Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 4627856, 15 pages
https://doi.org/10.1155/2017/4627856

Research Article

Mode-Based versus Activity-Based Search for a Nonredundant Resolution of the Multimode Resource-Constrained Project Scheduling Problem

Daniel Morillo, Federico Barber, and Miguel A. Salido

Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

Correspondence should be addressed to Miguel A. Salido; msalido@dsic.upv.es

Received 7 April 2017; Revised 18 June 2017; Accepted 5 September 2017; Published 11 October 2017

Academic Editor: Jorge Magalhaes-Mendes

This paper addresses an energy-based extension of the Multimode Resource-Constrained Project Scheduling Problem (MRCPSP) called MRCPSP-ENERGY. This extension considers the energy consumption as an additional resource that leads to different execution modes (and durations) of the activities. Consequently, different schedules can be obtained. The objective is to maximize the efficiency of the project, which takes into account the minimization of both makespan and energy consumption. This is a well-known NP-hard problem, such that the application of metaheuristic techniques is necessary to address real-size problems in a reasonable time. This paper shows that the Activity List representation, commonly used in metaheuristics, can lead to obtaining many redundant solutions, that is, solutions that have different representations but are in fact the same. This is a serious disadvantage for a search procedure. We propose a genetic algorithm (GA) for solving the MRCPSP-ENERGY, trying to avoid redundant solutions by focusing the search on the execution modes, by using the Mode List representation. The proposed GA is evaluated on different instances of the PSPLIB-ENERGY library and compared to the results obtained by both exact methods and approximate methods reported in the literature. This library is an extension of the well-known PSPLIB library, which contains MRCPSP-ENERGY test cases.

1. Introduction

The energy consumption in the industry sector is growing by leaps and bounds. Based on the U.S. Energy Information Administration report, in 2016, the industry sector, including manufacturing, consumed approximately a third of the total delivered energy in the world [1]. The environmental implications of the industrial process are gaining more and more importance. Therefore, energy consumption reduction in resource-allocation projects is a critical aspect in the industry sector [2]. For this reason, the interest of researchers is increasingly focused on the development of methodologies for obtaining energy-sustainable solutions. The energy-efficiency oriented scheduling is an actual challenge and a feasible way to save energy in process planning [3].

The Multimode Resource-Constrained Project Scheduling Problem (MRCPSP) is one of the most studied scheduling problems due to the fact that many problems can be modeled as variants of it. One extension of this problem, which incorporates energy consumption in activities, is the so-called MRCPSP-ENERGY that was proposed by Morillo Torres et al. [4]. It includes an additional resource, the energy, that gives rise to different execution modes of activities, and the objective is to maximize the efficiency of the project. This efficiency criterion is managed by combining the *makespan* and the energy consumption criteria into a new combined objective. The objective function of the MRCPSP-ENERGY is more sensitive than the traditional function of the MRCPSP, since solutions that generate the same objective function value for the MRCPSP can generate different values for the MRCPSP-ENERGY. This is because the traditional function of the MRCPSP does not distinguish between solutions with different execution modes if they do not affect the *makespan*; instead, these solutions may lead to different energy consumption in the MRCPSP-ENERGY. This paper addresses the MRCPSP-ENERGY for two main reasons: (1) the wide interest of reaching energy-efficient solutions in scheduling processes and (2) because the impact of redundant solutions can be deeply analyzed in the presence of a highly sensitive objective function.

Solving MRCPSP-ENERGY instances has an NP-hard complexity, and thus exact methods can only find the optimal solution to small-size instances in a reasonable time. Therefore, metaheuristic methods have become more important because they can find near-optimal solutions in a short time. Most metaheuristic methods use a solution representation based on the Activity List and apply movement rules based on order changes in this list of activities to explore the neighborhood of a solution. This paper shows that this commonly used representation (the Activity List) can produce a large number of redundant solutions, which has a negative impact on the search effort. In order to avoid this disadvantage, a new genetic algorithm (GA) is proposed to solve the MRCPSP-ENERGY that includes two optimization phases. The first is an optimization phase over the Mode List, in which the mutation operator plays the major role, since most of the population undergoes a mutation to improve the exploration. The second is an optimization phase over the Activity List, which includes an operator of mutation based on multiple insertions to decrease the number of redundant solutions.

In order to perform an assessment, four versions of the proposed GA are considered: the ML-GA only includes the optimization phase over the Mode List; the AL-GA only includes the optimization phase over the Activity List; the MIX-GA uses the two phases previously mentioned, simultaneously; and finally, the TP-GA considers the two phases separately. In addition, two fitness functions are considered: relative efficiency, in accordance with the MRCPSP-ENERGY proposal, and a weighted normalized function of the objectives. All algorithms are evaluated by using the PSPLIB-ENERGY library instances. The PSPLIB-ENERGY library [4] is an extension of the well-known PSPLIB library proposed by Kolisch and Sprecher [5] in order to provide MRCPSP-ENERGY instances. The PSPLIB-ENERGY includes four sets of problems ($j30$, $j60$, $j90$, and $j120$), which allows evaluating the performance of search algorithms with different sizes of problems. In addition, the results obtained by the proposed GA are compared with the results given by IBM CPLEX CP optimizer. This is a well-known toolbox that uses constraint programming for solving combinatorial optimization problems.

The main contribution is to show that performing the search through the Mode List is a different way to explore the solution space, which can achieve as competitive solutions or even better ones as the search through the Activity List. Moreover, both search procedures can be combined to achieve even better solutions.

The paper is organized as follows. Section 2 presents the problem description. Section 3 describes main methodologies applied for solving the MRCPSP. Section 4 shows some examples of redundant solutions of the Activity List-based representation. In Section 5, the new genetic algorithm is described, and then Section 6 gives the computational experiments and the result analysis. Section 7 summarizes some concluding remarks and Section 8 points out some future work.

2. Problem Description

The MRCPSP-ENERGY [4] is an extension of the well-known Resource-Constrained Project Scheduling Problem (RCPSP). In the MRCPSP-ENERGY, the activities have different execution modes, associated with different energy consumption levels. Activities also require an amount of renewable resources for their execution; these are resources that can be used by any activity and they are renewed every time the activity that uses them has ended, leaving those units of resources again available for being used by another activity. The goal is to maximize the relative efficiency of the project, which minimizes both the energy consumption and the *makespan* ($C_{\max}$). Formally, the problem can be described as a project that consists of a set of n activities $I = \{0, \ldots, i, \ldots, n\}$, a set B of K^{ρ} shared renewable resources $B = \{1, \ldots, b, \ldots, K^{\rho}\}$, and an available amount R_b^{ρ} of every renewable resource. Each activity i has M_i execution modes, where each mode $m \in M_i$ requires a nonpreemptive execution time d_{im}, a total of r_{ib}^{ρ} renewable resources of type b, and an amount of energy e_{im} for its realization. Activities are subject to precedence constraints, which indicate that each activity cannot be started before all its predecessor activities are completed. The different energy consumption for each activity gives rise to different execution modes and, consequently, different execution times.

Figure 1 shows an example of a MRCPSP-ENERGY instance. It has 11 activities; the first and the last activity are dummy activities that represent the beginning and the end of the project. There are 3 renewable resources with a maximum availability of 4 units for each of them. At the top of each node, the execution time and energy consumption are presented for each mode, and at the bottom, its resource usage is presented. The arrows show the precedence relations between activities.

