for the small neighbourhoods are so small compared to the larger neighbourhoods that no significant difference can be seen. However, in a second ANOVA without Insert(n,n), the accelerations for AI and Insert(2,2) are significant. Consulting Table 6.1, the Insert(2,2) local search with accelerations is about 40.1% faster than the version without accelerations. For AI this is even 46.8%. Note that makespan values remain unchanged.

In preliminary tests several local search implementations are compared. Several design choices have to made. The issues we address here are the

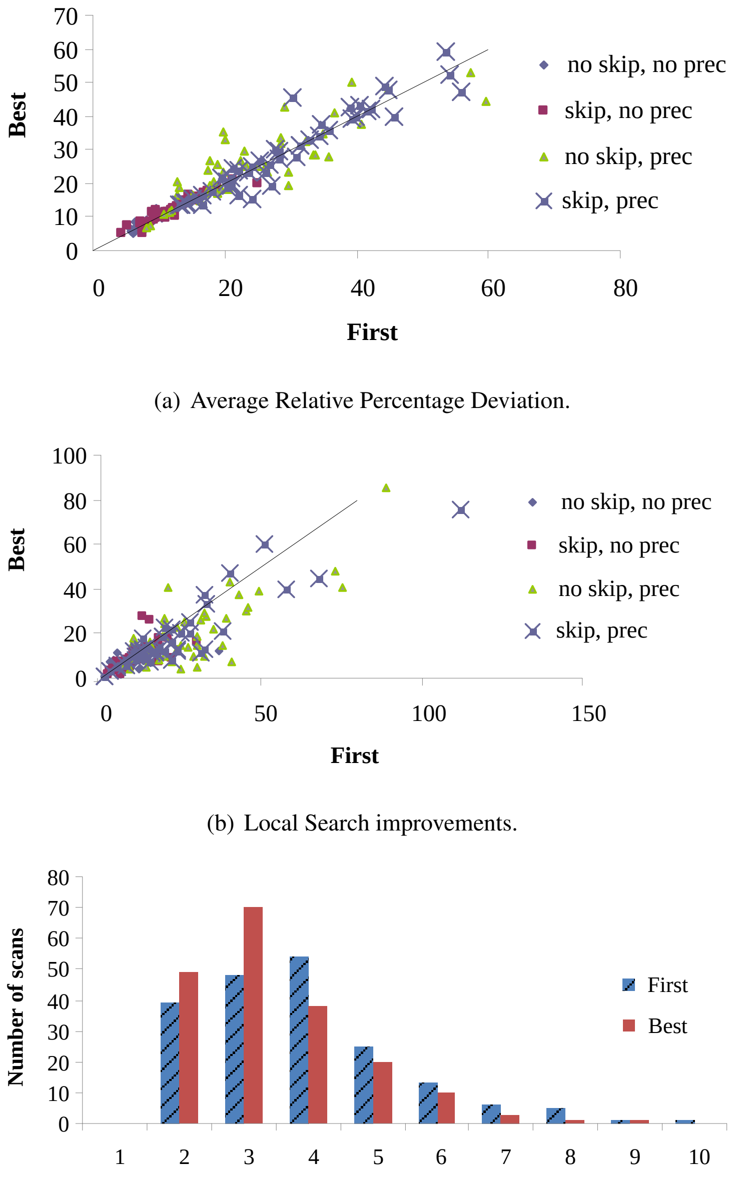
Local Search	**Acceleration**	**Mean**	**Stnd. Error**	**Lower Limit**	**Upper Limit**
AI	Standard	12.35	10.50	-14.70	39.40
	Accelerated	6.57	10.50	-20.48	33.62
	Time reduction	46.81			
Insert(2,2)	Standard	28.48	10.50	1.43	55.53
	Accelerated	17.07	10.50	-9.98	44.12
	Time reduction	40.05			
Insert(n,n)	Standard	1024.04	10.50	997.00	1051.09
	Accelerated	731.57	10.50	704.52	758.62
	Time reduction	28.56			

Table 6.1: Table of means and 99% confidence intervals for the relative percentage time increase.

following:

- When to decide to make a move,
- Whether to apply a preprocess with a smaller neighbourhood or not,
- Where to allow insertion,
- Which order to treat the jobs in.

We start with the first question: when to decide to make a move. The choices we have are first and best improvement: when we check all positions for a job, we can either insert the job at the first position that yields a better solution, or wait until we have seen all positions for this job and insert it at the best position. In Figure 6.4 the two options are compared, applying local search with insertion neighbourhood on a NEH solution for each large instance. Different symbols are used for instances with different characteristics, such that the influence of stage skipping and precedence constraints can be seen. Points at the diagonal 45 degree line indicate instances where first and best improvement have the same behaviour. If first improvement obtains a better makespan value for an instance than best improvement, a dot appears above the 45 degree line in Figure 6.4(a). In Figure 6.4(b) the number of successful local search operations is compared; the number of actually performed insertions. Figure 6.4(c) shows a histogram of the number of neighbourhood scans when reaching a local optimum. From these last two Figures, we can conclude that best improvement needs less local search steps and less neighbourhood scans. This enables us to limit the number of neighbourhood scans in a later stage of the calibration. We will therefore continue with best improvement.



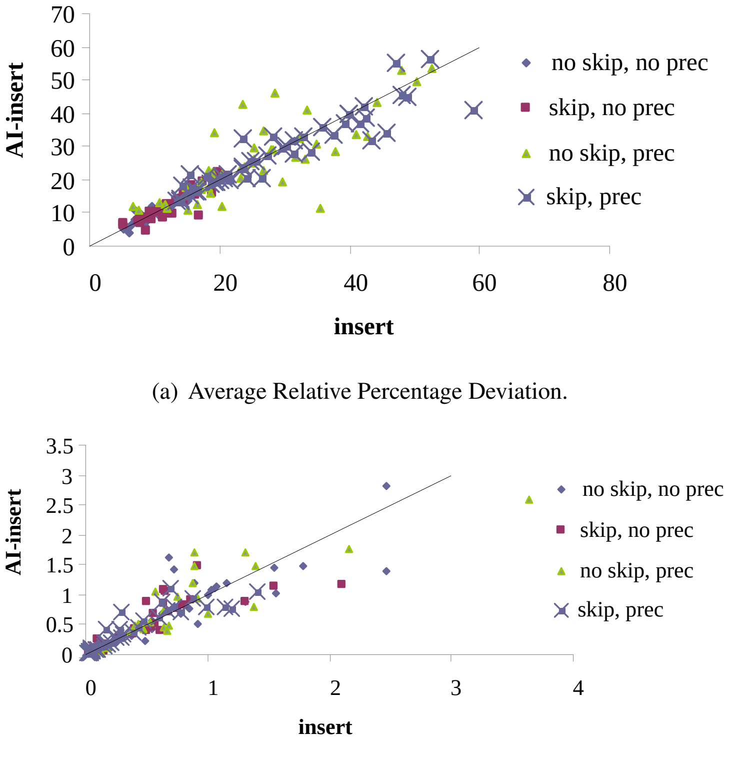
(a) Average Relative Percentage Deviation.

(b) Local Search improvements.

(c) Neighbourhood scans.

Figure 6.4: Comparison between first and best improvement.

Having the first issue solved, we consider the next choice; whether to apply a preprocess or not. To be more precise, the question is either to begin directly searching the insertion neighbourhood or first get a local optimum for the adjacent interchange neighbourhood and then change to insertion neighbourhood. Figure 6.5 shows the differences in a similar way as Figure 6.4. There are quite some instances where long processing time is needed for insertion local search from the start, while the processing time for these instances is shorter if adjacent interchange has been applied first (Figure 6.5(b)). On basis of this we decide to search the AI to local optimum first, when applying local search.



(a) Average Relative Percentage Deviation.

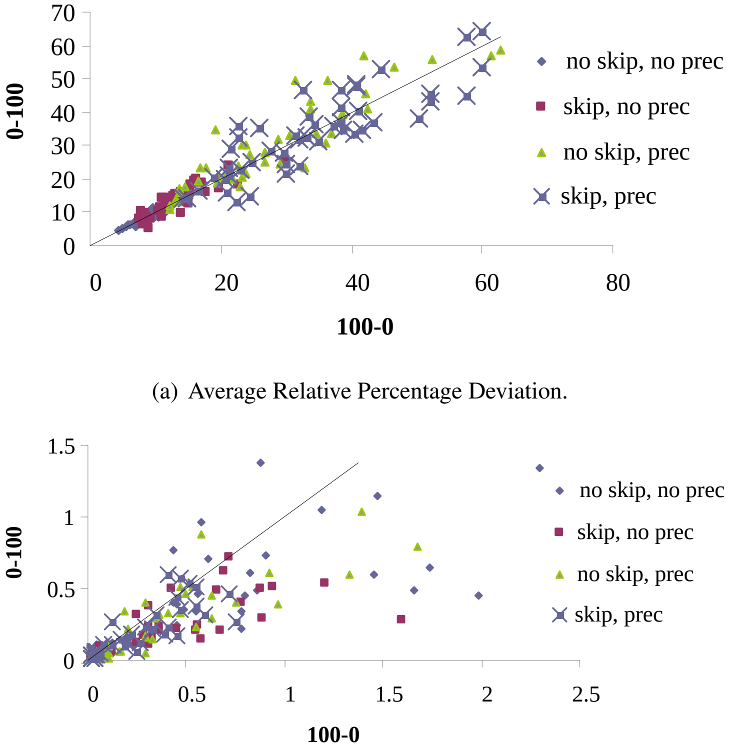
(b) CPU Time (seconds).

Figure 6.5: Comparison between only insertion and adjacent interchange followed by insertion.

The following design decision is where to allow insertion of a job. We are interested in knowing whether only insertions in an earlier position and only insertion in a later position lead to the same results. We test allowing insertion in an earlier position at a maximum distance of 100 positions (which in fact is each earlier position for the instances we use) and allowing insertion in a later at a maximum distance of 0 positions (which means no insertion in later positions). We denote this as “100-0 insertion”. In Figure 6.6 we compare this with the opposite: 0-100 insertion. The latter needs less CPU time on average, as more instances appear below the diagonal in Figure 6.6(b). The explication is simple: the acceleration reuse the information on first part of the permutation, until the first job that has changes position. If we always insert in a later position, the number of unchanged jobs at the start of the permutation is larger, such that more information can be reused.

We now arrive to the last issue: in which order to consider the jobs. The most straightforward implementation starts scanning new positions for the first job and works from the start (or front) of the permutation towards the end (or back) of the permutation. Because of the accelerations we are interested in trying the opposite as well, starting from the back and working towards the front. The comparison is shown in Figure 6.7. The test appears to be useful, as starting from the back of the permutation results in shorter CPU times (Figure 6.7(b)). The reason is the following: The last neighbourhood scan does not have be finished completely. If the last improvement has been made for job j, when reaching job j again without further improvements we know we have reached a local optimum and we can stop the local search. If we start from the back we can skip in this last neighbourhood scan all jobs before job j, for which only little information can be reused (as they are at the beginning of the permutation). Therefore we save more time than when we start from the front and skip the jobs after job j, where more information can be reused.

Summarising the foregoing: Considering CPU time and solution quality, the best implementation starts with a search in the adjacent interchange neigh-

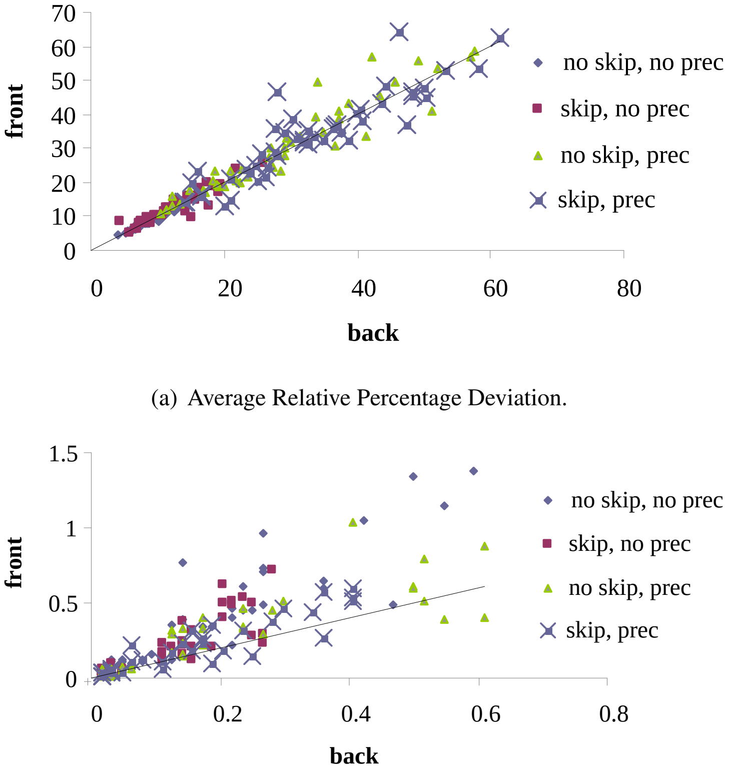


(a) Average Relative Percentage Deviation.

(b) CPU Time (seconds).

Figure 6.6: Comparison between only insertion in earlier positions and only insertion in later positions.

bourhood, until a local optimum is reached. Then the neighbourhood is made larger by allowing insertions of jobs at a larger distance, as insertion showed to be better than interchange in an earlier stage. For each job, the best insertion is performed until no improvements can be made anymore. Starting from the job in the last position results in lower computation times, as most information on the schedules can be reused. The pseudocode of the resulting local search that we will call Best Insertion Reverse Search is given in Algorithm 2.



(a) Average Relative Percentage Deviation.

(b) CPU Time (seconds).

Figure 6.7: Comparison between only insertion in earlier positions and only insertion in later positions.

6.2. Memetic Algorithm

A memetic algorithm (MA) is a GA with a local search on certain individuals at certain moments. We start from the SGA, as this appears to be the most effective GA (see Section 5.7) and as the solution representation is appropriate for a fast and efficient local search. Each time after creating two new individuals in the SGA algorithm, local search is applied with probability pLS. The procedure is not applied to the (possibly poor) new individuals, but

Algorithm 2: Best Insertion Reverse Search

```
Input: instance data, permutation $\pi$, $b$, $e$, $scans_{\max}$
Output: permutation $\pi$
begin
    set $C^*_{\max}$ to current makespan;
    set $scans$ to 0;
    repeat
        set improved to False;
        set $scans$ to $scans + 1$;
        for $j = n$ down to $1$ do
            for $i = \max\{j - b, 1\}$ to $\min\{j + e, n\}$ do
                if $i \neq j$ then
                    insert job $\pi(j)$ in position $i$;
                    calculate $C_{\max}$; if $\pi$ is feasible and $C_{\max} < C^*_{\max}$ then
                        set $C^*_{\max}$ to $C_{\max}$;
                        set $i^*$ to $i$;
                        set improved to True;
                    undo insertion;
                if improved = True then insert job $\pi(j)$ in position $i^*$;
    until improved = False or $scans = scans_{\max}$ ;
    return $\pi$;
end
```

to an already accepted individual in the population.

We apply local search with a adjacent interchange neighbourhood to one of the individuals with the best makespan. If no improvement is made, in the next iteration AI-search is applied to one of the individuals with the second-best makespan value. When all makespans have had one individual AI-investigated, the investigated individual with lowest makespan is taken for LS with a maximum insertion distance of two positions. Once all makespan values have had their individual searched for this neighbourhood, the full insertion neighbourhood is scanned for each job for the same individuals in the same order. When all jobs have been investigated, we know that the individuals are local optima for all implemented LS neighbourhoods and we start investigating the remaining individuals of the population, with makespans equal to the locally optimal individuals.

The number of individuals in the population with the same makespan plays an important role in this procedure. G_K represents the set of individuals in the current population with makespan value K. Because of the indirect solution representation, seemingly different individuals with different chromosomes might in fact represent the same solution. In order to avoid to have too many identical solutions in the population, a new individual is only accepted if it is better than the worst individual in the population and if the permutation does not exist in the population yet and if the number of individuals with this same makespan does not exceed a given number $max\#sol$. Pseudocode for the MA is given in Algorithm 3.

Algorithm 3: Memetic Algorithm

```
Input: instance data, $P_c, P_{mut}, P_{ma}, P_{LS}, max\#sol$
Output: schedule
begin
    initialise population pop;
    initialise $index_i$ with value 0, $\forall i \in pop$;
    while time < max time do
        select two random solutions from population;
        if random < $P_c$ then
            apply two-point order crossover;
        foreach offspring individual $i$ do
            for $j = 1$ to $n$ do
                if random < $P_{mut}$ then
                    perform mutation for the job at position $j$ in individual $i$;
            if random < $P_{ma}$ then
                change machine assignment rule for individual $i$;
            if unique solution AND $sol_i$ < $\max_{j \in pop} sol_j$ AND $card(G_{sol_i})$ < $max\#sol$
            then
                set $index_i$ to 0;
                replace worst individual by individual $i$;
        if random < $P_{LS}$ then
            //get the individuals with highest index of each set $G_{C_{\max}}$
            foreach distinct $C_{\max}$ that occurs in pop do
                define set $I_{C_{\max}}$ of individuals $i$ such that $index_g \leq index_i \leq 2$,
                $\forall g \in G_{C_{\max}}$;
            //get the union of all previously selected individuals
            define set $I = \bigcup I_{C_{\max}}, \forall C_{\max} \in pop$;
            //get the individuals with lowest index of the union
            define set $J$ of individuals $j$ where $index_j \leq index_i, \forall i \in I$;
            //get the individual with lowest makespan of the latter set
            get individual $k$ such that $sol_k \leq sol_j, \forall j \in J$;
            switch the value of $index_k$ do
                case 0 local search in $k$ on insert$_{1,0}$ neighbourhood;
                case 1 local search in $k$ on insert$_{2,2}$ neighbourhood;
                case 2 local search in $k$ on insert$_{n,n}$ neighbourhood;
                if $k$ improved then
                    set $index_k$ to 0;
                else
                    increase $index_k$;
    return schedule of individual $i$ such that $sol_i \leq sol_j, \forall j \in pop$;
end
```

Preliminary tests show that it is far more effective to apply local search in the Memetic Algorithm only in the iterations after half of the allowed CPU time, than to do so directly from the start. It seems plausible to allow the SGA to carry out the initial coarse search which is in turn also faster than with LS.
To test the configuration, we compare the local search probability pLS equal to 0, 10% and 100% and the maximum number of individuals with the same solution value $max\#sol$ equal to 1, 15 and 200 (total population). The algorithm is executed five times for each combination and for each instance. However, we now only concentrate on the large instances. These are the hardest and therefore most important instances. We define termination criterion parameter t to be 25 milliseconds. This corresponds to 80 seconds for the largest instances of 100 jobs and 32 machines. In Figure 6.8 the interactions and the 99% Tukey confidence intervals are shown. The best results are obtained for pLS $=10\%$. With respect to 100%, a smaller part of the running time is consumed and the genetic algorithm has more power. A maximum number of 15 individuals with the same makespan is the best tested configuration.

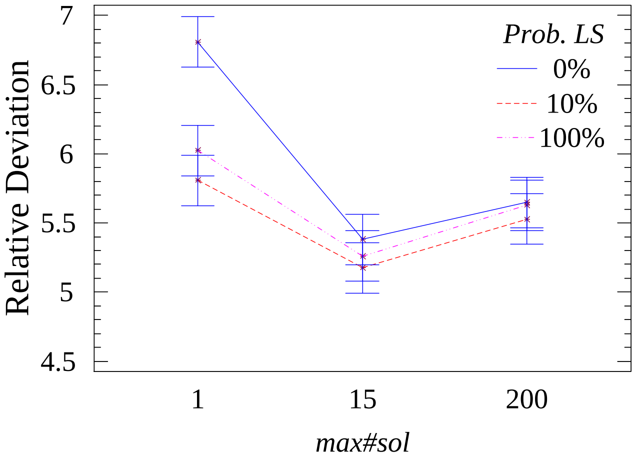


Figure 6.8: Interaction and 99% Tukey confidence intervals for the local search probability and $max\#sol$ in MA; large instances.

6.3. Iterated Local Search

ILS algorithms can be found in many different fields. Stützle (1998) uses it to optimise the permutation flowshop problem, den Besten et al. (2001) apply it to the single machine total weighted tardiness problem and Stützle (2006) shows another implementation for the quadratic assignment problem. The simplicity of this type of algorithms is their strongest point. They are relatively easy to implement, the number of parameters is low and if the local search is efficient, the performance is typically very good. The algorithm works as follows: An initial solution is chosen and local search is applied to this solution. The main loop of ILS does first a solution perturbation, next applies local search and then decides from which solution to continue. Generally, better solutions are always accepted; for worse solutions, Martin et al. (1991) propose to use the acceptance criterion of simulated annealing. If the new solution is better, it is directly accepted; otherwise it is accepted with a probability of $e^{gap/temp}$, where $temp$ is a temperature parameter and gap the percentage difference between the current and previous solution. The importance of a good calibration of this acceptance criterion is shown in Stützle (1998). Lourenço et al. (2002) can be used as a good guide for ILS.
The adapted NEH heuristic in combination with the FPNS machine assignment rule is used to generate the initial solution. In the ILS calibration, different perturbation possibilities are tested, all based in GA mutations. The permutation is subject to a number $NrPert$ of either random adjacent interchanges (similar to Position Mutation), random inserts (similar to Shift Mutation) or random interchanges. An interchange is defined as placing the job in position a in another position b and placing the job that was in position b in position a. Note that the precedence constraints for both jobs have to be checked. The last two perturbation types can be limited in length, that is how far away a job is moved from its current position. Pseudocode for the ILS implementation is given in Algorithm 4.

We will compare the configurations of full neighbourhood, allowing insertion until 4 positions towards the beginning and 9 towards the end, 9 towards the

Algorithm 4: Iterated Local Search

```
Input: instance data, $b$, $e$, $scans_{\max}$, $pert$
Output: permutation $\pi^{opt}$
begin
    created initial solution with NEH heuristic;
    repeat
        apply local search with insert$_{b,e}$ neighbourhood;
    until time > max time or $scans = scans_{\max}$ ;
    set $C^*_{\max}$ and $C^{opt}_{\max}$ to $C_{\max}$;
    set $\pi^*$ and $\pi^{opt}$ to current permutation $\pi$;
    while time < max time do
        for $i = 1$ to $pert$ do
            choose a random position $j$ in $U[1, n]$;
            insert the job from position $j$ in a random feasible position in
            $U[\max\{j - b, 1\}, \min\{j + e, n\}]$;
        repeat
            apply local search with insert$_{b,e}$ neighbourhood;
        until time >= max time or $scans = scans_{\max}$ ;
        if $C_{\max} <= C^*_{\max}$ or random $< e^{(C^*_{\max} - C_{\max})/temp}$ then
            set $C^*_{\max}$ to $C_{\max}$;
            set $\pi^*$ to current permutation $\pi$;
            if $C_{\max}$ < $C^{opt}_{\max}$ then
                set $C^{opt}_{\max}$ to $C_{\max}$;
                set $\pi^{opt}$ to current permutation $\pi$;
        else
            set current permutation $\pi$ to $\pi^*$;
    return $\pi^{opt}$;
end
```

beginning and 4 towards the end, 4 in both directions and 9 in both directions. All neighbourhood configurations are shown in Figure 6.9.
Another way to speed up local search is to not repeat scanning the neighbourhood until arrival in a local optimum, but stopping after a number of complete neighbourhood scans. We test 2 scans for the full neighbourhood and 3 scans for the restricted neighbourhoods.

The allowed running time parameter *t* is set to 25 milliseconds. An ANOVA shows that exploring the full neighbourhood until reaching a local optimum yields the best results, see Figure 6.10. This same figure shows that forward

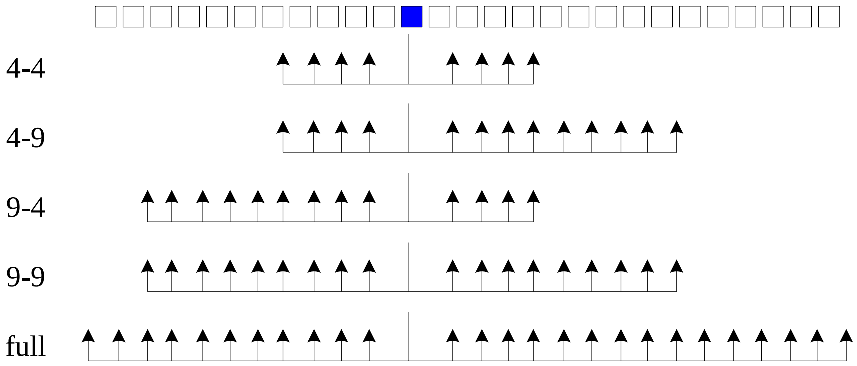


Figure 6.9: Distinct local search insertion neighbourhood restrictions.

insertions should not be treated equally as backward insertions, as a maximum of 4 positions towards the beginning of the sequence and 9 towards the end is clearly better than the opposite. The temperature parameter for the acceptance formula should be set to 1% for the best average results, although lower values are better for the instances without precedence constraints. The best perturbation is done by performing 6 random adjacent interchanges.

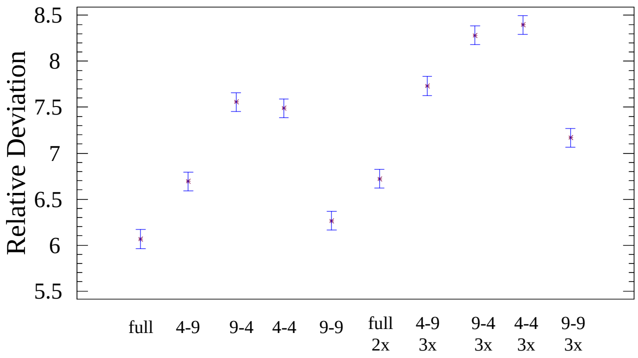


Figure 6.10: Factor means and 99% Tukey confidence intervals for the LS properties for the ILS algorithm; large instances.

6.4. Iterated Greedy

Ruiz and Stützle (2007, 2008) proposed an IG algorithm, which can been seen as a special variant of ILS for the permutation flowshop problem. Each iteration of IG consists of two phases. In the first phase, the solution is partially destructed by removing a number of randomly chosen jobs. In the second phase, the jobs are inserted again in random order. Insertion happens as in the NEH heuristic; a job is inserted at the best position and stays there when the next job arrives for insertion.
Adapted versions of IG can be applied to other scheduling problems. Fanjul Peyró and Ruiz (2010), for example, use IG techniques for an unrelated parallel machines problem. We designed the necessary adaptations in order to apply IG to the HFFL problem we consider. This section gives the details on this algorithm.
The construction phase takes more time than a regular perturbation, so one might expect this algorithm to be slower than the standard ILS implementations. However, because of its greediness the solution after perturbation is expected to be better. Therefore, the local search needs less time and it is generally expected to be more accurate for short running times or extremely large search spaces. Pseudocode for the destruction and construction phase is given in Algorithm 5; the rest of the algorithm is equal to ILS.

For IG, the same neighbourhoods are considered as for ILS. Different from this latter algorithm, a restricted neighbourhood shows a better performance than the full one in Figure 6.11. Note that the confidence intervals are larger then for ILS, as less parameters imply less data in the calibration. Allowing a maximum distance of 4 positions towards the beginning of the sequence and 9 positions towards the end has the lowest average deviation from the best known solution, although an ANOVA shows that there is no significant difference with a maximum of 9 in both directions. Another important result is that a limited number of neighbourhood scans has a negative influence on the results. The average best temperature parameter is $temp = 0.5\%$, although again lower temperatures are preferred for instances without precedence constraints and

Algorithm 5: Destruction and Construction phase of Iterated Greedy

```
Input: instance data, $\pi$, $dest$
Output: permutation $\pi$
begin
    //destruction
    for $i = 1$ to $dest$ do
        choose a random position $j$ in $U[1, n + 1 - i]$;
        remove the job at position $j$ from $\pi$;
        insert the job in set $R$;
    //construction
    for $i = 1$ to $dest$ do
        choose a random job $j$ from set $R$;
        find earliest feasible position $\min$ in $\pi$ for job $j$;
        find latest feasible position $\max$ in $\pi$ for job $j$;
        set $C^{-}_{\max}$ to a high number, e.g., $\sum_{j \in N} \sum_{i \in F_j} \max_{l \in E_{ij}} p_{ilj}$;
        for $k = \min$ to $\max$ do
            insert job $j$ in position $k$ of $\pi$;
            if current makespan $C_{\max} < C^{-}_{\max}$ then
                set $C^{-}_{\max}$ to current makespan $C_{\max}$;
                set $k^{-}$ to $k$;
            remove job $j$ from $\pi$;
        insert job $j$ in position $k^{-}$ of $\pi$;
        remove job $j$ from set $R$;
    return $\pi$;
end
```

higher with precedence constraints. Destructing 4 jobs yields the best results, especially when only half of the machines are eligible.

6.5. Computational Evaluation

Let us now compare the local search algorithms with SGA, the best performing GA. We now run all algorithms on the full set of large instances, for $t = 5$, $t = 25$ and $t = 125$. We can see a strong correlation between the eligibility of the machines and the performance of the distinct algorithms. Using a multi-factor ANOVA, Figure 6.12 shows that SGA and MA do better if any machine can be chosen, while the ILS and IG algorithms perform better if each job can be assigned only to a subset of the machines. In SGA and MA a

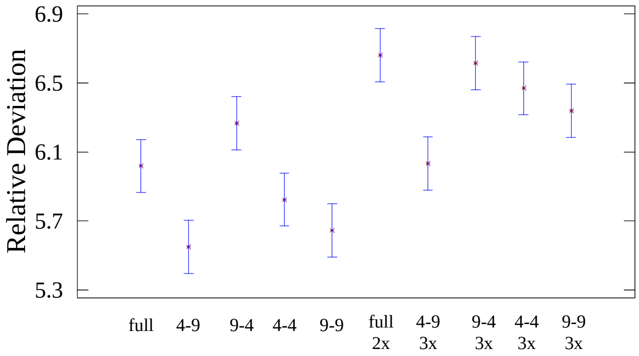


Figure 6.11: Factor means and 99% Tukey confidence intervals for the LS properties for the IG algorithm; large instances.

population of several solutions is used, while in ILS and IG there is only one solution. This might indicate an advantage for population algorithms if the number of eligible machines is high. The most important result, however, is that a subtle, intensively worked local search helps to improve or excel the GA performance, as both MA and IG improve the average results of the SGA and the MA is better or without significant difference in all cases.

The interaction with the allowed running time has a lower F-Ratio, but is interesting to light out as well. From Figure 6.13, we can conclude that the SGA obtains good results for short running times, but that local search is needed to obtain better results if more running time is allowed. Among the local search algorithms, the best average for IG is caused by the relatively good results for short CPU times. For medium or long execution times, no significant difference with MA is observed. A table of means is included in Appendix A.

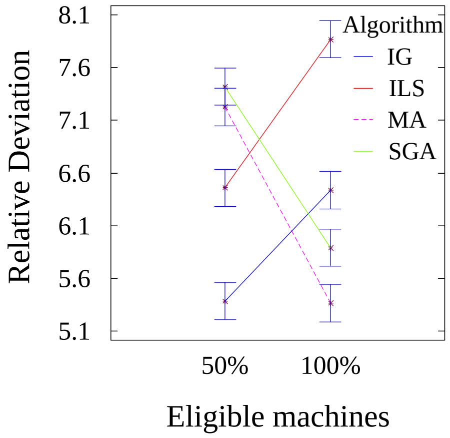


Figure 6.12: Interaction and 99% Tukey confidence intervals for the machine eligibility and the algorithm; large instances.

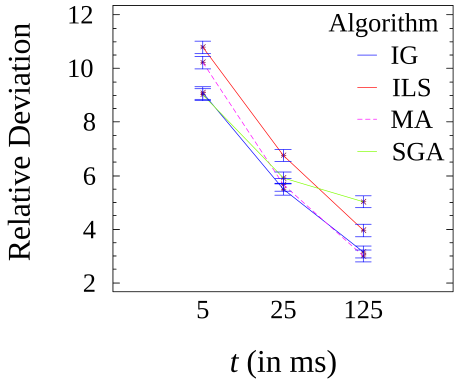


Figure 6.13: Interaction and 99% Tukey confidence intervals for the allowed running time and the algorithm; large instances.

6.6. Conclusions

In this Chapter, three more algorithms are presented. Adaptations of algorithms that were proposed for simpler scheduling problems or even other fields of research are able to find good solutions for this HFFL. However, we have found that the algorithmic issues that arise for these realistic problems are different from those on the very simple problems traditionally studied in scheduling theory. This is most noticeable at the local search component. While local search still plays a role even for these complex problems, its impact appears to be less dominant than for the simpler ones. Therefore, more complex search strategies need to be developed. This is what we will do in Chapter 7. The research in this chapter has lead to the publication of Urlings and Ruiz (2007), where the memetic algorithm is presented.

CHAPTER 7

SHIFTING REPRESENTATION ALGORITHMS

In the previous chapters, each of the algorithms has its own solution representation scheme, with its own disadvantages and limitations. The more indirect representations are efficient, but cover only a small part of the solution space. The optimal solution might be outside this subset of solution space. More verbose representations cover a larger subset or even the whole solution space. Those representations, however, tend to lead to highly inefficient algorithms. The EGA, for example, starts to be too time consuming for instances with more than 10 jobs, as can be seen in Figure 7.1. These observations are the motivation for a couple of novel algorithms with a solution representation that changes over time. In Section 7.1, a genetic algorithm with a changing solution representation is presented. The computational results indicate that the change in representation hardly yields any advantage in the case of this genetic algorithm. Section 7.2 introduces a local search algorithm, where the representation shifts from indirect to direct. The outcome of the empirical analysis is very promising in this case. The scientific results of the research contained in this chapter are summarised in Urlings et al. (2010b).

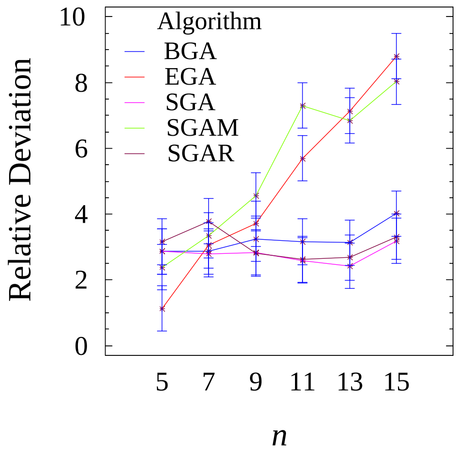


Figure 7.1: Influence of the number of jobs on the results of the genetic algorithms. Interaction and 99% Tukey confidence intervals for the instances with three machines per stage.

7.1. Mixed Genetic Algorithm

The mixed genetic algorithm (MGA) is basically a combination of the SGA and the EGA, both described in detail in Chapter 5. The algorithm starts with the most indirect representation, which is a single job permutation and a machine assignment rule. When part of the CPU time is consumed by the first phase, all solutions in the population are represented with the full solution representation of a ordered task list for each machine. Then the second phase, that searches the full search space, starts, and continues until the stopping criterion is met. Both the first and the second part use the steady state population renewal.

The philosophy of the algorithm is quite straightforward. The first phase serves to get a population of good solutions in an efficient way and the second phase does a more detailed search around these good solutions. This should help to overcome the drawbacks of both solution representations when they are used on their own. A compact pseudocode is given in Algorithm 6.

In order to calibrate the algorithm, we allow different values for the population

Algorithm 6: Mixed Genetic Algorithm

```
Input: instance data, pop
Output: schedule
begin
    initialise population;
    repeat
        update population using SGA;
    until time > 0.5 × max time ;
    convert representation of each solution;
    repeat
        update population using EGA;
    until time > max time ;
    find schedule of best individual;
    return schedule;
end
```

size, the crossover probability and the mutation probability, and we compare several methods for selection. The respective values are: 50, 80, 120 and 200 individuals; 40% and 60%; 1% and 2% per job; random, roulette and tournament selection. The running time is defined by $t \in \{5, 25\}$ms. We do an experiment with full factorial design, where the algorithm is executed 5 times for each parameter combination for each instance.

For the large instances, the most important parameter is the selection method. The parameter with the second most influence, is the population size. We can conclude from their interaction, shown graphically in Figure 7.2, that the algorithm has some trouble maintaining its population diversity. Random selection and a large population (120 individuals) are both needed to handle this problem. When these factors are fixed, only the mutation probability is statistically significant. A 2% probability is most advantageous. All details on the means and the interactions are given in Table 7.1.

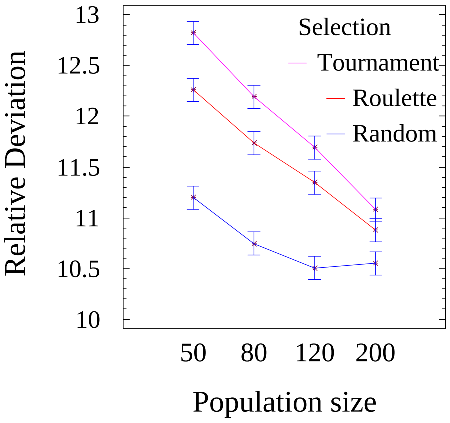


Figure 7.2: Selection method and population size levels for the MGA. Interaction and 99% Tukey confidence intervals for the large instances.

Table 7.1: Calibration for the MGA. Table of means and 99% confidence intervals for the large instances.

Level	2nd Level	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
Population size						
50		23040	12.094	0.0248759	12.0299	12.1581
80		23040	11.5585	0.0248759	11.4944	11.6225
120		23040	11.1815	0.0248759	11.1175	11.2456
200		23040	10.8371	0.0248759	10.7731	10.9012
Selection						
Random		30720	10.7503	0.0215431	10.6948	10.8057
Roulette		30720	11.5558	0.0215431	11.5003	11.6113
Tournament		30720	11.9473	0.0215431	11.8918	12.0027
Crossover probability						
40%		46080	11.4489	0.0175899	11.4035	11.4942
60%		46080	11.3867	0.0175899	11.3414	11.432
Mutation probability						
1%		46080	11.7069	0.0175899	11.6616	11.7522
2%		46080	11.1287	0.0175899	11.0834	11.174
Time parameter t						
5		46080	12.8212	0.0175899	12.7759	12.8665
25		46080	10.0144	0.0175899	9.96909	10.0597
Population size by selection						
50	Random	7680	11.1989	0.0430863	11.0879	11.3099

50	Roulette	7680	12.2611	0.0430863	12.1501	12.3721
50	Tournament	7680	12.822	0.0430863	12.711	12.9329
80	Random	7680	10.7462	0.0430863	10.6352	10.8572
80	Roulette	7680	11.7372	0.0430863	11.6262	11.8482
80	Tournament	7680	12.1919	0.0430863	12.081	12.3029
120	Random	7680	10.5046	0.0430863	10.3936	10.6156
120	Roulette	7680	11.348	0.0430863	11.237	11.459
120	Tournament	7680	11.692	0.0430863	11.581	11.803
200	Random	7680	10.5513	0.0430863	10.4403	10.6623
200	Roulette	7680	10.877	0.0430863	10.766	10.988
200	Tournament	7680	11.0831	0.0430863	10.9721	11.1941
Population size by crossover probability						
50	40%	11520	12.0748	0.0351798	11.9842	12.1655
50	60%	11520	12.1131	0.0351798	12.0225	12.2037
80	40%	11520	11.5624	0.0351798	11.4718	11.653
80	60%	11520	11.5545	0.0351798	11.4639	11.6452
120	40%	11520	11.2433	0.0351798	11.1527	11.3339
120	60%	11520	11.1198	0.0351798	11.0292	11.2104
200	40%	11520	10.9149	0.0351798	10.8243	11.0055
200	60%	11520	10.7594	0.0351798	10.6688	10.85
Population size by mutation probability						
50	1%	11520	12.4675	0.0351798	12.3769	12.5581
50	2%	11520	11.7204	0.0351798	11.6298	11.8111
80	1%	11520	11.8841	0.0351798	11.7935	11.9748
80	2%	11520	11.2328	0.0351798	11.1422	11.3234
120	1%	11520	11.4549	0.0351798	11.3643	11.5455
120	2%	11520	10.9082	0.0351798	10.8175	10.9988
200	1%	11520	11.0209	0.0351798	10.9303	11.1115
200	2%	11520	10.6534	0.0351798	10.5628	10.744
Population size by time parameter t						
50	5	11520	13.2056	0.0351798	13.1149	13.2962
50	25	11520	10.9824	0.0351798	10.8918	11.073
80	5	11520	12.8051	0.0351798	12.7145	12.8957
80	25	11520	10.3118	0.0351798	10.2212	10.4024
120	5	11520	12.6339	0.0351798	12.5433	12.7245
120	25	11520	9.72921	0.0351798	9.63859	9.81982
200	5	11520	12.6401	0.0351798	12.5495	12.7307
200	25	11520	9.03419	0.0351798	8.94358	9.12481
Selection by crossover probability						
Random	40%	15360	10.8223	0.0304666	10.7438	10.9008
Random	60%	15360	10.6782	0.0304666	10.5998	10.7567
Roulette	40%	15360	11.5885	0.0304666	11.51	11.667
Roulette	60%	15360	11.5232	0.0304666	11.4447	11.6016
Tournament	40%	15360	11.9358	0.0304666	11.8573	12.0142
Tournament	60%	15360	11.9587	0.0304666	11.8803	12.0372
Selection by mutation probability						
Random	1%	15360	10.9486	0.0304666	10.8701	11.027

Random	2%	15360	10.552	0.0304666	10.4735	10.6304
Roulette	1%	15360	11.8603	0.0304666	11.7818	11.9388
Roulette	2%	15360	11.2514	0.0304666	11.1729	11.3298
Tournament	1%	15360	12.3117	0.0304666	12.2333	12.3902
Tournament	2%	15360	11.5828	0.0304666	11.5043	11.6613
Selection by time parameter t						
Random	5	15360	12.5817	0.0304666	12.5032	12.6602
Random	25	15360	8.91879	0.0304666	8.84031	8.99727
Roulette	5	15360	12.8056	0.0304666	12.7271	12.8841
Roulette	25	15360	10.3061	0.0304666	10.2276	10.3846
Tournament	5	15360	13.0762	0.0304666	12.9977	13.1547
Tournament	25	15360	10.8183	0.0304666	10.7399	10.8968
Crossover probability by mutation probability						
40%	1%	23040	11.7022	0.0248759	11.6381	11.7663
40%	2%	23040	11.1955	0.0248759	11.1314	11.2596
60%	1%	23040	11.7115	0.0248759	11.6475	11.7756
60%	2%	23040	11.0619	0.0248759	10.9978	11.1259
Crossover probability by time parameter t						
40%	5	23040	12.8853	0.0248759	12.8213	12.9494
40%	25	23040	10.0124	0.0248759	9.94829	10.0764
60%	5	23040	12.757	0.0248759	12.6929	12.8211
60%	25	23040	10.0164	0.0248759	9.95235	10.0805
Mutation probability by time parameter t						
1%	5	23040	13.012	0.0248759	12.9479	13.076
1%	25	23040	10.4018	0.0248759	10.3377	10.4658
2%	5	23040	12.6304	0.0248759	12.5663	12.6944
2%	25	23040	9.62703	0.0248759	9.56295	9.69111

For the small instances with 3 machines per stage, the calibration leads to the same result, as shown in Figure 7.3. Random selection is better than both other options and a population of size 120 is good as well, although it is not significantly different from a population size of 80 or 200. A mutation probability of 2% seems slightly better, but the factor is also not significant for a confidence interval of 99%.

