Eficiencia Energética en la Programación de Tareas con Recursos Restringidos

Daniel Morillo Torres

Tesis Doctoral
Septiembre - 2017

Directores:
Dr. Federico Barber Sanchís
Dr. Miguel A. Salido Gregorio

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Eficiencia energética en la programación de tareas con recursos restringidos

Tesis doctoral

Abril 2017

Autor: Daniel Morillo Torres

Director: Dr. Federico Barber
Dr. Miguel A. Salido

Dedicado a
mi amada compañera

Agradecimientos

El desarrollo del presente trabajo ha sido posible gracias a la aportación de varias personas a las cuales me gustaría agradecer.

Inicialmente a mis dos directores de tesis, el profesor Federico Barber y el profesor Miguel A. Salido, por su constate apoyo, sus conocimientos, la paciencia y sus orientaciones, las cuales me han aportado enormemente en mi naciente carrera como investigador.

A mi madre por su incondicional apoyo, su tenacidad y su amor por la ciencia, ella ha sido mi inspiración para seguir mi formación profesional. De igual manera, a mi padre por su amor, su confianza y sus esfuerzos para educarme, de él he aprendido el sentido de la justicia y la perseverancia.

Por último, pero no menos importante, a una hermosa mujer que ha teñido mi vida de felicidad y bienestar, un maravilloso ser que ha decidido compartir sus intrépidas aventuras conmigo, de quien admiro su inteligencia y su capacidad de amar.

Resumen

En la investigación operativa, el conjunto de problemas de secuenciación de actividades es considerado como uno de los más relevantes debido a su gran aplicabilidad y complejidad. Dentro de la amplia variedad de problemas en este conjunto, destaca el problema de programación de tareas con recursos restringidos (RCPSP por su sigla en inglés), pues es considerado como el problema base más importante en esta área y ha sido objeto de estudio de numerosas investigaciones. Básicamente, consiste de un proyecto subdividido en un conjunto de actividades que se encuentran relacionadas mediante restricciones de precedencia y requieren, para ser ejecutadas, una cantidad de cada tipo de recurso cuya disponibilidad máxima se encuentra limitada. El objetivo es asignar los recursos a las actividades de la manera más eficiente posible para optimizar una medida de desempeño, por ejemplo, la duración total del proyecto. Igualmente importante es la versión multi-modal del RCPSP, llamada MRCPSP, en la que para cada actividad existen múltiples modos de ejecución que involucran una combinación diferente de recursos limitados, dando origen a un tiempo de ejecución distinto. En la literatura se han abordado ampliamente estos dos problemas tanto con métodos exactos como de aproximación, siendo estos últimos los más exitosos.

Estos trabajos se han centrado principalmente en la obtención de beneficios económicos, como la minimización de los costes o la obtención de la mínima duración del proyecto. Sin embargo, con la aceleración de la globalización y el rápido desarrollo de los países, la competencia por recursos energéticos ha aumentado drásticamente. Incluso, la importancia de tener en cuenta el consumo de energía en los modelos ha crecido de tal manera que, ahora es considerado con la misma relevancia que otras medidas de desempeño como la productividad y los

costes. Así, el objetivo principal de esta tesis es desarrollar un nuevo enfoque del RCPSP y del MRCPSP, basado en la eficiencia energética, la cual busca soluciones sostenibles en términos de tiempo y de consumo energético.

Para este fin, se ha propuesto una extensión del RCPSP denominada MRCPSP-ENERGY, la cual considera, además de los recursos tradicionales del RCPSP, un consumo de energía variable que da origen a distintos modos de ejecución de las actividades. Esta propuesta incluye un nuevo criterio de optimización basado en la eficiencia energética del proyecto, que tiene en cuenta de manera simultánea la minimización de la duración del proyecto y el consumo total de energía. Adicionalmente, con el objetivo de evaluar los métodos de solución para el MRCPSP-ENERGY, se ha ampliado la librería estándar de prueba más extendida para el RCPSP y se ha propuesto una nueva librería, denominada PSPLIB-ENERGY.

Para encontrar solución al problema propuesto, primero se analizaron los mejores métodos metaheurísticos que abordan el RCPSP. Luego, se identificó que estos métodos conducen a soluciones redundantes, entorpeciendo la búsqueda. Por tanto, se propuso un método evolutivo cuya principal aportación es el desarrollo de un nuevo operador de mutación que disminuye la generación de soluciones redundantes. Similarmente, en el caso multi-modal se detectó que los principales métodos de búsqueda también se centran en la representación de lista de actividades y por tanto generan soluciones redundantes. Como alternativa de solución para el MRCPSP-ENERGY, se mostró que la búsqueda puede realizarse enfocándose en la lista de modos, ya que diferentes listas de modos también pueden alcanzar soluciones distintas, generando un menor número de soluciones redundantes. Teniendo en cuenta estos hallazgos, se propuso un nuevo método evolutivo para resolver el MRCPSP-ENERGY, que unifica ambos métodos de búsqueda para realizarla en dos fases de optimización. Basándose en los resultados obtenidos en la PSPLIB-ENERGY, se concluye que el método propuesto es capaz de alcanzar soluciones altamente eficientes.

Resum

En la investigació operativa, el conjunt de problemes de seqüenciació d'activitats és considerat com un dels més rellevants a causa de la seua gran aplicabilitat i complexitat. Dins de l'àmplia varietat de problemes en este conjunt, destaca el problema de programació de tasques amb recursos restringits (RCPSP per la seua sigla en anglés) , perquè és considerat com el problema base més important en esta àrea i ha sigut objecte d'estudi de nombroses investigacions. Bàsicament, consistix d'un projecte subdividit en un conjunt d'activitats que es troben relacionades per mitjà de restriccions de precedència i requerixen, per a ser executades, una quantitat de cada tipus de recurs la disponibilitat màxima de la qual es troba limitada. L'objectiu és assignar els recursos a les activitats de la manera més eficient possible per a optimitzar una mesura d'exercici, per exemple, la duració total del projecte. Igualment important és la versió multi- modal del RCPSP, crida MRCPSP, en la que per a cada activitat hi ha múltiples modes d'execució que involucren una combinació diferent de recursos limitats, donant origen a un temps d'execució distint. En la literatura s'han abordat àmpliament estos dos problemes tant amb mètodes exactes com d'aproximació, sent estos últims els més reeixits.

Estos treballs s'han centrat principalment en l'obtenció de beneficis econòmics, com la minimització dels costos o l'obtenció de la mínima duració del projecte. No obstant això, amb l'acceleració de la globalització i el ràpid desenrotllament dels països, la competència per recursos energètics ha augmentat dràsticament. Inclús, la importància de tindre en compte el consum d'energia en els models ha crescut de tal manera que, ara és considerat amb la mateixa rellevància que altres mesures d'exercici com la productivitat i els costos. Així, l'objectiu principal d'esta tesi és desenrotllar un nou enfocament del RCPSP i del MRCPSP, basat en l'eficiència

energètica, la qual busca solucions sostenibles en termes de temps i de consum energètic.

Per a este fi, s'ha proposat una extensió del RCPSP denominada MRCPSP-ENERGY, la qual considera, a més dels recursos tradicionals del RCPSP, un consum d'energia variable que dóna origen a distints modes d'execució de les activitats. Esta proposta inclou un nou criteri d'optimització basat en l'eficiència energètica del projecte, que té en compte de manera simultània la minimització de la duració del projecte i el consum total d'energia. Addicionalment, amb l'objectiu d'avaluar els mètodes de solució per al MRCPSP-ENERGY, s'ha ampliat la llibreria estàndard de prova més estesa per al RCPSP i s'ha proposat una nova llibreria, denominada PSPLIB-ENERGY.

Per a trobar solució al problema proposat, primer es van analitzar els millors mètodes metaheurísticos que aborden el RCPSP. Després, es va identificar que estos mètodes conduïxen a solucions redundants, entorpint la busca. Per tant, es va proposar un mètode evolutiu la principal aportació del qual és el desenrotllament d'un nou operador de mutació que disminuïx la generació de solucions redundants. Semblantment, en el cas multi- modal es va detectar que els principals mètodes de busca també se centren en la representació de llista d'activitats i per tant generen solucions redundants. Com a alternativa de solució per al MRCPSP-ENERGY, es va mostrar que la busca pot realitzar-se enfocant-se en la llista de modes, ja que diferents llistes de modes també poden aconseguir solucions distintes, generant un menor nombre de solucions redundants. Tenint en compte estes troballes, es va proposar un nou mètode evolutiu per a resoldre el MRCPSP-ENERGY, que unifica ambdós mètodes de busca per a realitzar-la en dos fases d'optimització. Basant-se en els resultats obtinguts en la PSPLIB-ENERGY, es conclou que el mètode proposat és capaç d'aconseguir solucions altament eficients.

Abstract

In the field of operations research, the set of scheduling problems of activities is considered as one of the most relevant ones due to its great applicability and complexity. Within the broad variety of problems in this set, it is remarkable the Resource-Constrained Project Scheduling Problem (RCPSP), since it is regarded as the most important-base problem in this area and it has been the object of study in countless research projects. Basically, this problem consists of a project split into sets of activities that are related to each other by means of precedence-constraints, and require an amount of each limited resource, to be performed. The objective, then, is to allocate in the most efficient way those resources to the activities in order to optimize a scoring function such as the makespan. Similar in importance is the multimodal-version of the RCPSP, called MRCPSP, in which for each activity there exists multiple execution modes that involve a different combination of limited resources, giving rise to a different execution time. In the literature, it has been addressed widely these two problems with both exact methods and approximation methods, being these latter the most successful.

These research works have focused mainly on obtaining economic advantages such as costs and project time minimization. However, with the accelerating globalization and the fast countries' growing economies, the race for power resources have increased sharply. In fact, the importance of taking into account the energy consumption on modeling has become so important that it is now considered as important as other performance measures such as productivity and costs. Hence, the main goal of this Ph.D. dissertation is to develop a new RCPSP and MRCPSP approach based on the energetic efficiency, which is aimed at searching for sustainable solutions in terms of time and energy consumption.

To this end, it has been proposed an extension of the RCPSP, named MRCPSP-ENERGY, which considers besides the traditional resources of the RCPSP, a variable energetic consumption that generates different execution modes for the activities. This proposal includes a new optimization criterion based on the energetic efficiency of a project, which considers simultaneously the minimization of both the total duration and the energy consumption of such project. Moreover, in order to assess the solution methods for the MRCPSP-ENERGY, the standard library mostly used for this purpose has been extended and a new one has been proposed, called PSPLIB-ENERGY.

In order to solve the proposed problem, firstly, the most successful metaheuristics methods, which address the RCPSP, were analyzed. Secondly, it was shown that these methods lead to redundant solutions, hindering the search. Therefore, an evolutive method was proposed, whose main contribution is the development of a new mutation operator that reduces the number of redundant solutions. Similarly, in the multimodal case, it was determined that the most widespread searching methods are also focused on the activity list representation and therefore they yield redundant solutions. As a solution alternative for the MRCPSP-ENERGY, it was shown that such search can be carried out by focusing on the mode list representation, as different mode lists also reach diverse solutions, giving rise to a less number of redundant solutions. Keeping in mind this finds, it was proposed a new evolutive method for solving the MRCPSP-ENERGY, which unifies both searching methods such that the search is conducted with two optimization phases. Based on the obtained results given by the PSPLIB-ENERGY library, the proposed method proved to be able to reach highly efficient solutions.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se presentan los rasgos generales del área de investigación considerados en este trabajo. Posteriormente se describe el conjunto de problemas de programación de proyectos, derivados de esta línea de investigación, que contiene al problema que es objeto de estudio en esta tesis: el problema de programación de tareas con recursos restringidos. Adicionalmente, se presentan el marco de trabajo, la motivación de la investigación y los objetivos. Finalmente, se hace una descripción de la estructura de la tesis.

1.1 Generalidades

Desde el advenimiento de la primera revolución industrial que tuvo lugar en la segunda mitad del siglo $XVIII$ en el Reino Unido y que se expandió rápidamente por gran parte de Europa y Norteamérica, el mundo fue testigo del mayor crecimiento en tamaño y complejidad de la economía, tecnología y ciencias sociales en la historia de la humanidad. Los pequeños negocios artesanales de antaño se transformaron en las grandes multinacionales de hoy en día. Un componente esencial en este fenómeno fue el aumento de la división del trabajo y de las responsabilidades administrativas de las organizaciones. Esto generó un conjunto de problemas que aún existen en la industria actual, como, por ejemplo, la especialización e

independización relativa de los componentes en una organización, cada uno con sus propias metas; lo que ocasiona una pérdida de la visión general de empresa y hace difícil coordinar los objetivos de cada componente. De hecho, con frecuencia son objetivos contrarios. Por lo tanto, realizar la asignación de recursos disponibles a las diferentes actividades de cada componente de una empresa de la manera más eficaz posible para toda la organización, se convierte en una tarea titánica. Este tipo de problemas y la necesidad de ser resueltos promovieron el surgimiento de la *investigación operativa.*

Si bien las raíces de la investigación operativa pueden estar ligadas al momento en el que el ser humano fue capaz de sopesar diferentes alternativas y tomar decisiones con el fin de alcanzar un objetivo, está aceptado que la denominada investigación operativa como tal surgió en el conflicto bélico de la segunda guerra mundial. Para ese entonces, existía la urgente necesidad de asignar recursos escasos (provisiones, hombres o armas) a diferentes maniobras militares de la manera más eficientemente posible. Así, las grandes potencias implicadas convocaron a un gran número de científicos para que encuentren solución a estos problemas, en lo que ellos denominaron hacer investigación sobre operaciones militares. El uso del método científico sobre las decisiones militares fue decisivo para llevarse la victoria. Una vez finalizó la guerra, y gracias al éxito de la investigación operativa en las actividades bélicas, diversos sectores diferentes al militar se interesaron en implementar la investigación a cualquier sistema que involucre recursos, decisiones y objetivos, ya que en esencia los problemas en la industria eran iguales a los tratados en la segunda guerra mundial, pero en un contexto diferente. Ya para la década de los 50, la investigación operativa se estaba usando en organizaciones industriales, de negocios y del gobierno. Desde entonces, han crecido con rapidez.

1.2 Problemas de programación de proyectos

Dentro de la investigación operativa, existe un conjunto de problemas de suma relevancia llamados *problemas de programación de proyectos (scheduling problems*). Siguiendo la definición mostrada por Błażewicz y col. (2007), estos problemas pueden ser entendidos de manera general como problemas de asignación de recursos en el tiempo para ejecutar un conjunto de actividades que hacen parte de un proceso; especialmente relevantes para la manufactura, la industria, la computación y la construcción. Pero fácilmente ampliable a muchos otros campos, como la medicina, la biología, la genética, la física entre muchos otros. Básicamente a cualquier proyecto que se componga de actividades para alcanzar un fin determinado. Generalmente, las actividades compiten de forma individual por recursos, que pueden provenir de diferente naturaleza, por ejemplo, mano de obra, dinero,

máquinas, material, etc. De igual manera, las actividades pueden tener diferentes características como tiempos de espera, fechas de entrega, importancias diferentes, funciones que determinen su duración, etc. Además, la estructura del conjunto de actividades puede describir una relación entre ellas que puede ser descrita de diversas formas (por ejemplo, relaciones de precedencia). Finalmente, también se puede tener en cuenta diferentes criterios que miden la calidad del desempeño de las dichas actividades.

El uso de la investigación operativa en los problemas de programación de proyectos implica encontrar una mejor solución (frecuentemente denominada *solución óptima*) del problema considerado. El término *una* mejor solución en lugar de *la* mejor solución se debe a que es posible que exista un conjunto de soluciones que se puedan considerar mejores respecto a las otras. De esta manera, el propósito de resolver estos problemas es identificar el mejor curso de acción posible sobre un sistema que integra diferentes objetivos y recursos.

Inicialmente, se estudiaron y desarrollaron métodos exactos para resolver los problemas de programación de proyectos con gran éxito. Sin embargo, a medida que se plantean problemas más realistas, a gran escala y más complejos, estos métodos se volvían inviables para resolverlos en tiempos razonables. De hecho, existe una gran cantidad de problemas de enorme complejidad donde los métodos exactos no han podido encontrar una solución óptima, incluso en casos de prueba relativamente pequeños. Por esta razón, los investigadores han centrado su atención en métodos novedosos que son capaces de encontrar soluciones cercanas a una solución óptima en muy poco tiempo. Tales métodos son los denominados algoritmos metaheurísticos. Actualmente, estos dos enfoques de resolución, tanto el exacto como el aproximado, siguen siendo vigentes y son el objeto de estudio de numerosas investigaciones.

1.3 Definiciones y conceptos básicos

1.3.1 Elementos básicos de un problema de programación de proyectos

Un problema de programación de proyectos está principalmente constituido por cuatro elementos: actividades, recursos, relaciones de precedencia y funciones objetivo (Herroelen, De Reyck y Demeulemeester 1998), los cuales se describirán a continuación:

Actividades

Todo proyecto está constituido de elementos interrelacionados llamados actividades, las cuales deben ser ejecutadas en cierto orden para completar el proyecto de manera satisfactoria. Las actividades necesitan para su realización recursos y unidades de tiempo (periodos). Además, pueden ser ejecutadas en diversos modos, generalmente el cambio en el uso de recursos origina un modo de ejecución diferente y así, también cambia el tiempo de duración de la actividad. En algunos problemas las actividades pueden ser interrumpidas para completar otra actividad con mayor prioridad (*preemptive scheduling*), en otros casos por la naturaleza del problema, una vez se inicia una actividad no puede interrumpirse hasta ser completada (*non-preemptive scheduling*).

Adicionalmente, las actividades pueden tener un tiempo de duración determinista o estocástico. En el primer caso, las duraciones son conocidas desde el principio del proyecto, mientras que en el segundo caso las duraciones son aleatorias; pudiendo ser ajustadas a una función de distribución de probabilidad o bien ser estimadas teniendo en cuenta escenarios pesimistas, optimistas o el más probable, basado en datos históricos.

Relaciones de precedencia

Dentro de los proyectos, a menudo, por motivo de limitaciones tecnológicas o bien por la naturaleza de las actividades, estas solo pueden ser ejecutadas luego de que otra(s) finalicen. Ese conjunto de actividades que deben ser ejecutadas primero son denominadas actividades predecesoras, y la relación existente entre la actividad que se quiere ejecutar y sus predecesoras se denomina relación de precedencia. Si tenemos tres actividades A_1, A_2 y A_3, donde A_1 es predecesora de A_2 y esta última es predecesora de A_3, la actividad A_1 se denomina predecesora inmediata de A_2 y A_3 se denomina sucesora inmediata de A_2.

Estas relaciones suelen ser representadas mediante un grafo dirigido, se pueden encontrar en la literatura dos tipos de grafos, el primero denominado Actividades en los Nodos (*Activity-On-Node*, AON), donde cada nodo representa una actividad y cada arista representa una relación de precedencia; el segundo denominado Actividad en los Arcos (*Activity-On-Arc*, AOA), donde cada arista representa una actividad y cada par de nodos de una arista representan el inicio y finalización de la actividad. Sin embargo, la representación AON es más usada puesto que en la representación AOA un problema puede ser representado por varios grafos y además, suele necesitar la adición de nodos y aristas ficticios para su elaboración (Kolisch y Padman 2001).

En las relaciones de precedencia se define el parámetro FS_{ij} (Finish-Start), que determina el tiempo luego del cual la actividad j puede iniciarse una vez la actividad i haya terminado. Así, por ejemplo, un valor de $FS_{ij} = 10$ indica que luego de que la actividad i finalice debe pasar diez unidades de tiempo antes de iniciar la actividad j. El valor más usual es $FS_{ij} = 0$, es decir que inmediatamente luego de finalizada la actividad i, la actividad j puede iniciar. Estos periodos de tiempo entre actividades predecesoras y sucesoras son llamados los tiempos de preparación (*setup-times*). Un ejemplo práctico puede ser el tiempo de calibración de una máquina, o el periodo de limpieza de una sala de operaciones, entre otros.

Recursos

Generalmente, las actividades de los proyectos necesitan una cantidad de recursos para llevarse a cabo, los recursos pueden ser personas, energía, dinero, máquinas, materiales, entre muchos otros. De manera general, los recursos pueden ser definidos por tres elementos: unidades, tipos y categorías. Las unidades son la cantidad base del recurso a utilizar, los tipos de recursos hacen referencia a los conjuntos de unidades de recursos que tienen capacidad de realizar una misma función y las categorías son características que comparten un conjunto de restricciones. Teniendo en cuenta estos elementos los recursos pueden ser clasificados según el tipo de restricción, la divisibilidad y capacidad de interrupción.

Según la divisibilidad de los recursos se puede diferenciar dos clases: recursos continuos y discretos. Los recursos discretos son suministrados a las actividades en una cantidad contable en intervalos finitos, mientras que los recursos continuos pueden ser asignados en una cantidad arbitraria en un intervalo dado. Según la capacidad de interrupción de los recursos se puede encontrar dos tipos: recursos que permiten ser interrumpidos en un momento dado, con la posibilidad de ser usados en otra actividad y luego poder ser retornados, y aquellos que no permiten ser interrumpidos una vez son suministrados a una actividad. Según el tipo de restricción del recurso se puede diferenciar los recursos renovables, no renovables y doblemente restringidos, propuestos por Słowiński (1980) y Węglarz (1981). Esta última clasificación es la más usada en los problemas de programación de proyectos, por esta razón se describirán a continuación con mayor profundidad.

- **Recursos renovables.** Son aquellos cuyo total de unidades se encuentra disponible en cada unidad de tiempo. Es decir, una vez se haya completado una actividad que consuma recursos renovables, los liberará, dejándolos disponibles para ser usados en otras actividades a lo largo de los periodos del horizonte de planificación del proyecto. Un ejemplo típico de este tipo de recursos son las maquinas, herramientas o mano de obra.

- **Recursos no renovables.** Son aquellos que tienen una cantidad total de recursos disponibles para todo el proyecto, o un intervalo del mismo. A medida que las actividades consumen dicho recurso, este se va agotando. Así, las unidades de recurso asignadas a una actividad no podrán estar disponibles para su uso en otra actividad durante todo el proyecto. Como ejemplo representativo de los recursos no renovables se encuentra el dinero disponible para el proyecto (presupuesto o costos).

- **Recursos doblemente restringidos.** Estos están limitados de dos maneras simultáneas, en un intervalo de tiempo (como los no renovables) y por unidad de tiempo (como los renovables). Por ejemplo, cuando se dispone de un presupuesto para la realización de un proyecto, pero solo se puede usar parte de él cada día. Talbot (1982) demostró que los recursos doblemente restringidos pueden ser modelados mediante dos restricciones, una de recursos renovables y otra de recursos no renovables, siempre y cuando las unidades disponibles cada periodo de tiempo sean constantes.

- **Otras categorías.** Adicional a las categorías mencionadas anteriormente, en la literatura podemos encontrar una gran diversidad de recursos. Böttcher y col. (1999) propusieron los denominados *recursos renovables parcialmente restringidos*, los cuales están disponibles en ciertos intervalos de tiempo. Así, para cada recurso de este tipo existen un conjunto de intervalos de tiempo donde se limita su consumo. Bianco, Dell'Olmo y Grazia Speranza (1998) definieron los llamados *recursos dedicados*, los cuales solo pueden ser asignados a una actividad al mismo tiempo. Otro tipo de recursos son los denominados *adyacentes*, son recursos que tienen en cuenta, no solo la capacidad para de realizar una tarea cuando son suministrados a una actividad, sino también que dicho recurso usa un espacio físico que igualmente está limitado (Paulus y Hurink 2006). Los *recursos acumulativos* son estudiados por Neumann, Schwindt y Zimmermann (2003), su principal característica es que su disponibilidad depende del uso previo de otras actividades, por ejemplo, una bodega, donde su capacidad de almacenamiento depende de la cantidad de material guardado previamente. Una categoría especial de recursos son los llamados *recursos sincronizados* propuestos por Schwindt y Trautmann (2003), básicamente las unidades de este recurso solo pueden ser usadas si las actividades de un conjunto definido empiezan a ser procesadas al mismo tiempo. Un ejemplo puede ser un examen final en una universidad, donde los salones de clase disponibles para su realización pueden ser considerados recursos sincronizados, donde un conjunto de alumnos (actividades) debe iniciar el examen en un salón al mismo tiempo.

Funciones objetivo.

Por definición, todos los problemas de optimización tienen como objetivo encontrar una solución, dentro de un conjunto de posibles soluciones, que sea considerada la `"mejor"` basado en un criterio definido. Para poder distinguir si una solución es `"mejor"` respecto a otra se utiliza una *Función objetivo (performance measure).*

Cada función objetivo define un problema diferente, aunque el espacio de soluciones de búsqueda sea el mismo. Por ejemplo, la solución para un problema donde se busca que un proyecto se realice de la manera más rápida posible, es diferente para uno donde se busque completar un proyecto usando la menor cantidad de un recurso.

Las funciones objetivo para evaluar una solución de un proyecto se pueden catalogar en medidas regulares y no regulares. Las medidas regulares son funciones no decrecientes del tiempo de finalización de cada actividad. Es decir, busca que las actividades se programen lo más pronto posible y su valor de medida nunca mejoraría si se produce un retraso en la iniciación de alguna actividad. Un ejemplo típico es minimizar el *makespan*. Mientras que en las medidas no regulares un retraso en alguna actividad podría mejorar el valor de la función objetivo. Por ejemplo, en un problema donde cada unidad de tiempo en la que una actividad se encuentra ejecutándose está asociada a un flujo de dinero (costos o pagos) y se desea maximizar el valor presente neto (*Net Present Value*, NPV), aquí el valor de una solución puede mejorar si se incluye algún retraso en la iniciación de una actividad, pues el valor del dinero cambia con el tiempo.

A continuación, se describen las principales funciones de evaluación usadas en los problemas de programación de proyectos con recursos restringidos (Węglarz y col. 2011):

- **Objetivos basados en el tiempo.** Estos objetivos son los más estudiados en la literatura, son aquellos que están relacionados con el tiempo de finalización de las actividades C_i. El objetivo más representativo de este grupo es la minimización de la duración total del proyecto denominado en la literatura como $makespan = max\{C_i\}$. Su minimización, generalmente conduce a soluciones compactas y con un uso eficiente de recursos. Otro objetivo usual es la minimización del máximo retraso (*lateness*) $max\{L_i\}$. L_i es definida como la diferencia entre el tiempo de finalización de una actividad y su fecha de vencimiento (*due date*), cuando el valor es negativo L_i toma un valor de cero. También se han estudiado objetivos ponderados, como la suma ponderada de los tiempos de finalización de las actividades $\sum w_i * C_i$ o el número promedio de actividades con retraso.

- **Objetivos basados en recursos.** Estos objetivos son funciones basadas en el uso de recursos, donde cada recurso requerido para la finalización de una actividad tiene asociado un costo. Por ejemplo, Möhring (1984) propuso el problema de la inversión de recursos (*resource investment problem*), cuyo objetivo es minimizar el costo de adquisición de los recursos para un proyecto que debe terminarse antes de una fecha establecida. Otro objetivo relevante en esta categoría es el considerado en el problema de nivelación de recursos (*resource leveling problem*), donde se busca una solución que utilice los diferentes tipos de recursos lo más balanceado posible durante todo el proyecto, es decir que el número de unidades de recursos usados cambie lo menos posible de periodo a periodo (Brinkmann y Neumann 1996). El problema de la sobrecarga total (*total overload problem*) también considera objetivos basados en el uso de recursos, este problema asume que el exceso en el uso de un recurso tendrá un coste asociado, por tanto, el objetivo es minimizar el costo total del exceso de recurso.
- **Objetivos financieros.** La realización de un proyecto puede estar inmerso en un flujo constante de dinero. El flujo de salida suele estar relacionado con el uso de recursos o bien el poner en marcha una actividad, el flujo de entrada es usualmente obtenido por la finalización de una o un conjunto de actividades. En este tipo de problemas se usa el valor presente neto (NPV) para representar los flujos de dinero y tener en cuenta el valor del dinero en el tiempo. El objetivo del problema consiste en maximizar el NPV.

En esta tesis se aborda el problema de programación de tareas con recursos restringidos cuyo objetivo es asignar los recursos disponibles de tal forma que se complete un proyecto en el menor tiempo posible, esto es equivalente a encontrar los tiempos de inicio y finalización de cada actividad que compone el proyecto. Inicialmente, se estudia la versión básica, que considera una única forma de ejecutar las actividades. Luego se analiza el caso multi-modal, que consiste en considerar que las actividades pueden ser ejecutadas en diferentes formas, cada una de ellas con características diferentes.

Finalmente, el presente trabajo se enfoca en el estudio del problema de programación de tareas con recursos restringidos bajo un enfoque basado en la eficiencia energética, es decir incluir un componente energético a los problemas de programación, con el objetivo de encontrar soluciones sostenibles y energéticamente eficientes. Estos conceptos se analizarán con más detalle en la siguiente sección.

1.4 Marco de Trabajo

Las investigaciones realizadas en el campo de programación de proyectos se han enfocado principalmente en el estudio del problema de secuenciar actividades con el fin de minimizar la duración total del proyecto bajo un enfoque determinista, es decir donde se conocen los datos del problema de manera precisa, por ejemplo, las duraciones, las cantidades máximas de recursos disponibles o los requerimientos de cada actividad. Generalmente, las actividades están relacionadas entre sí por precedencias, es decir que una actividad no puede iniciarse sin antes haberse finalizado todas sus actividades predecesoras. Por otra parte, los recursos pueden ser renovables, no renovables o bien una mezcla de ambos. En la literatura se distinguen claramente dos problemas básicos, uno de ellos que considera un único modo de ejecución y el otro donde cada actividad tiene diversos modos de ejecución, generalmente relacionados con una combinación diferente de recursos.

1.4.1 El problema de programación de tareas con recursos restringidos, uni-modal y multi-modal

El problema de programación de tareas con recursos restringidos, (Resource - Constrained Project Scheduling Problem, RCPSP) en su versión uni-modal es un problema más general que el problema de programación de máquinas (Job-Shop Problem), el cual ha sido ampliamente estudiado por su aplicación en la industria, por esta razón el RCPSP ha sido objeto de numerosas investigaciones y es aún un problema de alto interés en la academia y la industria. Por otra parte, la versión multi-modal (MRCPSP) es la generalización del RCPSP, en este problema se consideran también actividades, precedencias y recursos, pero su objetivo no es solo encontrar los tiempos de inicio de cada actividad de tal manera que se minimiza la duración del proyecto, sino que además se debe decidir el modo de ejecución de cada actividad. De manera general, cada combinación de modos da a lugar a un RCPSP diferente.

Estos problemas han sido catalogados con una complejidad NP-Hard (Blazewicz, Lenstra y Kan 1983), esto quiere decir que el espacio de soluciones crece enormemente a medida que el tamaño del problema aumenta y hace imposible que los métodos de solución exacta puedan resolver todos los problemas de manera óptima en un tiempo razonable. Incluso existen problemas de prueba del RCPSP de 60 actividades para las cuales las soluciones óptimas aún no han sido encontradas. De manera similar, existen problemas del MRCPSP de 30 actividades para los cuales las soluciones óptimas siguen siendo desconocidas. Por esta razón, las investigaciones se han enfocado en desarrollo de métodos de aproximación (metaheurísticas) que son las únicas capaces de encontrar soluciones cercanas a una óptima en poco

tiempo. Kolisch y Hartmann (1999) y Kolisch y Hartmann (2006) presentan una amplia revisión de estos métodos para el RCPSP, de igual forma Węglarz y col. (2011) hacen una revisión de los principales métodos de solución para el MRCPSP.

De manera general, los algoritmos evolutivos y los híbridos son los que mejores resultados han obtenido al resolver el RCPSP en sus dos versiones. Estos algoritmos usualmente son poblacionales, es decir que realizan la búsqueda de manera simultánea usando varias soluciones (individuos). A pesar de ser problemas diferentes, los algoritmos usados para resolverlos comparten muchas características, por ejemplo, las representaciones de las soluciones están basadas en una lista de actividades o bien en una lista de valor clave, de igual forma para decodificar estas representaciones se usan dos esquemas que generan soluciones completas, el esquema en serie y el esquema en paralelo, incluso para generar las soluciones iniciales se utilizan algoritmos basados en reglas de prioridad. La principal diferencia entre los métodos aplicados para resolver la versión uni-modal y multi-modal, es el uso de diferentes procedimientos para considerar los modos de ejecución de las actividades, así los operadores que realizan la búsqueda deben tener en cuenta el amplio espacio de soluciones que genera la posibilidad de ejecutar una actividad en diferentes formas. Adicionalmente, el MRCPSP es usualmente asociado al uso de recursos renovables y no renovables, por esta razón, algunos problemas pueden no tener soluciones factibles con respecto a estos recursos no renovables, y por ello los métodos de solución suelen incorporar una función de penalización que considera dicha no factibilidad.

Con el desarrollo de nuevos métodos para la solución del RCPSP en sus dos versiones, surgió la necesidad de disponer de un conjunto de problemas de prueba para evaluar el rendimiento de los algoritmos y compararlos, dichos problemas deben ser lo suficientemente diversos y poseer diferentes grados de dificultad para que la evaluación sea objetiva. Kolisch y Sprecher (1996) propusieron la librería denominada PSPLIB que es considerada la librería estándar para el RCPSP uni-modal y multi-modal.

1.4.2 Optimización energética

El consumo de energía en el sector industrial crece a pasos agigantados. Basándose en el reporte de la Administración de Información de Energía de los Estados Unidos (EIA) del 2 016, el sector industrial, incluyendo el sector manufacturero, consumen aproximadamente un tercio del total de energía producida en el mundo (U.S. Energy Information Administration (EIA) 2016). La creciente demanda de energía ha provocado una enorme presión sobre el ambiente, por ejemplo, las emisiones de gases de efecto invernadero (como el CO_2 y el N_2O) tienen un impacto

notable sobre el calentamiento global, y el EIA estima que las emisiones de CO_2 relacionadas con la energía en todo el mundo aumentarán de 31 200 millones de toneladas métricas en 2 010 a 36 400 millones de toneladas métricas en 2 020 y 45 500 millones de toneladas métricas en 2 040. Así, las implicaciones medioambientales de los procesos industriales están ganando más y más importancia a medida que pasan los años. Por lo tanto, la reducción del consumo de energía en los proyectos de asignación de recursos es un aspecto crítico en el sector industrial actual (Zhang y col. 2016). El interés de los investigadores se centra cada vez más en el desarrollo de métodos para obtener soluciones energéticamente sostenibles y la programación orientada hacia la eficiencia energética es una forma clara para ahorrar energía en el proceso de planeación de un proyecto (Mouzon, Yildirim y Twomey 2007), además de ser un gran desafío.

Las principales estrategias para optimizar energéticamente los problemas de programación de proyectos en la industria se pueden clasificar en dos niveles (Salonitis y Ball 2013): la optimización a nivel de máquina y la optimización a nivel del sistema de manufactura. El primer nivel hace referencia a mejorar el proceso industrial en sí, para que sea más eficiente, es decir mejorar tecnológicamente las técnicas o las herramientas que son usadas en los procesos (por ejemplo, mejorar una técnica de corte o desarrollar un mejor sistema que cuantifique el consumo de energía). Por otra parte, el segundo nivel hace referencia a realizar la optimización teniendo en cuenta el sistema completo de manufactura, es decir, obtener soluciones más eficientes mediante la gestión actividades y la asignación adecuada de recursos al proyecto. Siendo este último nivel, un área relativamente reciente de estudio, pero del que se ha demostrado que provee una gran oportunidad de mejora a los problemas actuales. Por lo tanto, el desarrollo de métodos que proporcionen soluciones sostenibles a este nivel, constituyen un avance en esta área. Esta tesis se centra en el estudio del problema de programación de tareas con recursos restringidos bajo el este segundo enfoque de optimización.

1.5 Motivación y contribución

Considerando lo expuesto en las anteriores secciones, es fácil imaginar que los problemas de programación de actividades pueden ser encontrados casi en cualquier ámbito, en diversas situaciones del mundo real. Dentro de la amplia gama de problemas de programación de proyectos, destaca el denominado problema de programación de tareas con recursos restringidos (RCPSP), considerado uno de los problemas más generales y complejos. En esencia, se trata de un proyecto compuesto por actividades, las cuales deben coordinarse de tal manera que se minimice el tiempo de ejecución del proyecto, sujetas a dos tipos de restricciones:

restricciones de precedencia de las actividades, y restricciones de uso de recursos. Este problema pertenece al conjunto de problemas de complejidad *NP-Hard*, haciendo imperativo el uso y desarrollo de métodos aproximados para su solución. Dada la gran aplicabilidad y relevancia de este problema, este ha sido el seleccionado como objeto de estudio de la presente tesis.

En el pasado, los objetivos buscados en los problemas de programación de proyectos estaban orientados completamente al aumento en la productividad, maximización de utilidades o minimización de costos. Sin embargo, es innegable que, con el paso del tiempo, la acción del hombre produce serias e irreversibles consecuencias medioambientales. De hecho, solo el sector industrial consume aproximadamente la mitad de la energía producida del planeta (Zhang y col. 2016). De esta manera, la competición por los recursos energéticos está aumentando dramáticamente, y así también lo hace la presión sobre el medio ambiente con la alta producción de gases de efecto invernadero. Por esta razón, actualmente el esfuerzo de los investigadores está enfocado hacia la obtención de soluciones medioambientalmente sostenibles. Esta tesis también está enfocada en la resolución de problemas donde se minimice el consumo energético, y a su vez, los objetivos empresariales. Por lo tanto, se evidencia la gran importancia de abordar los problemas de programación de actividades mediante un enfoque de eficiencia energética.

1.5.1 Objetivos y principales contribuciones

El presente trabajo tiene como objetivo principal el desarrollo de una extensión del problema de programación de tareas con recursos restringidos (RCPSP) bajo un nuevo enfoque basado en la eficiencia energética, y el desarrollo de nuevas alternativas de resolución para el problema propuesto. De igual forma, los objetivos secundarios se detallan a continuación:

- Desarrollar nuevas estrategias de búsqueda que complementen los algoritmos actuales, basándose en el análisis de los métodos de solución metaheurísticos más exitosos reportados en la literatura para el RCPSP en su versión uni-modal y multi-modal, con especial atención en las ventajas y desventajas que presentan las representaciones de soluciones.
- Extender el RCPSP mediante la inclusión al problema de un consumo de energía y proponer un nuevo criterio de optimización basado en eficiencia del proyecto, que tenga en cuenta tanto el tiempo de complementación del proyecto como el consumo total de energía. Además, proponer un conjunto de casos de prueba para realizar comparaciones entre métodos de solución.

- Desarrollar un método de solución para el problema propuesto, analizando la estructura del espacio de soluciones, que sea capaz de aprovechar su topología para alcanzar soluciones eficientes.

Las principales contribuciones de esta tesis en relación al RCPSP, consisten en estudio de las representaciones de solución más relevantes, que son la piedra angular de los métodos metaheurísticos para resolver estos problemas; y el análisis del fenómeno de redundancia de soluciones que las dichas representaciones padecen. Adicionalmente, se desarrolla e implementa una mateheurística evolutiva cuya principal característica es el uso de un operador de mutación que disminuye la formación de soluciones redundantes. El método propuesto es evaluado usando la librería de PSPLIB alcanzando resultados altamente competitivos.

Respecto a la optimización energética, las contribuciones consisten en la propuesta de un nuevo problema de programación de tareas con recursos restringidos, denominado MRCPSP-ENERGY, el cual considera diferentes consumos de energía para cada actividad, las cuales dan origen a diversos modos de ejecución, pero a diferencia con el MRCPSP estándar, los recursos renovables se mantienen constantes y solo el consumo de energía es el que genera nuevos modos para las actividades. El objetivo de este problema usa un nuevo criterio basado en la eficiencia energética, el cual tiene en cuenta la minimización simultánea del tiempo total del proyecto y el consumo total de energía. Otra contribución, consiste en proponer una nueva librería, denominada PSPLIB-ENERGY, de casos de prueba del MRCPSP-ENERGY. Esta librería consta de 2 040 problemas divididos en 4 conjuntos, de 30, 60, 90, y 120 actividades.

Finalmente, respecto a los métodos de solución para el MRPCPS-ENERGY se desarrolla e implementa un algoritmo evolutivo que propone realizar la búsqueda principalmente sobre los modos de ejecución en lugar de hacerlo sobre la lista actividades como es usual hacerlo. Los resultados muestran que el espacio de soluciones creadas por los modos de ejecución no se ve afectado por la redundancia de las soluciones, además se muestra que el cambio sobre los modos, no solo afecta a la duración de las actividades afectadas, sino que, de hecho, pueden cambiar la secuencia de programación de las actividades.

1.6 Estructura del trabajo

En esta sección se describe la estructura por capítulos de la presente tesis. En el Capítulo 2 se describe completamente el problema objeto de estudio. Abarcando desde generalidades, definiciones, índices de caracterización y variantes del mismo, hasta una completa revisión del estado del arte referente a los principales métodos de solución reportados en la literatura académica. Por último, se presenta la librería de casos de prueba más usada en la literatura para realizar evaluaciones de rendimiento de los algoritmos desarrollados.

En el Capítulo 3 se estudia el fenómeno de redundancia en las soluciones que afecta a los procedimientos de búsqueda, luego se propone y se describe un algoritmo evolutivo para la resolución del problema, que incluye mecanismos para evitar la redundancia e introducir variabilidad en el proceso de búsqueda. Finalmente, se valida la propuesta mediante experimentación computacional sobre las librerías de casos de prueba más relevantes, comparando los resultados con los métodos reportados en la literatura.

En el Capítulo 4 se propone una extensión del RCPSP, denominado MRCPSP-ENERGY, que incluye además del uso de recursos renovables, un consumo de energía para cada actividad. Los diferentes consumos de energía dan a lugar a distintos modos de ejecución. De esta manera, el objetivo del problema propuesto es la maximización de la eficiencia relativa del proyecto. Este criterio tiene en cuenta la minimización tanto del *makespan* como del consumo energético. Además, se propone una librería de casos de prueba del problema propuesto, denominada PSPLIB-ENERGY, basada en la librería estándar para el RCPSP.