Let the total energy consumption of a project (CETP) be the sum of the energy consumption e_{im} for each activity i in a schedule. $LB0_{\min}$ is the critical path with the shortest execution time, $e_{\min}$ is the sum of energy consumption with lower consumption value, and P_j is the set of immediate predecessor activities of an activity j. Given an upper bound T for the project *makespan*, the latest start time (ls_i) and earliest start time (es_i) can be calculated by applying the forward and backward pass method. The binary decision variable ξ_{imt} takes the value 1 when the activity i is executed in mode m and starts at time t, and 0 otherwise. Therefore, the MRCPSP-ENERGY problem can be formulated as follows:

$$\max \quad \eta\left(C_{\max}, \mathrm{CETP}\right) \tag{1}$$

$$\text{Subject to} \quad \sum_{m=1}^{M_i} \sum_{t=\mathrm{es}_i}^{\mathrm{ls}_i} \xi_{imt} = 1 \quad \forall i \in I \tag{2}$$

$$\sum_{m=1}^{M_j} \sum_{t=\mathrm{es}_j}^{\mathrm{ls}_j} t * \xi_{jmt} \geq \sum_{m=1}^{M_i} \sum_{t=\mathrm{es}_i}^{\mathrm{ls}_i} \left(t + d_{im}\right) * \xi_{imt} \qquad \forall i \in P_j,\ \forall j \in I \tag{3}$$

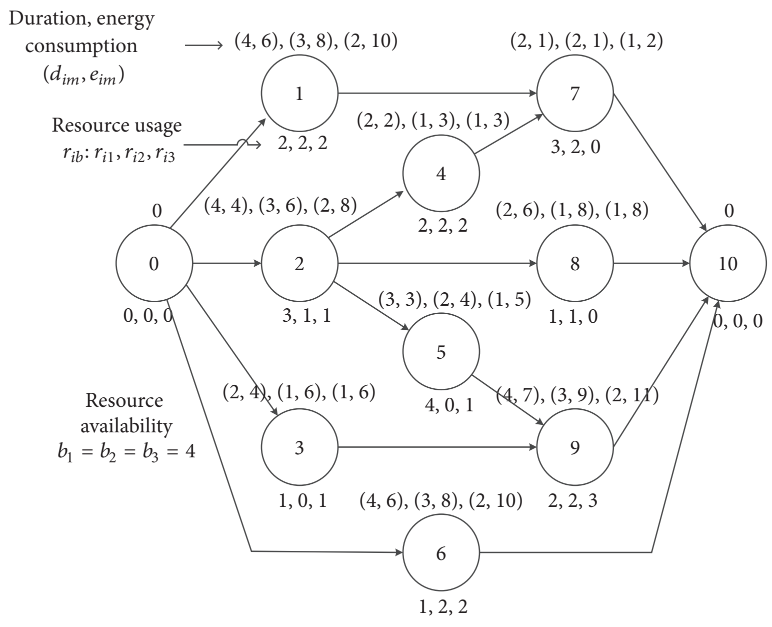

FIGURE 1: A MRCPSP-ENERGY example.

$$\sum_{i=0}^{n} r_{ib}^{\rho} * \sum_{m=1}^{M_i} \sum_{s=\max\{t-d_{im},\mathrm{es}_i\}}^{\min\{t-1,\mathrm{ls}_i\}} \xi_{ims} \leq R_b^{\rho} \quad \forall b \in B,\ t = 0,\ldots,T \tag{4}$$

$$\xi_{imt} \in \{0,1\}, \tag{5}$$

where

$$\eta\left(C_{\max}, \mathrm{CETP}\right) = \frac{1}{\mathrm{CSR}} * \frac{1/C_{\max}}{\mathrm{CETP}}. \tag{6}$$

$$\mathrm{CSR} = \frac{1/LB0_{\min}}{e_{\min}} \tag{7}$$

Expression (2) ensures that each activity starts only once. Expression (3) represents the precedence constraints. Expression (4) ensures that the capacity of resources is not exceeded. Finally, expression (1) shows the optimization function where the objective is to maximize the relative efficiency $\eta(C_{\max}, \mathrm{CETP})$, which has been defined by expressions (6) and (7). This criterion considers both the energy consumption and the *makespan*, simultaneously. Overall, the relative efficiency is interpreted as the efficiency of the project regarding an upper bound of the project performance (CSR) (expression (7)).

The MRCPSP-ENERGY is a strongly NP-hard problem, because this problem is a generalization of the standard RCPSP which is well known to be NP-hard.

3. Literature Review

Most of the related academic literature is dedicated to the MRCPSP with both renewable and nonrenewable resources and less attention has been given to addressing the MRCPSP with only renewable resources. However, the solution methods of both problems share many features. The main difference between them is that when nonrenewable resources are considered, infeasible solutions can be reached, and therefore the solution methods often include a penalty function. Following the academic literature, solution methods can be classified into two groups: exact approaches and metaheuristic approaches.

Elmaghraby [6] was the first to consider different execution modes for activities in the project scheduling problem. Later, Talbot [7] proposed a branch and bound (B&B) algorithm for solving the MRCPSP. This consisted of two stages. In the first one, activities, resources, and modes are organized based on several established priority rules. In the second stage, a heuristic based on a priority rule is used to calculate an upper bound and then a backward B&B is implemented. Patterson et al. [8] proposed another enumeration scheme-based procedure; it is a B&B based on the precedence tree to guide the search in the set of all precedence-feasible sequences of activities. Speranza and Vercellis [9] proposed a depth-first B&B, but Hartmann and Sprecher [10] showed that this algorithm might be unable to find the optimal solution for instances with two or more renewable resources.

Since then, several methods have been proposed based on the B&B with different variants. Sprecher et al. [11] proposed a B&B in which an enumeration scheme, called mode-and-delay alternatives, is used as an extension of the alternative concept of delay proposed by Christofides et al. [12], which is also used by Demeulemeester and Herroelen [13] for the RCPSP. The main differences between this approach and the traditional B&B are that at each level more than one activity can be scheduled and that decisions made at previous levels can be undone at the current level.

More recently, similar but more efficient methods have been developed; for example, Zhu et al. [14] proposed a branch and cut algorithm. Although it was proposed for the multimodal version of the RCPSP with resources partially renewable, it can be used for the traditional MRCPSP. In this approach, the linear relaxation of the linear integer programming model is used to obtain a lower bound of the project duration in each node of the search tree. If the search tree node has a fractional solution, then the algorithm tries to find cuts, that is, valid inequalities that are violated by the fractional solution but are satisfied by the feasible integer solutions represented by that node in the search tree. If no cuts are found in the node, then the branch is carried out, creating new nodes in the tree.

However, in spite of the encouraging results obtained by exact methods, it is important to highlight that exact algorithms in general are unable to optimally solve problems with more than 30 activities, thus leaving the metaheuristic methods as the unique alternative.

Metaheuristic approaches are search algorithms that include escaping strategies from local optima, with the aim of exploring and finding a good approximation to a global optimum. Following the definition stated by Van Peteghem and Vanhoucke [15], the metaheuristic procedures for solving the MRCPSP can be classified into schedule and mode representations, Schedule Generation Schemes (SGS), metaheuristic algorithms, and local search procedures.

Schedule and Mode Representations. A representation stands for how to code a complete solution (i.e., the execution mode and start time of each activity). The coding consists of one schedule representation and one mode representation. Kolisch and Hartmann [16] distinguished five different schedule representations but the *Activity List* representation and the Random Key representation are the most used. The *Activity List* representation consists of a vector of n activities; the order of these elements indicates the priority of an activity to be scheduled. A precedence-feasible list is generally used. The Random Key representation is also a vector with n elements but each of them contains a priority value of the activity in that position. On the other hand, there are mainly two ways of representing modes: a Mode List and a mode vector. The difference between them is that the Mode List represents the execution modes in an ascending order, while the mode vector does it according to the order of the Activity List.

Schedule Generation Schemes. In order to decode a schedule and mode representation in a complete solution, a Schedule Generation Scheme (SGS) is used. Kolisch and Hartmann [16] mainly distinguished two SGS: the serial and the parallel scheme. Both schemes produce feasible solutions. In the serial scheme, solution building is done through a single set of eligible activities that is updated at each step. In the parallel scheme, there are many sets of eligible activities determined by the span in which resources are available. The serial scheme produces a set of schedules that always contain the optimum, while the parallel scheme produces a set of schedules that may exclude it. In spite of this fact, the parallel scheme is also used in the literature because it usually builds more compact solutions than the serial scheme [17].

Metaheuristic Algorithms. There are several metaheuristic methods for solving the MRCPSP. One of the first ones, proposed by Kolisch and Drexl [18], consists of a local search that tries to find a feasible solution and then perform a single neighborhood search on the set of feasible mode assignments. Özdamar [19] proposed two versions of GAs: pure GA and hybrid GA. In the first one, the *Activity List* representation and the serial SGS are used. In the second, a Random Key representation and a parallel SGS are used. The experimental results show that the hybrid GA outperforms all other algorithms tested in that research. Later, Hartmann [20] proposed a GA, which uses a precedence-feasible *Activity List* representation and the serial SGS as the decoding rule. The algorithm includes two local search methods: one was used to deal with the feasibility problem and the other was used to improve the schedules. Józefowska et al. [21] proposed Simulated Annealing (SA) first considering and later disregarding a penalty function, with the latter being the alternative with the best results. They also used the *Activity List* representation and the serial SGS. Bouleimen and Lecocq [22] presented another implementation of SA, where neighbor solutions are generated by using two phases. In the first, a mode-feasible solution is searched and secondly it is improved by random shifts of activities. The first Tabu Search was proposed by Nonobe and Ibaraki [23]. There, the *Activity List* representation is implemented and the neighbor solutions are generated either by a change on the Mode List or by shifting some activities on the Activity List. The authors proposed a new solution building procedure, but the optimal solutions might not be reached by it. In the research made by Alcaraz et al. [24], a GA with equal schedule and mode representation was proposed, but an additional element was included: the forward/backward gene, which indicates the direction of serial SGS to generate a schedule. In the forward direction, the solutions are generated according to the precedence constraints, and in the backward direction they are generated by changing the precedence constraints by successor constraints. The results show that this algorithm outperforms the SA approach proposed by Józefowska et al. [21] but does not exceed the GA proposed by Hartmann [20]. Alternatively, Zhang et al. [25] proposed particle swarm optimization (PSO), which uses two particles as a solution representation. The first particle contains a vector with priority values for each activity (like the Random Key representation), and the second particle contains information about the activity execution modes. Based on their results, the PSO achieves competitive results, although the GA proposed by Hartmann [20] outperforms the PSO. Later, a combinatorial particle swarm optimization (CPSO) was proposed by Jarboui et al. [26]. It generates mode-feasible particles (coded solutions) and then, with a fixed mode assignment, a local search to improve the sequence associated with each particle is performed. The authors compared the algorithm performance with that of the PSO and SA, with CPSO being the algorithm with the best results.