In order to measure the contribution of each of the two phases in the algorithm, a series of tests is done, enabling and disabling each phase. If both stages are disabled, the population is only initialised with NEH and random solutions and the best initial solution is returned. The results for the large instances are shown in Figure 7.4. The figure shows that the SGA phase is very important and that

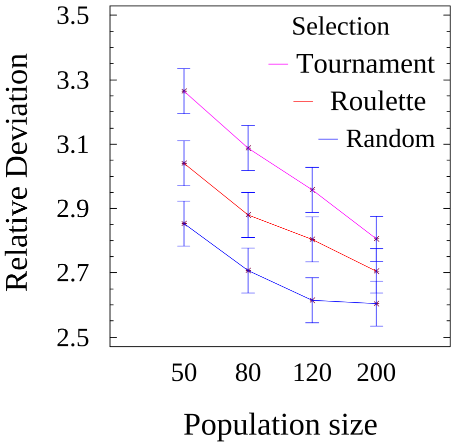


Figure 7.3: Selection method and population size levels for the MGA. Interaction and 99% Tukey confidence intervals for the small instances with three machines per stage.

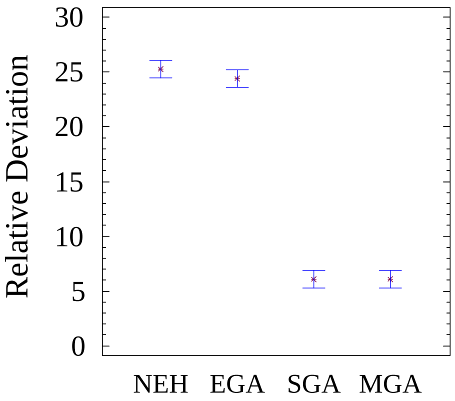


Figure 7.4: Influence of each MGA phase. Means and 99% Tukey confidence intervals for the large instances.

the EGA phase does not have a significant contribution to the results. The exact values of the means and intervals are given in Table 7.2.

SGA	EGA	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
No	No	320	25.291	0.3638	24.354	26.228
No	Yes	320	24.437	0.3638	23.500	25.374
Yes	No	320	6.071	0.3638	5.134	7.008
Yes	Yes	320	6.060	0.3638	5.123	6.997

Table 7.2: Influence of each MGA phase. Table of means and 99% confidence intervals for the large instances.

Another test that shows the contribution of each phase, is a calibration of the moment to change from the SGA phase to the EGA phase. We tested for a subset of the large instances (taking only the first of every three replicates for each instance parameter setting) a change in solution representation after 100, 200, 500 and 800 population generations. As can be observed in Figure 7.5, the later the change in solution representation, the better the results. The data for the figure are given in Table 7.3. This test, together with the previous one, proves that the EGA phase does not have a valuable contribution for the large instances in this algorithm.

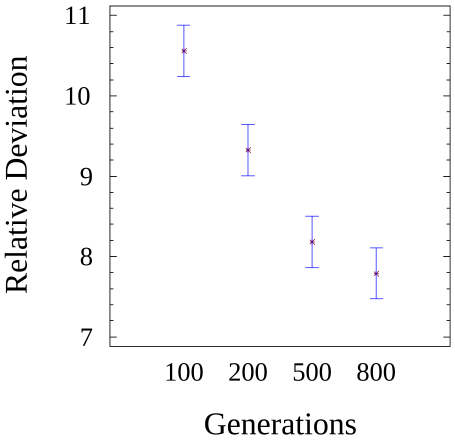


Figure 7.5: Calibration of number of generations in SGA phase. Means and 99% Tukey confidence intervals for a subset of the large instances.

Generations	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
100	640	10.5613	0.144623	10.1888	10.9339
200	640	9.32623	0.144623	8.9537	9.69875
500	640	8.18299	0.144623	7.81047	8.55552
800	640	7.78921	0.144623	7.41668	8.16173

Table 7.3: Calibration of number of generations in SGA phase. Table of means and 99% confidence intervals for a subset of the large instances.

7.2. Shifting Representation Search

Based on the foregoing observations, we developed a new algorithm we named shifting representation search (SRS). This algorithm starts with an indirect solution representation and changes to the full representation when half of the allocated CPU time has elapsed. For local search on the full representation, we do not insert complete jobs, but tasks, where a task is the processing of a job on one machine. The local search neighbourhood is defined as follows: a task is inserted into all possible positions in the task list of the current machine and in the lists of all other eligible machines in the same stage.

In the first phase, we apply the iterated greedy algorithm presented in Chapter 6, as it was shown to be the best performing algorithm with a single job permutation representation. Moreover, IG has a better performance than ILS algorithms in general, which has been shown in Ruiz and Stützle (2007). In the second phase, we perform an iterated local search in the full search space. Perturbation is a random insertion of a task in a feasible position of the task list of another machine. Iterated greedy is not chosen for this representation, since the makespan evaluation of schedules with missing tasks is questionable and both hard and inefficient in terms of implementation.

Having a close look at a schedule, one can see that the makespan of that particular schedule is actually determined by a path of critical tasks. In Figure 7.6, the critical path for the example solution of instance 1 is shown. The sum of the release time, the processing times of the tasks, the setup time and the time lag results in the makespan value of 366. The part that we can influence are the tasks on this critical path: job 4 and 3 at machine 1 and job 3 at machine 4.

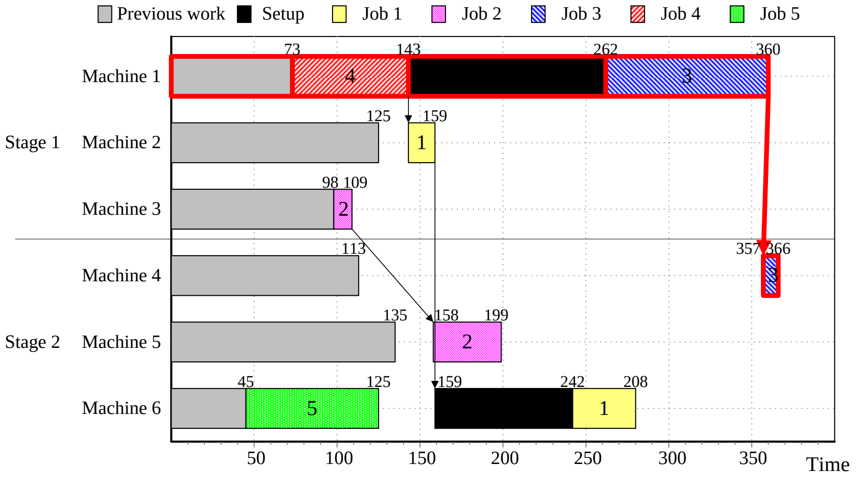


Figure 7.6: The critical path in the earlier shown solution for example instance 1.

If a non critical task a is inserted between tasks b and c, then the makespan can only decrease if task c is critical and if the $setup_{ba} + proc_a + setup_{ac} < setup_{bc}$, such that task c finishes earlier. Since this case is very rare, improving the makespan value by moving a task that is not on the critical path is very improbable. Taking into account that operations on a full representation are fairly time expensive, we only apply local search to tasks on the critical path. Nowicki and Smutnicki (1996) limit the neighbourhood size in a similar way; they apply local search on the critical tasks in a jobshop problem. Since they do not take setup times into account, only moves of tasks on the critical path can improve the solution value.

It is fairly easy to follow the critical path in opposite direction from the end of the schedule; it begins at the task with completion time equal to the makespan value. In the example in Figure 7.6 this is job 3 at machine 4. The previous critical task is either the foregoing task at the same machine, the previous task of the same job, or the last task of one of the predecessors. The previous task in the example is job 3 at machine 1. The critical path might also split when the completion time of two or more tasks is exactly equal to the makespan value, or when two or more tasks determine together the starting time of another task at the critical path. In this case, the makespan value can not be improved by insertion of only one task, thus local search is stopped. A pseudocode of the local search in this second phase is given in Algorithm 7.

Algorithm 7: Local Search on complete representation

```
Input: instance data, schedule
Output: schedule
begin
    get task with completion time = current makespan $C^*_{\max}$;
    repeat
        set improved to False;
        foreach eligible machine $l$ do
            foreach position $i$ at machine $l$ do
                insert task at machine $l$ in position $i$;
                if new schedule feasible and new makespan $C_{\max} < C^*_{\max}$ then
                    set $C^*_{\max}$ to $C_{\max}$;
                    set schedule$^*$ to current schedule;
                    set improved to True;
                undo insertion;
        if improved then
            set current schedule to schedule$^*$;
            get task with completion time = current makespan $C^*_{\max}$;
        else
            //previous task is the task that determines the start time of the current task
            if task has exactly one previous task in the critical path then
                set task to previous task in the critical path;
            else
                break;
    until task does not exists ;
    return current schedule;
end
```

We do not change the local search configuration within the IG algorithm, but we do calibrate the remaining algorithm parameters again. Note that, in interaction with the second phase, a different configuration might be better. We test destruction of 4 and of 6 jobs and temperatures ($t1$) of 0.001, 0.003, 0.01

and 0.03. For the second phase, we consider a number of perturbations of 2 and 4 and for the temperature ($t2$) the same values as in the first phase.

The strongest factor is the number of random insertions done in the perturbation operator in the second phase. Applying only two random movements is clearly more advantageous than applying four. The second most important parameter is the temperature $t1$ for the acceptance criterion in the first phase. Among the four values, 0.01 yields the best result. In Figure 7.7, an ANOVA plot shows the interaction between the perturbation and temperature $t1$. These two parameters fixed, temperature $t2$ can be chosen. A value of 0.001 is significantly better than 0.01 and 0.03, and better in mean but without significant difference compared to 0.003. Although the difference with 0.003 is not significant, we fix $t2$ at 0.001. For the number of excluded jobs in the destruction phase, no significant difference exists between the two levels. We choose a destruction of four jobs, which results in a slightly lower mean. More detailed data can be found in the means Table 7.4 and the ANOVA Table 7.5.

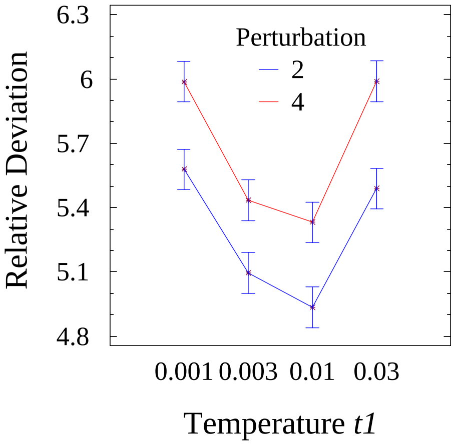


Figure 7.7: Calibration of the SRS algorithm parameters. Acceptance temperature $t1$ and the number of insertions in the perturbation. Means and 99% Tukey confidence intervals for a subset of the large instances.

Level	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
destr					
4	10240	5.51329	0.0188408	5.46475	5.56182
6	10240	5.44669	0.0188408	5.39815	5.49522
pert					
2	10240	5.27402	0.0188408	5.22549	5.32256
4	10240	5.68595	0.0188408	5.63742	5.73448
$t1$					
0.001	5120	5.78333	0.026645	5.7147	5.85197
0.003	5120	5.26444	0.026645	5.19581	5.33308
0.01	5120	5.13323	0.026645	5.06459	5.20186
0.03	5120	5.73894	0.026645	5.67031	5.80757
$t2$					
0.001	5120	5.3179	0.026645	5.24927	5.38653
0.003	5120	5.33819	0.026645	5.26956	5.40682
0.01	5120	5.50756	0.026645	5.43892	5.57619
0.03	5120	5.75629	0.026645	5.68766	5.82493

Table 7.4: Calibration of the SRS algorithm. Table of means and 99% confidence intervals for a subset of the large instances.

Table 7.5: Analysis of Variance for the Average deviation - calibration of the SRS algorithm.

Source	Sum of Squares	Degrees of freedom	Mean Square	F-Ratio	P-Value
Main effects					
A:destr	22.7098	1	22.7098	6.25	0.0124
B:pert	868.76	1	868.76	239.00	0.0000
C:$t1$	1667.98	3	555.992	152.96	0.0000
D:$t2$	632.242	3	210.747	57.98	0.0000
E:Repetition	32.2805	4	8.07012	2.22	0.0642
F:n	10470.9	1	10470.9	2880.62	0.0000
G:m	3985.35	1	3985.35	1096.39	0.0000
H:m_i	19360.2	1	19360.2	5326.11	0.0000
I:F	1855.79	1	1855.79	510.54	0.0000
J:E	16032.8	1	16032.8	4410.72	0.0000
K:P	8545.96	1	8545.96	2351.04	0.0000

Interactions					
AB	0.161326	1	0.161326	0.04	0.8331
AC	188.504	3	62.8346	17.29	0.0000
AD	2.52847	3	0.842824	0.23	0.8742
AE	237.708	4	59.427	16.35	0.0000
AF	2.64401	1	2.64401	0.73	0.3937
AG	143.37	1	143.37	39.44	0.0000
AH	21.0023	1	21.0023	5.78	0.0162
AI	4.09117	1	4.09117	1.13	0.2887
AJ	66.2139	1	66.2139	18.22	0.0000
AK	594.377	1	594.377	163.52	0.0000
BC	17.2062	3	5.7354	1.58	0.1924
BD	94.9563	3	31.6521	8.71	0.0000
BE	0.43235	4	0.108087	0.03	0.9983
BF	3.1641	1	3.1641	0.87	0.3508
BG	0.602913	1	0.602913	0.17	0.6838
BH	51.6389	1	51.6389	14.21	0.0002
BI	255.818	1	255.818	70.38	0.0000
BJ	33.0341	1	33.0341	9.09	0.0026
BK	20.6235	1	20.6235	5.67	0.0172
CD	26.3538	9	2.9282	0.81	0.6111
CE	283.554	12	23.6295	6.50	0.0000
CF	401.357	3	133.786	36.81	0.0000
CG	411.663	3	137.221	37.75	0.0000
CH	214.399	3	71.4665	19.66	0.0000
CI	217.612	3	72.5374	19.96	0.0000
CJ	23.7893	3	7.92975	2.18	0.0880
CK	5861.79	3	1953.93	537.54	0.0000
DE	3.96211	12	0.330176	0.09	1.0000
DF	20.0923	3	6.69742	1.84	0.1370
DG	9.49133	3	3.16378	0.87	0.4556
DH	91.1917	3	30.3972	8.36	0.0000
DI	293.966	3	97.9888	26.96	0.0000
DJ	123.906	3	41.302	11.36	0.0000
DK	4.66555	3	1.55518	0.43	0.7331
EF	19.0227	4	4.75567	1.31	0.2642
EG	153.016	4	38.254	10.52	0.0000
EH	44.5174	4	11.1293	3.06	0.0156
EI	48.7295	4	12.1824	3.35	0.0095
EJ	54.4991	4	13.6248	3.75	0.0047
EK	48.8171	4	12.2043	3.36	0.0094

FG	367.787	1	367.787	101.18	0.0000
FH	260.877	1	260.877	71.77	0.0000
FI	1839.01	1	1839.01	505.92	0.0000
FJ	2651.66	1	2651.66	729.49	0.0000
FK	1635.72	1	1635.72	450.00	0.0000
GH	67.597	1	67.597	18.60	0.0000
GI	2082.6	1	2082.6	572.93	0.0000
GJ	220.261	1	220.261	60.60	0.0000
GK	3942.78	1	3942.78	1084.68	0.0000
HI	9.21912	1	9.21912	2.54	0.1113
HJ	482.579	1	482.579	132.76	0.0000
HK	52.1988	1	52.1988	14.36	0.0002
IJ	236.246	1	236.246	64.99	0.0000
IK	432.927	1	432.927	119.10	0.0000
JK	122.494	1	122.494	33.70	0.0000
Residual	73862.5	20320	3.63497		
Total (corrected)	161836.0	20479			

In order to measure the efficiency of the quite specific local search in the second phase of the algorithm, we have measured the number of local search iterations that can be done within the time limit. Here, we understand as one iteration, one critical path search until the moment of improvement or until stopping because of a critical path split or because of reaching the end of the critical path. The results of this test are shown graphically in Figure 7.8, where distinction is made between problem instances of 50 or 100 jobs and between instances with and without stage skipping. As can be expected, the higher the number of tasks in the problem instance, the longer the critical path and the more time each neighbourhood search takes. Therefore, less local search iterations are done in the case of 100 jobs and if no stages are skipped.

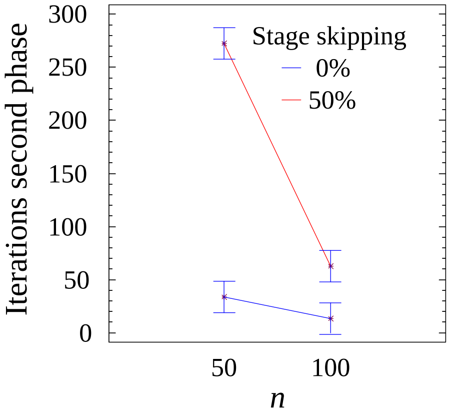


Figure 7.8: Number of local search iterations done in the second phase of the SRS algorithm. Means and 99% Tukey confidence intervals for a subset of the large instances.

7.3. Computational Evaluation

To the comparison at the end of Chapter 6, we can now add the two algorithms that are presented in this Chapter. For the large instances, Figure 7.9 shows the performance of each algorithm for each tested value of t, in milliseconds. We can see that SRS outperforms all other algorithms with a significant difference, regardless of the stopping criterion. MGA is about the worst algorithm for each t value, although the difference with ILS is not statistically significant for 5ms and 25 ms.

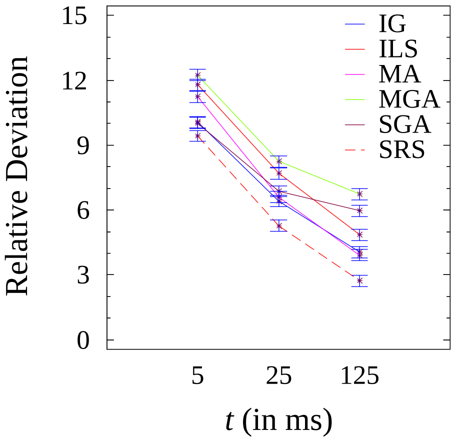


Figure 7.9: Comparison of algorithms. Interaction with the stopping criterion parameter t. Means and 99% Tukey confidence intervals for the large instances.

The most important interaction with the algorithms, is the interaction with the existence of precedence relationships, as we can see in Table 7.6. Recall that half of the instances incorporates precedence constraints and half does not. The results for this interaction are given in Figure 7.10. The instances with precedence constraints are clearly harder than the instances without these constraints. The SRS algorithm yields the best average results thanks to its good behaviour for instances with precedence restrictions. This is due to the fact that local search on the complete representation has a smaller neighbourhood in this case, since many insertions are not allowed. MGA obtains by far the worst results for these instances, since many infeasible solutions are generated in the second phase. For more details on this problem we refer to the analysis of the EGA results in Section 5.7. For the instances without precedence relationships, the differences among all algorithms are smaller, but the ranking does not really change. SRS algorithm is among the best methods and MGA among the worst.

Table 7.6: Analysis of Variance for the Average deviation - comparison of the SRS and the MGA with earlier presented algorithms for the set of large instances.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:Algorithm	17861.8	5	3572.37	426.05	0.0000
B:Repetition	7.3837	4	1.84593	0.22	0.9273
C:n	6377.88	1	6377.88	760.64	0.0000
D:m	20813.0	1	20813.0	2482.19	0.0000
E:m_i	7460.67	1	7460.67	889.77	0.0000
F:F	5340.23	1	5340.23	636.89	0.0000
G:E	7420.86	1	7420.86	885.02	0.0000
H:P	78765.1	1	78765.1	9393.67	0.0000
I:Replicate	19.3316	2	9.66578	1.15	0.3158
J:t	109960.0	2	54979.9	6557.00	0.0000
Interactions					
AB	114.265	20	5.71324	0.68	0.8488
AC	2957.59	5	591.518	70.55	0.0000
AD	809.846	5	161.969	19.32	0.0000
AE	4470.32	5	894.064	106.63	0.0000
AF	2234.32	5	446.864	53.29	0.0000
AG	11879.2	5	2375.84	283.35	0.0000
AH	8656.56	5	1731.31	206.48	0.0000
AI	111.701	10	11.1701	1.33	0.2063
AJ	3588.26	10	358.826	42.79	0.0000
BC	36.9046	4	9.22616	1.10	0.3544
BD	34.2071	4	8.55179	1.02	0.3954
BE	12.9359	4	3.23396	0.39	0.8190
BF	2.89691	4	0.724229	0.09	0.9867
BG	34.1083	4	8.52707	1.02	0.3969
BH	15.5794	4	3.89484	0.46	0.7619
BI	30.4534	8	3.80668	0.45	0.8887
BJ	14.9479	8	1.86848	0.22	0.9870
CD	1148.76	1	1148.76	137.00	0.0000
CE	117.926	1	117.926	14.06	0.0002
CF	930.442	1	930.442	110.97	0.0000
CG	1463.32	1	1463.32	174.52	0.0000
CH	5246.72	1	5246.72	625.73	0.0000
CI	74.9156	2	37.4578	4.47	0.0115

CJ	5220.15	2	2610.07	311.28	0.0000
DE	209.623	1	209.623	25.00	0.0000
DF	2135.8	1	2135.8	254.72	0.0000
DG	3279.85	1	3279.85	391.16	0.0000
DH	15841.6	1	15841.6	1889.29	0.0000
DI	155.872	2	77.9359	9.29	0.0001
DJ	3640.51	2	1820.26	217.09	0.0000
EF	139.434	1	139.434	16.63	0.0000
EG	620.784	1	620.784	74.04	0.0000
EH	454.66	1	454.66	54.22	0.0000
EI	54.7583	2	27.3791	3.27	0.0382
EJ	210.419	2	105.21	12.55	0.0000
FG	14.6231	1	14.6231	1.74	0.1866
FH	74.2158	1	74.2158	8.85	0.0029
FI	403.449	2	201.725	24.06	0.0000
FJ	254.297	2	127.148	15.16	0.0000
GH	11284.8	1	11284.8	1345.85	0.0000
GI	492.396	2	246.198	29.36	0.0000
GJ	1250.23	2	625.113	74.55	0.0000
HI	208.003	2	104.002	12.40	0.0000
HJ	18295.2	2	9147.6	1090.96	0.0000
IJ	27.0545	4	6.76362	0.81	0.5207
Residual	143441.0	17107	8.38492		
Total (corrected)	505721.0	17279			

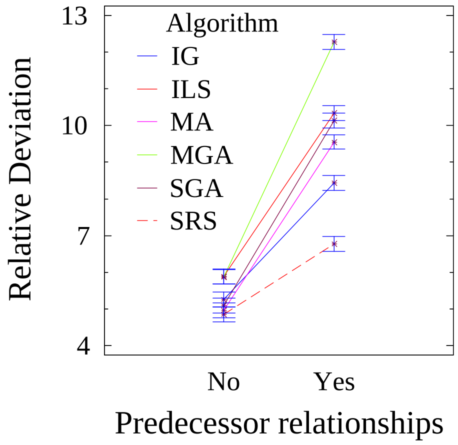


Figure 7.10: Comparison of algorithms. Interaction with the existence of precedence relationships. Means and 99% Tukey confidence intervals for the large instances.

For smaller instances, the comparison leads to a somewhat different ranking. If we consider the set of small instances with three machines per stage (recall that the remaining small instances have one machine per stage and are therefore regular flowline problems), we obtain the ANOVA results shown in Figure 7.11. The SRS algorithm dominates all other methods for this instance set as well, even in a more convincing way. The major difference between the small and the large instances can be observed for the MGA. Whereas the algorithm is not at all effective for the large instances, it is among the best methods after SRS. The most important interaction with the algorithms in this ANOVA, is the percentage of eligible machines. This interaction is shown in Figure 7.12, where the earlier conclusions for the small instances are confirmed.

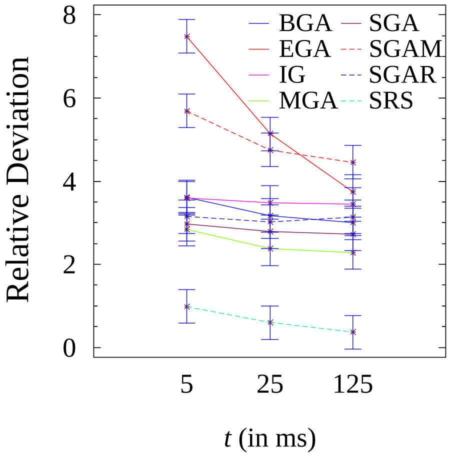


Figure 7.11: Comparison of algorithms. Interaction with the stopping criterion parameter t. Means and 99% Tukey confidence intervals for the small instances with three machines per stage.

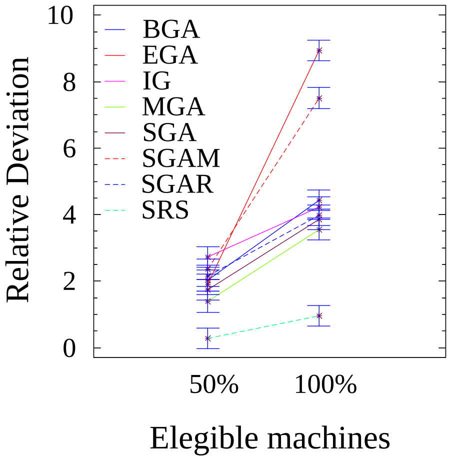


Figure 7.12: Comparison of algorithms. Interaction with the percentage of eligible machines. Means and 99% Tukey confidence intervals for the small instances with three machines per stage.

Since Figure 7.12 is a bit hard to read, we have split the analysis and created Figures 7.13 and 7.14 where each of the instance factors is shown separately.

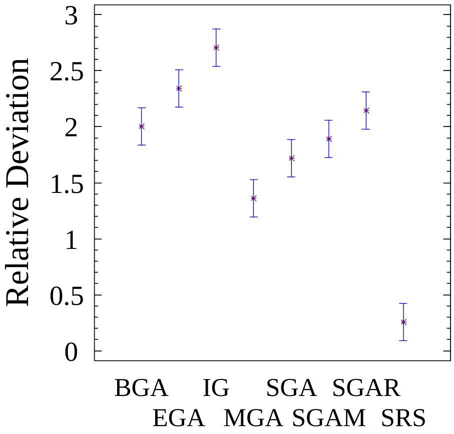


Figure 7.13: Comparison of algorithms. Means and 99% Tukey confidence intervals for the small instances with three machines per stage where 50% of the machines is eligible.

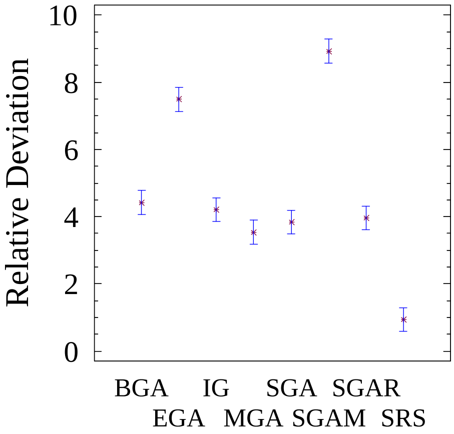


Figure 7.14: Comparison of algorithms. Means and 99% Tukey confidence intervals for the small instances with three machines per stage where all machines are eligible.

In order to know how far we actually are from the optimum, we can use the subset of small instances of 5 to 15 jobs, for which the optimum is found by the MIP approach. We compare the two best performing algorithms, namely SRS and IG, on this set of 272 instances with known optima, running each algorithm five times on each instance, for each of the t values that are used for the large instances as well. Out of the 4,080 algorithm runs, SRS gives the optimum in 3,906 cases (96%) and IG in 3,118 cases (76%). Note that the MIP approach needed up to 15 minutes to find the optimum for some instances, whereas the longest running time for the presented metaheuristics is less than 17 seconds for $t = 125$ms, for the largest instance of 15 jobs and 3 stages, with 3 machines at each stage. From the results in Table 7.7 we can see that the SRS metaheuristic obtains far better results than the IG. The IG gets close to the optimum for most instances, but is unable to reach the optimum due to its limited solution representation in many cases. Because of the representation shift in SRS, this new algorithm obtains the optimal solution for most instances. This shows that the use of a shifting solution representation is even more successful for small instances.

Algorithm	Mean	Stnd. Error	Lower Limit	Upper Limit
SRS	0.28	0.075	0.0867797	0.474453
IG	2.00	0.075	1.80473	2.19241

Table 7.7: Comparison of SRS and IG algorithm. Table of means and 99% confidence intervals for a subset of the small instances where the optimum is known.

7.4. Conclusions

In this chapter, we have shown some possibilities of changing the solution representation during the algorithm. For the case of the first presented genetic algorithm, no significant improvement is registered. The second phase, where a genetic algorithm searches the full search space, appears to be too inefficient to lead to any advantage for the large instances. In the case of carefully designed local search algorithm, however, a shift in the solution representation has a significant impact. As a result, the algorithm called SRS, a new algorithm for the hybrid flexible flowline problem, based on the problem characteristics, proves to outperform all earlier presented methods. Note that, although the local search implementation is quite case-specific, the main idea of shifting the solution representation is generally applicable.

The hybrid flexible flowline problem is a composite problem in the sense that it is composed of different subproblems. In fact, this composition is the case for many real-life problems. Vehicle routing problems, for instance, are composed of a partitioning and a routing problem. Another example is the very large scale integration (VLSI) design problem, that is composed of two subproblems: the choice which components to place and the choice where to place the chosen components. Although we have no data on the application on those problems, the idea of focussing on a subproblem in a first phase and considering all problem aspects in a later phase is likely to yield good solutions in those cases as well.

The research presented in this chapter is summarised in Urlings et al. (2010b).

CHAPTER 8

MULTI-OBJECTIVE SCHEDULING

In all previous chapters, a realistic production environment is studied, where many real-world restrictions are taken into account. For that environment, the maximum completion time was minimised. In industry, however, more goals than maximum completion time, or makespan, are faced. Total flowtime, which is the sum of the time each job remains in the system, is a common objective for schedulers. Total flowtime minimisation reduces work in process (WIP) and cycle times. Another common goal is the minimisation of tardiness. Typically, producers face due dates of the production orders, fixed with their clients. Tardiness can be defined as the non-negative difference between the completion time of a production order and its due date. Different tardiness variants can be considered: total tardiness, maximum tardiness, total weighted tardiness and maximum weighted tardiness are the most common examples.

The optimisation of only one objective has its limitations. An optimal solution for the makespan objective is very efficient from the production point of view, but might be terrible regarding client service. An optimal solution for a tardiness criterion might meet all client wishes for the current production planning, but can be highly inefficient and therefore decrease the production capacity. This indicates a clear tradeoff between efficiency and client service, indicating the need of multi-objective optimisation. In the recent review on hybrid flowshop problems by Ruiz and Vázquez Rodríguez (2010), the need for multi-objective approaches for this problem is confirmed.

This chapter is structured as follows: In Section 8.1, an introduction on multi-objective optimisation is given. Section 8.2 introduces some ways to evaluate the performance of multi-objective algorithms and explains the importance of this step. In Section 8.3, we introduce the problem that is treated in this chapter. The two algorithms that we implemented for the above mentioned problem are described in Section 8.4. Both the calibration of those algorithms and the comparison between them are given in Section 8.5. Finally, in Section 8.6, the results are compared and the conclusions of this chapter are drawn.

8.1. Introduction

Technically, the optimisation of multiple objectives can be done in different ways. In this section, we distinguish the three main streams within multi-objective optimisation, describe briefly in what each method consists and give some references.

8.1.1. Weighted objectives

The easiest and most common method is to summarise the objectives in one new objective. A linear combination is made of the objective functions in order to get a single objective function that represents each of the former goals partially. For a problem with two objectives functions F and G, the new objective function H will be defined as follows: $H : \alpha \cdot F + (1 - \alpha) \cdot G$ where α is a decision parameter that can be used to indicate the importance of each of the two objectives. This kind of optimisation is also referred to as the “a priori” approach, since the weights (α and $1 - \alpha$ in this case) are chosen before the optimisation process. Although it is done for many problems and applications, especially because of the fact that the actual optimisation process is not more complicated, there are some important drawbacks. It is not clear how the weights should be established and things get complicated if the original functions are not in the same scale. For a scheduling example we refer to Sivrikaya Şerifoğlu and Ulusoy (1999), who address a parallel machine problem and minimise a linear combination of earliness and tardiness. In a more recent paper by Davoud Pour and Ashrafi (2009), a linear combination of earliness, tardiness, completion time and the due date chosen by the decision maker is minimised for a flexible flowshop problem with setup times.

8.1.2. Lexicographical approaches

Another, less straightforward manner to take more than one objective into account, is by limiting the solution space to solutions that are "good enough" for all but one of the objectives, and optimise for this new solution space the remaining objective. This methodology is called lexicographic optimisation. If we again consider the two objective functions F and G, we either add a constraint limiting the value of F and optimise G, or add a constraint limiting the value of G and optimise F. In Ruiz and Allahverdi (2009), a combination of both lexicographical optimisation and a linear combination of objectives is applied. They set a maximum value for tardiness and optimise a linear combination of makespan and tardiness for the regular flowshop problem.

8.1.3. Pareto optimisation

The former two ways of working actually convert a problem with multiple separate objective functions into a problem with one single objective function; either by combining the functions or by converting all but one of the functions into constraints. The goal is, as usually, to find the best solution for this optimisation problem. A more desirable approach is Pareto optimisation, which works differently. In Pareto optimisation, we do not search for the best solution for one optimisation problem. Instead, we search for a set of solutions for a set of optimisation problems. For each solution, all objective functions are evaluated. One solution is said to dominate another solution if at least one of the objective values is better and none of them is worse. In this way we can define a set of non-dominated solutions that form the so called Pareto front. Among these solutions, none can be said to be better than another solution in the front. This method is also known as "a posteriori", since the choice which solution of the Pareto front to implement is made after the optimisation procedure. In the rest of this chapter, when speaking about multi-objective optimisation, we refer to Pareto optimisation.

Minella et al. (2008) give an overview and evaluation of multi-objective algorithms for the regular flowshop problem. For the hybrid flowshop problem, hardly any research on multi-objective methods exists. Behnamian et al. (2009) implemented a metaheuristic with three phases to tackle the hybrid flowshop with identical machines in each stage and with setup times. The first stage is a multi-objective adaptation of the genetic algorithm by Kurz and Askin (2004). The second and third phases are a hybrid metaheuristic and a constraint covering method. The description of this method, however, is not clear since they seem to consider a single objective function.
Dugardin et al. (2010) present a new algorithm called L-NSGA, using a Lorenz dominance relationship, and compare it with the optimum found by full solution enumeration, and to adaptations of NSGA-II and SPEA2. The problem they consider however, is different from the standard hybrid flowshop problem, since it models the reentrance of jobs in a stochastic way.

We present some definitions that will be used in this chapter. In order to do so, we consider M objective functions $f_1, \ldots, f_M$ for the problem. First we introduce the notation that has to do with solutions.
Better: for some objective function $f_j, j = 1, 2, \ldots, M$, a solution x_1 is better that another solution x_2 ($f_j(x_1) \lhd f_j(x_2)$) if and only if f_j is a minimisation function and $f_j(x_1) < f_j(x_2)$, or f_j is a maximisation function and $f_j(x_1) > f_j(x_2)$.
Strong (or strict) domination: a solution x_1 strongly dominates another solution x_2 ($x_1 \prec\!\prec x_2$) if and only if $f_j(x_1) \lhd f_j(x_2)$ $\forall j = 1, 2, \ldots, M$, i.e., x_1 is better than x_2 for all objective values.
Domination: a solution x_1 dominates another solution x_2 ($x_1 \prec x_2$) if and only if the following two conditions are met:

- $f_j(x_1) \ntriangleright f_j(x_2)$ $\forall j = 1, 2, \ldots, M$, i.e., x_1 is not worse than x_2 for any of the objectives.

- $\exists j \in \{1, 2, \ldots, M\} : f_j(x_1) \lhd f_j(x_2)$, i.e., at least for one objective, x_1 is better than x_2.

Weak domination: a solution x_1 weakly dominates another solution x_2 ($x_1 \preceq x_2$) if and only if the first domination condition is met, i.e., x_1 is not worse than x_2 for any of the objectives.
Incomparable solutions: solution x_1 and x_2 are incomparable ($x_1 \parallel x_2$ or $x_2 \parallel x_1$) if and only if the following two conditions are met:

- $\exists j \in \{1, 2, \ldots, M\} : f_j(x_1) \lhd f_j(x_2)$, i.e., at least for one objective, x_1 is better than x_2.
- $\exists j \in \{1, 2, \ldots, M\} : f_j(x_2) \lhd f_j(x_1)$, i.e., at least for one objective, x_2 is better than x_1.

This notation can be extended for solution sets as follows:
Better: set A is better that set B ($A \lhd B$) if and only if $\forall x_i \in B\ \exists x_j \in A : x_j \preceq x_i$ and $A \neq B$.
Strong (or strict) domination: a set A strongly dominates another set B ($A \prec\prec B$) if and only if $\forall x_i \in B\ \exists x_j \in A : x_j \prec\prec x_i$.
Domination: a set A strongly dominates another set B ($A \prec B$) if and only if $\forall x_i \in B\ \exists x_j \in A : x_j \prec x_i$.
Weak domination: a set A weakly dominates another set B ($A \preceq B$) if and only if $\forall x_i \in B\ \exists x_j \in A : x_j \preceq x_i$.
Incomparable sets: solution sets A and B are incomparable ($A \parallel B$ or $B \parallel A$) if and only if the following two conditions are met:

- $A \npreceq B$, i.e., A does not weakly dominate B.
- $B \npreceq A$, i.e., B does not weakly dominate A.

Nondominated set: Subset $A^* \subseteq A$ where $x^* \nsucc x\ \forall x^* \in A^*, x \in A$.
Pareto global optimum solution: A solution that is not dominated by any solution in the feasible solution space $x_i : \nexists x_j \prec x_i$.
Pareto global optimum set: A set is called a Pareto global optimum set if it contains all and only Pareto global optimum solutions. Such a set is also referred to as Pareto front.

8.2. Multi-objective quality measures

Since the output of a Pareto optimisation method is not a single solution to the scheduling problem, but a set of solutions that approximates the Pareto front (approximation set), the usual quality measures such as the average relative percentage deviation are no longer applicable. Instead, the quality of approximation sets should be compared among each other. This is not straightforward at all. If all solutions in an approximation set A are dominated by solutions in a approximation set B, then obviously set B is better than set A. But often some solutions from set A dominate some solutions of set B and some solutions of set B dominate some solutions of set A. In this case it is not clear which approximation set is preferable. In this section we show several ways to define the best approximation set in such a case.

Zitzler et al. (2008) distinguish three procedures for comparing multi-objective algorithms. A first procedure is based on the Pareto dominance relations among solution sets. One executes two algorithms A and B many times and counts the number of times that the approximation set of A strongly, regularly or weakly dominates the set of B; and the number of times that the set of B strongly, regularly or weakly dominates the set of A. A second procedure calculates a quality indicator for each approximation set. Using such an indicator, the usual methods for the comparison of single-objective algorithms can be applied. The last option the authors propose, is to use empirical attainment functions, that register the differences between approximation sets.