En el Capítulo 5 se proponen dos métodos evolutivos para resolver el problema propuesto en el Capítulo 4. El primero, está compuesto de elementos de los métodos más exitosos para el RCPSP, con la finalidad de comprobar si las estrategias de búsqueda para el problema base siguen siendo efectivas para un problema que considera el uso de energía. El segundo, es un método evolutivo en el cual se propone una novedosa alternativa de búsqueda basada en los modos de ejecución, en lugar de hacerlo sobre la representación de lista de actividades, como la mayoría de algoritmos en la literatura. Finalmente, se realiza una comparación entre los métodos de aproximación propuestos, y también respecto a un enfoque exacto.

En el Capítulo 6 se presentan las conclusiones de la investigación y las posibles líneas de trabajo futuro.

Capítulo 2

El problema de programación de tareas con recursos restringidos

Este capítulo está dedicado a la definición y descripción de uno de los problemas más estudiados en la literatura académica en el campo de la programación de proyectos: el Problema de Programación de Tareas con Recursos Restringidos (Resource-Constrained Project Scheduling Problem, RCPSP). Adicionalmente, se exponen las variantes más relevantes de este problema, así mismo se realiza una exhaustiva revisión del estado del arte acerca de los métodos de solución existentes. Posteriormente, se describen las librerías de prueba más usadas para la evaluación de estos métodos. Finalmente, se exponen las principales aportaciones presentes en la literatura sobre la optimización energética en problemas de programación de proyectos.

2.1 Introducción

Los problemas de programación de proyectos han sido estudiados desde la década de 1950, con el objetivo de encontrar métodos cuantitativos que ayuden a determinar la mejor secuencia de trabajos o actividades para la realización de un proyecto. Los modelos más representativos de aquella época fueron el método del camino crítico (*Critical Path Method*, CPM) y en el método PERT (*Program Evaluation and Review Technique*), que sentaron la base para la actual gestión de proyectos. La principal diferencia de estos métodos radica en la manera de estimar la duración de las actividades, siendo determinista en el caso del CPM y estocástica en el PERT. Sin embargo, con el paso del tiempo, se han fusionado para conformar el método del camino crítico actual. Este consiste en analizar las relaciones entre las actividades de un proyecto, usando un diagrama de red que lo representa, y determinar el momento de ejecución adecuado de cada actividad para minimizar el tiempo total del proyecto (llamado en la literatura *makespan*). Adicionalmente, permite identificar los denominados caminos críticos, que están compuestos de actividades que son las responsables de la duración del proyecto, es decir, que un cambio en el tiempo de duración en alguna de ellas afectará al *makespan*. Sin embargo, estos métodos tienen una gran limitación pues consideran que las actividades tienen a su disposición recursos ilimitados. Un supuesto poco realista en la mayoría de los casos reales.

En las siguientes décadas hasta el día hoy la gestión de proyectos y la programación de tareas se ha convertido en uno de los campos más estudiados tanto en la industria como en la academia, específicamente en la investigación operativa. Esto se debe a su gran aplicabilidad en diversos campos, pues un proyecto puede definirse, de manera general, como una combinación de actividades interrelacionadas que usan recursos y tiempo para lograr un objetivo. Así, un proyecto puede ser la construcción de un edificio, como la programación de los vuelos en un aeropuerto, hasta la planificación de producción de televisores o la organización de un evento deportivo.

Uno de los problemas de programación de proyectos más relevantes es el denominado: Problema de Programación de Tareas con Recursos Restringidos (*Resource-Constrained Project Scheduling Problem* ,RCPSP), pues es considerado el problema base o general en este campo. Este problema es el objeto de estudio de esta tesis. A continuación, en la sección 2.2 se describen y definen formalmente el RCPSP, junto con su principal generalización el llamado RCPSP multi-modal. En la sección 2.3 se exponen los principales índices de caracterización del RCPSP. Posteriormente, en la sección 2.4 se presentan las variantes más relevantes de estos problemas. En la sección 2.5 y la sección 2.6 se analizan y describen los métodos de solución más

relevantes desarrollados en la literatura académica. La sección 2.7 muestra las librerías de prueba más usadas en la evaluación de los métodos propuestos para resolver estos problemas. En la sección 2.8 se hace una revisión de la literatura sobre la optimización de la energía en los problemas de programación de proyectos. Finalmente, en la sección 2.9 se presentan las conclusiones del capítulo.

2.2 El problema de programación de tareas con recursos restringidos

2.2.1 El problema de programación de tareas con recursos restringidos en su versión uni-modal

El Problema de Programación de Tareas con Recursos Restringidos (*Resource-Constrained Project Scheduling Problem*, RCPSP) puede definirse formalmente de la siguiente manera: Sea un proyecto que está conformado por un conjunto de n actividades $X = \{1, ..., i, ..., n\}$ y tiene a su disposición un conjunto m de recursos renovables compartidos $R = \{1, ..., k, ..., m\}$, los cuales tienen una disponibilidad máxima de b_k unidades de cada recurso. Cada actividad i tiene una duración d_i que requiere para su realización un uso de r_{ik} unidades de cada k recurso. Generalmente, las actividades 1 y n son ficticias (sin duración y sin uso de recursos) y representan el inicio y el final del proyecto. Adicionalmente, las actividades tienen relaciones de precedencia, es decir una actividad no podrá iniciarse antes de que todas sus actividades predecesoras hayan finalizado. El objetivo es determinar el tiempo de inicio y finalización de cada actividad de manera que se minimice el *makespan*, teniendo en cuenta las restricciones de uso de recursos y de precedencias.

El RCPSP puede ser modelado como un problema de programación lineal entera mixta. Existen numerosos modelos matemáticos en la literatura. Como ejemplo, a continuación, se muestra una formulación basada en (Mingozzi y col. 1995).

Primero se define a Γ_j^{-1} como un conjunto que contiene las actividades predecesoras inmediatas de la actividad j y se define a ξ_{it} como una variable binaria de decisión que toma el valor de 1 cuando la actividad i empieza a ser ejecutada en un tiempo t y es 0 en otro caso. Para construir el modelo se debe acotar el número de variables de decisión, para lo cual se definen los intervalos de tiempo de inicio más temprano (*the earliest start time* (es_i)) y los intervalos de tiempo de inicio más tardío (*the latest start time* (ls_i)) para cada actividad. Estos valores se pueden calcular usando el método de revisión hacia adelante y hacia atrás sobre el problema sin tener en cuenta el uso de recursos. Adicionalmente, se requiere una cota superior del

tiempo total del proyecto (t_{max}). Cuanto menor sea el valor de la cota, menor será el número de variables a utilizar en el modelo. Finalmente, el modelo matemático es el siguiente:

$$Minimizar: \sum_{t=es_n}^{ls_n} t * \xi_{nt} \tag{2.1}$$

Sujeto a:

$$\sum_{t=es_i}^{ls_i} \xi_{it} = 1 \quad \forall\, i \in X \tag{2.2}$$

$$\sum_{t=es_j}^{ls_j} t * \xi_{jt} \geq d_i + \sum_{t=es_i}^{ls_i} t * \xi_{it} \quad \forall\, (i,j) : i \in \Gamma_j^{-1},\ j \in X \tag{2.3}$$

$$\sum_{i=1}^{n} r_{ik} * \sum_{\tau=\sigma(t,i)}^{t} \xi_{i\tau} \leq b_k \quad : \ \sigma(t,i) = max\{0, t - d_i + 1\}, \quad t = 0, ..., t_{max}, \quad k = 1, ..., m \tag{2.4}$$

$$\xi_{it} \in \{0,1\}, \quad i \in X,\ t = 0, ..., t_{max} \tag{2.5}$$

La Ecuación 2.1 representa la función objetivo, cuyo propósito es minimizar el inicio de la última actividad (n), es decir la actividad ficticia que representa la finalización de proyecto. La Ecuación 2.2 asegura que cada actividad solo inicia una vez. La Ecuación 2.3 representa las restricciones de precedencia, permitiendo que la actividad j inicie luego de que todas sus predecesoras hayan terminado. La Ecuación 2.4 asegura que nunca se excede la capacidad de cada tipo de recursos. Finalmente, la Ecuación 2.5 definen los dominios de las variables.

En la figura 2.1 se muestra un ejemplo de un RCPSP, este consiste de un proyecto con 11 actividades (incluidas las ficticias), donde se dispone de 3 diferentes tipos de recursos de los cuales se tiene una cantidad máxima disponible de 4 unidades para cada uno. Este ejemplo se representa mediante un grafo AON, donde cada nodo representa una actividad. En la parte inferior de cada nodo se encuentra el consumo de cada tipo de recurso y en la parte superior la duración de la actividad. Cada arco dirigido representa una relación de precedencia entre actividades. El camino

crítico del problema relajado (sin las restricciones de recursos) está diferenciado en negrita en el gráfico.

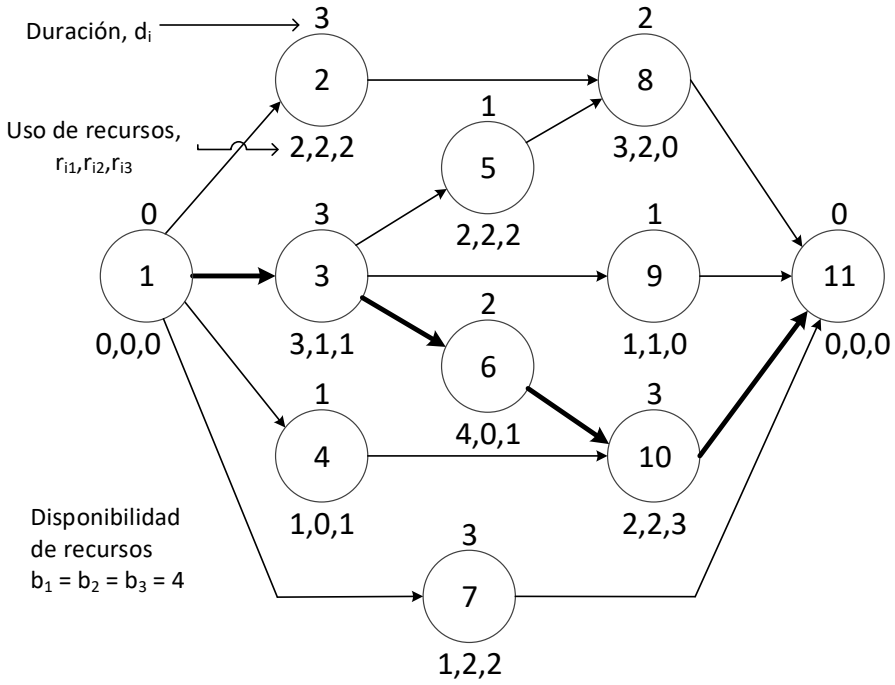

Figura 2.1: Un ejemplo del RCPSP representado por un grafo.

Una solución S es aquella secuencia de tiempos de inicio de cada actividad que compone un proyecto. Las soluciones se suelen representar mediante un diagrama de Gantt. En el eje horizontal se representa el tiempo y en el eje vertical el uso de recursos. Cuando hay más de un recurso se puede realizar tantos diagramas como recursos o bien usar una altura estándar en cada actividad. En la figura 2.2 se muestran dos diagramas de Gantt del problema ejemplo mostrado en la figura 2.1.

Las soluciones pueden ser clasificadas en no factibles, aquellas que violan alguna restricción de precedencia o de uso de recursos, soluciones factibles (figura 2.2.a), aquellas que representan una solución real al problema y que satisfacen todas las restricciones, y soluciones óptimas (figura 2.2.b), aquellas que tienen el menor

tiempo posible para la terminación del proyecto y no existe ninguna otra solución que las supere.

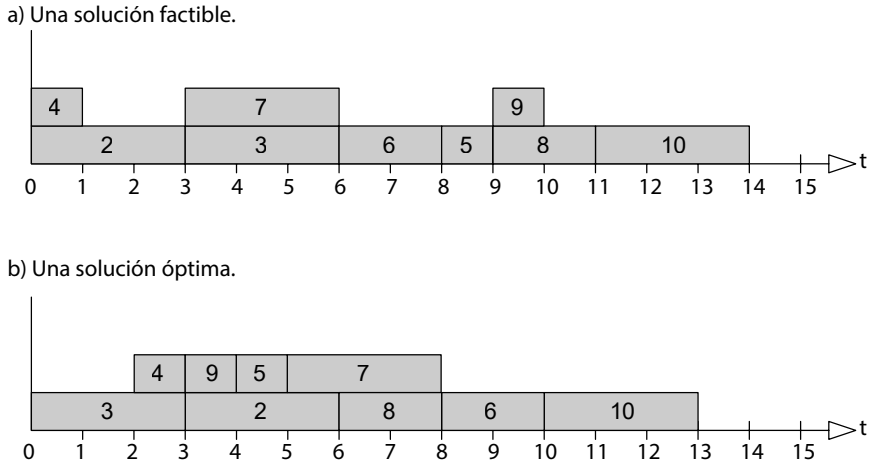

Figura 2.2: Dos diagramas de Gantt que representan una solución factible (a) y una solución óptima (b) del problema presentado en la figura 2.1

Es importante destacar que tanto las soluciones factibles como las óptimas son conjuntos de soluciones, algunas completamente diferentes, pero que logran obtener el mínimo *makespan* y otras muy similares, pero logran ser diferentes cambiando al menos un tiempo de inicio de una actividad (y, por tanto, el tiempo de finalización de la misma). Así, por ejemplo, otra solución óptima para el problema presentado en la figura 2.1 y diferente a la solución óptima presentada en la figura 2.2.b se puede obtener cambiando el tiempo de inicio de la actividad 4 en la figura 2.2.b de 2 a 0. Este intervalo de tiempo en el que una actividad puede `"moverse"` en la programación sin afectar al *makespan* y respetando las restricciones de recursos se denomina tiempo de holgura (*slack time*).

Blazewicz, Lenstra y Kan (1983) demostraron que el RCPSP es un problema de complejidad *NP-Hard*, pues es una generalización del problema clásico *job shop* que tiene a su vez una complejidad *NP-Complete*. Esto implica que, hasta donde se conoce, problemas relativamente pequeños se vuelven inviables de resolver por métodos exactos en tiempos razonables. Sin embargo, el RCPSP se convierte en un problema cuya solución puede encontrarse en tiempo polinomial si no se tienen en cuenta las restricciones de uso de recursos, el problema resultante es llamado `"problema relajado"` y se puede resolver mediante el método CPM/PERT.

2.2.2 El problema de programación de tareas con recursos restringidos en su versión multi-modal

El Problema Multi-Modal de Programación de Tareas con Recursos Restringidos (*Multi-Mode Resource-Constrained Project Scheduling Problem*, MRCPSP) es la generalización del RCPSP, donde se incluyen modos de ejecución a las actividades. Cada modo tiene un uso de recursos y un tiempo de duración diferente. Es usual encontrar en la literatura el MRCPSP que incluye el uso de recursos renovables y no renovables al mismo tiempo. Sin embargo, la principal característica de este problema, como se mencionó anteriormente, es la inclusión de diversos modos de ejecución a las actividades.

Siguiendo la definición de Talbot (1982), formalmente el MRCPSP está compuesto de los siguientes elementos: un proyecto que consiste en la realización de un conjunto I de n actividades $I = \{1, ..., i, ..., n\}$, un conjunto B de K^ρ tipos de recursos renovables ($B = \{1, ..., b, ...K^\rho\}$), de los cuales se tiene una disponibilidad máxima de R_b^ρ y un conjunto K de K^ν tipos de recursos no renovables ($K = \{1, ..., k, ...K^\nu\}$), de los cuales se tiene una disponibilidad máxima de R_k^ν.

Cada actividad $i \in I$ tiene M_i modos de ejecución, asociados a un duración d_{im} dependiente del modo $m \in M_i$. Cada modo requiere para su ejecución una cantidad r_{imb}^ρ de recursos renovables del tipo b y una cantidad r_{imk}^ν de recursos no renovables del tipo k. Como en el caso del RCPSP, las actividades 1 y n suelen ser actividades ficticias con duración y consumo cero de recursos, que representan el inicio y la terminación del proyecto.

Para ilustrar mejor las características que poseen las actividades en un MRCPSP, se presenta el siguiente ejemplo: supongamos que tenemos un proyecto compuesto de $n = 12$ actividades, cada una de ellas con $M_i = 3 \; \forall \; i \in I$ modos de ejecución. Además, las actividades requieren el uso de 4 tipos de recursos, dos renovables (RR1 y RR2) y dos no renovables (RN1 y RN2). Las cantidades disponibles en cada periodo de los recursos renovables son $R_1^\rho = 9$ y $R_2^\rho = 4$. Por otro lado, las cantidades máximas de los recursos no renovables disponibles para todo el proyecto son $R_1^\nu = 29$ y $R_k^\nu = 40$. Así, la tabla 2.1 muestra los datos necesarios para caracterizar la actividad ejemplo $i = 3$.

Adicionalmente y al igual que en RCPSP, las actividades en el MRCPSP están sujetas a restricciones de precedencia y de uso de recursos. Así, cada actividad no puede iniciarse antes de que todas sus actividades predecesoras hayan terminado. Por otra parte, no debe superarse el uso máximo de recursos renovables R_b^ρ en cada periodo de tiempo y tampoco debe superarse el uso máximo de recursos no

Actividad (i)	Modo (m)	Duración (d_{im})	RR1 (r^{ρ}_{im1})	RR2 (r^{ρ}_{im2})	RN1 (r^{ν}_{im1})	RN2 (r^{ν}_{im2})
	1	3	6	0	9	0
3	2	9	5	0	0	8
	3	10	0	6	0	6

Tabla 2.1: Caracterización de una actividad en un MRCPSP.

renovables disponibles para la totalidad del proyecto R^{ν}_{k}. El objetivo básico es minimizar el *makespan* respetando las restricciones anteriormente descritas.

El MRCPSP es un problema ampliamente conocido por su gran aplicabilidad a problemas reales y también debido a su dificultad inherente para ser resuelto. Blazewicz, Lenstra y Kan (1983) mostraron que el MRCPSP tiene una complejidad *NP-Hard*, pues es la generalización del RCPSP. Incluso, cuando hay más de un recurso no renovable el problema de encontrar una solución factible ya es *NP-Complete* (Kolisch 1995). Esta dificultad yace en que el espacio de búsqueda del MRCPSP crece enormemente en comparación con el RCPSP. Esto se debe a que una solución está compuesta de dos elementos: el tiempo de inicio y de finalización de cada actividad (al igual que el RCPSP), y un modo de ejecución para cada actividad. Cuando se fija un modo para las actividades, el problema se convierte en un RCPSP con la dificultad que conlleva resolverlo, es decir encontrar los tiempos de inicio de cada actividad que minimicen el *makespan*. Por lo tanto, cada combinación de modos entre las actividades representaría un RCPSP distinto a solucionar, ampliando cuantiosamente el espacio de solución.

La formulación matemática más usada en la literatura para el MRCPSP fue propuesta por Talbot (1982). En ella, se define la variable de decisión ξ_{imt} que toma el valor de 1 cuando la actividad i se ejecuta en el modo $m \in M_i$ e inicia en un tiempo t, y es 0 de otra manera. Igualmente que en caso uni-modal, el número de variables se define acorde a una cota superior T_{max}, cuanto menor sea la cota, menor será el número de variables en el modelo. La manera más simple de calcular dicha cota es sumando todas las duraciones de las actividades d_{im}, en el modo cuya duración sea la máxima. De esta manera, se pueden encontrar los intervalos de tiempo de inicio más temprano y más tardío $[es_i, ls_i]$ para cada actividad mediante el método de revisión hacia adelante y hacia atrás basado en las duraciones con menor valor.

Finalmente, siendo P_j el conjunto de actividades predecesoras inmediatas de la actividad j (conjunto equivalente a Γ_j^{-1} para el caso uni-modal), el modelo matemático se detalla a continuación:

$$Minimizar : \sum_{t=es_n}^{ls_n} t * \xi_{n1t} \tag{2.6}$$

Sujeto a:

$$\sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} \xi_{imt} = 1 \quad \forall\, i \in I \tag{2.7}$$

$$\sum_{m=1}^{M_i} \sum_{t=es_j}^{ls_j} t * \xi_{jmt} \geq \sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} (t + d_{im}) * \xi_{imt} \quad \forall\, i \in P_j,\ \forall\, j \in I \tag{2.8}$$

$$\sum_{i=0}^{n} \sum_{m=1}^{M_i} r_{imb}^{\rho} * \sum_{s=max\{t-d_{im},es_i\}}^{min\{t-1,ls_i\}} \xi_{ims} \leq R_b^{\rho} \quad \forall\, b \in B,\ \ t = 1, ..., T_{max} \tag{2.9}$$

$$\sum_{i=0}^{n} \sum_{m=1}^{M_i} r_{imk}^{\nu} * \sum_{t=es_i}^{ls_i} \xi_{imt} \leq R_k^{\nu} \quad \forall\, k \in K \tag{2.10}$$

$$\xi_{imt} \in \{0, 1\} \tag{2.11}$$

La Ecuación 2.6 representa la función objetivo a optimizar, que busca minimizar el tiempo de finalización de la actividad ficticia n, es decir el tiempo de finalización del proyecto. La Ecuación 2.7 garantiza que cada actividad solo puede iniciar una vez. La Ecuación 2.8 representa las restricciones de precedencia. La Ecuación 2.9 y la Ecuación 2.10 son las encargadas de que no se supere el uso de recursos renovables y no renovables, respectivamente. Finalmente, la Ecuación 2.11 define el dominio de las variables de decisión.

2.3 Índices de complejidad

Una de las características más relevantes en el RCPSP es la dificultad de estimar la complejidad de un problema antes de resolverlo, pues esta complejidad depende de la estructura de las restricciones de precedencia y de la relación entre la disponibilidad de los recursos y el consumo de las actividades. Ocasionando que no exista una medida global capaz de clasificar un problema según su dificultad para ser resuelto.

Por ejemplo, en la figura 2.3.a se muestra un proyecto sin relaciones de precedencias entre las actividades (la primera y la última son actividades ficticias), en este caso la solución es trivial pues las actividades pueden ser iniciadas todas a la vez para terminar el proyecto; por el contrario, la figura 2.3.c muestra un proyecto altamente restringido debido a las precedencias. Sin embargo, este problema también tiene una solución trivial, hacer una actividad tras otra. Finalmente, la figura 2.3.b muestra una combinación de las relaciones de precedencia de los otros esquemas, este problema puede considerarse el más difícil de los tres ya que su solución no es trivial.

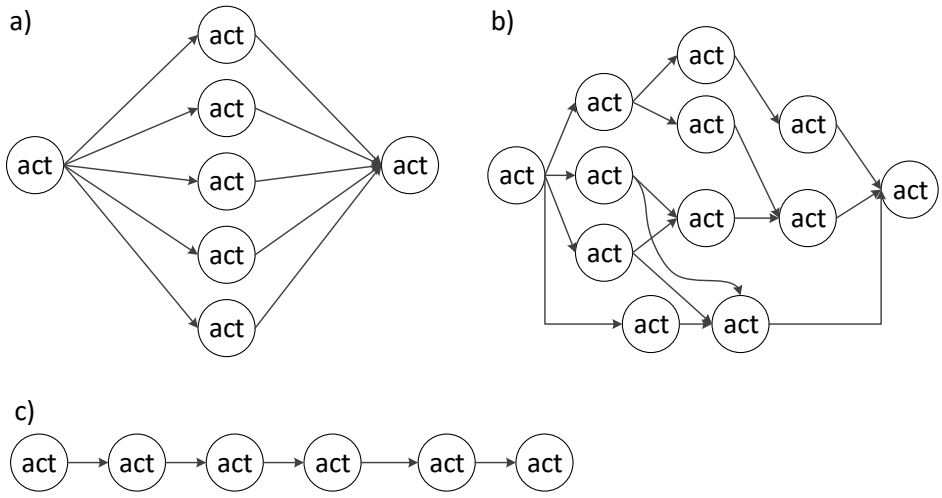

Figura 2.3: Ejemplos de la complejidad de la red de precedencias.

No obstante, un problema con una red de precedencias suficientemente compleja para que el problema no tenga una solución trivial, no podría considerarse `"fácil"` o `"difícil"` sin considerar la disponibilidad y el uso de los recursos. Por ejemplo,

si al problema de la figura 2.1 se le permite tener una abundante cantidad de cada tipo de recurso, el problema se reduce a un RCPSP relajado, es decir un problema fácil de programación de proyectos que no considera el uso de recursos. Cuya solución se puede encontrar fácilmente mediante el uso de los métodos CPM/PERT. Por otra parte, si se restringe demasiado la disponibilidad de recursos, el problema nuevamente es fácil de resolver pues las restricciones de recursos solo permitirían la programación de una actividad a la vez. Así, la dificultad de un RCPSP está ligada a la interacción entre relaciones de precedencia y restricciones de recursos. Es importante notar que no se ha mencionado el tamaño del problema, esto se debe a que un RCPSP puede tener una complejidad baja o alta de manera casi independiente del número de actividades. Así, existen problemas del RCPSP de 60 actividades cuya solución es equivalente a la solución del camino crítico (fáciles), mientras que la solución óptima de otros problemas del mismo número de actividades, aún no ha sido encontrada (difíciles). Por esta razón, aquellos indicadores que solo estén relacionados con el número de actividades no son apropiados para caracterizar la complejidad de un RCPSP.

Por esta razón, en las últimas décadas el esfuerzo de los investigadores se ha dedicado no solo a la búsqueda de métodos para resolver el RCPSP, sino también en encontrar métricas para caracterizar su complejidad. En esta sección se describen los índices de complejidad más usados en la literatura, basados en los trabajos de De Reyck y Herroelen (1996) y Kolisch, Sprecher y Drexl (1992).

- **Complejidad de la red (Network complexity, NC).** Presentado en la Ecuación 2.12, donde A es igual al número de precedencias no redundantes en un problema y n el número de actividades incluidas las ficticias. Este índice tiene en cuenta la estructura real del problema, estimando que tan restringido es el árbol de precedencias. Se interpreta como el número de precedencias no redundantes promedio por actividad. Sin embargo, como se ha mencionado anteriormente, al no tener en cuenta el uso de recursos, por sí solo, este indicador no es muy preciso.

$$NC = \frac{A}{n-2} \tag{2.12}$$

- **Índice de complejidad (Complexity Index, CI).** Presentado por De Reyck y Herroelen (1996), mide que tan lejos se encuentra la estructura del problema de una red de restricciones en paralelo o en serie. Para calcular su valor se deben usar los siguientes tres procedimientos (Bein, Kamburowski y Stallmann 1992): reducción en paralelo, reducción en serie y reducción de nodo. La reducción en paralelo, se aplica cuando dos actividades X y Y

tienen más de un arco que las une, reemplazando dichos arcos por uno solo. Estos arcos redundantes pueden surgir al implementar el procedimiento de reducción en serie o de nodo. La reducción en serie se puede aplicar cuando se tienen dos actividades X y Y que se encuentran conectadas, de manera que existen $e > 1$ actividades entre ellas $(X \to j_1 \to j_2, ..., j_e \to Y)$. Así, la reducción en serie reemplazaría todos los arcos existentes por uno solo $(X \to Y)$. La reducción de nodo se aplica cuando una actividad tiene un solo arco de entrada y varios de salida o bien varios arcos de entrada y uno solo de salida. Por ejemplo, supongamos que un arco de entrada $(X \to Y)$ y e arcos de salida $(Y \to j_1, Y \to j_2, ..., Y \to j_e)$, la reducción de nodo reemplazaría estos arcos por un conjunto $(X \to j_1, X \to j_2, ..., X \to j_e)$. Así, un grafo dirigido no cíclico puede transformarse en un número finito de reducciones aun grafo con un solo arco.

Finalmente, el índice de complejidad se define como el mínimo número de q de procedimientos de reducción (serie, paralelo y nodo) para transformar un grafo del RCPSP a una red con un solo arco.

Bein, Kamburowski y Stallmann (1992) desarrollaron un algoritmo para calcular el índice de complejidad en tiempo polinomial. Sin embargo, cuando el número de actividades aumenta, puede llevar mucho tiempo de cómputo el estimar este índice y basado en el estudio de De Reyck y Herroelen (1996), este indicador no explica más del 7 % de la variación del tiempo de solución de un RCPSP.

- **Factor de Recursos (Resource factor, RF).** Presentado en la Ecuación 2.13, donde existen K tipos de recursos renovables y r_{ik} es la cantidad de recursos tipo k que la actividad i usa para su realización. Este índice mide el número promedio de recursos usados por una actividad. Un valor de $RF = 1$ implica que las actividades utilizan todos los tipos de recursos y un valor $RF = 0$ implica que le problema no tiene restricciones de recursos.

  A pesar de que este índice involucra el uso de recursos, no considera las cantidades reales de consumo. Por esta razón, tampoco es útil para realizar una medida precisa sobre la complejidad.

$$RF = \frac{1}{(n-2) * k} * \sum_{i=2}^{n-2} \sum_{k=1}^{K} \begin{cases} 1 & \text{si } r_{ik} > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.13}$$

- **Intensidad de recursos (Resource strength, RS).** Presentado en la Ecuación 2.14, siendo b_k la cantidad máxima disponible del recurso k. Este

índice representa la relación entre el total disponible del recurso k y el consumo promedio de todas las actividades de ese recurso. Valores altos de este índice indican poca restricción en el recurso k, valores bajos indican gran restricción en la disponibilidad del recurso k.

Este índice es uno de los más usados para clasificar la complejidad de problema. Además, Kolisch, Sprecher y Drexl (1992) muestran que este indicador es significativo en la variación del tiempo de solución de un RCPSP.

$$RS_k = \frac{b_k}{\frac{1}{n-2} * \sum_{i=2}^{n-2} r_{ik}} \tag{2.14}$$

- **Recursos limitados (Resource constrainedness, RC).** Presentado en la Ecuación 2.15 y la Ecuación 2.16. Este índice compara el valor promedio de uso del recurso k, con respecto al total de recursos disponibles. Cuanto mayor sea el valor de este índice, mayor es la restricción del recurso k en función a su uso en las actividades.

$$RC_k = \frac{\bar{r_k}}{b_k} \tag{2.15}$$

$$\bar{r_k} = \frac{\sum_{i=2}^{n-2} r_{ik}}{\sum_{i=2}^{n-2} \begin{cases} 1 & \text{si } r_{ik} > 0 \\ 0 & \text{en otro caso} \end{cases}} \tag{2.16}$$

Hartmann y Kolisch (2000) realizaron un estudio con el fin de estudiar la importancia y el impacto de los índices de complejidad en el RCPSP y determinar cuáles de ellos son los que tienen mayor relevancia al clasificar la dificultad de un problema. El estudio consistió en resolver un conjunto de problemas de prueba del RCPSP, en los cuales la solución óptima es conocida, con heurísticas básicas y evaluar si al alterar los valores de los índices de complejidad, el tiempo de cómputo para resolver los casos de prueba cambiaba. Para lo cual se consideró los tres índices más usados en la literatura: complejidad de la red (NC), intensidad de recursos (RS) y factor de recursos (RF). Los resultados muestran que los índices que más afectan al tiempo de cómputo son el RS y el RF, mientras que los cambios al NC no eran significativos en todas las pruebas. Esto se debe a que es fácil construir un problema donde se tenga la misma complejidad de la red (es decir la estructura de precedencias) pero con complejidades para ser resueltos completamente diferente, cambiando las restricciones de uso de recursos.

Por otra parte, De Reyck y Herroelen (1996) analizaron los índices de RC y RC, en un experimento similar al propuesto por Hartmann y Kolisch (2000). Pero en este caso el objetivo era encontrar un patrón de comportamiento para la complejidad en lugar de evaluar su significación estadística. Los resultados muestran que existe una fase de transición de complejidad `fácil-difícil-fácil` con un comportamiento similar a una campana de Gauss con asimetría hacia la izquierda. Un resultado consistente con la hipótesis de que la complejidad no se encuentra en problemas levemente o altamente restringidos, sino en un rango medio de restricción.

Sin embargo, los estudios muestran que gran parte de la variación del tiempo de cómputo no es explicada por la variación de los índices estudiados. Así, se concluye que aún no se ha encontrado un índice que logre categorizar la dificultad de los problemas del RCPSP de forma precisa, y estudiar la complejidad de este tipo de problemas es aún un campo de estudio que está en desarrollo.

2.4 Variantes del problema RCPSP y MRCPSP

En esta sección se agrupan las principales variantes del RCPSP y el MRCPSP tratadas en la literatura. La clasificación está basada en las características más importantes que comparten los problemas. Dentro de cada grupo se pueden encontrar y definir otras variantes del problema, mezclando las características expuestas en la subsección 1.3.1.

2.4.1 El problema del costo-beneficio entre el tiempo y el coste

El problema del costo-beneficio entre el tiempo y el coste (time/cost trade-off problem, TCTP) es una variante del MRCPSP, pues se considera el uso de actividades en diferentes modos. Cada actividad tiene un tiempo de procesamiento basada en una función creciente respecto al costo. El costo de una actividad se define como un consumo agregado de los recursos no renovables de dicha actividad. Así, el TCTP se define de manera similar a un MRCPSP donde solo se considera un recurso no renovable, que representa el costo de ejecutar una actividad. Un ejemplo típico de este recurso es el presupuesto asignado a un proyecto. Normalmente, la aceleración de una actividad conlleva a mayor uso del recurso y viceversa. Además, se definen las duraciones mínimas y máximas que puede tener cada actividad, que corresponde a la máxima y mínima asignación del recurso, respectivamente.

Generalmente, en el TCTP se consideran tres objetivos. El primero, dada una fecha límite de terminación del proyecto, se desea encontrar los tiempos de inicio de cada actividad y su modo de ejecución de tal manera que se minimice el costo

total del proyecto, conocido como el problema de la fecha límite (deadline problem). En el segundo, se fija una cantidad máxima de recurso y el objetivo es minimizar el *makespan*, conocido como el problema del presupuesto (budget problem). El tercero se trata de encontrar los valores (T, R) (*makespan* y disponibilidad máxima del recurso, respectivamente), de tal manera que no existan otros puntos (T', R') menores o iguales a (T, R). A este problema con el último objetivo se le conoce como el problema de la curva de tiempo-costo (time-cost curve problem), pues por lo general se pueden encontrar varias duplas (T_l, R_l) de soluciones no dominadas que conforman una curva de soluciones denominadas eficientes.

El TCTP se puede clasificar en dos tipos de problemas en función del tipo de variables de la duración y del uso del recurso. Así, podemos definir el problema discreto del costo-beneficio entre el tiempo y el coste (discrete time/cost trade-off problem, DTCTP) y el problema continuo del costo-beneficio entre el tiempo y el coste (continuous time/cost trade-off problem, CTCTP). En esta sección se enfocará en el caso discreto.

Dentro de las principales variantes del DTCTP podemos encontrar el problema abordado por Vanhoucke, Demeulemeester y Herroelen (2002), el cual considera unas restricciones especiales de interrupción de tiempo. Estas imponen un tiempo específico de inicio para las actividades, pero las obliga también a permanecer inactivas en unos intervalos de tiempo específico. Por otra parte, Peng, Wang y Chengen (2009) propusieron el denominado problema muti-modal de recursos restringidos para el DTCTP (multi-mode resource-constrained DTCTP, MRC-DTCTP). Básicamente, los autores hacen una inclusión de recursos renovables al DTCTP, basando su propuesta en que la realización de un proyecto suele involucrar el uso de recursos renovables como la mano de obra. Para esto, los autores relacionan los recursos renovables y no renovables con costos indirectos y costos directos de mano de obra. Además, definen que el modo de ejecución de una actividad está originado en la posibilidad de hacer un pago de horas extras a la mano de obra, de manera que se puede finalizar una actividad en menor tiempo. Así, un costo indirecto lo definen como un salario fijo que debe ser pagado siempre que se requiera de mano de obra, y un costo directo es aquel que se incurre cuando se pagan horas extras a la mano de obra.

En la versión de fecha límite del DTCTP la función objetivo de la minimización del costo puede ser reemplazada por la maximización del valor presente neto (NPV). En tal caso, flujos positivos de efectivo están asociados con eventos o instantes en el tiempo, mientras que los costos están asociados con las actividades. Erenguc, Tufekci y Zappe (1993) fueron los primeros en considerar el DTCTP con flujos de fondos negativos a lo largo del proyecto. Además, las duraciones de las actividades pueden ser reducidas incurriendo en mayores costos directos. El

objetivo del problema es determinar las duraciones de las actividades (los modos) y la secuenciación de los tiempos de inicio de las actividades, de tal manera que el valor presente neto de todos los flujos sea maximizado.

Otras variantes relevantes del DTCTP fueron definidas por Vanhoucke y Debels (2007), quienes detallan tres extensiones del problema con el fin de hacer frente a ajustes más realistas: restricciones de cambio de tiempos, de continuidad del trabajo, y la maximización del valor presente neto. Adicionalmente, Tareghian y Taheri (2006) propusieron una variante que considera la duración y calidad de las actividades de un proyecto como variables discretas, las cuales son funciones no crecientes de un único recurso no renovable: el dinero. El objetivo es minimizar el costo total del proyecto, maximizar la calidad del mismo y cumplir con una fecha límite establecida.

2.4.2 Problema del costo-beneficio entre el tiempo y los recursos

El problema del costo-beneficio entre el tiempo y los recursos (time/resource trade-off problem) fue introducido en De Reyck, Demeulemeester y Herroelen (1998). En este, cada actividad tiene una carga de trabajo pre-establecida W, usualmente medida en hombres por actividad y puede ser ejecutada en múltiples o infinitos modos, es decir con diferentes duraciones y requerimientos de recursos; siempre y cuando el producto de la duración y su consumo asociado de recursos sea como mínimo igual a W. Se considera un único recurso renovable, y la duración de cada actividad está representada por una función no creciente del uso de este recurso. Usualmente, mientras menor sea la duración de una actividad, mayor será el uso del recurso renovable y viceversa. El objetivo del problema es minimizar la duración del proyecto, al mismo tiempo que se determina el modo en que cada actividad es secuenciada, sujeto a las restricciones de precedencia y recursos.

Existen dos variantes de este problema. La primera, considera que la duración de una actividad es una función discreta no creciente de la cantidad del recurso renovable asignado a cada actividad, mientras que la otra variante considera que la duración es una función continua. Esta última variante no ha sido considerada explícitamente. Sin embargo, una variedad de problemas de secuenciación de maquinado usando tanto duraciones continuas como discretas han sido estudiados (Błażewicz y col. 2007). Hasta ahora pocos artículos han tratado el problema en su versión discreta. De Reyck, Demeulemeester y Herroelen (1998) proponen una búsqueda tabú. Luego, los mismos autores en (Demeulemeester, De reyck y Herroelen 2000) desarrollaron un procedimiento de ramificación y acotamiento para el problema. Tanto enfoques de solución como una extensa descripción del problema pueden ser consultados en (Demeulemeester y Herroelen 2002). Desde

entonces solamente Ranjbar y Kianfar (2007); Vanhoucke y Debels (2008); y Ranjbar, Reyck y Kianfar (2009) han tratado este problema.

Como se puede observar las principales variantes al RCPSP en su versión multimodal consideran generalmente como objetivo el optimizar metas productivas, por ejemplo, la minimización del tiempo total del proyecto dado un presupuesto o la minimización los costos asociados al proyecto dada una fecha de límite, o maximizar las utilidades con la terminación de ciertas actividades bajo una fecha establecida. Además, consideran el uso de recursos relacionados con el dinero, los cuales usualmente son no renovables, o bien relacionados con la mano de obra, siendo estos renovables. Sin embargo, los problemas de programación con recursos restringidos bajo un enfoque sostenible, donde se buscan soluciones eficientes que minimicen el impacto ambiental y que, a su vez, sigan siendo competitivos para los fines del mercado, no han sido ampliamente estudiados. Por esta razón en esta tesis se propone abarcar esta temática con una extensión del RCPSP basada en un criterio de eficiencia energética.

2.5 Métodos de búsqueda exacta

Dentro de esta categoría se agrupan los algoritmos que tienen como característica el uso de técnicas analíticas o matemáticas, las cuales aseguran la convergencia a un óptimo global, si este existe. Estos métodos son diseñados bajo supuestos y características específicas tales como continuidad, diferenciabilidad, espacio de búsqueda pequeño o linealidad, entre otros. Usando estas características y con base en teoremas matemáticos, los métodos de búsqueda exacta garantizan una solución óptima (Michalewicz y Fogel 2000).

Sin embargo, no siempre se puede hacer uso de los métodos exactos, ya que presentan varias desventajas que impiden su uso en muchos problemas matemáticos. Michalewicz y Fogel (2000) afirman que la razón por la cual existen una gran cantidad de métodos exactos es que ninguno de ellos es realmente robusto, es decir que se pueda aplicar a una gran diversidad de problemas y que siga siendo eficiente en el proceso de encontrar la solución. Esta problemática es ocasionada por las características inherentes de un problema, ya que estas pueden impedir el uso de ciertos métodos exactos y crean la necesidad de elaborar otros métodos, también bajo un enfoque exacto. Sin embargo, existen otros problemas que bajo este enfoque no pueden ser resueltos por el gran tamaño del espacio de búsqueda o por la complejidad del mismo.

El RCPSP en sus dos versiones, pertenecen a esta categoría de problemas de gran complejidad, donde el espacio de búsqueda crece exponencialmente con el número

de actividades, por tanto, enfoques como la búsqueda exhaustiva no son eficientes. Además, las formulaciones matemáticas propuestas usando programación lineal entera mixta, a pesar de ser diversas, no han sido lo suficientemente eficientes como para encontrar soluciones exactas para problemas de tamaño real. Aún existen instancias con solo 30 actividades para las cuales se desconocen los valores óptimos. Por estos motivos, la mayoría de los algoritmos exactos para resolver el RCPSP en su versión uni-modal y multi-modal están basados en el algoritmo de ramificación y acotamiento (Branch and Bound, $B\&B$) y en la idea de enumerar secuencias parciales. Para mayor detalle referimos al lector a (Morillo, Moreno y Díaz 2014a).