More recently, Li and Zhang [27] proposed an ant colony optimization (ACO) which uses two levels of pheromones: the first level is used to make the selection of an activity and insert it into a sequence and the second level is used

Activity List:	10	7	6	1	4	9	8	5	2	3	0

Activities:	0	1	2	3	4	5	6	7	8	9	10
Mode List:	0	2	2	3	2	1	2	1	3	1	0

Figure 2: Example of an Activity List and a Mode List.

to perform the execution mode assignment. They defined the ACO solution: it is an adaptation of the *Activity List* and the *Mode List* representation to be used in the ACO procedure. The serial SGS was also adapted to generate a complete solution. The results show a high performance but the ACO does not outperform the CPSO. Tseng and Chen [28] proposed a genetic local search with two phases. In the first one, an initial population is generated, and the best solutions are grouped into an elite set. In the second phase, a deep search is carried out in regions defined by the elite set. The solutions in this algorithm are encoded by using the *Activity List* representation. Lova et al. [29] developed an hybrid GA (MM-HGA), which uses, in addition to the *Activity List* representation, two additional genes: forward/backward gene and a serial/parallel gene. The first gene is related to the decoding direction and the second gene is related to the decoding algorithm. The authors also proposed a new normalized fitness function that relates the *makespan* to the units of resources from the excess of nonrenewable resources. Their results show that MM-HGA outperforms the other heuristics. Finally, a bipopulation version of GA was proposed by Van Peteghem and Vanhoucke [30]. The main difference with other genetic algorithms is the use of two different populations: one contains schedules only right-justified and the other contains schedules only left-justified. They adapted the genetic operators to be used with two populations and proposed an extended serial SGS with a local search for improving the mode execution. This algorithm uses the Random Key representation. The obtained results show that this algorithm achieves one of the best solutions in the literature.

To summarize, there are several proposed metaheuristic procedures to solve the MRCPSP and, based on the reported computational experiments, population-based algorithms as well as hybrid metaheuristics are those which achieve the best results. For a wide study on these methodologies, we refer the reader to [31, 32].

Local Search Procedures. The purpose of local search procedures is to iteratively improve the current solution. They often focus on achieving feasibility or improving the *makespan*. One of the most relevant local searches is the forward-backward improvement (FBI) proposed by Tormos and Lova [33] and adapted to be used in the MRCPSP by Lova et al. [29]. The FBI consists of a backward pass and a forward pass. In the first one, the activities are listed from the right to the left and scheduled at the latest feasible time possible. Then, in the forward pass, the activities are listed from the left to the right and scheduled at the earliest feasible time possible. In this way, more compact schedules are often achieved.

4. Redundant Solutions in *Activity List*-Based Representations

It can be deduced from the conclusions of the previous section that the *Activity List*-based representation has a fundamental role in metaheuristic procedures to solve MRCPSP instances. Formally, the *Activity List* representation is an array with n elements that represents activities. This list is feasible with respect to precedence constraints. The position of each activity in the array represents the priority of the activity to be scheduled by using an SGS. On the other hand, the Mode List is also an array with n elements but the elements represent the execution mode of the activities in an ascending order; that is, the element i represents the execution mode of the activity i.

To decode a complete solution, both an Activity List and a Mode List are needed. The decoding can be carried out in several ways: by using the serial or the parallel SGS and the forward or the backward direction. In the backward direction, the predecessor and successor activities are swapped. In Figure 2, an example of an *Activity List* representation and a *Mode List* representation of a feasible solution for the problem represented in Figure 1 is shown.

Metaheuristic procedures use this representation scheme for coding the solutions and then apply movement rules to do the search procedure in the neighborhood. Most procedures do the search through modifications in the Activity List. However, it can be easily deduced that different Activity Lists can obtain equal solutions when they are decoded. For example, Figure 3 shows two different Activity Lists (Activity List 1 and Activity List 2) and one Mode List of feasible solutions for the problem presented in Figure 1. Both Activity Lists were decoded by using the serial SGS in the forward direction. Although both Activity Lists are different (see the underlined activities in Figure 3), the solutions obtained are exactly the same. It is worth noting that the differences between the Activity Lists are not just a simple activity swap. These solutions are named redundant solutions.

Redundancy in solutions can occur at any search procedure when permutations are generated over the Activity List. Estimating the number of redundant solutions of the MRCPSP is not possible without, first, generating, decoding, and checking all permutations of the Activity List. This is because redundant solutions depend on the sequence of scheduled activities and their use of resources at each time point of each solution. Nevertheless, based on the results of Section 6, the number of redundant solutions is much larger than the number of unique solutions. This is one of the first conclusions of our experiments and it is of relevance to the efficiency of the search process. There have been important

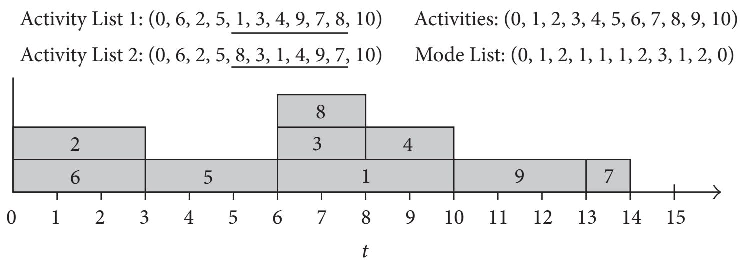

Figure 3: Example of redundant solutions using the *Activity List* representation.

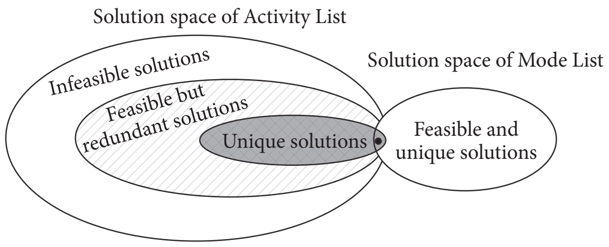
Solution space of Activity List
Solution space of Mode List
Infeasible solutions
Feasible but redundant solutions
Unique solutions
Feasible and unique solutions

Figure 4: The solution space related to the *Activity List* and the *Mode List* representation.

developments in the literature that address these issues. In particular, Debels et al. [34] proposed a scattered search algorithm that includes four mechanisms to avoid the generation of redundant solutions in Random Key representation. Subsequently, Paraskevopoulos et al. [35] proposed a new representation of solutions called event list. In their proposal, first, a real feasible solution is created and then it is coded into sets of activity events, such as sets of activities that can be performed at the same time. Both approaches address the single-mode version of the RCPSP.

On the other hand, only two conditions must be fulfilled so that any change in the Mode List guarantees a different solution (a complete schedule), even when the Activity List remains constant. The first condition is that the duration of the execution modes of the modified activity is different, and the second condition is that there are other activities that can consume the released resources or cannot be executed at that time due to the lack of them. In fact, if an activity i changes its execution mode, the duration of that activity will also change, and this will cause the serial or parallel decoding algorithms to either have or not have available resources at different times of the scheduling. Furthermore, the successor activities will also be available at different time points.

The space of solutions for a representation based only on a list of n activities is bounded by the number of permutations ($n!$). Similarly, the solution space for a representation based on a list of m execution modes is bounded by the number of combinations (m^n). Of course, the permutation space is rather larger than the combination space, as the number of activities increases. Figure 4 shows an illustration about the solution space of the *Activity List* representation and the *Mode List* representation. In this figure, the solution space of the Mode List can be seen as the solutions related to the combination of the execution modes of a list of activities.