The first method using Pareto dominance has several important drawbacks. First of all, this technique can only be applied for the comparison of two algorithms. If three algorithms are to be analysed, comparison of A with B, B with C and A with C is required. More generally, for k algorithms, $\sum_{i=1}^{k-1} i = k(k-1)/2$ pairs of algorithms should be compared. This can easily get out of hand. Moreover, some information is lost during this approach. The only information that is used is if one approximation set dominates another, but not in which extend. Therefore, an algorithm always producing approximation sets that just dominate the sets of another algorithm is evaluated in the same way as an algorithm that outperforms the other algorithm with a huge difference.

Each time two approximation sets are incomparable, that is, none of the two dominates the other, no information is added to the analysis. Because of the previously mentioned grounds, we do not include dominance ranking in this Ph.D. thesis.

8.2.1. Quality indicators

A quality indicator assigns a real value to a set of solutions. This value can be used to compare the quality of distinct sets of solutions. The first and most important requirement of such an indicator, is the so called Pareto-compliance. This means that a set of solutions that dominates another set of solutions should have a better value than the other set that it dominates. Knowles et al. (2006) show that several common used indicators are not Pareto-compliant, which can lead to wrong or misleading conclusions. Some examples of such metrics are generational distance and maximum deviation from the best Pareto front. These measures are applied in the quite recent publications by Rahimi-Vahed et al. (2007) and Geiger (2007).
Zitzler et al. (2008) appoint the hypervolume indicator (I_H) and the unary multiplicative epsilon indicator (I_ϵ^1) as the state-of-the-art regarding quality measures and show that both fulfill the Pareto-compliance requirements. In the following, the two indicators are highlighted and their calculation is explained.

The hypervolume indicator (I_H) is proposed in Zitzler and Thiele (1999). Given a problem with a set of M objective functions, we can consider an M-dimensional space of objective values. An approximation set divides this space in two: the part that is dominated or covered by the approximation set and the part that is not covered. If the approximation set is optimal, or equal to the Pareto front, then no feasible solutions have their objectives values in the uncovered part of the objective values space. The hypervolume indicator is based on the volume of the covered space by each of the approximation sets. The volume that we measure is limited by the approximation set on the one hand, and by a reference point on the other hand. Without loss of generality, we will assume from here on that all objective functions are minimisation functions. Then the reference point is chosen to be r, where

$f_j(r) = f_j^+ + 0.2 \cdot (f_j^+ - f_j^-)$ for $j = 1, 2, \ldots, M$, and f_j^+ and f_j^- are the maximum and the minimum values found for objective j, respectively. When calculating the hypervolume indicator for a number of approximation sets, first the objective values are normalised. The resulting normalised hypervolume indicator is denoted $I_{|H|}$. For each solution x, the normalised value $g_j(x)$ is defined as $g_j(x) = (f_j(x) - f_j^-)/(f_j^+ - f_j^-)$, so that 0 will correspond to the best value found in all approximation sets, and 1 to the worst value. It is easy to verify that the maximum possible hypervolume is 1.2^M for a approximation set consisting of a single solution x_b dominating all other sets. In this case, $f_j(x_b) = f_j^-$ for $j = 1, 2, \ldots, M$. On the other hand, the worst possible approximation set has a volume of 0.2^M. Such a set exists of a single solution x_w, for which $f_j(x_w) = f_j^+$ for $j = 1, 2, \ldots, M$.

Zitzler et al. (2003) define in their article the concept of weak ϵ-dominance. One solution x_1 weakly ϵ-dominates another solution x_2 for a given $\epsilon > 0$ ($x_1 \succeq_\epsilon x_2$) if and only if $f_j(x_1) \not> \epsilon \cdot f_j(x_2)$, $\forall j = 1, 2, \ldots, M$. Based on this definition, they define a binary ϵ-indicator I_ϵ in order to compare two approximation sets A and B. The definition is rewritten by Minella et al. (2008) and results as follows:

$$\begin{aligned} I_\epsilon^2(A, B) &= \inf_{\epsilon \in \mathbb{R}}\{\forall x_B \in B \exists x_A \in A : x_A \succeq_\epsilon x_B\} \\ &= \max_{x_B} \min_{x_A} \max_j f_j(x_A)/f_j(x_B) \end{aligned} \tag{8.1}$$

where x_A and x_B are solutions given by the algorithms A and B, respectively. In order to better understand the last expression, we explain how to obtain this result. For all possible pairs x_A and x_B, the objective j is chosen that maximises the quotient $f_j(x_A)/f_j(x_B)$. Then, for each solution x_B, x_A is chosen so that $\max_j f_j(x_A)/f_j(x_B)$ is minimised. Finally, x_B is chosen in such a way that $\min_{x_A} \max_j f_j(x_A)/f_j(x_B)$ is maximised. When one of the objective takes the value of zero, which can happen in the case of tardiness, I_ϵ can not be calculated. Therefore, and since a normalisation of objectives is required to obtain fair results, a transformation is applied to the objective values. The normalisation function $g_j(x)$ is defined as follows: $g_j(x) = (f_j(x) - f_j^-)/(f_j^+ - f_j^-) + 1$.

The normalised indicator is defined

$$I^2_{|\epsilon|}(A,B) = \max_{x_B} \min_{x_A} \max_{j} g_j(x_A)/g_j(x_B) \tag{8.2}$$

In order to avoid the necessity to compare each pair of two approximation sets, the authors also introduce a unary ϵ-indicator $I^1_{|\epsilon|}$, which is defined as follows:

$$I^1_{|\epsilon|}(A) = I^2_{|\epsilon|}(A,P) \tag{8.3}$$

where P is the Pareto global optimum set. Since the Pareto front is usually not known when comparing metaheuristics, the indicator can slightly be modified in order to apply it. Instead of the Pareto optimum set P, we use the Pareto best known set $\widehat{P}$. In order to obtain this set we select the non-dominated solutions from the union of all approximation sets. That is, $\widehat{P}$ contains only and all Pareto best known solutions.
Due to the transformation, values for $I^1_{|\epsilon|}$ are between 1 and 2. A value of 1 is obtained if there is a solution $x_A \in A$ for which the following holds: $f_j(x_A) = f_z^- \; \forall j \in \{1, 2, \ldots, M\}$. Since the minimum for all objectives is found in one solution, it is easy to see that $A = \widehat{P} = \{x_A\}$ in this case. Since $\widehat{P}$ is not influenced by A, and usually contains more than one solution, for most problems it is impossible to obtain a $I^1_{|\epsilon|}$ equal to 1, even if the Pareto front is completely covered. On the contrary, $I^1_{|\epsilon|} = 2$ only if for all solution $x_A \in A$, there is a $j \in \{1, 2, \ldots, M\}$, such that $f_j(x_A) = f_z^+$.

When comparing the two indicators, one of the differences is that the hypervolume indicator reacts directly on whatever change in the approximation set, but might ignore changes in the best known Pareto approximation set, while the ϵ-indicator might not change if a solution in the set is improved or added, but is more sensitive to changes in the best known Pareto approximation. The results of the two indicators can be contradictory. This can be shown with a simple example:
Consider in the bi-dimensional objective space two approximation sets A and B, where $A = \{a_1, a_2\}$ and $B = \{b_1\}$. The three objective vectors have the following values: $a_1 = \{1, 6\}$, $a_2 = \{6, 1\}$ and $b_1 = \{3, 3\}$. These vectors are

visualised in Figure 8.1. It is easy to verify that $A \| B$. Let us now calculate both quality indicators.

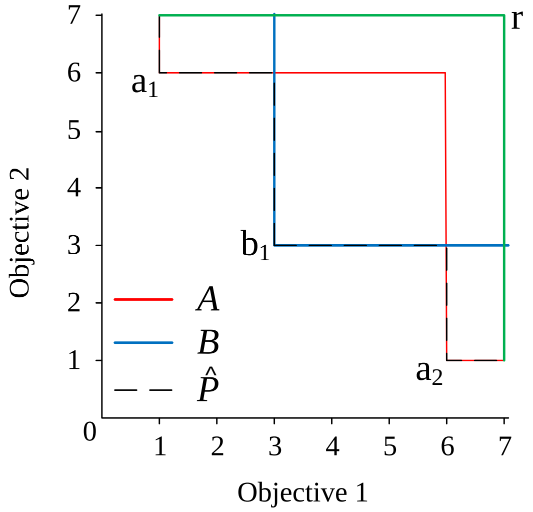


Figure 8.1: Example of two Pareto approximation sets in bi-dimensional objective space.

- The hypervolume indicator. We will first calculate the non normalised hypervolume I_H for both approximation sets. The maximum objective values are given by $f_1^+ = f_2^+ = 6$, while the minima are $f_1^- = f_2^- = 1$. Therefore, the reference point r that limits the area to the top and to the right is $\{6 + 0.2(6-1), 6 + 0.2(6-1)\} = \{7, 7\}$. Now calculation of the indicator is easy: $I_H(A) = ((7-6) \cdot (7-1)) + ((6-1) \cdot (7-6)) = 6 + 5 = 11$ and $I_H(B) = ((7-3) \cdot (7-3)) = 16$. When applying normalisation, $I_{|H|}(A) = I_H(A)/(6-1)^2 = 11/25 = 0.44$ and $I_{|H|}(B) = I_H(B)/25 = 16/25 = 0.64$. This indicates a victory for approximation set B.

- The ϵ-indicator. Since the choice of the objective j in the first step is already influenced by the normalisation, we start directly with the calculation of $I_{|\epsilon|}^1$. Since $a_1 \| a_2$, $a_1 \| b_1$ and $a_2 \| b_1$, the best known Pareto approximation set for this problem instance is $P^* = \{a_1, a_2, b_1\}$. Now $I^1_{|\epsilon|}(A) = I^2_{|\epsilon|}(A, \widehat{P}) = \max\{\min\{1, 2\}, \min\{2/1.4, 2/1.4\}, \min\{2, 1\}\} = \max\{1, 10/7, 1\} \approx 1.43$ and $I^1_{|\epsilon|}(B) = I^2_{|\epsilon|}(B, \widehat{P}) = \max\{1.6, 1, 1.6\} = 1.6$. In this case, approximation set A wins.

In this example, the hypervolume indicator signalised that B is preferred to A, while the ϵ-indicator shows a preference for set A. This expresses in numbers what can otherwise only be seen graphically: that the two approximation sets are not comparable. Since the use of these two indicators can help to distinguish these cases, we opt for using both the hypervolume indicator and the ϵ-indicator.

8.2.2. Empirical attainment functions

Another way for evaluating approximation sets is based on goal-attainment and was initiated by Grunert da Fonseca et al. (2001). They say that an optimiser attains a goal, in this case an objective vector, if at least one of the elements of its resulting approximation set weakly dominates the objective vector. Given this concept, they define the attainment function α for a given Pareto optimiser and a point z in the M-dimensional objective space as the probability that z is (weakly) dominated by any approximation set $A = a_1, a_2, \ldots, a_N$ obtained by the optimiser, where N is the number of solutions in set A. More formally:

$$\alpha_A(z) = P(A \preceq z) = P(a_1 \preceq z \vee a_2 \preceq z \vee \cdots \vee a_N \preceq z) \tag{8.4}$$

Here $P(\cdot)$ is the probability of a certain event and $\vee$ is the logical operator “or”. The problem when using this attainment function in practice, is that the exact probabilities are unknown. However, when the optimiser has been executed several times, the function can be estimated empirically. The resulting estimated function is denominated the empirical attainment function (EAF). The empirical attainment function $\alpha_n(z)$ that estimates the probability of attaining z with the help of n approximation sets $A_1, A_2, \ldots, A_n$ generated independently by the considered multi-objective optimiser is defined as follows:

$$\alpha_n(z) = \frac{1}{n} \sum_{i=1}^{n} I(A_i \preceq z) \cdot \tag{8.5}$$

where $I(\cdot)$ takes the value 1 or 0 when the described event happens or not, respectively.

Since the attainment function depends on the objective vector z, it is not easy to draw a conclusion when comparing two Pareto optimisers. In order to cope with this inconvenience, Zitzler et al. (2008) propose a way to visualise the outcome of multiple optimiser runs. This method is based on the concept of $k\%$-attainment sets. A Pareto approximation set A^k is a $k\%$-attainment set if and only if A^k weakly dominates all objective vectors z that have been attained in at least $k\%$ of the n runs. Consequently, the attainment surface S^k of A^k can be defined as all vectors z, weakly dominated by A^k.

$$S^k = z \in \Re^M \| \frac{1}{n} \sum_{i=1}^{n} I(A_i \preceq z) \geq k/100 \tag{8.6}$$

where $A_i, 1 \leq i \leq n$ are the independently generated Pareto approximation sets. These attainment surfaces can visually be shown when two objectives are considered. In order to show more information in one graph, several attainment surfaces can be shown one over the other, always with the $k\%$-attainment surface on top of the $l\%$-attainment surface, for $k > l$. If a darker colour is used for higher attainment surfaces, the incremental graph is quite intuitive to understand. The example of Subsection 8.2.1 is used to draw the attainment functions of the set of two Pareto approximation sets. In this case they are treated as is they were generated by one algorithm. The white area is not covered by any of the sets, the grey area is covered by one of them and the black area is attained by both approximation sets.

For comparison between two algorithms, we propose to use the differential empirical attainment functions (Diff-EAF). The definition of a Diff-EAF between the Pareto optimisers A and B in a given point z is as follows: $\delta_n(z) = \frac{1}{n}\sum_{i=1}^{n} I(A_i \preceq z) - I(B_i \preceq z)$. The outcome is in between 100%, when optimiser A attains vector z in all runs and optimiser B in none of them, and -100%, when the contrary happens. For clear visualisation, two colours can be used; one colour for the area where $\delta > 0$ and another colour for $\delta < 0$. In such a graph, the performance of A is equal to the performance of B wherever the area is not coloured. The stronger either of the colours is, the larger the

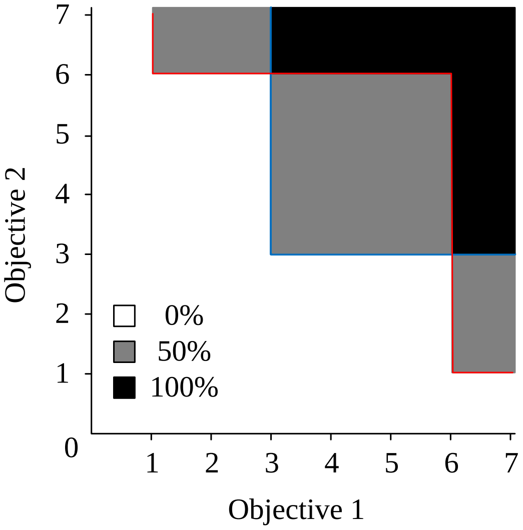


Figure 8.2: Example of visualised empirical attainment functions in bi-dimensional objective space.

difference between one algorithm and the other, for this instance. Such a graph is given for the earlier used example in Figure 8.3. The area coloured in red is attained in 100% of the cases by algorithm A and in 0% of the cases by algorithm B. The blue area, in contrast, is attained in 0% of the cases by A and in 100% by B. Note that there is only one case for each of the algorithms in this example.

The attainment function approach is different from the quality indicators in the sense that the output is in M-dimensional space, instead of a real number. Therefore, it contains more information and differences between Pareto optimisation algorithms can be analysed in more detail. The computational cost, however, is considerably higher for the attainment functions. Moreover, the visualised empirical attainment functions are applied only for two optimisers and one instance. Consequently, they cannot be used for the evaluation of massive experiments. However, some instances can be analysed in detail using this technique. These example instances can give some visual indications about how two algorithms differ among each other. We will therefore use the empirical

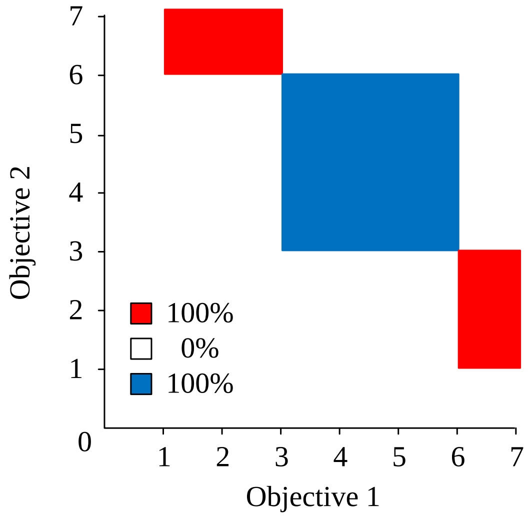


Figure 8.3: Example visualised differential empirical attainment functions in bi-dimensional objective space.

attainment function visualisation in addition to the hypervolume and epsilon indicators.

8.3. Problem description

In this chapter, we consider a hybrid flowshop problem. The flowshop consists in a set M of m stages, where each stage i contains a set M_i of m_i parallel unrelated machines. A set N of n jobs is given, where each job j has a due date d_j. Each job j has to be processed by exactly one machine l at every stage i, where the processing time is defined p_{ilj}. Two objectives are considered simultaneously, namely makespan and total tardiness. If we define C_j to be the the completion time of job j at the last stage, then makespan can be denoted $\max\limits_{j \in N} C_j$ and total tardiness $\sum_{j=1}^{n} \max(C_j - d_j, 0)$.

Note that most of the constraints that are treated in the previous chapters are dropped now. One has to realise that multi-objective scheduling is far more complex than single-objective scheduling. In the multi-objective scheduling literature, hardly any research on hybrid flowshop problems can be found, as demonstrated in Ruiz and Vázquez Rodríguez (2010) and Ribas et al. (2010). If we would have converted the complete problem of the previous chapters into a Pareto multi-objective problem, a sound basis would have been missing and the connection with existing literature would have been lost. The most similar problem considered in the literature, is a hybrid flowshop problem with identical machines, by Behnamian et al. (2009). The assumption of identical machines reduces the complexity in an important way, since machine assignments lose their importance.

The single objective mathematical model for this problem is considerably shorter than the model for the highly constrained HFFL, presented in Chapter 3. The model involves the following decision variables:

$$\begin{aligned}
X_{jki} &= \begin{cases} 1, & \text{if job } j \text{ precedes job } k \text{ at stage } i \\ 0, & \text{otherwise} \end{cases} \\
Y_{jil} &= \begin{cases} 1, & \text{if job } j \text{ on stage } i \text{ is scheduled in machine } l \\ 0, & \text{otherwise} \end{cases} \\
C_{ji} &= \text{Completion time of job } j \text{ at stage } i \\
C_{\max} &= \text{Maximum completion time}
\end{aligned}$$

The objective function is either:

$$\min C_{\max} \tag{8.7}$$

or

$$\min \sum_{j=1}^{n} \max(C_j - d_j, 0) \tag{8.8}$$

And the constraints are:

$$\sum_{l=1}^{m_i} Y_{jil} = 1, \qquad j \in N,\ i \in M \tag{8.9}$$

$$\sum_{l=1}^{m_i} Y_{jil} \cdot p_{ilj} \leq C_{ij} - C_{i-1,j}, \qquad j \in N,\ i \in M \tag{8.10}$$

$$V(2 - Y_{jil} - Y_{kil} + X_{jki}) + c_{ij} - c_{ik} \geq p_{ilj}, \quad j,k \in N,\ j < k \tag{8.11}$$

$$V(3 - Y_{jil} - Y_{kil} - X_{jki}) + c_{ik} - c_{ij} \geq p_{ilk}, \quad j,k \in N,\ j < k \tag{8.12}$$

$$C_{0j} = 0, \qquad j \in N, \tag{8.13}$$

$$Y_{jil} \in \{0,1\}, \qquad j \in N,\ i \in M,\ l \in M_i \tag{8.14}$$

$$X_{jki} \in \{0,1\}, \qquad j \in N,\ i \in M,\ l \in M_i \tag{8.15}$$

$$C_{ij} \geq 0, \qquad j \in N,\ i \in M \tag{8.16}$$

$$C_{\max} \geq C_{mj}, \qquad j \in N \tag{8.17}$$

where V is a high positive value.

The set of constraints (8.9) guarantees that each job is assigned to exactly one machine at every stage. Constraints set (8.10) assures that a job does not start in certain stage, before it finishes in the previous stage. Constraint sets (8.11) and (8.12) prevent two jobs assigned to the same machine from overlapping. The constraints set (8.13) represents the fact that C_{0j} is the release time of job j, which is assumed to be zero in this chapter. Constraint sets (8.14), (8.15) and (8.16) define the domain for the decision variables. Finally, the set (8.17) is needed for the makespan objective.

Not only the constraints are different in this HFS model, compared to the HFFL model in Chapter 3, but also the decision variables. Recall that the decision variables X_{iljk} in the HFFL model equal 1 if job j precedes job k on machine l at stage i. The amount of variables is $\sum_{i=1}^{m} m_i n^2$, which is usually more than the amount of variables for the HFS model presented in this Chapter: $mn^2 + \sum_{i=1}^{m} m_i n$. The rare condition that the number of decision variables is higher for the HFS model can be deducted as follows:

$$\begin{array}{ll} \text{variables } HFFL & < \text{variables } HFS \\ \sum_{i=1}^{m} m_i n^2 & < mn^2 + \sum_{i=1}^{m} m_i n \\ \sum_{i=1}^{m} m_i n - \sum_{i=1}^{m} m_i & < mn \\ \sum_{i=1}^{m} m_i & < m\frac{n}{n-1} \end{array} \tag{8.18}$$

This only occurs if most stages have only one machine and if the number of jobs is very low, i.e., $m = 3$, $\sum_{i=1}^{m} m_i = 4$ and $n = 2$.

In order to generate valid instances for this problem, we start from the instances used for the HFFL in the previous chapters. We take the 288 small instances with three machines per stage and ignore all constraints that are not considered in this chapter: setup times, release dates, precedence relationships and time lags. The small instances with a single machine per stage are not used in this chapter, since those do not represent the hybrid flowshop problem, strictly taken. According to most definitions, a hybrid flowshop has more than one machine in at least one of the stages. Of the instances with three machines per stage, we eliminate all instances with stage skipping and with machines that are not eligible. This allows us to continue with 72 small instances. In order to generate relevant due dates for the jobs, we need the optimal makespan value, or an estimation of it. We run the above given model with the makespan objective in CPLEX, with a time limit of one hour. If the optimum is found by CPLEX within the time limit, we are done; otherwise we need to estimate the optimal makespan. Fortunately, CPLEX found a feasible solution that can serve as an upper bound for every problem instance. CPLEX also returns a lower bound for each problem instance.
Apart from that, we implemented two fast lower bounds. For both lower bounds we define the processing time for a job j at a stage i to be the minimum processing time among the different processing times on the machines in stage i, as formally described in Equation 8.19. The first lower bound ($LB1$) in Equation 8.20, is the highest sum of processing times for a job. The second lower bound ($LB2$) in Equation 8.21 is less straightforward. For each stage i, we calculate the minimum total processing time at stage i and deduce the average minimum workload per machine. Then we add for each stage i the minimum time needed to be able to start the first task at stage i. As there are m_i machines at stage i, and the first job at each machine first needs to be processed first in all previous stages, the minimum time previous to stage i is the m_ith smallest sum of processing times over the stages previous to stage i. Similarly, the minimum time after stage i equals the m_i-th smallest sum of processing

times over the stages after stage i. Note that $sort_z^+(S)_c$ is used to denote the c-th element of a set S, where the set S is arranged in increasing order.

$$p_{ij} = \min_{l \in M_i} p_{ilj} \tag{8.19}$$

$$LB1 = \max_{j \in N} \sum_{i=1}^{m} p_{ij} \tag{8.20}$$

$$LB2 = \max_{i \in M} \Big(\sum_{j=1}^{n} p_{ij} + sort_k^+ (\sum_{a=1}^{i-1} p_{ak})_{m_i} + sort_o^+ (\sum_{b=i+1}^{m} p_{bo})_{m_i} \Big) \tag{8.21}$$

The initial estimation of the optimal makespan ($\widetilde{C_{\max}}$) is calculated as the average between the upper bound given by the feasible solution found by CPLEX, and the minimum of the three lower bounds; the lower bound by CPLEX, $LB1$ and $LB2$. In mathematic notation: $\widetilde{C_{\max}} = \frac{1}{2}(CPLEX_{\max} + \max\{CPLEX_{\min}, LB1, LB2\})$. Some initial tests showed that the estimation was very high in some cases, which means that the upper bound is further away from the optimal makespan than the lower bound. We have therefore adapted the NEH implementation of Section 4.5 to this problem. If the NEH solution has a smaller solution value than the initial estimation $\widetilde{C_{\max}}$, we substitute the estimation by the makespan of the NEH solution. This defines the final estimation $\widehat{C_{\max}}$ as follows:

$$\widehat{C_{\max}} = \min\{\widetilde{C_{\max}}, NEH\} \tag{8.22}$$

The due dates are generated considering the makespan estimation and two instance parameters: the tardiness factor (T) and the due date range (R). Each due date d_j for job j is chosen with the help of a uniform probability distribution between $C_{\max}(1 - T - R)$ and $C_{\max}(1 - T + R)$. This method to generate due dates is introduced by Potts and Van Wassenhove (1982) and later used by many others, e.g. Armentano and Ronconi (1999). The values for T and R are chosen equal to the values in the review of Vallada et al. (2008): $T = \{0.2, 0.4, 0.6\}$ and $R = \{0.1, 0.3, 0.5\}$. All combinations of the parameters lead to a total of $72 \cdot 9 = 578$ small instances. One third of them is used as a set of test instances;

the rest form the benchmark for the final comparison of algorithms. For the large instances the same procedure is used, which leads to a calibration set of 144 instances and a final benchmark of 288 large instances. All instances can be downloaded from `http://soa.iti.es/problem-instances`.

We introduce example instance 4, based on the instance used in Section 4.3. The processing times remain unchanged, and the remaining constraints and problem data are ignored. See Table 8.1 for the processing times and the added due dates. In Figure 8.4, an optimal solution with respect to the makespan objective is shown. The makespan is determined by the processing times of job 4; it is easy to determine that the makespan of 112 is the optimum. Note that in most cases the jobs are assigned to the same machine as in Figure 4.21, but that machine assignments change in several cases. The lateness for the jobs is 13, -9, -13, 14 and -7, respectively. The jobs that do not meet their due dates are job 1 and job 4. Their lateness determines the total tardiness for this solution to be $13 + 14 = 27$.

Stage	1			2			3			
Machine	1	2	3	4	5	6	7	8	9	d_j
Job										
1	18	62	23	61	36	24	90	13	86	94
2	65	29	98	76	37	18	82	93	17	92
3	6	53	39	28	30	78	70	58	52	105
4	56	69	34	81	95	31	47	48	71	98
5	25	38	57	22	99	68	14	2	15	101

Table 8.1: Example instance 4. Processing times and due dates for each job.

8.4. Proposed Algorithms

If finding optimal solutions for hybrid flowshop problems with a single objective is hard, and in practice only feasible for small problem instances, finding an optimal Pareto front is even more so. Full enumeration of all feasible

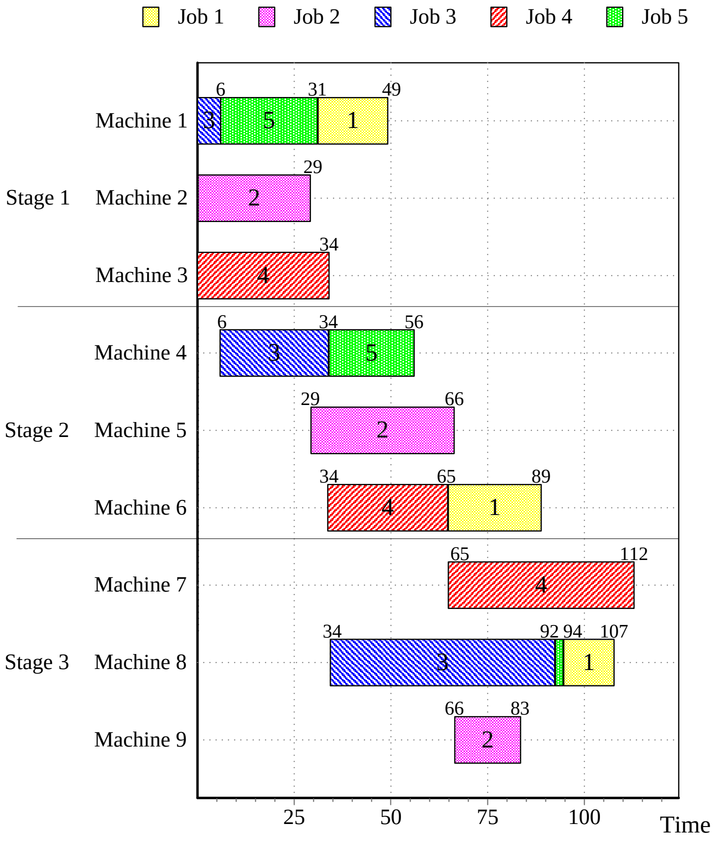


Figure 8.4: Gantt of an optimal solution with respect to the makespan objective for the problem instance defined in Table 8.1.

solutions, as applied by Dugardin et al. (2010), is limited to tiny problems. In the search for a good Pareto front, the use of metaheuristics is therefore more adequate. In this section we show the hybrid flowshop adaptation of the state-of-the-art NSGA-II method and we present a new multi-objective algorithm called RIPG.

Both algorithms use a job permutation π^1 as solution representation. The permutation π^1 determines the order in which the jobs are assigned in the first stage. For the assignment of jobs to machines, the Earliest Completion Time (ECT) rule, presented in Chapter 4, is used. According to this rule, jobs are assigned to the machine that is able to finish the job earliest. When all jobs are scheduled in the first stage, the jobs are put in increasing order of finishing time. This order, or job permutation, denoted π^2, is the order in which the jobs are launched in the second stage. In general, the order π^i in which jobs are scheduled in stage i, is the order in which they are finished processing in stage $i-1$, for $2 \leq i \leq m$. Note that this stage-by-stage scheduling of tasks usually leads to better solutions than job-by-job scheduling, since changing the job permutation through the stages avoids unnecessary idle times. For the HFFL problem considered in the previous chapters, assignment on stage-by-stage basis is impossible due to the precedence constraints, but that does not play a role here.

8.4.1. NSGA-II

After the first Non-dominated Sorting Genetic Algorithm (NSGA) by Srivinas and Deb (1994), Deb et al. (2002) presented its successor, NSGA-II. The algorithm starts with a population of randomly generated individuals, where the population size *pop* is an input parameter. Random selection is used in order to choose two individuals for One-Point Order Crossover. Then shift mutation is applied to the outcome of the crossover. The new individuals are inserted in a new population. When the new population has reached size *pop*, a fast non-dominated sorting method is applied to the population. This method divides the population in subsets $F = (F_1, F_2, \ldots, F_N)$, where F_i dominates F_j if and only if $i < j$. We maximise k such that $\sum_{i=1}^{k} |F_i| \leq \mathit{pop}$ and define the new population $F_1 \cup, \ldots, \cup F_k$. A crowding-comparison operator is used to sort the individuals in F_{k+1} and defines the new population as the first l individuals, where $l = \mathit{pop} - \sum_{i=1}^{k} |F_i|$. This procedure is repeated until the stopping criterion is met.

A pseudocode for NSGA-II is given in Algorithm 8. For more details, the reader is referred to Deb et al. (2002), who are the founders of the original algorithm, and to Minella et al. (2008), who evaluated the algorithm for the permutation flowshop problem, together with 22 other algorithms.

Algorithm 8: Non-dominated Sorting Genetic Algorithm II

```
Input: instance data, pop
Output: set of non-dominated solutions
begin
    for $i = 1$ to pop do
        generate one random initial solution;
        insert individual in Population;
    repeat
        for $i = 1$ to $pop/2$ do
            randomly select two solutions of Population;
            apply one-point order crossover to selected solutions;
            apply shift mutation to crossed solutions;
            insert new solutions in NewPopulation;
        copy NewPopulation into Population;
        empty NewPopulation;
        apply non-dominated sorting to Population in order to define fronts;
        while size of Population > pop do
            empty Temp;
            copy individuals of worst front of Population to Temp;
            delete individuals of worst front from Population;
        apply crowding-distance comparison sorting to Temp;
        copy the first (pop - size of Population) individuals to Population;
    until time > max time ;
    return non-dominated solutions in NewPopulation;
end
```

8.4.2. RIPG

The new Restarted Iterated Pareto Greedy (RIPG) algorithm is based on the Iterated Greedy algorithm, that is originally presented by Ruiz and Stützle (2007) for the permutation flowshop problem. In Chapter 6 we have proposed a new version adapted to the hybrid flexible flowline problem. A computational evaluation has proven good performance for the hybrid problem. In this chapter we have further developed the algorithm in order to make it applicable to and effective for Pareto optimisation. Since the consideration of multiple objectives implies working with a set of solutions instead of working with a single solution, the required changes are important and allows us to speak of a new algorithm, rather than a modified existing algorithm.

In the case of single objective optimisation, it is common knowledge that a good initialisation is important for good final results. When optimising more than one objective, it is difficult to find a initial solution that is good for all objectives. Instead, an initial set of solutions with one good solution for each of the objectives, is a good start for the algorithm. The presence of a specific solution for each optimisation criterion avoids the Pareto approximation set to be concentrated too much in one direction and assures a "wide" approximation set. When optimising makespan and tardiness, the initial solutions are generated with two distinct constructive heuristics. The heuristic by Nawaz et al. (1983) is famous for its good results and efficiency for the makespan criterion in flowshop problems. Its adaptation for hybrid flowshop in Chapter 4 had been proven to be just as effective. The modified heuristic is therefore used to generate an initial solution with a good makespan value. For the creation of a solution with a low tardiness value, the heuristic of Rajendran and Ziegler (1997) is applied. Beginning with these two solutions, the main loop is entered, where each iteration includes the following operators: Selection, which chooses one solution that will be subject to a Greedy Phase (GP). This phase, directly inspired in the IG, first excludes a number of jobs (given by a parameter) to include them again one by one. When a job is included, this is done at all positions respectively. The resulting partial solutions are compared and the non non-dominated ones are maintained for inclusion of the next job. The IG phase therefore has one solution as input and a set of solutions as output. Selection is applied again in order to choose a solution for local search, where a job can be inserted in a new position within a maximum distance (parameter) of its current position.

In preliminary tests, the algorithm sometimes got stuck, i.e., no new solutions were added to the approximation front before the termination criterion is met. Therefore, a restart mechanism is added to the algorithm. If a certain number of iterations without improvement is done, the algorithm is restarted. The number of iterations is given by an input parameter for the algorithm. All non-dominated solutions are saved in a global archive and a new random population is initialised. When the termination criterion is met, the dominated solutions in the global archive are deleted and the other solutions are the output of the algorithm. The pseudocode for RIPG is given in Algorithm 9.

Algorithm 9: Restarted Iterated Pareto Greedy

```
Input: instance data, $restart$
Output: set of non-dominated solutions
begin
    foreach objective function do
        generate directed initial solution;
        insert solution in WorkingSet;
    eliminate dominated solutions from WorkingSet;
    set Iteration to 0;
    set Cardinality to size of WorkingSet;
    repeat
        select one solution of WorkingSet;
        apply greedy phase to selected solution in order to obtain NewSet;
        copy NewSet into WorkingSet;
        empty NewSet;
        eliminate dominated solutions from WorkingSet;
        select one solution of WorkingSet;
        apply local search to selected solution in order to obtain NewSet;
        copy NewSet into WorkingSet;
        empty NewSet;
        eliminate dominated solutions from WorkingSet;
        if size of WorkingSet $\neq$ Cardinality then
            set Iteration to 0;
        set Cardinality to size of WorkingSet;
        if Iteration > $restart$ then
            insert WorkingSet into GlobalSet;
            empty WorkingSet;
            for $i = 1$ to $100$ do
                generate random initial solution;
                insert solution in WorkingSet;
            eliminate dominated solutions from WorkingSet;
            set Iteration to 0;
    until time > max time ;
    insert WorkingSet into GlobalSet;
    return non-dominated solutions in GlobalSet;
end
```

8.5. Computational Evaluation

In this section, the computational results for the two developed multi-objective metaheuristics are presented, analysed and interpreted. Different from the tests done in the previous chapters, the computational experiments are executed on a cluster of 12 identical computers, each with Intel Core 2 Duo E6600 processors running at 2.4 GHz with 2 GB of RAM. Both algorithms are compiled in Delphi 2009 and run under Windows XP. Newer computers are used for the tests in this chapter, since computers evolve over time and the computers of the previous chapters started to be old over the course of time. Moreover, they required more maintenance and are now outnumbered by the new cluster. No quantitative comparison is made between the experiments done in each of the clusters, so the results are not affected by the change in hardware. The stopping criterion is the same for both algorithms and given by a CPU time limit depending on the instance size. The limit is calculated with the following formula: $t \cdot n \cdot \sum_{i=1}^{m} m_i$ milliseconds, where t is a time parameter. We have fixed t at 100 milliseconds for the tests reported in this section. Multi-objective problems are harder to solve, because of the more complicated solution dominance definitions. Therefore smaller values for t do not show the full potential of the algorithms.

8.5.1. Calibrations

In Chapter 5 we have seen that the calibration of genetic algorithms is a key element for good performance. The same holds for algorithms based in local search, as shown in Chapter 6. Taking this into account, NSGA-II and RIPG should also be calibrated. For calibration of the algorithm we use the 144 large test instances that were generated as described in Section 8.3. For each parameter setting and each instance, 5 independent runs are done. We use the hypervolume indicator in order to measure outcome quality of the algorithm. The advantage of the hypervolume indicator is that it is fast and practical on the one hand, and trustworthy on the other hand, since it fulfills the requisitions for being Pareto-compliant. An analysis of variance (ANOVA) is used to interpret the results.

In the calibration of NSGA-II, we take three parameters into account:

- Crossover probability: 50%, 70%, 90%.
- Mutation probability (per job): 30%, 70%, $1/n$.

- Population size: 10, 30, 100.

The mutation type has already been fixed to shift mutation in a prior stage, due to clear dominance with respect to the other mutation types. For the same reason, the crossover type that is used is one-point order crossover. For a description of both operators, we refer the reader to Chapter 5.
This results in a total amount of $3^3 \cdot 144 \cdot 5 = 6,480$ algorithm runs. In CPU time, this means a calibration of 729 hours. The F-values, that determine the statistical significance of each parameter, are given in Table 8.2. As can be observed, the mutation probability has the highest F-value; this parameter is therefore fixed first. Two of the tested probabilities are constant values (30% per job and 70% per job, respectively). The third value, however, depends on the problem instance. The probability is given by the formula $\frac{1}{n} \cdot 100\%$, which for the large instances is either 2% when $n = 50$ of 1% when $n = 100$. From Figure 8.5 we can conclude that the highest probability of 70% is preferable over the lower probabilities. Note that the hypervolume is to be maximised, different from the average percentage deviation in the graphs of the previous chapters. More exact data can be taken from Table 8.3.

Table 8.2: Analysis of Variance for the Hypervolume - calibration of NSGA-II for the set of large instances.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	65.7549	1	65.7549	1606.17	0.0000
B:m	0.354098	1	0.354098	8.65	0.0033
C:m_i	132.004	1	132.004	3224.43	0.0000
D:T	0.373747	2	0.186874	4.56	0.0104
E:R	1.97975	2	0.989873	24.18	0.0000
F:Pop	70.1695	2	35.0847	857.00	0.0000
G:Mut	1310.44	2	655.218	16004.78	0.0000
H:Cross	13.6614	2	6.83069	166.85	0.0000
I:Rep	1.28425	4	0.321062	7.84	0.0000
Interactions					
AB	0.531077	1	0.531077	12.97	0.0003
AC	6.90459	1	6.90459	168.66	0.0000
AD	0.330212	2	0.165106	4.03	0.0177
AE	0.295131	2	0.147565	3.60	0.0272
AF	1.42348	2	0.71174	17.39	0.0000
AG	52.5022	2	26.2511	641.23	0.0000
AH	0.299526	2	0.149763	3.66	0.0258
AI	0.609992	4	0.152498	3.73	0.0049
BC	6.91643	1	6.91643	168.95	0.0000
BD	0.590489	2	0.295245	7.21	0.0007
BE	1.02873	2	0.514366	12.56	0.0000
BF	0.161732	2	0.080866	1.98	0.1387
BG	0.67392	2	0.33696	8.23	0.0003
BH	0.0510211	2	0.0255105	0.62	0.5363
BI	0.370119	4	0.0925297	2.26	0.0601
CD	5.7089	2	2.85445	69.72	0.0000
CE	1.29974	2	0.649869	15.87	0.0000
CF	2.53179	2	1.26589	30.92	0.0000
CG	3.04683	2	1.52341	37.21	0.0000
CH	0.110518	2	0.0552589	1.35	0.2593
CI	0.362189	4	0.0905472	2.21	0.0651
DE	12.6098	4	3.15244	77.00	0.0000
DF	0.336991	4	0.0842478	2.06	0.0835
DG	0.0336013	4	0.00840032	0.21	0.9356
DH	0.130036	4	0.032509	0.79	0.5288

DI	0.185477	8	0.0231846	0.57	0.8063
EF	0.109149	4	0.0272873	0.67	0.6152
EG	0.39749	4	0.0993725	2.43	0.0457
EH	0.121935	4	0.0304837	0.74	0.5614
EI	0.231152	8	0.028894	0.71	0.6868
FG	2.14082	4	0.535205	13.07	0.0000
FH	0.237482	4	0.0593705	1.45	0.2146
FI	0.626715	8	0.0783394	1.91	0.0535
GH	2.10206	4	0.525516	12.84	0.0000
GI	0.472164	8	0.0590205	1.44	0.1734
HI	0.17387	8	0.0217338	0.53	0.8342
Residual	789.26	19279	0.0409389		
Total (corrected)	2500.89	19439			

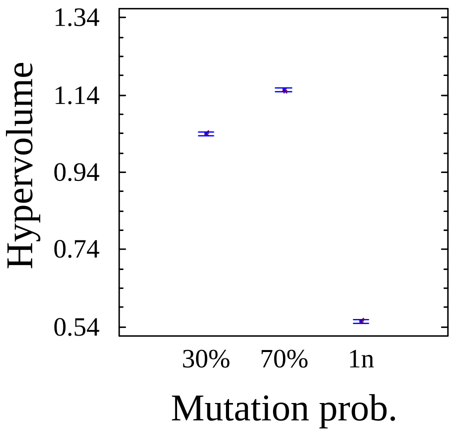


Figure 8.5: Factor means and 99% Tukey confidence intervals for the mutation probability in NSGA-II; large instances. (Higher is better.)