Los esquemas de ramificación y acotamiento son en esencia una búsqueda exhaustiva pero más eficiente, tratan de explorar el espacio de búsqueda mediante soluciones parciales, desplegando cada posible decisión en nodos y ramas de un grafo de árbol, pero no se enumeran de manera explícita todas las posibles combinaciones, sino que, cuando se tiene seguridad de que una secuencia parcial no puede encontrar una mejor solución que la actual, se corta esa rama del árbol con todas las posibles soluciones subyacentes que se puedan construir. Para el MRCPSP, estos métodos deben enumerar un número aún mayor de posibilidades por la presencia de modos de ejecución. Talbot (1982) fue uno de los primeros en proponer un algoritmo de ramificación y acotamiento para el MRCPSP. Este constaba de dos etapas. En la primera, las actividades, recursos y modos son organizados usando unas reglas de prioridad establecidas. En la segunda, se usa una heurística basada en una regla de prioridad para calcular una cota superior y entonces se implemente un $B\&B$ con retroceso. En cada nivel del árbol de búsqueda, la primera actividad de la lista ordenada es adicionada a la secuencia parcial usando su primer modo, de tal manera que su tiempo de finalización sea el menor posible, respetando las restricciones de precedencia y recursos. Si el modo usado para la actividad no permite que esta sea adicionada a la secuencia, entonces se le asigna el siguiente modo. Si ninguno de los modos es factible por recursos, entonces el algoritmo retrocede al nivel anterior en el árbol, y se intenta programar nuevamente la actividad de ese nivel, de tal forma que su nuevo tiempo de finalización sea el menor posible en la secuencia parcial, pero mayor al tiempo de finalización anterior. De no ser posible esta reprogramación, se intenta nuevamente su adición a la secuencia usando el siguiente modo disponible y así sucesivamente. El proceso continúa hasta que se intenta retroceder de nivel estando en el nodo raíz o hasta que la actividad n es programada. En el primer caso, la mejor solución encontrada hasta el momento es la óptima, mientras que, en el segundo, una nueva solución mejorada es encontrada, estableciendo una nueva cota superior y el proceso comienza nuevamente con la actividad en la primera posición del nivel cero programada en su primer modo. Esto es cierto a menos que el valor de la solución sea igual al de la cota inferior y en tal caso, la solución también es óptima.

Desde entonces, se han propuesto diversos métodos basados el algoritmo de ramificación y acotamiento con diferentes variantes, Sprecher, Hartmann y Drexl (1997) proponen un $B\&B$ en el cual se usa un esquema de enumeración llamado alternativas de modo y retraso (mode and delay alternatives), el cual es una extensión del concepto alternativo de retraso propuesto por Christofides, Alvarez-Valdes y Tamarit (1987) y usado por Demeulemeester y Herroelen (1992) para el RCPSP. Las principales diferencias entre este enfoque y el $B\&B$ son que, en cada nivel, más de una actividad puede ser programada y que en el nivel actual se pueden deshacer decisiones tomadas en el nivel previo.

Más recientemente, métodos similares, pero más eficientes han sido desarrolladas, por ejemplo Zhu, Bard y Yu (2006) propusieron un algoritmo de ramificación y corte (branch and cut), aunque fue propuesto para la versión multi-modal del RCPSP con recursos parcialmente renovables, puede ser usado para el problema clásico del MRCPSP porque tanto los recursos renovables como no renovables pueden ser fácilmente modelados usando recursos parcialmente renovables. En este enfoque la relajación lineal del modelo de programación lineal entera es usada para obtener una cota inferior de la duración del proyecto en cada nodo del árbol de búsqueda. Si el nodo del árbol de búsqueda tiene una solución fraccionaria, entonces el algoritmo trata de encontrar cortes, es decir, desigualdades válidas que son violadas por la solución fraccionaria, pero son satisfechas por las soluciones enteras factibles representadas por ese nodo en el árbol de búsqueda. Si no se encuentran cortes en el nodo, entonces la ramificación es llevada a cabo, creando nuevos nodos en el árbol. El desempeño de su algoritmo es evaluado sobre la base de experimentos computacionales donde los datos contienen instancias del MRCPSP con 20 y 30 actividades de la PSPLIB (subsección 2.7.1). El algoritmo encontró el óptimo de todos los casos de prueba con 20 actividades y 506 de las 552 instancias con 30 actividades. Adicionalmente, para 5 instancias con 30 actividades, el algoritmo de $B\&C$ encontró soluciones con un *makespan* menor que el mejor conocido hasta entonces y para 23 instancias la solución encontrada fue peor que la reportada por la PSPLIB.

2.6 Métodos de búsqueda aproximada

En la práctica, muchos problemas de programación de proyectos, como el RCPSP, tienen una alta complejidad. Algunas veces debido a su gran tamaño, o bien a combinaciones de restricciones de recursos que amplían enormemente el espacio de búsqueda. En estos casos, los métodos exactos son incapaces de encontrar la solución o las soluciones óptimas para realizar la asignación de recursos en tiempos razonables. De hecho, en la literatura existen casos de prueba para el RCPSP, con

60 actividades de las cuales aún no se conocen las soluciones óptimas. Así mismo, en el caso multi-modal existen problemas con solo 30 actividades de las que se desconoce su solución.

En estos casos es deseable encontrar al menos una solución factible que se encuentre cerca de la óptima. Una alternativa ampliamente usada y eficiente para buscar esa solución son los métodos de búsqueda aproximada, también conocidos como métodos heurísticos y metaheurísticos. Estos podrían definirse como métodos, generalmente iterativos, que realizan la búsqueda de la mejor solución mediante la generación de nuevas soluciones, modificando previas o bien creándolas bajo diferentes criterios. De estar bien diseñada, una metaheurística puede encontrar soluciones muy cercanas a las óptimas o bien ser iguales a ellas. Sin embargo, no pueden garantizar una convergencia hacia una solución óptima, e incluso, generalmente no se puede saber que tan lejos se está del óptimo. Algunas de las principales razones de la popularidad y éxito de estos métodos de aproximación son su gran versatilidad que permite su implementación para la solución de diversos problemas y la relativa simplicidad de sus conceptos subyacentes (Morillo, Moreno y Díaz 2014b).

De manera general, se pueden establecer dos grandes diferencias entre las heurísticas y las metaheurísticas. La primera diferencia es que las heurísticas son creadas y ajustadas a problemas específicos, mientras que las metaheurísticas son métodos generales que proporcionan una estructura básica para la aplicación de las heurísticas. La segunda diferencia, es que las metaheurísticas incorporan una estrategia *inteligente* que les permiten escapar de óptimos locales.

En esta sección se describen los principales métodos de búsqueda aproximada para resolver el RCPSP, así como para el MRCPSP. Como se ha mencionado en la sección 2.2, el caso multi-modal es una generalización del caso uni-modal y la principal diferencia entre estos problemas, es la inclusión de diferentes modos de ejecución asociados a diferentes consumos de recursos (renovables o no renovables), por esta razón los métodos de solución aproximada para cada problema comparten muchas características, como los métodos usados para generar soluciones factibles, las diferentes formas de representar soluciones, entre muchas otras. De esta manera, y adaptando la clasificación presentada por Van Peteghem y Vanhoucke (2014), los métodos aproximados están agrupados en: esquemas de representación, esquemas generadores de secuencia, reglas de prioridad, algoritmos metaheurísticos y procedimientos de búsquedas locales.

2.6.1 Esquemas de representación

Uno de los principales elementos que comparten las metaheurísticas para resolver el RCPSP es la representación de las soluciones (Representation schemes). Pues generalmente, los enfoques metaheurísticos no realizan la búsqueda utilizando programaciones de tareas `"reales"`, sino esquemas que las representan y las codifican. Es decir, no se usan las actividades programadas en el tiempo como tal, sino una abstracción de esa realidad codificada en una representación. Así, una representación es un modelo simplificado de una solución, a la cual se le aplicarán diferentes mecanismos para generar nuevas soluciones.

Principalmente existen 5 esquemas de representación para el RCPSP (Kolisch y Hartmann 1999), que se describen a continuación:

- **Representación de lista de actividades (Activity list representantion, AL).** Esta representación consiste en un vector de j_n actividades, como se presenta en la Ecuación 2.17. La posición que tenga cada actividad en dicho vector, indica la prioridad de esa actividad de ser programada. Así, por ejemplo, dada una AL, la actividad j_3 tiene una prioridad menor que la actividad j_1.

$$AL = (j_1, j_2, ..., j_i, ..., j_n) \tag{2.17}$$

  Generalmente se usa una lista factible en precedencias, es decir una actividad no puede tener mayor prioridad (estar ubicada más a la izquierda de la lista) que su predecesora. Esto permite generar siempre soluciones factibles al problema.

- **Representación de valor clave (Random key representantion, RK).** El esquema de representación de valor clave o de valor clave aleatorio consiste en un vector de n elementos (Ecuación 2.18). El valor del elemento v_i representa el valor de la prioridad de la actividad i. Estos valores de prioridad v_i de cada actividad son generados aleatoriamente.

$$RK = (v_1, v_2, ..., v_i, ..., v_n) \tag{2.18}$$

- **Representación de la regla de prioridad (Priority rule representantion, PR).** Esta representación es muy similar a la RK, pues también consiste en un vector de n elementos, como se muestra en la Ecuación 2.19, y cada elemento π_i representa el valor de prioridad para la actividad i. Sin embargo, cada π_i es calculado mediante una regla de prioridad, las cuales

se detallan en la subsección 2.6.3. En resumen, son valores basados en las características propias de cada actividad en comparación con las demás, por ejemplo, una regla de prioridad puede ser la actividad de menor tiempo de ejecución. En esta representación cada π_i puede ser calculado con una regla de prioridad distinta.

$$PR = (\pi_1, \pi_2, ..., \pi_i, ..., \pi_n) \tag{2.19}$$

- **Representación de vectores de desplazamiento (Shift vector representation, SV).** Esta representación consiste en un vector de n enteros no negativos, como se muestra en la Ecuación 2.20. Cada valor σ_i representa las unidades de tiempo que desplazan el instante de inicio determinado para la actividad i. Así, el tiempo de inicio de una actividad se define como el máximo tiempo de sus predecesoras sumándole σ_i. Como es de esperarse, esta representación puede dar a lugar a soluciones no factibles.

$$SV = (\sigma_1, \sigma_2, ..., \sigma_i, ..., \sigma_n) \tag{2.20}$$

- **Representación directa de programación (Direct schedule representation).** Esta representación consiste en un vector de n elementos de tiempos (Tis_i), donde cada uno de ellos define directamente el tiempo de inicio de la actividad i. A pesar de que este esquema representa directamente una solución, definir las reglas de movimiento para generar un vecindario de soluciones es complejo y poco eficiente, además que puede generar soluciones no factibles.

Como se puede apreciar, la codificación tiene un rol fundamental sobre los métodos metaheurísticos de solución para el RCPSP en sus dos versiones, puesto que todos los mecanismos de búsqueda son implementados mediante cambios en esta codificación para luego ser decodificados, obteniendo así, los tiempos de inicio de cada actividad y el *makespan* correspondiente del proyecto. Por lo tanto, para que la codificación permita un buen funcionamiento del algoritmo debe tener ciertas características, dentro de las cuales destaca la capacidad de que cada solución tenga una única codificación diferente, esto garantiza que cada movimiento que se realice sobre la codificación, por ejemplo, en una iteración, esté buscando en una solución real distinta de la que se parte. Cuando existe más de una codificación para una solución real se denomina solución redundante. Todas las codificaciones que hemos descrito, salvo la representación directa, pueden verse afectadas por este problema, es decir que diferentes codificaciones pueden generar a una misma solución decodificada. En esta tesis abordamos esta problemática analizando sus

causas y posibles alternativas para evitar la formación de soluciones con múltiples representaciones.

Respecto a la representación de los modos en el caso del MRCPSP, se puede encontrar, principalmente, dos esquemas:

- **Representación de lista de modos (Mode list representation).** Consiste en una lista de n elementos, donde cada posición i en la lista representa el modo de ejecución de la actividad i.
- **Representación de vector de modos (Mode vector representation).** Esta representación se usa junto con una AL. De esta manera, los modos de ejecución de las actividades se representan en un vector de n elementos de modos (m_i) que se encuentra organizado según la lista de actividades. Por ejemplo, el elemento m_3 corresponde al modo de ejecución de la actividad que se encuentra en la posición 3 de la lista de actividades.

2.6.2 Esquemas generadores de secuencia

Un esquema generador de secuencia (Sequence generator schemes, SGS) es uno de los procedimientos más importantes en las metaheurísticas, porque permiten construir una solución factible a partir de un esquema de representación de soluciones que defina una prioridad entre las actividades. Es decir, son algoritmos iterativos que permiten decodificar una solución, expresada, por ejemplo, con una lista de actividades, la cual tiene definida la prioridad cada actividad. Estos algoritmos construyen en cada iteración una solución parcial, es decir una solución donde solo están presentes un subconjunto del total de actividades, y al final del algoritmo se genera la solución completa.

En la literatura se distinguen principalmente dos esquemas generadores de secuencia: en paralelo y en serie, los cuales serán detallados a continuación. Para ello se tomará como problema ejemplo el presentado en la figura 2.1, además sin perder generalidad se usará la representación de lista de actividades para explicar el proceso.

- **SGS en serie.** Es un algoritmo constructivo que genera una secuencia en n iteraciones, es decir en un número de pasos igual al número de actividades ($n-2$ si se incluyen las ficticias). El método toma una a una, cada actividad dependiendo de la representación, por ejemplo, de una lista de actividades, y la programa lo más pronto posible, teniendo en cuenta las restricciones de precedencia y de consumo de recursos de la solución parcial. El proceso se repite hasta que no quedan actividades por programar.

Por ejemplo, tomemos la lista de actividades presentada en la figura 2.4: $AL = (1, 3, 2, 4, 5, 6, 10, 9, 7, 8, 11)$ para el problema que muestra la figura 2.1. Recordamos que la actividad 1 y 11 son ficticias. El algoritmo inicia tomando la actividad 3, al ser la primera se programa en un tiempo $t = 0$. La siguiente actividad es la 2, esta no tiene ningún predecesor así que podría iniciar en un tiempo $t = 0$ junto a la 3, salvo que el consumo del recurso tipo 1 de la actividad 2 ($r_{21} = 2$) y de la actividad ya programada 3 ($r_{31} = 3$) superan la cantidad máxima disponible $b_1 = 4$, por lo tanto, la actividad 2 se programa luego de la actividad 3 en un tiempo $t = 3$. La siguiente actividad es la 4, esta tampoco tiene predecesores y su el consumo de recursos junto con la actividad 3 no superan el límite para ningún tipo de recurso k. Así que es programada en el tiempo $t = 0$. La siguiente es la actividad 5, esta sí tiene predecesores, así que solo podrá ser programada luego de que la actividad 3 haya finalizado. En ese punto $t = 3$ ya existe otra actividad siendo programada: la actividad 2, por tanto, debe verificarse si el uso de los k recursos sumados de la actividad 2 y la 5 superan el máximo disponible. En este caso, $r_{2k} + r_{5k} <= b_k \; \forall \; k \in (1, 2, 3)$, así que la actividad 5 se programa en un tiempo $t = 3$. Este proceso se repite con todas las actividades. En la figura 2.4 se muestra el resultado final, es decir la solución completa.

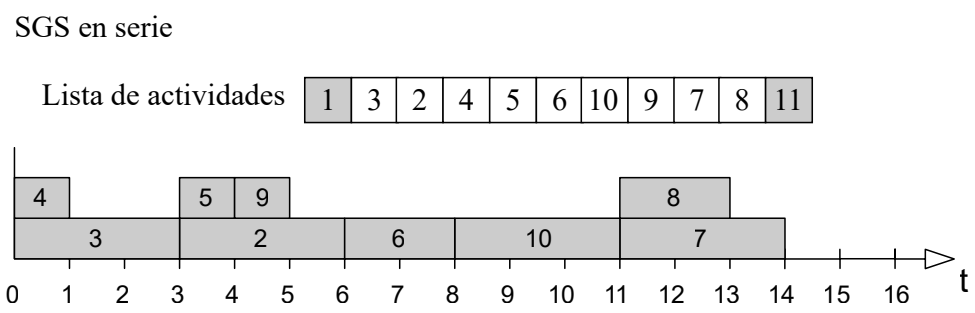

Figura 2.4: Construcción de una solución mediante el SGS en serie.

- **SGS en paralelo.** Este algoritmo genera soluciones en máximo n iteraciones. Cada iteración está asociada a un punto de decisión en un tiempo t, denominado $PD(t)$. Estos puntos se definen como el instante t en el cual se liberan o están disponibles los recursos. Adicionalmente, en cada $PD(t)$ se crea un conjunto de actividades elegibles ($AE(t)$) para ese instante t. Estas actividades deben ser factibles en precedencias. Así, en cada $PD(t)$ se programa una actividad que se encuentre dentro del conjunto $AE(t)$, la cual será elegida basado en una representación, por ejemplo, en el orden de prioridad de una lista de actividades. Este proceso se repite hasta que no

se puedan programar más actividades en ese instante y se define un nuevo punto de decisión $PD(t)$. El algoritmo finaliza cuando el conjunto $AE(t)$ está vacío.

Por ejemplo, tomemos la lista de actividades presentada en la figura 2.5: $AL = (1, 3, 2, 4, 5, 6, 10, 9, 7, 8, 11)$ para el problema que muestra la figura 2.1. Cabe mencionar que la lista es la misma que se usó en el esquema en serie, con la finalidad de mostrar las diferencias. El algoritmo inicia definiendo el primer punto de decisión $PD(0)$ en $t = 0$ pues es ahí donde se dispone de los recursos. Luego se crea el conjunto de actividades $AE(0) = (2, 3, 4, 7)$, estas actividades conforman el conjunto $AE(0)$ pues al no tener predecesores, cualquiera de ellas podría ser programada. Ahora, el conjunto elegible se reordena basándose en las prioridades de la lista de actividades. Así, el conjunto quedaría $AE(0) = (3, 2, 4, 7)$. La primera actividad a programar en el instante $t = 0$ es la actividad 3, como la actividad 2 no puede estar programada junto con la actividad 3 se pasa a la siguiente, la actividad 4 si puede ser programada junto con la actividad 3 así que es programada en $t = 0$, la última actividad es la 7, esta no puede ser programada junto con la actividad 3 y 4 simultáneamente, puesto que $r_{31} + r_{41} + r_{71} \not\leq b_1 \rightarrow (3+1+1 \not\leq 4)$. Luego, se define el siguiente punto de decisión en el instante donde se liberan recursos, en este caso es al finalizar la actividad 4 en $t = 1$, definiéndose $PD(1)$. El conjunto de actividades elegibles reordenado es $AE(1) = (2, 7)$. Esto se debe a que la finalización de la actividad 4 no libera nuevas actividades a ser programadas. Como ya hemos comprobado que la actividad 2 no puede hacerse junto con la 3 (que en el instante $t = 1$ se encuentra activa), se programa la actividad 7 que no viola las restricciones de uso de recursos junto con la actividad 3. El siguiente punto de decisión se define en $t = 3$ ($PD(3)$), donde se liberan recursos por la finalización de la actividad 3, el conjunto de elegibles reordenado es $AE(3) = (2, 5, 6, 9)$, como la actividad 2 se puede ejecutar junto con la actividad 7 (activa en $t = 3$), es programada en ese instante, ninguna de las demás pueden ser programadas junto con la actividad 7 y la 2. El proceso continúa hasta que el conjunto de actividades elegibles esté vacío. En la tabla 2.2 se muestran las iteraciones del anterior ejemplo, los conjuntos generados y las actividades programadas. Finalmente, la figura 2.5 muestra la solución completa generada por el esquema en paralelo.

Adicionalmente, el uso de un SGS para decodificar una solución puede ser en dos direcciones: hacia adelante y hacia atrás. Cuando se realiza hacia adelante, el procedimiento se realiza como se ha descrito anteriormente. En contraste, cuando se realiza hacia atrás, se intercambian las relaciones de precedencia por las relaciones sucesoras, cambiando toda la dirección del grafo. En la figura 2.6 se muestran dos

$PD(t)$	$AE(t)$	Actividades programadas
$PD(0)$	$AE(0) = (3, 2, 4, 7)$	$(3, 4)$
$PD(1)$	$AE(1) = (2, 7)$	(7)
$PD(3)$	$AE(3) = (2, 5, 6, 9)$	(2)
$PD(4)$	$AE(4) = (5, 6, 9)$	(5)
$PD(5)$	$AE(5) = (6, 9)$	(9)
$PD(6)$	$AE(6) = (6, 8)$	(6)
$PD(8)$	$AE(8) = (10, 8)$	(10)
$PD(11)$	$AE(11) = (8)$	(8)
$PD(13)$	$AE(13) = (\emptyset)$	$(\emptyset)$

Tabla 2.2: Conjuntos de actividades generados en cada iteración mediante el SGS en paralelo del ejemplo presentado en la figura 2.5.

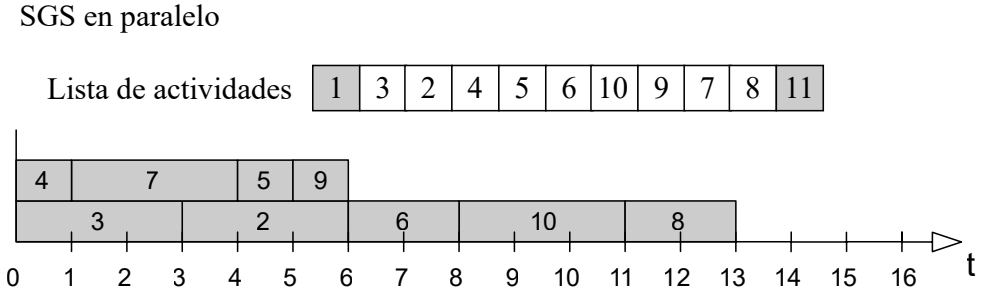

Figura 2.5: Construcción de una solución mediante el SGS en paralelo.

soluciones generadas con un SGS en serie y usan una representación de lista de actividades. La figura 2.6.b usa la lista de actividades invertida de la figura 2.6.a. Como se puede apreciar, las soluciones alcanzadas cambiando la dirección de un SGS pueden ser diferentes.

Los SGS siempre producen una solución factible para el RCPSP y el MRCPSP (siempre y cuando la asignación de modos no supere el uso de los recursos no renovables). Estas soluciones pueden ser clasificadas en tres conjuntos (Kim 2009): secuencias semi-activas (semi-active schedules, SAS), secuencias activas (active schedule, AS) y secuencias no retrasadas (non-delay schedules, NDS). El conjunto de SAS está compuesto de secuencias donde las actividades son programadas lo más pronto posible, es decir, no hay soluciones donde alguna actividad pueda iniciar más temprano sin afectar el orden de otra actividad. El conjunto de AS está

Figura 2.6: Dos soluciones factibles construidas mediante un SGS serial en diferentes direcciones.

compuesto por secuencias donde ninguna actividad pueda iniciar más temprano sin retrasar el proyecto o violando la restricción de recursos. Finalmente, el conjunto NDS está compuesto por secuencias donde ningún recurso se mantiene inactivo cuando se puede utilizar para programar una actividad. En la figura 2.7 se muestra un esquema de la clasificación de soluciones.

Los esquemas en serie y en paralelo siempre encuentran la solución óptima para el RCPSP relajado. Sin embargo, las soluciones óptimas del RCPSP completo siempre se encuentran dentro del conjunto AS, pero como se puede observar en la figura 2.7 el conjunto NDS está contenido en el conjunto AS. Así, algunas soluciones óptimas podrían estar dentro del conjunto NDS. El SGS en serie produce soluciones del tipo AS mientras que el SGS en paralelo produce soluciones del tipo (NDS). A pesar de que las soluciones óptimas siempre pueden ser alcanzadas por el esquema en serie, el esquema en paralelo, por lo general, produce soluciones más compactas (de menor *makespan*). Por este motivo, en la literatura se usan los dos esquemas

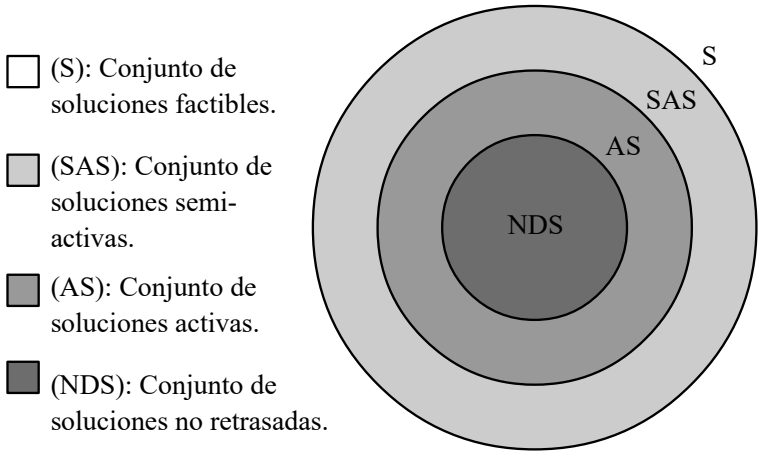

Figura 2.7: Espacio de soluciones del RCPSP.

generadores. Respecto a la complejidad de los algoritmos, tanto el esquema en serie como en paralelo tienen una complejidad $O(n^2K)$ (Kolisch y Hartmann 1999).

2.6.3 Reglas de prioridad

Como se ha mencionado anteriormente, un SGS requiere de una prioridad preestablecida de las actividades. Por lo general, los esquemas de representación expresan los valores de prioridad intrínseca, como la representación AL o la RK. Así, las reglas de prioridad (Priority rules) se definen como heurísticas que determinan el orden de prioridad de las actividades que son expresadas en los esquemas de representación o bien son usadas directamente por los SGS para construir una solución.

Existen numerosas reglas de prioridad propuestas en la literatura. Desde la más simple, asignando la prioridad aleatoriamente, hasta las más complejas que son calculadas de manera dinámica. Siguiendo la clasificación hecha por Klein (2000), estas podrían ser catalogadas en 4 grupos.

- **Reglas basadas en la red (Network-based rules).** Usan la información contenida en el grafo del problema pero sin considerar el uso de recursos. Así, estas reglas de prioridad se definen, por ejemplo, basándose en la duración de las actividades, el número de actividades sucesoras inmediatas o el total de actividades sucesoras.

- **Reglas basadas en la ruta crítica (Critical path-based rules).** Estas reglas de prioridad usan la información obtenida al resolver el RCPSP relajado (sin tener en cuenta las restricciones de uso de recursos) mediante el método de CPM/PERT. Por tanto, las reglas se basan en los valores de los tiempos de inicio más temprano y tardío (the earliest and latest starting time), así como en los tiempos de finalización más temprano y tardío (the earliest and latest finishing time).

- **Reglas basadas en el uso de recursos (Resource-based rules).** Definen las prioridades dependiendo del uso de los recursos de las actividades. En esta categoría también se puede dar mayor prioridad al recurso que se considere más relevante, mediante el uso de pesos en las ecuaciones o bien luego de estimar el recurso más restringido. En esta categoría, por ejemplo, encontramos la regla de la actividad con el mayor valor medio de uso de recursos.

- **Reglas compuestas (Composite rules).** El objetivo de estas reglas es integrar aquellas que usen diferentes tipos de información. Por ejemplo, mezclar la información contenida en la red y en el uso de recursos.

Adicionalmente, las reglas de prioridad pueden ser dinámicas o estáticas. Las primeras basan su clasificación en la información proveniente de secuencias parciales. Así, en cada iteración, deben ser recalculadas. Las reglas estáticas, son aquellas que no dependen la una secuencia parcial, sino de la información provista del proyecto o bien de una simplificación de la misma.

No existe una regla de prioridad que supere a las demás en todos los casos. Esto se debe a que establecer un orden específico en el cual las actividades deben ser programadas, es equivalente a resolver el problema. Sin embargo, es de gran importancia estudiar las reglas que mejores resultados obtienen para su uso en los métodos metaheurísticos. Klein (2000) realizó un estudio de más de 70 reglas de prioridad para comparar su desempeño. Basados en los resultados obtenidos en ese estudio, en la tabla 2.3 se muestran las mejores reglas de prioridad (de mejor desempeño). Esta tabla usa la misma notación presentada en la sección 2.2, la notación adicional es definida en la tabla 2.4.

Regla de prioridad	Criterio	Valor de prioridad
Greatest rank positional weight all (GRPW)	max	$d_j + \sum_{i \in F_j} d_i$
Latest starting time (LST)	min	Ls_j
TIRMOS	max	$\omega \frac{(Ls_n - Ls_j)}{(Ls_n - Ls_1)} + (1 - \omega) * \frac{acr_j}{acr_1}$

Tabla 2.3: Mejores reglas de prioridad.

Expresión	Definición
F_j	Conjunto de actividades predecesoras totales de la actividad j
ω	Parámetro referente a la importancia entre los factores $0 < \omega < 1$
acr_j	$acr_j = max \left\{ \sum_{i \in \Pi_{jh}} \sum_{k=1}^{m} \frac{r_{ik}}{b_k} \right\} \quad h = 1, ..., \pi_j$
$h = 1, ..., \pi_j$	Es el conjunto de caminos posibles que conectan la actividad j con la actividad ficticia n
Π_{jh}	Es el conjunto de actividades comprendidas entre el camino h

Tabla 2.4: Notación de las reglas de prioridad.

2.6.4 Métodos metaheurísticos

Como se mencionó anteriormente, los algoritmos metaheurísticos (Metaheuristic algorithms) son métodos generales de resolución de problemas. Dentro de sus principales características destacan la inclusión de estrategias inteligentes que les permiten escapar de óptimos locales y la posibilidad de adaptarse para la resolución de múltiples problemas de optimización. Dentro de este grupo podemos encontrar los algoritmos genéticos (genetic algorithm, GA), el enfriamiento simulado (annealing simulated, AS), búsqueda tabú (tabu search, TS), búsqueda dispersa (scatter search, SS), optimización de colonia de hormigas (ant colony optimization, ACO), entre otras.

El propósito de esta sección es brindar una revisión de la literatura académica sobre los métodos metaheurísticos aplicados al MRCPSP y al RCPSP, omitiendo descripciones generales acerca de estos métodos, puesto que están fuera del en-

foque de la tesis. Para descripciones generales referimos al lector al *Handbook of approximation algorithms and metaheuristics* de Gonzalez (2007).

Inicialmente, se hará un inciso en los denominados métodos de x pasadas (x-pass methods) y en método de pre-proceso para el MRPCSP, y luego, se describen los principales métodos metaheurísticos para resolver el MRCPSP y el RCPSP.

Métodos de x pasadas

Siguiendo la descripción de Hartmann y Kolisch (2000), los métodos de x pasadas (x-pass methods) consisten en heurísticas basadas en reglas de prioridad. En esencia, cuentan con dos elementos para construir una solución factible: un SGS y una o más reglas de prioridad, además pueden ser de una pasada o de múltiples pasadas. El caso más simple es con una pasada (single pass methods), en el cual se selecciona un SGS y una regla de prioridad para generar una sola solución. En contraste, en el caso de múltiples pasadas (multi-pass methods) existen muchas posibilidades de combinar los SGS y las reglas de prioridad. Hartmann y Kolisch (2000) agrupan estos métodos en: métodos de múltiples reglas de prioridad (multi-priority rule methods), métodos de programación hacia adelante y hacia atrás (forward-backward scheduling methods), y métodos de muestreo (sampling methods). Los primeros, en cada pasada eligen una regla de prioridad diferente y generan una solución nueva. Los segundos, cambian de dirección de programación en cada pasada. Los terceros se diferencian por asignar una probabilidad a las actividades a programar, en lugar de seleccionar la de mejor valor según una regla de prioridad. A pesar de las diferencias, todos los métodos de múltiples pasadas no tienen en cuenta la información provista por soluciones pasadas.

Método de pre-proceso para el MRCPSP

Dado que el MRCPSP puede dar a lugar a problemas que no son factibles, el primer procedimiento que debe hacerse es el pre-proceso propuesto por Sprecher, Hartmann y Drexl (1997). El cual consiste de tres pasos: el primero, eliminar todos los modos que no pueden ser ejecutados (modos que superan el uso de recursos); el segundo, consiste en eliminar los recursos no renovables redundantes (recurso con una disponibilidad mayor que la máxima total requerida) y finalmente, el tercer paso que consiste en eliminar todos los modos ineficientes (aquellos con una mayor duración y uso de recursos que otros).

Algoritmos genéticos

Los algoritmos genéticos (Genetic Algorithm, GA) fueron propuestos inicialmente por Holland (1975) quien se inspiró en la teoría Darwiniana de la evolución de los seres vivos. La idea fundamental es que, al igual que los individuos mejor adaptados son quienes sobreviven, las mejores soluciones serán quienes prevalezcan luego de varias generaciones, mientras que los individuos con características que no favorecen a su supervivencia, perecen. En un algoritmo genético, las `"adaptaciones"` se pueden conseguir mediante la herencia de características deseables de los padres y a través de la mutación, que es la encargada de introducir diversidad en la población. Para implementar este método se debe codificar las soluciones factibles de tal manera que puedan ser manipuladas por los operadores genéticos. Principalmente, en un algoritmo genético se consideran los siguientes componentes: la generación de la población inicial, la selección de los padres, el cruce, la mutación y el reemplazo. Como esta tesis desarrolla e implementa algoritmos genéticos para la solución de los problemas de programación de tareas con recursos restringidos, en el Capítulo 3 se profundiza sobre el funcionamiento de estos algoritmos y se describen a detalle cada uno de los operadores.

Los algoritmos genéticos han sido ampliamente usados para los problemas de programación de tareas, incluyendo el RCPSP en sus dos versiones. Uno de los pioneros fue Ozdamar (1999) quien propuso dos versiones de algoritmos genéticos, denominados: GA puro y GA híbrido, en el primero se usa una representación de lista de actividades y un SGS serial, en el segundo se usa una representación de valores clave y un SGS en paralelo. Los resultados reportados, muestran que el GA híbrido supera a los otros algoritmos considerados en la investigación. Hartmann (2002) fue de los primeros en aplicarla al RCPSP. El autor propuso un GA adaptativo que usaba una representación de lista de actividades, que adiciona un gen para determinar el SGS a usar como decodificador. Posteriormente, Alcaraz, Maroto y Ruiz (2003) propusieron un nuevo GA, cuya principal diferencia radica en la inclusión de un gen que determina la dirección del SGS, que puede ser hacia adelante o hacia atrás. Uno de los mejores algoritmos en su momento fue el propuesto por Mendes, Gonçalves y Resende (2009), el cual consistía en un GA que usa una representación de valor clave aleatoria y un SGS en paralelo modificado para que pueda construir soluciones activas. Tseng y Chen (2009) desarrollaron un GA en dos etapas, en la primera se genera una población factible y las mejores soluciones son agregadas en un conjunto de élite. En la segunda etapa, se realiza una búsqueda profunda sobre la región de soluciones definida por las soluciones contenidas en el grupo élite. Recientemente, Peteghem y Vanhoucke (2010) propusieron una versión bi-poblacional de un GA. La principal diferencia yace en el uso de dos poblaciones: una de ellas que contiene soluciones justificadas

a la izquierda y otra con soluciones justificadas a la derecha. Ellos adoptaron los operadores genéticos actuales, al igual que el SGS en serie, para ser usados en las dos poblaciones. Adicionalmente, su esquema generador de secuencias se complementa con un procedimiento de búsqueda local que mejora la solución a través de cambios entre los modos. Este algoritmo usa la representación de valores clave. Los resultados obtenidos son uno de los mejores reportaros en la literatura.

Enfriamiento simulado

Enfriamiento simulado o temple simulado (Simulated Annealing, SA) fue desarrollado por Kirkpatrick, Gelatt y Vecchi (1983). Este está basado en proceso químico del enfriamiento de materiales. Este nos describe que, por ejemplo, cuando un metal con una temperatura muy alta, se sumerge en agua fría, la temperatura promedio del metal va disminuyendo con el tiempo. Sin embargo, la temperatura molecular aumenta brevemente en pequeños intervalos, cada vez con menor temperatura. Esto se debe a que, al enfriarse, las partículas se juntan unas con otras provocando calor gracias a la energía cinética de las mismas, este calor se dispersa rápidamente y sigue enfriándose hasta que el calor producido por la colisión de las partículas nuevamente produce un breve calentamiento.

El temple simulado también ha sido implementado para la resolución del MRCPPS, por ejemplo, Józefowska y col. (2001), propusieron dos versiones de un SA, usando una lista de actividades y un SGS en serie. La primera versión con una función de penalidad y la otra sin ella. La finalidad de la función de penalidad es empeorar la función objetivo de soluciones que violen una restricción, en este caso soluciones que superen el uso máximo de recursos no renovables, para que tengan menor probabilidad de ser seleccionadas en la búsqueda. Bouleimen y Lecocq (2003) presentaron otra implementación de un SA, donde el vecindario de soluciones es generado usando dos etapas. En la primera se busca una solución factible respecto a los modos de ejecución y en la segunda se ejecutaba un proceso de mejora basado en cambios aleatorios de las actividades.

Respecto al RCPSP, Boctor (1996) fue uno de los pioneros en proponer un SA, este algoritmo opera sobre una representación de lista de actividades, generando nuevos vecindarios de soluciones mediante inserciones de actividades sobre la lista de actividades. Este operador de inserción es ampliamente usado en los algoritmos genéticos actuales como operador de mutación. Bouleimen y Lecocq (2003) desarrollaron otro procedimiento basado en el SA, el cual usa la inserción de Boctor para generar nuevos vecindarios de soluciones. Adicionalmente, usa múltiples cadenas de enfriamiento, donde el número de soluciones visitadas aumenta con cada cadena. El algoritmo se reinicia desde una solución inicial diferente en

cada cadena la búsqueda. Valls, Ballestín y Quintanilla (2005) también propusieron un SA, pero la investigación estaba enfocada en mostrar el impacto de varios algoritmos metaheurísticos al implementar métodos de mejora basados en pasadas hacia adelante y hacia atrás.

Búsqueda tabú

La búsqueda Tabú (Tabú Search, TS) fue introducida por Glover y Laguna (1997). Esta metaheurística es una de las pocas que no es inspirada en un fenómeno natural. Su principal característica es la inclusión de estructuras de memoria con el objetivo de dirigir la búsqueda, en otras palabras, trata de extraer información de iteraciones pasadas para tomar decisiones futuras. Su principal estructura de memoria es la denominada lista tabú, la cual almacena de forma finita las mejores soluciones históricamente para evitar volver a ellas en pasos posteriores de la búsqueda.

La primera búsqueda tabú para el MRCPSP fue propuesta por Nonobe y Baraki (2002). Esta opera con una representación de lista de actividades. Los vecindarios de solución se generan mediante cambios en los modos o bien mediante cambios sobre la lista de actividades. Los resultados muestran que le algoritmo propuesto supera al presentado por Józefowska y col. (2001), pero no logra superar al GA propuesto por Hartmann (2001).

Respecto al RCPSP, Thomas y Salhi (1998) fueron de los primeros en proponer una búsqueda tabú. El algoritmo usaba una representación real de las soluciones y tenía dos estructuras diferentes de vecindarios de soluciones, denominadas intercambio e inserción. Como estos vecindarios podían contener soluciones no factibles, ellos empelaron un procedimiento de reparación para hacerlas factibles. Posteriormente, Nonobe y Baraki (2002) propusieron un TS para una variante generalizada del RCPSP, la heurística usa una representación de lista de actividades, un SGS serial y un mecanismo específico de reducción de vecindario. Uno de los TS más relevantes es el propuesto por Artigues, Michelon y Reusser (2003), quienes diseñaron una técnica de inserción para ser usada en la búsqueda tabú. Esta regla está basada en un SGS en paralelo y la regla de prioridad de la peor holgura (worst case slack). En esencia, de manera iterativa se selecciona una actividad que es removida del problema, para luego ser insertada de manera que la extensión del *makespan* sea mínima.

Optimización por enjambre de partículas

La optimización por enjambre de partículas (Particle Swarm Optimization, PSO) fue propuesto por Kennedy y Eberhart (1995) y aunque inicialmente estos métodos fueron concebidos para construir modelos de conductas sociales, como el movimiento descrito por una bandada de aves o un banco de peces, luego fueron usados como técnicas de optimización. En esencia parte de una población inicial de partículas (soluciones) las cuales exploran el espacio de búsqueda mediante reglas de movimiento que tiene en cuenta la posición y la velocidad de las todas las partículas. En otras palabras, el movimiento de cada partícula se ve influenciado por el movimiento y la posición de aquellas partículas con mejor función de aptitud, así las partículas convergen rápidamente en las zonas del espacio de búsqueda con mejores soluciones.

La PSO ha sido implementada exitosamente para el RCPSP, por ejemplo Zhang, Tam y Li (2006) proponen una PSO que usa dos partículas para representar una solución. La primera basada en una representación de valores clave, y la segunda basada en una lista de modos. Aunque los resultados son competitivos, aún no superaban al GA propuesto por Hartmann (2001). Más adelante, Jarboui y col. (2008) propusieron la denominada optimización de enjambre de partículas combinatorias (combinatorial particle swarm optimization, CPSO). El cual es usado para generar una solución codificada en partículas que es factible respecto a modos de ejecución. Para luego, con los modos fijos, ser sometida a una búsqueda local para mejorar la secuencia de actividades. Basado en los resultados computaciones, se muestra que el CPSO alcanza uno de los mejores resultados.

Optimización de la colonia de hormigas

La optimización de la colonia de hormigas (Ant Colony Optimization, ACO) fue propuesta por Dorigo (1992). Al igual que los algoritmos genéticos, la ACO es una metaheurística poblacional que intenta reproducir el mecanismo utilizado por las hormigas para localizar el alimento e informar a la colonia donde este se encuentra. La ruta seguida por las hormigas tiene la característica de ser la de menor distancia desde su colonia hasta el alimento. La idea general consiste en que las hormigas exploran su territorio en busca de comida, aquellas que la encuentran dejan una feromona que las demás hormigas pueden percibir, algunas de ellas son atraídas a recorrer la misma ruta y dejar su propia feromona. Cuanta más feromona tenga una ruta, mayor probabilidad tendrá de que una hormiga la elija para recorrerla y dejar su feromona. Al final la ruta con mayor cantidad de feromona será la ruta de menor distancia.

Merkle, Middendorf y Schmeck (2002) fueron los primeros en usar una ACO en un RCPSP. Cada hormiga correspondía a una aplicación de un SGS en serie sobre una lista de actividades. Las actividades a ser elegidas en el esquema generador, eran seleccionadas basadas en la regla de prioridad LST, junto con la feromona que representa el efecto del aprendizaje de cada solución generada. Más recientemente, Li y Zhang (2013) propusieron una optimización de la colonia de hormigas, la cual usa dos niveles de feromonas para hacer la búsqueda, uno para los modos de ejecución y otro para la secuencia de actividades. Ellos definen una solución ACO, la cual es una adaptación de la representación de la lista de actividades y de modos. Adicionalmente, desarrollaron una adaptación del SGS en serie para generar una solución completa a partir de una solución ACO. Los resultados reportados muestran que el ACO alcanza soluciones competitivas, aunque no supera las alcanzadas por Jarboui y col. (2008).