Therefore, the problem lies in the fact that the computational effort of a search based only on permutations of the Activity List could be wasted on redundant solutions. Taking into account the fact that most metaheuristics are mainly focused on the permutations of the Activity List, this paper proposes an in-depth study of searching over the Mode List, instead.

5. A *Mode* and *Activity List*-Based Genetic Algorithm

In this section, a new genetic algorithm (GA) to solve MRCPSP-ENERGY that uses two optimization phases, based on the Activity List and the Mode List, is described. First, we describe the codification scheme, the fitness function, the selection criteria, and the replacement process. Then, the genetic operations of crossover and mutation, related to the two proposed optimization phases (based on the Activity List and the Mode List), are detailed.

5.1. Basic Elements of the Genetic Algorithm

Solutions Encoding and Fitness Function. Following the codification proposed by Lova et al. [29], we used an Activity List and a Mode List plus two genes that are used as codification of solutions in MRCPSP-ENERGY. The two additional genes are the SGS gene and the direction gene. The SGS gene represents the method to be used to decode the solution: its value will be 0 when a serial scheme is used or 1 when a parallel scheme is used. The direction gene can be forward or backward (with a value equal to 0 or 1, resp.).

Regarding the fitness function, two different alternatives to take into account the two objectives (minimizing *makespan* and energy consumption) are considered: first maximizing the relative efficiency of the project (expression (1)), as it was described in Section 2, and second minimizing normalized convex combination of these two objectives. Expression (8) shows the definition of the former alternative where $C_{\max}$ is the *makespan* of a project, CETP is the total energy consumption of a project, α represents the priority of minimizing the *makespan* assigned by the decision-maker, and $(1-\alpha)$ represents the priority of minimizing the total energy consumption, also assigned by the decision-maker. The parameters $LB0_{\min}$, T, and $e_{\min}$ were described in Section 2, and $e_{\max}$ is the sum of energy consumption with the highest value. Under these criteria, the objective is to minimize $F(C_{\max}, \text{CETP})$.

$$F\left(C_{\max}, \text{CETP}\right) = \alpha * \frac{C_{\max} - LB0_{\min}}{T - LB0_{\min}} + (1-\alpha) * \frac{\text{CETP} - e_{\min}}{e_{\max} - e_{\min}}. \tag{8}$$

Initial Population. To generate the initial population, the Regret Based Biased Random Sampling (RBBRS) with the latest start time (LST) as a priority rule is applied; this is one of the best sampling methods to create an initial population [16, 36]. Each solution is obtained in the following way: at the beginning, an eligible set (ele) of activities, where all their predecessors have already been scheduled, is computed. Afterwards, the probability of selecting each activity i of ele is calculated through RBBRS, and one of them is chosen (expressions (9) and (10)). Next, a new eligible set of activities is again computed until no task remains. The values of the SGS gene, direction gene, and execution mode are randomly chosen. The p_i value is called *regret value* of the activity i, which compares the value of the activity priority rule $v(i)$ with the worst value of the priority rule for all activities in the set ele. In this case, the worst priority value is the activity with the maximum LST.

$$p_i = \max_{j \in \text{ele}} v(j) - v(i), \tag{9}$$

where $v(i)$ is a priority rule value to be minimized.

$$\text{Probability}(i) = \frac{\left(p_i + \epsilon\right)^{\alpha}}{\sum_{j \in \text{ele}} \left(p_j + \epsilon\right)^{\alpha}}. \tag{10}$$

The parameters of expressions (9) and (10) are $\epsilon \geq 0$ and $\alpha \geq 0$. A value of $\epsilon \neq 0$ allows a positive probability to the activity with the worst value in the priority rule to be selected. The α parameter determines the selection mechanism: a large value makes a deterministic selection and a value of zero makes a completely random selection.

Population Size. There is not a standard method to estimate the best population size in this kind of problems, although it is known that the population should be related to the complexity and the problem size. Some studies state that the population size should decrease with the increasing number of activities [37, 38]. However, most authors carry out computational experiments to estimate these parameters in a GA.

Because an iteration was defined as a complete solution, it was found that the population size should also be related to the number of iterations available. Thus, if the number of iterations is similar to the population size, there will be very few generations that can be produced. Based on our experiments, we apply expressions (11) to compute population and generation number of the proposed GA.

$$\begin{aligned} \text{Population} &= \left(\frac{1}{n} * \text{iterations}\right) + 15 \\ \text{generations} &= \frac{\text{iterations}}{\text{population}}. \end{aligned} \tag{11}$$

Selection. Stochastic sampling with replacement is used, on the basis of the fitness value of the solutions. Therefore, each individual in the population has a probability of being chosen according to its fitness value. When an individual is chosen, a replica of that individual is included in the next selection.

Replacement. Replacement refers to how to create the next population $P3$ from the parent population $P1$ and the offspring population $P2$. First, $P1$ and $P2$ are sorted based on their fitness value. Then, $P3$ is built with 50% of the best $P1$ individuals and 50% of the best $P2$ individuals.

Local Improvement. Two naive local improvements are used in the proposed GA. The first one is the well-known forward-backward improvement (FBI) described in Section 3 and proposed by Tormos and Lova [33]. It is applied over the initial population. The second local improvement consists of reviewing all activities, checking whether they can be executed with less energy consumption (longer execution time) without breaking precedence and resource constraints as long as the *makespan* remains unchanged. It is applied on the best final solution found.

5.2. Optimization Phases. As it was pointed out, the proposed GA divides the search process into two optimization phases, on the Mode List and on the Activity List. The first optimization phase uses crossover and mutation operators over the Mode List in order to expand the search. This plays the most important role in our proposed GA algorithm. The second optimization phase uses crossover and mutation operators over the Activity List, and it is done at the end of the algorithm to intensify the search. Each phase has specific genetic operators (crossover and mutation), which are detailed below.

It is important to note that although each optimization phase focuses on the Activity List or the Mode List, both lists are needed to decode a solution using an SGS. These lists are obtained from the initial population.

5.2.1. Genetic Operators for the Optimization over the Mode List

Crossover. Crossover allows building new solutions from two selected parents. These solutions must share features from both parents. We use a two-point crossover operator only on the Mode List based on the operator proposed by

Alcaraz et al. [24]. Initially, two random integers q_1 and q_2 with $0 < q_1 < q_2 < N$ are generated. The first genes from 0 to q_1 are taken from parent 1, the next genes from q_1 to q_2 are taken from parent 2, and the remaining genes are taken from parent 1 again. The Activity List is randomly inherited only from one of the two parents. The SGS gene and direction gene are inherited when they are of the same value in both parents; otherwise, they are randomly generated.

Mutation. Although mutation does not generally have a main role in genetic operators, it introduces new genetic material, which encourages the exploration. Particularly, a slight perturbation over the Mode List would cause great changes over the solution. The mutation consists of randomly selecting an activity and changing its execution mode. The mutation is applied on each solution (an individual of the population) rather than on each activity, and thus the probability of mutation is independent of the number of project activities. Based on the experimentation performed, the probability value that obtained the best results was 90%.

5.2.2. Genetic Operators for the Optimization over the Activity List

Crossover. A modified two-point crossover is used over the Activity List. Thus, two random integers q_1 and q_2 with $0 < q_1 < q_2 < N$ are generated. Thus, the first genes from 0 to q_1 are taken from parent 1, and the next genes from q_1 to q_2 are taken from parent 2. Here, these genes inherited (q_1 to q_2) might not be those that are in the positions q_1 to q_2 of parent 2, such that they must be searched from the beginning of the Activity List of parent 2, and the first activities that are not repeated in the child are inherited. In the same way, the remaining ones (q_2 to N) are taken from parent 1 again, without repeating genes from both parent 1 and parent 2. The modes of activities are inherited from their corresponding parents. For SGS and direction genes, the gene value is inherited when it is the same in both parents; otherwise, it is randomly generated.

Mutation. A multi-insertion mutation operator based on the research conducted by Boctor [39] is proposed to minimize the probability of producing the same solution. It consists of randomly selecting an activity i and then inserting it in a randomly chosen position. The new position must be higher than that of its predecessors and lower than that of its successors. Usually, the mutation is applied on each activity in a solution with a fixed probability; this implies that the average number of mutated activities is dependent on the total number of activities: the more the activities, the greater the probability of a mutation occurring. In contrast, our experimental results show that the average number of mutated activities should be independent of the total number of project activities. That value was estimated by using the PSPLIB-ENERGY library; an appropriate number of three mutated activities per solution was obtained. Similarly, the probability of mutation is also fixed per solution, independent of the total number of activities, and the estimated value of that probability is 90%.