Table 8.3: Calibration of NSGA-II. Table of means and 99% confidence intervals for the large instances.

Level	**Count**	**Mean**	**Stnd. Error**	**Lower Limit**	**Upper Limit**
Grand mean	19440	0.915288			
n					
50	9720	0.973447	0.00205227	0.968161	0.978733
100	9720	0.857129	0.00205227	0.851843	0.862415
m					
4	9720	0.919556	0.00205227	0.91427	0.924842
8	9720	0.91102	0.00205227	0.905734	0.916306
m_i					
2	9720	0.997692	0.00205227	0.992405	1.00298
4	9720	0.832884	0.00205227	0.827598	0.838171
T					
0.2	6480	0.920936	0.00251351	0.914461	0.92741
0.4	6480	0.910247	0.00251351	0.903773	0.916721
0.6	6480	0.914681	0.00251351	0.908207	0.921155
R					
0.2	6480	0.922653	0.00251351	0.916179	0.929128
0.6	6480	0.901019	0.00251351	0.894544	0.907493
1	6480	0.922192	0.00251351	0.915717	0.928666
Pop					
10	6480	0.984202	0.00251351	0.977728	0.990677
30	6480	0.923871	0.00251351	0.917396	0.930345
100	6480	0.837791	0.00251351	0.831317	0.844265
Mut					
0.3	6480	1.03952	0.00251351	1.03304	1.04599
0.7	6480	1.15241	0.00251351	1.14593	1.15888
1n	6480	0.553943	0.00251351	0.547468	0.560417
Cross					
0.5	6480	0.94874	0.00251351	0.942266	0.955215
0.7	6480	0.913219	0.00251351	0.906745	0.919693
0.9	6480	0.883905	0.00251351	0.87743	0.890379
Rep					
1	3888	0.911915	0.00324493	0.903557	0.920274
2	3888	0.909108	0.00324493	0.90075	0.917466
3	3888	0.930821	0.00324493	0.922463	0.93918
4	3888	0.915582	0.00324493	0.907224	0.92394
5	3888	0.909013	0.00324493	0.900655	0.917371

n by Pop						
50	10	3240	1.05437	0.00355464	1.04521	1.06352
50	30	3240	0.977335	0.00355464	0.968179	0.986491
50	100	3240	0.888638	0.00355464	0.879482	0.897794
100	10	3240	0.914036	0.00355464	0.90488	0.923193
100	30	3240	0.870407	0.00355464	0.861251	0.879563
100	100	3240	0.786944	0.00355464	0.777788	0.7961
n by Mut						
50	0.3	3240	1.07153	0.00355464	1.06238	1.08069
50	0.7	3240	1.16415	0.00355464	1.15499	1.17331
50	1n	3240	0.684658	0.00355464	0.675502	0.693814
100	0.3	3240	1.0075	0.00355464	0.998341	1.01665
100	0.7	3240	1.14066	0.00355464	1.13151	1.14982
100	1n	3240	0.423227	0.00355464	0.414071	0.432384
n by Cross						
50	0.5	3240	1.00169	0.00355464	0.992536	1.01085
50	0.7	3240	0.975648	0.00355464	0.966492	0.984804
50	0.9	3240	0.943	0.00355464	0.933844	0.952157
100	0.5	3240	0.895788	0.00355464	0.886632	0.904944
100	0.7	3240	0.85079	0.00355464	0.841634	0.859946
100	0.9	3240	0.824809	0.00355464	0.815653	0.833965
m by Pop						
4	10	3240	0.991792	0.00355464	0.982636	1.00095
4	30	3240	0.928528	0.00355464	0.919371	0.937684
4	100	3240	0.838348	0.00355464	0.829192	0.847504
8	10	3240	0.976612	0.00355464	0.967456	0.985768
8	30	3240	0.919214	0.00355464	0.910058	0.92837
8	100	3240	0.837234	0.00355464	0.828078	0.84639
m by Mut						
4	0.3	3240	1.04947	0.00355464	1.04031	1.05863
4	0.7	3240	1.1591	0.00355464	1.14994	1.16825
4	1n	3240	0.5501	0.00355464	0.540944	0.559256
8	0.3	3240	1.02956	0.00355464	1.0204	1.03872
8	0.7	3240	1.14571	0.00355464	1.13656	1.15487
8	1n	3240	0.557786	0.00355464	0.548629	0.566942
m by Cross						
4	0.5	3240	0.954345	0.00355464	0.945189	0.963501
4	0.7	3240	0.918429	0.00355464	0.909273	0.927586
4	0.9	3240	0.885893	0.00355464	0.876737	0.895049
8	0.5	3240	0.943135	0.00355464	0.933979	0.952291
8	0.7	3240	0.908009	0.00355464	0.898852	0.917165

8	0.9	3240	0.881917	0.00355464	0.87276	0.891073
m_i by Pop						
2	10	3240	1.0771	0.00355464	1.06794	1.08626
2	30	3240	1.01165	0.00355464	1.00249	1.0208
2	100	3240	0.904329	0.00355464	0.895172	0.913485
4	10	3240	0.891305	0.00355464	0.882149	0.900461
4	30	3240	0.836095	0.00355464	0.826939	0.845251
4	100	3240	0.771253	0.00355464	0.762097	0.780409
m_i by Mut						
2	0.3	3240	1.1356	0.00355464	1.12644	1.14476
2	0.7	3240	1.2377	0.00355464	1.22854	1.24686
2	1n	3240	0.619774	0.00355464	0.610618	0.62893
4	0.3	3240	0.94343	0.00355464	0.934274	0.952586
4	0.7	3240	1.06711	0.00355464	1.05796	1.07627
4	1n	3240	0.488112	0.00355464	0.478955	0.497268
m_i by Cross						
2	0.5	3240	1.0289	0.00355464	1.01974	1.03805
2	0.7	3240	0.998924	0.00355464	0.989768	1.00808
2	0.9	3240	0.965253	0.00355464	0.956097	0.97441
4	0.5	3240	0.868583	0.00355464	0.859427	0.877739
4	0.7	3240	0.827514	0.00355464	0.818358	0.836671
4	0.9	3240	0.802556	0.00355464	0.7934	0.811712

The other parameters are fixed similarly. The population size, which is the next algorithm parameter in importance, is fixed to the smallest value. This fixes the population at a size of 10 individuals, as we can see in Figure 8.6. The last parameter to be fixed is the crossover probability. Figure 8.7 shows that the smallest probability, namely 50%, yields the best results. This fixes the last instance parameter for NSGA-II.

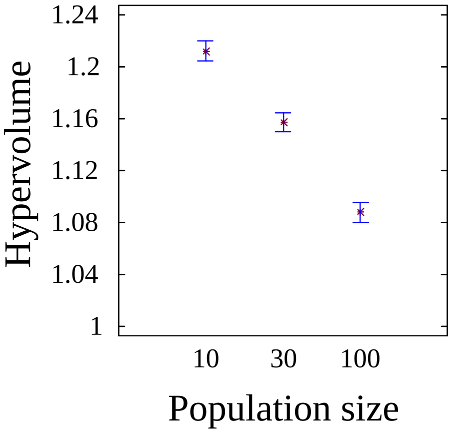


Figure 8.6: Factor means and 99% Tukey confidence intervals for the population size in NSGA-II; large instances. (Higher is better.)

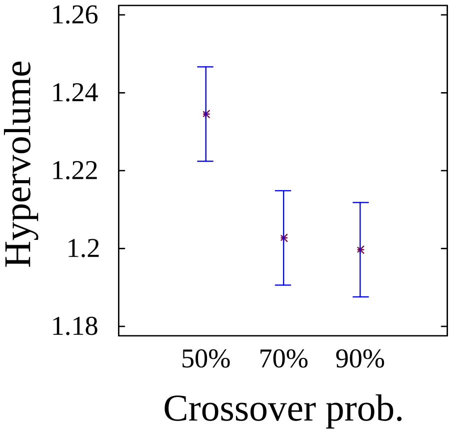


Figure 8.7: Factor means and 99% Tukey confidence intervals for the crossover probability in NSGA-II; large instances. (Higher is better.)

RIPG also has three parameters that should be calibrated. The parameters and its levels are:

- Number of jobs destructed in IG phase:
    - 3 jobs,
    - 5 jobs,
    - 10 jobs.
- Neighbouring jobs considered in local search:
    - 3 jobs at both sides,
    - 5 jobs at both sides,
    - No local search.
- Moment of restart:
    - After 10 iterations without change in the population,
    - After $2n$ iterations without change in the population,
    - No restart.

Since the number of parameter configurations is equal to the number of different setting for NSGA-II, the CPU time required for this calibration is equal as well: 729 hours. The ANOVA results are given in Table 8.4. In that table we can see that the restart parameter has the highes F-value and should therefore be fixed first. In Figure 8.8, one can see that applying a restart when $2n$ iterations have been done without any improvement in the populations, is the best option for RIPG. This is confirmed by the data in Table 8.5.

Table 8.4: Analysis of Variance for the Hypervolume - Calibration of RIPG.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	19.5218	1	19.5218	467.86	0.0000
B:m	5.66431	1	5.66431	135.75	0.0000
C:m_i	0.198711	1	0.198711	4.76	0.0291
D:T	2.12475	2	1.06238	25.46	0.0000
E:R	0.0691669	2	0.0345834	0.83	0.4366
F:Restart	623.177	2	311.589	7467.58	0.0000
G:Greedy	73.1477	2	36.5738	876.53	0.0000
H:LocalSearch	5.3514	2	2.6757	64.13	0.0000
I:Rep	0.327426	4	0.0818565	1.96	0.0974
Interactions					
AB	7.45596	1	7.45596	178.69	0.0000
AC	1.78673	1	1.78673	42.82	0.0000
AD	4.272	2	2.136	51.19	0.0000
AE	0.300696	2	0.150348	3.60	0.0273
AF	186.735	2	93.3677	2237.66	0.0000
AG	67.2393	2	33.6197	805.73	0.0000
AH	9.90714	2	4.95357	118.72	0.0000
AI	0.131665	4	0.0329162	0.79	0.5322
BC	4.9609	1	4.9609	118.89	0.0000
BD	2.66438	2	1.33219	31.93	0.0000
BE	4.74424	2	2.37212	56.85	0.0000
BF	4.93911	2	2.46955	59.19	0.0000
BG	0.644688	2	0.322344	7.73	0.0004
BH	0.0238262	2	0.0119131	0.29	0.7516
BI	0.0982541	4	0.0245635	0.59	0.6708
CD	2.08448	2	1.04224	24.98	0.0000
CE	0.271025	2	0.135512	3.25	0.0389
CF	33.5871	2	16.7935	402.48	0.0000
CG	60.9849	2	30.4924	730.79	0.0000
CH	1.41409	2	0.707046	16.95	0.0000
CI	0.129239	4	0.0323098	0.77	0.5417
DE	1.97393	4	0.493482	11.83	0.0000
DF	0.154316	4	0.0385791	0.92	0.4484
DG	1.02371	4	0.255926	6.13	0.0001
DH	0.342078	4	0.0855194	2.05	0.0846

DI	0.37819	8	0.0472738	1.13	0.3370
EF	0.0790415	4	0.0197604	0.47	0.7552
EG	0.113566	4	0.0283914	0.68	0.6054
EH	0.280328	4	0.070082	1.68	0.1516
EI	0.236372	8	0.0295465	0.71	0.6847
FG	49.1041	4	12.276	294.21	0.0000
FH	9.82821	4	2.45705	58.89	0.0000
FI	0.0902494	8	0.0112812	0.27	0.9756
GH	9.58266	4	2.39566	57.41	0.0000
GI	0.445293	8	0.0556616	1.33	0.2211
HI	0.256038	8	0.0320047	0.77	0.6320
Residual	804.427	19279	0.0417255		
Total (corrected)	2019.32	19439			

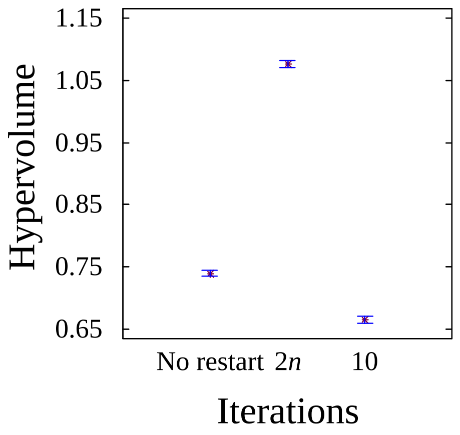


Figure 8.8: Means and 99% Tukey confidence intervals between the number of iterations without population improvement done before restart in RIPG; large instances. (Higher is better.)

Table 8.5: Calibration of RIPG. Table of means and 99% confidence intervals for the large instances.

Level		**Count**	**Mean**	**Stnd.**	**Lower**	**Upper**
				Error	**Limit**	**Limit**
Grand mean		19440	0.827002			
n						
50		9720	0.858692	0.0020719	0.853355	0.864028
100		9720	0.795313	0.0020719	0.789976	0.80065
m						
4		9720	0.809933	0.0020719	0.804596	0.81527
8		9720	0.844072	0.0020719	0.838735	0.849409
m_i						
2		9720	0.823805	0.0020719	0.818468	0.829142
4		9720	0.830199	0.0020719	0.824863	0.835536
T						
0.2		6480	0.82448	0.00253754	0.817944	0.831017
0.4		6480	0.815647	0.00253754	0.80911	0.822183
0.6		6480	0.84088	0.00253754	0.834344	0.847416
R						
0.2		6480	0.824407	0.00253754	0.81787	0.830943
0.6		6480	0.828833	0.00253754	0.822296	0.835369
1		6480	0.827768	0.00253754	0.821232	0.834304
Restart						
R0		6480	0.739553	0.00253754	0.733016	0.746089
R1		6480	1.07652	0.00253754	1.06998	1.08305
R2		6480	0.664938	0.00253754	0.658402	0.671474
Greedy						
G0		6480	0.740461	0.00253754	0.733925	0.746997
G1		6480	0.865073	0.00253754	0.858536	0.871609
G2		6480	0.875474	0.00253754	0.868937	0.88201
LocalSearch						
LS0		6480	0.848115	0.00253754	0.841579	0.854651
LS1		6480	0.80758	0.00253754	0.801044	0.814116
LS2		6480	0.825312	0.00253754	0.818776	0.831848
n by Restart						
50	R0	3240	0.642793	0.00358863	0.633549	0.652036
50	R1	3240	1.12733	0.00358863	1.11808	1.13657
50	R2	3240	0.805954	0.00358863	0.79671	0.815198
100	R0	3240	0.836313	0.00358863	0.827069	0.845557
100	R1	3240	1.0257	0.00358863	1.01646	1.03495

100	R2	3240	0.523922	0.00358863	0.514679	0.533166
n by Greedy						
50	G0	3240	0.839977	0.00358863	0.830733	0.849221
50	G1	3240	0.904536	0.00358863	0.895292	0.91378
50	G2	3240	0.831562	0.00358863	0.822318	0.840805
100	G0	3240	0.640945	0.00358863	0.631701	0.650189
100	G1	3240	0.825609	0.00358863	0.816365	0.834853
100	G2	3240	0.919385	0.00358863	0.910142	0.928629
n by LocalSearch						
50	LS0	3240	0.848022	0.00358863	0.838779	0.857266
50	LS1	3240	0.852539	0.00358863	0.843296	0.861783
50	LS2	3240	0.875513	0.00358863	0.866269	0.884757
100	LS0	3240	0.848208	0.00358863	0.838964	0.857452
100	LS1	3240	0.762621	0.00358863	0.753377	0.771865
100	LS2	3240	0.775111	0.00358863	0.765867	0.784354
m by Restart						
4	R0	3240	0.739188	0.00358863	0.729945	0.748432
4	R1	3240	1.0642	0.00358863	1.05496	1.07344
4	R2	3240	0.626408	0.00358863	0.617165	0.635652
8	R0	3240	0.739917	0.00358863	0.730673	0.749161
8	R1	3240	1.08883	0.00358863	1.07959	1.09807
8	R2	3240	0.703468	0.00358863	0.694224	0.712712
m by Greedy						
4	G0	3240	0.731012	0.00358863	0.721768	0.740256
4	G1	3240	0.84668	0.00358863	0.837436	0.855924
4	G2	3240	0.852106	0.00358863	0.842862	0.86135
8	G0	3240	0.74991	0.00358863	0.740666	0.759154
8	G1	3240	0.883465	0.00358863	0.874221	0.892709
8	G2	3240	0.898841	0.00358863	0.889597	0.908085
m by LocalSearch						
4	LS0	3240	0.832316	0.00358863	0.823072	0.84156
4	LS1	3240	0.789083	0.00358863	0.779839	0.798327
4	LS2	3240	0.808399	0.00358863	0.799155	0.817643
8	LS0	3240	0.863914	0.00358863	0.85467	0.873158
8	LS1	3240	0.826077	0.00358863	0.816833	0.835321
8	LS2	3240	0.842225	0.00358863	0.832981	0.851468
m_i by Restart						
2	R0	3240	0.769098	0.00358863	0.759855	0.778342
2	R1	3240	1.09923	0.00358863	1.08998	1.10847
2	R2	3240	0.60309	0.00358863	0.593847	0.612334
4	R0	3240	0.710007	0.00358863	0.700763	0.719251

4	R1	3240	1.05381	0.00358863	1.04456	1.06305
4	R2	3240	0.726786	0.00358863	0.717542	0.73603
m_i by Greedy						
2	G0	3240	0.800621	0.00358863	0.791377	0.809865
2	G1	3240	0.871368	0.00358863	0.862124	0.880611
2	G2	3240	0.799427	0.00358863	0.790183	0.808671
4	G0	3240	0.680301	0.00358863	0.671057	0.689545
4	G1	3240	0.858777	0.00358863	0.849534	0.868021
4	G2	3240	0.95152	0.00358863	0.942276	0.960764
m_i by LocalSearch						
2	LS0	3240	0.833907	0.00358863	0.824663	0.843151
2	LS1	3240	0.805624	0.00358863	0.79638	0.814868
2	LS2	3240	0.831884	0.00358863	0.822641	0.841128
4	LS0	3240	0.862323	0.00358863	0.853079	0.871567
4	LS1	3240	0.809536	0.00358863	0.800292	0.81878
4	LS2	3240	0.818739	0.00358863	0.809496	0.827983
Restart by Greedy						
R0	G0	2160	0.667793	0.00439515	0.656472	0.679115
R0	G1	2160	0.772588	0.00439515	0.761267	0.783909
R0	G2	2160	0.778277	0.00439515	0.766956	0.789598
R1	G0	2160	1.05433	0.00439515	1.04301	1.06565
R1	G1	2160	1.1224	0.00439515	1.11108	1.13372
R1	G2	2160	1.05281	0.00439515	1.04149	1.06413
R2	G0	2160	0.499256	0.00439515	0.487935	0.510577
R2	G1	2160	0.700227	0.00439515	0.688906	0.711548
R2	G2	2160	0.795331	0.00439515	0.78401	0.806652
Restart by LocalSearch						
R0	LS0	2160	0.735482	0.00439515	0.724161	0.746803
R0	LS1	2160	0.732247	0.00439515	0.720926	0.743569
R0	LS2	2160	0.750929	0.00439515	0.739608	0.76225
R1	LS0	2160	1.07828	0.00439515	1.06696	1.0896
R1	LS1	2160	1.07092	0.00439515	1.0596	1.08225
R1	LS2	2160	1.08035	0.00439515	1.06902	1.09167
R2	LS0	2160	0.730586	0.00439515	0.719264	0.741907
R2	LS1	2160	0.619568	0.00439515	0.608247	0.630889
R2	LS2	2160	0.644661	0.00439515	0.63334	0.655982
Greedy by LocalSearch						
G0	LS0	2160	0.796469	0.00439515	0.785148	0.80779
G0	LS1	2160	0.706137	0.00439515	0.694816	0.717458
G0	LS2	2160	0.718776	0.00439515	0.707455	0.730098
G1	LS0	2160	0.892046	0.00439515	0.880725	0.903368

G1	LS1	2160	0.844882	0.00439515	0.833561	0.856203
G1	LS2	2160	0.858289	0.00439515	0.846968	0.86961
G2	LS0	2160	0.85583	0.00439515	0.844508	0.867151
G2	LS1	2160	0.871721	0.00439515	0.860399	0.883042
G2	LS2	2160	0.89887	0.00439515	0.887549	0.910192

If we generate another ANOVA with fixed restart configuration (see Table B.7), the highest F-value is obtained by the interaction between the number of destructed jobs in the IG phase of the algorithm on the one hand and the number of machines per stage on the other hand. This interaction is shown in Figure 8.9. Destructing only 3 jobs appears to have a bad performance for 4 machines per stage, while a destruction of 10 jobs leads to bad results for 2 machines per stage. However, since we are interested in which parameter setting works best in general, for fixing the destruction setting we concentrate on the means plot in Figure 8.10. From there we can conclude that the best value for the destruction is a number of 5 jobs. Now, the local search parameter does not have any significant influence on the results any more. We therefore choose to keep the algorithm as simple as possible and exclude the local search phase.

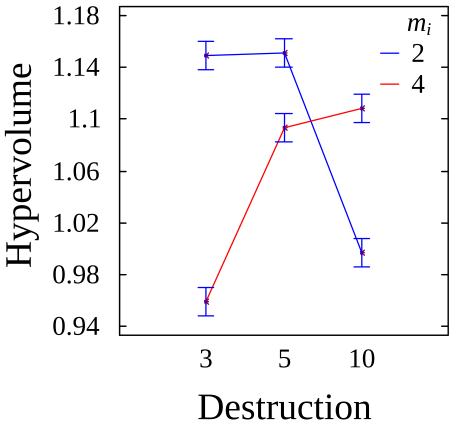


Figure 8.9: Interaction and 99% Tukey confidence intervals between the number of jobs destructed in the IG phase and the number of machines per stage; large instances. (Higher is better.)

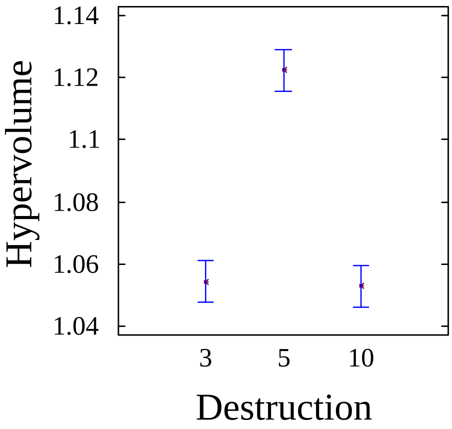


Figure 8.10: Means and 99% Tukey confidence intervals for the number of jobs destructed in the IG phase; large instances. (Higher is better.)

8.5.2. Comparison among multi-objective algorithms

After calibration of both algorithms in the previous subsection, the two presented algorithms can now be compared with their final parameter settings. We will first compare them for the large instances. Instead of the 144 test instances used for the calibration, the algorithms are now run on the 288 final instances. For a better precision, 10 replicates are done for each algorithm on each instance. The allowed CPU time is the same as in the calibration; t is fixed at 100 milliseconds, which means that the largest instances of 100 jobs and 32 machines distributed over 8 stages are processed for 320 seconds.

Table 8.6 gives the ANOVA for the hypervolume indicator on the set of large instances. The used method is most significant in this analysis, which facilitates to draw clear conclusions. In Figure 8.11 it becomes clear that NSGA-II outperforms RIPG for the large instances with a big difference. The detailed numbers are given in Table 8.7.

Source	Sum of Squares	Degrees of freedom	Mean Square	F-Ratio	P-Value
Main effects					
A:n	7.10795	1	7.10795	55.08	0.0000
B:m	0.102194	1	0.102194	0.79	0.3735
C:m_i	5.86954	1	5.86954	45.48	0.0000
D:T	1.37121	2	0.685604	5.31	0.0050
E:R	1.18473	2	0.592367	4.59	0.0102
F:Method	64.8675	1	64.8675	502.68	0.0000
G:Rep	0.688255	9	0.0764728	0.59	0.8042
Interactions					
AB	0.0186164	1	0.0186164	0.14	0.7041
AC	1.24052	1	1.24052	9.61	0.0019
AD	4.78962	2	2.39481	18.56	0.0000
AE	0.489927	2	0.244964	1.90	0.1499
AF	6.59181	1	6.59181	51.08	0.0000
AG	1.23904	9	0.137672	1.07	0.3839
BC	0.718889	1	0.718889	5.57	0.0183
BD	0.407864	2	0.203932	1.58	0.2060
BE	0.229893	2	0.114946	0.89	0.4104
BF	4.49792	1	4.49792	34.86	0.0000
BG	0.615938	9	0.0684375	0.53	0.8535
CD	2.91408	2	1.45704	11.29	0.0000
CE	4.70003	2	2.35002	18.21	0.0000
CF	17.5095	1	17.5095	135.69	0.0000
CG	0.578191	9	0.0642434	0.50	0.8770
DE	1.76385	4	0.440962	3.42	0.0085
DF	0.414087	2	0.207043	1.60	0.2011
DG	2.06203	18	0.114557	0.89	0.5940
EF	0.702781	2	0.351391	2.72	0.0658
EG	0.555534	18	0.030863	0.24	0.9996
FG	0.953314	9	0.105924	0.82	0.5969
Residual	728.322	5644	0.129044		
Total (corrected)	862.507	5759			

Table 8.6: Analysis of Variance for the Hypervolume - comparison of NSGA-II and RIPG for the set of large instances.

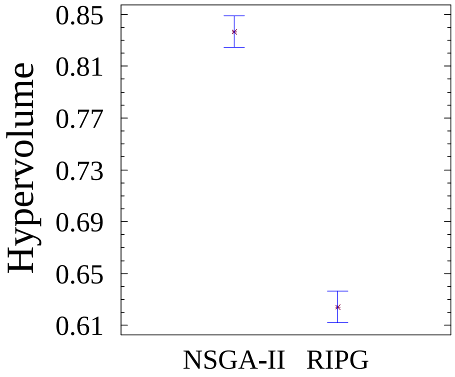


Figure 8.11: Hypervolume means and 99% Tukey confidence intervals for the multi-objective algorithms; large instances. (Higher is better.)

Table 8.7: Hypervolume means and 99% Tukey intervals - comparison of NSGA-II and RIPG for the set of large instances.

Level	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
Grand mean	5760	0.730402			
n					
50	2880	0.765531	0.00669379	0.748289	0.782773
100	2880	0.695274	0.00669379	0.678032	0.712516
m					
4	2880	0.72619	0.00669379	0.708948	0.743432
8	2880	0.734614	0.00669379	0.717372	0.751857
m_i					
2	2880	0.69848	0.00669379	0.681238	0.715722
4	2880	0.762324	0.00669379	0.745082	0.779566
T					
0.2	1920	0.747478	0.00819818	0.726361	0.768595
0.4	1920	0.733629	0.00819818	0.712511	0.754746
0.6	1920	0.7101	0.00819818	0.688983	0.731217
R					
0.2	1920	0.746132	0.00819818	0.725014	0.767249
0.6	1920	0.733627	0.00819818	0.71251	0.754744

1		1920	0.711449	0.00819818	0.690331	0.732566
Method						
NSGA-II		2880	0.836524	0.00669379	0.819281	0.853766
RIPG		2880	0.624281	0.00669379	0.607039	0.641523
n by Method						
50	NSGA-II	1440	0.905481	0.00946644	0.881097	0.929865
50	RIPG	1440	0.625581	0.00946644	0.601197	0.649965
100	NSGA-II	1440	0.767566	0.00946644	0.743182	0.79195
100	RIPG	1440	0.622982	0.00946644	0.598598	0.647366
m by Method						
4	NSGA-II	1440	0.860256	0.00946644	0.835872	0.88464
4	RIPG	1440	0.592125	0.00946644	0.567741	0.616509
8	NSGA-II	1440	0.812791	0.00946644	0.788407	0.837175
8	RIPG	1440	0.656438	0.00946644	0.632054	0.680822
m_i by Method						
2	NSGA-II	1440	0.859736	0.00946644	0.835352	0.88412
2	RIPG	1440	0.537224	0.00946644	0.51284	0.561608
4	NSGA-II	1440	0.813311	0.00946644	0.788927	0.837695
4	RIPG	1440	0.711338	0.00946644	0.686954	0.735722
T by Method						
0.2	NSGA-II	960	0.844836	0.011594	0.814972	0.8747
0.2	RIPG	960	0.65012	0.011594	0.620256	0.679984
0.4	NSGA-II	960	0.837043	0.011594	0.807179	0.866908
0.4	RIPG	960	0.630214	0.011594	0.60035	0.660078
0.6	NSGA-II	960	0.827691	0.011594	0.797827	0.857555
0.6	RIPG	960	0.592509	0.011594	0.562645	0.622374
R by Method						
0.2	NSGA-II	960	0.837358	0.011594	0.807493	0.867222
0.2	RIPG	960	0.654906	0.011594	0.625041	0.68477
0.6	NSGA-II	960	0.84312	0.011594	0.813255	0.872984
0.6	RIPG	960	0.624134	0.011594	0.59427	0.653998
1	NSGA-II	960	0.829094	0.011594	0.799229	0.858958
1	RIPG	960	0.593804	0.011594	0.56394	0.623668

For validation on the results, the Epsilon indicator can also be consulted for the same data. In Figure 8.12 is shown that the ϵ-indicator support the conclusion drawn by consulting the hypervolume. Recall that a lower ϵ-indicator corresponds to a better Pareto approximation front.

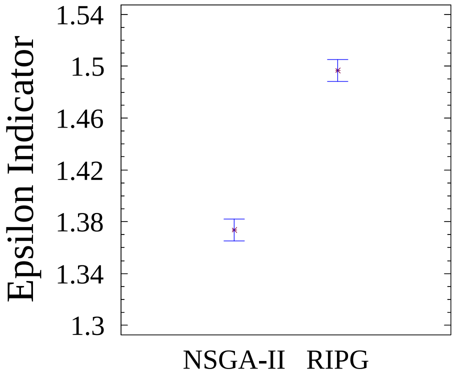


Figure 8.12: ϵ-indicator means and 99% Tukey confidence intervals for the multi-objective algorithms; large instances. (Lower is better.)

For the small instances, the same analysis can be done. The two algorithms are compared with the same parameter setting as for the large instances. This time, the input data is the set of 462 instances with 5 to 15 jobs. The highest CPU time per run is limited by $n \sum_i m_i{\cdot}t = 15{\cdot}3{\cdot}3{\cdot}100 =$ 13,500 milliseconds, which is equal to 13.5 seconds.

In Table 8.8, the ANOVA results for the hypervolume indicator are shown. One can see that none of the interactions has a higher F-value than the F-value of the used method. In Figure 8.13 the comparison of NSGA-II and RIPG is shown graphically. In contrast to the results for the large instances, RIPG outperforms NSGA-II for the small instances tested here. The more deterministic approach that requires more objective evaluations is inefficient and slow for the large instances, but appears to be very effective for the small instances. The most important instance factor is the number of jobs n. The interaction between the used algorithm and n is shown in Figure 8.14. We can see that no significant difference can be found for 5 or 7 jobs; those instances are too easy. For 7 to 15 jobs, RIPG is the better algorithm. The underlying numbers for both figures can be found in Table 8.9.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	134.82	5	26.9639	148.14	0.0000
B:m	10.943	1	10.943	60.12	0.0000
C:T	0.157465	2	0.0787327	0.43	0.6489
D:R	0.81652	2	0.40826	2.24	0.1062
E:Method	22.515	1	22.515	123.70	0.0000
F:Repetition	0.260285	9	0.0289205	0.16	0.9976
Interactions					
AB	55.177	5	11.0354	60.63	0.0000
AC	13.514	10	1.3514	7.42	0.0000
AD	11.1208	10	1.11208	6.11	0.0000
AE	12.7651	5	2.55302	14.03	0.0000
AF	2.2069	45	0.0490422	0.27	1.0000
BC	0.0251207	2	0.0125604	0.07	0.9333
BD	1.64656	2	0.823282	4.52	0.0109
BE	1.0199	1	1.0199	5.60	0.0179
BF	0.0852685	9	0.00947427	0.05	1.0000
CD	7.47692	4	1.86923	10.27	0.0000
CE	0.236158	2	0.118079	0.65	0.5227
CF	0.357004	18	0.0198336	0.11	1.0000
DE	0.164898	2	0.0824492	0.45	0.6357
DF	0.262171	18	0.0145651	0.08	1.0000
EF	0.339626	9	0.0377362	0.21	0.9934
Residual	1542.94	8477	0.182014		
Total (corrected)	1818.85	8639			

Table 8.8: Analysis of Variance for the Hypervolume - comparison of NSGA-II and RIPG for the set of small instances.

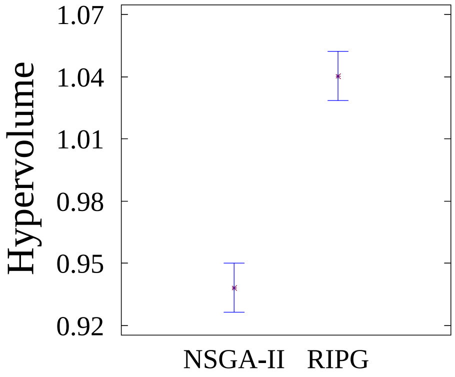


Figure 8.13: Hypervolume indicator means and 99% Tukey confidence intervals for the multi-objective algorithms; small instances. (Higher is better.)

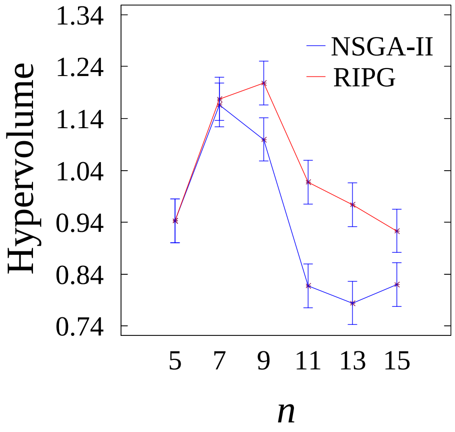


Figure 8.14: Interaction for the hypervolume indicator and 99% Tukey confidence intervals between the algorithm and the number of jobs; small instances. (Higher is better.)

Table 8.9: Hypervolume comparison of NSGA-II and RIPG. Table of means and 99% confidence intervals for the small instances.

Level		Count	Mean	Stnd.	Lower	Upper
				Error	Limit	Limit
Grand mean		8640	0.989299			
n						
5		1440	0.942778	0.0112427	0.913818	0.971737
7		1440	1.1715	0.0112427	1.14254	1.20046
9		1440	1.1537	0.0112427	1.12474	1.18265
11		1440	0.917478	0.0112427	0.888518	0.946437
13		1440	0.878938	0.0112427	0.849978	0.907897
15		1440	0.871404	0.0112427	0.842445	0.900364
m						
2		4320	0.95371	0.00649099	0.93699	0.97043
3		4320	1.02489	0.00649099	1.00817	1.04161
T						
0.2		2880	0.994609	0.00794981	0.974131	1.01509
0.4		2880	0.984156	0.00794981	0.963678	1.00463
0.6		2880	0.989132	0.00794981	0.968655	1.00961
R						
0.2		2880	0.999117	0.00794981	0.978639	1.01959
0.6		2880	0.992725	0.00794981	0.972247	1.0132
1		2880	0.976055	0.00794981	0.955578	0.996533
Method						
NSGA-II		4320	0.938251	0.00649099	0.921531	0.954971
RIPG		4320	1.04035	0.00649099	1.02363	1.05707
n by Method						
5	NSGA-II	720	0.942778	0.0158996	0.901823	0.983733
5	RIPG	720	0.942778	0.0158996	0.901823	0.983733
7	NSGA-II	720	1.16537	0.0158996	1.12441	1.20632
7	RIPG	720	1.17763	0.0158996	1.13668	1.21859
9	NSGA-II	720	1.09949	0.0158996	1.05854	1.14045
9	RIPG	720	1.2079	0.0158996	1.16694	1.24885
11	NSGA-II	720	0.817939	0.0158996	0.776984	0.858894
11	RIPG	720	1.01702	0.0158996	0.976062	1.05797
13	NSGA-II	720	0.784539	0.0158996	0.743584	0.825494
13	RIPG	720	0.973336	0.0158996	0.932382	1.01429
15	NSGA-II	720	0.819392	0.0158996	0.778437	0.860347
15	RIPG	720	0.923417	0.0158996	0.882462	0.964372

m by Method						
2	NSGA-II	2160	0.891797	0.00917965	0.868152	0.915443
2	RIPG	2160	1.01562	0.00917965	0.991978	1.03927
3	NSGA-II	2160	0.984704	0.00917965	0.961059	1.00835
3	RIPG	2160	1.06507	0.00917965	1.04143	1.08872
T by Method						
0.2	NSGA-II	1440	0.93663	0.0112427	0.90767	0.965589
0.2	RIPG	1440	1.05259	0.0112427	1.02363	1.08155
0.4	NSGA-II	1440	0.938803	0.0112427	0.909843	0.967762
0.4	RIPG	1440	1.02951	0.0112427	1.00055	1.05847
0.6	NSGA-II	1440	0.93932	0.0112427	0.910361	0.96828
0.6	RIPG	1440	1.03894	0.0112427	1.00998	1.0679
R by Method						
0.2	NSGA-II	1440	0.944729	0.0112427	0.91577	0.973689
0.2	RIPG	1440	1.0535	0.0112427	1.02454	1.08246
0.6	NSGA-II	1440	0.938845	0.0112427	0.909885	0.967804
0.6	RIPG	1440	1.0466	0.0112427	1.01765	1.07556
1	NSGA-II	1440	0.931179	0.0112427	0.902219	0.960138
1	RIPG	1440	1.02093	0.0112427	0.991973	1.04989

For the ϵ-indicator the importance of the algorithm is higher than the importance of the number of jobs, as can be learned from Table 8.10. In Figure 8.15, the comparison of NSGA-II and RIPG is shown for the ϵ-indicator. The result is coherent with the hypervolume comparison. Figure 8.16, however, shows some slight differences between the two indicators. For 7 jobs, there is quite some difference in the advantage of RIPG when the ϵ-indicator is used, although the significance intervals overlap. When using the hypervolume, though, the averages practically coincide. Also, the performance of RIPG seems to be practically equal for 5 to 13 jobs, in the case of the ϵ-indicator. For these values of n, $I^1_{|\epsilon|} = 1$ or $I^1_{|\epsilon|} \approx 1$. Recall that $1 \leq I^1_{|\epsilon|} \leq 2$ and that the best possible value for the ϵ-indicator is 1. For the hypervolume indicator, however, the obtained values fluctuate more and the maximum value of 1.44 is not closely approximated in any of the cases. These differences support the earlier claim that the hypervolume indicator represents more information than the ϵ-indicator.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	75.3747	5	15.0749	307.36	0.0000
B:m	0.872704	1	0.872704	17.79	0.0000
C:T	0.239978	2	0.119989	2.45	0.0867
D:R	0.12455	2	0.0622752	1.27	0.2810
E:Method	54.1929	1	54.1929	1104.94	0.0000
F:Repetition	0.57565	9	0.0639611	1.30	0.2287
Interactions					
AB	13.4805	5	2.69611	54.97	0.0000
AC	0.769726	10	0.0769726	1.57	0.1090
AD	0.977938	10	0.0977938	1.99	0.0300
AE	39.0716	5	7.81432	159.33	0.0000
AF	2.01467	45	0.0447705	0.91	0.6386
BC	0.0279469	2	0.0139734	0.28	0.7521
BD	0.0263367	2	0.0131683	0.27	0.7645
BE	0.0271147	1	0.0271147	0.55	0.4572
BF	0.319807	9	0.0355341	0.72	0.6869
CD	1.27444	4	0.318609	6.50	0.0000
CE	0.130026	2	0.0650132	1.33	0.2657
CF	0.277645	18	0.0154247	0.31	0.9974
DE	0.166221	2	0.0831104	1.69	0.1837
DF	0.228771	18	0.0127095	0.26	0.9993
EF	0.546119	9	0.0606799	1.24	0.2668
Residual	415.763	8477	0.049046		
Total (corrected)	606.483	8639			

Table 8.10: Analysis of Variance for the Epsilon indicator - comparison of NSGA-II and RIPG for the set of small instances.