Búsqueda dispersa

La búsqueda dispersa (Scatter Search, SS) fue propuesta inicialmente por Glover (1977), aunque no fue sino hasta la década de los 90s cuando empezó a aplicarse y difundirse. Básicamente, este método está basado en la idea de usar la información sobre la calidad de dos o más soluciones para crear una nueva mejor solución mediante la combinación de soluciones. A pesar de ser considerado un método poblacional, tiene grandes diferencias respecto a ellos, por ejemplo, los algoritmos genéticos combinan soluciones de manera aleatoria en una gran población, mientras que la SS combina soluciones de forma sistemática sobre un conjunto pequeño de población de referencia.

La búsqueda dispersa ha sido usada para el RCPSP con resultados muy competitivos, por ejemplo Mahdi Mobini y col. (2008) desarrollaron un algoritmo de búsqueda dispersa para el RCPSP que operan sobre una lista de actividades y emplean el método de mejora FBI. Los operadores están basados en un cruce básico de dos puntos y proponen dos nuevos operadores, el re-encadenamiento de trayectorias y un operador basado en permutaciones. Por su parte Paraskevopoulos, Tarantilis y Ioannou (2012) propusieron una nueva representación de soluciones llamada lista de eventos (event-list), la cual agrupa las actividades según dos eventos: tiempos de inicio o finalización simultáneos. Además, proponen un método de solución llamado SAILS, que consiste en una fase inicial de construcción de soluciones y una fase de búsqueda dispersa para probar el rendimiento de la nueva representación. Basado experimentación expuesta, este algoritmo alcanza uno de los mejores resultados reportados hasta ahora.

Algoritmos híbridos

La idea fundamental de los algoritmos híbridos es la posibilidad de combinar o unir varias metaheurísticas en un solo método de manera que se complementen entre sí. Es decir, aprovechar las ventajas o especialidades de un algoritmo y compensar sus deficiencias con otro que se desempeñe mejor en ese aspecto. Por ejemplo, se puede mezclar la gran capacidad de exploración de un algoritmo genético, con la buena explotación de un enfriamiento simulado. Siguiendo la definición de Talbi (2002), existen tres principios básicos para generar un algoritmo híbrido: el primero es sustituir un procedimiento no deseable de una metaheurística por el de otro que se especialice en ese tipo de procedimientos. El segundo es implementar dos o más metaheurísticas en secuencia, una tras otra. Por último, es implementar dos o más metaheurísticas para que trabajen en paralelo.

El primer algoritmo híbrido para resolver el RCPSP fue propuesto por Valls, Quintanilla y Ballestín (2003), el cual usa una representación adaptada de valor clave e integra un TS dentro de un algoritmo poblacional, en el cual se consideran dos fases: en la primera, se definen dos movimientos aleatorios de búsqueda, basados en la información de la red respecto a las holguras de las actividades; en la segunda fase, se realiza una búsqueda basada en métodos de muestreo sobre la mejor solución obtenida en la fase previa. De manera similar, Kochetov y Stolyar (2003) desarrollaron un GA basado en el re-encadenamiento de trayectorias (path relinking, PR), que incorpora un TS con vecindarios variables. La evolución se hace eligiendo dos soluciones y construyendo el camino de las soluciones que enlazan las soluciones seleccionadas. La mejor solución del camino es mejorada mediante un TS. Luego, la mejor solución encontrada por el TS es adicionada a la población y la peor es eliminada. Este método obtuvo uno de los mejores resultados reportados en la literatura.

Tseng y Chen (2006) proponen un método híbrido denominada ANGEL, que está compuesto por un ACO, un GA y una búsqueda local. El ACO es usado para generar la población inicial, luego el GA mejora dicha población. Mientras el GA hace la búsqueda, cuando se encuentra una mejor solución, las feromonas del ACO se actualizan con esa nueva información. Finalmente, el ACO busca nuevamente, pero con los nuevos valores de las feromonas. Valls, Ballestin y Quintanilla (2008) presentaron otro algoritmo híbrido basado en un método poblacional y una búsqueda local que opera sobre una lista de actividades.

Dentro de las investigaciones más recientes podemos encontrar el algoritmo ACOSS propuesto por Chen y col. (2010). Este consiste en la combinación de ACO y SS que usan la lista de actividades como representación de soluciones. De manera similar al algoritmo ANGEL, el ACO se usa al inicio y al final del algoritmo,

mientras que el SS es usado para mejorar la población generada por el ACO. En cada iteración la feromona es actualizada para realizar la segunda búsqueda mediante el ACO.

A pesar de no ser tan común, otras investigaciones también se han centrado en mezclar enfoques exactos y metaheurísticos. Esta idea se puede plantear de diversas maneras, implementando un método metaheurístico que gobierna la búsqueda y se usa un método exacto para resolver de manera iterativa sub-problemas (generalmente problemas relajados del original), por otro lado, también existen trabajos donde se ha usado un método exacto para coordinar la búsqueda mediante el uso de ciertas heurísticas. Por ejemplo, Morillo Torres, Moreno Velásquez y Diaz Serna (2015) propusieron un algoritmo de ramificación y acotamiento (enfoque exacto), que realizaba una búsqueda sistemática del espacio de soluciones, pero con el objetivo de no recorrer explícitamente todas las soluciones, se realizaban cortes en las ramas basados en la cantidad de conjuntos recorridos por cada nodo del árbol y en el número de iteraciones disponibles (enfoque heurístico). De esta forma, cuando el algoritmo haya usado la mitad de iteraciones, se debe haber recorrido la mitad de conjuntos en cada nodo del árbol.

En conclusión, los métodos basados en búsquedas poblacionales, al igual que los métodos híbridos son los que alcanzan los mejores resultados al tratar de resolver el RCPSP tanto en su versión uni-modal como multi-modal.

2.6.5 Procedimientos de mejora local

Los procedimientos de mejora local, generalmente, son heurísticas que parten de una solución factible y mediante reglas de movimiento, generan un vecindario de soluciones para quedarse con la de mejor valor de la función objetivo. Es decir, permite mejorar una solución dentro de un conjunto de soluciones con características similares. Por lo tanto, no generan una secuencia por sí mismas. Usualmente se utilizan para mejorar una solución encontrada previamente a través de una metaheurística.

Las mejoras locales para el RCPSP pueden ser clasificadas en 2 grupos, basándose en el objetivo a mejorar.

- **Búsquedas que mejoran la factibilidad.** Estas mejoras locales son más frecuentes en el MRCPSP, dado que al estar presentes los recursos no renovables, una solución construida a partir de otra o bien mediante una heurística, puede ser no factible. Esto se debe a que los algoritmos para construir soluciones (como los SGS), iterativamente generan soluciones parciales desde las cuales no se puede determinar si la solución será factible hasta que se viola

una restricción (en soluciones parciales subsecuentes) o bien se termina de construir la solución completa. Por lo tanto, estos métodos de búsqueda local suelen cambiar los modos de ejecución hasta encontrar una solución factible.

- **Búsquedas que mejoran el *makespan*.** Son mejoras locales enfocadas en reducir el tiempo de ejecución del proyecto de una solución dada. Por ejemplo, la mejora en un paso propuesta por Hartmann (2001), donde se recorren $i = (1, .., n)$ actividades una vez, comprobando si se puede reducir el tiempo de finalización de la actividad i sin cambiar el modo de ejecución ni aumentar el tiempo de finalización de otra actividad y sin violar las restricciones de recursos. La reducción se puede lograr cambiado de modo la actividad i y comparar si se puede programar antes de su tiempo de inicio.

  La mejora local más relevante en esta clasificación es la denominada mejora hacia adelante y hacia atrás (Forward-backward improvement, FBI), propuesta por Tormos y Lova (2001). También llamada en la literatura como el método de la doble justificación. Este método consiste aplicar un método de dos etapas. En la primera etapa (pasada hacia atrás) todas las actividades son programadas lo más tarde posible sin afectar al actual *makespan*. Luego, en la segunda etapa (pasada hacia adelante) todas las actividades son programadas lo más temprano posible. Así, algunos `"espacios"` vacíos pueden ser generados en la pasada hacia atrás y luego son cubiertos en la pasada hacia adelante, logrando, en algunas ocasiones, un mejor *makespan*. Este método también fue extendido al caso multi-modal (Lova y col. 2009). En la figura 2.8 se muestra un ejemplo aplicando la mejora FBI a una solución factible del problema presentado en la figura 2.1.

  Más recientemente, Fahmy, Hassan y Bassioni (2014) propusieron una nueva técnica de justificación denominada (stacking justification, SJ). Su principal diferencia entre la técnica de doble justificación es que esta última se basa en el orden de los tiempos de inicio y fin, respetando las restricciones de recursos, mientras que la SJ se basa en respetar primero las restricciones de recursos, programando actividades que permitan menores brechas en el uso de recursos. Los resultados muestran que el uso de las dos justificaciones (la tradicional y la que propusieron los autores) mejora significativamente la calidad de las soluciones.

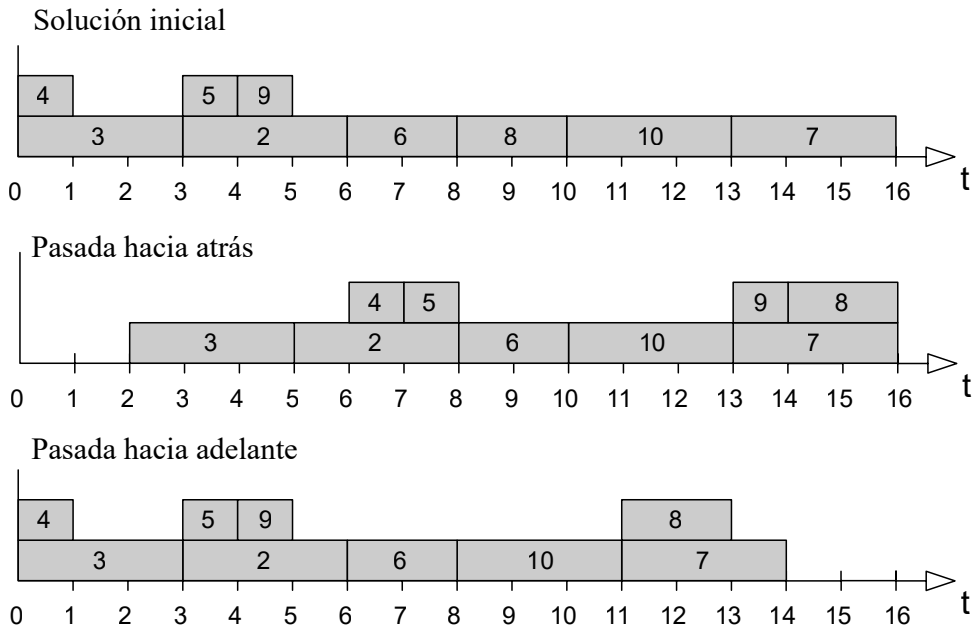

Figura 2.8: Ejemplo de la aplicación del método FBI.

2.7 Librerías de prueba

Con el desarrollo de nuevas metaheurísticas y nuevos métodos de solución exactos para resolver el RCPSP y el MRCPSP, surge la necesidad de evaluar el desempeño de estos métodos. Puesto que, dada su complejidad, en la mayoría de casos no se puede encontrar su solución óptima. Inicialmente, muchos autores creaban sus propios conjuntos de casos de prueba. Sin embargo, esto generaba diversos inconvenientes en los resultados computacionales, entre ellos se destacan:

- La disponibilidad de los conjuntos de prueba, puesto que la mayoría de autores usaban conjuntos diferentes, y como se describió en la sección 2.3, aunque los problemas contengan el mismo número de actividades, pueden tener complejidades muy diferentes.
- La generación de los conjuntos de prueba no se realizaba bajo un diseño controlado con parámetros definidos.

- No se tenía garantía de que los problemas generados tuvieran distintos niveles de complejidad, pues un método de resolución debe ser capaz de encontrar soluciones para problemas simples y complejos.

Patterson (1984) recopiló 110 problemas de diversos autores. Por varios años, estos casos de prueba fueron denominados `"los problemas de Patterson"` y fueron usados como una librería de prueba estándar para medir el desempeño de los algoritmos. Sin embargo, dicho conjunto también tenía algunas de las desventajas mencionadas anteriormente, puesto que los problemas provenían de diversas fuentes. Incluso, Herroelen, De Reyck y Demeulemeester (1998) mostraron que los problemas eran resolubles en muy poco tiempo por procedimientos exactos.

Boctor (1993) propuso una librería de casos de prueba para el MRCPSP. Este consiste de 2 conjuntos de 120 problemas, el primer conjunto con 50 actividades y el segundo con 100 actividades. Cada proyecto tiene $k \in (1, 2, 4)$ tipos de recursos renovables y $m \in (1, 2, 4)$ modos de ejecución. Esta librería aún es usada como punto de referencia. Sin embargo, actuales estudios (Van Peteghem y Vanhoucke 2014) muestran como principales desventajas que el uso de recursos está débilmente restringido. Además, varios problemas tienen una red de precedencias principalmente en serie.

Actualmente, la librería más usada es la denominada PSPLIB (Project Scheduling Problem Library) propuesta por Kolisch y Sprecher (1996), la cual será descrita a continuación.

2.7.1 La librería PSPLIB

Kolisch y Sprecher (1996) propusieron un algoritmo generador de casos de prueba de una clase general de problemas de secuenciación para el RCPSP uni-modal y multi-modal. Los casos de prueba son generados mediante un diseño de experimentos factorial basado en dos conjuntos de parámetros: uno fijo y otro variable.

El primer conjunto de parámetros, los fijos, son aquellos que se mantienen estáticos durante la generación de todos los conjuntos de casos de prueba. Estos parámetros se listan a continuación:

- Número de actividades.
- Número de modos de ejecución.
- Número de recursos renovables.
- Disponibilidad máxima de cada recurso renovable.

- Número de recursos no renovables.
- Disponibilidad máxima de cada recurso no renovable.
- Número de sucesores de la actividad ficticia de inicio.
- Número de sucesores y predecesores de las actividades ficticias.
- Duración de las actividades.

El segundo conjunto de parámetros, los variables, son aquellos cambian de problema a problema. Estos parámetros son los que caracterizan la complejidad del problema y están basados en los índices de complejidad explicados en la sección 2.3. El conjunto de parámetros variables está compuesto por los índices de complejidad de la red (NC), factor de recursos (RF), e intensidad de recursos (RS). Los parámetros que incluyan el uso de recursos, tienen un subíndice R o NR, refiriéndose a renovable y no renovable, respectivamente.

A continuación, se describen los parámetros y los conjuntos de casos de prueba propuestas en la PSPLIB, para el RCPSP y el MRCPSP. Las tablas presentadas para cada tipo de problema usan la notación de la sección 2.2. Adicionalmente, se define a NK^{ρ} y NK^{ν}, como el número de recursos renovables y no renovables respectivamente, usados por una actividad.

- **PSPLIB para el RCPSP.**

  Esta librería está compuesta por 4 conjuntos de problemas, denominados $j30$, $j60$, $j90$, y $j120$, los cuales tienen 30, 60, 90, y 120 actividades respectivamente. Los parámetros usados se muestran en la tabla 2.5 y en la tabla 2.6. Cada combinación de parámetros variables (tabla 2.6) genera 10 casos de prueba así, los conjuntos 30, 60, y 90 están conformados por 48 subconjuntos de 10 problemas cada uno, para un total de 480 casos de prueba por conjunto. El conjunto $j120$ tiene un total de 60 combinaciones de parámetros, cada uno con 10 casos de prueba generados, para un total de 600.

	n	m	d_i	K^{ρ}	r^{ρ}_{ik}	NK^{ρ}	K^{ν}	r^{ν}_{imk}	NK^{ν}	Suce	Prede
min	Cte	1	1	4	1	1	0	0	0	1	1
max	Cte	1	10	4	10	4	0	0	0	3	3

Tabla 2.5: Parámetros fijos para la generación de casos de instancias de la PSPLIB para el RCPSP.

  Para casos de prueba que se conozcan las soluciones óptimas, se puede calcular la desviación porcentual promedio respecto a la solución óptima

Factor	$j30, j60, j90$				$j120$				
NC	1,5	1,80	2,10		1,5	1,8	2,10		
RF_R	0,25	0,50	0,75	1,00	0,25	0,50	0,75	1,00	
RS_R	0,20	0,50	0,70	1,00	0,1	0,2	0,3	0,4	0,5

Tabla 2.6: Parámetros variables para la generación de casos de instancias de la PSPLIB para el RCPSP.

($\%DOS$) para cada problema y luego hacer un promedio de todo el conjunto ($\%ADOS$). Así, sea R_h la solución obtenida por un método metaheurístico y O^* sea el valor óptimo del caso de prueba, las desviaciones pueden ser calculadas mediante las Ecuaciones 2.21 y 2.22. El único conjunto del que se conocen las soluciones óptimas es para el $j30$.

$$\%DOS_w = \frac{R_h - O^*}{O^*} * 100 \tag{2.21}$$

$$\%ADOS = \frac{1}{nP} \sum_{w=1}^{nP} \%DOS_w \tag{2.22}$$

Para los conjuntos donde las soluciones optimas no son conocidas ($j60$, $j90$, y $j120$), se calcula la desviación respecto a la longitud de la ruta crítica ($\%DLB$). Posteriormente, se calcula el valor medio de las desviaciones ($\%ADLB$). Si $LB0$ es la ruta crítica del problema evaluado, la $\%DLB$ se define en la Ecuación 2.23 y el $\%ADLB$ en la Ecuación 2.24. A pesar de que esta desviación puede ser calculada con otra cota inferior, la mayoría de autores usan la ruta crítica, la principal razón es que así, se evita que los resultados puedan cambiar de autor a autor dependiendo su cota inferior.

$$\%DLB_w = \frac{R_h - LB0}{LB0} * 100 \tag{2.23}$$

$$\%ADLB = \frac{1}{nP} \sum_{w=1}^{nP} \%DLB_w \tag{2.24}$$

- **PSPBLIB para el MRCPSP.**

  La librería PSPLIB para el MRCPSP está compuesta por 7 conjuntos $j10$, $j12$, $j14$, $j16$, $j18$, $j20$, y $j30$, cada uno con 10, 12, 14, 16, 18, 20, y 30

actividades respectivamente. Los valores de los parámetros usados para generar las secuencias se muestran en la tabla 2.7 (para los fijos) y en la tabla 2.8 (para los variables). En este caso se cuentan con 64 combinaciones de parámetros variables, de los cuales se generaron 10 casos de prueba para cada conjunto, para un total de 640 problemas por conjunto. Sin embargo, aproximadamente 15 % de los problemas de cada conjunto no tienen solución factible, por tanto, no son considerados para resolverlos.

	n	m	d_i	K^ρ	r^ρ_{ik}	NK^ρ	K^ν	r^ν_{imk}	NK^ν	Suce	Prede
min	Cte	3	1	2	1	1	2	1	1	1	1
max	Cte	3	10	2	10	2	2	10	2	3	3

Tabla 2.7: Parámetros fijos para la generación de casos de instancias de la PSPLIB para el MRCPSP.

Factor	$j10$				$j12, j14, j16, j18, j20, j30$			
NC	1,8				1,8			
RF_R		0,25	1,00			0,50	1,00	
RS_R	0,20	0,50	0,70	1,00	0,25	0,50	0,75	1,00
RF_{NR}		0,50	1,00			0,25	0,75	1,00
RS_{NR}	0,20	0,50	0,70	1,00		0,25	0,75	1,00

Tabla 2.8: Parámetros variables para la generación de casos de instancias de la PSPLIB para el MRCPSP.

Para reportar los resultados, también se usan las Ecuaciones 2.21, 2.22, 2.23, y 2.24. En este caso se conocen las soluciones óptimas para todos los conjuntos excepto el conjunto $j30$.

La PSPLIB está disponible en: `http://www.om-db.wi.tum.de/psplib/main.html`

2.8 Optimización energética en la programación de proyectos

Como lo afirma WWF World Wildlife Fund (2010), la economía global, dominada por un mundo industrializado, ya consume más recursos naturales de los que ecológicamente se pueden soportar. La transformación del nivel de vida de regiones rurales a un nivel de vida industrializado ha diezmado el ecosistema de manera permanente. La necesidad de interrumpir la relación entre el crecimiento económico y el consumo de recursos surge como un desafío esencial para la humanidad. La industria es uno de los mayores consumidores de energía, acaparando más de un

tercio de la energía producida en el mundo. Actualmente, gracias a la conciencia medioambiental y a las regulaciones de los países, gran parte de la investigación está enfocada al desarrollo de nuevos métodos energéticamente eficientes para los problemas del sector industrial, y por ende a los problemas de programación de proyectos.

Debido al gran interés referente a la optimización energética en dicho campo, en los últimos años se ha incrementado el número de publicaciones en áreas multidisciplinares, que incluyen campos como la química, la ingeniería, las matemáticas, ciencias de la computación, entre otras. Esto ha generado un campo de investigación muy heterogéneo, con definiciones ambiguas. Además, suele ser difícil clasificar las investigaciones y evaluar sus objetivos. Sin embargo, se sabe que existen diferentes estrategias para realizar la optimización energética en los procesos de manufactura. Salonitis y Ball (2013) proponen que dichas estrategias pueden ser clasificadas en dos niveles: optimización a nivel de máquina y optimización a nivel de sistema.

2.8.1 Optimización a nivel de máquina

El primer nivel hace referencia al mejoramiento del proceso teniendo en cuenta el diseño de la máquina, el cual puede ser logrado mediante un punto de vista tecnológico. En este grupo se encuentran, por ejemplo, las investigaciones realizadas por Aggarwal y col. (2008) donde se optimizan algunas características de la técnica de corte para el maquinado. Otros autores están enfocados en mejorar la gestión de los refrigerantes residuales de máquinas (Kobya y col. 2008). Li, Yan y Xing (2013) presentan un modelo de consumo de energía como una función dependiente de la tasa de remoción de materia y la velocidad del motor para máquinas de fresado, o bien la investigación de Seow y Rahimifard (2011), quienes proponen un nuevo esquema para modelar y estimar el consumo energético total requerido para generar una unidad de producto. Para una amplia revisión sobre la optimización a nivel de máquina se remite al lector al trabajo de Moradnazhad y Unver (2016).

2.8.2 Optimización a nivel de sistema de manufactura

En el segundo nivel, la optimización a nivel de sistema de manufactura, hace hincapié en la optimización energética a través de la gestión de actividades y la asignación de recursos con el objetivo de minimizar los requerimientos energéticos. De hecho, estudios como los de He y Liu (2010) y Duflou y col. (2012) han mostrado que la optimización energética a este nivel es de gran importancia y ofrece diversas oportunidades de mejora. El presente trabajo está enfocado en el último nivel de optimización.

Gahm y col. (2016), además de realizar una completa revisión del estado del arte referente a la energía y la programación de tareas, proponen un marco de investigación denominado Energy-efficient scheduling (EES) para caracterizar las investigaciones en este nivel. Los autores en su publicación, proponen este marco de investigación para clasificar y diferenciar los principales elementos de la optimización energética en la programación de proyectos y así, poder identificar nuevas direcciones de investigación. Para describir el marco de investigación, primero se debe hacer algunas definiciones.

Teniendo en cuenta las implicaciones del uso de energía en las industrias, la optimización a nivel de sistema no solo considera el proceso de producción como fuente de mejora, sino también a toda la cadena de transformación de energía, es decir, también incluye los procesos y sistemas que suministran la energía. De esta manera, se definen las fuentes de energía primaria (primary energy sources, PES), que incluyen los combustibles fósiles o minerales y la energía eólica o solar. La energía producida es transportada y convertida por los proveedores de energía en las fuentes de energía secundaria (secondary energy sources, SES), como por ejemplo, electricidad o combustibles refinados. Una vez las SES son transferidas al usuario final (las empresas de producción) se denominan fuentes de energía final (final enregy sources, FES). Dentro de las empresas de producción la energía usada para realizar las actividades o procesos de transformación de bienes se denomina fuentes de energía aplicada (applied energy sources, AES), como la corriente eléctrica o el gas natural. Si bien la programación de proyectos afecta directamente al final de esta cadena, la secuenciación de actividades y la asignación de energía para su realización pueden afectar la eficiencia energética del sistema. Es decir, no solo afecta a la demanda de AES sino también a la demanda de FES y SES. Por ejemplo, una disminución en los picos de demanda de AES conlleva a una menor demanda de SES.

Además, en la literatura sobre la programación de proyectos bajo un enfoque de eficiencia energética, se consideran dos principales agentes y tres sistemas: el primer agente es la compañía manufacturera (el usuario de energía) que usa AES para producir bienes. El segundo agente hace referencia a los proveedores de energía, quienes ofrecen uno o más SES en el mercado. Por otra parte, los tres sistemas son el sistema de producción (production system, PS), el sistema de conversión interna (internal conversion system, iCS) y el sistema de conversión externa (external conversión system, eCS). Los dos primeros son controlados por la compañía manufacturera, mientras que el último es controlado por el proveedor.

Basándose en estas definiciones, el marco de investigación para la optimización energética en la programación de proyectos se divide en tres dimensiones (Gahm y col. 2016):

- **Cobertura energética.** Esta dimensión especifica las posiciones dentro de la cadena de conversión de energía en las que los efectos de la programación de proyectos conducen a una mejora en la eficiencia energética. Estas posiciones pueden ser los tres sistemas anteriormente mencionados: sistema de producción, sistema de conversión interna o sistema de conversión externa.

  Dentro del PS, la programación de actividades directamente reduce la demanda de AES. Esta reducción puede lograrse mediante el apagado de máquinas que se encuentran inactivas (idle machines), por la secuenciación de órdenes de trabajo para evitar picos de energía, asignando trabajos a las máquinas teniendo en cuenta sus necesidades energéticas, ajustando la velocidad de procesamiento de las máquinas, o explorando el potencial de recuperación de energía (por ejemplo, la reutilización de material).

  Por otra parte, las diferentes estrategias del iCS y el eCS no afectan a la cantidad total de energía demandada (como si lo hace el PS), pero sí influyen en el curso o disponibilidad temporal de la demanda de energía tanto para AES o FES. Es decir, no se afecta la cantidad de energía del proyecto, sino la disponibilidad del mismo en el tiempo (se explicará en mayor detalle en la siguiente dimensión). Sin embargo, una clara diferencia entre los sistemas iCS y eCS es que el primero es el sistema de conversión de energía controlado por la empresa productora y el segundo es controlado por el proveedor de energía.

- **Proveedor de energía.** Esta dimensión incluye las características del aprovisionamiento del sistema productivo de energía (FES y AES). Para esto se distinguen dos categorías: la infraestructura interna y los mecanismos de coordinación. La primera hace referencia a la infraestructura de la empresa productora y, por lo tanto, a las relaciones entre el PS y el iCS, es decir, los medios en los que la empresa provee de energía al sistema de producción para realizar la transformación de bienes. Por otra parte, los mecanismos de coordinación, hacen referencia principalmente a las estrategias relacionadas con los eCS, estas usan dos mecanismos de respuesta a la demanda, una impulsada por los precios y otra impulsada por eventos. Según Merkert y col. (2015), el mecanismo impulsado por los precios corresponde al cambio en el patrón normal de uso de energía, en respuesta a la variación del precio de la energía. El segundo mecanismo, busca recompensar a los usuarios de energía por ajustar su demanda en respuesta a ciertos eventos, como condiciones climáticas extremas o fallas en los equipos de generación.

- **Demanda de energía.** Esta dimensión describe las características de la demanda de energía usadas en los problemas de programación de proyectos

que incorporan el uso de energía como criterio de optimización. Estas características se agrupan en dos categorías: demanda de energía no procesada y demanda de energía procesada. La primera surge cuando se utiliza energía en el proceso de producción, pero no se agrega ningún valor al producto durante ese periodo. Por ejemplo, máquinas encendidas pero no en funcionamiento, máquinas en preparación o hibernación. La segunda categoría hace referencia al uso de energía de los procesos que realmente transforman la materia prima en los productos deseados.

De esta forma, las investigaciones sobre la optimización energética en la programación de proyectos pueden definirse dentro de estas dimensiones, ya que cada una caracteriza una parte del sistema energético. La primera dimensión especifica el sistema que se desea optimizar, la segunda dimensión describe las características de los proveedores de energía y la tercera describe las características de la energía demandada.

Dentro de las principales aportaciones podemos encontrar el trabajo de Mouzon, Yildirim y Twomey (2007), quienes desarrollaron un modelo multi-objetivo para minimizar el consumo energético y el *makespan*, para un problema de programación con una sola máquina. Ellos encontraron que era posible ahorrar hasta un 80 % de la energía cuando las máquinas *cuello de botella* eran apagadas hasta que eran necesitadas durante el periodo de llegada del siguiente trabajo. Fang y col. (2011) desarrollaron un nuevo modelo de programación entera mixta para resolver un *flow shop problem* en el que se minimiza la carga máxima de energía permitida, el *makespan* y la huella de carbono. Bruzzone y col. (2012) propusieron la integración de un modelo matemático y un modelo de planeación y programación avanzado (Advanced Planning and Scheduling) para modificar una programación dada en un *flexible flow shop problem* con el objetivo de reprogramarla teniendo en cuenta los picos máximos de energía. Artigues, Lopez y Haït (2013) analizaron un caso industrial de estudio que les condujo a proponer un problema general llamado *the Energy Scheduling Problem*. Este consiste en programar un conjunto de actividades, cada una con una fecha de llegada y fecha de entrega, que demandan una cantidad total de energía para su realización (E_i). Las actividades tienen que ser procesadas usando un recurso energético restringido (P) en una cantidad que puede ser diferente en cada periodo de tiempo. Además, cada actividad tiene un valor mínimo y máximo de uso de ese recurso energético (p_i^{min} y p_i^{max}). Así, por ejemplo, si la energía total requerida por una actividad i es $E_i = 12$, el límite de energía por periodos es $P = 5$ y $p_i^{min} = 1$, $p_i^{max} = 5$, la activad i podría realizarse usando 5 unidades de energía por dos periodos y usando 2 unidades de energía en un periodo. Finalmente, el objetivo es encontrar los tiempos de inicio y finalización de cada actividad.

De igual manera, otros aportes están relacionados a la unificación de los sistemas que componen el proceso productivo, por ejemplo, Agha y col. (2010) proponen un enfoque integrado que combina la programación de tareas (producción) con la planificación operativa de los sistemas de suministro. Este tipo de investigaciones se clasifican en la primera y segunda dimensión de los iCS. Adicionalmente, se puede considerar un mecanismo para coordinar de manera más directa las demandas de las FES, como en el trabajo de Nolde y Morari (2010), en el cual se analiza el problema de programación de tareas cuando los proveedores de energía exigen a sus clientes que planifiquen su consumo de energía con antelación, con el objetivo de negociar diferentes planes de consumo con un precio más bajo de energía.

Más recientemente, Okubo y col. (2015) propusieron un modelo que consideraba el consumo de energía durante los tiempos de preparación (setup times) en el RCPSP mediante el uso de recursos parcialmente renovables. Los autores incluyen un conjunto de máquinas como un recurso especial que da origen a los modos. Zhang y col. (2016) propusieron una formulación matemática para minimizar el consumo de energía en un sistema de maquinado integrando planeación y programación de proyectos. Aquel modelo considera un conjunto de actividades con varios planes alterativos de procesamiento y un conjunto de máquinas. Cada actividad es procesada acorde a una secuencia predeterminada del plan de procesamiento. Así, el objetivo es encontrar el plan de procesos óptimo y la asignación de cada máquina para cada actividad. Li, Liu y Zhou (2016) propusieron un modelo matemático multi-objetivo para el *permutation flow shop scheduling problem* con el objetivo de minimizar de manera simultánea el tiempo de flujo total y el consumo de energía. Dada su complejidad *NP-Hard*, para su solución fue necesario adaptar el algoritmo *non-dominated sorting genetic algorithm II*. Estos trabajos son ejemplos de implementaciones de un enfoque EES principalmente sobre la primera dimensión.

Por otra parte, la propuesta de Luo y col. (2013), aborda un *hybrid flow shop*, el cual considera no solo la eficiencia en la producción sino también el costo de la energía, la cual está sujeta a variaciones en el precio dependiendo de la hora en la que se usa. Este es un ejemplo de la programación en dos dimensiones, tanto en la dimensión de cobertura energética como en la dimensión de proveedor de energía.

Todos estos trabajos tienen características específicas sobre la forma en que se considera el uso de energía descritas por la tercera dimensión, por ejemplo, si el consumo de energía está ligado a la máquina que realiza la operación o bien está definido por la actividad a ejecutar.

En el análisis de la literatura realizada por Gahm y col. (2016) sobre la optimización energética en los problemas de programación de proyectos, los autores

recopilaron más de 85 publicaciones de las últimas décadas hasta la actualidad y las categorizaron dentro del marco de investigación explicado anteriormente. Dentro de los trabajos incluidos, se pueden encontrar una gran diversidad de problemas como máquinas paralelas, *flow shop*, *job shop*, *hoist scheduling*, entre otros. Sin embargo, teniendo en cuenta que el área de estudio es novedosa, el RCPSP abordado bajo un enfoque de eficiencia energética aún no ha sido estudiado en profundidad. Los autores, además de demostrar con una recopilación de casos prácticos el impacto de los EES en la industria, concluyen reafirmando la importancia de incursionar en otros problemas de programación, así como desarrollar nuevas librerías de prueba para poder evaluar los nuevos métodos que se desarrollen en el futuro.

De este modo, aunque la eficiencia de la producción es esencial en los problemas de secuenciación, de ninguna manera debe ser el único factor a considerar en la programación de proyectos. En los últimos años se ha reconocido cada vez más que el desarrollo económico sin consideraciones ambientales puede causar daños irreversibles al medio ambiente en todo el mundo. Por lo tanto, el concepto de asignación y programación de recursos de manera energéticamente eficiente, es vital para lograr el crecimiento económico y reducir al mismo tiempo los impactos negativos al ambiente. Así, dada la necesidad del desarrollo de nuevos enfoques sobre los problemas de programación de proyectos desde un punto de vista sostenible, en esta tesis se propone una extensión del RCPSP, denominada MRCPSP-ENERGY, el cual considera el problema de secuenciar actividades y asignar recursos para finalizar un proyecto teniendo en cuenta el consumo de energía. Esta extensión incluye diversos modos de ejecución de las actividades que son originados por los distintos consumos de energía, y cuyo objetivo está basado en maximizar la eficiencia energética. Además, teniendo en cuenta que no existen librerías estándares que incluyan la minimización del *makespan* y el consumo de energía para el RCPSP, se propone una librería de casos de prueba para realizar comparaciones en el desempeño de nuevos métodos de solución.

2.9 Conclusiones

En este capítulo se ha realizado una revisión del estado del arte acerca de los principales métodos de solución para el RCPSP y el MRCPSP. Incluyendo definiciones, conceptos básicos, formulaciones matemáticas y librerías de casos de prueba. Adicionalmente, se ha revisado las aportaciones más importantes en el ámbito de la optimización energética en la programación de proyectos.

En el RCPSP se busca determinar la secuencia de actividades de tal manera que se minimice el tiempo de ejecución total del proyecto (*makespan*). Respetando

dos restricciones: una restricción de precedencias y una restricción sobre el uso de recursos renovables. En el MRCPSP se tienen dos conjuntos de decisiones a tomar, el modo de ejecución de cada actividad y la secuencia de actividades a programar, con el objetivo de minimizar el *makespan*, bajo las restricciones de precedencia y las restricciones de uso de recursos. En el MRCPSP es usual que se consideren tanto recursos renovables como no renovables. Los dos problemas abordados tienen una complejidad NP-Hard, con lo cual los métodos exactos no pueden ser usados para la resolución en un tiempo razonable. Adicionalmente, se ha mostrado que el interés en el área de programación de actividades es vigente y de gran aplicabilidad a problemas reales.

Para ambos problemas, los métodos metaheurísticos, específicamente los evolutivos y los híbridos poblacionales, son los que mejores resultados han reportado en la literatura. Estos métodos usan procedimientos heurísticos para generar soluciones, denominados esquemas generadores de secuencias (SGS) y operan sobre una abstracción de las soluciones reales llamadas representaciones. Existen principalmente dos SGS: el serial y el paralelo, que son heurísticas claves en el desarrollo de una metaheurísitca. Dentro de las representaciones se pueden distinguir principalmente dos: la representación de lista de actividades y la representación de valor clave, las cuales han alcanzado los mejores resultados. Sin embargo, se encontró que el rendimiento de los algoritmos puede verse afectado por el fenómeno de las soluciones redundantes. Este fenómeno ocurre cuando una solución completa puede ser codificada mediante diversas representaciones, diferentes entre sí pero que generan la misma solución completa. De esta manera, se evidencia una necesidad de alterativas de búsqueda que intenten evitar esta problemática.

Por otro lado, la creciente preocupación respecto a la contaminación generada por el desarrollo industrial en el medio ambiente y al alto consumo de recursos no renovables en los últimos años, ha generado una mayor atención a la incorporación aspectos de sostenibilidad en los problemas de programación de proyectos. Dado que la energía tiene un impacto sustancial en el desarrollo de empresas manufactureras sostenibles, se concluye que es de gran importancia aumentar la eficiencia energética en los procesos productivos. Para esto, existen dos principales enfoques reportados en la literatura: el primero, está centrado en el mejoramiento de las técnicas y las herramientas para lograr minimizar el consumo de energía en las operaciones. El segundo enfoque considera la optimización energética a nivel de sistema, consiguiendo mejorar el rendimiento de las operaciones y disminuir el consumo de energía mediante la gestión adecuada de los recursos y la asignación de las actividades.

Los trabajos en la optimización energética a nivel de sistema han mostrado que existe un campo de estudio en apogeo y una gran oportunidad para mejorar los

procesos, siendo competitivos y al mismo tiempo alcanzando soluciones sostenibles. Este enfoque se puede ampliar a un sistema aún más general, que incluya todo el sistema de conversión de energía, el cual no solo considera el proceso productivo, sino también el sistema de conversión externo de energía (desde los proveedores de energía) hasta el sistema de conversión interno controlado por la empresa. Teniendo en cuenta el gran potencial de la programación de proyectos bajo un enfoque de eficiencia energética que alcance soluciones sostenibles de menor coste y más productivas, se concluye que existe un gran campo de investigación relativamente nuevo para ser estudiado, en el que se requieren nuevos modelos de problemas de secuenciación que incorporen el uso de la energía dentro de los objetivos. De igual manera, surge la necesidad de generar casos de prueba para evaluar los nuevos procedimientos que se desarrollen.

Capítulo 3

La redundancia de soluciones durante el proceso de búsqueda y una propuesta evolutiva de mejora

En este capítulo se analizará el fenómeno de soluciones redundantes presente en las principales representaciones de soluciones usadas por las metaheurísitcas. Además de su definición, se evalúa su impacto sobre el proceso de búsqueda. Posteriormente, se propone detalladamente un algoritmo genético que incluye un nuevo operador de mutación, denominado mutación agresiva, el cual disminuye la formación de soluciones redundantes. Adicionalmente, se propone una decodificación alternativa al método de mejora hacia adelante y hacia atrás (forward-backward improvement, FBI) que puede obtener soluciones más compactas. Finalmente, se realiza una evaluación computacional del algoritmo propuesto en las librerías de prueba más relevantes en la literatura, alcanzando soluciones altamente competitivas.

3.1 Introducción

En el Capítulo 2 se hizo un repaso sobre los métodos metaheurísticos más relevantes para la resolución del RCPSP. También se expuso que existen dos elementos que son la piedra angular de las metaheurísticas que, de una forma u otra, todas comparten. Estos son las representaciones de soluciones y los esquemas generadores de secuencia.

Este capítulo está dedicado al estudio de una de las representaciones de soluciones más usadas en la literatura para resolver el RCPSP y el MRCPSP: la lista de actividades (activity list, AL) con el objetivo de analizar el fenómeno de redundancia de soluciones, evaluar su implicación y proponer alternativas para disminuir su efecto.

Dado que los métodos evolutivos son los que obtienen mejores resultados al resolver el RCPSP, se propone un algoritmo genético que incorpora un nuevo operador de mutación, diseñado para disminuir la formación de soluciones redundantes y promover la exploración en la búsqueda. Además, se propone una decodificación alternativa para el método de mejora local FBI, el cual puede encontrar soluciones más compactas. Para la evaluación de los métodos propuestos, se usan los casos de prueba provistos por la librería PSPLIB y se compara su desempeño con las mejores metaheurísticas de la literatura.

Este capítulo está estructurado de la siguiente forma: en la sección 3.2 se explica el fenómeno de redundancia de las soluciones. En la sección 3.3 se describe la propuesta evolutiva de mejora. En la sección 3.4 se presentan los resultados computacionales. Finalmente, en la sección 3.5 se exponen las conclusiones.

3.2 Análisis de soluciones redundantes

Como se concluyó en el Capítulo 2, las representaciones de soluciones juegan un papel fundamental en los diferentes métodos metaheurísticos. Esto se debe a que además de codificar una solución completa, es decir, transformar el conjunto de tiempos de inicio de cada actividad en una sola lista de elementos, permiten aplicar diversas reglas de movimiento para generar nuevos vecindarios de soluciones y así realizar la búsqueda.

Las representaciones de soluciones dominantes en la literatura son la lista de actividades (AL) y la lista de valores clave (RK) (véase subsección 2.6.1). Sin embargo, las representaciones mencionadas pueden dar lugar a soluciones redundantes en el proceso de búsqueda. Una solución redundante se define como dos

o más representaciones diferentes que originan (mediante el uso de un SGS) una misma solución real, es decir, que todas las actividades tengan igual tiempo de inicio. Como resultado, el espacio de búsqueda se incrementa considerablemente, con la gran desventaja de que la estructura de una solución codificada puede no contribuir con información valiosa que lleve a una mejor solución, e incluso puede usar parte del esfuerzo computacional (iteraciones) en generar soluciones redundantes haciendo la búsqueda ineficiente.

Para mostrar este fenómeno se usará el problema prototipo descrito en el Capítulo 2, el cual se encuentra representando en un grafo dirigido en la figura 2.1. Como ejemplo, tomaremos la lista de actividades $AL1 = \{1, 7, 3, 6, 2, 4, 5, 10, 8, 9, 11\}$. Esta es una lista factible en precedencias y el orden en el cual aparecen las actividades señala su prioridad. Así, por ejemplo, la actividad 7 tiene mayor prioridad a ser programada que la actividad 3. Ahora, supongamos que tras aplicar varias reglas de movimiento para generar una solución vecina, de $AL1$ obtenemos la lista $AL2 = \{1, 7, 3, 6, \underline{9, 4, 2, 5, 10, 8,}11\}$, donde las actividades subrayadas son las que difieren de la lista $AL1$. Asumiendo que el SGS usado es en serie y en una dirección hacia adelante, las dos listas de actividades $AL1$ y $AL2$ generan la misma solución, como se puede apreciar en la figura 3.1.