Finally, the GA's parameters were fixed through experimentation on the PSPLIB-ENERGY library. The number of iterations to change from the optimization phase over the Mode List to the optimization phase over the Activity List was determined as 2/3 of the total number of iterations. This value represents the importance of the optimization phase over the Mode List in relation to the optimization phase over the Activity List.

Figure 5 shows an example of how the offspring is generated by using both the Activity List optimization and the Mode List optimization. In this example, the serial SGS and the forward direction are used. Schedule (a) is parent 1 (genes with overline) and schedule (b) is parent 2 (underlined genes), which were used to generate schedules (c) and (d). Schedule (c) is created by the optimization over the Activity List. To this end, a two-point crossover was used; the first 5 activities are inherited from parent 1; then, the following 3 activities are sought from the beginning of the Activity List of parent 2 and are inherited in only those that are not repeated in the Activity List of the child (activities inherited are 1, 4, and 7); finally, the 3 remaining activities are inherited again from parent 1; these are searched from the beginning of the Activity List of parent 1 and only those that are not repeated are taken. As can be seen in this figure, despite using a two-point crossover from different parents, schedule (c) has the same Activity List as parent 1. The only reason why solutions are different is because the modes of activities inherited from parent 2 have different durations to the same activities of parent 1. Schedule (d) is created by the optimization over the Mode List; it inherits the Activity List from parent 2 and the Mode List is created by a two-point crossover, where the fifth and sixth positions of the Mode List are inherited from parent 2; the remaining activities' modes are taken from parent 1. It can be observed that although the Activity List of parent 2 and the Activity List of schedule (d) are exactly the same, the corresponding solutions are very different, due to the different Mode Lists. In fact, schedule (d) has the best fitness value among the four schedules.

Usually, it is assumed that modifications in a Mode List only affect the execution modes of a preestablished order of activities in the Activity List. But it is not like that. In fact, that preestablished order also changes, as shown in Figure 5.

6. Empirical Assessment and Results

The performed empirical assessment of the proposed GA and the obtained computational results are divided into two groups: the first group is focused on the impact of the redundant solutions of the *Activity List*-based representation and the second group is focused on the performance assessment of the proposed GA by using the PSPLIB-ENERGY library.

This PSPLIB-ENERGY library [4] is a set of MRCPSP-ENERGY instances. It consists of four sets of problems: *j*30, *j*60, *j*90, and *j*120. Each of them has 480 instances, except the last one with 600 instead. Each set has 30, 60, 90, and 120 jobs, respectively. All problems are composed of activities with three execution modes.

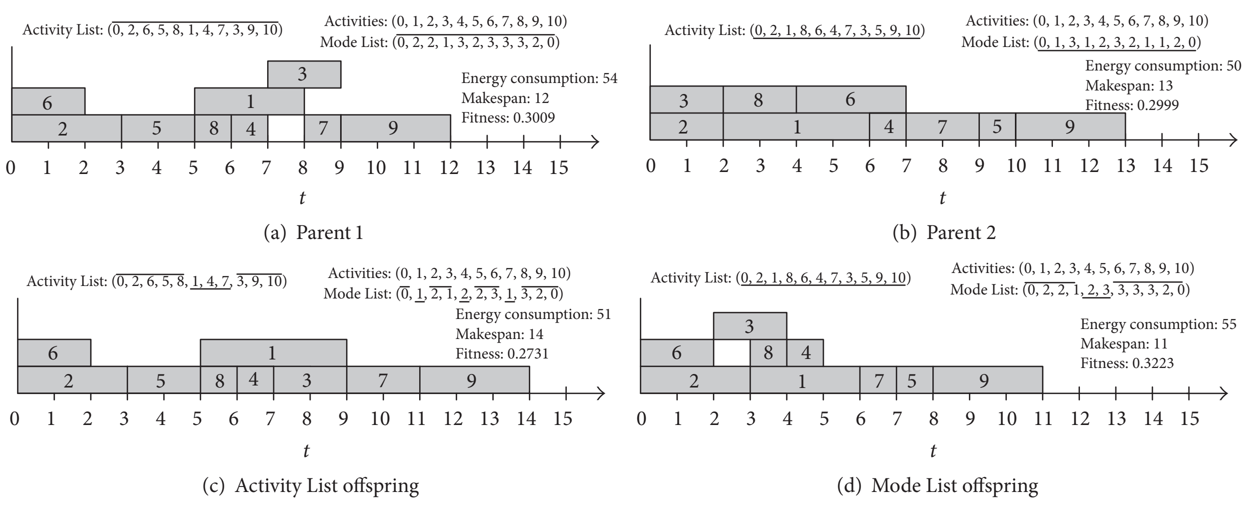
Activity List: (0, 2, 6, 5, 8, 1, 4, 7, 3, 9, 10)
Activities: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Mode List: (0, 2, 2, 1, 3, 2, 3, 3, 3, 2, 0)
Energy consumption: 54
Makespan: 12
Fitness: 0.3009
(a) Parent 1
Activity List: (0, 2, 1, 8, 6, 4, 7, 3, 5, 9, 10)
Activities: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Mode List: (0, 1, 3, 1, 2, 3, 2, 1, 1, 2, 0)
Energy consumption: 50
Makespan: 13
Fitness: 0.2999
(b) Parent 2
Activity List: (0, 2, 6, 5, 8, 1, 4, 7, 3, 9, 10)
Activities: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Mode List: (0, 1, 2, 1, 2, 2, 3, 1, 3, 2, 0)
Energy consumption: 51
Makespan: 14
Fitness: 0.2731
(c) Activity List offspring
Activity List: (0, 2, 1, 8, 6, 4, 7, 3, 5, 9, 10)
Activities: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Mode List: (0, 2, 2, 1, 2, 3, 3, 3, 3, 2, 0)
Energy consumption: 55
Makespan: 11
Fitness: 0.3223
(d) Mode List offspring
t

Figure 5: An example of offspring, by using the optimization phases on the Activity List (offspring (c)) and on the Mode List (offspring (d)).

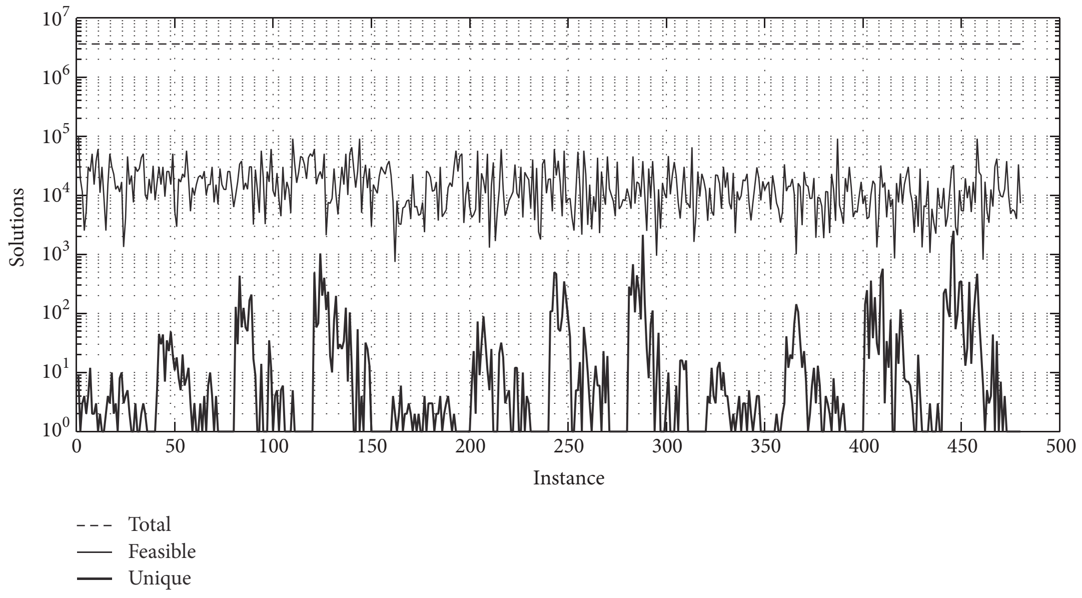

Figure 6: Activity List permutations for the set $j10$.

6.1. Impact of Redundant Solutions in the Activity List-Based Representation. To carry out this evaluation, an exhaustive search was made through all the permutations of the Activity List and all the combinations of the Mode List. Due to the high computational cost of this exhaustive search in problems with many activities, a set of 480 instances was used, each with 10 activities. These instances are based on the set $J30$ of the PSPLIB-ENERGY.