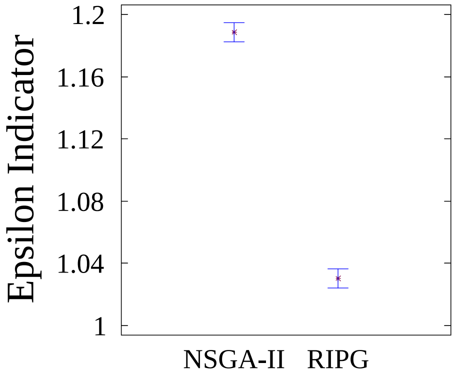


Figure 8.15: ϵ-indicator means and 99% Tukey confidence intervals for the multi-objective algorithms; small instances. (Lower is better.)

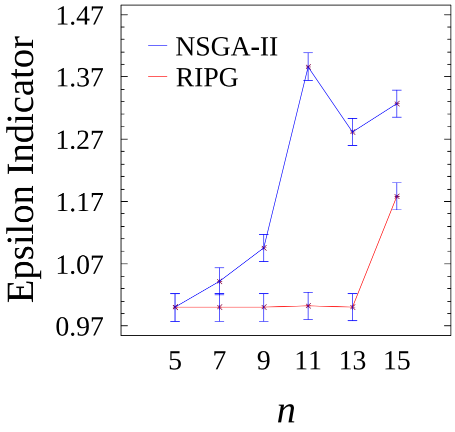


Figure 8.16: Interaction for the ϵ-indicator and 99% Tukey confidence intervals between the algorithm and the number of jobs; small instances. (Lower is better.)

Table 8.11: Epsilon indicator means and 99% Tukey intervals - comparison of NSGA-II and RIPG for the set of small instances.

Level		**Count**	**Mean**	**Stnd.**	**Lower**	**Upper**
				Error	**Limit**	**Limit**
Grand mean		8640	1.10944			
n						
5		1440	1.0	0.00583607	0.984967	1.01503
7		1440	1.02083	0.00583607	1.0058	1.03587
9		1440	1.04765	0.00583607	1.03262	1.06268
11		1440	1.19453	0.00583607	1.1795	1.20956
13		1440	1.14121	0.00583607	1.12618	1.15625
15		1440	1.25243	0.00583607	1.2374	1.26747
m						
2		4320	1.09939	0.00336946	1.09071	1.10807
3		4320	1.11949	0.00336946	1.11081	1.12817
T						
0.2		2880	1.11683	0.00412673	1.1062	1.12746
0.4		2880	1.10664	0.00412673	1.09601	1.11727
0.6		2880	1.10486	0.00412673	1.09423	1.11549
R						
0.2		2880	1.11446	0.00412673	1.10383	1.12509
0.6		2880	1.10858	0.00412673	1.09795	1.11921
1		2880	1.10528	0.00412673	1.09465	1.11591
Method						
NSGA-II		4320	1.18864	0.00336946	1.17996	1.19732
RIPG		4320	1.03025	0.00336946	1.02157	1.03892
n by Method						
5	NSGA-II	720	1.0	0.00825345	0.97874	1.02126
5	RIPG	720	1.0	0.00825345	0.97874	1.02126
7	NSGA-II	720	1.04167	0.00825345	1.02041	1.06293
7	RIPG	720	1.0	0.00825345	0.97874	1.02126
9	NSGA-II	720	1.0953	0.00825345	1.07404	1.11656
9	RIPG	720	1.0	0.00825345	0.97874	1.02126
11	NSGA-II	720	1.38634	0.00825345	1.36508	1.4076
11	RIPG	720	1.00272	0.00825345	0.981458	1.02398
13	NSGA-II	720	1.28169	0.00825345	1.26043	1.30295
13	RIPG	720	1.00074	0.00825345	0.979483	1.022
15	NSGA-II	720	1.32686	0.00825345	1.3056	1.34812
15	RIPG	720	1.17801	0.00825345	1.15675	1.19927

m by Method						
2	NSGA-II	2160	1.18036	0.00476513	1.16809	1.19264
2	RIPG	2160	1.01842	0.00476513	1.00615	1.0307
3	NSGA-II	2160	1.19692	0.00476513	1.18465	1.20919
3	RIPG	2160	1.04207	0.00476513	1.02979	1.05434
T by Method						
0.2	NSGA-II	1440	1.2001	0.00583607	1.18507	1.21513
0.2	RIPG	1440	1.03355	0.00583607	1.01852	1.04858
0.4	NSGA-II	1440	1.18062	0.00583607	1.16559	1.19566
0.4	RIPG	1440	1.03266	0.00583607	1.01763	1.0477
0.6	NSGA-II	1440	1.1852	0.00583607	1.17017	1.20024
0.6	RIPG	1440	1.02452	0.00583607	1.00949	1.03955
R by Method						
0.2	NSGA-II	1440	1.19834	0.00583607	1.1833	1.21337
0.2	RIPG	1440	1.03059	0.00583607	1.01556	1.04563
0.6	NSGA-II	1440	1.18898	0.00583607	1.17394	1.20401
0.6	RIPG	1440	1.02819	0.00583607	1.01316	1.04322
1	NSGA-II	1440	1.17861	0.00583607	1.16358	1.19365
1	RIPG	1440	1.03196	0.00583607	1.01692	1.04699

In order to obtain more information about the differences between the two algorithms, empirical attainment functions can be visualised for some instances. In order to generate the visualisations, 100 replicates are done for each algorithm, for each instance we want to show. Algorithm run covering a point therefore corresponds to 1% in the EAF. The first instance for which the EAF are analysed is an instance of 50 jobs, 4 stages and 2 parallel machines in each stage. We will refer to this instance as example instance 5. In Figure 8.17, the empirical attainment function are shown for NSGA-II. The solid red area is covered in all 100 runs; the solid white area is not covered in any run. The grade of the curved area in between indicates the number of runs in which the objective vectors are attained.

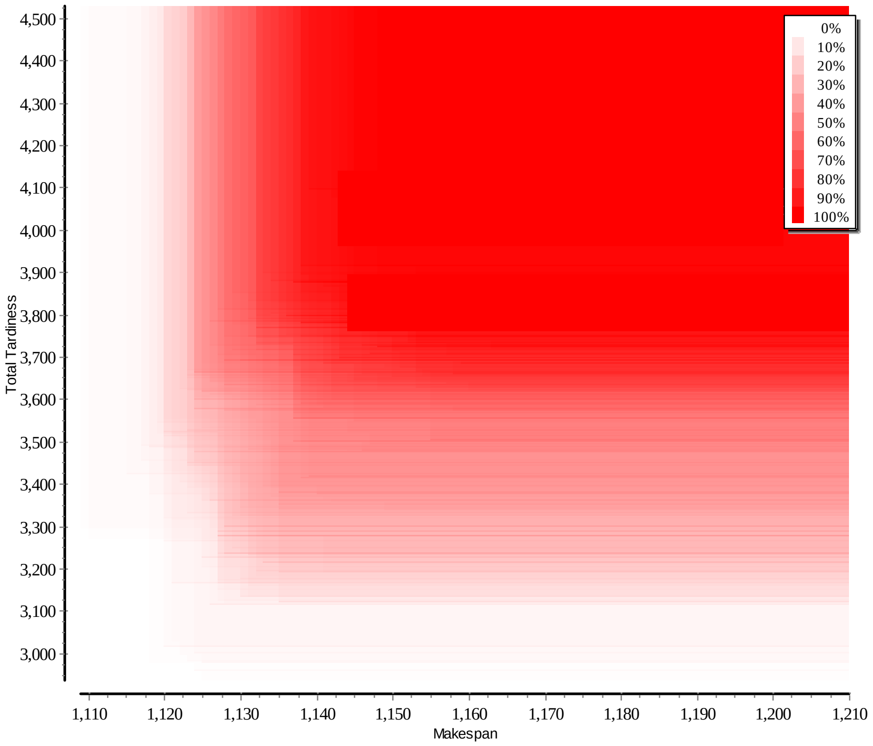


Figure 8.17: Plot of EAF for NSGA-II. Example instance 5 with 50 jobs, 4 stages and 2 machines per stage.

Figure 8.18 shows the empirical attainment functions graphically for RIPG for the same example instance. It is easy to see that RIPG performs worse for this example instance. Both on total tardiness and on makespan, NSGA-II achieves to cover the area with a higher probability, if we compare one figure with the other.

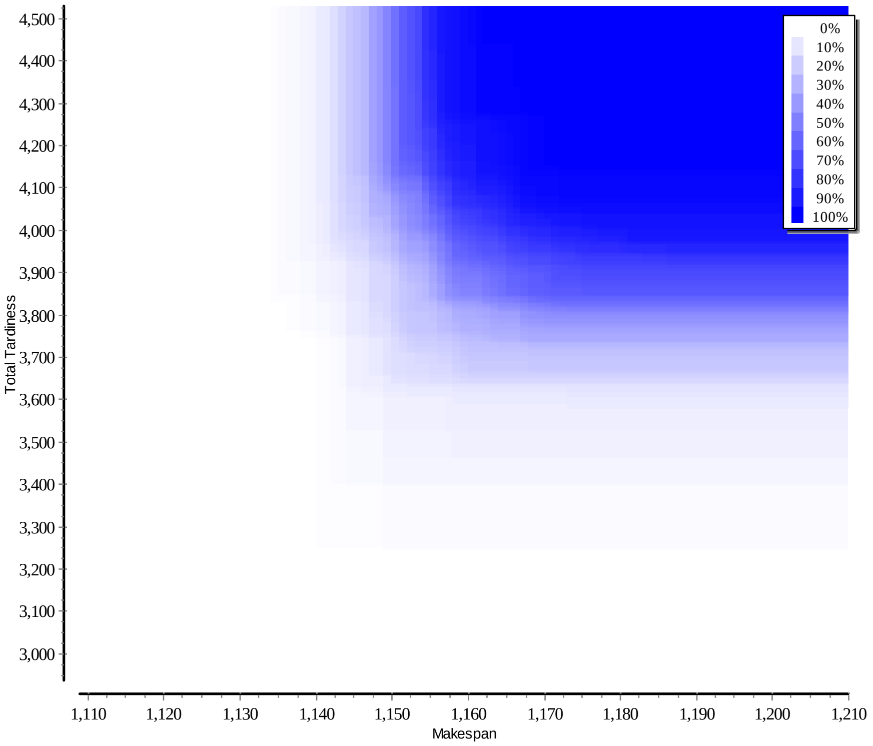


Figure 8.18: Plot of EAF for RIPG. Example instance 5 with 50 jobs, 4 stages and 2 machines per stage.

In order to facilitate the comparison between the empirical attainment functions of two algorithms, the difference between the EAFs can be summarised in one graph. This graph is referred to as the differential empirical attainment function. In Figure 8.19, such a graph is shown for example instance 5. On the bottom and on the left, the difference is zero since neither of the two algorithms covers this part in any run. In the right upper corner, the difference is zero since both algorithms always cover this point. In between, the grade of red indicates in which extend NSGA-II outperforms RIPG.

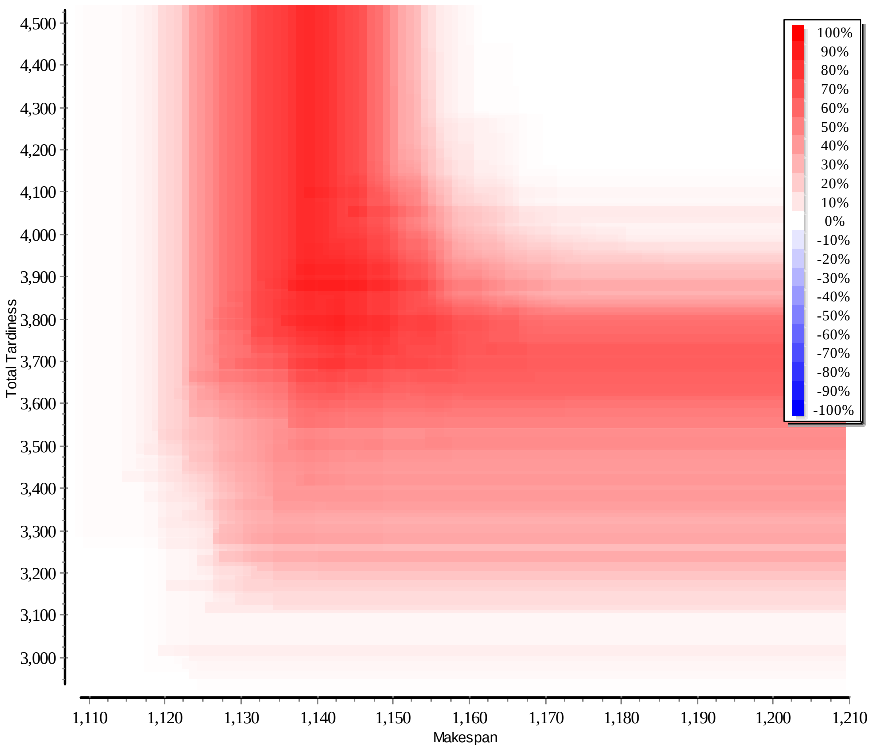


Figure 8.19: Plot of Diff-EAF for NSGA-II and RIPG. In red the area where NSGA-II outperforms RIPG. Example instance 5 with 50 jobs, 4 stages and 2 machines per stage.

NSGA-II is not better than RIPG for all large instances. In Figure 8.20 an example is given where RIPG outperforms NSGA-II. The differential empirical attainment function is shown for example instance 6; an instance with 100 jobs, 4 stages and 4 parallel machines per stage. The blue colour indicates where RIPG covers the objective space more often than NSGA-II does. It is curious to see the full blue area on the bottom of the graph. This means that in practically all runs, RIPG finds solutions with a good makespan and zero tardiness, whereas NSGA-II almost never finds solutions with zero tardiness.

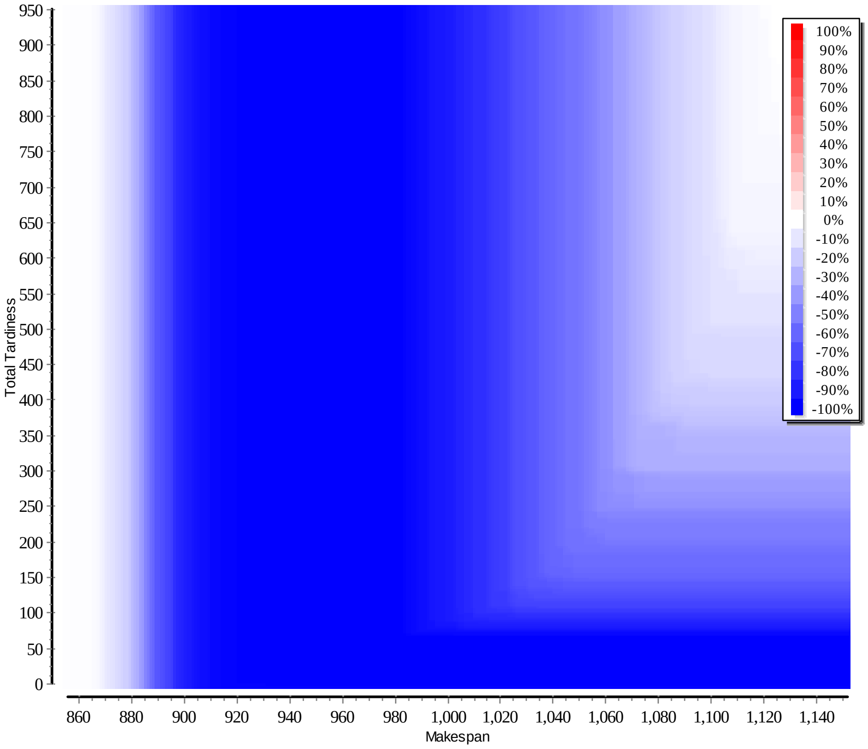


Figure 8.20: Plot of Diff-EAF for NSGA-II and RIPG. In blue the area where RIPG outperforms NSGA-II. Example instance 6 with 100 jobs, 4 stages and 4 machines per stage.

Another interesting instance, is example instance 7 of exactly the same size as example instance 6. In Figure 8.21, if one looks well, one can see that there is a light red area close to the axes and a bit stronger blue area above and to the right of this red area. This means that NSGA-II obtains on the one hand very good approximation sets in a slightly higher number of cases than RIPG. On the other hand, NSGA-II also returns bad approximation sets more often than RIPG. We can interpret this as an indication that the best Pareto approximation sets for this instance are obtained by NSGA-II, but that the genetic algorithm is less stable than RIPG.

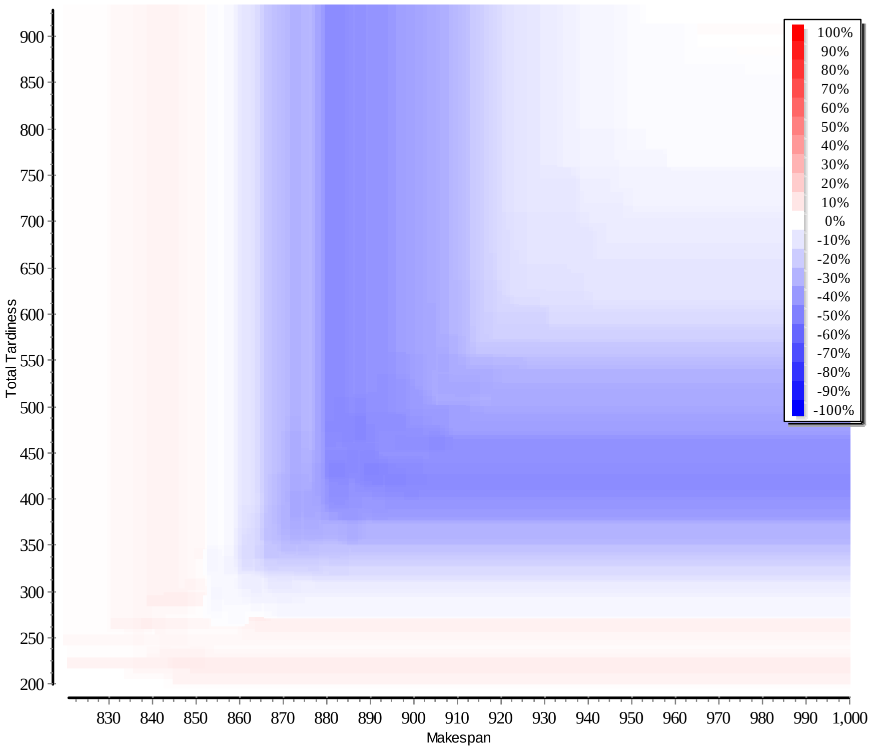


Figure 8.21: Plot of Diff-EAF for NSGA-II and RIPG. In blue the area where RIPG outperforms NSGA-II and in red the area where NSGA-II outperforms RIPG. Example instance 7 with 100 jobs, 4 stages and 4 machines per stage.

8.6. Conclusions

In this chapter, it has been shown that there is practically no multi-objective research done for the hybrid flexible flowline problem. This is confirmed by the hybrid flowshop reviews of Vignier et al. (1999), Linn and Zhang (1999), Quadt and Kuhn (2007), Ruiz and Vázquez Rodríguez (2010) and Ribas et al. (2010). We have presented two algorithms for solving the hybrid flowshop problem with unrelated parallel machines in each of the stages. Pareto optimisation is done for two objectives, namely makespan and total tardiness. Different quality measures are used to analyse the results in detail and to avoid biases caused by the chosen performance indicator. The conclusions that can be drawn from the analysis are given in this section.

More randomness contributes to a better algorithm in the case of large problem instances for this multi-objective scheduling problem. This can be concluded both from the comparison between NSGA-II and RIGP, and from the calibration of NSGA-II. In NSGA-II, as in all genetic algorithms, the solution changes are done mainly in a random way. The changes done in RIPG are more greedy and compare many options before changing a solution. This comparison of options is costly for the problem we treat here and therefore results in an algorithm that is inefficient for large instances.

Moreover, as shown in Chapter 6, accelerations are an important ingredient for success when local search is applied. However, the accelerations implemented in that chapter are not suitable for the hybrid flowshop problem treated here. Because of the precedence constraints in the HFFL problem considered in the previous chapters, none of the tasks of a job can be planned before all tasks of its predecessor are fixed in the schedule. This obliges the algorithms to generate the schedule on job-by-job basis. Since the job-permutation solution representation is coherent with the scheduling order, the proposed accelerations are possible in that case. In the HFS problem in this chapter, however, no precedence constraints are considered. This allows the algorithms to assign the jobs to machines on a stage-by-stage basis, establishing the job permutation at each stage as the order in which they are finished in the previous stage. This results in better solutions, but contrasts with the solution representation. The earlier used accelerations are therefore not possible for the methods proposed in this chapter. This clearly affects the RIPG results, especially for large instances.

We are currently working on a journal publication, based on the research presented in this chapter. Some initial results have been accepted for presentation in the Twelfth International Workshop on Project Management and Scheduling and publication in the proceedings.

CHAPTER 9

CONCLUSIONS AND FUTURE RESEARCH

Since the first publications in the field of scheduling, more than half a century ago, there has existed a gap between the literature in this field on the one hand and the necessities of production schedulers in practice on the other hand. This can be learned from the literature review in this Ph.D. thesis. Scientific research has been concentrated in most of the cases on problems that represent a simplification of reality, which somehow restricts the practical interest. Usually a number of assumptions is made in order to obtain a favourable mathematical formulation or facilitate the implementation of efficient and effective algorithms. In other cases, practical applications are studied. This kind of research is obviously connected to the real world in a close way. However, the applications are usually too specific to create the possibility of large-scale implementation. What has been shown in several research papers over the last decades, is that the industry could take profit of an easy and flexible tool that generates good solutions exactly for their problems without assumptions that may cause the solution of the algorithm to be infeasible in practice. These solutions, moreover, need to be available in a short timespan, so that different scenarios can easily be compared and changes in the production plan can almost immediately be adopted in the schedule. This requirement excludes the possibility of lengthy overnight calculations.

In this Ph.D. thesis, the central objective has been to contribute on closing this gap between the scheduling literature and what real-life production planning asks for, as described in Chapter 1. Whereas most papers in the line of realistic scheduling consider only one or two realistic restrictions, we therefore treat a hybrid flexible flow line together with various constraints that occur in many production environments. Among the modelled real-life problem characteristics we can find stage skipping, machine eligibility, precedence constraints, time lags, machine release dates and sequence dependent setup times, either anticipatory or not. From the literature review in Chapter 2, it can be concluded that the addressed problem is more complex than the problems usually treated in literature and allows for direct implementation in real-world situations.

For this problem, we have shown a complete formulation, as well as a mixed integer linear programming model in Chapter 3. We introduced a benchmark of problem instances, taking into account the numerous problem restrictions. A commercial solver (CPLEX) was used to obtain the optimal solutions for a number of instances, and analysis of the results gave information about the addition in complexity for each of the problem restrictions.

Since the mathematical model is only suitable for problem instances of small size, some heuristic methods were adapted for this problem in Chapter 4. Among those heuristics we find six dispatching rules and the well-known NEH heuristic. All had to be modified in order to be able to generate solutions for the hybrid flexible flow line problem. Usually the heuristics are applied to regular flowshop problems.

Furthermore, in Chapter 5 we have implemented and compared genetic algorithms with distinct solution representations. The solution representations range from simple job sequences with a single machine assignment rule (BGA and SGA), job sequences with per-job machine assignment rules (SGAR), complete machine assignments (SGAM), to the exact representation of the solution with per-machine job sequences (EGA).

Neither crossover nor mutation operators were defined in the literature for this last representation. We have introduced two new crossover operators and two new mutation operators are introduced for EGA. For the other algorithms, several new machine assignment rules are proposed.

For the evaluation and comparison of the different algorithms, a subset of the earlier created benchmark of problem instances is used. All five genetic algorithms are subject to a parameter calibration, using ANOVA statistical techniques. The algorithms prove to be robust with respect to the allowed running time and to the characteristics of the instances.
Once calibrated, the genetic algorithms are compared to some other existing methods. For the small instances the solution values of a MIP model with 15 minutes time limit are available. For both the small and the large instances, five dispatching rules and a NEH adaptation, all using the machine assignment rules, are used for comparison. All genetic algorithms outperform the MIP model and all heuristics. A random solution generator (RS) with the same time limit as for the GAs is used for comparison. For small instances RS is comparable to EGA and better than SGAM; for large instances all proposed GAs show a better performance.
The algorithms with less direct solution representations (BGA, SGA, SGAR) already show good results for small allowed running time. More running time causes an insignificantly small improvement in the solution value. The algorithms with more verbose solution representation (SGAM and especially EGA) profit more from the extra time, but still do not reach the solution quality of the earlier mentioned algorithms.

Local search is a very powerful technique that is profusely used in the literature. However, when applied to practical and realistic problems, the CPU time requirements are extremely high. The complexity of the problem dealt with in this Ph.D. thesis, is a clear disadvantage for local search techniques. Contrary to simple scheduling problems, where straightforward local search techniques are frequently applied, it seems that such techniques are neither easy nor apparently as profitable when applied to much more complex environments.

In Chapter 6, we have advanced towards a better understanding of local search based algorithms for complex scheduling problems. We have evaluated various local search neighbourhood implementations for the hybrid flexible flow line problem. We have shown the limited possibilities of applying regular accelerations. The consequences of each neighbourhood search have been shown.
A new compound way of applying local search within a genetic algorithm has been presented. The resulting memetic algorithm scans the local search neighbourhood less often than usually, as the problem does not allow for all common accelerations. In addition to that, adaptations were given of algorithms that were proposed for simpler scheduling problems or other fields of research, namely Iterated Local Search and Iterated Greedy. The results indicated that curtailed and carefully crafted local search procedures are able to find good solutions for this HFFL and start to show their promise. However, we have found that the algorithmic issues that arise for these realistic problems are different from those on the very simple problems traditionally studied in scheduling theory. This was most noticeable at the local search component. While local search still plays a role even for these complex problems, its impact appears to be less dominant than for the simpler ones. More complex search strategies need to be developed.

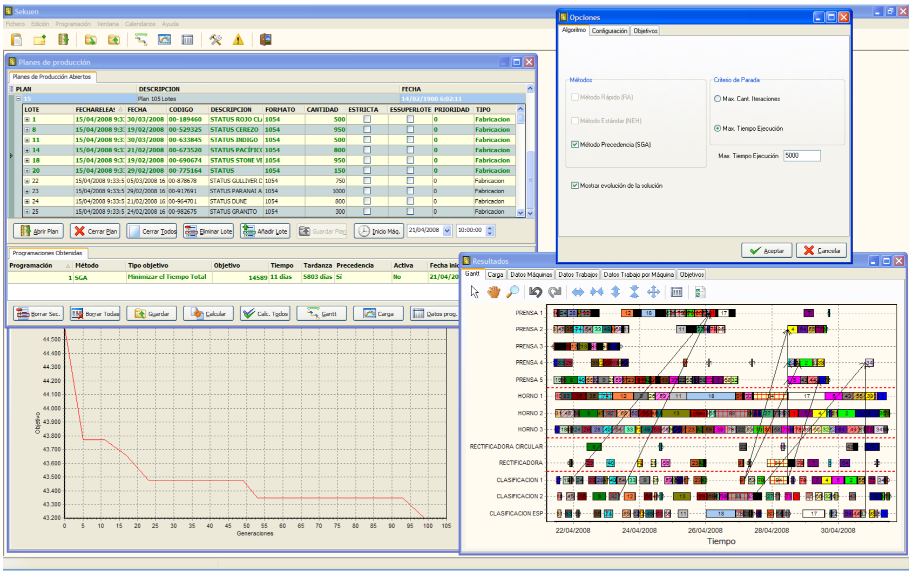
In Chapter 7, a shift of representation in the search has been shown as a novel idea to improve the solution quality. In the case of a Mixed Genetic Algorithm, the second phase lacks to have a significant impact. A new local search based algorithm called Shifting Representation Search, is also proposed. The first phase, consisting in a iterated greedy search on a job permutation, assures that a good solution can be found in little time. When it gets hard for the iterated greedy to improve the solution due to the limitations of the compact solution representation, a shift in the solution representation is done. Starting from then, an iterated local search continues with the best found solution, searching the full solution space. In order to improve the efficiency, only the tasks on the critical path are considered in the local search. This novel algorithm has been proven to outperform all earlier presented methods, both for the large

and for the small instances. Note that, although the local search implementation is quite case-specific and based on the problem characteristics, the main idea of shifting the solution representation is generally applicable.
The hybrid flexible flowline problem is a composite problem in the sense that it is composed of different subproblems. In fact, this composition is the case for many real-life problems. Vehicle routing problems, for instance, are composed of a partitioning and a routing problem. Another example is the very large scale integration (VLSI) design problem, that is composed of two subproblems: the choice which components to place and the choice where to place the chosen components. Although we have no data on the application on those problems, the idea of focussing on a subproblem in a first phase and considering all problem aspects in a later phase is likely to yield good solutions in those cases as well.

In Chapter 8, a multi-objective hybrid flowshop problem has been tackled. Just as in the more restricted HFFL problem, the parallel machines in each stage have been assumed to be unrelated, which is the most general case. Both makespan as a measure for efficiency and total tardiness as a indicator for client satisfaction were optimised at the same time. Pareto domination techniques were applied to define a set of non-dominated solutions, among which none is better nor worse than one of the others on both objective values.
Two algorithms have been developed for this problem. A genetic algorithm candidate was found in the form of an adaptation for the hybrid flowshop problem of the known NSGA-II. In this thesis, a new local search algorithm called RIPG has been presented, especially designed for multi-objective scheduling problems. The computational results showed that RIPG outperforms NSGA-II for small problem instances, but that NSGA-II is more effective for larger instances.

9.1. Scheduling software

We claim that the scheduling problem we consider is very realistic. The best way to prove this is by an implementation in a real life environment. The presented SGA and the NEH adaptation are implemented in SeKuen, a finite capacity production scheduling software, currently being deployed at two of the biggest partners in the Spanish ceramic tile sector, namely Porcelanosa S.A. and TAULELL S.A. Other important ceramic tile manufacturing companies using SeKuen are Halcón Cerámicas S.A. and ColorKer Cerámicas S.A. SeKuen is used in other sectors as well, like in Frost-Trol S.A., a producer of refrigerators for supermarkets. Framinan and Ruiz (2010) describe in a detailed way what the necessities and the difficulties are, when implementing a scheduling system in an industrial environment. In Figure 9.1 an implementation of the SGA algorithm in a real life application is shown. The graph shows the evolution of the makespan value over time. The diagonal lines in the Gantt chart represent the precedence relationships.



PLAN	DESCRIPCION	FECHA
15	Plan 105 Lotes	14/02/1900 6:02:11

LOTE	FECHARELEA!	FECHA	CODIGO	DESCRIPCION	FORMATO	CANTIDAD	ESTRICTA	ESSUPERLOTE	PRIORIDAD	TIPO
1	15/04/2008 9:3:	30/03/2008	00-189460	STATUS ROJO CL/	1054	500	□	□	0	Fabricacion
8	15/04/2008 9:3:	19/02/2008	00-529325	STATUS CEREZO	1054	950	□	□	0	Fabricacion
11	15/04/2008 9:3:	30/03/2008	00-633845	STATUS INDIGO	1054	500	□	□	0	Fabricacion
14	15/04/2008 9:3:	21/02/2008	00-673520	STATUS PACÍFIC(	1054	800	□	□	0	Fabricacion
18	15/04/2008 9:3:	19/02/2008	00-690674	STATUS STONE VE	1054	950	□	□	0	Fabricacion
20	15/04/2008 9:3:	29/02/2008	00-775164	STATUS	1054	150	□	□	0	Fabricacion
22	15/04/2008 9:33:5	05/03/2008 16	00-878678	STATUS GULLIVER D	1054	750	□	□	0	Fabricacion
23	15/04/2008 9:33:5	29/02/2008 16	00-917691	STATUS PARANAI A	1054	1000	□	□	0	Fabricacion
24	15/04/2008 9:33:5	21/02/2008 16	00-964701	STATUS DUNE	1054	800	□	□	0	Fabricacion
25	15/04/2008 9:33:5	24/02/2008 16	00-982675	STATUS GRANITO	1054	300	□	□	0	Fabricacion

Inicio Máq. 21/04/2008 10:00:00

Programación	Método	Tipo objetivo	Objetivo	Tiempo	Tardanza	Precedencia	Activa	Fecha inic
1	SGA	Minimizar el Tiempo Total	14589	11 días	5803 días	Sí	No	21/04/20

Métodos
□ Método Rápido (RA)
□ Método Estándar (NEH)
☑ Método Precedencia (SGA)

Criterio de Parada
○ Max. Cant. Iteraciones
● Max. Tiempo Ejecución

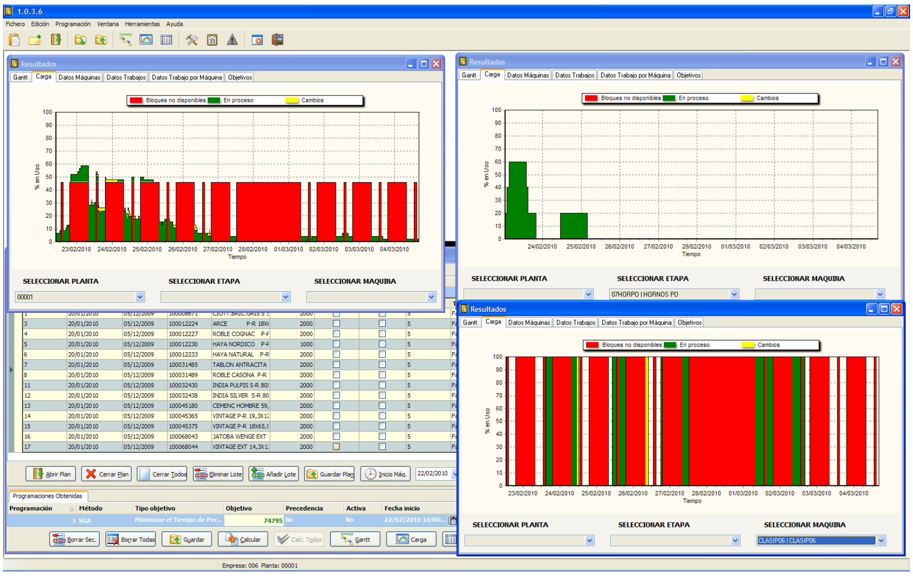
Max. Tiempo Ejecución 5000

☑ Mostrar evolución de la solución

Figure 9.1: Application of SGA for HFFL problems as treated in this Ph.D. thesis.

Figure 9.2 shows three different machine use diagrams for three different schedules. The schedules are generated by the rapid access heuristic of Dannenbring (1977), by the NEH heuristic presented in Section 4.5 and by the

SGA introduced in Section 5.2. The diagrams show the percentage of used machines on the vertical axis, and the time on the horizontal axis.



SELECCIONAR PLANTA: 00001
SELECCIONAR ETAPA:
SELECCIONAR MAQUINA:

SELECCIONAR PLANTA:
SELECCIONAR ETAPA: 07HORPO | HORNOS PO
SELECCIONAR MAQUINA:

SELECCIONAR PLANTA:
SELECCIONAR ETAPA:
SELECCIONAR MAQUINA: CLASIP06 | CLASIP06

1	20/01/2010	05/12/2009	100008671	CIOTT.BRIC.GRIS S	2000	□	□	5	Fa
3	20/01/2010	05/12/2009	100012224	ARCE P-R 18X	2000	□	□	5	Fa
4	20/01/2010	05/12/2009	100012227	ROBLE COGNAC P-F	2000	□	□	5	Fa
5	20/01/2010	05/12/2009	100012230	HAYA NORDICO P-F	1000	□	□	5	Fa
6	20/01/2010	05/12/2009	100012233	HAYA NATURAL P-R	2000	□	□	5	Fa
7	20/01/2010	05/12/2009	100031485	TABLON ANTRACITA	2000	□	□	5	Fa
8	20/01/2010	05/12/2009	100031489	ROBLE CASONA P-R	2000	□	□	5	Fa
11	20/01/2010	05/12/2009	100032430	INDIA PULPIS S-R 80	2000	□	□	5	Fa
12	20/01/2010	05/12/2009	100032438	INDIA SILVER S-R 80	2000	□	□	5	Fa
13	20/01/2010	05/12/2009	100045180	CEMENC HOMBRE 59,	2000	□	□	5	Fa
14	20/01/2010	05/12/2009	100045365	VINTAGE P-R 19,3X1	2000	□	□	5	Fa
15	20/01/2010	05/12/2009	100045375	VINTAGE P-R 18X65,	2000	□	□	5	Fa
16	20/01/2010	05/12/2009	100068043	JATOBA WENGE EXT	2000	□	□	5	Fa
17	20/01/2010	05/12/2009	100068044	VINTAGE EXT 14,3X1	2000	□	□	5	Fa

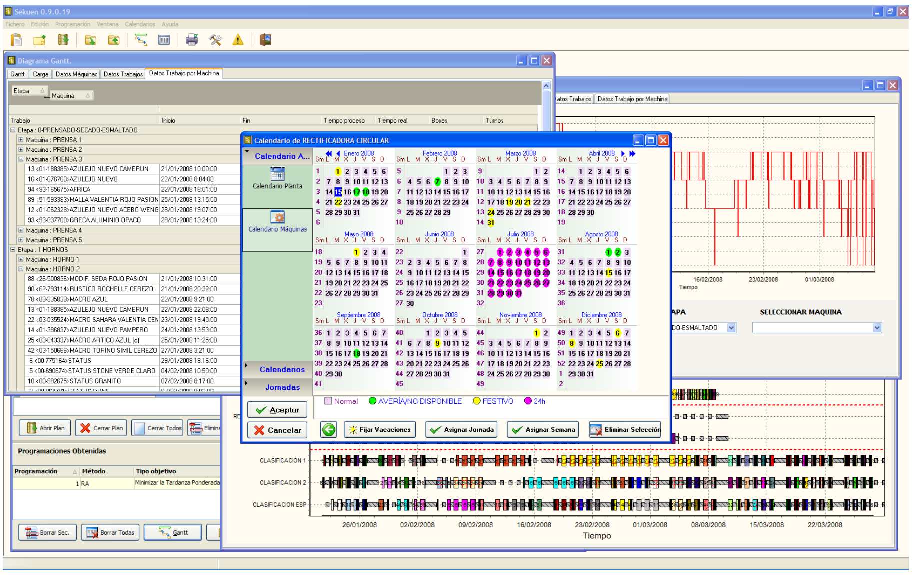
Inicio Máq.: 22/02/2010

Programación	Método	Tipo objetivo	Objetivo	Precedencia	Activa	Fecha inicio
1	SGA	Minimizar el Tiempo de Per...	74795	No	No	22/02/2010 10:00:...

Empresa: 006 Planta: 00001

Figure 9.2: Machine use over time for three different schedules.

In addition to the problem characteristics treated in this thesis, SeKuen offers the possibility to enter machine downtimes, as shown in Figure 9.3. Availability of machines can be defined for the entire plant, per stage, and/or per machine. Machines might not be available because of holidays, breakdowns, weekends, maintenance, etcetera. Unavailability of machines is indicated in the Gantt diagrams with a narrow grey bar, as can be seen in the Gantt below the calendar.



Etapa / Maquina

Trabajo	Inicio
Etapa : 0-PRENSADO-SECADO-ESMALTADO	
Maquina : PRENSA 1	
Maquina : PRENSA 2	
Maquina : PRENSA 3	
13 <01-188385>AZULEJO NUEVO CAMERUN	21/01/2008 10:00:00
16 <01-676760>AZULEJO NUEVO	22/01/2008 8:04:00
94 <93-165675>AFRICA	22/01/2008 18:01:00
89 <51-593383>MALLA VALENTIA ROJO PASION	25/01/2008 13:15:00
12 <01-062328>AZULEJO NUEVO ACEBO WENG	28/01/2008 19:07:00
93 <93-037700>GRECA ALUMINIO OPACO	29/01/2008 13:24:00
Maquina : PRENSA 4	
Maquina : PRENSA 5	
Etapa : 1-HORNOS	
Maquina : HORNO 1	
Maquina : HORNO 2	
88 <26-500836>MODIF. SEDA ROJO PASION	21/01/2008 10:31:00
90 <62-793114>RUSTICO ROCHELLE CEREZO	21/01/2008 20:32:00
78 <03-335839>MACRO AZUL	22/01/2008 9:21:00
13 <01-188385>AZULEJO NUEVO CAMERUN	22/01/2008 22:08:00
22 <03-035524>MACRO SAHARA VALENTIA CEM	23/01/2008 19:40:00
14 <01-386837>AZULEJO NUEVO PAMPERO	24/01/2008 13:53:00
25 <03-043337>MACRO ARTICO AZUL (c)	25/01/2008 11:25:00
42 <03-150666>MACRO TORINO SIMIL CEREZO	27/01/2008 3:21:00
6 <00-775164>STATUS	29/01/2008 18:16:00
5 <00-690674>STATUS STONE VERDE CLARO	04/02/2008 10:50:00
10 <00-982675>STATUS GRANITO	07/02/2008 8:17:00

Programaciones Obtenidas

Programación	Método	Tipo objetivo
1	RA	Minimizar la Tardanza Ponderada

Figure 9.3: Introduction of timetables for machine breakdowns.