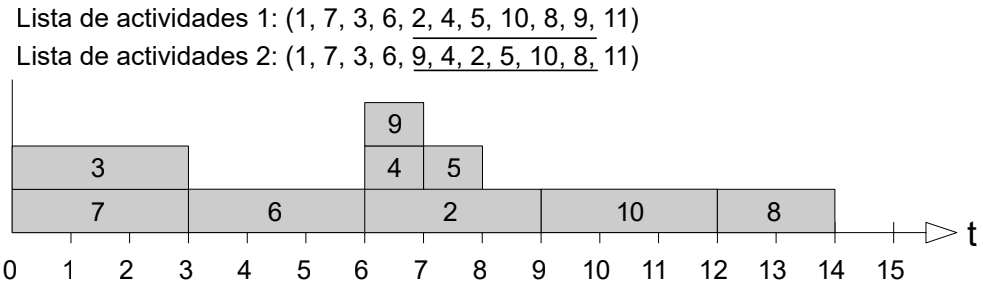

Figura 3.1: Soluciones redundantes de dos listas de actividades del problema presentado en la figura 2.1.

Es importante notar que la diferencia entre las listas $AL1$ y $AL2$ no es solo un intercambio de actividades consecutivas, sino una modificación con mayor alcance, por ejemplo, la actividad 9 pasa de la posición 10 en la lista a la posición 5. Aun así, las representaciones son redundantes. De esta forma, este fenómeno puede ocurrir en cualquier procedimiento de búsqueda sobre la lista de actividades. Estos procedimientos consisten en realizar la búsqueda sobre el número de permutaciones factibles que se puedan obtener de la lista de actividades. Identificar *a priori* el número de soluciones redundantes para un RCPSP no es posible, puesto que se necesitaría decodificar y comprobar todas las permutaciones factibles.

Las principales causas subyacentes por las cuales una solución puede dar lugar a varias representaciones redundantes en una lista de actividades son: cuando un subconjunto de actividades puede iniciar al mismo tiempo y cuando las restricciones del problema permiten una *flexibilidad* de programación a una actividad. Las dos causas son originadas en las características propias de una solución completa.

La primera causa se da cuando algunas actividades pueden iniciar en un mismo instante dado. Como ejemplo, tomaremos de nuevo la lista $AL1 = \{1, 7, 3, 6, 2, 4, 5, 10, 8, 9, 11\}$, como se puede observar en la solución real de la figura 3.1 las actividades 2 y 4 en el instante $t = 6$ pueden iniciar al mismo tiempo dado que no comparten precedencias y la suma de recursos no supera el límite ($\sum_{i=1}^{3} r_{ik} \leq b_k \; \forall \; k = \{1, 2, 3\}$). Por lo tanto, cualquier representación que contenga la secuencia parcial $\{1, 7, 3, 6, ...\}$ y cuyas siguientes actividades a programar sean la 2 o la 4, da origen a representaciones redundantes, pues el orden de prioridad $2, 4$ o $4, 2$ no cambia la solución real. Esto se puede extender a cualquier estructura que permita la ejecución simultánea en un instante dado para las actividades $2, 4$.

La segunda causa se da cuando la estructura de una solución parcial permite que algunas actividades tengan cierta *flexibilidad* para ser programadas. Esta característica les permite a las actividades usar una cantidad de recursos, la cual no puede ser utilizada por las demás actividades debido a las restricciones. Como consecuencia, estas actividades no se ven afectadas por la lista de prioridades a programar. Esta causa se puede apreciar también en el ejemplo de la figura 3.1. Tomemos nuevamente la lista $AL1 = \{1, 7, 3, 6, 2, 4, 5, 10, 8, 9, 11\}$, como sabemos que las prioridades de las actividades $4, 2$ y $2, 4$ son equivalentes en esta solución específica, podemos crear la lista $AL1^* = \{1, 7, 3, 6, 9, 2, 4, 5, 10, 8, 11\}$ cuyo único cambio respecto a lista $AL1$ es la inserción de la actividad 9 pasando de la posición 10 a la posición 5. Esta actividad, la número 9, es precisamente la que tiene la denominada *flexibilidad*, puesto que a partir de la secuencia parcial $\{1, 7, 3, 6, ...\}$ no importa en qué posición de la lista de prioridad se inserte ya que en todas ellas siempre se podrá programar en el instante $t = 6$ junto con las actividades 2 y 4. Esto se debe a características y circunstancias específicas de la solución y del problema. Por ejemplo, si la duración de la actividad 9 no fuese 1 unidad sino 2, programar primero la actividad 5 y luego la 9 sí generaría una solución diferente, o bien si la actividad 5 consumiera menos recursos o la actividad 9 consumiera más, no se generarían soluciones redundantes.

Por otro lado, estas causas también afectan a la representación de valor clave (RK) y pueden conducir a la generación de soluciones redundantes. Sin embargo, la representación de RK tiene dos causas adicionales que fomentan el fenómeno de redundancia: la escala de la representación y las restricciones de precedencia. La primera hace referencia a que se pueden obtener un número infinito de repre-

sentaciones diferentes que generan la misma estructura de prioridades. Esto se debe a que la representación de RK es un vector de valores del cual se pueden construir infinitos vectores que sean una combinación lineal de él. La segunda causa es debido a que esta representación no tiene en cuenta las restricciones de precedencias.

En la literatura existen pocos trabajos que abordan el problema de las soluciones redundantes. Debels y col. (2006) propusieron un algoritmo de búsqueda dispersa que incluía 4 mecanismos para evitar la generación de soluciones redundantes en la representación de RK. Posteriormente, Paraskevopoulos, Tarantilis y Ioannou (2012) propusieron una nueva representación de soluciones llamada lista de eventos (event list). En su propuesta primero se crea una solución factible real y luego se codifica en conjuntos de eventos de actividades, por ejemplo, en conjuntos de actividades que se puedan realizar al mismo tiempo. Adicionalmente, los autores proponen una búsqueda dispersa como método principal de búsqueda. Los resultados obtenidos son unos de los mejores reportados en la literatura, pero no proveen un estudio para comprobar si los buenos resultados son debidos al funcionamiento y diseño de la búsqueda dispersa o al uso de la nueva representación.

3.3 Una propuesta evolutiva de mejora

Como se pudo observar en el Capítulo 2, uno de los métodos metaheurísticos más exitosos en la resolución de los problemas combinatorios, como el RCPSP, son los algoritmos genéticos, cuya búsqueda también se ve afectada por las soluciones redundantes. En esta sección nos centraremos en el impacto de este fenómeno sobre los operadores que introducen diversidad al método de búsqueda. En el caso de los algoritmos genéticos es el operador de mutación el encargado de introducir diversidad. Así, la redundancia afecta a la mutación principalmente por dos razones: la primera es que suele tratarse de pequeños cambios sobre las representaciones de soluciones que buscan generar diversidad en la población y en muchas ocasiones no pueden lograrlo debido a la redundancia; la segunda razón es que el número de actividades afectadas por la mutación suele ser muy bajo (5 %), para no generar una búsqueda con un nivel de aleatoriedad muy alto. De esta manera, el objetivo de la mutación es introducir nuevo material genético (diversidad) para lo cual se aplica un número reducido de veces, y si en aquellas ocasiones se produce una solución redundante, el algoritmo pierde eficiencia en la búsqueda. La solución a este problema no es simplemente incrementar la frecuencia de la mutación puesto que tanta diversidad podría dificultar la convergencia de los otros operadores.

Para analizar las implicaciones de la redundancia, en esta sección se propone un algoritmo genético (GA) con diferentes estrategias para implementar la mutación y se evalúan los resultados. Además, se propone un nuevo operador de mutación, denominado *mutación agresiva*, que permite ampliar la búsqueda y disminuir la generación de soluciones redundantes. Esta propuesta también incluye una decodificación alternativa para el método de mejora FBI, cuyo objetivo no es reemplazar la existente, sino explorar una forma diferente de decodificación que puede alcanzar soluciones más compactas. A continuación, se describe detalladamente cada elemento que compone el algoritmo genético. Haremos especial énfasis en la mutación, donde se explican los principales operadores considerados en el análisis, y finalmente, todas las propuestas serán evaluadas en la sección 3.4.

Codificación y función de aptitud

La codificación es uno de los elementos más importantes en un algoritmo genético, puesto que todos los operadores que realizan la búsqueda, como el cruce o la mutación, usan dicha codificación como base para generar nuevas soluciones.

Basándose en los resultados computacionales, Hartmann y Kolisch (2000) constataron que la lista de actividades (AL) puede alcanzar mejores resultados que la representación de valor clave (RK). Por esta razón, esta será la codificación usada en el GA propuesto. Además, se le añadirán dos genes: el primero corresponde al SGS usado para decodificar la solución, este puede ser en serie o en paralelo; el segundo gen corresponde a la dirección usada para generar la solución, esta puede ser hacia adelante o hacia atrás. Un esquema del cromosoma completo (codificación) se muestra en la figura 3.2.

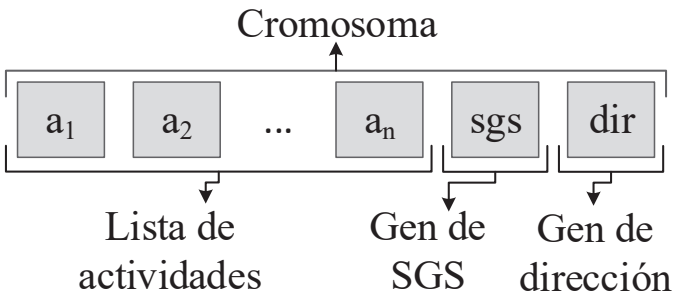

Figura 3.2: Codificación del GA propuesto para resolver el RCPSP.

La función de aptitud será la duración total del proyecto (*makespan*), al igual que la función objetivo para el RCPSP en el modelo de programación entera mixta.

Población inicial

La población inicial en un algoritmo genético debe ser lo suficientemente diversa para evitar la convergencia prematura, y estar compuesta de soluciones que tengan un buen valor de la función de aptitud (*makespan* bajo) para que las generaciones subyacentes puedan encontrar soluciones cercanas a la óptima.

De esta manera, hemos seleccionado el método de muestreo denominado *regret-based biased random sampling* (RBBRS), pues es uno de los métodos de x-pasadas con mejores resultados en la literatura (Drexl 1991; Kolisch y Hartmann 1999). Este método se describe a continuación.

Sea *ele* un conjunto de actividades elegibles en una de las etapas de un SGS, $v(i)$ el valor de la regla de prioridad elegida para la actividad i, p_i el valor estandarizado que depende de dicha regla de prioridad. La probabilidad de que la actividad i sea seleccionada para generar la solución p_i se define en las Ecuaciones 3.1 y 3.2.

$$p_i = \begin{cases} max_{j \in ele} v(j) - v(i) & \text{cuando el objetivo es maximizar } v(i). \\ v(i) - min_{j \in ele} v(j) & \text{cuando el objetivo es minimizar } v(i). \end{cases} \tag{3.1}$$

$$pSelec(i) = \frac{(p_i + \epsilon)^\alpha}{\sum_{j \in ele} (p_j + \epsilon)^\alpha} \tag{3.2}$$

Donde $\epsilon \geq 0$ y $\alpha \geq 0$. Un valor de $\epsilon \neq 0$ asigna un valor positivo a la probabilidad de ser seleccionada a la actividad con peor valor de la regla de prioridad. El valor de α determina el grado de aleatoriedad de la selección. Un valor muy grande hará que la selección sea determinista, es decir que siempre será seleccionada la actividad con mejor valor según la regla de prioridad, y un valor de 0 hará que la selección sea completamente aleatoria. Los parámetros usados en el GA propuesto fueron $\alpha = \epsilon = 1$.

Tamaño de la población

El tamaño de la población afecta al coste/beneficio entre la exploración y la explotación en un GA. Generalmente, el tamaño de la población está relacionado con el tamaño del espacio de búsqueda y la dificultad del problema a tratar. Sin embargo, como se puede apreciar en la sección 2.3, estimar la complejidad de un problema o bien el espacio factible del RCPSP es bastante complicado y poco preciso. Por lo tanto, diversos autores optan por usar los llamados *standard setting* en los GAs ($50 - 100$ individuos) o bien lo determinan mediante experimentación (Lobo y Lima 2005).

Así, no existe un método apropiado para estimar la población en un GA para resolver el RCPSP. Algunos autores declaran que la población decrece a medida que incrementa el número de actividades, como se aprecia en la investigación de Debels y Vanhoucke (2007), y otros proponen poblaciones dinámicas como en el algoritmo desarrollado por Cervantes y col. (2008), donde la población crece a medida que se evalúan cierto número de soluciones, pero el tamaño de la solución inicial también la definen de manera inversa. Finalmente, el tamaño de población del GA propuesto está basado en estas afirmaciones y en la experimentación. La Ecuación 3.3 define el tamaño de la población y la Ecuación 3.4 define el número de generaciones.

$$population = \left(\frac{1}{n} * iterations\right) + 15 \tag{3.3}$$

$$generations = \frac{iterations}{population} \tag{3.4}$$

Selección

La selección en un algoritmo genético se define como la técnica a usar para seleccionar un subconjunto de individuos de una población que serán padres que generarán la futura población de soluciones. Entre los diferentes algoritmos evolutivos para resolver el RCPSP, los métodos de ruleta y de *ranking* son los más usados para realizar la selección dentro de un algoritmo genético. En la ruleta, las soluciones son seleccionadas acorde directamente a su valor de la función de aptitud (*makespan*). El método de *ranking* también basa su selección en la función de aptitud, pero se usa cuando hay individuos con valores de aptitud muy diferentes y de usarse la ruleta, la mayor parte de la probabilidad de selección sería asignada a aquellos individuos con un valor alto de la función de aptitud, dejando a las demás con una probabilidad prácticamente nula. En el GA propuesto, hemos seleccionado el método de ruleta puesto que las soluciones tienen poca variabilidad en su valor de *makespan* como para tener los problemas anteriormente mencionados.

Reemplazo

El reemplazo en los algoritmos genéticos es la manera como se forman las siguientes generaciones. Es decir, construir una nueva población a partir de la población actual (de padres) y la población de hijos. Las estrategias de reemplazo más comunes son el reemplazo total, el reemplazo elitista, el reemplazo aleatorio y el reemplazo por torneo. En el primero la nueva población es directamente tomada de la población de hijos. El reemplazo elitista toma solo los mejores

individuos de la población de padres y de hijos para formar la nueva. El reemplazo aleatorio selecciona completamente de manera aleatoria a los individuos de la nueva población. Finalmente, el reemplazo por torneo, consiste en seleccionar un par de soluciones cualquiera y comparar cuál de ellas tiene mejor valor de la función de aptitud, la ganadora será puesta directamente en la nueva población.

El GA propuesto en esta investigación usa una estrategia semi-elitista, la cual consiste en construir la nueva población a partir del 50 % de los mejores individuos de la población de padres y el 50 % restante de los mejores individuos de la población de hijos. Dado que este operador tiene un componente de elitismo considerable, el operador de mutación propuesto será el encargado de evitar la convergencia prematura del algoritmo.

Operador de cruce

El operador de cruce es uno de los operadores más relevantes para un algoritmo genético, puesto que es el mecanismo principal para realizar la búsqueda. El objetivo es transmitir parte de las características *buenas* de los padres a los hijos, de manera que en promedio los descendientes posean los mejores genotipos y así, mejorar el valor de la función de aptitud.

El GA propuesto usa un cruce estándar en dos puntos adaptado al RCPSP y a la codificación propuesta. Primero se seleccionan dos individuos de la población inicial para hacer el cruce: p_1 y p_2, de los cuales se generan dos hijos de la siguiente manera: se generan dos valores aleatorios q_1 y q_2 de tal manera que $0 < q_1 < q_2 < n$. A continuación el hijo toma los primeros genes de la posición 0 a q_1 de p_1, de la posición $q_1 + 1$ hasta q_2 los genes son tomados de p_2 pero no en el orden directo, es decir, se recorre el cromosoma de p_2 y se seleccionan las primeras $q_2 - (q_1 + 1)$ actividades que no estén repetidas en el hijo. De manera similar, los genes restantes $q_2 + 1$ a n son tomados de manera indirecta de p_1. Para ilustrar el procedimiento usado, la figura 3.3 muestra un cruce simplificado de un solo punto entre dos padres factibles del problema ilustrado en la figura 2.1.

Respecto a los genes de SGS y dirección, estos son heredados si los padres tienen esos mismos genes, de lo contrario son generados aleatoriamente.

Mutación

La función del operador de mutación es introducir nueva información genética (variedad) a la población, con el objetivo de evitar una convergencia prematura generada por el cruce y la selección, y permitir explorar lugares del espacio de búsqueda diferentes a los generados por la estructura de la población actual, a los que de otra forma no se podrían llegar. La mutación en los algoritmos

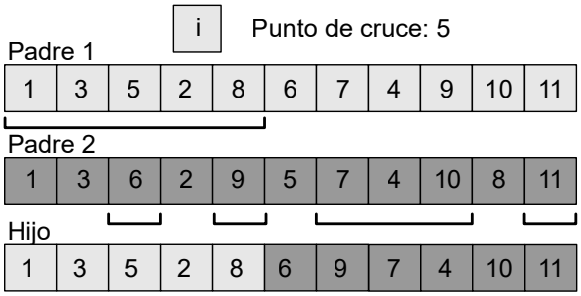

Figura 3.3: Ejemplo del operador de cruce del GA propuesto.

genéticos genera grandes perturbaciones a las soluciones para lograr explorar dichos espacios, por esta razón la probabilidad de mutación suele ser baja, cerca del 5 %, puesto que una probabilidad alta de la mutación dirigiría la búsqueda a una casi aleatoria. Sin embargo, como se mencionó en la sección 3.2, si la mutación produce soluciones redundantes, no estaría aportando la variabilidad necesaria en el algoritmo genético.

Los métodos más relevantes para la mutación en el RCPSP son el intercambio de actividades (Hartmann 1998), el movimiento hacia la derecha (Hartmann 2002), y la inserción de Boctor (Boctor 1996), siendo esta última la más usada puesto que alcanza uno de los mejores resultados reportados en la literatura. Por esta razón, este es el mecanismo seleccionado para realizar el análisis sobre el impacto de la redundancia en la mutación.

La inserción de Boctor se puede definir de la siguiente manera: inicialmente, una actividad i es seleccionada aleatoriamente de una lista de actividades $L1 = (a_1, ..., a_i, ..., a_n)$, luego se calculan las posiciones $maxPred$ (máxima posición del último predecesor) y $minSuc$ (mínima posición del primer sucesor) de la actividad i. Entonces, se genera un valor aleatorio Pos de tal manera que $maxPred < Pos < minSuc$, y la actividad i es insertada en la posición Pos.

Como ejemplo, en la figura 3.4 se muestra una inserción de la actividad 6 en una lista de actividades con dirección hacia atrás.

Existen principalmente dos maneras de hacer las inserciones basándose en la forma en que se implementa la mutación. Las cuales se describen a continuación.

- **Mutación por solución.** Esta forma implementa el operador de mutación sobre una solución codificada (un individuo), es decir sobre el cromosoma

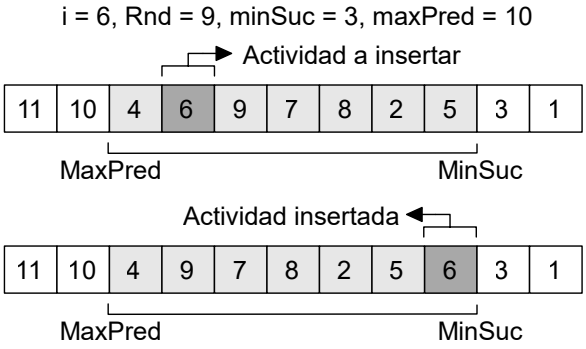

Figura 3.4: Ejemplo de una inserción para el operador de mutación propuesto.

completo (lista de actividades y los genes de SGS y de dirección). En este caso, si la probabilidad de mutación es de 0,05, en promedio solo el 5 % de los individuos en la población sufrirán la mutación, que consiste en la inserción de una sola actividad.

- **Mutación por actividad.** Esta forma implementa el operador de mutación sobre cada actividad, es decir sobre cada gen sin diferenciar individuos, así el operador recorre cada gen generando un número aleatorio para determinar si dicho gen muta o no. Para estimar el número promedio de soluciones afectadas podemos definir esta mutación como un experimento Bernoulli, donde cada mutación (ma) puede catalogarse como éxito o fracaso. Para estas estimaciones se seleccionará la probabilidad habitual de 0,05 en caso de éxito (se realiza la mutación). De esta manera y asumiendo que el evento de mutación es independiente entre genes, la probabilidad de que una solución sea afectada (PS), es decir, que al menos se realice una mutación, se define en la Ecuación 3.5, donde n es el número de actividades que componen una solución y e_b es el número de ensayos acorde a la definición de un experimento Bernoulli.

$$PS = P(ma >= 1) = 1 - P(ma = 0) = 1 - (0{,}95^{e_b}) \quad : \quad e_b = n \qquad (3.5)$$

Bajo estos mismos supuestos, si definimos a p_l como la probabilidad de realizar l inserciones sobre una solución de n actividades, el número de inserciones promedio (Nl) que realiza esta mutación se puede calcular usando la Ecuación 3.6.

$$Nl = \sum_{l=0}^{e_b} l * p_l = \sum_{l=1}^{e_b} l * (0{,}95^{e_b - l} * 0{,}05^{l}) * \frac{e_b!}{l!(e_b - l)!} \quad : \quad e_b = n \qquad (3.6)$$

De esta manera, la probabilidad de mutación de una solución (PS) depende del número de actividades, puesto que cuantas más actividades tenga una solución, mayor será la probabilidad de que ocurra al menos una mutación sobre una actividad. El comportamiento de esta mutación puede verse en la figura 3.5. En esta figura se muestra la probabilidad de mutación de una solución, definiendo la probabilidad de mutación para cada actividad en el valor estándar 0,05. Para problemas con más de 45 actividades la probabilidad de mutación asciende a 90 % y a partir de 100 actividades la probabilidad es prácticamente 100 %.

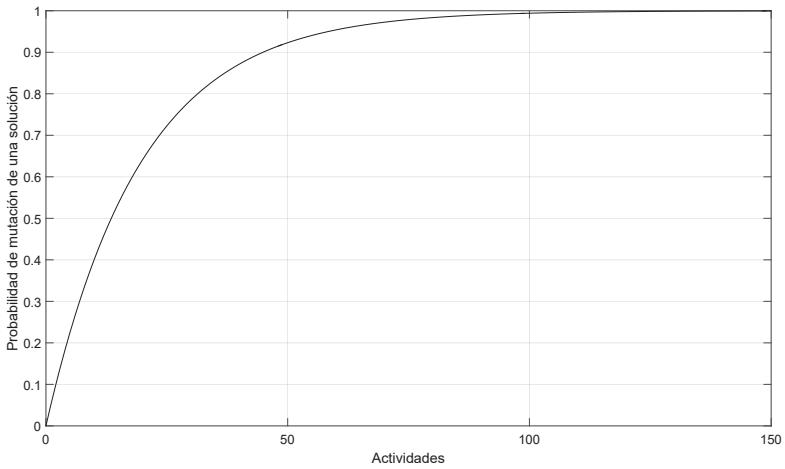

Figura 3.5: Probabilidad de mutación de una solución cuando se implementa la mutación por actividad.

De manera similar, y fijando nuevamente la probabilidad de mutación por actividad al el valor estándar 0,05, en la figura 3.6 se muestra en número promedio de inserciones. Como se puede observar tiene una tendencia creciente lineal, cuantas más actividades tenga el problema más inserciones se realizarán.

En esta investigación se propone un nuevo operador de inserciones múltiples, basado en el operador de inserción de Boctor. Realizando las mutaciones por solución en lugar de hacerlo por actividades como es común en la literatura. De

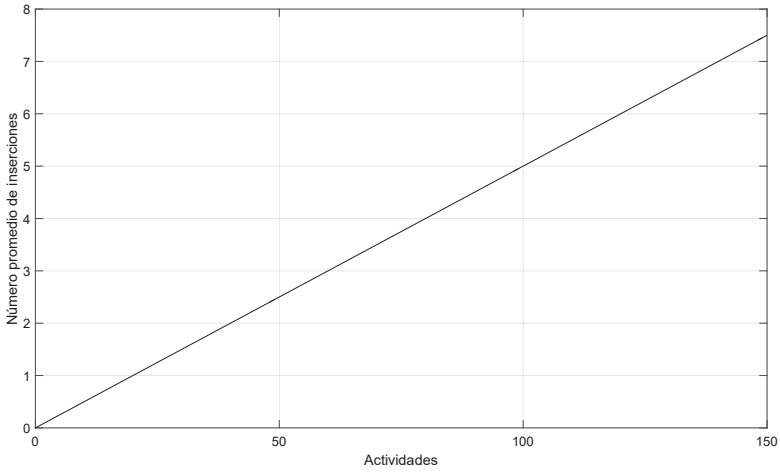

Figura 3.6: Número promedio de inserciones cuando se implementa la mutación por actividad.

esta manera se pueden controlar los parámetros de la mutación de forma más precisa, es decir, el número de inserciones y la probabilidad sobre la solución no dependen del número de actividades y se les puede asignar un valor diferente. En la sección 3.4 se explica a detalle la experimentación y los resultados de este operador en sus dos versiones (mutación por actividad o por solución) y el nuevo operador propuesto. Basándose en dichos resultados, se establece que el número de inserciones (NI) adecuado para el operador de mutación es de $NI = 3$, y la probabilidad de mutación sobre las soluciones que mejores resultados alcanza es del 90 %.

Mejora local

Como se describió en la subsección 2.6.5, uno de los procedimientos de mejora local más importantes es el (forward-backward improvement, FBI), pues este puede encontrar soluciones más compactas, llenando espacios *vacíos* en una solución dada mediante una pasada hacia adelante y una hacia atrás.

Por definición, al ordenar las actividades tras las dos pasadas del FBI, se obtiene una solución donde cada actividad fue considerada de menor a mayor tiempo de inicio para ser programada lo más pronto posible. Así, esta solución corresponde a una lista de actividades cuyo SGS es el serial. Sin embargo, hemos encontrado que, si se usa un SGS en paralelo sobre la lista encontrada por el FBI, se puede alcanzar mejores soluciones.

Como ejemplo, tomaremos la lista de actividades y la solución obtenida por el ejemplo del FBI en la figura 2.8. Esta solución se muestra de nuevo en la figura 3.7.a junto con su lista de actividades. En la figura 3.7.b se aprecia la misma lista de actividades, pero decodificada mediante un SGS en paralelo.

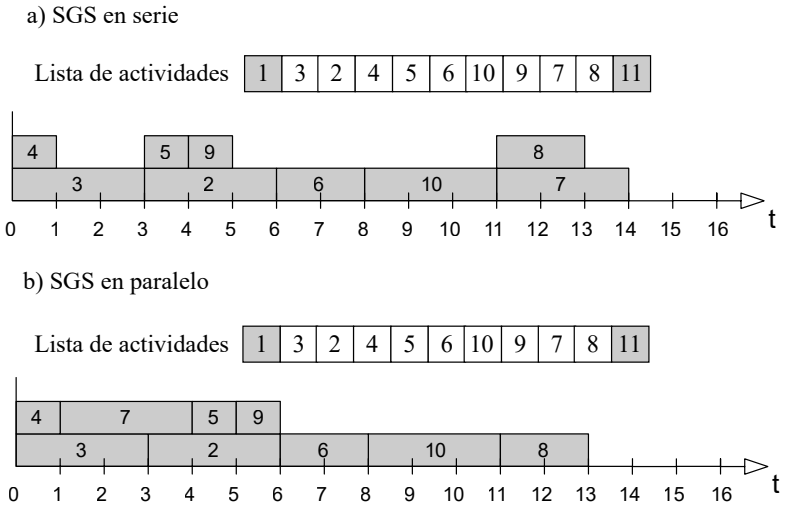

Figura 3.7: Un ejemplo de la decodificación en paralelo de una AL obtenida por el FBI.

3.4 Evaluación de los métodos propuestos

Para realizar la evaluación del GA propuesto, se llevaron a cabo diversas pruebas experimentales sobre la bien conocida librería PSPLIB para el RCPSP (véase sección 2.7). Esta librería consiste de 4 conjuntos de problemas: $j30$, $j60$, $j90$, y $j120$, pero como es común en las investigaciones sobre el RCPSP, se tomarán todos los conjuntos excepto el $j90$. La principal razón de esto es que los parámetros son los mismos para los conjuntos $j30$, $j60$ y $j90$, por tanto, de manera general los autores han preferido directamente tomar los problemas del conjunto $j120$ como referencia para problemas de mayor tamaño, pues este conjunto tiene más actividades y los parámetros usados para su generación son diferentes del de los otros conjuntos. La evaluación en esta sección se divide en dos partes: la primera dedicada a la experimentación acerca de la mutación y el análisis del efecto de las soluciones redundantes; la segunda, dedicada a la comparación con otras metaheurísticas reportadas en la literatura.

3.4.1 La mutación y el efecto de la redundancia

Para evaluar el impacto de las soluciones redundantes, primero se ejecutó el GA propuesto pero solo con una inserción sobre la PSPLIB, incrementando la probabilidad del operador de mutación. En la figura 3.8 se muestra la desviación promedio, respecto al valor de referencia, usando el operador de mutación (una inserción) con 1 000 iteraciones respecto a la probabilidad de mutación sobre cada conjunto $j30$, $j60$, y $j120$. La figura 3.8 es una gráfica para cada conjunto, donde se puede observar que cuanto mayor es la probabilidad de mutación, mejores resultados (menor *makespan*) se obtienen. Esto indica que el operador de una inserción no introduce suficiente nueva información para mejorar las soluciones.

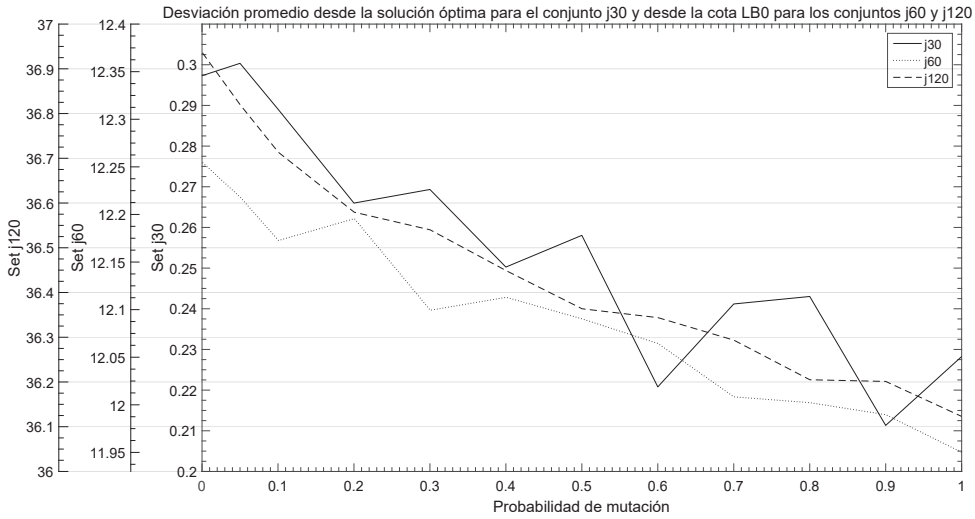

Figura 3.8: Impacto del cambio en la probabilidad de **mutación por soluciones** respecto al *makespan*.

A pesar que los mejores resultados en esta figura fueron alcanzados con una probabilidad igual a 1,0; esto no implica que todas las soluciones realmente hayan mutado debido a que, de hecho, una inserción no garantiza construir una solución diferente. Consecuentemente, la probabilidad real de mutación debe ser inferior a 1,0; lo que implica que la población solo está guiada por el cruce cuando la mutación no tiene efecto debido a la construcción de una solución redundante, obteniendo así mejores resultados. Contrario a lo que debería ocurrir si realmente la probabilidad de mutación fuese de 1,0 donde la búsqueda sería completamente aleatoria. Sin embargo, en la figura 3.8, la desviación promedio decrece cuando

la probabilidad de mutación aumenta, evidenciando que el incluir diversidad a la población tiene un efecto positivo al rendimiento del algoritmo.

Por otra parte, en la figura 3.9 se muestra el impacto sobre el *makespan* del operador de mutación por actividades respecto al cambio de la probabilidad de mutación. En este caso podemos observar que los mejores valores son encontrados con probabilidades cercanas a la estándar 0,05, de tal manera que al aumentar la probabilidad en promedio empeora el rendimiento de la búsqueda. Por este motivo para efectos de comparación sobre la mutación por actividades, fijaremos la probabilidad en 0,05.

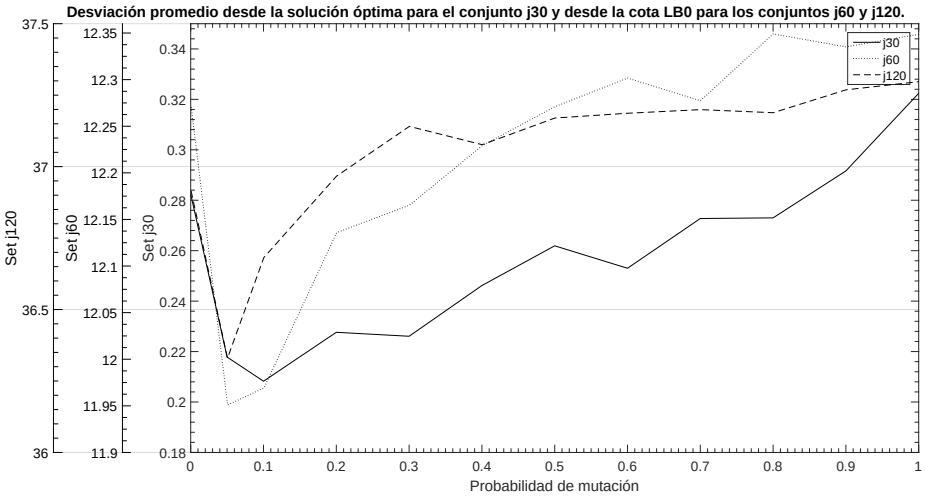

Figura 3.9: Impacto del cambio en la probabilidad de **mutación por actividades** respecto al *makespan*.

Para realizar las comparaciones entre las diferentes formas de implementar la mutación (por actividad o por solución) y establecer el número adecuado de inserciones, primero se estableció una equivalencia entre la mutación por actividades a la mutación por solución. En la tabla 3.1 se resumen las equivalencias. Por ejemplo, realizar la mutación por actividades con un valor de 0,05 sobre el conjunto $j120$, es equivalente en promedio a realizar la mutación por soluciones con una probabilidad de 0,998, realizando 6 inserciones cada vez.

En las figuras 3.10, 3.11 y 3.12, se muestran las superficies de las desviaciones promedio del *makespan* en función del número de inserciones y la probabilidad

Mutación por actividad con probabilidad 0,05			
	Conjunto	Probabilidad	Número inserciones
Mutación por solución	$j30$	0,785	1,5
	$j60$	0,953	3
	$j120$	0,998	6

Tabla 3.1: Equivalencias entre mutaciones por solución y por actividad.

de mutación para los conjuntos $j30$, $j60$ y $j120$, respectivamente. De manera general, se puede apreciar que todos los conjuntos tienden a obtener mejores resultados en probabilidades cercanas al 90 % y aproximadamente 3 inserciones por solución. La figura 3.10 es la que tiene un mayor espectro de posibilidades de inserciones y probabilidades, además de ser la más irregular. Pero esto se debe a que el conjunto $j30$ es el más pequeño y el algoritmo genético puede encortar soluciones relativamente buenas y en algunos casos óptimas, haciendo que los promedios de las desviaciones fluctúen. De hecho el rango de las desviaciones es de solo 0,16. Las figuras 3.11 y 3.12 muestran un comportamiento más definido, con un rango de diferencias mucho mayor, de 0,5 y 0,9, respectivamente. En estas figuras, también se puede apreciar el desempeño de la mutación por actividades, usando los valores equivalentes suministrados en la tabla 3.1. Como se puede observar en la figura 3.10, mientras que la mutación por actividades realiza solo 1,5 inserciones en promedio, los mejores resultados se obtienen con un número mayor de 3 inserciones en promedio. En la figura 3.11, la mutación por actividades inserta en promedio 3 actividades por solución con una probabilidad de 0,953, estos valores coinciden aproximadamente a los determinados experimentalmente por medio de estas gráficas. Sin embargo, en la mutación por actividades el número de inserciones aumenta a medida que crece el número de actividades del problema, al igual que la probabilidad (figuras 3.5 y 3.6). Esto se puede observar en la figura 3.12, donde la mutación por actividad realiza 6 inserciones en promedio con una probabilidad 0,998. En contraste, la figura claramente sugiere una probabilidad del 0,9 con un número de inserciones menor, nuevamente 3. De este modo la mutación agresiva propuesta usa el esquema de mutación por solución, con una probabilidad de 90 % y con 3 inserciones en todos los casos.

Existen dos principales razones que sustentan el hecho de que los resultados sugieran un número constante de probabilidad por solución y de número de inserciones. La primera es que las soluciones redundantes son independientes del número de actividades de un proyecto, así que con 3 inserciones en promedio es suficiente para disminuir el número de soluciones redundantes generadas. La segunda, es que un número mayor de inserciones podría entorpecer el desempeño

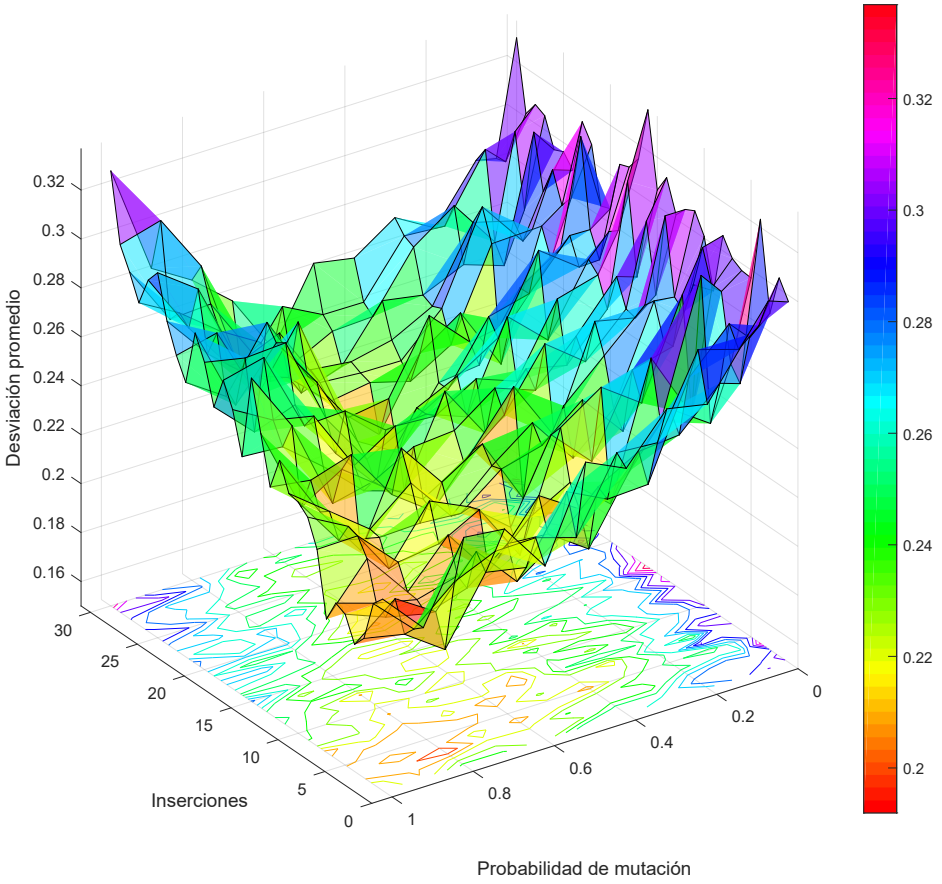

Figura 3.10: Desviaciones promedio del *makespan* en función del número de inserciones y la probabilidad de mutación por solución para el conjunto $j30$.

de los otros operadores en su objetivo de converger (explotación) el espacio de búsqueda. Por esta misma razón, una probabilidad cercana a 90 % es suficiente para introducir nueva información a la búsqueda.

Finalmente, en la figura 3.13 se muestra el número de soluciones redundantes en función del número de iteraciones (1 000, 5 000, y 50 000) sobre el conjunto $j30$ para la mutación por actividad y la mutación agresiva. En todos los casos, la mutación agresiva encuentra aproximadamente un 20 % menos de soluciones redundantes.

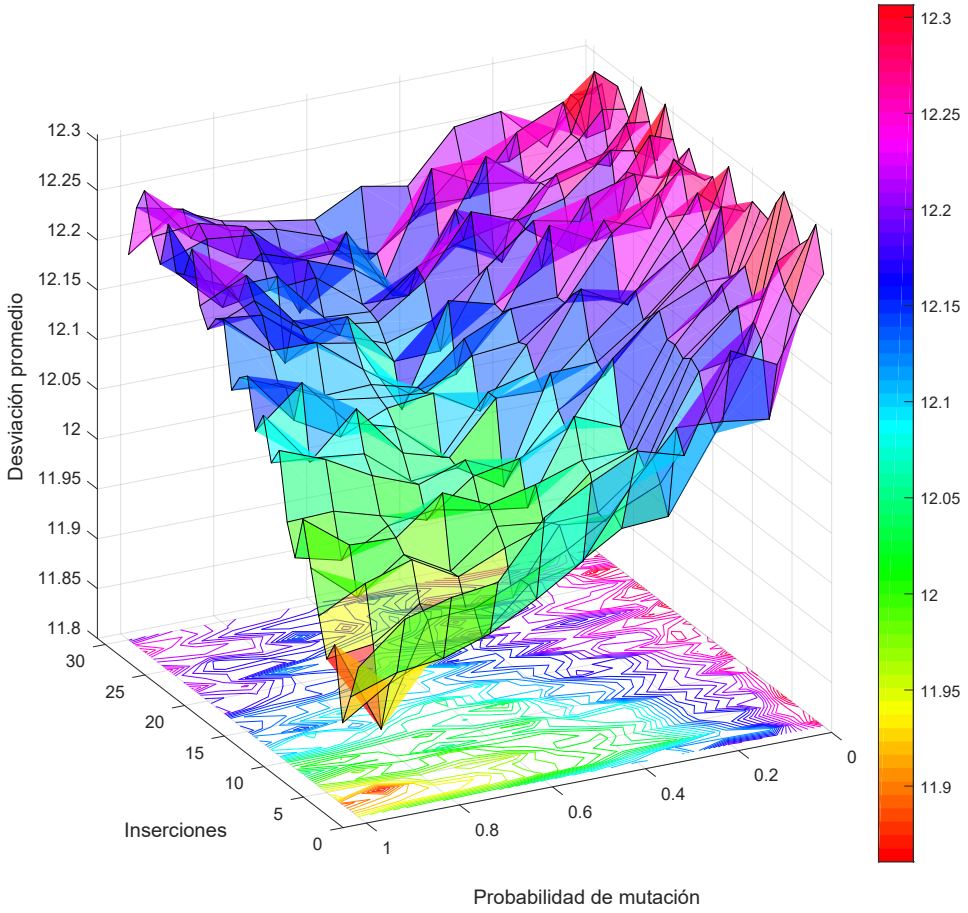

Figura 3.11: Desviaciones promedio del *makespan* en función del número de inserciones y la probabilidad de mutación por solución para el conjunto $j60$.