Figures 6 and 7 show (i) the total number of solutions (total), (ii) the number of feasible solutions including redundant ones (feasible), and (iii) the number of unique solutions (unique), of all permutations generated by the Activity List (Figure 6) and the full set of combinations generated by the Mode List (Figure 7).

In Figure 6, the total number of permutations ($10! = 3{,}628{,}800$) is the same for all instances. Due to the huge number of permutations, Figure 6 is shown in a logarithmic scale. As can be seen, the number of redundant solutions is rather higher than that of the unique ones. Indeed, there are some instances with only one unique solution.

In Figure 7, the total number of combinations for each Activity List ($3^{10} = 59{,}049$) is the same for all instances, and all combinations are feasible. There are some instances with a number of unique solutions different from the number of feasible solutions. This is because some activities have equal modes. For instance, in the problem shown in Figure 1, activity 3 has mode 2, which corresponds to a duration of 1 time unit and energy consumption of 6. This activity cannot be executed in a lower time than 1, and thus mode 3 is equal to mode 2.

Therefore, although the solution space of the *Activity List*-based representation is higher than the solution space

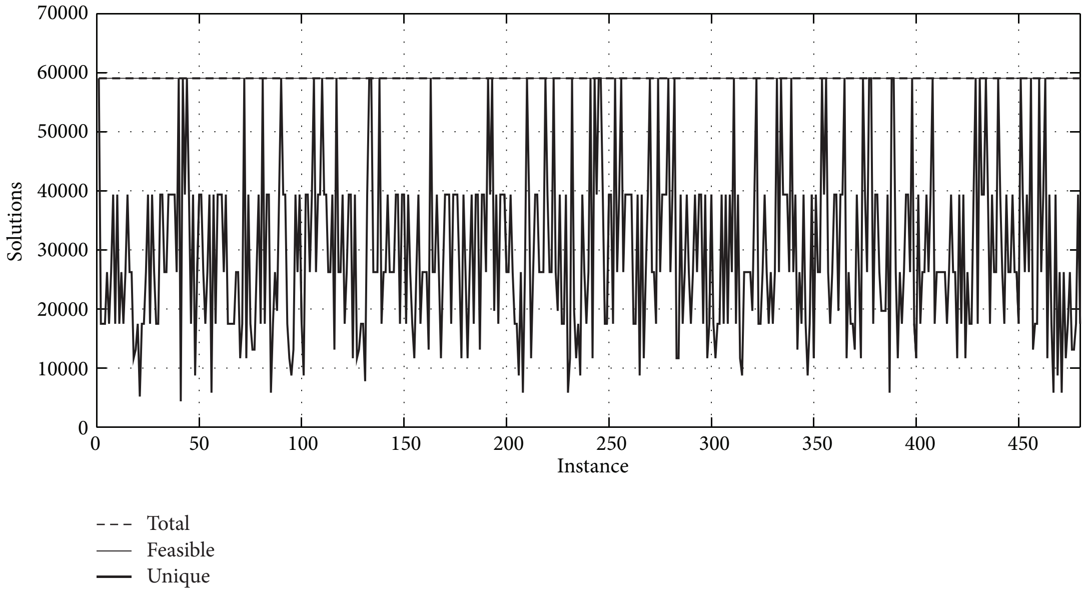

Figure 7: Mode list combinations for the set $j10$.

of the *Mode List*-based representation, most of its solutions are redundant, which makes the search unsuccessful. In contrast, the solution space of the *Mode List* representation is composed of feasible and unique solutions. There are only redundant solutions when there are activities with duplicated execution modes.

6.2. Assessment of the Proposed Genetic Algorithm. In this point, we assess the performance of the proposed GA by using the PSPLIB-ENERGY library. Since this library is based on the well-known PSPLIB library, instance sets were created based on two sets of variable parameters. The first set was used to create the 480 instances of each set: $j30$, $j60$, and $j90$. The second set of parameters was used to create 600 instances for the set $j120$. As is usual in the evaluation of the RCPSP, in this work, the first two instance sets ($j30$, $j60$) will be taken as a reference for small instances, and the set $j120$ will be taken as a reference of large instances. To the best of our knowledge, Morillo Torres et al. [4] are the only authors that have reported results for the instances in this library. Therefore, the proposed GA was compared with that algorithm (called in this paper Basic-GA). Basic-GA uses a coding based on the list of activities and modes, a crossover operator at a single point, and a mutation based on the insertion of Boctor. When an activity is selected, its position is changed in the list of activities, as well as its execution mode. The main difference between this algorithm and the proposed GA is that the Basic-GA takes the standard elements of a genetic algorithm for the MRCPSP, focusing the genetic operators on modifying the Activity List, leaving the mutation as the main tool to change the execution modes.

On the other hand, one of the most used stop criteria in the literature for RCPSP and MRCPSP is the maximum number of iterations. Kolisch and Hartmann [32] defined an iteration as a complete scheduling and showed the reason to use this termination condition as a comparing rule with other solving methods. In this paper, the same stop criteria are used.

The empirical assessment was done considering four variants of the proposed GA. In this way, objective comparisons can be made to identify the elements that contribute to the search. The variants of the proposed algorithm are described below.

(i) The genetic algorithm based on the optimization phase over the Activity List (AL-GA): this algorithm is focused on performing the search by exploring the permutations of the Activity List by using the operators described in Section 5.2.2. The mode of the activities of the population remains constant after the creation of the initial population. The offspring inherits the activities with the execution mode of the corresponding parent.

(ii) The genetic algorithm based on the optimization phase over the Mode List (ML-GA): the search performed by this algorithm is focused on modifying the Mode List of the population (over all combinations) by using the operators described in Section 5.2.1. Activity lists of the population remain constant after the creation of the initial population.

(iii) The genetic algorithm based on the simultaneous mixing of the optimization phases (MIX-GA): this algorithm mixes the two optimization approaches: the optimization phase over the Mode List and over the Activity List. In this way, the operators of both phases are used simultaneously. In this case, neither activity nor Mode Lists remain constant.

(iv) The genetic algorithm based on the two separate optimization phases (TP-GA): this algorithm incorporates the two optimization phases separately, but

Table 1: $\overline{\eta}$ obtained by using the proposed genetic algorithms for solving MRCPSP-ENERGY library.

j#	Algorithm	Iterations/$\overline{\eta}$		
		1000	5000	50,000
$j30$	*TP-GA*	*0.6397*	*0.6517*	*0.6564*
	ML-GA	0.6381	0.6506	0.6559
	AL-GA	0.6345	0.6489	0.6551
	MIX-GA	0.6290	0.6398	0.6469
	Basic-GA	0.5966	0.6091	0.6293
$j60$	*TP-GA*	0.6565	*0.6803*	*0.6919*
	ML-GA	*0.6576*	0.6794	0.6907
	AL-GA	0.6418	0.6700	0.6890
	MIX-GA	0.6498	0.6672	0.6773
	Basic-GA	0.6029	0.6182	0.6424
$j120$	*TP-GA*	0.5192	0.5382	*0.5590*
	ML-GA	*0.5211*	*0.5397*	0.5589
	AL-GA	0.5092	0.5237	0.5523
	MIX-GA	0.5158	0.5310	0.5477
	Basic-GA	0.4760	0.4875	0.5032

with greater emphasis on the search for execution modes by using 2/3 of the total number of available iterations in the optimization phase over the Mode List and in the end 1/3 of the total number of available iterations in the optimization phase over the Activity List.

Table 1 shows the average relative efficiency of different variants of the proposed GA and the Basic-GA to solve three sets of instances of the PSPLIB-ENERGY for 1000, 5000, and 50,000 iterations. This table is sorted by the 50,000-iteration column.

As it can be seen from Table 1, the ML-GA outperforms the AL-GA. Note that the objective is to maximize the relative efficiency of the project. It is interesting to remark that, in the ML-GA, the Activity Lists of all instances have remained constant once the initial population is created, and then only the modes have changed. On the other hand, in AL-GA, the Mode Lists of all instances have remained constant, and only the Activity Lists were modified. This indicates that searching through Mode Lists can achieve high quality solutions, as compared to searching through Activity Lists.

Due to redundant solutions with different Activity Lists, an optimal solution can be represented by a Mode List and several Activity Lists. Thus, by fixing a list and searching over the other, optimal solutions can be excluded. However, the proposed TP-GA searches over the Mode List and, in the final phase, over the Activity List. This combined search procedure achieves better results. In fact, Table 1 shows that the TP-GA outperforms all other algorithms in respect of maximizing the relative efficiency of the project.