Stage revisiting is also allowed and the order of the stages is not fixed in SeKuen. In Figure 9.4, for example, job 11 first visits stage 1 (where it is assigned to PRENSA2), then it is processed at stage 2 (HORNO1) and after that returns to stage 1 (PRENSA2). This actually goes beyond the complexity of a flow line problem. Furthermore, limited buffers between stages are considered. The processing times at stage 1 are considerably shorter than the ones at stage 2. So when job 11 visits stage 1 for the second time, PRENSA2 has to wait after processing a part of the job, until HORNO1 delivers the rest of the job at stage 1. This waiting time is indicated with diagonal black lines.

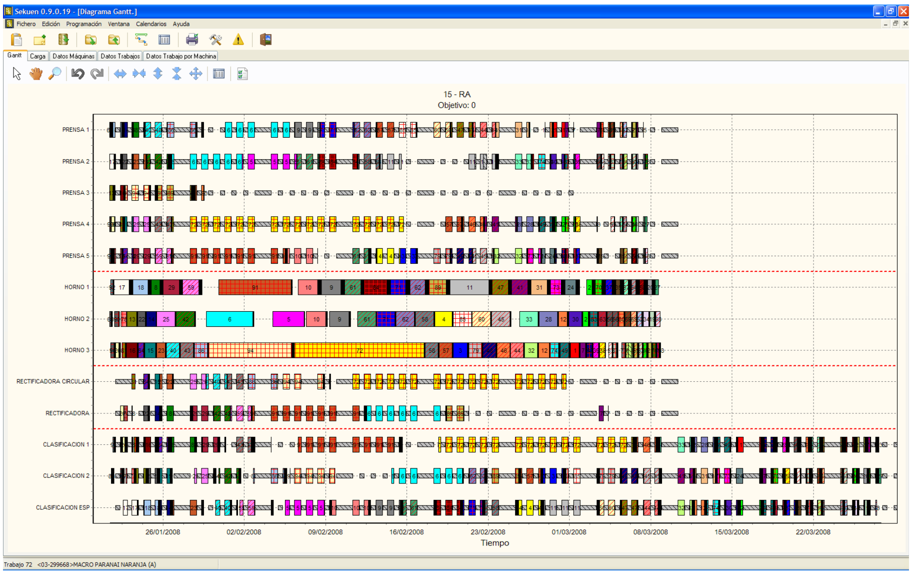
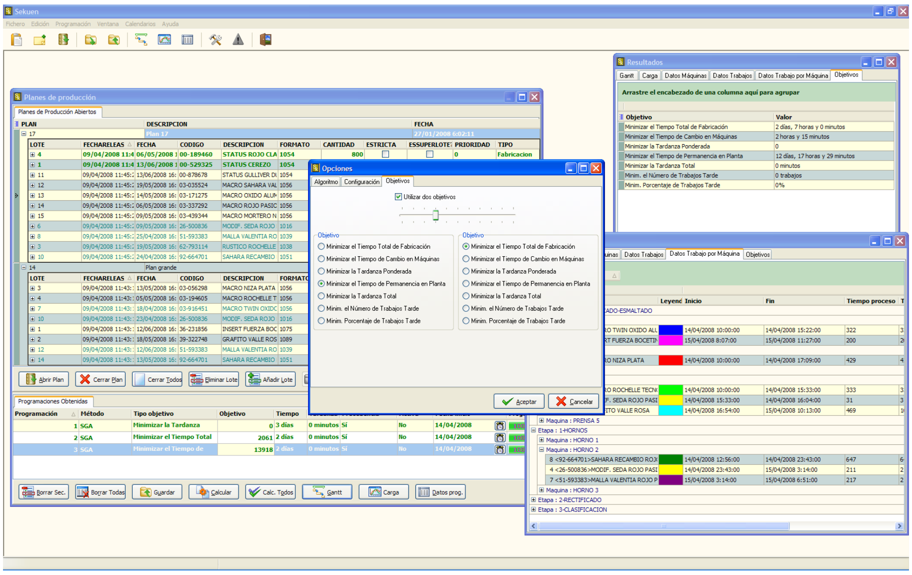


Figure 9.4: Gantt chart with revisited stages and limited buffers between stages.

We have done an effort as well to work with more than one objective. Since Pareto optimisation is very time consuming and the output of a frontier of solutions is not very practical, we have chosen for linear combinations of objectives. In the menu in Figure 9.5, the two objectives and the weight of each objective can be chosen. In the upper right corner, the values for all implemented objectives are given for a certain solution.



PLAN	DESCRIPCION	FECHA
17	Plan 17	27/01/2008 6:02:11

LOTE	FECHARELEAS	FECHA	CODIGO	DESCRIPCION	FORMATO	CANTIDAD	ESTRICTA	ESSUPERLOTE:	PRIORIDAD	TIPO
4	09/04/2008 11:4	06/05/2008 1	00-189460	STATUS ROJO CLA	1054	800	□	□	0	Fabricacion
1	09/04/2008 11:4	13/06/2008 1	00-529325	STATUS CEREZO	1054					
11	09/04/2008 11:45::	13/06/2008 16:	00-878678	STATUS GULLIVER DI	1054					
12	09/04/2008 11:45::	19/05/2008 16:	03-035524	MACRO SAHARA VAL	1056					
13	09/04/2008 11:45::	14/05/2008 16:	03-171275	MACRO OXIDO ALUN	1056					
14	09/04/2008 11:45::	06/05/2008 16:	03-337292	MACRO ROJO PASIC	1056					
15	09/04/2008 11:45::	09/05/2008 16:	03-439344	MACRO MORTERO N	1056					
6	09/04/2008 11:45::	09/05/2008 16:	26-500836	MODIF. SEDA ROJO	1016					
8	09/04/2008 11:45::	25/04/2008 16:	51-593383	MALLA VALENTIA RO	1039					
3	09/04/2008 11:45::	19/05/2008 16:	62-793114	RUSTICO ROCHELLE	1038					
10	09/04/2008 11:45::	24/04/2008 16:	92-664701	SAHARA RECAMBIO	1051					

PLAN	DESCRIPCION
14	Plan grande

LOTE	FECHARELEAS	FECHA	CODIGO	DESCRIPCION	FORMATO
3	09/04/2008 11:43::	13/05/2008 16:	03-056298	MACRO NIZA PLATA	1056
4	09/04/2008 11:43::	05/05/2008 16:	03-194605	MACRO ROCHELLE T	1056
7	09/04/2008 11:43::	18/04/2008 16:	03-916451	MACRO TWIN OXIDC	1056
10	09/04/2008 11:43::	23/04/2008 16:	26-500836	MODIF. SEDA ROJO	1016
1	09/04/2008 11:43::	12/06/2008 16:	36-231856	INSERT FUERZA BOC	1075
2	09/04/2008 11:43::	18/05/2008 16:	39-322748	GRAFITO VALLE ROS	1089
12	09/04/2008 11:43::	12/06/2008 16:	51-593383	MALLA VALENTIA RO	1039
14	09/04/2008 11:43::	13/05/2008 16:	92-664701	SAHARA RECAMBIO	1051

Programación	Método	Tipo objetivo	Objetivo	Tiempo						
1	SGA	Minimizar la Tardanza	0	3 días	0 minutos	Sí	No	14/04/2008		100
2	SGA	Minimizar el Tiempo Total	2061	2 días	0 minutos	Sí	No	14/04/2008		100
3	SGA	Minimizar el Tiempo de	13918	2 días	0 minutos	Sí	No	14/04/2008		100

☑ Utilizar dos objetivos

Objetivo
○ Minimizar el Tiempo Total de Fabricación
○ Minimizar el Tiempo de Cambio en Máquinas
○ Minimizar la Tardanza Ponderada
● Minimizar el Tiempo de Permanencia en Planta
○ Minimizar la Tardanza Total
○ Minim. el Número de Trabajos Tarde
○ Minim. Porcentaje de Trabajos Tarde

Objetivo
● Minimizar el Tiempo Total de Fabricación
○ Minimizar el Tiempo de Cambio en Máquinas
○ Minimizar la Tardanza Ponderada
○ Minimizar el Tiempo de Permanencia en Planta
○ Minimizar la Tardanza Total
○ Minim. el Número de Trabajos Tarde
○ Minim. Porcentaje de Trabajos Tarde

Objetivo	Valor
Minimizar el Tiempo Total de Fabricación	2 días, 7 horas y 0 minutos
Minimizar el Tiempo de Cambio en Máquinas	2 horas y 15 minutos
Minimizar la Tardanza Ponderada	0
Minimizar el Tiempo de Permanencia en Planta	12 días, 17 horas y 29 minutos
Minimizar la Tardanza Total	0 minutos
Minim. el Número de Trabajos Tarde	0 trabajos
Minim. Porcentaje de Trabajos Tarde	0%

	Leyend	Inicio	Fin	Tiempo proceso
ADO-ESMALTADO				
O TWIN OXIDO ALL	🟦	14/04/2008 10:00:00	14/04/2008 15:22:00	322
T FUERZA BOCETIN	🟪	15/04/2008 8:07:00	15/04/2008 11:27:00	200
O NIZA PLATA	🟥	14/04/2008 10:00:00	14/04/2008 17:09:00	429
O ROCHELLE TECN	🟩	14/04/2008 10:00:00	14/04/2008 15:33:00	333
F. SEDA ROJO PASI	🟨	14/04/2008 15:33:00	14/04/2008 16:04:00	31
ITO VALLE ROSA	🟦	14/04/2008 16:54:00	15/04/2008 10:13:00	469
Maquina : PRENSA 5				
Etapa : 1-HORNOS				
Maquina : HORNO 1				
Maquina : HORNO 2				
8 <92-664701>SAHARA RECAMBIO ROJ	🟩	14/04/2008 12:56:00	14/04/2008 23:43:00	647
4 <26-500836>MODIF. SEDA ROJO PASI	🟨	14/04/2008 23:43:00	15/04/2008 3:14:00	211
7 <51-593383>MALLA VALENTIA ROJO P	🟪	15/04/2008 3:14:00	15/04/2008 6:51:00	217
Maquina : HORNO 3				
Etapa : 2-RECTIFICADO				
Etapa : 3-CLASIFICACION				

Figure 9.5: Linear combination of two optimisation criterea.

The creation, maintenance and implementation of SeKuen is a joint effort of the research group Sistemas de Optimización Aplicada of the Instituto Tecnológico de Informática. Note that SeKuen is in continuous development and that more current information about the research group and about SeKuen is available at `http://www.soa.iti.es` and `http://www.sekuen.com`, respectively.

9.2. Future research

First of all, our intention is to continue in the line of multi-objective optimisation for the hybrid flowshop problem. Advanced techniques such as local search acceleration or shifting representation can lead to a new algorithm that outperforms the existing ones. Apart from developing new solution techniques, more work is needed in order to further close the gap between the theory and practice of scheduling. In order to do so, the more restricted

hybrid flexible flow line can be considered in a multi-objective environment. Obviously, this will make the task for the optimisation methods even harder, situation which causes new design issues.
Although we are considering highly realistic and complex hybrid flexible flowshop problems, there is still a number of restrictions that are currently not addressed in our research. The scheduling software SeKuen includes some of these issues. We refer to the possibility to take machine downtimes into account from a scientific point of view, even as limited buffer capacity between machines, or recirculation. It would be very interesting to model these issues and investigate in which extend the problem increases in complexity by adding those constraints.
Adopting other real-world situations such as the possibility of preemption or the need for resources other than machines make the solutions offered even more generally applicable. Errors when processing the jobs on machines and breakdowns or wearing on the machines might ask for adaptation of the existing schedule. This is intrinsically a multi-objective scheduling problem, since both the efficiency of the new schedule and the amount of changes between the old and the new schedule should be taken into account. Furthermore, we should take into account that all data used in the considered HFFL is deterministic and known in advance. In reality, input data like processing and setup times are usually stochastic, which is something that could be taken into account as well when modelling the problem.

Apart from all open research issues, we see as an important part of our contribution the conversion of research results to industrial results. The outcomes of this Ph.D. thesis will therefore be translated into extensions and advances of SeKuen, so that our progress in this field truly helps the industry in improving their production scheduling process.

9.3. Publications

Our research has already lead to one publication in Computers and Operations Research; an international journal with an impact factor of 1.366 in 2008.

In the paper, a mixed linear integer programming model is given and CPLEX is used to obtain solutions a bench of small problem instances. The results are used to analyse the complexity of each of the problem restrictions. Furthermore, several heuristics are presented, in order to obtain solutions for larger problem instances.

Ruiz, R., Sivrikaya Şerifoğlu, F., and Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35(4):1151-1175.

Another paper is published in the opening issue of International Journal of Metaheuristics; a promising new journal, given the good reputation of the board. The paper contains the genetic algorithms presented in this Ph.D. thesis, with the machine assignment rules and the various solution representations. Comparison is made with improved versions of the heuristics that were presented in the above mentioned paper.

Urlings, T., Ruiz, R., and Sivrikaya Şerifoğlu, F. (2010). Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems. *International Journal of Metaheuristics*, 1(1):30-54.

A third article is accepted for publication in the European Journal of Operational Research; an leading international journal in the field, with an impact factor of 1.627 in 2008. This paper is the fruit of a visit of three months in IRIDIA, a research institute in the Université Libre de Bruxelles. There, analysis of and important improvements in the local search algorithms were made. Apart from the adapted ILS and IG, a completely new algorithm is proposed in the paper, using a shift in the solution presentation. This representation shift allows the algorithm to quickly achieve a strong local optimum and then continue with a more sophisticated search in order to improve on this solution.

Urlings, T., Ruiz, R., and Stützle, T. (2010). Shifting representation search for hybrid flexible flowline problems. *European Journal of Operational*

Research, (In Press).

For further transference of the results of our research, we presented our work in numerous national and international conferences. One conference lead to publication in Lecture Notes in Computer Science:

Urlings, T., and Ruiz, R. (2007). Local Search in Complex Scheduling Problems. *Engineering Stochastic Local Search Algorithms*, 4638:202-206. SLS Workshop, Brussels, Belgium; September 6-8, 2007.

The remaining conference contributions have been published in the respective proceedings:

Ruiz, R., Sivrikaya Şerifoğlu, F., and Urlings, T. (2006). An evolutionary approach to realistic hybrid flexible flowshop scheduling problems. *Tenth International Workshop on Project Management and Scheduling*, Poznań, Poland; April 26-28, 2006.

Ruiz, R., Sivrikaya Şerifoğlu, F., and Urlings, T. (2006). Secuenciación mediante algoritmos evolutivos en complejos talleres flexibles. *XXIX Congreso Nacional de Estadística e Investigación Operativa*, Tenerife, Spain; May 15-19, 2006.

Urlings, T., Ruiz, R., and Sivrikaya Şerifoğlu, F. (2006). Heuristics for Highly Constrained Hybrid Flexible Flowshop Scheduling Problems. *21st European Conference on Operational Research*, Reykjavik, Iceland; July 2-5, 2006.

Urlings, T., Ruiz, R., and Sivrikaya Şerifoğlu, F. (2007). Genetic algorithms for complex hybrid flexible flow line problems. *Eighth Workshop on Models and Algorithms for Planning and Scheduling Problems*, Istanbul, Turkey; July 2-6, 2007.

Urlings, T., Ruiz, R., and Sivrikaya Şerifoğlu, F. (2007). Genetic algorithms for complex hybrid flexible flow line problems. *Fourth OR Peripatetic Post-Graduate Programme*, Guimarães, Portugal; September 12-15, 2007.

Urlings, T., Stützle, T., and Ruiz, R. (2008) Local Search Engineering for highly constrained flow line problems. *Eleventh International Workshop on Project Management and Scheduling*, Istanbul, Turkey; April 28-30, 2008.

Urlings, T., and Ruiz, R. (2009) Búsqueda local avanzada para talleres de flujo híbridos altamente restringidos. *XXXI Congreso Nacional de Estadística e Investigación Operativa* Murcia, Spain; February 10-13, 2009.

Urlings, T., and Ruiz, R. (2009) A new algorithm for multidimensional scheduling problems. *Ninth Workshop on Models and Algorithms for Planning and Scheduling Problems* Abbey Rolduc, The Netherlands; June 29-July 3, 2009.

Urlings, T., and Ruiz, R. (2009) A new algorithm with shifting representation for hybrid flowline problems. *23rd European Conference on Operational research* Bonn, Germany; July 5-8, 2009.

Urlings, T., Minella, G., and Ruiz, R. (2010) Bi-objective Pareto optimization for the hybrid flowshop problem. *Twelfth International Workshop on Project Management and Scheduling*, Tours, France; April 26-28, 2010.

BIBLIOGRAPHY

Adams, J., Balas, E., and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401.

Allahverdi, A., Gupta, J. N. D., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega-International Journal of Management Science*, 27(2):219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032.

Allaoui, H. and Artiba, A. (2004). Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Computers and Industrial Engineering*, 47:431–450.

Almada-Lobo, B. and James, R. J. W. (2010). Neighbourhood search metaheuristics for capacitated lot-sizing with sequence-dependent setups. *International Journal of Production Research*, 48(3):861–878.

Armentano, V. A. and Ronconi, D. P. (1999). Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers and Operations Research*, 26(3):219–235.

Behnamian, J., Fatemi Ghomi, S. M. T., and Zandieh, M. (2009). A multiphase covering pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *Expert Systems with Applications*, 36(8):11057–11069.

Bertel, S. and Billaut, J. C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research*, 159(3):651–662.

Bierwirth, C. (1995). A generalized permutation approach to job-shop scheduling with genetic algorithms. *OR Spektrum*, 17(2-3):87–92.

Biggs, D., De Ville, B., and Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, 18(1):49–62.

Botta-Genoulaz, V. (1997). Considering bills of metrial in hybrid flow shop scheduling problems. In *Proceedings of the 1997 IEEE, International Symposium on Assembly and Task Planning*, pages 194–199, Marina del Rey, CA. IEEE.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*, 64(1-3):101–111.

Boyer, W. F. and Hura, G. S. (2005). Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments. *Journal of Parallel and Distributed Computing*, 65(9):1035–1046.

Brah, S. A. and Hunsucker, J. L. (1991). Branch and bound algorithm for the flow-shop with multiple processors. *European Journal of Operational Research*, 51(1):88–99.

Cavory, G., Dupas, R., and Gonçalves, G. (2004). A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints. *European Journal of Operational Research*, 161(1):73–85.

Chen, Z. L. (2004). Simultaneous job scheduling and resource allocation on parallel machines. *Annals of Operations Research*, 129(1-4):135–153.

Dannenbring, D. G. (1977). Evaluation of flow shop sequencing heuristics. *Management Science*, 23(11):1174–1182.

Davoud Pour, H. and Ashrafi, M. (2009). Solving multi-objective sdst flexible flow shop using grasp algorithm. *International Journal of Advanced Manufacturing Technology*, 44(7-8):737–747.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

den Besten, M., Stützle, T., and Dorigo, M. (2001). Design of iterated local search algorithms - an example application to the single machine total weighted tardiness problem. *Applications of Evolutionary Computing, Proceedings*, 2037:441–451. Lecture Notes in Computer Science.

Dhodhi, M. K., Ahmad, I., Yatama, A., and Ahmad, I. (2002). An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 62(9):1338–1361.

Dorn, J., Girsch, M., Skele, G., and Slany, W. (1996). Comparison of iterative improvement techniques for schedule optimization. *European Journal of Operational Research*, 94(2):349–361.

Dudek, R. A., Panwalkar, S. S., and Smith, M. L. (1992). The lessons of flowshop scheduling research. *Operations Research*, 40(1):7–13.

Dugardin, F., Yalaoui, F., and Amadeo, L. (2010). New multi-objective method to solve reentrant hybrid flow shop scheduling problem. *European Journal of Operational Research*, 203(1):20–31.

Fanjul Peyró, L. and Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*. doi: 10.1016/j.ejor.2010.03.030.

Ford, F. N., Bradbard, D. A., Ledbetter, W. N., and Cox, J. F. (1987). Use of operations research in production management. *Production and Inventory Management*, 28(3):59–62.

Framinan, J. M., Gupta, J. N. D., and Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55(12):1243–1255.

Framinan, J. M., Leisten, R., and Rajendran, C. (2003). Different initial sequences for the heuristic of nawaz, enscore and ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. *International Journal of Production Research*, 41(1):121–148.

Framinan, J. M. and Ruiz, R. (2010). Architecture of manufacturing scheduling systems: Literature review and an integrated proposal. *European Journal of Operational Research*, 205(1):237–246.

França, P. M., Gupta, J. N. D., Mendes, A. S., Moscato, P., and Veltink, K. J. (2005). Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers and Industrial Engineering*, 48(3):491–506.

Ge, Q. W. (1999). Paradeg-processor scheduling for acyclic switch-less program nets. *Journal of the Franklin Institute-Engineering and Applied Mathematics*, 336(7):1135–1153.

Geiger, M. J. (2007). On operators and search space topology in multi-objective flow shop scheduling. *European Journal of Operational Research*, 181(1):195–206.

Ghedjati, F. (1999). Genetic algorithms for the job-shop scheduling problem with unrelated parallel constraints: heuristic mixing method machines and precedence. *Computers and Industrial Engineering*, 37(1-2):39–42.

Gilkinson, J. C., Rabelo, L. C., and Bush, B. O. (1995). A real-world scheduling problem using genetic algorithm. *Computers and Industrial Engineering*, 29:177–181.

Gladky, A. A., Shafransky, Y. M., and Strusevich, V. A. (2004). Flow shop scheduling problems under machine-dependent precedence constraints. *Journal of Combinatorial Optimization*, 8(1):13–28.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Gonçalves, J. F., Mendes, J. J. D. M., and Resende, M. G. C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77–95.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Operations Research*, 5:287–326.

Graves, S. C. (1981). A review of production scheduling. *Operations Research*, 29(4):646–675.

Grunert da Fonseca, V., Fonseca, C. M., and Hall, A. O. (2001). Inferential performance assessment of stochastic optimisers and the attainment function. *In Evolutionary Multi-Criterion Optimization, First International Conference, vol. 1993 of Lecture Notes in Computer Science. Spring-Verlang*, 1993:213–225.

Gupta, J. N. D. (1986). Flowshop schedules with sequence dependent setup times. *Journal of the Operations Research Society of Japan*, 29(3):206–219.

Gupta, J. N. D. and Stafford, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711.

Haouari, M. and Hidri, L. (2008). On the hybrid flowshop scheduling problem. *International Journal of Production Economics*, 113(1):495–497.

Haq, A. N., Ravindran, D., Aruna, V., and Nithiya, S. (2004). A hybridisation of metaheuristics for flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, 24(5-6):376–380.

Hasija, S. and Rajendran, C. (2004). Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 42(11):2289–2301.

Hejazi, S. R. and Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, 43(14):2895–2929.

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA.

Hooker, J. (1995). Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:33–42.

Hoos, H. H. and Stützle, T. (2004). *Stochastic Local Search: Foundations and Applications*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, USA.

Jenabi, M., Fatemi Ghomi, S. M. T., Torabi, S. A., and Karimi, B. (2007). Two hybrid meta-heuristics for the finite horizon elsp in flexible flow lines with unrelated parallel machines. *Applied Mathematics and Computation*, 186(1):230–245.

Jin, Z. H., Ohno, K., Ito, T., and Elmaghraby, S. E. (2002). Scheduling hybrid flowshops in printed circuit board assembly lines. *Production and Operations Management*, 11(2):216–230.

Jin, Z. H., Yang, Z., and Ito, T. (2006). Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics*, 100(2):322–334.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F. (2008). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *International Journal of Advanced Manufacturing Technology*, 37(3-4):354–370.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., and Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research*, 36(2):358–378.

Kass, G. V. (1980). En exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127.

Kis, T. and Pesch, E. (2005). A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *European Journal of Operational Research*, 164(3):592–608.

Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial of stochastic multiobjective optimizers. Technical Report Technical Report 214, Computer Engineering and Networks Laboratory (TIK), ETH. Revised version.

Kochhar, S., Morris, R. J. T., and Wong, W. S. (1988). The local search approach to flexible flow line scheduling. *Engineering Costs and Production Economics*, 14(1):25–37.

Kurz, M. E. and Askin, R. G. (2003). Comparing scheduling rules for flexible flow lines. *International Journal of Production Economics*, 85(3):371–388.

Kurz, M. E. and Askin, R. G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, 159(1):66–82.

Kwok, Y. K. and Ahmad, I. (1997). Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm. *Journal of Parallel and Distributed Computing*, 47(1):58–77.

Ledbetter, W. N. and Cox, J. F. (1977). Operations research in production management: An investigation of past and present utilization. *Production and Inventory Management*, 18(3):84–91.

Lee, C. Y. and Vairaktarakis, G. L. (1994). Minimizing makespan in hybrid flowshops. *Operations Research Letters*, 16(3):149–158.

Lee, I., Sikora, R., and Shaw, M. J. (1997). A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes. *Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 27(1):36–54.

Leon, V. J. and Ramamoorthy, B. (1997). An adaptable problem-space-based search method for flexible flow line scheduling. *IIE Transactions*, 29(2):115–125.

Li, K. and Yang, S. L. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied Mathematical Modelling*, 33(4):2145–2158.

Linn, R. and Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers and Industrial Engineering*, 37(1-2):57–61.

Lohl, T., Schulz, C., and Engell, S. (1998). Sequencing of batch operations for a highly coupled production process: Genetic algorithms versus mathematical programming. *Computers and Chemical Engineering*, 22:S579–S585.

Lourenço, H., Martin, O., and Stützle, T. (2002). Iterated local search. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 321–353, Norwell, MA, USA. Kluwer Academic Publishers.

Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers and Operations Research*, 32(8):2013–2025.

MacCarthy, B. L. and Liu, J. Y. (1993). Addressing the gap in scheduling research - a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1):59–79.

Martin, O. C., Otto, S. W., and Felten, E. W. (1991). Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–326.

McKay, K. N., Pinedo, M., and Webster, S. (2002). Practice-focused research issues for scheduling systems. *Production and Operations Management*, 11(2):249–258.

McKay, K. N., Safayeni, F. R., and Buzacott, J. A. (1988). Job-shop scheduling theory - what is relevant. *Interfaces*, 18(4):84–90.

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Minella, G., Ruiz, R., and Ciavotta, M. (2008). A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3):451–471.

Morgan, J. N. and Sonquist, J. A. (1963). Problems in analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434.

Naderi, B. and Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research*, 37(4):754–768.

Naderi, B., Ruiz, R., and Zandieh, M. (2010). Algorithms for a realistic variant of flowshop scheduling. *Computers & Operations Research*, 37(2):236–246.

Nawaz, M., Enscore, E. E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. *Omega-International Journal of Management Science*, 11(1):91–95.

Nossal, R. (1998). An evolutionary approach to multiprocessor scheduling of dependent tasks. *Future Generation Computer Systems*, 14(5-6):383–392.

Nowicki, E. and Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813.

Nowicki, E. and Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research*, 106(2-3):226–253.

Oduguwa, V., Tiwari, A., and Roy, R. (2005). Evolutionary computing in manufacturing industry: an overview of recent applications. *Applied Soft Computing*, 5(3):281–299.

Oguz, C. and Ercan, M. (2005). A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling*, 8(4):323–351.

Olhager, J. and Rapp, B. (1995). Operations research techniques in manufacturing planning and control systems. *International Transactions in Operational Research*, 2(1):7–13.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms and Systems*. Springer, New York, third edition.

Potts, C. N. and Van Wassenhove, L. N. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, 1:177–81.

Proth, J. M. (2007). Scheduling: New trends in industrial environment. *Annual Reviews in Control*, 31(1):157–166.

Quadt, D. and Kuhn, H. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, 178(3):686–698.

Rad, S. F., Ruiz, R., and Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops. *Omega-International Journal of Management Science*, 37(2):331–345.

Rahimi-Vahed, A. R., Mirghorbani, S. M., and Rabbani, M. (2007). A new particle swarm algorithm for a multi-objective mixed-model assembly line sequencing problem. *Soft Computing*, 11(10):997–1012.

Rajendran, C. and Chaudhuri, D. (1992). A multistage parallel-processor flowshop problem with minimum flowtime. *European Journal of Operational Research*, 57(1):111–122.

Rajendran, C. and Ziegler, H. (1997). A heuristic for scheduling to minimize the sum of weighted flowtime of jobs in a flowshop with sequence-dependent setup times of jobs. *Computers & Industrial Engineering*, 33(1-2):281–284.

Ramachandra, G. and Elmaghraby, S. E. (2006). Sequencing precedence-related jobs on two machines to minimize the weighted completion time. *International Journal of Production Economics*, 100(1):44–58.

Reeves, C. (1995a). Heuristics for scheduling a single-machine subject to unequal job release times. *European Journal of Operational Research*, 80(2):397–403.

Reeves, C. R. (1995b). A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, 22(1):5–13.

Reisman, A., Kumar, A., and Motwani, J. (1997). Flowshop scheduling/sequencing research: A statistical review of the literature, 1952-1994. *Ieee Transactions on Engineering Management*, 44(3):316–329.

Ribas, I., Leisten, R., and Framinan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production systems and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439–1454.

Rinnooy Kan, A. H. G. (1976). Machine scheduling problems: Clasification, complexity and computations. Ph.D. Thesis.

Ruiz, R. (2003). Técnicas metaheurísticas para la programación flexible de la producción. Ph.D. Thesis. In Spanish.

Ruiz, R. and Allahverdi, A. (2007a). No-wait flowshop with separate setup times to minimize maximum lateness. *International Journal of Advanced Manufacturing Technology*, 35(5-6):551–565.

Ruiz, R. and Allahverdi, A. (2007b). Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. *Annals of Operations Research*, 156(1):143–171.

Ruiz, R. and Allahverdi, A. (2009). Minimizing the bicriteria of makespan and maximum tardiness with an upper bound on maximum tardiness. *Computers & Operations Research*, 36(4):1268–1283.

Ruiz, R. and Andrés, C. (2007). Scheduling unrelated parallel machines with resource-assignable sequence dependent setup times. Technical Report DEIOAC-2007-01, Universidad Politécnica de Valencia.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494.

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3):781–800.

Ruiz, R., Maroto, C., and Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics - discrete optimization. *European Journal of Operational Research*, 165(1):34–54.

Ruiz, R., Maroto, C., and Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega-International Journal of Management Science*, 34(5):461–476.

Ruiz, R., Sivrikaya Şerifoğlu, F., and Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers and Operations Research*, 35(4):1151–1175.

Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049.

Ruiz, R. and Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187(3):1143–1159.

Ruiz, R. and Vázquez Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1–18.

Salveson, M. E. (1952). On a quantitative method in production planning and scheduling. *Econometrica*, 20(4):554–590.

Santos, D. L., Hunsucker, J. L., and Deal, D. E. (1995). Global lower bounds for flow shops with multiple processors. *European Journal of Operational Research*, 80(1):112–120.

Sawik, T. (2000). Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, 31(13):39–52.

Schutten, J. M. J. (1998). Practical job shop scheduling. *Annals of Operations Research*, 83:161–177.

Segura, B., Vallada, E., Maroto, C., and Ruiz, R. (2004). Operations strategy in spanish tile industry firms. *Boletin de la Sociedad Espanola de Ceramica y Vidrio*, 43(6):929–932. In Spanish.

Sivrikaya Şerifoğlu, F. and Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, 26(8):773–787.

Sivrikaya Şerifoğlu, F. and Ulusoy, G. (2004). Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach. *Journal of the Operational Research Society*, 55(5):504–512.

Srivinas, N. K. and Deb, K. (1994). Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(1):221–248.

Stafford, E. F. and Tseng, F. T. (2002). Two models for a family of flowshop sequencing problems. *European Journal of Operational Research*, 142(2):282–293.

Stützle, T. (1998). Applying iterated local search to the permutation flow shop problem. Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt.

Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539.

Taillard, E. (1990). Some efficient heuristic methods for the flow-shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74.

Tanev, I. T., Uozumi, T., and Morotome, Y. (2004). Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach. *Applied Soft Computing*, 5(1):87–100.

Tavakkoli-Moghaddam, R., Safaei, N., and Sassani, F. (2009). A memetic algorithm for the flexible flow line scheduling problem with processor blocking. *Computers & Operations Research*, 36(2):402–414.

Torabi, S. A., Ghomi, S. M. T. F., and Karimi, B. (2006). A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains. *European Journal of Operational Research*, 173(1):173–189.

Ullman, J. D. (1975). Np-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3):384–393.

Urlings, T. and Ruiz, R. (2007). Local search in complex scheduling problems. In T. Stützle, M. Birattari, H. H., editor, *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, volume 4638 of *Lecture Notes in Computer Science*, pages 202–206. Springer Verlag / Heidelberg.

Urlings, T., Ruiz, R., and Sivrikaya Şerifoğlu, F. (2010a). Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems. *International Journal of Metaheuristics*, 1(1):30–54.

Urlings, T., Ruiz, R., and Stützle, T. (2010b). Shifting representation search for hybrid flexible flowline. *European Journal of Operational Research*. doi: 10.1016/j.ejor.2010.05.041.

Vallada, E., Maroto, C., Ruiz, R., and Segura, B. (2005). Analysis of production scheduling in spanish tile industry. *Boletin de la Sociedad Espanola de Ceramica y Vidrio*, 44(1):39–44. In Spanish.

Vallada, E. and Ruiz, R. (2009). Genetic algorithms for the unrelated parallel machine scheduling problem with sequence dependent setup times. Technical Report DEIOAC-2009-04, Universidad Politécnica de Valencia.

Vallada, E., Ruiz, R., and Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*, 35(4):1350–1373.

Vignier, A., Billaut, J. C., and Proust, C. (1999). Hybrid flowshop scheduling problems: State of the art. *Rairo-Recherche Operationnelle-Operations Research*, 33(2):117–183.

Voss, S. (1993). The two-stage hybrid flow shop with sequence-dependent setup times. In A. Jones, G. Fandel, T. G., editor, *Operations Research in Production Planning and Control, Proceedings of a Joint German/US Conference*, pages 215–220. Springer-Verlag.

Wang, L., Siegel, H. J., Roychowdhury, V. P., and Maciejewski, A. A. (1997). Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. *Journal of Parallel and Distributed Computing*, 47(1):8–22.

Woo, S.-H., Yang, S.-B., Kim, S.-D., and Han, T.-D. (1997). Task scheduling in distributed computing systems with a genetic algorithm. In *HPC-ASIA '97: Proceedings of the High-Performance Computing on the Information Superhighway, HPC-Asia '97*, page 301, Washington, DC, USA. IEEE Computer Society.

Yang, T., Kuo, Y., and Cho, C. (2007). A genetic algorithms simulation approach for the multi-attribute combinatorial dispatching decision problem. *European Journal of Operational Research*, 176(3):1859–1873.

Zitzler, E., Knowles, J., and Thiele, L. (2008). Quality assessment of pareto set approximations. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 373–404, Berlin, Heidelberg. Springer-Verlag. doi: 10.1007/978-3-540-88908-3_14.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.

APPENDIX A

DATA FOR FIGURES

In this appendix, the data is given for many analyses in the text, where we preferred not to interrupt the reader with means tables. For completeness, however, we did not want to leave out those data. The tables are given in the order that the respective analyses appear in this thesis and the different chapters are indicated between them.

Chapter 4

Assignment rule	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
1	192	60.3378	0.968766	57.8424	62.8332
2	192	43.3786	0.968766	40.8832	45.874
3	192	35.4981	0.968766	33.0027	37.9935
4	192	32.7386	0.968766	30.2432	35.234
5	192	33.3729	0.968766	30.8776	35.8683
6	192	64.7043	0.968766	62.2089	67.1997
7	192	34.3306	0.968766	31.8353	36.826
8	192	43.3549	0.968766	40.8595	45.8502
9	192	45.6835	0.968766	43.1881	48.1789
All	192	26.9882	0.968766	24.4928	29.4836
Job1	192	34.4807	0.968766	31.9853	36.976
Job3	192	29.8532	0.968766	27.3578	32.3486

Table A.1: NEH heuristic with distinct machine assignment methods. Table of means and 99% confidence intervals for the large instances. Deviation from best known solution value.

Heuristic	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
LPT	192	222.779	1.72908	218.326	227.233
LWR	192	137.419	1.72908	132.965	141.873
MWR	192	95.568	1.72908	91.1142	100.022
MWRST	192	92.5678	1.72908	88.1139	97.0216
NEH	192	27.7906	1.72908	23.3368	32.2445
SPT	192	225.074	1.72908	220.621	229.528

Table A.2: Comparison of heuristic methods. Table of means and 99% confidence intervals for the large instances. Deviation from best known solution value.

Heuristic	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
LPT	4322	33.5335	0.339439	32.6591	34.4078
LWR	4322	35.7746	0.339439	34.9002	36.6489
MWR	4322	25.1718	0.339439	24.2975	26.0461
MWRST	4322	24.7208	0.339439	23.8465	25.5952
NEH	4322	12.5832	0.339439	11.7089	13.4575
SPT	4322	34.5381	0.339439	33.6638	35.4124

Table A.3: Comparison of heuristics. Table of means and 99% confidence intervals for the small instances. Deviation from the optimum.

Chapter 5

Level		Count	Mean	Stnd. Error	Lower Limit	Upper Limit
Algorithm						
BGA		4320	0.821561	0.0224735	0.763673	0.879449
EGA		4320	2.00096	0.0224735	1.94307	2.05884
SGA		4320	0.702576	0.0224735	0.644688	0.760463
Algorithm by NP_j						
BGA	0-0	2160	0.55055	0.0317823	0.468685	0.632416
	1-3	2160	1.09257	0.0317823	1.01071	1.17444
EGA	0-0	2160	2.17322	0.0317823	2.09136	2.25509
	1-3	2160	1.82869	0.0317823	1.74682	1.91056
SGA	0-0	2160	0.353784	0.0317823	0.271918	0.43565
	1-3	2160	1.05137	0.0317823	0.969501	1.13323
Algorithm by t						
BGA	5	1440	0.94846	0.0389252	0.848195	1.04873
	25	1440	0.79214	0.0389252	0.691875	0.892405
	125	1440	0.724082	0.0389252	0.623818	0.824347
EGA	5	1440	2.41988	0.0389252	2.31962	2.52015
	25	1440	1.86981	0.0389252	1.76955	1.97008
	125	1440	1.71317	0.0389252	1.6129	1.81343
SGA	5	1440	0.788726	0.0389252	0.688461	0.888991
	25	1440	0.672914	0.0389252	0.572649	0.773179
	125	1440	0.646086	0.0389252	0.545821	0.746351

Table A.4: Table of means and 99% Tukey intervals for the genetic algorithms. Small instances with one machine per stage.

Algorithm	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
BGA	2070	0.473894	0.0334844	0.387644	0.560144
EGA	2070	0.755312	0.0334844	0.669062	0.841563
SGA	2070	0.466672	0.0334844	0.380422	0.552922

Table A.5: Table of means and 99% Tukey intervals for the genetic algorithms. Small instances with one machine per stage for which the optimum is known.

Level		Count	Mean	Stnd. Error	Lower Limit	Upper Limit
Algorithm						
BGA		4320	3.21587	0.0955778	2.96967	3.46206
EGA		4320	4.92046	0.0955778	4.67427	5.16665
SGA		4320	2.78079	0.0955778	2.53459	3.02698
SGAM		4320	5.41219	0.0955778	5.166	5.65838
SGAR		4320	3.0585	0.0955778	2.81231	3.30469
Algorithm by PE_{ij}						
BGA	50	2160	2.00547	0.135167	1.6573	2.35364
	100	2160	4.42626	0.135167	4.0781	4.77443
EGA	50	2160	2.34242	0.135167	1.99425	2.69059
	100	2160	7.4985	0.135167	7.15033	7.84666
SGA	50	2160	1.71887	0.135167	1.3707	2.06704
	100	2160	3.8427	0.135167	3.49453	4.19087
SGAM	50	2160	1.89161	0.135167	1.54344	2.23977
	100	2160	8.93278	0.135167	8.58461	9.28095
SGAR	50	2160	2.14398	0.135167	1.79581	2.49215
	100	2160	3.97301	0.135167	3.62484	4.32118
Algorithm by t						
BGA	5	1440	3.57115	0.165546	3.14474	3.99757
	25	1440	3.1263	0.165546	2.69988	3.55271
	125	1440	2.95015	0.165546	2.52373	3.37657
EGA	5	1440	5.64194	0.165546	5.21552	6.06836
	25	1440	4.70846	0.165546	4.28204	5.13488
	125	1440	4.41098	0.165546	3.98456	4.8374
SGA	5	1440	2.92176	0.165546	2.49534	3.34817
	25	1440	2.74213	0.165546	2.31571	3.16855
	125	1440	2.67848	0.165546	2.25206	3.10489
SGAM	5	1440	7.44111	0.165546	7.0147	7.86753
	25	1440	5.09536	0.165546	4.66894	5.52178
	125	1440	3.7001	0.165546	3.27368	4.12652
SGAR	5	1440	3.10368	0.165546	2.67726	3.5301
	25	1440	2.97738	0.165546	2.55096	3.40379
	125	1440	3.09444	0.165546	2.66802	3.52086

Table A.6: Table of means and 99% Tukey intervals for the genetic algorithms. Small instances with three machines per stage.

Algorithm	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
BGA	2010	0.35934	0.16651	0.31645	0.40223
EGA	2010	0.26741	0.16651	0.22452	0.31030
SGA	2010	0.35115	0.16651	0.30826	0.39404
SGAM	2010	0.36996	0.16651	0.32706	0.41285
SGAR	2010	0.40766	0.16651	0.36477	0.45055

Table A.7: Table of means and 99% Tukey intervals for the genetic algorithms. Small instances with three machines per stage for which the optimum solution is known.