3.4.2 Evaluación comparativa

Kolisch y Hartmann (2006) propusieron una tabla comparativa y resumieron los resultados obtenidos por diversas metaheurísticas para resolver el RCPSP. Con el objetivo de proveer una comparación justa entre los métodos, Kolisch y Hartmann (2006) usaron las iteraciones como criterio de comparación en lugar del tiempo de ejecución del algoritmo. La principal razón es que el tiempo de ejecución depende de diversos factores como la compilación, el lenguaje de programación

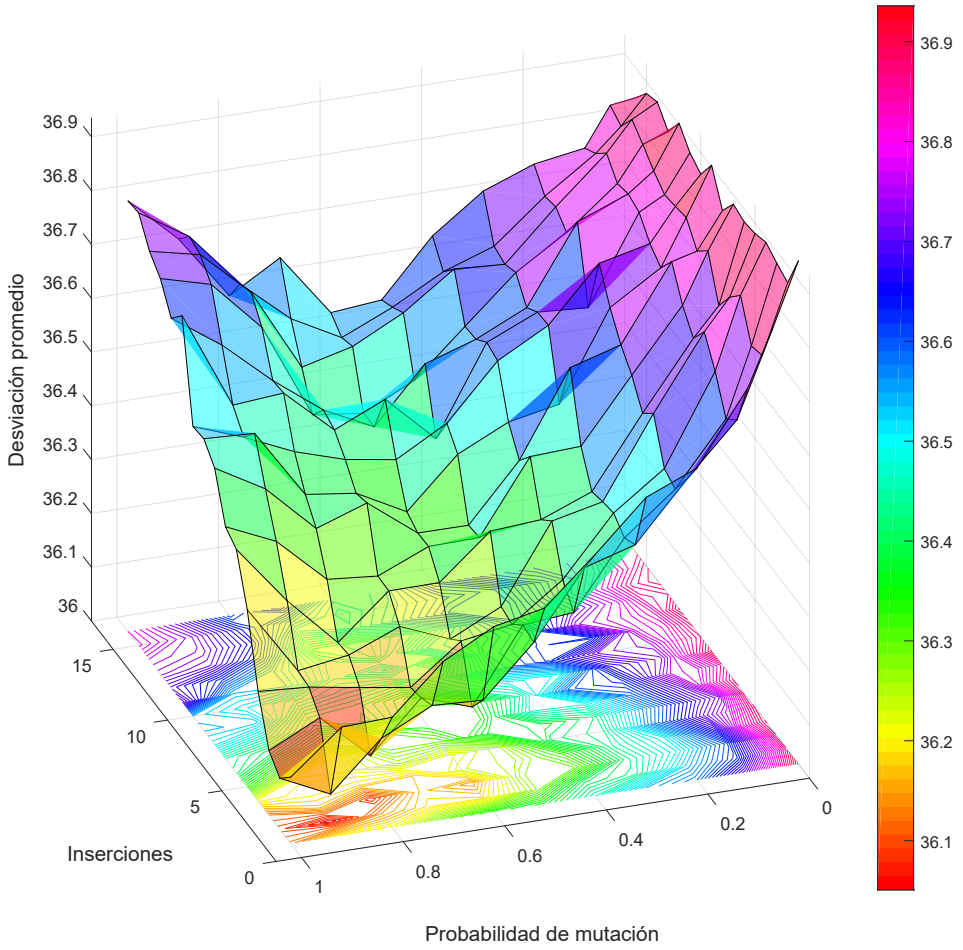

Figura 3.12: Desviaciones promedio del *makespan* en función del número de inserciones y la probabilidad de mutación por solución para el conjunto $j120$.

usado, el *hardware*, etc., mientras que las iteraciones son independientes de estas características. Los mismos autores, definen una iteración como la construcción de una solución completa o bien una solución parcial de la cual se puede deducir el *makespan*. Esta tabla se ha convertido en una referencia estándar en la comparación de los métodos para resolver el RCPSP.

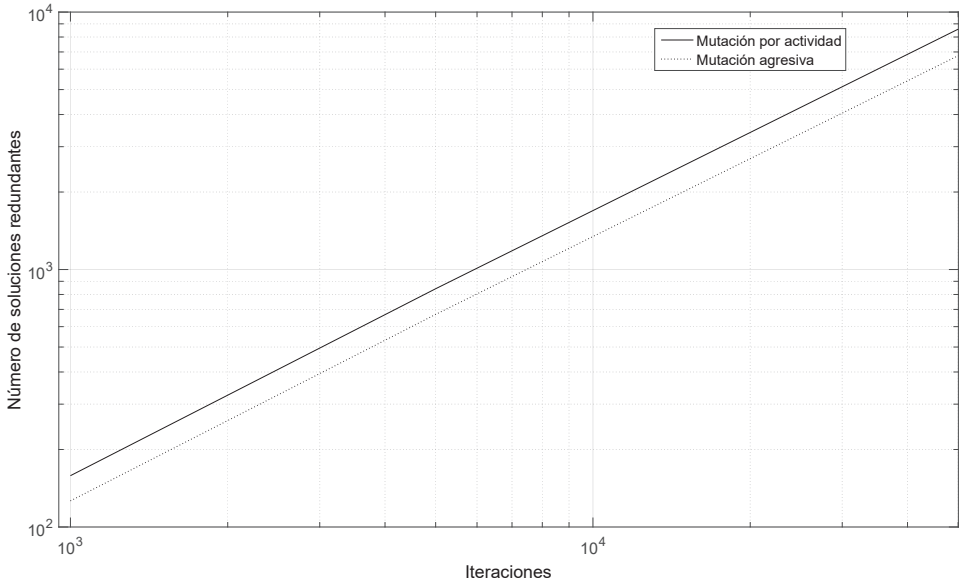

Figura 3.13: Número de soluciones redundantes de la mutación por actividad y mutación agresiva

Siguiendo la clasificación propuesta por Kolisch y Hartmann (2006), Paraskevopoulos, Tarantilis y Ioannou (2012) actualizaron los resultados de la tabla de comparaciones considerando los algoritmos más recientes. Hasta donde tenemos conocimiento esta es la comparación más reciente de los algoritmos para el RCPSP.

De este modo, los resultados computacionales obtenidos por el GA propuesto, se presentan en las tablas 3.2, 3.3, y 3.4, como una adaptación de las tablas mostradas en (Paraskevopoulos, Tarantilis y Ioannou 2012). La primera fila muestra el conjunto a tratar de la librería PSPLIB. La primera columna muestra el algoritmo usado, la segunda muestra a los autores y su referencia. La tercera, cuarta y quinta columna, muestran la desviación promedio (%) de la solución óptima en el caso del conjunto $j30$ y respecto a la cota inferior del camino crítico $LB0$ para el resto. El GA propuesto fue ejecutado en un PC con un procesador Intel(R) Core(TM) $i7$ CPU a 3,30GHz. Todos los resultados pueden ser reproducidos. El archivo ejecutable y las instrucciones se pueden encontrar en `http://gps.webs.upv.es/psplib-energy/es-GA-AM.php`.

Se han considerado dos enfoques del GA propuesto para hacer la comparación. El primero, considera que la mejora FBI usa 3 iteraciones: la solución actual, otra en el paso hacia adelante y la última en el paso hacia atrás; denominadas programaciones generadas. El segundo, considera que el FBI usa una sola iteración puesto no se requiere la llamada de un decodificador, las soluciones en este caso son denominadas programaciones evaluadas.

Las abreviaciones usadas en las tablas 3.2, 3.3, y 3.4 son: **GA-AMGS** y **GA-AMES** (el GA con la mutación agresiva propuesta en esta investigación, contando las programaciones generadas y las programaciones evaluadas, respectivamente), **SAILS** (Scatter search-Adaptive Iterated Local Search), **GA** (Genetic Algorithm), **ACOSS** (Ant Colony Optimization Scatter Search), **SS** (Scatter Search), **PR** (Path Relinking), **GAPS** (Genetic Algorithm for Project Scheduling), **FBI** (Forward Backward Improvement), **TS** (Tabu Search), y **ANGEL** (ANt colony optimization GEnetic algorithm Local search).

Tabla 3.2: Comparación computacional entre las principales metaheurísticas para resolver el RCPSP para el conjunto $j30$ de la librería PSPLIB.

Conjunto $j30$				
Algoritmo	Referencia	%dev		
		1 000	5 000	50 000
SAILS	Paraskevopoulos, Tarantilis y Ioannou (2012)	0.03	0.01	0.00
GA-AMES	**Esta investigación**	**0.06**	**0.01**	**0.00**
GA, TS-PR	Kochetov y Stolyar (2003)	0.10	0.04	0.00
SS-PR	Mahdi Mobini y col. (2008)	0.05	0.02	0.01
GAPS	Mendes, Gonçalves y Resende (2009)	0.06	0.02	0.01
GA-AMGS	**Esta investigación**	**0.13**	**0.05**	**0.01**
ACOSS	Chen y col. (2010)	0.14	0.06	0.01
SS-FBI	Debels y col. (2006)	0.27	0.11	0.01
GA	Debels y Vanhoucke (2007)	0.15	0.04	0.02
GA-hybrid FBI	Valls, Ballestin y Quintanilla (2008)	0.27	0.06	0.02
TS	Nonobe y Baraki (2002)	0.46	0.16	0.05
GA	Hartmann (2002)	0.38	0.22	0.08
ANGEL	Tseng y Chen (2006)	0.22	0.09	-

Las tablas 3.2, 3.3, y 3.4 están organizadas acorde a la quinta columna (50 000 iteraciones). Adicionalmente, en la tabla 3.5 se presentan el tiempo promedio del algoritmo GA-AMES y GA-AMGS, respectivamente. Basándose en los resultados obtenidos, se puede concluir que el GA propuesto alcanza resultados altamente

Tabla 3.3: Comparación computacional entre las principales metaheurísticas para resolver el RCPSP para el conjunto $j60$ de la librería PSPLIB.

Conjunto $j60$				
Algoritmo	Referencia	%dev		
		1 000	5 000	50 000
SAILS	Paraskevopoulos, Tarantilis y Ioannou (2012)	11.05	10.72	10.54
GA-AMES	**Esta investigación**	**11.10**	**10.76**	**10.56**
SS-PR	Mahdi Mobini y col. (2008)	11.12	10.74	10.57
GAPS	Mendes, Gonçalves y Resende (2009)	11.72	11.04	10.67
ACOSS	Chen y col. (2010)	11.72	10.98	10.67
GA	Debels y Vanhoucke (2007)	11.45	10.95	10.68
SS-FBI	Debels y col. (2006)	11.73	11.10	10.71
GA-hybrid FBI	Valls, Ballestin y Quintanilla (2008)	11.56	11.10	10.73
GA, TS-PR	Kochetov y Stolyar (2003)	11.71	11.17	10.74
GA-AMGS	**Esta investigación**	**11.50**	**11.04**	**10.79**
GA	Hartmann (2002)	12.21	11.70	11.21
ANGEL	Tseng y Chen (2006)	11.94	11.27	-
TS	Nonobe y Baraki (2002)	12.97	12.18	11.58

competitivos usando poco tiempo de cómputo en comparación con las mejores metaheurísticas reportadas en la literatura. Como se esperaba, el algoritmo GA-AMES obtiene mejores resultados que el algoritmo GA-AMGS. Esto es debido a que el algoritmo GA-AMES usa un mayor número de programaciones completas, como se puede apreciar también en la tabla 3.5, pues tiene mayor tiempo de cómputo. Es importante notar que los mejores resultados obtenidos por el GA propuesto no solo implementaban el operador de mutación agresiva, sino también las dos fases de búsqueda, la primera con una alta probabilidad de mutación y la segunda con baja probabilidad de mutación. Esto evidencia la importancia de que los operadores incluyan diversidad en etapas tempranas de la búsqueda para este tipo de problemas.

Tabla 3.4: Comparación computacional entre las principales metaheurísticas para resolver el RCPSP para el conjunto $j120$ de la librería PSPLIB.

Conjunto $j120$				
Algoritmo	Referencia	%dev		
		1 000	5 000	50 000
ACOSS	Chen y col. (2010)	35.19	32.48	30.56
GA-AMES	**Esta investigación**	**32.98**	**32.01**	**30.65**
SAILS	Paraskevopoulos, Tarantilis y Ioannou (2012)	33.32	32.12	30.78
GA	Debels y Vanhoucke (2007)	34.19	32.34	30.82
GA-hybrid FBI	Valls, Ballestin y Quintanilla (2008)	34.07	32.54	31.24
GAPS	Mendes, Gonçalves y Resende (2009)	35.87	33.03	31.44
SS-PR	Mahdi Mobini y col. (2008)	34.51	32.61	31.37
SS-FBI	Debels y col. (2006)	35.22	33.10	31.57
GA-AMGS	**Esta investigación**	**34.13**	**32.80**	**31.93**
GA, TS-PR	Kochetov y Stolyar (2003)	34.74	33.36	32.06
GA	Hartmann (2002)	37.19	35.39	33.21
ANGEL	Tseng y Chen (2006)	36.39	34.49	-
TS	Nonobe y Baraki (2002)	40.86	37.88	35.85

Tabla 3.5: Tiempo promedio de ejecución del GA propuesto sobre la librería PSPLIB. (**GA-AMES**/**GA-AMGS**).

$j\#$	Iteraciones		
	1 000	5 000	50 000
$j30$	$0{,}05s/0{,}02s$	$0{,}19s/0{,}08s$	$1{,}70s/0{,}074s$
$j60$	$0{,}25s/0{,}17s$	$0{,}66s/0{,}28s$	$5{,}89s/2{,}43s$
$j120$	$3{,}11s/1{,}62s$	$7{,}13s/2{,}87s$	$69{,}4s/25{,}27s$

3.5 Conclusiones

En este capítulo se abordó el problema de programación con recursos restringidos en su versión uni-modal, y se propuso un algoritmo genético con un operador de mutación agresiva que disminuye el número de soluciones redundantes generadas.

La principal aportación fue analizar el fenómeno de redundancia que puede presentarse en las representaciones de soluciones. Estas últimas son codificaciones usadas por las metaheurísticas para realizar los procedimientos de búsqueda. Así,

una solución redundante es una solución real (definida por los tiempos de inicio de cada actividad) que tiene asociadas varias representaciones. Como consecuencia, los procedimientos de búsqueda pueden gastar esfuerzo computacional en soluciones que no aportan información nueva al algoritmo, además de no suministrarle suficiente diversidad a la búsqueda impidiendo que encuentre mejores soluciones.

El impacto del fenómeno de redundancia fue demostrado de manera experimental en los algoritmos evolutivos, específicamente en los algoritmos genéticos. Para este cometido, se seleccionó el operador de mutación que es el encargado de introducir nueva información (diversidad) en la población. El operador de mutación más relevante usado en la literatura está basado en la inserción de Boctor (Boctor 1996). En esencia, es un procedimiento que permite insertar una actividad en una lista de actividades, generando una solución factible por precedencias. La inserción de Boctor se puede implementar, principalmente, de dos maneras: ejecutando la inserción por gen o por cromosoma. En el primer caso, denominado en esta tesis como mutación por actividad, la mutación se ejecuta con una probabilidad sobre cada actividad de las soluciones en la población. En el segundo caso, denominado mutación por solución, la mutación se ejecuta con una probabilidad sobre un individuo (una solución completa) de la población.

En la mutación por actividad, tanto el número promedio de inserciones como la probabilidad de que un individuo mute dependen del número de actividades, con una tendencia creciente en los dos casos. De este modo, para un problema con 30, 60, y 120 actividades el número promedio de inserciones y la probabilidad de mutación por solución son de $(1{,}5; 78{,}5\,\%)$, $(3; 95{,}3\,\%)$, y $(6; 99{,}8\,\%)$, respectivamente. En la mutación por solución se mantienen constantes tanto el número de inserciones como la probabilidad de mutación, debido a que siempre se considera una solución completa, la cual es independiente del número de actividades. En los dos casos se mostró que estas estrategias no eran las adecuadas para reducir el número de soluciones redundantes, ni alcanzar los mejores resultados.

En la mutación por solución se constató que se lograban mejores resultados a medida que la probabilidad de mutación aumentaba. Esto se debía a que el operador de mutación, además de realizar muy pocas inserciones, generaba algunas soluciones redundantes, haciendo que la mutación pierda efecto y que se llegue a una solución ya encontrada. Respecto a la mutación por actividad, se mostró que introducía demasiada diversidad, dada su alta probabilidad de mutación dependiente del número de actividades, y a pesar de ello no disminuyó el número promedio de soluciones redundantes.

Por esta razón, se propuso un nuevo operador de mutación por solución, denominado mutación agresiva para disminuir la generación de soluciones redundantes, el

cual consiste en una múltiple inserción con una alta probabilidad de ocurrencia. Basándose en la experimentación realizada, el valor de inserciones por solución se estableció en 3 con una probabilidad del 90 %. Los resultados muestran que los casos de prueba evaluados usando la mutación propuesta alcanzan menores valores de *makespan* respecto a las mutaciones mencionadas anteriormente, incluso se encuentra un 20 % menos de soluciones redundantes en comparación con la mutación por actividad. Además, en contraste al número de inserciones y la probabilidad de mutación utilizados por esta última mutación, los resultados sugieren que para disminuir la generación de soluciones redundantes e introducir suficiente diversidad, dichos valores deben ser constantes.

La evaluación del GA propuesto también se realizó en comparación con los mejores métodos de solución para el RCPSP reportados en la literatura. Para dicha evaluación el GA propuesto incluye una manera alternativa de decodificar la solución mejorada usando el método FBI, la cual puede obtener soluciones más compactas. Los resultados fueron obtenidos mediante la resolución de la conocida librería PSPLIB para problemas de programación de proyectos con recursos restringidos. Los resultados exponen que el GA propuesto logra soluciones altamente competitivas con respecto a las metaheurísticas más exitosas reportadas en la literatura.

Finalmente, como contribución general, se destaca que el operador de mutación agresiva puede ser fácilmente incorporado o adaptado para ser usado con otras metaheurísticas y así generar perturbaciones sobre la población y suministrar diversidad al proceso de búsqueda, disminuyendo el número de soluciones redundantes.

Capítulo 4

Extensión del RCPSP a un problema basado en la eficiencia energética: MRCPSP-ENERGY

En este capítulo se propone una extensión del RCPSP a un nuevo problema denominado MRCPSP-ENERGY, en el que se incluye un recurso adicional no restringido: la energía, la cual es consumida por cada actividad y da origen a diversos modos de ejecución. El objetivo es la obtención de soluciones sostenibles y energéticamente eficientes. Adicionalmente, se propone una extensa librería de casos de prueba para el problema propuesto denominada PSPLIB-ENERGY, siendo esta una adaptación de la librería PSPLIB, ampliamente usada y considerada una base de casos de prueba estándar en problemas de programación de proyectos.

4.1 Introducción

Uno de los principales desafíos en la industria es llevar a cabo de manera óptima el proceso de la toma de decisiones. La actual tendencia en diversas áreas académicas está orientada hacia la creación de conciencia ambiental, como puede constatarse en trabajos como (Mouzon, Yildirim y Twomey 2007; Rahimifard, Seow y Childs 2010; Cucala y col. 2012), en donde los autores ponen en manifiesto la gran importancia de la problemática energética no solo como un problema ambiental, sino como una alternativa sustancial para ahorrar costes y alcanzar una ventaja competitiva en las empresas. En particular, en la última década, las soluciones ecológicamente eficientes han tomado una posición relevante en las empresas debido a las presiones de las regulaciones gubernamentales, activistas de la comunidad, la competencia global y las organizaciones no gubernamentales.

Esta problemática no es ajena al área de la programación de proyectos, la cual al estar compuesta de problemas cuyo objetivo es asignar apropiadamente recursos disponibles a las actividades, tienen una gran importancia en la industria debido al amplio espectro de aplicaciones en diferentes campos como producción, distribución, trasporte, administración de proyectos y optimización de la cadena de suministros, por nombrar algunos. Por esta razón, el esfuerzo de los investigadores actuales está enfocado en el análisis y desarrollo de nuevos métodos que asignen de manera óptima los recursos para minimizar tanto objetivos industriales (tiempo, costes, etc.), como objetivos medioambientales (consumo energético, contaminación, etc.).

En este capítulo se aborda uno de los problemas más generales de la programación de proyectos: el RCPSP (véase subsección 2.2.1) para proponer una extensión del problema denominado el MRCPSP-ENERGY, el cual considera que las actividades pueden ser ejecutadas consumiendo diferentes cantidades de energía, dando origen a diversos modos de ejecución. El objetivo es maximizar la eficiencia energética, que tiene en cuenta tanto el *makespan* como el consumo energético. Adicionalmente, se propone una extensión de la librería estándar PSPLIB para generar casos de prueba del MRCPSP-ENERGY. Para este fin, se propone un modelo matemático realista que relaciona el consumo energético con el tiempo de procesamiento de las actividades. La nueva librería, denominada PSPLIB-ENERGY está disponible en: `http://gps.webs.upv.es/psplib-energy/`.

Este capítulo está estructurado de la siguiente forma: en la sección 4.2 se describe y propone el MRCPSP-ENERGY. En la sección 4.3 se expone el método propuesto para la creación de la librería PSPLIB-ENERGY. En la sección 4.4 se presenta un ejemplo de un caso de prueba del MRCPSP-ENERGY. Finalmente, en la sección 4.5 se exponen las conclusiones del capítulo.

4.2 Propuesta del MRCPSP-ENERGY

Esta investigación se ha enfocado en el RCPSP considerando solo recursos renovables para proponer la variante denominada MRCPSP-ENERGY (Morillo Torres, Barber y Salido 2017). La razón de partir de un problema uni-modal para proponer uno multi-modal, es que se incorpora un nuevo recurso, el consumo de energía, que permite el cambio de los tiempos de ejecución de las actividades, manteniendo el uso constante de los otros recursos. Inicialmente, para realizar la propuesta, se analizará la relación entre la energía consumida y el *makespan* en problemas de programación de proyectos.

Se asume que la energía es un recurso que es consumido por las actividades. Como usualmente ocurre en los procesos de producción de manufactura, el consumo de energía de una actividad es independiente de cuándo se programa la actividad (Oberg y col. 2004). Generalmente, cuanto mayor sea el consumo de energía de una actividad, menor será su tiempo de procesamiento. Ejemplos de máquinas en la industria con este comportamiento son los tornos, las fresadoras, los elevadores, los vehículos de material rodante, entre muchos otros (Li, Yan y Xing 2013; Fang y col. 2011). Por esta razón, asumiremos un comportamiento similar para las actividades del problema a proponer. Finalmente, se define la cantidad de energía total consumida por un proyecto ($CETP_w$) en la Ecuación 4.1, donde e_{im} es el consumo de energía de la actividad i ejecutada en el modo $m \in M_i$.

$$CETP_w = \sum_{i=1}^{n} e_{im},\ m \in \{1, 2, .., M\} \tag{4.1}$$

4.2.1 Eficiencia de un proyecto: el criterio de optimización en el MRCPSP-ENERGY

En esta sección se propone un criterio de optimización basado en la eficiencia energética. El MRCPSP-ENERGY requeriría el uso de algún método de optimización multi-objetivo que minimice de forma simultánea el *makespan* y el consumo de energía. En la literatura, siguiendo la clasificación presentada por Ballestín y Blanco (2011), cuando se hace frente a problemas con más de un objetivo, estos pueden ser abordados principalmente de tres maneras, las cuales dependen de la manera en que el tomador de decisiones (decision maker, DM) interviene en el proceso de búsqueda:

- Cuando la decisión se toma antes de la búsqueda: la información provista por el DM permite la generación de una función con un solo objetivo, la cual

es, generalmente, una función lineal ponderada de los objetivos originales. Por ejemplo, supongamos un problema cuyas variables se representan en un vector de solución x pertenecientes al espacio de soluciones. Además, supongamos que dicho problema tiene dos funciones objetivos $(F_1(x), F_2(x))$ cuyos valores asociados a cada objetivo de una solución (x) son f_1 y f_2 y el DM tiene unos pesos relacionados con cada objetivo (α y $1-\alpha$). La búsqueda puede hacerse creando una nueva función objetivo (F^*) como se muestra en la Ecuación 4.2.

$$F^*(f_1(x), f_2(x)) = \alpha * f_1(x) + (1-\alpha) * f_2(x) \tag{4.2}$$

Cuando los objetivos tienen escalas diferentes es usual normalizarlos, esto se realiza basándose en los valores f_w^{Nadir} y f_w^{Utopia} (peor y mejor) que puede alcanzar cada objetivo w. En la Ecuación 4.3 se muestra un ejemplo de una función estandarizada para un problema de minimización.

$$F^*(f_1(x), f_2(x)) = \alpha * \frac{f_1(x) - f_1^{Utopia}}{f_1^{Nadir} - f_1^{Utopia}} + (1-\alpha) * \frac{f_2(x) - f_2^{Utopia}}{f_2^{Nadir} - f_2^{Utopia}} \tag{4.3}$$

- Cuando la decisión se toma después de la búsqueda: en este caso la búsqueda conduce a la obtención de las llamadas soluciones no dominadas. Una solución x_1 domina a otra solución x_2 cuando se cumplen dos condiciones: la primera, cuando la solución x_1 no tiene un peor valor que x_2 respecto a todos los objetivos; y la segunda, cuando la solución x_1 es estrictamente mejor que x_2 en al menos un objetivo. Si alguna de las condiciones es violada, se dice que la solución x_1 no domina a la solución x_2. De esta manera, una solución no dominada es aquella que no es dominada por ningún otra solución, este conjunto de soluciones componen la denominada frontera de Pareto. Finalmente, el DM selecciona una solución perteneciente a esta frontera.

- Cuando la decisión se toma durante la búsqueda: la optimización debe dividirse en varios pasos, después de cada uno de ellos se presenta una serie de alternativas con diferentes relaciones de costo/beneficio al DM y este suministra una información basada en dichas alternativas que conducen la búsqueda del siguiente paso.

En cada enfoque de solución descrito anteriormente es el DM quien debe seleccionar una solución basándose en su experticia o sus intereses particulares. Sin embargo, no se obtiene una *mejor* solución, en su lugar se obtiene un conjunto de soluciones de las cuales se debe elegir una, o bien se obtiene una solución cuya búsqueda ha sido

guiada considerando la importancia relativa de los objetivos; incluso las soluciones obtenidas varían dependiendo del DM. En contraste, en este capítulo se presenta una nueva función objetivo para el MRCPSP-ENERGY basada únicamente en el concepto de eficiencia usado en la física que busca la mejor solución en términos de eficiencia energética. De esta manera, este nuevo criterio de optimización es una alternativa para hacer frente de manera independiente al DM a un problema de programación de proyectos que considera minimizar tanto la duración total del proyecto como el consumo de energía.

De manera general, el concepto de eficiencia puede ser definido como la relación entre la energía suministrada a un proceso y la energía transformada que este libera. Por ejemplo, una máquina en manufactura que usa un motor eléctrico, como un torno, un taladro, una fresadora, etc., tiene una eficiencia teórica definida como la conversión de energía eléctrica en energía mecánica. Esta relación se muestra en la Ecuación 4.4, donde η es la eficiencia, $P_{mechanical}$ representa la energía de salida del eje del motor y, $P_{electrical}$ es la energía absorbida (Boglietti y col. 2004).

$$\eta = \frac{P_{mechanical}}{P_{electrical}} = \frac{P_{mechanical}}{P_{mechanical} + \sum losses} \tag{4.4}$$

La Ecuación 4.4 es una eficiencia teórica, la cual no puede alcanzar un valor de 1 (100 %) debido a que durante la transformación de energía eléctrica en energía mecánica existen *pérdidas* de energía (*losses*), la mayoría en forma de calor. Estas pérdidas tienden a incrementar cuando se aumenta la energía eléctrica sobre la máquina (Saidur 2010). Por lo tanto, la curva de eficiencia está caracterizada por un crecimiento inicial hasta que alcanza una asíntota horizontal, como se puede observar en la figura 4.1.a. Las diferentes curvas de energía de diversos procesos o máquinas mantienen esta tendencia, variando acorde a sus especificaciones técnicas. El comportamiento de la eficiencia teórica difiere levemente de la real, puesto que la curva de eficiencia real suele decrecer tras alcanzar su máximo. Por ejemplo, en la figura 4.1.b se muestra la eficiencia real de un motor eléctrico de $4kw$ tomado de (Boglietti y col. 2004).

Cuando la energía es administrada a un motor, la salida del sistema es energía mecánica. Por lo tanto, la salida del sistema para un proyecto en el que las actividades son realizadas por máquinas o personas podría ser la duración de las actividades, que es afectada por la cantidad de energía suministrada. Cuanta más energía, menor será el tiempo de duración hasta que se alcance el tiempo mínimo posible, en tal situación el suministro de más energía ocasionaría que la máquina opere más despacio debido a la sobrecarga, como ocurre en la figura 4.1.b, o en el peor de los casos que colapse. La Ecuación 4.5 describe este comportamiento

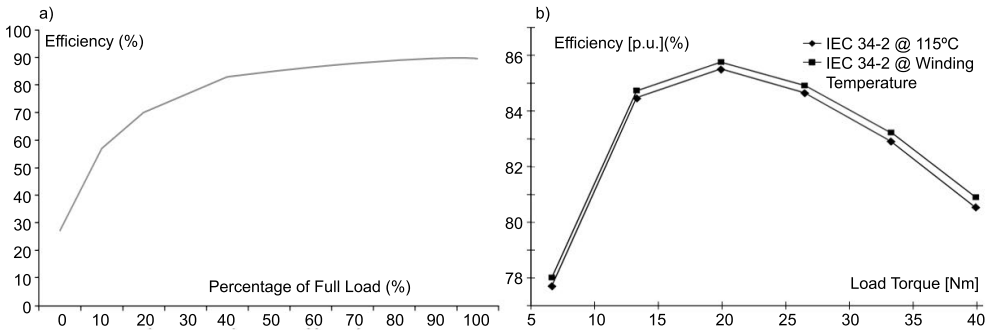

Figura 4.1: Eficiencia teórica (a) y real (b) de un motor eléctrico.

decreciente que relaciona la duración de una actividad d_i y su consumo de energía e_i. Sin embargo, el rango de esta expresión no se encuentra necesariamente entre 0 y 1, como sí lo hace una expresión que represente la eficiencia.

$$\eta_i^{'}(e_i) = \frac{1/d_i(e_i)}{e_i} \tag{4.5}$$

El concepto previo puede extenderse a un proyecto entero, como se muestra en la Ecuación 4.6. No obstante, como esta relación también está fuera del intervalo $[0-1]$, la Ecuación 4.6 no puede ser considerada como una función de eficiencia. Si los valores óptimos para el *makespan* y para el consumo total de energía fueran conocidos, dicha expresión podría ser estandarizada, pero estos valores óptimos son desconocidos. Sin embargo, pueden ser aproximados mediante el cálculo de dos cotas inferiores: $emin_w$ y $LB0min_w$ respectivamente. El primero es calculado como $(\sum_{i=1}^{n} e_{i1})$ donde e_{i1} es el mínimo consumo de energía de la actividad i. La segunda cota, es calculada usando el método del camino crítico, considerando el mínimo tiempo de procesamiento para cada actividad i dentro de sus $m \in M_i$ modos. De esta manera, proponemos una cota superior de rendimiento del proyecto (CSR_w), la cual puede calcularse mediante la Ecuación 4.7, con el objetivo de estandarizar la Ecuación 4.6. Esta cota superior puede ser interpretada como la eficiencia ideal de un proyecto.

$$\eta_w^* = \frac{1/makespan_w}{CETP_w} \tag{4.6}$$

Finalmente, la Ecuación 4.8 muestra la estandarización de la Ecuación 4.6 mediante la división de la Ecuación 4.7. Así, $\eta_w(makespan_w, CETP_w)$ es definida como la eficiencia relativa de un proyecto w con respecto a su valor ideal.

$$CSR_w = \frac{1/LB0min_w}{emin_w} \tag{4.7}$$

$$\eta_w(makespan_w, CETP_w) = \frac{1}{CSR_w} * \frac{1/makespan_w}{CETP_w} \tag{4.8}$$

Como se ha mencionado anteriormente, el consumo de energía de un proyecto es la suma de los consumos de energía de sus actividades. Sin embargo, es importante remarcar que la eficiencia de un proyecto no es igual a la suma de las eficiencias de sus actividades. La razón es que el *makespan* (que representa el inverso del término $P_{mechanical}$) de un proyecto no es la suma de las duraciones de sus actividades.

Basándose en todos los conceptos desarrollados en este capítulo, el MRCPSP-ENERGY se define de la siguiente manera:

Definición 1: el MRCPSP-ENERGY es un proyecto que consta de un conjunto I de n actividades $I = \{1, ..., i, ..., n\}$, un conjunto B de K^ρ recursos renovables compartidos $B = \{1, ...b, ...K^\rho\}$, y existe una cantidad limitada R_b^ρ de cada recurso. Cada actividad i tiene M_i modos de ejecución y un tiempo no interrumpible de d_{im} que depende el modo $m \in M_i$, estas requieren para su realización un total de r_{ib} unidades de recursos renovables de cada tipo K^ρ y una cantidad de energía e_{im}. Adicionalmente, las actividades están sujetas a un conjunto de restricciones de precedencia, las cuales indican que una actividad no puede ser ejecutada sin antes haber finalizado todas sus actividades predecesoras. Los diferentes consumos de energía para una actividad dan origen a los modos de ejecución. Así, el MRCPSP-ENERGY es similar al RCPSP, donde las actividades tienen diferentes modos asociados al consumo de energía.

Las principales diferencias respecto al MRCPSP son las siguientes: (1) los modos dependen completamente del consumo de energía; (2) La relación entre el consumo de energía y el tiempo de procesamiento es inverso; y (3) el objetivo es maximizar la eficiencia del proyecto, que implica la minimización simultánea del *makespan* y el consumo energético.

La formulación matemática para el MRCPSP-ENERGY mantiene las restricciones del MRCPSP excepto la restricción de los recursos no renovables y se sustituye la función objetivo por la maximización de la Ecuación 4.8. Así, el modelo completo

se muestra en las Ecuaciones 4.9 a 4.13, manteniendo la misma notación usada en la subsección 2.2.2.

$$Maximizar : \eta_w(makespan_w, CETP_w) \tag{4.9}$$

Sujeto a:

$$\sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} \xi_{imt} = 1 \quad \forall\, i \in I \tag{4.10}$$

$$\sum_{m=1}^{M_i} \sum_{t=es_j}^{ls_j} t * \xi_{jmt} \geq \sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} (t + d_{im}) * \xi_{imt} \quad \forall\, i \in P_j,\ \forall\, j \in I \tag{4.11}$$

$$\sum_{i=0}^{n} r_{ib}^{\rho} * \sum_{m=1}^{M_i} \sum_{s=max\{t-d_{im}, es_i\}}^{min\{t-1, ls_i\}} \xi_{ims} \leq R_b^{\rho} \quad \forall\, b \in B;\ t = 1, ..., T_{max} \tag{4.12}$$

$$\xi_{imt} \in \{0, 1\} \tag{4.13}$$

4.3 Propuesta de la librería PSPLIB-ENERGY

En esta sección, se propone la librería PSPLIB-ENERGY (Morillo Torres, Barber y Salido 2014). Esta complementa la librería estándar PSPLIB (véase sección 2.7) con diferentes niveles de consumo energético y tiempos de procesamiento asociados a cada actividad. Esta extensión pretende ser usada como banco de casos de prueba del MRCPSP-ENERGY para realizar comparaciones de desempeño de los métodos de solución que se desarrollen.

4.3.1 Propuesta de un modelo para el consumo de energía de una actividad

Con el fin de expandir la librería PSPLIB, los valores que se asignen en los consumos de energía y duraciones para cada actividad deben ser consistentes con el comportamiento real que tienen las máquinas. Para esto, primero se asigna un valor de consumo de energía estándar e_i y una duración estándar d_i a cada actividad. Luego, un modelo matemático relaciona los valores estándares con cada modo para calcular los nuevos valores de consumo de energía y duración.

En el área de maquinado no existe un modelo matemático global que describa todos los comportamientos de consumos de energía para las máquinas, esto es debido al hecho de que el consumo de energía depende de diversos factores técnicos. Sin embargo, el comportamiento es bastante similar para la mayoría de máquinas. Oberg y col. (2004) muestran que la relación entre el consumo de energía y el tiempo de procesamiento de maquinado tiene una tendencia decreciente. De esta manera, proponemos la Ecuación 4.14, donde $t_i(c_i)$ es la proporción de tiempo de procesamiento con respecto a la duración estándar de la actividad i y c_i es la proporción del consumo de energía comparada con el consumo estándar de la actividad i. La Ecuación 4.14 tiene una tendencia decreciente de manera similar al comportamiento energético en las máquinas. Así, esta función representa una aproximación del modelo de eficiencia propuesto en la sección anterior. Los valores de las constantes $A_1 = 4{,}0704$ y $A_2 = 2{,}5093$ centran la función en $c_i = 1(100\,\%)$ y $t_i = 1(100\,\%)$, representando el valor estándar de consumo de energía y duración, respectivamente. Como ejemplo, un valor de $c_i = 1,6(160\,\%)$ representa un consumo adicional del 60 % respecto al estándar de la actividad i, y el valor obtenido $t_i = 0,67(67\,\%)$ representa que, con ese consumo adicional, el tiempo de ejecución se reduce en un 33 %. La figura 4.2 muestra la gráfica de la Ecuación 4.14.

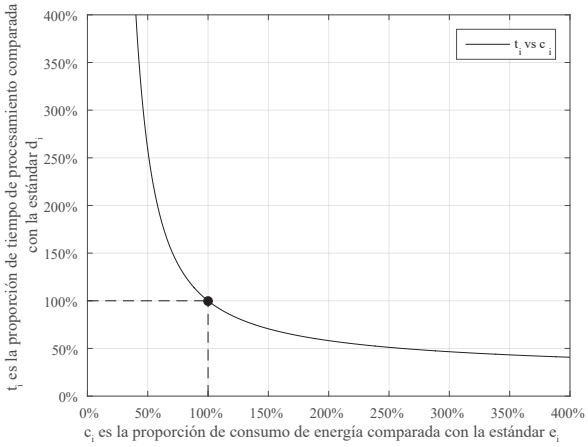

Figura 4.2: La proporción de duración con relación a la proporción de consumo de energía.

$$t_i(c_i) = \frac{A_1 * ln(2)}{ln(1 + (c_i * A_2)^3)} \tag{4.14}$$

Como resultado del modelo previo, se define la expresión $\eta_i(c_i)$ (Ecuación 4.15) como la eficiencia relativa de la actividad i en función de c_i. Esta expresión puede ser interpretada como la eficiencia porcentual de la actividad i respecto a su eficiencia estándar (cuando se usa el consumo de energía y la duración estándar). Por ejemplo, la expresión $\eta_i(c_i) = 130\,\%$ indica que, dada una proporción de consumo de energía c_i, se obtiene una eficiencia del 30 % mayor respecto a la eficiencia estándar. La figura 4.3 muestra la gráfica de la eficiencia relativa (Ecuación 4.15). Es importante notar que la tendencia de la curva es similar a la eficiencia real (véase subsección 4.2.1).

$$\eta_i(c_i) = \frac{1/t_i(c_i)}{c_i} = \frac{ln(1 + (c_i * A_2)^3)}{c_i * A_1 * ln(2)} \tag{4.15}$$

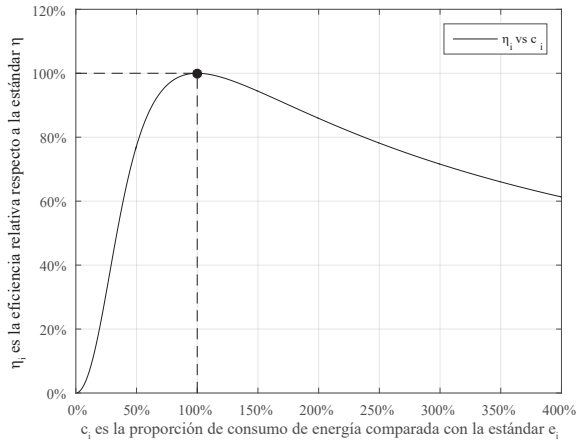

Figura 4.3: Eficiencia relativa en función de la proporción del consumo de energía.

4.3.2 Extensión energética para la librería PSPLIB

Para realizar la expansión de la librería PSPLIB de problemas del RCPSP al MRCPSP-ENERGY, se incluyeron tiempos de procesamiento y consumos de energía a cada actividad para cada caso de prueba del RCPSP, a través del modelo matemático descrito en la subsección 4.3.1. Por lo tanto, se crean cuatro conjuntos de problemas: $j30$, $j60$, $j90$, y $j120$, cada uno de ellos con 30, 60, 90, y 120 actividades, respectivamente.