On the other hand, we also compared the proposed algorithms based on the minimization of the normalized value of the convex combination for both objectives: the *makespan* and the total energy consumption (expression (8)). Without losing generality, three different values of α are selected: 0.25, 0.5, and 0.75. Table 2 shows a summary of the results. The first row shows the alpha value and the second row shows the iterations number. The first, fifth, and ninth columns show the algorithm version used. The remaining columns show the value of the normalized objective function ($F(C_{\max}, \text{CETP})$).

Based on the results of Table 2, it can be seen that the TP-GA is still the algorithm that obtains better results on average; it outperforms the other algorithms in 7 out of 9 experimental sets. In addition, in the sets $j60$ with $\alpha = 0.25$ and $j60$ with $\alpha = 0.5$, the differences regarding the first position are 0.09% and 0.27%, respectively. In addition, it can be observed that the search by means of the optimization over the Mode List (ML-GA) can reach solutions similar to those reached by the search using the optimization over the Activity List (AL-GA) and outperforms it in some cases. On the other hand, simultaneously mixing the two phases of search does not seem appropriate because it obtains worse results on average; this could be because the mixture can generate too much diversity in the population, thus slowing the convergence.

Since there are no reported optimal solutions for the test cases provided in this library, it is useful to obtain results by an exact approach. Thus, IBM ILOG CPLEX CP optimizer 12.6.2 was used to solve some instances of the MRCPSP-ENERGY library. This toolbox uses constraint programming because it has shown significant results in combinatorial problems such as scheduling problems.

Since the MRCPSP-ENERGY is an NP-hard problem, it is for now impossible to determine optimal solutions in a reasonable time for real-size instances. Therefore, we set a deadline of 30 minutes for solving each problem. We consider this as a reasonable time to address scheduling problems with an exact approach. When the deadline is reached, the best solution is returned. In addition, considering that the optimal solutions for unimodal RCPSP instances with 60 activities are unknown, trying to find the optimal solutions for the MRCPSP-ENERGY with 60 activities is actually more difficult, and we only run the set $j30$.

The exact approach has been able to obtain 70% of the optimal solutions for set $j30$, with a maximum time of 30 minutes for each problem. Table 3 shows a summary of such results. The first column shows the set of instances used. The second column presents the method used to resolve the instances. The third column shows the average value of the objective function (relative efficiency of the project). The fourth column shows the number of optimal solutions found and the total number of instances. The fifth column shows the average execution time for all instances. Finally, the last column shows the limit of execution time.

The exact approach (CPLEX) reached an average value of 65.97% of relative efficiency of the project with an average computational time of 601.15 seconds. The proposed GA reaches an average value of 65.75% of relative efficiency with an average computational time of 2.51 seconds. Thus, the difference between the objective function average provided by CPLEX and that of the proposed GA is 0.3335%, taking into account the fact that the proposed GA requires 99.58% less time than CPLEX to achieve these results. As it can be

Table 2: Normalized value obtained by using the proposed genetic algorithms for solving MRCPSP-ENERGY library.

Alg.	$\alpha = 0.25$			Alg.	$\alpha = 0.5$			Alg.	$\alpha = 0.75$		
	Iterations/F				Iterations/F				Iterations/F		
	1000	5000	50,000		1000	5000	50,000		1000	5000	50,000
Set *j*30											
TP-GA	*0.0728*	*0.0531*	*0.0525*	*TP-GA*	*0.1185*	*0.1044*	*0.1029*	*TP-GA*	*0.1329*	*0.1247*	*0.1225*
AL-GA	0.0907	0.0550	0.0525	AL-GA	0.1282	0.1059	0.1031	ML-GA	0.1335	0.1254	0.1227
ML-GA	0.0781	0.0542	0.0529	ML-GA	0.1207	0.1055	0.1034	AL-GA	0.1364	0.1260	0.1230
MIX-GA	0.1011	0.0693	0.0571	MIX-GA	0.1358	0.1170	0.1088	MIX-GA	0.1417	0.1329	0.1281
Set *j*60											
AL-GA	0.1390	0.0605	*0.0331*	*AL-GA*	0.1316	0.0828	*0.0651*	*TP-GA*	0.1034	*0.0866*	*0.0828*
TP-GA	0.0965	*0.0356*	0.0331	TP-GA	0.1064	*0.0685*	0.0653	AL-GA	0.1137	0.0928	0.0832
ML-GA	*0.0937*	0.0363	0.0339	ML-GA	*0.1037*	0.0693	0.0664	ML-GA	*0.1023*	0.0870	0.0833
MIX-GA	0.1061	0.0538	0.0367	MIX-GA	0.1142	0.0819	0.0705	MIX-GA	0.1095	0.0949	0.0886
Set *j*120											
TP-GA	0.1599	0.0551	*0.0304*	*TP-GA*	0.1442	0.078	*0.0604*	*TP-GA*	0.1239	0.0981	*0.0866*
ML-GA	0.1467	*0.0467*	0.0322	ML-GA	*0.1361*	*0.0743*	0.0632	ML-GA	*0.1207*	*0.0960*	0.0878
MIX-GA	*0.1443*	0.0641	0.0343	AL-GA	0.1718	0.1267	0.0641	AL-GA	0.1358	0.1177	0.0894
AL-GA	0.2059	0.1322	0.0355	MIX-GA	0.1380	0.0885	0.0669	MIX-GA	0.1261	0.1055	0.0928

Table 3: Results obtained by using IBM ILOG CPLEX CP optimizer for solving set $j30$ of the MRCPSP-ENERGY library.

Method	$\overline{\eta}$	# optimal	Average time (s)	Deadline (s)
Set $j30$				
CPLEX	0.6597	340 (480)	601.15	1800
TP-GA	*0.6575*	*218 (480)*	*2.51*	—

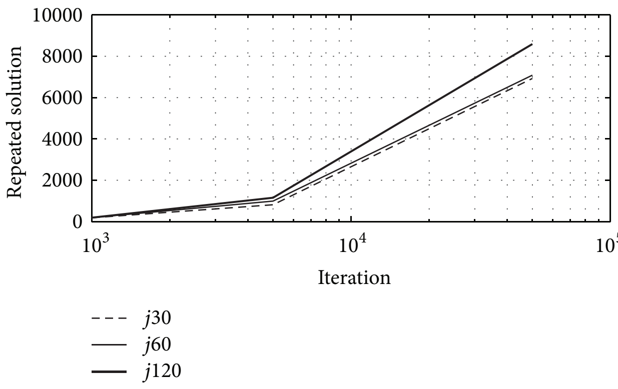

Figure 8: The average number of redundant offspring.

seen, the GA achieves a beneficial trade-off between time and accuracy.

Finally, in order to experimentally assess the impact of redundant solutions, we calculate the average number of redundant offspring generated by the genetic operators in the optimization phase of the Activity List, where both parents are different from each other. The offspring from equal parents were not counted. The estimates were worked out for sets $j30$, $j60$, and $j120$, for 1000, 5000, and 50,000 iterations, respectively. The results are shown in Figure 8. As expected, the number of redundant solutions is related to the number of iterations and, to a lesser degree, to the activities number. From the results, the average number of redundant solutions that do not contribute with new information to the GA is at least 15% of the total number of solutions.

7. Conclusions

In this paper, the Multimode Resource-Constrained Project Scheduling Problem, particularly the MRCPSP-ENERGY, has been addressed with the aim of showing the importance of the solution representation in the metaheuristic techniques. Particularly, we analyze the redundant solutions issue of one of the most important solution representations: the *Activity List*.

A new genetic algorithm with two phases of optimization (TP-GA) has been proposed, whose main contribution is to show that there is an undervalued search alternative in multimode resource scheduling problems. Here, the search is focused on Mode Lists instead of doing it on Activity Lists. This proposal is based on the fact that although genetic operators can modify the Activity List, the resulting solution (complete scheduling) can be exactly the same, even if they come from different Activity Lists. Therefore, the computational effort of a search when it is based only on changes of the Activity List might be wasted on redundant solutions. While a change on the Activity List may lead to the same scheduling, a change on the Mode List always guarantees a different scheduling as long as activities have a different execution time and there exist activities that can take advantage of the availability or lack of resources released by a change in the execution mode.

The results show that when the objective function is the relative efficiency, the proposed genetic algorithm achieves the best results in comparison with the other algorithms for solving the MRCPSP-ENERGY in all problem sets. With regard to minimizing the weighted normalized value of both *makespan* and total energy consumption, the proposed GA also obtained one of the best results; it outperforms the other algorithms in seven out of nine experimental sets and in the other cases achieves the second place. The results also show that the search when modifying only Mode Lists can achieve the best results compared to the search when modifying only Activity Lists. In addition, a comparison of the results between the proposed genetic algorithm and an exact approach was performed. For this, we used IBM ILOG CPLEX CP optimizer. Although the exact approach produces slightly better results, the proposed algorithm uses 99.58% less time. We can conclude that the proposed genetic algorithm achieves highly efficient results.