Level		Count	Mean	Stnd. Error	Lower Limit	Upper Limit
Algorithm						
BGA		2880	8.8056	0.08263	8.5927	9.0184
EGA		2880	18.6670	0.08263	18.4542	18.8799
SGA		2880	6.6548	0.08263	6.4420	6.8677
SGAM		2880	13.6697	0.08263	13.4569	13.8825
SGAR		2880	6.9965	0.08263	6.7837	7.2093
Algorithm by NP_j						
BGA	0	1440	6.4223	0.11686	6.1213	6.7233
	$U[1,5]$	1440	11.1889	0.11686	10.8879	11.4899
EGA	0	1440	8.7110	0.11686	8.4100	9.0120
	$U[1,5]$	1440	28.6231	0.11686	28.3221	28.9241
SGA	0	1440	4.8120	0.11686	4.5110	5.1130
	$U[1,5]$	1440	8.4976	0.11686	8.1966	8.7986
SGAM	0	1440	7.1490	0.11686	6.8480	7.4500
	$U[1,5]$	1440	20.1904	0.11686	19.8894	20.4914
SGAR	0	1440	4.9821	0.11686	4.6811	5.2831
	$U[1,5]$	1440	9.0108	0.11686	8.7098	9.3118
Algorithm by t						
BGA	5	960	10.6829	0.14312	10.3143	11.0516
	25	960	8.3856	0.14312	8.0170	8.7543
	125	960	7.3481	0.14312	6.9795	7.7168
EGA	5	960	22.7095	0.14312	22.3408	23.0781
	25	960	18.3607	0.14312	17.9921	18.7294
	125	960	14.9309	0.14312	14.5622	15.2995
SGA	5	960	9.0301	0.14312	8.6615	9.3988
	25	960	5.9149	0.14312	5.5463	6.2835
	125	960	5.0194	0.14312	4.6508	5.3880
SGAM	5	960	17.4448	0.14312	17.0762	17.8135
	25	960	12.5643	0.14312	12.1957	12.9330
	125	960	11.0000	0.14312	10.6313	11.3686
SGAR	5	960	9.4481	0.14312	9.0795	9.8167
	25	960	6.3098	0.14312	5.9411	6.6784
	125	960	5.2316	0.14312	4.8630	5.6003

Table A.8: Table of means and 99% Tukey intervals for the genetic algorithms. Large instances

Level	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
BGA	4320	3.0980	0.10566	2.8259	3.3702
EGA	4320	4.7924	0.10566	4.5202	5.0645
LPT	288	33.2207	0.40921	32.1666	34.2748
LWR	288	32.2851	0.40921	31.2310	33.3391
MIP	271	16.5266	0.42593	15.4294	17.6237
MWR	288	25.4196	0.40921	24.3655	26.4736
MWRST	288	25.5293	0.40921	24.4753	26.5834
NEH	288	9.0387	0.40921	7.9846	10.0928
RS	4320	4.1938	0.10566	3.9216	4.4659
SGA	4320	2.6643	0.10566	2.3921	2.9364
SGAM	4320	5.2844	0.10566	5.0123	5.5566
SGAR	4320	2.9422	0.10566	2.6701	3.2144
SPT	288	34.3612	0.40921	33.3071	35.4152

Table A.9: Table of means and 99% Tukey intervals for the genetic algorithms and the heuristics. Small instances with three machines per stage.

Level	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
BGA	4320	0.6738	0.13728	0.3202	1.0274
EGA	4320	1.8488	0.13728	1.4952	2.2024
LPT	288	49.6651	0.53168	48.2956	51.0346
LWR	288	41.2746	0.53168	39.9051	42.6441
MIP	250	8.1803	0.57088	6.7098	9.6507
MWR	288	30.1554	0.53168	28.7859	31.5249
MWRST	288	29.5640	0.53168	28.1945	30.9335
NEH	288	4.5977	0.53168	3.2282	5.9672
RS	4320	1.7264	0.13728	1.3728	2.0800
SGA	4320	0.5552	0.13728	0.2016	0.9088
SPT	288	51.4959	0.53168	50.1264	52.8654

Table A.10: Table of means and 99% Tukey intervals for the genetic algorithms and the heuristics. Small instances with one machine per stage.

Level	**Count**	**Mean**	**Stnd. Error**	**Lower Limit**	**Upper Limit**
BGA	2880	8.8056	0.38343	7.8179	9.7932
EGA	2880	18.6670	0.38343	17.6794	19.6547
LPT	192	225.0500	1.48500	221.2250	228.8750
LWR	192	138.1520	1.48500	134.3270	141.9770
MWR	192	95.7769	1.48500	91.9518	99.6021
MWRST	192	92.7195	1.48500	88.8944	96.5446
NEH	192	25.7333	1.48500	21.9082	29.5585
RS	2880	35.6955	0.38343	34.7079	36.6832
SGA	2880	6.6548	0.38343	5.6672	7.6425
SGAM	2880	13.6697	0.38343	12.6821	14.6573
SGAR	2880	6.9965	0.38343	6.0089	7.9841
SPT	192	227.0340	1.48500	223.2090	230.8600

Table A.11: Table of means and 99% Tukey intervals for the genetic algorithms and the heuristics. Large instances.

Chapter 6

Level		Count	Mean	Stnd. Error	Lower Limit	Upper Limit
Algorithm						
IG		2880	5.9096	0.04983	5.7812	6.0379
ILS		2880	7.1641	0.04983	7.0358	7.2925
MA		2880	6.2947	0.04983	6.1663	6.4231
SGA		2880	6.6548	0.04983	6.5265	6.7832
Algorithm by PE_{ij}						
IG	50	1440	5.3826	0.07047	5.2011	5.5641
	100	1440	6.4366	0.07047	6.2550	6.6181
ILS	50	1440	6.4584	0.07047	6.2769	6.6400
	100	1440	7.8699	0.07047	7.6883	8.0514
MA	50	1440	7.2260	0.07047	7.0444	7.4075
	100	1440	5.3634	0.07047	5.1819	5.5450
SGA	50	1440	7.4196	0.07047	7.2380	7.6011
	100	1440	5.8901	0.07047	5.7085	6.0716
Algorithm by t						
IG	5	960	9.0909	0.08631	8.8686	9.3132
	25	960	5.4911	0.08631	5.2688	5.7134
	125	960	3.1468	0.08631	2.9244	3.3691
ILS	5	960	10.7903	0.08631	10.5679	11.0126
	25	960	6.7486	0.08631	6.5263	6.9709
	125	960	3.9535	0.08631	3.7312	4.1759
MA	5	960	10.2260	0.08631	10.0036	10.4483
	25	960	5.6552	0.08631	5.4329	5.8776
	125	960	3.0029	0.08631	2.7806	3.2252
SGA	5	960	9.0301	0.08631	8.8078	9.2525
	25	960	5.9149	0.08631	5.6926	6.1372
	125	960	5.0194	0.08631	4.7971	5.2417

Table A.12: Comparison of the local search algorithms with SGA. Table of means and 99% Tukey intervals for the large instances.

Algorithm	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
IG	2880	6.84441	0.0539577	6.70542	6.9834
ILS	2880	8.11402	0.0539577	7.97503	8.253
MA	2880	7.2531	0.0539577	7.11411	7.39208
MGA	2880	9.07505	0.0539577	8.93607	9.21404
SGA	2880	7.61617	0.0539577	7.47718	7.75515
SRS	2880	5.81073	0.0539577	5.67174	5.94971

Table A.13: Comparison of the SRS and the MGA with earlier presented algorithms. Table of means and 99% Tukey intervals for the large instances.

Table A.14: Comparison of the SRS and the MGA with earlier presented algorithms. Table of means and 99% Tukey intervals for the small instances with three machines per stage.

Level	Count	Mean	Stnd. Error	Lower Limit	Upper Limit
Algorithm					
BGA	4320	3.21	0.081	3.00	3.42
EGA	4320	4.92	0.081	4.71	5.12
IG	4320	3.46	0.081	3.25	3.66
MGA	4320	2.45	0.081	2.24	2.66
SGA	4320	2.78	0.081	2.57	2.99
SGAM	4320	5.41	0.081	5.20	5.62
SGAR	4320	3.05	0.081	2.85	3.26
SRS	4320	0.60	0.081	0.39	0.81
t					
5	11520	3.74	0.050	3.61	3.87
25	11520	3.12	0.050	2.99	3.24
125	11520	2.84	0.050	2.72	2.97
n					
5	5760	2.26	0.070	2.08	2.44
7	5760	2.87	0.070	2.69	3.05
9	5760	2.94	0.070	2.76	3.12
11	5760	3.53	0.070	3.35	3.71
13	5760	3.53	0.070	3.35	3.71
15	5760	4.28	0.070	4.10	4.46

m						
2		17280	2.90	0.040	2.79	3.00
3		17280	3.57	0.040	3.47	3.68
F						
0		17280	4.57	0.040	4.47	4.68
50		17280	1.89	0.040	1.79	2.00
E						
50		17280	1.80	0.040	1.70	1.91
100		17280	4.67	0.040	4.56	4.77
P						
0-0		17280	2.25	0.040	2.15	2.36
1-5		17280	4.21	0.040	4.11	4.32
Replicate						
1		11520	3.51	0.050	3.39	3.64
2		11520	3.08	0.050	2.96	3.21
3		11520	3.10	0.050	2.98	3.23
Algorithm by t						
BGA	5	1440	3.57	0.140	3.21	3.93
BGA	25	1440	3.12	0.140	2.76	3.48
BGA	125	1440	2.95	0.140	2.59	3.31
EGA	5	1440	5.64	0.140	5.28	6.00
EGA	25	1440	4.70	0.140	4.34	5.07
EGA	125	1440	4.41	0.140	4.05	4.77
IG	5	1440	3.54	0.140	3.18	3.90
IG	25	1440	3.43	0.140	3.07	3.80
IG	125	1440	3.39	0.140	3.03	3.75
MGA	5	1440	2.79	0.140	2.43	3.15
MGA	25	1440	2.32	0.140	1.96	2.68
MGA	125	1440	2.24	0.140	1.87	2.60
SGA	5	1440	2.92	0.140	2.56	3.28
SGA	25	1440	2.74	0.140	2.38	3.10
SGA	125	1440	2.67	0.140	2.31	3.04
SGAM	5	1440	7.44	0.140	7.08	7.80
SGAM	25	1440	5.09	0.140	4.73	5.45
SGAM	125	1440	3.70	0.140	3.34	4.06
SGAR	5	1440	3.10	0.140	2.74	3.46
SGAR	25	1440	2.97	0.140	2.61	3.33
SGAR	125	1440	3.09	0.140	2.73	3.45
SRS	5	1440	0.93	0.140	0.57	1.29
SRS	25	1440	0.55	0.140	0.19	0.91
SRS	125	1440	0.32	0.140	-0.04	0.68

Algorithm by n

BGA	5	720	2.86	0.198	2.35	3.37
BGA	7	720	2.87	0.198	2.36	3.38
BGA	9	720	3.23	0.198	2.72	3.74
BGA	11	720	3.16	0.198	2.65	3.67
BGA	13	720	3.13	0.198	2.62	3.65
BGA	15	720	4.02	0.198	3.51	4.53
EGA	5	720	1.13	0.198	0.62	1.64
EGA	7	720	3.05	0.198	2.54	3.56
EGA	9	720	3.68	0.198	3.17	4.20
EGA	11	720	5.69	0.198	5.18	6.21
EGA	13	720	7.14	0.198	6.63	7.65
EGA	15	720	8.80	0.198	8.29	9.32
IG	5	720	3.22	0.198	2.71	3.73
IG	7	720	4.54	0.198	4.03	5.05
IG	9	720	4.01	0.198	3.50	4.52
IG	11	720	3.73	0.198	3.22	4.24
IG	13	720	2.80	0.198	2.28	3.31
IG	15	720	2.44	0.198	1.93	2.95
MGA	5	720	2.34	0.198	1.83	2.85
MGA	7	720	2.19	0.198	1.68	2.70
MGA	9	720	2.11	0.198	1.60	2.62
MGA	11	720	2.38	0.198	1.87	2.89
MGA	13	720	2.43	0.198	1.92	2.94
MGA	15	720	3.25	0.198	2.74	3.76
SGA	5	720	2.86	0.198	2.35	3.37
SGA	7	720	2.79	0.198	2.28	3.30
SGA	9	720	2.81	0.198	2.30	3.32
SGA	11	720	2.59	0.198	2.08	3.10
SGA	13	720	2.42	0.198	1.91	2.93
SGA	15	720	3.18	0.198	2.67	3.69
SGAM	5	720	2.38	0.198	1.87	2.89
SGAM	7	720	3.35	0.198	2.84	3.86
SGAM	9	720	4.54	0.198	4.03	5.05
SGAM	11	720	7.30	0.198	6.79	7.81
SGAM	13	720	6.85	0.198	6.34	7.36
SGAM	15	720	8.03	0.198	7.52	8.54
SGAR	5	720	3.15	0.198	2.64	3.66
SGAR	7	720	3.78	0.198	3.27	4.29
SGAR	9	720	2.78	0.198	2.27	3.29
SGAR	11	720	2.62	0.198	2.11	3.13

SGAR	13	720	2.68	0.198	2.17	3.19
SGAR	15	720	3.31	0.198	2.80	3.82
SRS	5	720	0.15	0.198	-0.36	0.66
SRS	7	720	0.36	0.198	-0.15	0.87
SRS	9	720	0.36	0.198	-0.15	0.87
SRS	11	720	0.75	0.198	0.24	1.26
SRS	13	720	0.80	0.198	0.29	1.31
SRS	15	720	1.19	0.198	0.68	1.70
Algorithm by m						
BGA	2	2160	2.75	0.114	2.46	3.05
BGA	3	2160	3.67	0.114	3.38	3.97
EGA	2	2160	4.28	0.114	3.99	4.58
EGA	3	2160	5.55	0.114	5.26	5.85
IG	2	2160	3.24	0.114	2.95	3.53
IG	3	2160	3.67	0.114	3.38	3.97
MGA	2	2160	2.19	0.114	1.89	2.48
MGA	3	2160	2.71	0.114	2.42	3.01
SGA	2	2160	2.44	0.114	2.14	2.73
SGA	3	2160	3.12	0.114	2.82	3.41
SGAM	2	2160	4.85	0.114	4.55	5.14
SGAM	3	2160	5.97	0.114	5.67	6.26
SGAR	2	2160	2.98	0.114	2.68	3.27
SGAR	3	2160	3.13	0.114	2.84	3.43
SRS	2	2160	0.46	0.114	0.16	0.75
SRS	3	2160	0.74	0.114	0.45	1.04
Algorithm by F						
BGA	0	2160	4.71	0.114	4.42	5.01
BGA	50	2160	1.71	0.114	1.42	2.01
EGA	0	2160	7.40	0.114	7.10	7.69
EGA	50	2160	2.43	0.114	2.14	2.73
IG	0	2160	4.42	0.114	4.13	4.72
IG	50	2160	2.49	0.114	2.19	2.78
MGA	0	2160	3.68	0.114	3.38	3.97
MGA	50	2160	1.22	0.114	0.93	1.52
SGA	0	2160	4.01	0.114	3.72	4.31
SGA	50	2160	1.54	0.114	1.25	1.84
SGAM	0	2160	7.21	0.114	6.91	7.50
SGAM	50	2160	3.61	0.114	3.31	3.90
SGAR	0	2160	4.14	0.114	3.85	4.44
SGAR	50	2160	1.97	0.114	1.67	2.26
SRS	0	2160	1.02	0.114	0.73	1.32

SRS	50	2160	0.18	0.114	-0.12	0.47
Algorithm by E						
BGA	50	2160	2.00	0.114	1.71	2.30
BGA	100	2160	4.42	0.114	4.13	4.71
EGA	50	2160	2.34	0.114	2.05	2.64
EGA	100	2160	7.49	0.114	7.20	7.79
IG	50	2160	2.71	0.114	2.41	3.00
IG	100	2160	4.20	0.114	3.91	4.50
MGA	50	2160	1.36	0.114	1.07	1.66
MGA	100	2160	3.54	0.114	3.24	3.83
SGA	50	2160	1.72	0.114	1.42	2.01
SGA	100	2160	3.84	0.114	3.54	4.13
SGAM	50	2160	1.89	0.114	1.60	2.18
SGAM	100	2160	8.92	0.114	8.63	9.22
SGAR	50	2160	2.14	0.114	1.85	2.44
SGAR	100	2160	3.97	0.114	3.67	4.26
SRS	50	2160	0.26	0.114	-0.04	0.55
SRS	100	2160	0.94	0.114	0.65	1.24
Algorithm by P						
BGA	0-0	2160	1.86	0.114	1.57	2.16
BGA	1-5	2160	4.56	0.114	4.26	4.85
EGA	0-0	2160	5.05	0.114	4.76	5.35
EGA	1-5	2160	4.78	0.114	4.49	5.07
IG	0-0	2160	0.90	0.114	0.61	1.20
IG	1-5	2160	6.01	0.114	5.71	6.30
MGA	0-0	2160	1.48	0.114	1.18	1.77
MGA	1-5	2160	3.42	0.114	3.13	3.72
SGA	0-0	2160	1.32	0.114	1.02	1.61
SGA	1-5	2160	4.24	0.114	3.94	4.53
SGAM	0-0	2160	5.50	0.114	5.21	5.80
SGAM	1-5	2160	5.31	0.114	5.02	5.61
SGAR	0-0	2160	1.36	0.114	1.07	1.66
SGAR	1-5	2160	4.74	0.114	4.45	5.04
SRS	0-0	2160	0.55	0.114	0.25	0.84
SRS	1-5	2160	0.65	0.114	0.35	0.94

Chapter 8

Table A.15: Epsilon indicator means and 99% Tukey intervals - comparison of NSGA-II and RIPG for the set of large instances.

Level		**Count**	**Mean**	**Stnd.**	**Lower**	**Upper**
				Error	**Limit**	**Limit**
Grand mean		5760	1.43514			
n						
50		2880	1.42282	0.00457497	1.41103	1.4346
100		2880	1.44747	0.00457497	1.43568	1.45925
m						
4		2880	1.43921	0.00457497	1.42742	1.45099
8		2880	1.43108	0.00457497	1.4193	1.44286
m_i						
2		2880	1.45455	0.00457497	1.44276	1.46633
4		2880	1.41574	0.00457497	1.40395	1.42752
T						
0.2		1920	1.43851	0.00560317	1.42408	1.45294
0.4		1920	1.42548	0.00560317	1.41104	1.43991
0.6		1920	1.44144	0.00560317	1.42701	1.45587
R						
0.2		1920	1.43949	0.00560317	1.42506	1.45392
0.6		1920	1.42734	0.00560317	1.4129	1.44177
1		1920	1.4386	0.00560317	1.42417	1.45304
Method						
NSGA-II		2880	1.3736	0.00457497	1.36181	1.38538
RIPG		2880	1.49669	0.00457497	1.4849	1.50847
n by Method						
50	NSGA-II	1440	1.34026	0.00646999	1.3236	1.35693
50	RIPG	1440	1.50538	0.00646999	1.48871	1.52204
100	NSGA-II	1440	1.40693	0.00646999	1.39027	1.4236
100	RIPG	1440	1.488	0.00646999	1.47133	1.50467
m by Method						
4	NSGA-II	1440	1.35907	0.00646999	1.3424	1.37573
4	RIPG	1440	1.51934	0.00646999	1.50268	1.53601
8	NSGA-II	1440	1.38813	0.00646999	1.37146	1.40479
8	RIPG	1440	1.47403	0.00646999	1.45737	1.4907
m_i by Method						

2	NSGA-II	1440	1.35705	0.00646999	1.34038	1.37372
2	RIPG	1440	1.55205	0.00646999	1.53538	1.56871
4	NSGA-II	1440	1.39014	0.00646999	1.37348	1.40681
4	RIPG	1440	1.44133	0.00646999	1.42466	1.458
T by Method						
0.2	NSGA-II	960	1.37891	0.00792408	1.3585	1.39932
0.2	RIPG	960	1.49811	0.00792408	1.4777	1.51853
0.4	NSGA-II	960	1.36678	0.00792408	1.34636	1.38719
0.4	RIPG	960	1.48418	0.00792408	1.46377	1.50459
0.6	NSGA-II	960	1.37511	0.00792408	1.3547	1.39552
0.6	RIPG	960	1.50777	0.00792408	1.48736	1.52819
R by Method						
0.2	NSGA-II	960	1.38565	0.00792408	1.36524	1.40606
0.2	RIPG	960	1.49333	0.00792408	1.47292	1.51374
0.6	NSGA-II	960	1.36292	0.00792408	1.34251	1.38333
0.6	RIPG	960	1.49175	0.00792408	1.47134	1.51216
1	NSGA-II	960	1.37222	0.00792408	1.35181	1.39264
1	RIPG	960	1.50498	0.00792408	1.48457	1.52539

APPENDIX B

ANOVA TABLES

In this appendix, the Analysis of Variance tables are given, in the cases that they were not considered necessary or desirable in the text. The tables are sorted by chapter and given in the order of appearance of the analyses in the text.

Chapter 5

Table B.1: Analysis of Variance for the SGA calibration. Large instances.

Source	Sum of Squares	Degrees of freedom	Mean Square	F-Ratio	P-Value
Main effects					
A:Assignment	3202.54	1	3202.54	274.24	0.0000
B:Pc	1825.42	1	1825.42	156.32	0.0000
C:$Pmut$	23932.9	1	23932.9	2049.44	0.0000
D:Population	26273.7	2	13136.8	1124.95	0.0000
E:Selection	14567.4	2	7283.69	623.72	0.0000
F:n	901.162	1	901.162	77.17	0.0000

G:m	181249.0	1	181249.0	15520.83	0.0000
H:m_i	193016.0	1	193016.0	16528.46	0.0000
I:F	208451.0	1	208451.0	17850.25	0.0000
J:E	49425.7	1	49425.7	4232.46	0.0000
K:P	$1.36245E6$	1	$1.36245E6$	116670.74	0.0000
L:Replicate	269.495	2	134.748	11.54	0.0000
M:t	278029.0	1	278029.0	23808.42	0.0000
Interactions					
AB	2817.27	1	2817.27	241.25	0.0000
AC	301.84	1	301.84	25.85	0.0000
AD	1805.23	2	902.616	77.29	0.0000
AE	1194.27	2	597.135	51.13	0.0000
AF	19.2478	1	19.2478	1.65	0.1992
AG	78.127	1	78.127	6.69	0.0097
AH	129.863	1	129.863	11.12	0.0009
AI	26.1865	1	26.1865	2.24	0.1343
AJ	40.2896	1	40.2896	3.45	0.0632
AK	365.448	1	365.448	31.29	0.0000
AL	5.15415	2	2.57707	0.22	0.8020
AM	2299.89	1	2299.89	196.95	0.0000
BC	5.36951	1	5.36951	0.46	0.4977
BD	1107.01	2	553.505	47.40	0.0000
BE	996.562	2	498.281	42.67	0.0000
BF	295.456	1	295.456	25.30	0.0000
BG	71.6749	1	71.6749	6.14	0.0132
BH	233.779	1	233.779	20.02	0.0000
BI	20.5118	1	20.5118	1.76	0.1851
BJ	291.252	1	291.252	24.94	0.0000
BK	512.919	1	512.919	43.92	0.0000
BL	28.8749	2	14.4375	1.24	0.2905
BM	2144.72	1	2144.72	183.66	0.0000
CD	1563.95	2	781.977	66.96	0.0000
CE	872.673	2	436.337	37.36	0.0000
CF	54.5925	1	54.5925	4.67	0.0306
CG	322.007	1	322.007	27.57	0.0000
CH	3951.33	1	3951.33	338.36	0.0000
CI	1227.89	1	1227.89	105.15	0.0000
CJ	3.40819	1	3.40819	0.29	0.5890
CK	8914.45	1	8914.45	763.37	0.0000
CL	19.1024	2	9.55121	0.82	0.4414
CM	2494.65	1	2494.65	213.62	0.0000

DE	8788.45	4	2197.11	188.14	0.0000
DF	1159.29	2	579.647	49.64	0.0000
DG	796.155	2	398.077	34.09	0.0000
DH	13543.5	2	6771.73	579.88	0.0000
DI	2291.99	2	1146.0	98.13	0.0000
DJ	1324.27	2	662.135	56.70	0.0000
DK	15749.8	2	7874.92	674.35	0.0000
DL	277.839	4	69.4596	5.95	0.0001
DM	13377.1	2	6688.57	572.76	0.0000
EF	910.075	2	455.037	38.97	0.0000
EG	736.615	2	368.307	31.54	0.0000
EH	10127.7	2	5063.87	433.63	0.0000
EI	919.05	2	459.525	39.35	0.0000
EJ	1188.3	2	594.152	50.88	0.0000
EK	11816.9	2	5908.45	505.96	0.0000
EL	275.754	4	68.9385	5.90	0.0001
EM	11966.5	2	5983.27	512.36	0.0000
FG	27289.2	1	27289.2	2336.85	0.0000
FH	96.4808	1	96.4808	8.26	0.0040
FI	30.9012	1	30.9012	2.65	0.1038
FJ	29653.4	1	29653.4	2539.30	0.0000
FK	36108.0	1	36108.0	3092.03	0.0000
FL	4158.37	2	2079.19	178.05	0.0000
FM	7927.56	1	7927.56	678.86	0.0000
GH	4759.13	1	4759.13	407.54	0.0000
GI	38659.2	1	38659.2	3310.50	0.0000
GJ	7432.89	1	7432.89	636.50	0.0000
GK	89281.6	1	89281.6	7645.43	0.0000
GL	1677.72	2	838.862	71.83	0.0000
GM	5085.24	1	5085.24	435.46	0.0000
HI	1069.04	1	1069.04	91.54	0.0000
HJ	10549.1	1	10549.1	903.35	0.0000
HK	1794.94	1	1794.94	153.71	0.0000
HL	5574.66	2	2787.33	238.69	0.0000
HM	0.560325	1	0.560325	0.05	0.8266
IJ	23865.9	1	23865.9	2043.70	0.0000
IK	7324.13	1	7324.13	627.19	0.0000
IL	4065.43	2	2032.71	174.07	0.0000
IM	7.41688	1	7.41688	0.64	0.4255
JK	46955.6	1	46955.6	4020.94	0.0000
JL	12975.7	2	6487.85	555.57	0.0000

JM	1666.94	1	1666.94	142.74	0.0000
KL	131.514	2	65.7568	5.63	0.0036
KM	31560.5	1	31560.5	2702.62	0.0000
LM	587.914	2	293.957	25.17	0.0000
Residual	$1.61277E6$	138106	11.6778		
Total (corrected)	$4.48612E6$	138239			

Table B.2: Analysis of Variance for the SGA calibration, $Pmut$ fixed at 2%. Large instances.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:Assignment	769.006	1	769.006	71.99	0.0000
B:Pc	816.389	1	816.389	76.43	0.0000
C:Population	12297.5	2	6148.75	575.61	0.0000
D:Selection	4600.48	2	2300.24	215.34	0.0000
E:n	256.074	1	256.074	23.97	0.0000
F:m	83145.7	1	83145.7	7783.67	0.0000
G:m_i	70867.0	1	70867.0	6634.20	0.0000
H:F	88840.9	1	88840.9	8316.82	0.0000
I:E	25125.0	1	25125.0	2352.07	0.0000
J:P	575477.0	1	575477.0	53873.18	0.0000
K:Replicate	72.6041	2	36.3021	3.40	0.0334
L:t	166598.0	1	166598.0	15596.03	0.0000
Interactions					
AB	687.98	1	687.98	64.41	0.0000
AC	1240.91	2	620.457	58.08	0.0000
AD	183.993	2	91.9966	8.61	0.0002
AE	11.3374	1	11.3374	1.06	0.3029
AF	40.7426	1	40.7426	3.81	0.0508
AG	38.9567	1	38.9567	3.65	0.0562
AH	24.6893	1	24.6893	2.31	0.1284
AI	10.694	1	10.694	1.00	0.3170
AJ	87.7902	1	87.7902	8.22	0.0041
AK	7.11336	2	3.55668	0.33	0.7168
AL	613.332	1	613.332	57.42	0.0000
BC	1044.33	2	522.165	48.88	0.0000
BD	93.3222	2	46.6611	4.37	0.0127
BE	236.031	1	236.031	22.10	0.0000
BF	22.1743	1	22.1743	2.08	0.1496
BG	53.2414	1	53.2414	4.98	0.0256
BH	25.5196	1	25.5196	2.39	0.1222
BI	73.54	1	73.54	6.88	0.0087
BJ	364.02	1	364.02	34.08	0.0000
BK	4.08121	2	2.0406	0.19	0.8261
BL	424.465	1	424.465	39.74	0.0000
CD	4152.77	4	1038.19	97.19	0.0000

CE	950.43	2	475.215	44.49	0.0000
CF	498.939	2	249.469	23.35	0.0000
CG	6387.79	2	3193.89	299.00	0.0000
CH	1264.74	2	632.371	59.20	0.0000
CI	594.965	2	297.483	27.85	0.0000
CJ	5826.52	2	2913.26	272.72	0.0000
CK	168.496	4	42.124	3.94	0.0033
CL	7556.81	2	3778.41	353.71	0.0000
DE	712.836	2	356.418	33.37	0.0000
DF	303.413	2	151.707	14.20	0.0000
DG	4513.25	2	2256.63	211.25	0.0000
DH	620.692	2	310.346	29.05	0.0000
DI	569.399	2	284.7	26.65	0.0000
DJ	3978.3	2	1989.15	186.21	0.0000
DK	182.409	4	45.6021	4.27	0.0019
DL	5757.1	2	2878.55	269.47	0.0000
EF	10660.2	1	10660.2	997.96	0.0000
EG	167.267	1	167.267	15.66	0.0001
EH	8.5715	1	8.5715	0.80	0.3704
EI	16893.9	1	16893.9	1581.52	0.0000
EJ	12814.6	1	12814.6	1199.64	0.0000
EK	2157.44	2	1078.72	100.98	0.0000
EL	6129.91	1	6129.91	573.85	0.0000
FG	1483.81	1	1483.81	138.91	0.0000
FH	16854.6	1	16854.6	1577.84	0.0000
FI	4241.45	1	4241.45	397.06	0.0000
FJ	39025.4	1	39025.4	3653.36	0.0000
FK	677.805	2	338.903	31.73	0.0000
FL	2994.12	1	2994.12	280.29	0.0000
GH	914.144	1	914.144	85.58	0.0000
GI	2904.91	1	2904.91	271.94	0.0000
GJ	63.5549	1	63.5549	5.95	0.0147
GK	2894.09	2	1447.04	135.46	0.0000
GL	6.03509	1	6.03509	0.56	0.4523
HI	10495.7	1	10495.7	982.55	0.0000
HJ	2939.89	1	2939.89	275.22	0.0000
HK	1616.67	2	808.336	75.67	0.0000
HL	4.31842	1	4.31842	0.40	0.5249
IJ	21720.1	1	21720.1	2033.32	0.0000
IK	5561.11	2	2780.55	260.30	0.0000
IL	928.785	1	928.785	86.95	0.0000

JK	124.529	2	62.2643	5.83	0.0029
JL	17385.2	1	17385.2	1627.51	0.0000
KL	276.904	2	138.452	12.96	0.0000
Residual	737085.0	69002	10.6821		
Total (corrected)	$1.99722E6$	69119			

Table B.3: Analysis of Variance for the comparison of the genetic algorithms. Small instances with one machine per stage.

Source	Sum of Squares	Degrees of freedom	Mean Square	F-Ratio	P-Value
Main effects					
A:Algorithm	4428.78	2	2214.39	1207.06	0.0000
B:n	1651.37	5	330.273	180.03	0.0000
C:m	367.355	1	367.355	200.24	0.0000
D:F	1447.87	1	1447.87	789.23	0.0000
E:E	4.3657	1	4.3657	2.38	0.1229
F:P	201.796	1	201.796	110.00	0.0000
G:$repli$	236.312	2	118.156	64.41	0.0000
H:t	294.34	2	147.17	80.22	0.0000
I:Repetition	0.665597	4	0.166399	0.09	0.9854
Interactions					
AB	2478.81	10	247.881	135.12	0.0000
AC	116.096	2	58.048	31.64	0.0000
AD	657.427	2	328.714	179.18	0.0000
AE	3.62605	2	1.81303	0.99	0.3722
AF	681.836	2	340.918	185.83	0.0000
AG	37.6089	4	9.40222	5.13	0.0004
AH	141.601	4	35.4001	19.30	0.0000
AI	1.07317	8	0.134146	0.07	0.9998
BC	66.025	5	13.205	7.20	0.0000
BD	49.9946	5	9.99892	5.45	0.0001
BE	130.779	5	26.1559	14.26	0.0000
BF	128.994	5	25.7988	14.06	0.0000
BG	387.639	10	38.7639	21.13	0.0000
BH	236.252	10	23.6252	12.88	0.0000
BI	5.35827	20	0.267914	0.15	1.0000
CD	2.30961	1	2.30961	1.26	0.2618
CE	116.164	1	116.164	63.32	0.0000
CF	29.7595	1	29.7595	16.22	0.0001
CG	41.9259	2	20.963	11.43	0.0000
CH	9.7273	2	4.86365	2.65	0.0706
CI	0.212923	4	0.0532307	0.03	0.9984
DE	27.2637	1	27.2637	14.86	0.0001
DF	824.798	1	824.798	449.59	0.0000
DG	360.635	2	180.318	98.29	0.0000

DH	24.5259	2	12.263	6.68	0.0013
DI	0.1055	4	0.0263749	0.01	0.9996
EF	1.57732	1	1.57732	0.86	0.3538
EG	80.0993	2	40.0496	21.83	0.0000
EH	0.660186	2	0.330093	0.18	0.8353
EI	0.181586	4	0.0453965	0.02	0.9988
FG	17.2451	2	8.62254	4.70	0.0091
FH	8.99881	2	4.4994	2.45	0.0861
FI	2.10122	4	0.525305	0.29	0.8870
GH	4.84247	4	1.21062	0.66	0.6198
GI	1.62405	8	0.203007	0.11	0.9989
HI	3.2787	8	0.409837	0.22	0.9868
Residual	23460.1	12788	1.83454		
Total (corrected)	38774.1	12959			

Chapter 7

Table B.4: Analysis of Variance for the Average deviation - comparison of the SRS and the MGA with earlier presented algorithms. Small instances with three machines per stage.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:Algorithm	66533.8	7	9504.83	336.22	0.0000
B:t	4865.41	2	2432.71	86.05	0.0000
C:n	13993.1	5	2798.63	99.00	0.0000
D:m	3928.37	1	3928.37	138.96	0.0000
E:F	62090.0	1	62090.0	2196.36	0.0000
F:E	70784.1	1	70784.1	2503.90	0.0000
G:P	33224.0	1	33224.0	1175.26	0.0000
H:Replicate	1357.71	2	678.854	24.01	0.0000
Interactions					
AB	7527.61	14	537.686	19.02	0.0000
AC	39025.5	35	1115.01	39.44	0.0000
AD	1215.29	7	173.614	6.14	0.0000
AE	11275.7	7	1610.81	56.98	0.0000
AF	34077.2	7	4868.17	172.21	0.0000
AG	28555.8	7	4079.4	144.30	0.0000
AH	1515.91	14	108.28	3.83	0.0000
BC	860.929	10	86.0929	3.05	0.0007
BD	47.1528	2	23.5764	0.83	0.4343
BE	477.387	2	238.693	8.44	0.0002
BF	2289.09	2	1144.55	40.49	0.0000
BG	149.693	2	74.8466	2.65	0.0708
BH	117.587	4	29.3967	1.04	0.3849
CD	3047.53	5	609.507	21.56	0.0000
CE	1633.52	5	326.705	11.56	0.0000
CF	2149.74	5	429.948	15.21	0.0000
CG	3828.7	5	765.741	27.09	0.0000
CH	9494.62	10	949.462	33.59	0.0000
DE	3937.01	1	3937.01	139.27	0.0000
DF	28.3789	1	28.3789	1.00	0.3164
DG	107.043	1	107.043	3.79	0.0517
DH	136.433	2	68.2165	2.41	0.0896

EF	9880.97	1	9880.97	349.53	0.0000
EG	439.12	1	439.12	15.53	0.0001
EH	1212.27	2	606.136	21.44	0.0000
FG	3219.84	1	3219.84	113.90	0.0000
FH	1385.69	2	692.843	24.51	0.0000
GH	127.625	2	63.8124	2.26	0.1046
Residual	971961.0	34382	28.2695		
Total (corrected)	$1.3965E6$	34559			

Chapter 8

Table B.5: Analysis of Variance for the Hypervolume - Calibration of NSGA-II, mutation probability fixed.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	0.893666	1	0.893666	30.30	0.0000
B:m	0.290182	1	0.290182	9.84	0.0017
C:m_i	47.1426	1	47.1426	1598.48	0.0000
D:T	0.169023	2	0.0845113	2.87	0.0570
E:R	0.214626	2	0.107313	3.64	0.0263
F:Pop	16.8469	2	8.42346	285.62	0.0000
G:Cross	1.53251	2	0.766257	25.98	0.0000
H:Rep	0.621627	4	0.155407	5.27	0.0003
Interactions					
AB	0.331328	1	0.331328	11.23	0.0008
AC	2.4817	1	2.4817	84.15	0.0000
AD	0.532526	2	0.266263	9.03	0.0001
AE	0.0494439	2	0.0247219	0.84	0.4325
AF	0.95501	2	0.477505	16.19	0.0000
AG	0.360581	2	0.18029	6.11	0.0022
AH	0.190608	4	0.047652	1.62	0.1673
BC	2.20963	1	2.20963	74.92	0.0000
BD	0.197266	2	0.0986332	3.34	0.0353
BE	0.084421	2	0.0422105	1.43	0.2391
BF	0.018136	2	0.00906801	0.31	0.7353
BG	0.107415	2	0.0537076	1.82	0.1619
BH	0.125949	4	0.0314873	1.07	0.3707
CD	1.70039	2	0.850193	28.83	0.0000
CE	0.310947	2	0.155474	5.27	0.0052
CF	0.0331317	2	0.0165658	0.56	0.5703
CG	0.370769	2	0.185384	6.29	0.0019
CH	0.10225	4	0.0255625	0.87	0.4830
DE	3.48334	4	0.870836	29.53	0.0000
DF	0.121684	4	0.0304211	1.03	0.3893
DG	0.0988174	4	0.0247044	0.84	0.5010
DH	0.335035	8	0.0418794	1.42	0.1824
EF	0.0423573	4	0.0105893	0.36	0.8379

EG	0.0419791	4	0.0104948	0.36	0.8401
EH	0.250832	8	0.031354	1.06	0.3859
FG	0.18878	4	0.047195	1.60	0.1713
FH	0.183777	8	0.0229721	0.78	0.6213
GH	0.222267	8	0.0277834	0.94	0.4801
Residual	187.364	6353	0.0294922		
Total (corrected)	272.269	6479			

Table B.6: Analysis of Variance for the Hypervolume - Calibration of NSGA-II, mutation probability and population size fixed.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	0.0096754	1	0.0096754	0.39	0.5334
B:m	0.0883773	1	0.0883773	3.54	0.0598
C:m_i	14.6164	1	14.6164	586.05	0.0000
D:T	0.146424	2	0.0732122	2.94	0.0533
E:R	0.032214	2	0.016107	0.65	0.5243
F:Cross	0.534246	2	0.267123	10.71	0.0000
G:Rep	0.230427	4	0.0576067	2.31	0.0558
Interactions					
AB	0.059378	1	0.059378	2.38	0.1228
AC	0.781786	1	0.781786	31.35	0.0000
AD	0.809964	2	0.404982	16.24	0.0000
AE	0.177993	2	0.0889967	3.57	0.0284
AF	0.300177	2	0.150088	6.02	0.0025
AG	0.483891	4	0.120973	4.85	0.0007
BC	0.532596	1	0.532596	21.35	0.0000
BD	0.0299214	2	0.0149607	0.60	0.5490
BE	0.0289296	2	0.0144648	0.58	0.5600
BF	0.138773	2	0.0693863	2.78	0.0621
BG	0.0862298	4	0.0215575	0.86	0.4846
CD	0.343405	2	0.171702	6.88	0.0010
CE	0.258943	2	0.129472	5.19	0.0056
CF	0.234615	2	0.117308	4.70	0.0092
CG	0.118569	4	0.0296422	1.19	0.3138
DE	0.997242	4	0.24931	10.00	0.0000
DF	0.0133065	4	0.00332662	0.13	0.9701
DG	0.25203	8	0.0315038	1.26	0.2584
EF	0.102117	4	0.0255293	1.02	0.3936
EG	0.0822061	8	0.0102758	0.41	0.9143
FG	0.144391	8	0.0180489	0.72	0.6708
Residual	51.452	2063	0.0249404		
Total (corrected)	74.0182	2159			

Table B.7: Analysis of Variance for the Hypervolume - Calibration of RIPG, restart fixed.