Sin perder generalidad, se especifican tres valores diferentes para el consumo de energía e_{im}. De esta manera, son definidos tres modos: el primero ($c_{i1} = 0{,}8$) corresponde a disminuir en un 20 % el consumo de energía respecto a su valor estándar e_{i2}; el segundo modo ($c_{i2} = 1$) representa el modo estándar, con un consumo de energía estándar $e_i^{std} = e_{i2}$, y una duración estándar $d_i^{std} = d_{i2}$ que es proveída por el valor original de duración en la librería PSPLIB; el tercer modo ($c_{i1} = 1{,}2$) corresponde a un incremento del 20 % de la energía consumida respecto al valor estándar e_{i2}.

Los valores correspondientes al tiempo de procesamiento (d_{im}) y consumo de energía (e_{im}) son calculados mediante las Ecuaciones 4.16 y 4.17. Los valores de $t_i(c_{im})$ son calculados usando la Ecuación 4.14.

$$d_{im} = t_i(c_{im}) * d_i^{std}, \;\; i \in I, \; m \in \{1, 2, 3\} \tag{4.16}$$

$$e_{im} = c_{im} * e_i^{std}, \;\; i \in I, \; m \in \{1, 2, 3\} \tag{4.17}$$

Los valores de e_i^{std} son definidos aleatoriamente dentro de un intervalo de $[1, 10]$ para cada actividad de un proyecto. Este rango fue propuesto considerando el mismo intervalo usado para los parámetros originales en la librería PSPLIB (véase sección 2.7). Adicionalmente, como todos los parámetros son valores enteros en la librería PSPLIB, los valores de los parámetros en la librería propuesta también lo son. Cabe notar que una actividad i con una duración de $d_{im} = 1$ no puede ser reducida, de manera similar una actividad con un consumo de energía $e_{im} = 1$ no puede ser reducido. Las aproximaciones en los otros casos fueron realizadas teniendo en cuenta el modo $m \in M_i$ correspondiente. Entonces, para $m = 3$, d_{im} es aproximado hacia el entero más cercano hacia abajo y e_{im} es aproximado hacia el entero más cercano hacia arriba. En el caso del modo $m = 1$, d_{im} es aproximado hacia el entero más cercano hacia arriba, y e_{im} es aproximado al entero más cercano hacia abajo.

Para la evaluación de futuros métodos para resolver el MRCPSP-ENERGY usando la librería PSPLIB-ENERGY, se requiere estimar la cota superior CSR_w (Ecuación 4.7) para calcular el criterio de evaluación del problema, es decir la eficiencia relativa (Ecuación 4.8). Los valores de $LB0min_w$ y $emin_w$, que permiten la estimación de CSR_w, están disponibles en la librería PSPLIB-ENERGY.

El valor promedio de eficiencia $\bar{\eta}$ para cada conjunto de problemas, puede ser calculado mediante la Ecuación 4.18, donde nP es en número de proyectos en cada conjunto.

$$\bar{\eta} = \frac{\sum_{w=1}^{nP} \eta_w(makespan_w, CETP_w)}{nP} \tag{4.18}$$

4.4 Ejemplo de un caso de prueba del MRCPSP-ENERGY

En esta sección se muestra ejemplo ilustrativo de un caso de prueba de un MRCPSP-ENERGY. Este consiste de un proyecto con $n = 8$ actividades, $M_i = 3 \; \forall \; i \in I$ modos de ejecución y $K^{\rho} = 3$ tipos de recursos renovables, cada uno con una capacidad máxima de $R_b^{\rho} = 3 \; \forall \; b \in B$. La tabla 4.1 presenta los consumos de recursos (e_{im}), la duración de cada actividad (d_{im}), y el uso de recursos (r_{ib}). La primera columna muestra las actividades (la actividad 1 y 8 son ficticias con duración y consumo cero). De la segunda a la séptima columna se muestran las duplas de valores (d_{im}, e_{im}) para cada modo. Los valores correspondientes a d_{i2} y e_{i2} (columnas en negrita) son las duraciones y consumos estándares para cada actividad. De la octava a la décima columna se muestran el uso de los recursos renovables.

Tabla 4.1: Consumo de energía (e_{im}), duración (d_{im}), y uso de recursos (r_{ib}) para el ejemplo presentado en la sección 4.4.

Actividad	d_{i1}	e_{i1}	d_{i2}	e_{i2}	d_{i3}	e_{i3}	r_{i1}	r_{i2}	r_{i3}
1	0	0	**0**	**0**	0	0	0	0	0
2	4	4	**3**	**5**	2	6	1	1	1
3	3	1	**2**	**2**	1	3	2	1	1
4	2	5	**1**	**7**	1	7	1	0	1
5	7	3	**5**	**4**	4	5	0	1	1
6	3	4	**2**	**6**	1	8	2	1	0
7	8	4	**6**	**6**	5	8	2	1	2
8	0	0	**0**	**0**	0	0	0	0	0

La figura 4.4 muestra un grafo que representa las relaciones de precedencia y la información provista en la tabla 4.1.

En la figura 4.5 se muestran tres soluciones factibles para el problema ejemplo. La figura 4.5.a muestra la solución óptima del problema relajado, considerando solo el modo estándar $m = 2$. La figura 4.5.b y la figura 4.5.c muestran soluciones usando diferentes consumos de energía y, consecuentemente, diferentes tiempos de procesamiento.

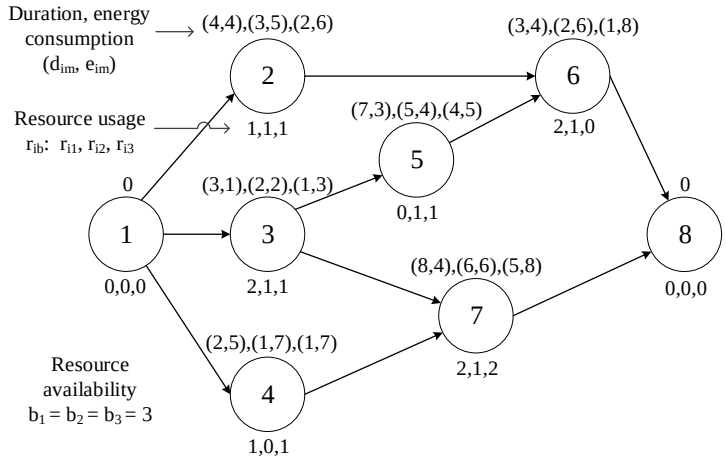

Figura 4.4: Un grafo del problema ejemplo del MRCPSP-ENERGY.

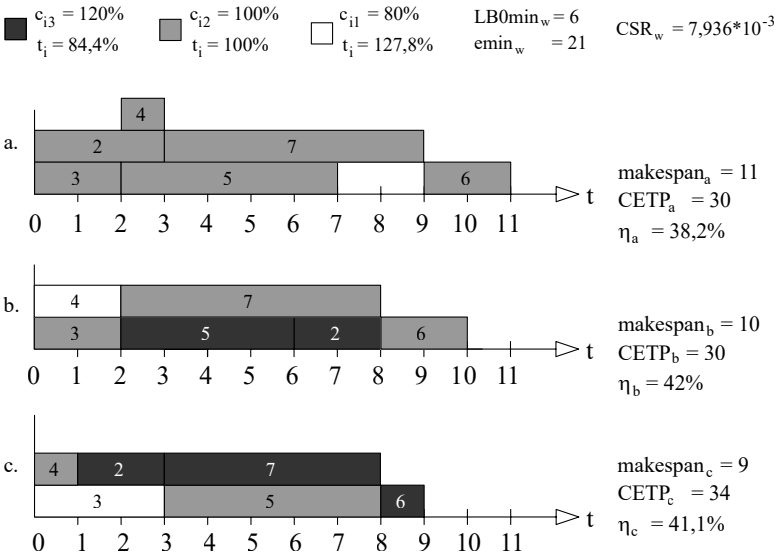

Figura 4.5: Tres soluciones factibles para el problema ejemplo del MRCPSP-ENERGY.

Para calcular la eficiencia η_w de cada solución, primero se estimó $LB0_w = 6$ y $emin_w = 21$. Así, se determina la cota superior de rendimiento del proyecto ($CSR_w = 7{,}936 * 10^{-3}$). De esta manera se calcula η_w usando la Ecuación 4.8. En la figura 4.5, se puede apreciar al lado izquierdo de cada solución, el valor tanto del *makespan*, como del consumo total de energía $CETP_w$ y la eficiencia η_w correspondiente.

La solución a tiene el mejor valor de eficiencia $\eta_a = 38,2\,\%$. Por otra parte, el mejor (menor) *makespan* es alcanzado por la solución c, con una eficiencia de $\eta_c = 41,1\,\%$, por tanto, se puede considerar que la solución c es mejor que la solución a. Sin embargo, la solución b mantiene el mismo consumo de energía que la solución a pero mejora su *makespan*. Mientras que respecto a la solución c, esta mejora en una unidad el *makespan* a cambio de 4 unidades de energía. A pesar de ello, la solución b es la que alcanza la mayor eficiencia con un valor de $\eta_b = 42\,\%$, considerándose la mejor de ellas.

Cabe mencionar que los valores de eficiencias son relativamente bajos en todos los casos (cercanos al 40 %), esto se debe a que son comparadas con la cota superior de rendimiento del proyecto CSR_w, la cual nunca podrá ser alcanzada, pues se calcula con base en el problema relajado con las mínimas duraciones y mínimos consumos de energía.

Con el fin de comparar diferentes enfoques multi-objetivo, a pesar de ser un ejemplo muy pequeño, se puede construir la frontera de Pareto, como se muestra en la figura 4.6. En el eje horizontal se muestra el *makespan* de las soluciones, y el eje vertical muestra el consumo de energía total del proyecto. Adicionalmente, se muestra la ubicación de las soluciones en este gráfico, como puede observarse la solución c y la solución a son soluciones dominadas, es decir se puede encontrar al menos una solución que tenga un mejor valor en sus dos objetivos al mismo tiempo, mientras que la solución b es una no dominada, por tanto, no existe ninguna solución que pueda mejorar el valor en los dos objetivos simultáneamente. Como se mencionó anteriormente en este capítulo, la frontera de Pareto no encuentra la mejor solución, sino que nos suministra un conjunto de soluciones no dominadas y debe ser el decisor quien elija la que considere mejor. Finalmente, también puede apreciarse la solución utópica que consiste en la dupla de valores obtenidos por las soluciones al optimizar cada objetivo por separado.

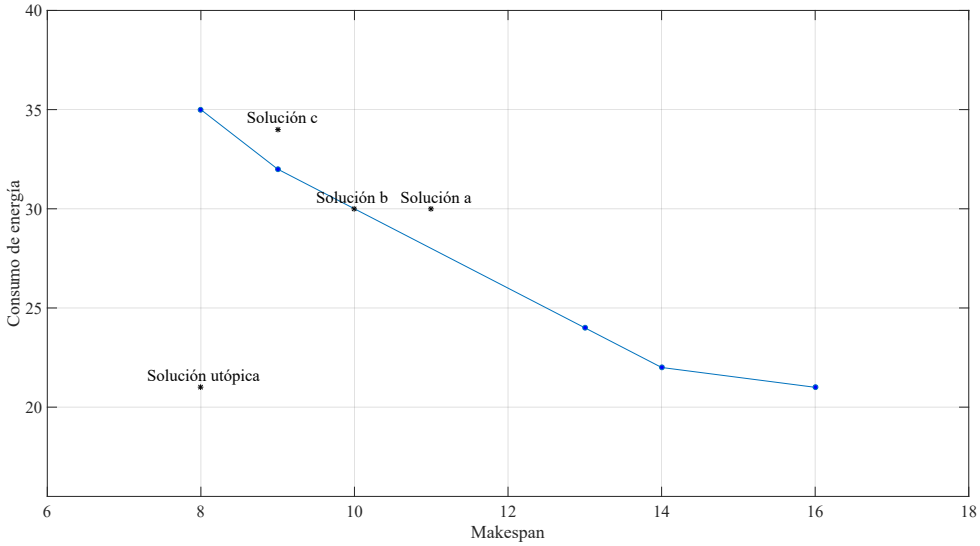

Figura 4.6: Frontera de Pareto para el ejemplo de la figura 4.4.

4.5 Conclusiones

Actualmente, muchas investigaciones están siguiendo la fuerte tendencia hacia el desarrollo de modelos que incluyan un componente energético para obtener soluciones sostenibles en problemas de programación de proyectos. Como resultado, es necesario también, el desarrollo de nuevas liberarías de casos de prueba para evaluar los métodos que se desarrollen para resolver estos problemas.

En este capítulo se ha propuesto una variante del RCPSP basándose en la eficiencia energética, denominado MRCPSP-ENERGY, el cual incluye un consumo de energía a un problema de programación de proyectos. El problema propuesto tiene un enfoque bi-objetivo que consiste en maximizar la eficiencia relativa de un proyecto. Esto se lleva a cabo minimizando simultáneamente el *makespan* y el consumo de energía. Este concepto de eficiencia unifica los dos objetivos en uno, para tratar de encontrar la mejor solución para el sistema en conjunto, proporcionando una alternativa independiente del decisor a los enfoques multi-objetivo tradicionales.

El MRCPSP-ENERGY está basado, principalmente, en sistemas de manufactura, donde el consumo de energía está asociado a cada actividad, siendo independiente del punto en el tiempo en que se programe y de los recursos que se utilicen. Por ejemplo, el aumento energía en un torno para acelerar la velocidad de remoción de material manteniendo constante el número de operadores de la máquina (recursos).

De esta manera, cada valor de consumo energético contiene la totalidad de energía usada para realizar una actividad.

El criterio propuesto de eficiencia relativa toma la definición de eficiencia en la física (la relación entre energía obtenida y energía suministrada) y la traslada al campo de programación de proyectos, definiendo la eficiencia para una actividad y extrapolando su concepto para ser aplicado a todo un proyecto. Así, la inclusión de un consumo energético en las actividades, que da lugar a los modos de ejecución, permite alcanzar soluciones energéticamente eficientes.

Adicionalmente, se propone la librería PSPLIB-ENERGY, que consiste en una extensión de la librería estándar PSPLIB del RCPSP y el MRCPSP, para la evaluación de nuevos métodos de solución para el MRCPSP-ENERGY. Esta extensión está soportada por un modelo propuesto de consumo de energía, el cual es consistente con los estudios reportados en la literatura académica acerca del consumo energético de las máquinas.

La librería PSPLIB-ENERGY se ha desarrollado manteniendo el mismo formato que la actual PSPLIB, con cuatro conjuntos de problemas ($j30$, $j60$, $j90$, y $j120$), cada uno con 480 problemas, (excepto el conjunto $j120$, que tiene 600 problemas). La librería está disponible en http://gps.webs.upv.es/psplib-energy/.

Capítulo 5

Obtención de soluciones energéticamente eficientes: una nueva propuesta

En este capítulo se aborda el problema propuesto en la sección 4.2: el MRCPSP-ENERGY. Para esto, primero se diseña y desarrolla un método evolutivo basado en los elementos de los métodos más exitosos para resolver el MRCPSP y RCPSP, adaptados para ser implementados en el MRCPSP-ENERGY. El propósito de método es brindar un punto de partida para las comparaciones. Posteriormente, se analiza el problema de soluciones redundantes, presente en el RCPSP, para evaluar su impacto sobre la búsqueda en el MRCPSP-ENERGY. Finalmente, se propone un nuevo método evolutivo que mezcla dos enfoques para realizar la búsqueda: uno basado en los modos y el otro basado en las listas de actividades. Para hacer la evaluación y experimentación computacional, usamos la librería PSPLIB-ENERGY propuesta en la sección 4.3, las comparaciones se realizan sobre los métodos propuestos, y también sobre un enfoque de resolución exacto. Los resultados muestran que el método propuesto alcanza soluciones altamente eficientes.

5.1 Introducción

Como se expresó en el Capítulo 4, la propuesta de un problema de programación de proyectos con recursos restringidos que incorpore un consumo de energía en sus actividades, se debe a la fuerte tendencia, tanto de la industria como de la academia, de hacer frente a los nuevos requerimientos relacionados con el impacto ambiental y la productividad. Además, como se concluyó en el mismo capítulo, el considerar el consumo de energía en la asignación de recursos como problema de toma de decisiones no solo permite dar solución a los problemas ambientales, sino también es una gran oportunidad para realizar los procesos de manera más eficiente.

En virtud de lo anteriormente expuesto, en este capítulo se proponen diferentes métodos de solución al MRCPSP-ENERGY (véase sección 4.2). El primero consiste en un algoritmo metaheurístico basado en los principales elementos de los métodos más exitosos en la literatura académica para resolver el MRCPSP y el RCPSP, cuyo objetivo es comprobar si los métodos de búsqueda para estos problemas siguen siendo vigentes en la variante energética propuesta, de igual modo busca establecer un punto de partida para realizar futuras comparaciones. El segundo método, está basado en un método exacto de programación lineal entera mixta que es resuelto mediante el software especializado *IBM CPLEX CP Optimizer*, el cual usa *constraint programming* como método de solución debido a su gran eficiencia en problemas de combinatoria. El tercero, es un método de solución que plantea una alternativa de búsqueda a la comúnmente establecida por los métodos metaheurísticos de la literatura. Esta se sustenta en la presencia del problema de soluciones redundantes, que en la versión multi-modal tiene mayor impacto al tener un espacio de búsqueda más grande.

A lo largo de este capítulo se usará como ejemplo el caso de prueba de un MRCPSP-ENERGY mostrado en la figura 5.1, dicho problema consta de $n = 11$ actividades, $M_i = 3\ \forall i \in I$ modos de ejecución, $K^\rho = 3$ recursos renovables, cada uno de ellos con una disponibilidad máxima de $R_b^\rho = 4\ \forall\ b \in B$ unidades. El grafo dirigido muestra las relaciones de precedencias. Cada nodo representa una actividad, en la parte superior de cada nodo se encuentran las duplas (d_{im}, e_{im}) que contienen las duraciones y los consumos de energía dependiendo del modo de ejecución. En la parte inferior de cada nodo se encuentran los valores r_{ib}, que representan el uso de cada K^ρ recurso renovable. De esta manera, por ejemplo, la actividad 7 consume 1, 2, y 2 unidades para cada tipo de recurso renovable, esta actividad tiene 3 modos de ejecución: en el primero se consume la mínima cantidad posible de energía (6 unidades) correspondiente a la duración más lenta de 4 unidades de tiempo; en el segundo modo, el considerado estándar, se consumen 8 unidades

de energía con una duración asociada de 3 unidades; finalmente, el tercer modo consume 10 unidades de energía con la duración más rápida de 2 unidades de tiempo.

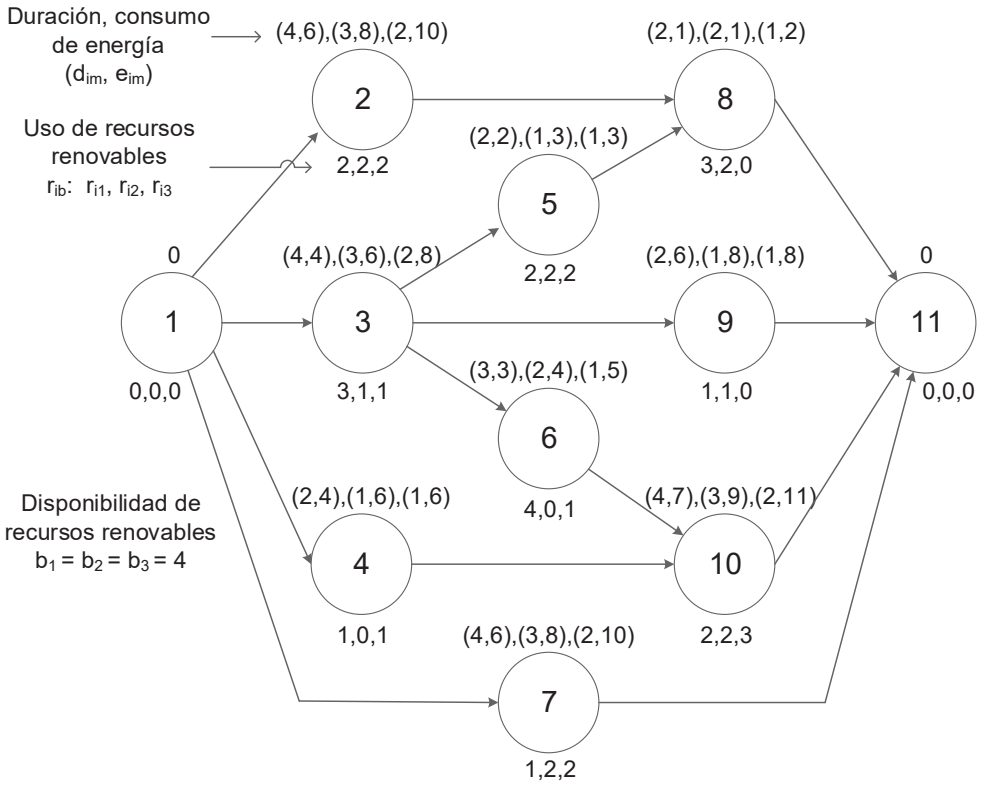

Figura 5.1: Ejemplo de un MRCPSP-ENERGY con 11 actividades.

El presente capítulo está estructurado de la siguiente manera: en la sección 5.2 se analiza la problemática de las soluciones redundantes en el MRCPSP-ENERGY. En la sección 5.3 se describe el algoritmo genético básico, adaptado de la literatura, que se usará como esquema general para implementar la nueva búsqueda. Esta última se describe en la sección 5.4, la cual está basada en la búsqueda a través de los modos de ejecución de las actividades. La sección 5.5 muestra la experimentación y evaluación computacional. Finalmente, la sección 5.6 se presentan las conclusiones del capítulo.

5.2 Análisis de las soluciones redundantes en el MRCPSP-ENERGY

El problema de las soluciones redundantes fue analizado en la sección 3.2 para el RCPSP. Sin embargo, este fenómeno también afecta al MRCPSP-ENERGY, puesto que la redundancia está ligada a la representación de las soluciones, como se mostró en dicho capítulo, tanto la representación de lista de actividades como la representación de valor clave podían dar a lugar soluciones redundantes. Recordemos que una solución redundante se define como una solución completa (decodificada) que tiene más de una representación asociada. En el MRCPSP-ENERGY además de la lista de actividades, también se usa una lista de modos, por tanto, cabe preguntarse si la redundancia afecta también a esta última representación y si el fenómeno de redundancia en las listas de actividades afecta en mayor o menor medida al MRCPSP-ENERGY. Así pues, en este capítulo se dará respuesta a estas preguntas.

Para decodificar una solución en el MRCPSP-ENERGY es necesario tanto una lista de actividades como una lista de modos. Posteriormente, dichas listas se usan en la implementación de un esquema generador de secuencias (SGS), que puede realizar la decodificación de 4 formas diferentes combinando un esquema en serie o en paralelo con una dirección hacia adelante o hacia atrás. En general, las metaheurísticas para resolver la versión multi-modal del RCPSP usan la representación de lista de actividades para generar nuevos vecindarios de soluciones y así realizar la búsqueda, mientras que la alteración de la lista de modos se usa principalmente para encontrar factibilidad e introducir diversidad mediante la mutación.

Para mostrar el fenómeno de mutación se adaptará la secuencia mostrada en la sección 3.2 (figura 3.1) a un problema MRCPSP-ENERGY. Para ello, reemplazaremos las actividades del ejemplo de la secuencia original por actividades del ejemplo del MRCPSP-ENERGY mostrado en la anterior sección (figura 5.1) y se define un modo de ejecución para cada actividad. Ahora, consideremos las listas de actividades $AL1 = \{1, 7, 3, 6, 2, 4, 5, 10, 8, 9, 11\}$ y $AL2 = \{1, 7, 3, 6, 9, 4, 2, 5, 10, 8, 11\}$ que comparten una lista de modos $LM = \{0, 1, 2, 1, \ 1, 1, 2, 3, 1, 2, 0\}$. Como se aprecia en la figura 5.2, al decodificar estas listas de actividades, la solución resultante es la misma para ambos casos, a pesar que las listas son diferentes. Para este ejemplo se usó un SGS en serie con dirección hacia adelante para la decodificación.

Existen dos principales causas del fenómeno de soluciones redundantes asociadas a la representación de lista de actividades: la primera, es la posibilidad de ejecutar un conjunto de actividades al mismo tiempo en un instante dado en una solución

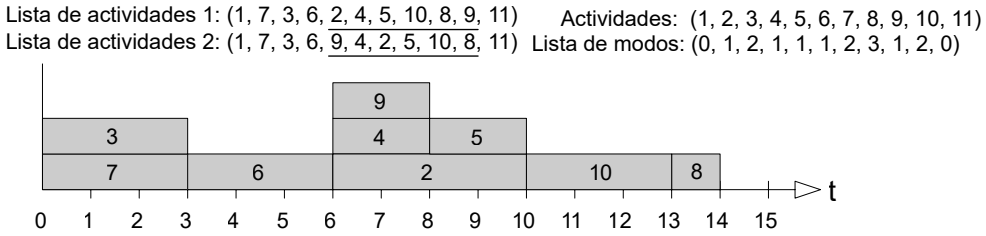

Figura 5.2: Ejemplo de soluciones redundantes en el MRCPSP-ENERGY.

parcial, estos conjuntos de soluciones varían a lo largo de la formación de una solución puesto que dependen de la estructura específica de la solución parcial a la que pertenecen. La segunda causa también está relacionada con una estructura específica de una solución parcial, y consiste en el otorgamiento de un grado de flexibilidad de programación a una actividad (véase sección 3.2). De esta manera, el espacio de soluciones para la lista de actividades, está definido por el conjunto de permutaciones de actividades. El cual se puede dividir en tres subconjuntos: soluciones no factibles, soluciones factibles, y soluciones únicas. De tratarse de una lista que sea factible por precedencias, la búsqueda se realiza solo en los dos últimos conjuntos. Estimar que tan grandes son estos conjuntos no es posible sin antes verificar cada permutación. Sin embargo, basados en los resultados de la sección 5.5, el número de soluciones redundantes es mucho mayor que el número de soluciones únicas.

Por otra parte, casi cualquier cambio en una lista de modos garantiza una solución real diferente (decodificada), incluso considerando que se mantenga constante una lista de actividades. De hecho, si una actividad i cambia su modo de ejecución, la duración de esa actividad también cambiará, lo cual provoca que la decodificación en serie o en paralelo tenga o no disponibles recursos en diferentes momentos de la programación, dependiendo si aumenta o reduce su duración. Además, las actividades sucesoras de la actividad i también estarán disponibles en instantes diferentes, puesto que dependen del tiempo de finalización de la actividad i. Como resultado, la decodificación puede construir nuevas combinaciones de actividades que puedan ser programadas de la siguiente manera: si la nueva duración para la actividad i es menor, entonces los cambios en la programación empezarán desde el nuevo tiempo de finalización de dicha actividad; si la nueva duración es mayor, entonces los cambios empezarán desde el antiguo tiempo de finalización. Así, si los modos de las actividades son diferentes, el espacio de búsqueda de soluciones factibles y el espacio de soluciones únicas son el mismo.

En la figura 5.3 se muestra un fragmento de un diagrama de Gantt que representa el final de una solución para un caso de prueba del MRCPSP-ENERGY con 30 actividades, donde cada unidad de tiempo es un día. Esta figura muestra que el cambio de un modo de ejecución de una actividad puede causar una reprogramación casi completa de las actividades, como se explicará a continuación. En la figura 5.3.a se muestra la solución original, donde se resalta la actividad número 4 en su modo estándar, pues será la que sufra una modificación en su modo de ejecución. Dicha actividad tiene 3 posibles modos, el primero con una duración de 2 días, el segundo es su modo estándar con una duración de 4 días y tercer modo con una duración de 6 días. Supongamos que la actividad 4 se ejecuta en su primer modo (figura 5.3.b), en este caso su menor duración permite que a partir del día 15 se encuentren disponibles recursos que antes estaban siendo usados por la misma actividad, y por tanto otra actividad, como la número 20 puede ser programada en los días 15 a 18. Sin embargo, la actividad 20 no pude ser ejecutada con las actividades 28 y 22, por lo que estas deben ser programadas en otro lugar dependiendo de las restricciones de recursos y precedencias (no solo desplazarlas). Al mismo tiempo, otras actividades que antes no fueron consideradas quizá puedan ser ejecutadas junto con la 20, así que la estructura de la solución luego del día 15 puede cambiar totalmente.

De manera similar, supongamos que la actividad 4 se ejecuta en su tercer modo (figura 5.3.c), en este caso el tiempo de finalización de esta actividad pasa de ser el día 15 al día 19, por lo tanto en ese intervalo de tiempo, mientras que la solución de la figura 5.3.a tiene disponible la cantidad de recursos usada por la actividad 4, la solución de la figura 5.3.c no tiene disponible esa cantidad, y las actividades 16 y 22 que usan dichos recursos no pueden ser programadas en el día 18, así que deben ser reprogamadas. Además, en su lugar puede que existan otras actividades que sí se puedan ejecutar junto con las actividades 4, 10 y 28.

Solo existen dos condiciones para que un cambio en los modos genere una solución distinta: la primera es que los modos en las actividades sean diferentes, es decir que cada modo tenga una duración diferente. La segunda condición es que existan otras actividades que se afecten por el uso de recursos, ya sea por la falta de ellos o por la disponibilidad. Por ejemplo si la actividad número 23 de la figura 5.3.a cambia de modo de tal manera que su tiempo de finalización sea el día 33 y suponiendo que las actividades 29, 30 y 23 puedan ejecutarse al mismo tiempo, no se generaría una solución nueva. De igual forma, si su duración disminuye en una unidad, al no haber más actividades que puedan hacer uso de dichos recursos, y al no ser la actividad responsable del *makespan* (las responsables en este caso son la actividad 30 y 29), no se generaría una nueva solución.

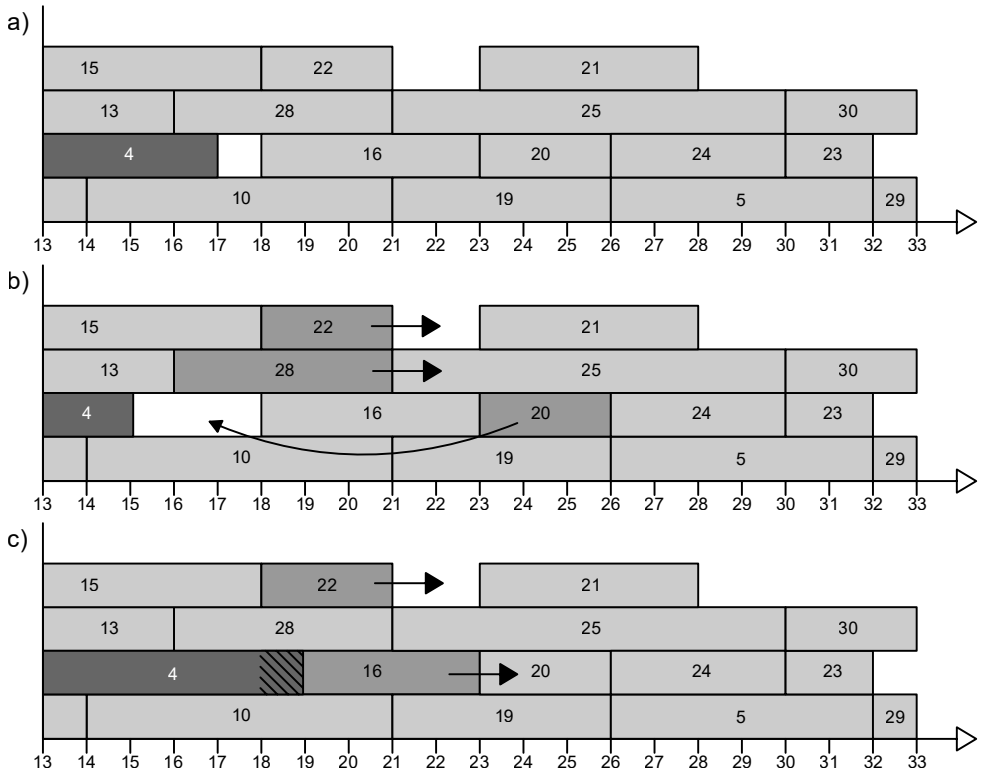

Figura 5.3: Reprogramación de una solución afectada por un cambio de modo de ejecución.

El espacio de soluciones para la representación basada solo en la lista de n actividades está acotada por el número de permutaciones $n!$. De manera similar, el espacio de soluciones para la representación basada en los modos de ejecución está acotada por el número de combinaciones m^n. Por supuesto, el espacio de permutaciones es mucho más grande que el espacio de combinaciones a medida que el número de actividades aumenta. La figura 5.4 muestra un esquema sobre la comparación entre el espacio de soluciones generadas por la búsqueda basada en la lista de actividades y basada en la lista de modos. En esta figura el espacio de soluciones de la lista de modos puede considerase como las soluciones generadas por la combinación de una lista de actividades fija.

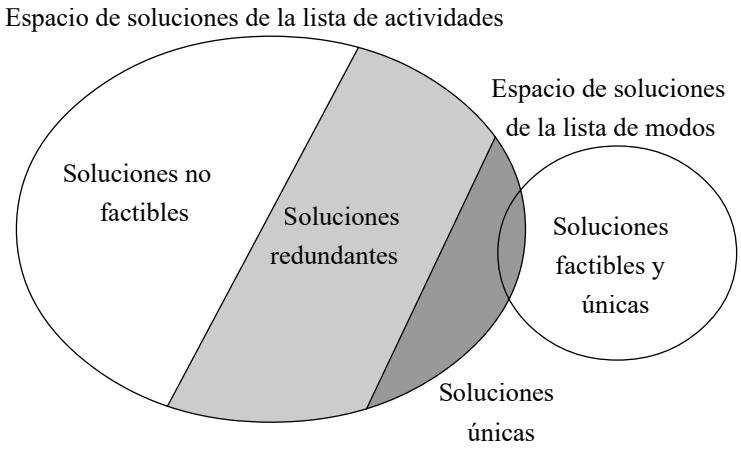

Figura 5.4: Esquema del espacio de soluciones generado por las listas de actividades y por las listas de modos.

Por lo tanto, el problema radica en el hecho de que el esfuerzo computacional realizado por un método de búsqueda basado solo en las permutaciones de la lista de actividades podría ser desperdiciado en soluciones redundantes.

5.3 Propuesta de un método evolutivo para la solución del MRCPSP-ENERGY

En esta sección se propone el denominado algoritmo genético básico (GA básico) para resolver el MRCPSP-ENERGY. El diseño del GA-básico consiste en la adaptación de los métodos más exitosos para las versiones uni-modales y multi-modales del RCPSP al MRCPSP-ENERGY. Como se concluyó en el Capítulo 2, las metaheurísticas con mejores resultados para ambos problemas son los algoritmos poblacionales y los algoritmos híbridos. Dentro de las representaciones de solución, Hartmann y Kolisch (2000) comprobaron empíricamente que la lista de actividades (AL) es la que obtiene mejores resultados. Respecto a los esquemas generadores de secuencia (SGS), no existe un método que supere al otro en todos los casos, se sabe que el esquema en serie siempre contendrá al óptimo pero su espacio de búsqueda es grande. En contraste el esquema en paralelo puede no contener al óptimo, pero las soluciones que construye en promedio son de menor duración.

Con esto en mente, a continuación, se describen los operadores genéticos del GA básico propuesto. Como este algoritmo comparte ciertos elementos con el algoritmo propuesto en el Capítulo 3, se hará especial énfasis en las diferencias y adaptaciones para ser implementado en el MRCPSP-ENERGY. De igual manera, para definiciones básicas sobre los operadores se remite al lector a dicho capítulo.

- **Codificación.** Se definieron dos vectores de n elementos cada uno, con dos genes adicionales como un cromosoma que codifica una solución completa del MRCPSP-ENERGY. El primer vector corresponde a la representación de lista de actividades y el segundo vector contiene los modos de ejecución de cada actividad en un orden ascendente al número de actividades. Los genes adicionales son el gen SGS y el gen de dirección, el primero representa el método usado para la decodificación (en serie o en paralelo) y el segundo la dirección del método (hacia adelante o hacia atrás). En la figura 5.5 se muestra una codificación de una solución factible del problema presentado en la figura 5.1.

Lista de actividades:	10	7	6	1	4	9	8	5	2	3	0
Actividades	0	1	2	3	4	5	6	7	8	9	10
Lista de modos:	0	2	2	3	2	1	2	1	3	1	0

Figura 5.5: Ejemplo de la codificación del problema mostrado en la figura 5.1

- **Función de aptitud.** Se han considerado dos diferentes alternativas para tener en cuenta los dos objetivos (minimizar el *makespan* y el consumo total de energía del proyecto): la primera alternativa consiste en la maximización de la eficiencia relativa del proyecto (Ecuación 4.8), este criterio fue propuesto y desarrollado en el Capítulo 4. La segunda alternativa consiste en la minimización de la combinación convexa de los dos objetivos normalizados. La Ecuación 5.1 muestra la función a minimizar anteriormente mencionado donde $C_{\text{máx}}$ es el *makespan* del proyecto, $CETP$ es el consumo energético del proyecto, α representa la importancia o peso asociado a minimizar el *makespan*, y $(1-\alpha)$ representa la importancia de minimizar el consumo energético. El valor de α debe ser definido por el decisor. Los parámetros $LB0_{\text{mín}}$, T, y $e_{\text{mín}}$ fueron definidos en el Capítulo 4, específicamente en la subsección 4.2.1, y $e_{\text{máx}}$ es la suma del consumo energético con mayor valor respecto al modo de ejecución. Bajo este criterio el objetivo es minimizar $F(C_{\text{máx}}, CETP)$.

$$F(C_{\text{máx}}, CETP) = \alpha * \frac{C_{\text{máx}} - LB0_{\text{mín}}}{T - LB0_{\text{mín}}} + (1 - \alpha) * \frac{CETP - e_{\text{mín}}}{e_{\text{máx}} - e_{\text{mín}}} \quad (5.1)$$

- **Población inicial.** Para generar la población inicial, se usó el método de muestreo *Regret Based Biased Random Sampling (RBBRS)*, usando la regla de prioridad de tiempo de inicio más tardío (Latest Start Time, LTS). De esta manera, se selecciona aleatoriamente una a una cada actividad para ser programada, dándole mayor probabilidad a las actividades que tengan menor valor de LTS. Los parámetros usados para el muestreo son iguales a los mostrados en el Capítulo 3.

- **Tamaño de población.** Debido a que las iteraciones son definidas como una solución completa, se ha encontrado que el tamaño de la población sigue estando relacionado con el número de iteraciones disponibles. Así, si el número de iteraciones es similar al tamaño de la población solo se pueden producir pocas generaciones, por el contrario, un tamaño de población demasiado pequeño, no aportaría la suficiente diversidad al algoritmo genético. De esta manera, se estima el tamaño de la población según las ecuaciones 3.3 y 3.4.

- **Selección y reemplazo.** Se utilizó un muestreo estocástico con reemplazo basado en el valor de la función de aptitud de cada solución. De esta manera, cada individuo de la población tiene una probabilidad diferente de cero de ser seleccionada acorde a su desempeño. Cuando un individuo es seleccionado, una réplica de esta se incluye en la siguiente selección.

- **Cruce y mutación.** Dentro de los algoritmos genéticos para resolver el RCPSP y el MRCPSP, el cruce estándar en uno o dos puntos suele ser el más usado. Teniendo en cuenta que este algoritmo genético es un punto de partida para realizar comparaciones futuras, se ha elegido el más simple: el cruce en un punto. Así, dos soluciones padres son seleccionadas p_1 y p_2, luego un valor aleatorio q es generado, los primeros genes de la posición 1 a q son tomadas de p_1, y las actividades restantes son tomadas en el orden de la lista de actividades de p_2. Para los genes de SGS y dirección, estos son heredados directamente cuando los padres comparten los genes, es decir que los padres tengan el mismo SGS o la misma dirección de programación, de lo contrario se generan aleatoriamente.

  Como se ha mencionado en la subsección 2.6.4 el método de mutación más usado es la inserción de Boctor, pues es un método simple y eficaz para alterar la lista de actividades. Este consiste en seleccionar una actividad aleatoria e insertarla en una posición aleatoria dentro de las posiciones que

no violen las restricciones de precedencia. La inserción de Boctor se usa con una probabilidad estándar de mutación de 0,05.

- **Mejora local.** De manera contundente, la técnica más utilizada como búsqueda local para mejorar una solución existente, es el método de mejora hacia adelante y hacia atrás (FBI), como se mencionó en la subsección 2.6.5. Este es usado en el GA básico propuesto.

5.4 Propuesta de una búsqueda alternativa basada en los modos de ejecución

En esta sección se propone un nuevo algoritmo genético denominado TP-AG (*Two Phases Genetic Algorithm*) para resolver el MRCPSP-ENERGY, este consiste en dos fases de optimización: la primera basada en la búsqueda mediante los modos de ejecución, y la segunda basada en la búsqueda mediante la lista de actividades (Morillo, Barber y Salido 2017). Con el objetivo de comparar el desempeño de la nueva búsqueda propuesta, se usará el esquema del algoritmo genético básico explicado en la sección 5.3 y solo se cambiarán los operadores implicados en la nueva propuesta. De esta manera, la definición de la codificación, la población inicial, el tamaño de la población, y el reemplazo son tomados del GA básico. A continuación, se detallan las dos fases de optimización y los operadores respectivos.

5.4.1 Fases de optimización

Como se ha mencionado anteriormente, el GA propuesto divide el proceso de búsqueda en dos fases de optimización. La primera fase usa unos operadores genéticos de cruce y mutación específicos que afectan solo la lista de modos de las soluciones, con el objetivo de expandir la búsqueda (exploración). Esta fase juega el rol principal en el GA propuesto. La segunda fase usa los operadores de cruce y mutación sobre la lista de actividades, este método es implementado en la etapa final del algoritmo para intensificar la búsqueda (explotación).

Operadores para la fase de optimización sobre la lista de modos

- **Cruce.** Para la construcción de nuevas soluciones a partir de dos previamente seleccionadas, se usó el cruce en dos puntos sobre la lista de modos. De esta manera, se generan dos valores enteros aleatorios q_1 y q_2 de tal manera que $0 < q_1 < q_2 < n$ y se seleccionan dos padres p_1 y p_2. Los primeros genes de la posición 0 a q_1 son tomados de p_1, los siguientes genes de $q_1 + 1$ a q_2 son

tomados de p_2 y los genes restantes, se toman nuevamente de p_1. Además, la lista de actividades de la solución generada, se hereda aleatoriamente. Los genes referentes a la SGS y a la dirección de programación son heredados directamente cuando los padres tienen los mismos genes, de lo contrario se generan aleatoriamente.