Based on these results and the fact that the literature about scheduling problems with different execution modes is focused on the optimization of the Activity List, we think there exists a valuable field of research focused not only on the optimization of the Activity List, but also on the optimization of the Mode List. The latter optimization process becomes the most successful search procedure when compared with the former.

8. Future Work

The paper has focused on the analysis of solution representations in the MRCPSP-ENERGY which seeks to minimize both the *makespan* and the energy consumption. The next step would be to consider energy as a nonrenewable resource. In addition, energy consumption can be generalized to the use of other nonrenewable resources, such as budget and fossil fuels. The analysis could be extended to the traditional MRSPCP, which only considers the *makespan*. Here, the impact of the search focused on the *Mode List* representation can be analyzed. Likewise, an analysis on how the existence of redundant solutions affects other solution representations could be performed. Finally, a new solution representation that can take advantage of the search through the Mode List could be proposed, being independent of an Activity List to decode a solution, or a combined representation that avoids redundant solutions.

Disclosure

An earlier version of this work was presented at COPLAS'17 (informal proceedings).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper has been partially supported by the Spanish Research Projects TIN2013-46511-C2-1-P and TIN2016-80856-R.

References

[1] Energy Information Administration (EIA), "Monthly Energy Review - October 2016," Tech. Rep., Office of Energy Statistics, 2016.

[2] Z. Zhang, R. Tang, T. Peng, L. Tao, and S. Jia, "A method for minimizing the energy consumption of machining system: integration of process planning and scheduling," *Journal of Cleaner Production*, vol. 137, pp. 1647–1662, 2016.

[3] G. Mouzon, M. B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *International Journal of Production Research*, vol. 45, no. 18-19, pp. 4247–4271, 2007.

[4] D. Morillo Torres, F. Barber, and M. A. Salido, "A new model and metaheuristic approach for the energy-based resource-constrained scheduling problem," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, pp. 1–13, 2017.

[5] R. Kolisch and A. Sprecher, "PSPLIB—a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

[6] S. E. Elmaghraby, *Activity Networks: Project Planning and Control by Network Models*, John Wiley & Sons Inc, 1977.

[7] F. B. Talbot, "Resource-constrained project scheduling with time-resource tradeoffs: the nompreemptive case," *Management Science*, vol. 28, no. 10, pp. 1197–1210, 1982.

[8] J. H. Patterson, R. Slowinski, F. B. Talbot, and J. Weglarz, "An algorithm for a general class of precedence and resource constrained scheduling problems," *Advances in Project Scheduling*, pp. 3–28, 1989.

[9] M. G. Speranza and C. Vercellis, "Hierarchical models for multi-project planning and scheduling," *European Journal of Operational Research*, vol. 64, no. 2, pp. 312–325, 1993.

[10] S. Hartmann and A. Sprecher, "A note on "hierarchical models for multi-project planning and scheduling"," *European Journal of Operational Research*, vol. 94, no. 2, pp. 377–383, 1996.

[11] A. Sprecher, S. Hartmann, and A. Drexl, "An exact algorithm for project scheduling with multiple modes," *OR Spektrum. Quantitative Approaches in Management*, vol. 19, no. 3, pp. 195–203, 1997.

[12] N. Christofides, R. 'Alvarez-Valdes, and J. M. Tamarit, "Project scheduling with resource constraints: a branch and bound approach," *European Journal of Operational Research*, vol. 29, no. 3, pp. 262–273, 1987.

[13] E. Demeulemeester and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem," *Management Science*, vol. 38, no. 12, pp. 1803–1818, 1992.

[14] G. Zhu, J. F. Bard, and G. Yu, "A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem," *INFORMS Journal on Computing*, vol. 18, no. 3, pp. 377–390, 2006.

[15] V. Van Peteghem and M. Vanhoucke, "An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances," *European Journal of Operational Research*, vol. 235, no. 1, pp. 62–72, 2014.

[16] R. Kolisch and S. Hartmann, "Heuristic algorithms for the resource-constrained project scheduling problem: classification and computational analysis," in *Project Scheduling*, vol. 14 of *International Series in Operations Research & Management Science*, pp. 147–178, Springer, 1999.

[17] J.-L. Kim, "Proposed methodology for comparing schedule generation schemes in construction resource scheduling," in *Proceedings of the 2009 Winter Simulation Conference, (WSC '09)*, pp. 2745–2750, December 2009.

[18] R. Kolisch and A. Drexl, "Local for multi-mode resource-constrained project," *IIE Transactions (Institute of Industrial Engineers)*, vol. 29, no. 11, pp. 987–999, 1997.

[19] L. Özdamar, "A genetic algorithm approach to a general category project scheduling problem," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 29, no. 1, pp. 44–59, 1999.

[20] S. Hartmann, "Project scheduling with multiple modes: a genetic algorithm," *Annals of Operations Research*, vol. 102, pp. 111–135, 2001.

[21] J. Józefowska, M. Mika, R. Rózycki, G. Waligóra, and J. Weglarz, "Simulated annealing for multi-mode resource-constrained project scheduling," *Annals of Operations Research*, vol. 102, pp. 137–155, 2001.

[22] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European Journal of Operational Research*, vol. 149, no. 2, pp. 268–281, 2003.

[23] K. Nonobe and T. Ibaraki, "Formulation and tabu search algorithm for the resource constrained project scheduling problem," in *Essays and surveys in metaheuristics (Angra dos Reis, 1999)*, vol. 15 of *Oper. Res./Comput. Sci. Interfaces Ser.*, pp. 557–588, Kluwer Acad. Publ., Boston, MA, USA, 2002.

[24] J. Alcaraz, C. Maroto, and R. Ruiz, "Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms," *Journal of the Operational Research Society*, vol. 54, no. 6, pp. 614–626, 2003.

[25] H. Zhang, C. M. Tam, and H. Li, "Multimode project scheduling based on particle swarm optimization," *Computer-Aided Civil and Infrastructure Engineering*, vol. 21, no. 2, pp. 93–103, 2006.

[26] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299–308, 2008.

[27] H. Li and H. Zhang, "Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints," *Automation in Construction*, vol. 35, pp. 431–438, 2013.

[28] L. Y. Tseng and S. C. Chen, "Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 848–857, 2009.

[29] A. Lova, P. Tormos, M. Cervantes, and F. Barber, "An efficient hybrid genetic algorithm for scheduling projects with resource

constraints and multiple execution modes," *International Journal of Production Economics*, vol. 117, no. 2, pp. 302–316, 2009.

[30] V. Van Peteghem and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 201, no. 2, pp. 409–418, 2010.

[31] J. Weglarz, J. Józefowska, M. Mika, and G. Waligóra, "Project scheduling with finite or infinite number of activity processing modes—a survey," *European Journal of Operational Research*, vol. 208, no. 3, pp. 177–205, 2011.

[32] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update," *European Journal of Operational Research*, vol. 174, no. 1, pp. 23–37, 2006.

[33] P. Tormos and A. Lova, "A competitive heuristic solution technique for resource-constrained project scheduling," *Annals of Operations Research*, vol. 102, pp. 65–81, 2001.

[34] D. Debels, B. De Reyck, R. Leus, and M. Vanhoucke, "A hybrid scatter search/electromagnetism meta-heuristic for project scheduling," *European Journal of Operational Research*, vol. 169, no. 2, pp. 638–653, 2006.

[35] D. C. Paraskevopoulos, C. D. Tarantilis, and G. Ioannou, "Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm," *Expert Systems with Applications*, vol. 39, no. 4, pp. 3983–3994, 2012.

[36] A. Drexl, "Scheduling of project networks by job assignment," *Management Science*, vol. 37, no. 12, pp. 1590–1602, 1991.

[37] D. Debels and M. Vanhoucke, "A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem," *Operations Research*, vol. 55, no. 3, pp. 457–469, 2007.

[38] M. Cervantes, A. Lova, P. Tormos, and F. Barber, "A dynamic population steady-state genetic algorithm for the resource-constrained project scheduling problem," in *New Frontiers in Applied Artificial Telligence*, L. Thanh Nguyen, A. Borzemski, Grzech., and., and M. Ali, Eds., vol. volume, 502, pp. 611–620, Springer, Berlin, Germany, 2008.

[39] F. F. Boctor, "Resource-constrained project scheduling by simulated annealing," *International Journal of Production Research*, vol. 34, no. 8, pp. 2335–2351, 1996.