Source	**Sum of**	**Degrees**	**Mean**	**F-Ratio**	**P-Value**
	Squares	**of freedom**	**Square**		
Main effects					
A:n	16.7304	1	16.7304	730.54	0.0000
B:m	0.982761	1	0.982761	42.91	0.0000
C:m_i	3.34221	1	3.34221	145.94	0.0000
D:T	0.4964	2	0.2482	10.84	0.0000
E:R	0.088878	2	0.044439	1.94	0.1437
F:Greedy	6.82459	2	3.41229	149.00	0.0000
G:LocalSearch	0.105901	2	0.0529505	2.31	0.0991
H:Rep	0.118847	4	0.0297117	1.30	0.2685
Interactions					
AB	3.80268	1	3.80268	166.05	0.0000
AC	2.07028	1	2.07028	90.40	0.0000
AE	2.30637	2	1.15319	50.35	0.0000
AE	0.699258	2	0.349629	15.27	0.0000
AF	18.4145	2	9.20723	402.04	0.0000
AG	2.81392	2	1.40696	61.44	0.0000
AH	0.0118271	4	0.00295678	0.13	0.9719
BC	1.47263	1	1.47263	64.30	0.0000
BD	0.386614	2	0.193307	8.44	0.0002
BE	1.59288	2	0.79644	34.78	0.0000
BF	0.523508	2	0.261754	11.43	0.0000
BG	0.00538029	2	0.00269015	0.12	0.8892
BH	0.0123125	4	0.00307812	0.13	0.9697
CD	0.240253	2	0.120126	5.25	0.0053
CE	0.00484679	2	0.0024234	0.11	0.8996
CF	24.8026	2	12.4013	541.51	0.0000
CG	0.649803	2	0.324902	14.19	0.0000
CH	0.132695	4	0.0331739	1.45	0.2152
DE	0.9847	4	0.246175	10.75	0.0000
DF	0.468092	4	0.117023	5.11	0.0004
DG	0.391536	4	0.097884	4.27	0.0019
DH	0.143224	8	0.017903	0.78	0.6188
EF	0.0288057	4	0.00720143	0.31	0.8685
EG	0.199771	4	0.0499427	2.18	0.0685
EH	0.0887463	8	0.0110933	0.48	0.8682
FG	6.8889	4	1.72222	75.20	0.0000

FH	0.1302	8	0.0162749	0.71	0.6824
GH	0.11167	8	0.0139588	0.61	0.7707
Residual	145.493	6353	0.0229014		
Total (corrected)	249.298	6479			

Table B.8: Analysis of Variance for the Hypervolume - Calibration of RIPG, restart and greedy phase fixed.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	5.91195	1	5.91195	297.93	0.0000
B:m	0.307871	1	0.307871	15.51	0.0001
C:m_i	1.75754	1	1.75754	88.57	0.0000
D:T	0.117775	2	0.0588877	2.97	0.0516
E:R	0.0900006	2	0.0450003	2.27	0.1038
F:LocalSearch	0.00806631	2	0.00403316	0.20	0.8161
G:Rep	0.0755584	4	0.0188896	0.95	0.4329
Interactions					
AB	1.22219	1	1.22219	61.59	0.0000
AC	0.485919	1	0.485919	24.49	0.0000
AD	0.554101	2	0.277051	13.96	0.0000
AE	0.237388	2	0.118694	5.98	0.0026
AF	0.749387	2	0.374694	18.88	0.0000
AG	0.0746342	4	0.0186585	0.94	0.4395
BC	0.276526	1	0.276526	13.94	0.0002
BD	0.0696755	2	0.0348377	1.76	0.1731
BE	0.504069	2	0.252035	12.70	0.0000
BF	0.0000573752	2	0.0000286876	0.00	0.9986
BG	0.0469969	4	0.0117492	0.59	0.6684
CD	0.0301489	2	0.0150745	0.76	0.4680
CE	0.033463	2	0.0167315	0.84	0.4305
CF	0.575934	2	0.287967	14.51	0.0000
CG	0.102802	4	0.0257005	1.30	0.2696
DE	0.264315	4	0.0660786	3.33	0.0100
DF	0.240037	4	0.0600093	3.02	0.0169
DG	0.128494	8	0.0160617	0.81	0.5942
EF	0.230393	4	0.0575983	2.90	0.0207
EG	0.332956	8	0.0416196	2.10	0.0330
FG	0.126467	8	0.0158084	0.80	0.6056
Residual	40.9372	2063	0.0198435		
Total (corrected)	57.5954	2159			

Table B.9: Analysis of Variance for the Epsilon indicator - comparison of NSGA-II and RIPG for the set of large instances.

Source	**Sum of Squares**	**Degrees of freedom**	**Mean Square**	**F-Ratio**	**P-Value**
Main effects					
A:n	0.874859	1	0.874859	14.51	0.0001
B:m	0.0950898	1	0.0950898	1.58	0.2091
C:m_i	2.16909	1	2.16909	35.98	0.0000
D:T	0.27736	2	0.13868	2.30	0.1003
E:R	0.176332	2	0.0881659	1.46	0.2317
F:Method	21.818	1	21.818	361.95	0.0000
G:Rep	0.377536	9	0.0419484	0.70	0.7133
Interactions					
AB	0.0644992	1	0.0644992	1.07	0.3009
AC	0.282509	1	0.282509	4.69	0.0304
AD	1.38941	2	0.694703	11.52	0.0000
AE	0.154565	2	0.0772824	1.28	0.2775
AF	2.54314	1	2.54314	42.19	0.0000
AG	0.504928	9	0.0561031	0.93	0.4968
BC	0.299689	1	0.299689	4.97	0.0258
BD	0.260065	2	0.130033	2.16	0.1157
BE	0.283723	2	0.141861	2.35	0.0951
BF	1.99124	1	1.99124	33.03	0.0000
BG	0.183932	9	0.0204368	0.34	0.9622
CD	0.403629	2	0.201814	3.35	0.0352
CE	1.00438	2	0.502188	8.33	0.0002
CF	7.44538	1	7.44538	123.51	0.0000
CG	0.267436	9	0.0297151	0.49	0.8803
DE	1.03797	4	0.259491	4.30	0.0018
DF	0.0667931	2	0.0333965	0.55	0.5747
DG	0.897753	18	0.0498752	0.83	0.6692
EF	0.174688	2	0.087344	1.45	0.2349
EG	0.368888	18	0.0204938	0.34	0.9957
FG	0.478319	9	0.0531466	0.88	0.5408
Residual	340.217	5644	0.0602795		
Total (corrected)	386.109	5759			

APPENDIX C

BEST SOLUTION VALUES

The data in the following tables give the best known solution value for each of the problem instances. The instances are numbered and the parameters used to generate the instances are given. For the small instances, the last column indicates if the solution value has been proven to be the optimum or not. Not that even if no proof has been found that the solution value is the optimal value, it might still be the optimum.

Table C.1: Best found solution values for the small instances with one machine per stage. Optimum indicates if the optimum is guaranteed by the MIP model.

#	n	m	m_i	rel	F	E	S	A	lag	P	rep	Value	Optimum
1	11	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1626	No
2	11	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1537	No
3	11	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1631	No
4	11	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	1789	No
5	11	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	1839	No
6	11	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	1805	No
7	11	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	1656	No
8	11	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	1534	No
9	11	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1586	No
10	11	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	1789	No
11	11	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	1832	No
12	11	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	1742	No
13	11	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	1044	Yes
14	11	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	1281	No
15	11	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	900	Yes
16	11	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1272	Yes
17	11	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1234	Yes
18	11	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1078	Yes
19	11	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	1128	Yes
20	11	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	1127	No
21	11	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	958	Yes
22	11	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1184	Yes
23	11	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	1522	Yes
24	11	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1208	Yes
25	11	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1833	No
26	11	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1895	No
27	11	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1863	No
28	11	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	2238	No
29	11	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	2055	No
30	11	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	2135	No
31	11	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	1640	No
32	11	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	1739	No
33	11	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1759	No
34	11	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	2039	No
35	11	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	1866	No
36	11	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	1988	No
37	11	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	938	No
38	11	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	919	Yes
39	11	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	793	Yes
40	11	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1291	Yes
41	11	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1585	Yes
42	11	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1502	Yes
43	11	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	818	Yes
44	11	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	886	Yes
45	11	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	1119	No
46	11	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1180	Yes
47	11	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	1592	No
48	11	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1407	Yes
49	13	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1713	No
50	13	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	2006	No
51	13	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1728	No
52	13	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	2096	No
53	13	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	2172	No
54	13	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	2156	No
55	13	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	1718	No
56	13	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	1982	No

57	13	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1841	No
58	13	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	1796	No
59	13	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	2355	No
60	13	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	2153	No
61	13	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	1190	No
62	13	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	1327	No
63	13	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	1303	No
64	13	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1431	No
65	13	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1701	No
66	13	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1057	Yes
67	13	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	1006	Yes
68	13	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	1273	Yes
69	13	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	1051	No
70	13	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1274	No
71	13	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	1333	No
72	13	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1447	No
73	13	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1977	No
74	13	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1946	No
75	13	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	2029	No
76	13	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	2471	No
77	13	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	2465	No
78	13	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	2469	No
79	13	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	2112	No
80	13	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	2033	No
81	13	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	2020	No
82	13	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	2786	No
83	13	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	2379	No
84	13	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	2305	No
85	13	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	1271	No
86	13	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	1075	No
87	13	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	1155	No
88	13	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	928	Yes
89	13	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1254	Yes
90	13	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1388	No
91	13	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	1352	No
92	13	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	1055	No
93	13	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	977	Yes
94	13	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1553	No
95	13	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	1577	No
96	13	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1588	No
97	15	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	2160	No
98	15	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1894	No
99	15	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	2206	No
100	15	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	2265	No
101	15	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	2333	No
102	15	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	2404	No
103	15	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	2137	No
104	15	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	2194	No
105	15	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	2024	No
106	15	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	2339	No
107	15	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	2473	No
108	15	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	2362	No
109	15	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	1367	No
110	15	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	1293	No
111	15	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	1446	No
112	15	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1579	No
113	15	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1226	No
114	15	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1894	No
115	15	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	1359	Yes
116	15	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	1530	No
117	15	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	1207	No
118	15	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1398	No
119	15	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	1605	No
120	15	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1497	No

121	15	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	2235	No
122	15	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	2295	No
123	15	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	2504	No
124	15	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	2739	No
125	15	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	2555	No
126	15	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	2752	No
127	15	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	2203	No
128	15	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	2192	No
129	15	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	2469	No
130	15	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	2578	No
131	15	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	2741	No
132	15	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	2590	No
133	15	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	1119	No
134	15	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	1361	No
135	15	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	1621	No
136	15	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1721	No
137	15	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1480	No
138	15	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1512	No
139	15	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	1388	No
140	15	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	1065	Yes
141	15	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	1519	No
142	15	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1661	No
143	15	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	1248	No
144	15	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1426	No
145	5	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	802	Yes
146	5	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	903	Yes
147	5	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	783	Yes
148	5	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	1142	Yes
149	5	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	1297	Yes
150	5	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	1292	Yes
151	5	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	830	Yes
152	5	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	716	Yes
153	5	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	874	Yes
154	5	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	1172	Yes
155	5	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	1024	Yes
156	5	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	1330	Yes
157	5	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	392	Yes
158	5	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	422	Yes
159	5	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	544	Yes
160	5	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	774	Yes
161	5	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	688	Yes
162	5	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	771	Yes
163	5	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	494	Yes
164	5	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	517	Yes
165	5	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	593	Yes
166	5	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	713	Yes
167	5	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	856	Yes
168	5	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1050	Yes
169	5	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1058	Yes
170	5	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1010	Yes
171	5	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1077	Yes
172	5	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	1903	Yes
173	5	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	1324	Yes
174	5	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	2079	Yes
175	5	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	918	Yes
176	5	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	1028	Yes
177	5	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1124	Yes
178	5	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	1330	Yes
179	5	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	1654	Yes
180	5	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	1572	Yes
181	5	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	557	Yes
182	5	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	456	Yes
183	5	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	545	Yes
184	5	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	790	Yes

185	5	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1052	Yes
186	5	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	589	Yes
187	5	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	515	Yes
188	5	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	564	Yes
189	5	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	404	Yes
190	5	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	786	Yes
191	5	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	984	Yes
192	5	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	859	Yes
193	7	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1062	No
194	7	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1200	No
195	7	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1221	No
196	7	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	1447	Yes
197	7	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	1260	Yes
198	7	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	1396	Yes
199	7	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	1011	No
200	7	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	975	No
201	7	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1015	No
202	7	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	1409	Yes
203	7	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	1527	Yes
204	7	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	1585	Yes
205	7	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	562	Yes
206	7	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	765	Yes
207	7	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	769	Yes
208	7	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	928	Yes
209	7	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	897	Yes
210	7	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1059	Yes
211	7	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	678	Yes
212	7	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	769	Yes
213	7	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	574	Yes
214	7	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	941	Yes
215	7	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	1028	Yes
216	7	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	967	Yes
217	7	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1245	No
218	7	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1062	No
219	7	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1174	No
220	7	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	1340	Yes
221	7	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	2024	Yes
222	7	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	1710	Yes
223	7	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	1303	No
224	7	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	1373	No
225	7	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1258	No
226	7	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	1613	Yes
227	7	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	2079	Yes
228	7	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	1691	Yes
229	7	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	537	Yes
230	7	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	535	Yes
231	7	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	606	Yes
232	7	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1111	Yes
233	7	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	808	Yes
234	7	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1248	Yes
235	7	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	750	Yes
236	7	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	831	Yes
237	7	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	595	Yes
238	7	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1136	Yes
239	7	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	701	Yes
240	7	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	1214	Yes
241	9	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1396	No
242	9	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1390	No
243	9	2	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1528	No
244	9	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	1506	No
245	9	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	1694	No
246	9	2	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	1569	Yes
247	9	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	1328	No
248	9	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	1314	No

249	9	2	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1288	No
250	9	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	1407	Yes
251	9	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	1710	Yes
252	9	2	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	1616	No
253	9	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	893	Yes
254	9	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	685	Yes
255	9	2	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	1051	Yes
256	9	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1302	Yes
257	9	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1280	Yes
258	9	2	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	955	Yes
259	9	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	774	Yes
260	9	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	925	Yes
261	9	2	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	994	Yes
262	9	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1068	Yes
263	9	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	932	Yes
264	9	2	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	945	Yes
265	9	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	1	1402	No
266	9	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	2	1484	No
267	9	3	1	1-200	0	100	75-125	50-100	-99-99	0-0	3	1553	No
268	9	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	1	1845	No
269	9	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	2	2077	No
270	9	3	1	1-200	0	100	75-125	50-100	-99-99	1-3	3	1883	No
271	9	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	1	1410	No
272	9	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	2	1422	No
273	9	3	1	1-200	0	50	75-125	50-100	-99-99	0-0	3	1479	No
274	9	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	1	2000	No
275	9	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	2	2188	Yes
276	9	3	1	1-200	0	50	75-125	50-100	-99-99	1-3	3	2090	No
277	9	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	1	1092	Yes
278	9	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	2	1210	Yes
279	9	3	1	1-200	50	100	75-125	50-100	-99-99	0-0	3	903	Yes
280	9	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	1	1146	Yes
281	9	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	2	1054	Yes
282	9	3	1	1-200	50	100	75-125	50-100	-99-99	1-3	3	1146	Yes
283	9	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	1	1075	No
284	9	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	2	771	Yes
285	9	3	1	1-200	50	50	75-125	50-100	-99-99	0-0	3	768	Yes
286	9	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	1	1132	Yes
287	9	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	2	812	Yes
288	9	3	1	1-200	50	50	75-125	50-100	-99-99	1-3	3	911	Yes

Table C.2: Best found solution values for the small instances with three machines per stage. Optimum indicates if the optimum is guaranteed by the MIP model.

#	n	m	m_i	rel	F	E	S	A	lag	P	rep	Value	Optimum
289	11	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	501	No
290	11	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	471	No
291	11	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	489	No
292	11	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	730	No
293	11	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	641	No
294	11	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	712	No
295	11	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	557	No
296	11	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	695	No
297	11	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	924	No
298	11	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	978	No
299	11	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	991	No
300	11	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	977	No
301	11	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	321	No
302	11	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	478	No
303	11	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	388	No
304	11	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	391	Yes
305	11	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	483	No
306	11	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	429	No
307	11	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	465	Yes
308	11	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	496	Yes
309	11	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	405	Yes
310	11	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	408	Yes
311	11	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	575	Yes
312	11	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	762	Yes
313	11	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	598	No
314	11	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	568	No
315	11	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	674	No
316	11	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	948	No
317	11	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	786	No
318	11	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	933	No
319	11	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	822	No
320	11	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	735	No
321	11	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	752	No
322	11	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1372	No
323	11	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1610	No
324	11	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1305	No
325	11	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	320	No
326	11	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	293	Yes
327	11	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	327	No
328	11	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	350	No
329	11	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	393	No
330	11	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	692	No
331	11	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	551	No
332	11	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	542	Yes
333	11	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	383	Yes
334	11	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	869	Yes
335	11	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	722	Yes
336	11	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	745	Yes
337	13	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	610	No
338	13	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	606	No
339	13	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	577	No
340	13	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	657	No
341	13	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	871	No
342	13	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	683	No
343	13	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	683	No
344	13	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	846	No

345	13	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	840	No
346	13	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1061	No
347	13	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	950	No
348	13	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1225	No
349	13	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	497	No
350	13	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	314	Yes
351	13	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	379	No
352	13	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	550	No
353	13	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	531	No
354	13	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	450	No
355	13	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	698	Yes
356	13	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	575	Yes
357	13	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	455	Yes
358	13	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	793	Yes
359	13	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	683	Yes
360	13	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	976	Yes
361	13	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	708	No
362	13	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	642	No
363	13	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	641	No
364	13	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	842	No
365	13	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	903	No
366	13	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	835	No
367	13	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	942	No
368	13	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	941	No
369	13	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	873	No
370	13	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1455	No
371	13	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1184	No
372	13	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1164	No
373	13	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	455	No
374	13	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	415	No
375	13	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	432	No
376	13	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	536	No
377	13	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	461	No
378	13	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	617	No
379	13	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	545	No
380	13	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	465	Yes
381	13	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	377	Yes
382	13	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	1191	No
383	13	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	995	Yes
384	13	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	736	Yes
385	15	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	637	No
386	15	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	688	No
387	15	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	712	No
388	15	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	784	No
389	15	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	765	No
390	15	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	730	No
391	15	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	787	No
392	15	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	787	No
393	15	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	738	No
394	15	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1304	No
395	15	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1170	No
396	15	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	968	No
397	15	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	480	No
398	15	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	448	No
399	15	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	438	No
400	15	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	541	No
401	15	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	507	No
402	15	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	554	No
403	15	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	579	No
404	15	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	579	Yes
405	15	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	552	Yes
406	15	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	786	No
407	15	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	583	Yes
408	15	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	866	No

409	15	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	813	No
410	15	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	770	No
411	15	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	739	No
412	15	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	977	No
413	15	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	835	No
414	15	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	885	No
415	15	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	977	No
416	15	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	1021	No
417	15	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	1010	No
418	15	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1523	No
419	15	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1466	No
420	15	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1343	No
421	15	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	424	No
422	15	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	499	No
423	15	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	434	No
424	15	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	481	No
425	15	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	441	No
426	15	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	610	No
427	15	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	654	No
428	15	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	504	No
429	15	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	480	Yes
430	15	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	1079	No
431	15	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	1023	No
432	15	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	898	No
433	5	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	284	Yes
434	5	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	295	Yes
435	5	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	244	Yes
436	5	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	366	Yes
437	5	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	394	Yes
438	5	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	625	Yes
439	5	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	449	Yes
440	5	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	407	Yes
441	5	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	439	Yes
442	5	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	773	Yes
443	5	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1103	Yes
444	5	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	508	Yes
445	5	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	216	Yes
446	5	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	146	Yes
447	5	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	193	Yes
448	5	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	456	Yes
449	5	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	236	Yes
450	5	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	428	Yes
451	5	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	293	Yes
452	5	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	413	Yes
453	5	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	343	Yes
454	5	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	382	Yes
455	5	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	354	Yes
456	5	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	305	Yes
457	5	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	317	Yes
458	5	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	345	Yes
459	5	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	285	Yes
460	5	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	839	No
461	5	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	732	No
462	5	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	557	Yes
463	5	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	387	Yes
464	5	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	597	Yes
465	5	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	554	Yes
466	5	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1098	Yes
467	5	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1069	Yes
468	5	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1281	Yes
469	5	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	193	Yes
470	5	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	260	Yes
471	5	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	255	Yes
472	5	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	364	Yes

473	5	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	375	Yes
474	5	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	437	Yes
475	5	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	324	Yes
476	5	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	489	Yes
477	5	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	235	Yes
478	5	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	465	Yes
479	5	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	489	Yes
480	5	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	294	Yes
481	7	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	433	No
482	7	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	406	No
483	7	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	353	No
484	7	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	504	Yes
485	7	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	638	No
486	7	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	623	No
487	7	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	591	Yes
488	7	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	521	Yes
489	7	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	510	Yes
490	7	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	808	Yes
491	7	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	889	Yes
492	7	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1295	Yes
493	7	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	164	Yes
494	7	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	184	Yes
495	7	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	266	Yes
496	7	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	352	Yes
497	7	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	308	Yes
498	7	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	443	Yes
499	7	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	311	Yes
500	7	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	361	Yes
501	7	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	342	Yes
502	7	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	602	Yes
503	7	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	538	Yes
504	7	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	526	Yes
505	7	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	447	No
506	7	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	407	No
507	7	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	440	No
508	7	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	853	No
509	7	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	696	No
510	7	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	898	No
511	7	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	635	Yes
512	7	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	509	No
513	7	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	583	No
514	7	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1525	Yes
515	7	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1474	Yes
516	7	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1370	Yes
517	7	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	303	Yes
518	7	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	258	Yes
519	7	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	284	Yes
520	7	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	335	Yes
521	7	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	377	Yes
522	7	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	453	Yes
523	7	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	339	Yes
524	7	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	305	Yes
525	7	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	403	Yes
526	7	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	681	Yes
527	7	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	709	Yes
528	7	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	542	Yes
529	9	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	425	No
530	9	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	429	No
531	9	2	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	471	No
532	9	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	605	No
533	9	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	773	No
534	9	2	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	590	No
535	9	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	526	No
536	9	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	566	No

537	9	2	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	668	No
538	9	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	782	Yes
539	9	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1007	Yes
540	9	2	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1189	Yes
541	9	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	267	Yes
542	9	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	296	Yes
543	9	2	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	399	Yes
544	9	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	418	Yes
545	9	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	624	No
546	9	2	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	422	Yes
547	9	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	326	Yes
548	9	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	438	Yes
549	9	2	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	374	Yes
550	9	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	614	Yes
551	9	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	634	Yes
552	9	2	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	623	Yes
553	9	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	1	476	No
554	9	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	2	508	No
555	9	3	3	1-200	0	100	75-125	50-100	-99-99	0-0	3	599	No
556	9	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	1	746	No
557	9	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	2	728	No
558	9	3	3	1-200	0	100	75-125	50-100	-99-99	1-3	3	757	No
559	9	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	1	615	No
560	9	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	2	761	No
561	9	3	3	1-200	0	50	75-125	50-100	-99-99	0-0	3	658	No
562	9	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	1	1008	No
563	9	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	2	1431	No
564	9	3	3	1-200	0	50	75-125	50-100	-99-99	1-3	3	1342	No
565	9	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	1	213	Yes
566	9	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	2	323	Yes
567	9	3	3	1-200	50	100	75-125	50-100	-99-99	0-0	3	382	No
568	9	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	1	392	Yes
569	9	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	2	483	Yes
570	9	3	3	1-200	50	100	75-125	50-100	-99-99	1-3	3	469	Yes
571	9	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	1	368	Yes
572	9	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	2	427	Yes
573	9	3	3	1-200	50	50	75-125	50-100	-99-99	0-0	3	376	Yes
574	9	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	1	923	Yes
575	9	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	2	837	Yes
576	9	3	3	1-200	50	50	75-125	50-100	-99-99	1-3	3	974	Yes

Table C.3: Best found solution values for the large instances.

#	n	m	m_i	rel	F	E	S	A	lag	P	rep	Value
577	100	4	2	1-200	0	100	75-125	50-100	-99-99	0-0	1	6728
578	100	4	2	1-200	0	100	75-125	50-100	-99-99	0-0	2	6736
579	100	4	2	1-200	0	100	75-125	50-100	-99-99	0-0	3	6882
580	100	4	2	1-200	0	100	75-125	50-100	-99-99	1-5	1	7140
581	100	4	2	1-200	0	100	75-125	50-100	-99-99	1-5	2	7105
582	100	4	2	1-200	0	100	75-125	50-100	-99-99	1-5	3	7274
583	100	4	2	1-200	0	50	75-125	50-100	-99-99	0-0	1	7480
584	100	4	2	1-200	0	50	75-125	50-100	-99-99	0-0	2	7154
585	100	4	2	1-200	0	50	75-125	50-100	-99-99	0-0	3	7358
586	100	4	2	1-200	0	50	75-125	50-100	-99-99	1-5	1	8047
587	100	4	2	1-200	0	50	75-125	50-100	-99-99	1-5	2	7778
588	100	4	2	1-200	0	50	75-125	50-100	-99-99	1-5	3	7569
589	100	4	2	1-200	50	100	75-125	50-100	-99-99	0-0	1	3657
590	100	4	2	1-200	50	100	75-125	50-100	-99-99	0-0	2	3261
591	100	4	2	1-200	50	100	75-125	50-100	-99-99	0-0	3	3489
592	100	4	2	1-200	50	100	75-125	50-100	-99-99	1-5	1	3506
593	100	4	2	1-200	50	100	75-125	50-100	-99-99	1-5	2	3838
594	100	4	2	1-200	50	100	75-125	50-100	-99-99	1-5	3	3714
595	100	4	2	1-200	50	50	75-125	50-100	-99-99	0-0	1	3924
596	100	4	2	1-200	50	50	75-125	50-100	-99-99	0-0	2	3530
597	100	4	2	1-200	50	50	75-125	50-100	-99-99	0-0	3	3924
598	100	4	2	1-200	50	50	75-125	50-100	-99-99	1-5	1	3982
599	100	4	2	1-200	50	50	75-125	50-100	-99-99	1-5	2	3843
600	100	4	2	1-200	50	50	75-125	50-100	-99-99	1-5	3	3704
601	100	4	4	1-200	0	100	75-125	50-100	-99-99	0-0	1	3452
602	100	4	4	1-200	0	100	75-125	50-100	-99-99	0-0	2	3450
603	100	4	4	1-200	0	100	75-125	50-100	-99-99	0-0	3	3394
604	100	4	4	1-200	0	100	75-125	50-100	-99-99	1-5	1	3601
605	100	4	4	1-200	0	100	75-125	50-100	-99-99	1-5	2	3616
606	100	4	4	1-200	0	100	75-125	50-100	-99-99	1-5	3	3664
607	100	4	4	1-200	0	50	75-125	50-100	-99-99	0-0	1	3828
608	100	4	4	1-200	0	50	75-125	50-100	-99-99	0-0	2	3944
609	100	4	4	1-200	0	50	75-125	50-100	-99-99	0-0	3	3871
610	100	4	4	1-200	0	50	75-125	50-100	-99-99	1-5	1	4330
611	100	4	4	1-200	0	50	75-125	50-100	-99-99	1-5	2	4323
612	100	4	4	1-200	0	50	75-125	50-100	-99-99	1-5	3	4450
613	100	4	4	1-200	50	100	75-125	50-100	-99-99	0-0	1	1695
614	100	4	4	1-200	50	100	75-125	50-100	-99-99	0-0	2	1545
615	100	4	4	1-200	50	100	75-125	50-100	-99-99	0-0	3	1599
616	100	4	4	1-200	50	100	75-125	50-100	-99-99	1-5	1	1786
617	100	4	4	1-200	50	100	75-125	50-100	-99-99	1-5	2	1763
618	100	4	4	1-200	50	100	75-125	50-100	-99-99	1-5	3	1918
619	100	4	4	1-200	50	50	75-125	50-100	-99-99	0-0	1	1991
620	100	4	4	1-200	50	50	75-125	50-100	-99-99	0-0	2	1844
621	100	4	4	1-200	50	50	75-125	50-100	-99-99	0-0	3	1917
622	100	4	4	1-200	50	50	75-125	50-100	-99-99	1-5	1	2158
623	100	4	4	1-200	50	50	75-125	50-100	-99-99	1-5	2	2173
624	100	4	4	1-200	50	50	75-125	50-100	-99-99	1-5	3	2122
625	100	8	2	1-200	0	100	75-125	50-100	-99-99	0-0	1	7697
626	100	8	2	1-200	0	100	75-125	50-100	-99-99	0-0	2	7803
627	100	8	2	1-200	0	100	75-125	50-100	-99-99	0-0	3	7785
628	100	8	2	1-200	0	100	75-125	50-100	-99-99	1-5	1	8881
629	100	8	2	1-200	0	100	75-125	50-100	-99-99	1-5	2	8667
630	100	8	2	1-200	0	100	75-125	50-100	-99-99	1-5	3	8869
631	100	8	2	1-200	0	50	75-125	50-100	-99-99	0-0	1	8631
632	100	8	2	1-200	0	50	75-125	50-100	-99-99	0-0	2	8427
633	100	8	2	1-200	0	50	75-125	50-100	-99-99	0-0	3	8544

634	100	8	2	1-200	0	50	75-125	50-100	-99-99	1-5	1	10265
635	100	8	2	1-200	0	50	75-125	50-100	-99-99	1-5	2	10852
636	100	8	2	1-200	0	50	75-125	50-100	-99-99	1-5	3	9904
637	100	8	2	1-200	50	100	75-125	50-100	-99-99	0-0	1	3582
638	100	8	2	1-200	50	100	75-125	50-100	-99-99	0-0	2	3649
639	100	8	2	1-200	50	100	75-125	50-100	-99-99	0-0	3	3662
640	100	8	2	1-200	50	100	75-125	50-100	-99-99	1-5	1	4234
641	100	8	2	1-200	50	100	75-125	50-100	-99-99	1-5	2	4263
642	100	8	2	1-200	50	100	75-125	50-100	-99-99	1-5	3	4234
643	100	8	2	1-200	50	50	75-125	50-100	-99-99	0-0	1	3993
644	100	8	2	1-200	50	50	75-125	50-100	-99-99	0-0	2	4021
645	100	8	2	1-200	50	50	75-125	50-100	-99-99	0-0	3	3860
646	100	8	2	1-200	50	50	75-125	50-100	-99-99	1-5	1	4867
647	100	8	2	1-200	50	50	75-125	50-100	-99-99	1-5	2	4633
648	100	8	2	1-200	50	50	75-125	50-100	-99-99	1-5	3	4982
649	100	8	4	1-200	0	100	75-125	50-100	-99-99	0-0	1	4051
650	100	8	4	1-200	0	100	75-125	50-100	-99-99	0-0	2	4032
651	100	8	4	1-200	0	100	75-125	50-100	-99-99	0-0	3	4089
652	100	8	4	1-200	0	100	75-125	50-100	-99-99	1-5	1	4537
653	100	8	4	1-200	0	100	75-125	50-100	-99-99	1-5	2	4490
654	100	8	4	1-200	0	100	75-125	50-100	-99-99	1-5	3	4692
655	100	8	4	1-200	0	50	75-125	50-100	-99-99	0-0	1	4830
656	100	8	4	1-200	0	50	75-125	50-100	-99-99	0-0	2	4757
657	100	8	4	1-200	0	50	75-125	50-100	-99-99	0-0	3	4767
658	100	8	4	1-200	0	50	75-125	50-100	-99-99	1-5	1	5868
659	100	8	4	1-200	0	50	75-125	50-100	-99-99	1-5	2	6560
660	100	8	4	1-200	0	50	75-125	50-100	-99-99	1-5	3	6159
661	100	8	4	1-200	50	100	75-125	50-100	-99-99	0-0	1	1680
662	100	8	4	1-200	50	100	75-125	50-100	-99-99	0-0	2	1921
663	100	8	4	1-200	50	100	75-125	50-100	-99-99	0-0	3	1885
664	100	8	4	1-200	50	100	75-125	50-100	-99-99	1-5	1	2223
665	100	8	4	1-200	50	100	75-125	50-100	-99-99	1-5	2	2242
666	100	8	4	1-200	50	100	75-125	50-100	-99-99	1-5	3	2419
667	100	8	4	1-200	50	50	75-125	50-100	-99-99	0-0	1	2187
668	100	8	4	1-200	50	50	75-125	50-100	-99-99	0-0	2	2110
669	100	8	4	1-200	50	50	75-125	50-100	-99-99	0-0	3	2094
670	100	8	4	1-200	50	50	75-125	50-100	-99-99	1-5	1	3509
671	100	8	4	1-200	50	50	75-125	50-100	-99-99	1-5	2	2834
672	100	8	4	1-200	50	50	75-125	50-100	-99-99	1-5	3	2952
673	50	4	2	1-200	0	100	75-125	50-100	-99-99	0-0	1	3520
674	50	4	2	1-200	0	100	75-125	50-100	-99-99	0-0	2	3468
675	50	4	2	1-200	0	100	75-125	50-100	-99-99	0-0	3	3505
676	50	4	2	1-200	0	100	75-125	50-100	-99-99	1-5	1	3975
677	50	4	2	1-200	0	100	75-125	50-100	-99-99	1-5	2	3718
678	50	4	2	1-200	0	100	75-125	50-100	-99-99	1-5	3	3861
679	50	4	2	1-200	0	50	75-125	50-100	-99-99	0-0	1	3859
680	50	4	2	1-200	0	50	75-125	50-100	-99-99	0-0	2	3835
681	50	4	2	1-200	0	50	75-125	50-100	-99-99	0-0	3	3862
682	50	4	2	1-200	0	50	75-125	50-100	-99-99	1-5	1	4136
683	50	4	2	1-200	0	50	75-125	50-100	-99-99	1-5	2	4067
684	50	4	2	1-200	0	50	75-125	50-100	-99-99	1-5	3	4496
685	50	4	2	1-200	50	100	75-125	50-100	-99-99	0-0	1	1790
686	50	4	2	1-200	50	100	75-125	50-100	-99-99	0-0	2	1782
687	50	4	2	1-200	50	100	75-125	50-100	-99-99	0-0	3	1935
688	50	4	2	1-200	50	100	75-125	50-100	-99-99	1-5	1	2178
689	50	4	2	1-200	50	100	75-125	50-100	-99-99	1-5	2	1721
690	50	4	2	1-200	50	100	75-125	50-100	-99-99	1-5	3	1646
691	50	4	2	1-200	50	50	75-125	50-100	-99-99	0-0	1	2094
692	50	4	2	1-200	50	50	75-125	50-100	-99-99	0-0	2	1720
693	50	4	2	1-200	50	50	75-125	50-100	-99-99	0-0	3	2085
694	50	4	2	1-200	50	50	75-125	50-100	-99-99	1-5	1	2163
695	50	4	2	1-200	50	50	75-125	50-100	-99-99	1-5	2	2416
696	50	4	2	1-200	50	50	75-125	50-100	-99-99	1-5	3	2258
697	50	4	4	1-200	0	100	75-125	50-100	-99-99	0-0	1	1828

698	50	4	4	1-200	0	100	75-125	50-100	-99-99	0-0	2	1875
699	50	4	4	1-200	0	100	75-125	50-100	-99-99	0-0	3	1891
700	50	4	4	1-200	0	100	75-125	50-100	-99-99	1-5	1	2073
701	50	4	4	1-200	0	100	75-125	50-100	-99-99	1-5	2	2030
702	50	4	4	1-200	0	100	75-125	50-100	-99-99	1-5	3	2022
703	50	4	4	1-200	0	50	75-125	50-100	-99-99	0-0	1	2133
704	50	4	4	1-200	0	50	75-125	50-100	-99-99	0-0	2	2209
705	50	4	4	1-200	0	50	75-125	50-100	-99-99	0-0	3	2015
706	50	4	4	1-200	0	50	75-125	50-100	-99-99	1-5	1	2439
707	50	4	4	1-200	0	50	75-125	50-100	-99-99	1-5	2	2620
708	50	4	4	1-200	0	50	75-125	50-100	-99-99	1-5	3	2458
709	50	4	4	1-200	50	100	75-125	50-100	-99-99	0-0	1	900
710	50	4	4	1-200	50	100	75-125	50-100	-99-99	0-0	2	906
711	50	4	4	1-200	50	100	75-125	50-100	-99-99	0-0	3	838
712	50	4	4	1-200	50	100	75-125	50-100	-99-99	1-5	1	1079
713	50	4	4	1-200	50	100	75-125	50-100	-99-99	1-5	2	1023
714	50	4	4	1-200	50	100	75-125	50-100	-99-99	1-5	3	1047
715	50	4	4	1-200	50	50	75-125	50-100	-99-99	0-0	1	955
716	50	4	4	1-200	50	50	75-125	50-100	-99-99	0-0	2	851
717	50	4	4	1-200	50	50	75-125	50-100	-99-99	0-0	3	1041
718	50	4	4	1-200	50	50	75-125	50-100	-99-99	1-5	1	1266
719	50	4	4	1-200	50	50	75-125	50-100	-99-99	1-5	2	1615
720	50	4	4	1-200	50	50	75-125	50-100	-99-99	1-5	3	1132
721	50	8	2	1-200	0	100	75-125	50-100	-99-99	0-0	1	4349
722	50	8	2	1-200	0	100	75-125	50-100	-99-99	0-0	2	4132
723	50	8	2	1-200	0	100	75-125	50-100	-99-99	0-0	3	4257
724	50	8	2	1-200	0	100	75-125	50-100	-99-99	1-5	1	4988
725	50	8	2	1-200	0	100	75-125	50-100	-99-99	1-5	2	4945
726	50	8	2	1-200	0	100	75-125	50-100	-99-99	1-5	3	5168
727	50	8	2	1-200	0	50	75-125	50-100	-99-99	0-0	1	4824
728	50	8	2	1-200	0	50	75-125	50-100	-99-99	0-0	2	4895
729	50	8	2	1-200	0	50	75-125	50-100	-99-99	0-0	3	4779
730	50	8	2	1-200	0	50	75-125	50-100	-99-99	1-5	1	6305
731	50	8	2	1-200	0	50	75-125	50-100	-99-99	1-5	2	6450
732	50	8	2	1-200	0	50	75-125	50-100	-99-99	1-5	3	6267
733	50	8	2	1-200	50	100	75-125	50-100	-99-99	0-0	1	1796
734	50	8	2	1-200	50	100	75-125	50-100	-99-99	0-0	2	1981
735	50	8	2	1-200	50	100	75-125	50-100	-99-99	0-0	3	2241
736	50	8	2	1-200	50	100	75-125	50-100	-99-99	1-5	1	2657
737	50	8	2	1-200	50	100	75-125	50-100	-99-99	1-5	2	2460
738	50	8	2	1-200	50	100	75-125	50-100	-99-99	1-5	3	2479
739	50	8	2	1-200	50	50	75-125	50-100	-99-99	0-0	1	2135
740	50	8	2	1-200	50	50	75-125	50-100	-99-99	0-0	2	2302
741	50	8	2	1-200	50	50	75-125	50-100	-99-99	0-0	3	2229
742	50	8	2	1-200	50	50	75-125	50-100	-99-99	1-5	1	3283
743	50	8	2	1-200	50	50	75-125	50-100	-99-99	1-5	2	3268
744	50	8	2	1-200	50	50	75-125	50-100	-99-99	1-5	3	3728
745	50	8	4	1-200	0	100	75-125	50-100	-99-99	0-0	1	2298
746	50	8	4	1-200	0	100	75-125	50-100	-99-99	0-0	2	2340
747	50	8	4	1-200	0	100	75-125	50-100	-99-99	0-0	3	2276
748	50	8	4	1-200	0	100	75-125	50-100	-99-99	1-5	1	3216
749	50	8	4	1-200	0	100	75-125	50-100	-99-99	1-5	2	3246
750	50	8	4	1-200	0	100	75-125	50-100	-99-99	1-5	3	2844
751	50	8	4	1-200	0	50	75-125	50-100	-99-99	0-0	1	2784
752	50	8	4	1-200	0	50	75-125	50-100	-99-99	0-0	2	2752
753	50	8	4	1-200	0	50	75-125	50-100	-99-99	0-0	3	2800
754	50	8	4	1-200	0	50	75-125	50-100	-99-99	1-5	1	4596
755	50	8	4	1-200	0	50	75-125	50-100	-99-99	1-5	2	5764
756	50	8	4	1-200	0	50	75-125	50-100	-99-99	1-5	3	5040
757	50	8	4	1-200	50	100	75-125	50-100	-99-99	0-0	1	1130
758	50	8	4	1-200	50	100	75-125	50-100	-99-99	0-0	2	1055
759	50	8	4	1-200	50	100	75-125	50-100	-99-99	0-0	3	983
760	50	8	4	1-200	50	100	75-125	50-100	-99-99	1-5	1	1479
761	50	8	4	1-200	50	100	75-125	50-100	-99-99	1-5	2	1499

762	50	8	4	1-200	50	100	75-125	50-100	-99-99	1-5	3	1290
763	50	8	4	1-200	50	50	75-125	50-100	-99-99	0-0	1	1159
764	50	8	4	1-200	50	50	75-125	50-100	-99-99	0-0	2	1302
765	50	8	4	1-200	50	50	75-125	50-100	-99-99	0-0	3	1064
766	50	8	4	1-200	50	50	75-125	50-100	-99-99	1-5	1	1983
767	50	8	4	1-200	50	50	75-125	50-100	-99-99	1-5	2	1770
768	50	8	4	1-200	50	50	75-125	50-100	-99-99	1-5	3	2054