- **Mutación.** A pesar de que la mutación, generalmente, no tiene un rol fundamental en los algoritmos genéticos, permite introducir nuevo material genético a la población. En el caso del MRCPSP-ENERGY, un ligero cambio sobre la lista de modos puede causar grandes cambios sobre la solución, como se explicó en la sección 5.2. La mutación en esta etapa consiste en la selección de una actividad de manera aleatoria y cambiar su modo de ejecución.

Operadores para la fase de optimización sobre la lista de actividades

- **Cruce.** Se usó un cruce en dos puntos modificado sobre la lista de actividades. Así, primero se seleccionan dos padres (p_1, p_2) y dos valores enteros ($0 < q_1 < q_2 < n$) de forma aleatoria. Luego, los primeros genes de la posición 0 a q_1 son tomados del p_1, las siguientes $q_2 - (q_1 + 1)$ actividades son tomadas de la lista p_2, pero solo aquellas que no estén repetidas en la solución parcial (se verifica desde el inicio de la lista de p_2). Finalmente, las últimas actividades, se toman de la lista de p_2, recorriendo de nuevo el vector desde el principio, sin tomar las actividades repetidas en la solución parcial. En este caso, las actividades heredan los modos de ejecución directamente. Los genes de SGS y dirección se heredan si son compartidos por los padres, de lo contrario se generan aleatoriamente.

- **Mutación.** Se implementó la mutación agresiva propuesta en el Capítulo 3 sobre la lista de actividades.

Finalmente, el número de iteraciones necesario para cambiar de una fase de optimización de la lista de modos a la fase de optimización de la lista de actividades fue determinado mediante experimentación, con un valor de 2/3 del total de iteraciones disponibles. Este valor representa la importancia de la fase de optimización sobre los modos. De igual forma, se determinó que una probabilidad de mutación alta (80 %) en la primera fase, es la que en promedio obtiene mejores resultados. En contraste, se usó un valor estándar de probabilidad de mutación en la fase final, con un valor de 5 %.

En la figura 5.6 se muestra un ejemplo de la generación de descendencia usando los dos enfoques de optimización. En este ejemplo se usa un SGS en serie y una

dirección hacia adelante para la decodificación. Las soluciones A (genes rojos en la figura) y B (genes azules en la figura) son los padres, denominados p_1 y p_2. La solución C es creada mediante la optimización sobre la lista de actividades. Esta hereda las primeras 5 y las últimas 3 actividades de p_1, las actividades en las posiciones 5, 6, y 7 son heredadas de p_2; en este caso cuando se hereda una actividad, se hereda también su modo de ejecución. Por otra parte, la solución D es generada mediante la optimización sobre la lista de modos, esta hereda la lista de actividades de p_2, y la lista de modos se genera usando un cruce en dos puntos, donde la posición 5 y 6 de la lista de modos es heredada de p_2, y los modos restantes son heredados de p_1. Como se puede observar, a pesar de que la lista de actividades de la solución D es exactamente igual a la lista de p_2, las soluciones correspondientes son muy diferentes, debido solo a cambios sobre la lista de modos. De hecho, la solución D es la que tiene mejor valor de la función de aptitud. En comparación, la solución C a pesar de ser construida mediante un cruce en dos puntos sobre la lista de actividades, la lista de actividades resultante es igual a la lista de p_1, con la excepción de los modos de las actividades heredadas de p_2 (2, 5 y 8); mostrando nuevamente como las alteraciones a la lista de modos pueden alcanzar soluciones con estructuras diferentes.

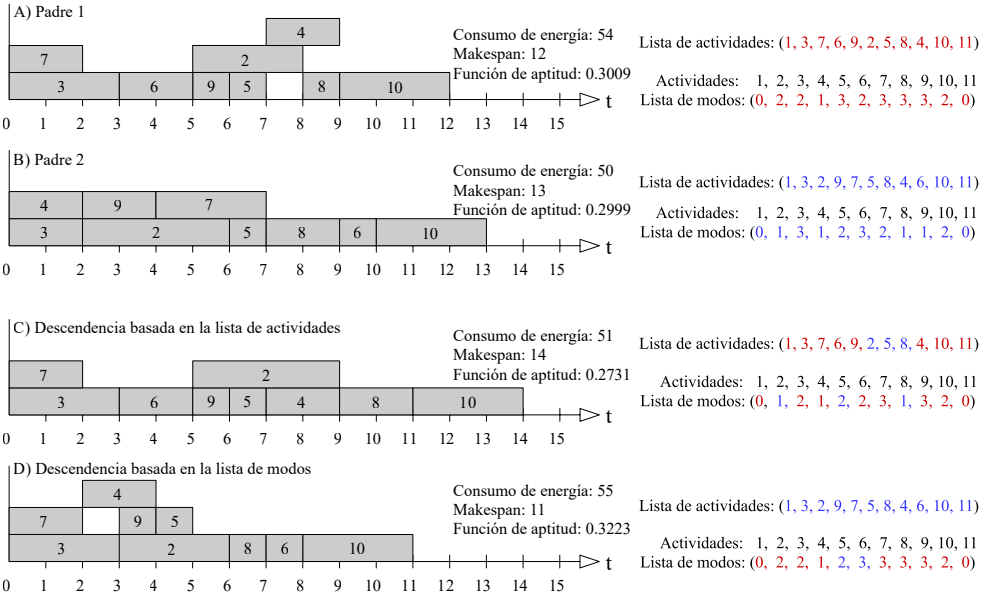

Figura 5.6: Un ejemplo de la generación de descendencia mediante la optimización sobre lista de actividades (descendencia C) y sobre la lista de modos (descendencia D).

Usualmente, se asume que, al modificar una lista de modos, solo se afecta el modo de ejecución de las actividades que serán programadas en un orden específico preestablecido (por la lista de actividades), pero realmente no es así. De hecho, el orden preestablecido también cambia como se aprecia en la figura 5.6.

5.4.2 Mejora local

Dos mejoras locales son implementadas en el algoritmo propuesto. La primera es la mejora denominada en la literatura como FBI *(Forward-Backward Improvement)*. La cual consiste en reprogramar una solución dada en dos pasos, el primero tomando las actividades de derecha a izquierda según su tiempo de finalización y las programa lo más tarde posible (más hacia la derecha), luego se toman las actividades de izquierda a derecha según su tiempo de iniciación y se programan lo más pronto posible (más hacía la izquierda). Durante estos pasos se suele obtener holguras entre las actividades que son aprovechadas para ser reprogramadas y así, obtener una solución más compacta. La FBI es implementada en la creación de la población inicial.

La segunda mejora local, consiste en comprobar si existe alguna actividad que pueda ser ejecutada con menor consumo de energía (mayor duración) sin afectar al *makespan*, sin superar el uso máximo de recursos renovables, ni violar alguna restricción de precedencia entre las actividades. Esta mejora se aplica a la mejor solución encontrada.

5.5 Evaluación de los métodos propuestos

En esta sección se presentan los resultados computacionales de los métodos propuestos para resolver el MRCPSP-ENERGY. Para ello se usó la librería propuesta en la sección 4.3: la PSPLIB-ENERGY. Para mayor claridad y análisis de los resultados, esta sección se divide en dos partes: la primera está enfocada en la evaluación del impacto de las soluciones redundantes presentes en la representación de lista de actividades y en la segunda parte se presentan los resultados sobre la librería de casos de prueba y se compara el rendimiento del GA propuesto.

5.5.1 Impacto de las soluciones redundantes en el MRCPSP-ENERGY

Para llevar a cabo esta evaluación, se realizó una búsqueda exhaustiva a través de todas las permutaciones de la lista de actividades y todas las combinaciones de la lista de modos. Debido al alto costo computacional de realizar esta búsqueda sobre problemas con muchas actividades, se usó un conjunto de 480 casos de prueba, cada uno con 10 actividades (sin incluir las ficticias). Estos nuevos problemas, denominados el conjunto $j10$, fueron basados en el conjunto $j30$ de la librería PSPLIB-ENERGY.

Las figuras 5.7 y 5.8 muestran los siguientes valores: el número total de soluciones (total), el número de soluciones factibles que incluyen las redundantes (feasible), y el número de soluciones únicas (unique), del total de permutaciones generados por medio de la lista de actividades (figura 5.7) y el conjunto completo de combinaciones generadas por medio de la lista de modos (figura 5.8).

En la figura 5.7, el número total de permutaciones ($10! = 3\,628\,800$) es igual para todos los problemas, puesto que este valor depende solo del número de actividades. Debido al enorme número de permutaciones, esta gráfica se muestra en una escala logarítmica. Como se puede observar, el número de soluciones redundantes es muy superior al número de soluciones únicas. Incluso, hay algunos problemas con solo una solución única.

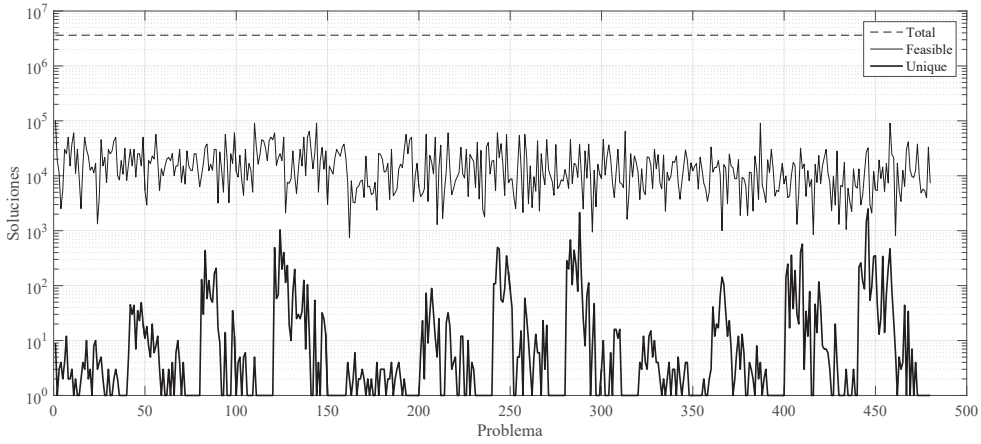

Figura 5.7: Permutaciones de la lista de actividades para el conjunto $j30$.

En la figura 5.8, el número de combinaciones para todos los problemas es 10! = 3 628 800, todas ellas factibles. Se puede observar que existen algunos casos donde el número de soluciones únicas es diferente del número de soluciones factibles, esto se debe a que algunas actividades tienen modos equivalentes por definición del problema. Por ejemplo, tomando la actividad 3 del problema de la figura 5.1, se observa que tiene 3 modos; el modo 2 corresponde a una duración de 1 unidad de tiempo, con un consumo de energía de 6 unidades. Esta actividad no puede ser ejecutada en menor tiempo por lo tanto el modo 3 es equivalente al modo 2.

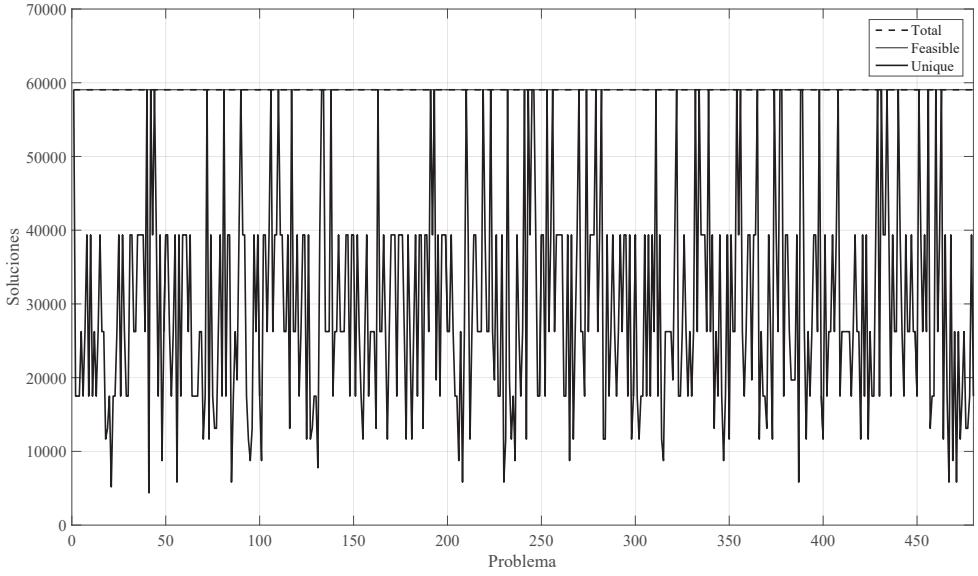

Figura 5.8: Combinaciones de la lista de modos para el conjunto $j30$.

Por lo tanto, aunque el espacio de soluciones que genera la búsqueda basada en la representación de lista de actividades es mucho más grande que el espacio de soluciones generado por la búsqueda basada en la lista de modos, la mayoría de ellas son soluciones redundantes ocasionando una búsqueda poco eficiente. En contraste, el espacio de soluciones basado en la lista de modos está conformado solo por soluciones factibles, la gran mayoría únicas. Solo podría haber soluciones redundantes si existen modos equivalentes o si existen actividades que no se vean afectadas por el cambio en el uso de recursos ocasionada por el distinto modo de ejecución de una actividad.

5.5.2 Evaluación de las metaheurísticas propuestas

En esta sección se realiza la evaluación empírica considerando 5 variantes del algoritmo propuesto generados a partir de los operadores genéticos descritos en este capítulo (Morillo, Barber y Salido en prensa). De esta manera se puede realizar una comparación objetiva para identificar los elementos que contribuyen a la búsqueda. Las variantes consideradas son descritas a continuación:

- El algoritmo genético básico (Basic-GA): descrito en la sección 5.3, este algoritmo usa una codificación basada en la lista de actividades, usa un operador de cruce estándar en un solo punto y la mutación está basada en la inserción de Boctor. Este algoritmo toma los principales elementos del algoritmo genético estándar usado para resolver el RCPSP, centrando sus operadores genéticos en la modificación de las listas de actividades y dejando a la mutación como principal herramienta para cambiar los modos.

- El algoritmo genético basado en la optimización sobre la lista de actividades (AL-GA). Este algoritmo se centra en realizar la búsqueda explorando las permutaciones de las listas de actividades usando los operadores genéticos respectivos descritos en la subsección 5.4.1. Los modos de las actividades en la población permanecen constantes luego de la creación de la población inicial. Los descendientes heredan las actividades con el modo de ejecución del correspondiente padre.

- El algoritmo genético basado en la optimización sobre la lista de modos (ML-GA). La búsqueda realizada por este algoritmo se centra en la modificación de la lista de modos de la población (sus combinaciones) usando los operadores genéticos respectivos descritos en la sección subsección 5.4.1. Las listas de actividades permanecen constantes luego de su creación en la población inicial.

- El algoritmo genético basado en la mezcla simultánea de las fases de optimización (MIX-GA). Este algoritmo mezcla los dos enfoques de optimización: la optimización sobre la lista de actividades y la optimización sobre la lista de modos. De este modo, los operadores de las dos fases se implementan de forma simultánea. En este caso ninguna lista de actividades ni de modos permanece constante.

- El algoritmo genético basado en las dos fases de optimización de forma separada (TP-GA). Este algoritmo incorpora las dos fases de forma independiente, pero se le da mayor importancia a la fase de optimización sobre la lista de modos usando 2/3 del número total de iteraciones disponibles sobre

esta fase, y al final del algoritmo se usa 1/3 de las iteraciones disponibles en la fase de optimización de la lista de actividades.

Para realizar esta evaluación se usaron los casos de prueba de los conjuntos $j30$, $j60$, y $j120$ de la librería PBPLIB-ENERGY presentada en el sección 4.3. Por otra parte, el criterio de parada más usado en la literatura para los algoritmos que encaran el RCPSP y el MRCPSP es el número máximo de iteraciones, específicamente las comparaciones estándares usan como máximo 1 000, 5 000, y 50 000 iteraciones. Así, en esta evaluación este criterio también es usado, como se mencionó en la subsección 3.4.2, una iteración es considerada una solución completa, es decir aquella en la que se puede calcular su *makespan*.

La tabla 5.1 resume los resultados obtenidos. La primera columna muestra el conjunto de problemas resuelto, la segunda indica el algoritmo usado, la tercera, cuarta y quinta columna muestran el valor de la eficiencia relativa promedio de todo el conjunto para 1 000, 5 000, y 50 000 iteraciones, respectivamente. Esta tabla se encuentra organizada por los valores encontrados con 50 000 iteraciones.

Tabla 5.1: $\bar{\eta}$ obtenido usando los algoritmos propuestos para resolver el MRCPSP-ENERGY.

j#	Algorithm	Iterations/$\bar{\eta}$		
		1 000	5 000	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

Como se puede observar de la tabla 5.1, el ML-GA obtiene en promedio un valor de la eficiencia relativa mayor que la obtenida por AL-GA. Es interesante remarcar que en el ML-GA, las listas de actividades de todo el conjunto de problemas se mantienen constantes una vez se hayan generado con la población inicial, y solo han cambiado los modos de ejecución. Por otra parte, en el AL-GA, los modos de listas de todos los problemas permanecen constantes, y solo se han alterado las listas de actividades. Esto indica que la búsqueda a través de la lista de modos puede alcanzar soluciones de alta calidad, en comparación con la búsqueda realizada mediante la lista de actividades. Por su parte, el $MIX-GA$ no alcanza los mejores resultados, esto puede deberse a que al mezclar los dos enfoques de optimización se esté logrando una gran exploración pero a cambio de una baja explotación en el espacio de búsqueda.

Debido a la existencia de soluciones redundantes, una solución óptima puede ser representada mediante una lista de modos y varias listas de actividades. Por lo tanto, cuando se fija una lista y se busca sobre la otra, se puede excluir a las soluciones óptimas. Sin embargo, el TP-GA propuesto busca sobre la lista de modos y, en la fase final, sobre la lista de actividades. Esta combinación puede alcanzar mejores resultados. De hecho, el TP-GA supera a los otros algoritmos como se puede apreciar en la tabla 5.1.

Por otra parte, en el presente estudio también se compararon los resultados obtenidos por algoritmos propuestos considerando la función de aptitud como la combinación convexa de los objetivos: la minimización del *makespan* y la minimización del consumo total de energía (Ecuación 5.1). Sin perder generalidad, se han establecido tres valores de α: 0,25, 0,5 y 0,75. La tabla 5.2 muestra un resumen de los resultados obtenidos. La primera fila muestra el valor de α y la segunda fila muestra el número de iteraciones consideradas. La primera, quinta y novena columna muestran la versión del algoritmo usado. Las columnas restantes muestran el valor de la función objetivo normalizada ($F(C_{\text{máx}}, CETP)$).

Basándose en los resultados de la tabla 5.2, se puede observar que el TP-GA sigue siendo el algoritmo que obtiene mejores resultados en promedio, de hecho supera a los demás métodos en 7 del total de 9 conjuntos experimentales. Adicionalmente, en los conjuntos $j60$ con un $\alpha = 0{,}25$ y $j60$ con un $\alpha = 0{,}5$ las diferencias respecto al primer puesto son de 0,09 % y 0,27 %, respectivamente. Además, se puede apreciar que la búsqueda mediante la optimización de la lista de modos (ML-GA) puede alcanzar soluciones similares a aquellas alcanzadas por la búsqueda basada en la optimización de la lista de actividades (AL-GA), incluso la sobrepasa en algunos casos. Por otra parte, el enfoque que mezcla de forma simultánea las fases de optimización (MIX-GA) nuevamente parece no ser el método adecuado pues es el que obtiene peores resultados en promedio, debido a su gran diversidad,

Tabla 5.2: Valores normalizados obtenidos usando las variantes del algoritmo genético propuesto para resolver la librería MRCPSP-ENERGY.

$\alpha = 0{,}25$				$\alpha = 0{,}5$				$\alpha = 0{,}75$			
Alg	Iterations/F			Alg	Iterations/F			Alg	Iterations/F		
	1 000	5 000	50 000		1 000	5 000	50 000		1 000	5 000	50 000
Set $j30$											
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 $j60$											
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 $j120$											
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

los resultados sugieren que esta búsqueda puede dificultar la convergencia a una buena solución.

Finalmente, abarcando un enfoque diferente y puesto que aún no se han reportado soluciones óptimas para los casos de prueba de la PSPLIB-ENERGY, en esta investigación se propone un método exacto con la finalidad de obtener dichos resultados. La programación fue realizada usando el software IBM ILOG CPLEX CP optimizer 12.6.2 para resolver las instancias de la librería. Esta herramienta permite el uso de la programación de restricciones *(Constraint Programming)* para solucionar este tipo de problemas, se ha seleccionado este método debido a su gran éxito en problemas combinatoriales, específicamente en problemas de programación de proyectos.

Puesto que el MRCPSP-ENERGY también es un problema NP-Hard, es imposible determinar las soluciones óptimas en un tiempo razonable para problemas de tamaño real. Por lo tanto, se fijó un tiempo máximo de ejecución de 30 minutos al método exacto propuesto. Consideramos que es un límite aceptable para este tipo de problemas. Finalmente, considerando que las soluciones óptimas para el problema uni-modal del RCPSP con 60 actividades de la PSPLIB, aún no han sido encontradas, e intentar resolver de manera óptima el MRCPSP-ENERGY es en efecto, más complejo, solo se tomó el conjunto $j30$ de la PSPLIB-ENERGY de 480 problemas.

El enfoque exacto fue capaz de encontrar el 70 % de las soluciones óptimas para el conjunto evaluado. La tabla 5.3 muestra un resumen de los resultados, comparándolos con el TP-GA propuesto. La primera columna muestra el conjunto de instancias evaluado. La segunda muestra el método utilizado para resolver los problemas. La tercera columna muestra el valor promedio de la función de aptitud (la eficiencia relativa de un proyecto). La cuarta columna muestra el número de soluciones óptimas encontradas y el total de problemas. La quinta columna muestra el tiempo promedio usado por problema. Finalmente, la última columna muestra el tiempo límite.

El enfoque exacto (CPLEX) alcanzó un valor promedio de 65,97 % de eficiencia relativa promedio de un proyecto con un tiempo computacional de 601,15 segundos. El TP-GA propuesto logra un valor promedio de 65,75 % de la eficiencia relativa promedio de un proyecto con un tiempo computación de 2,51 segundos usando 50 000 iteraciones. Así, la diferencia entre los promedios de la función objetivo de cada método es de 0,3335 %, teniendo en cuenta que el TP-GA propuesto requiere un 99,58 % menos de tiempo que CPLEX. Por lo tanto, se puede concluir que el TP-GA propuesto logra un gran balance entre tiempo y precisión.

Tabla 5.3: Resultados obtenidos por IBM ILOG CPLEX CP optimizer para resolver el conjunto $j30$ de la librería MRCPSP-ENERGY.

Set $j30$

Método	$\bar{\eta}$	# óptimos	Tiempo promedio (s)	Límite de tiempo (s)
CPLEX	0,6597	340(480)	601,15	1 800
TP-GA	**0.6575**	**218(480)**	**2.51**	**-**

Por último, con el objetivo de evaluar el impacto de las soluciones redundantes en las instancias de la PSPLIB-ENERGY, se calculó el número promedio de descendientes redundantes generados por los operadores genéticos sobre la lista de actividades, cuando los padres eran diferentes el uno del otro. La generación de descendientes de padres iguales no fue considerada. Las estimaciones se realizaron sobre los conjuntos $j30$, $j60$ y $j120$, para 1 000, 5 000, y 50 000 iteraciones. Los resultados pueden ser observados en la figura 5.9, como se esperaba, el número de soluciones redundantes parece estar relacionado con el número de iteraciones y, en menor grado, con el número de actividades. De los resultados, el número promedio de soluciones redundantes que no contribuyen en la búsqueda con nueva información en el GA es como máximo el 15 % del número total de iteraciones.

5.6 Conclusiones

En este capítulo se proponen dos algoritmos evolutivos para resolver el MRCPSP-ENERGY. El primero es un algoritmo genético (GA básico) que incorpora los principales elementos de los mejores métodos reportadas en la literatura para el RCPSP y el MRCPSP, adaptadas para la variante energética, cuyo propósito es comprobar si las estrategias basadas en los problemas base, que no consideran el uso de energía siguen siendo eficientes, además de establecer un primer punto de comparación para futuros métodos. El segundo también es un algoritmo genético (TP-GA), cuya principal aportación es mostrar que existe un campo infravalorado de búsqueda basado en los modos de ejecución. Este algoritmo propuesto se enfoca en realizar la búsqueda basándose en la lista de modos en lugar de hacerlo en la lista de actividades, como es común hacerse. Para ello, el algoritmo genético incorpora como estrategia de resolución dos fases de optimización: la primera usa como medio de búsqueda la representación de lista de modos de ejecución, y la segunda usa en su lugar la lista de actividades. Siendo la búsqueda a través de los modos la más relevante. A partir del algoritmo genético propuesto, se proponen 4

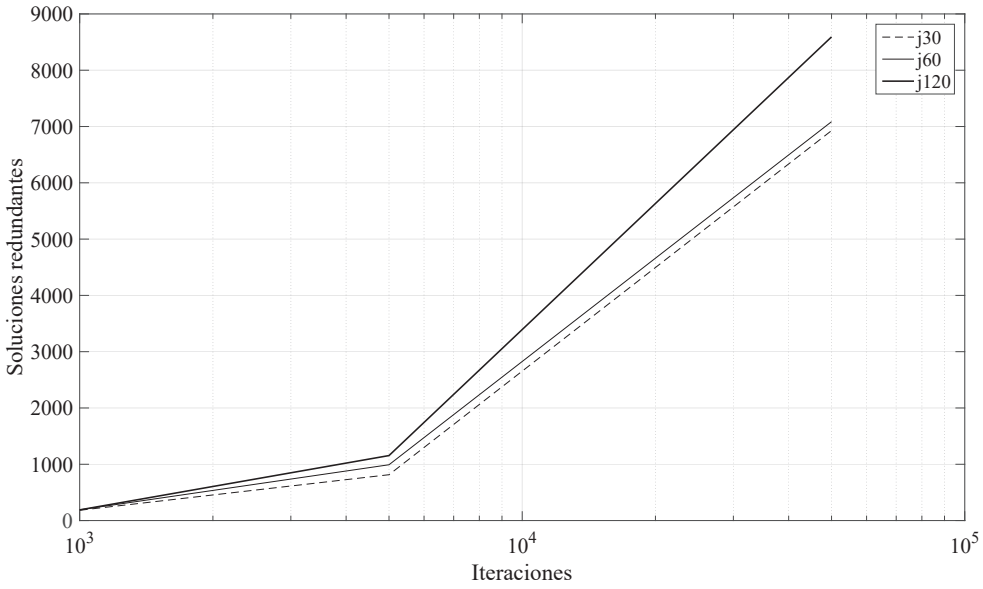

Figura 5.9: Número promedio de soluciones redundantes usando el GA propuesto en la PSPLIB-ENERGY.

variantes del mismo para realizar la comparación de resultados y poder establecer los elementos que realmente aportan a la búsqueda.

Adicionalmente, se estudió el fenómeno de soluciones redundantes en el MRCPSP-ENERGY. Este consiste en la representación múltiple usando diferentes representaciones de listas de actividades que producen una misma solución real; aunque se puede extrapolar a otras representaciones. Por lo tanto, el esfuerzo computacional puede ser desperdiciado en aquellas soluciones ya visitadas, que no aportan material genético nuevo para mejorar la búsqueda. Por otra parte, se mostró que la redundancia afecta a la lista de modos en dos escenarios: el primero es que existan modos equivalentes en una actividad (que se puede evitar la lectura de los datos de entrada), y el segundo es que las actividades restantes de la solución parcial no se vean afectadas por el cambio del uso de recursos, ya sea por la falta de ellos o por su disponibilidad. Por tanto, si existen modos diferentes, un solo cambio sobre la lista de modos generalmente genera una solución diferente. A pesar que el espacio de soluciones generado por la lista de actividades (permutaciones) es mucho mayor que el espacio generado por la lista de modos (combinaciones), la mayoría de estas soluciones son redundantes.

Finalmente se realizó una evaluación experimental de los métodos propuestos usando la PSPLIB-ENERGY. Los resultados muestran que el TP-GA propuesto alcanza los mejores resultados en todos los conjuntos de prueba. Los resultados también sugieren que una búsqueda basada solo en modificaciones sobre la lista de modos (algoritmo denominado, ML-GA) puede alcanzar mejores resultados que una búsqueda basada solo en modificaciones sobre la lista de actividades (algoritmo denominado, AL-GA). Adicionalmente, se realizó una comparación entre la metaheurística propuesta y un método exacto, para esto se utilizó el software IBM ILOG CPLEX CP optimizer. A pesar de que el enfoque exacto produjo unos resultados levemente mejores en promedio (una diferencia de 0,3335 %), el TP-GA propuesto requirió 99,58 % menos de tiempo. Teniendo en cuenta la experimentación realizada se concluye que el método propuesto es capaz de obtener soluciones altamente eficientes. Adicionalmente, se realizó una evaluación de los algoritmos propuestos reemplazando la función de aptitud por una combinación convexa normalizada de los dos objetivos, como es usual en la optimización multi-objetivo; el propósito de esta última evaluación era determinar si los resultados de los métodos propuestos se debían al uso del nuevo criterio de optimización (la eficiencia relativa del proyecto) o a las diferentes estrategias de búsqueda. Los resultados se muestran consistentes a las conclusiones antes obtenidas con el nuevo criterio de optimización, demostrando de manera empírica que la búsqueda mediante la optimización de las listas de modos, a pesar de ser poco explorada, es una alternativa de búsqueda con gran potencial, alcanzando soluciones tan buenas y en promedio mejores que la búsqueda usual sobre la lista de actividades.

Teniendo en cuenta que la mayoría de métodos reportados en la literatura prestan mayor atención en realizar la búsqueda sobre las permutaciones de la lista de actividades, pensamos que existe un valioso campo de investigación donde se enfoque la búsqueda sobre la lista de modos.

Capítulo 6

Conclusiones

En este capítulo se resumen las aportaciones y conclusiones realizadas en esta tesis, basándose en los objetivos planteados inicialmente. Además, se plantean algunas líneas de investigación futuras a seguir y se presentan las publicaciones producto de la actual tesis.

6.1 Aportaciones

En esta sección se resumen los resultados y las aportaciones más relevantes de la tesis. Siguiendo la orientación de la investigación y los objetivos planteados, las principales contribuciones de esta tesis se pueden agrupar en cuatro puntos:

- Inicialmente, se ha realizado una revisión bibliográfica de los diferentes métodos de solución para el problema de programación de tareas con recursos restringidos en su versión uni-modal (RCPSP) y en su versión multi-modal (MRCPSP). En la literatura se distinguen dos métodos para resolver estos problemas: métodos exactos y métodos de aproximación. Los primeros se basan, principalmente, en programación lineal entera mixta, algoritmos de ramificación y acotamiento, y enumeración exhaustiva o implícita. Su principal característica es que garantizan la obtención de una solución óptima. Dado el alto nivel de complejidad del RCPSP y el MRCPSP (ambos con un nivel de complejidad NP-Hard) los métodos exactos no son capaces de

resolver problemas de tamaño mediano-grande en un tiempo razonable. De hecho, existen instancias de tan solo 30 actividades que aún no han sido resueltas óptimamente. El segundo grupo abarca un gran conjunto de métodos: *esquemas de representación de soluciones*, encargados de codificar una solución del problema para generar un vecindario nuevo de soluciones a partir de dicha solución; *esquemas generadores de secuencia*, que permiten decodificar las representaciones; *reglas de prioridad*, heurísticas para seleccionar las actividades a programar dentro de un conjunto de alternativas; *algoritmos metaheurísticos*, que son los métodos de búsqueda aproximada más avanzados, en esencia son métodos generales para elaborar búsquedas iterativas que incorporan mecánicas de escape a óptimos locales; por último, las *mejoras locales*, que generalmente consisten en métodos iterativos para mejorar una solución existente.

- Respecto al RCPSP uni-modal y multi-modal se llegó a la conclusión que las metaheurísticas evolutivas (especialmente los algoritmos genéticos) son, junto con los métodos híbridos, los métodos de búsqueda más eficaces para resolver estos problemas. Además, se demostró que uno de los elementos más relevantes de las metaheurísticas en general, es la representación de las soluciones, siendo la representación de lista de actividades y la de valor clave las más usadas. Sin embargo, se mostró que, el denominado fenómeno de soluciones redundantes en las representaciones afecta de manera significativa a la búsqueda de los métodos metaheurísticos, puesto que obstaculiza la introducción de diversidad (nueva información) al algoritmo. Especialmente, se mostró experimentalmente que dentro de un algoritmo genético los operadores encargados de generar perturbaciones sobre las soluciones, y así introducir diversidad, como la mutación, no cumplían su cometido a cabalidad debido al fenómeno de redundancia.

  El principal operador de mutación tanto para RCPSP como para el MRCPSP se basa en la inserción de Boctor. Este método consiste en la selección aleatoria de una actividad en una lista y posterior inserción en una posición de la lista mayor a la última actividad predecesora y menor a la primera actividad sucesora. Existen fundamentalmente dos formas de implementar este operador: por actividad o por solución. En la primera, la mutación es ejecutada con una probabilidad para cada actividad, es decir cada gen, de la lista de actividades. En este caso, la mutación incrementa el número de inserciones en una solución a medida que el problema tiene mayor número de actividades. De manera similar, la probabilidad de mutación sobre una solución también aumenta con el número de actividades, llegando a ser del 99,8 % para un problema con 120 actividades. Basándose en los resultados,

estas dos características no son las indicadas para alcanzar los mejores resultados, pues la cantidad de diversidad introducida es excesiva y tampoco permiten disminuir el número de soluciones redundantes. En la segunda forma de implementar la inserción, la mutación se aplica con una probabilidad definida sobre una solución, es decir sobre un conjunto de actividades que conforman a un individuo completo de una población. Como es de esperarse, el número de mutaciones disminuye considerablemente con respecto a la primera forma y la proporción de soluciones afectadas se mantiene constante, rigiéndose por la probabilidad de mutación. Sin embargo, los resultados muestran que en este caso la mutación no introduce suficiente diversidad a la búsqueda para lograr los mejores resultados.

Considerando lo anterior, se propuso un algoritmo genético cuya principal aportación incluye un nuevo operador de mutación agresiva que disminuye la generación de soluciones redundantes e introduce una gran perturbación sobre la población. Esta consiste en una mutación efectuada por solución que usa múltiples inserciones con una alta probabilidad de ocurrencia. Tanto el número de inserciones y como la probabilidad son independientes del número de actividades del problema. La propuesta del nuevo operador se basa en el hecho de que el fenómeno de redundancia también es independiente del número de actividades, y que una mutación con una función creciente de inserciones relacionada con el número de actividades, puede generar demasiada diversidad y entorpecer la búsqueda de los operadores que fomentan la convergencia del algoritmo cuando un problema tenga un gran número de actividades.

Basándose en la experimentación realizada, el número de inserciones se estableció en 3 con una probabilidad del 90 %. Los resultados muestran que la mutación propuesta genera un 20 % menos de soluciones redundantes y alcanza mejores soluciones (de menor *makespan*) en comparación con la mutación clásica usada en la literatura. De manera similar, al comparar el algoritmo propuesto con los mejores métodos reportados en la literatura, se concluye que el algoritmo genético con mutación agresiva alcanza soluciones altamente competitivas.

Adicionalmente, como aporte general, cabe destacar que el operador propuesto se puede incorporar o adaptar fácilmente a otras metaheurísticas.

- Como contribución general al área de programación de proyectos, se propuso una extensión del RCPSP denominada MRCPSP-ENERGY, basada en la eficiencia energética, que tiene en cuenta tanto el tiempo total de finalización del proyecto (*makespan*), como el consumo de energía de las actividades.

Además de las características propias del RCPSP, como son el consumo de recursos renovables y la estructura de precedencias entre actividades, el problema considera que los diferentes consumos de energía en una actividad dan a lugar a diversos modos de ejecución. El objetivo del problema propuesto es maximizar la denominada eficiencia relativa del proyecto. Este criterio unifica el *makespan* y el consumo de energía en un solo objetivo denominado eficiencia relativa. La propuesta se basó en la definición física de la eficiencia, es decir, la relación entre la energía suministrada a un sistema y la energía transformada por el mismo. Dicho concepto se extrapoló a la eficiencia de una actividad y posteriormente a la eficiencia de un proyecto. Adicionalmente, se ha provisto de una librería de casos de prueba para realizar comparaciones entre diferentes métodos de solución para el MRCPSP-ENERGY que se desarrollen en un futuro, esta librería está basada en la actual librería PSPLIB, considerada un estándar para la evaluación de nuevos métodos de resolución del RCPSP y el MRCPSP.

- En cuanto a los métodos de solución para el MRCPSP-ENERGY, inicialmente se mostró cómo el fenómeno de soluciones redundantes afecta a los métodos de solución metaheurísticos, siendo de mayor impacto en los problemas multi-modales pues el espacio de búsqueda es mayor. Posteriormente, se diseñaron e implementaron dos algoritmos genéticos para ser comparados entre sí: el primero, denominado GA básico, tiene como objetivo evaluar si los métodos más exitosos para el RCPSP uni-modal y multi-modal, que no consideran el consumo energético, son igualmente eficaces para resolver el MRCPSP-ENERGY. El segundo, tiene como objetivo mostrar que realizar la búsqueda a través de la lista de modos es una forma diferente y eficiente de explorar el espacio de soluciones, que puede lograr mejores resultados en comparación con la búsqueda tradicional enfocada en la lista de actividades. De este modo, el segundo algoritmo denominado TP-GA consiste en un algoritmo genético que divide la búsqueda en dos etapas: la primera y más importante, basa su búsqueda en la optimización de la lista de modos de una población inicial en la que se fijan sus listas de actividades; la segunda etapa ocurre al final del algoritmo en donde una fracción del esfuerzo computacional es dedicado a la optimización de la lista de actividades. El algoritmo propuesto encuentra su justificación en el hecho de que las modificaciones sobre una lista de modos garantizan siempre una solución diferente, siempre y cuando se cumplan dos condiciones: que existan modos diferentes entre las actividades, es decir con consumos y duraciones distintas; y que existan actividades cuya ejecución se vea afectada por el cambio en la disponibilidad de recursos que se lleva a cabo al cambiar un modo de ejecución de una actividad. Mientras que

las modificaciones sobre la lista de actividades generan múltiples soluciones redundantes que no pueden ser verificadas sin antes decodificar la solución.

Para realizar la evaluación, se utilizó la librería propuesta que permite comparar el desempeño de los algoritmos propuestos. Dado que no existen estimaciones de las soluciones óptimas para la librería, se propuso también un enfoque exacto basado en programación lineal entera mixta. Respecto al GA básico, al comparar los resultados con el enfoque exacto podemos concluir que, si bien el GA básico logra encontrar soluciones eficientes a nivel energético, existe aún un intervalo considerable de mejora. Adicionalmente, al comparar los resultados que puede alcanzar el GA básico aplicado al RCPSP estándar (sin tener en cuenta el consumo de energía) con los resultados considerando el consumo de energía, se puede concluir que asignar diferentes consumos de energía y tiempos de procesamiento a las actividades en problemas de secuenciación de tareas permite lograr mejores soluciones en términos de eficiencia energética.

Para evaluar el desempeño del algoritmo TP-GA, primero se compararon las dos fases de optimización por separado. Con este fin, se implementaron dos algoritmos que caracterizan cada fase: el primer algoritmo, denominado AL-GA, representa la búsqueda tradicional sobre la lista de actividades, una vez se crea una población inicial, los operadores genéticos modifican únicamente la lista de actividades de las soluciones para realizar la búsqueda, manteniendo constante las listas de modos de todos los individuos. El segundo algoritmo, denominado ML-GA, genera una población inicial y a partir de ese punto, las listas de actividades se mantienen constantes, mientras que los operadores realizan la búsqueda sobre las listas de modos. Basado en los resultados presentados, se observa que el ML-GA, a pesar de no cambiar las listas de actividades en su búsqueda, puede alcanzar soluciones más eficientes que el AL-GA. Este hecho confirma la hipótesis de que los cambios sobre una lista de modos, a pesar de mantener una lista de actividades fija, generan diferentes soluciones, es decir diferentes secuencias de actividades con estructuras de secuenciación distintas.

Adicionalmente, los resultados muestran que el algoritmo propuesto TP-GA (que incluye las dos fases de búsqueda) es el que obtiene los mejores resultados en comparación con los demás algoritmos propuestos. Al evaluar su desempeño con respecto a los resultados alcanzados por el enfoque exacto, el algoritmo propuesto es superado por solo 0,3335 % pero usando un 99,58 % menos de tiempo. Se concluye, que la mezcla de los dos enfoques de búsqueda (sobre la lista de modos y sobre la lista de actividades) es una estrategia

que encuentra soluciones altamente eficientes, y que estas fases no buscan excluirse entre sí, sino complementarse.

6.2 Líneas futuras de investigación

Esta tesis tiene como propósito, además de los objetivos planteados, fomentar el desarrollo de futuras investigaciones para encontrar soluciones sostenibles y eficientes en problemas de programación de proyectos, especialmente en los que los recursos se encuentran restringidos. Por este motivo, es de vital importancia identificar las diferentes líneas de trabajo para dar continuidad a esta investigación. En esta sección se resumen algunas líneas futuras de investigación.

- Extender el problema considerando el uso de la energía como un recurso no renovable, esto podría ocurrir, por ejemplo, cuando se tiene un presupuesto definido para el proyecto o bien si los recursos energéticos son escasos. Siempre y cuando la disponibilidad permita la construcción de soluciones factibles, las mejores soluciones podrían centrarse en usar altos niveles de energía solo sobre las actividades más relevantes para el proyecto, estas serían aquellas que ofrezcan el mejor coste/beneficio o bien sean actividades con gran impacto sobre la estructura, es decir actividades cuya rápida finalización permitiría la construcción de un gran número de nuevas soluciones.

- Teniendo en cuenta que el nuevo criterio de optimización para el MRCPSP-ENERGY (la eficiencia relativa) unifica los objetivos de minimización de *makespan* y minimización del consumo de energía, podría estudiarse desde un enfoque multi-objetivo tradicional, para determinar en qué parte de la frontera de soluciones de Pareto se encuentra la eficiencia relativa, y qué implicaciones tiene dicha posición. Por ejemplo, se podría determinar si la eficiencia relativa es una cota para problemas donde los objetivos se relacionen de manera similar al *makespan* y al consumo de energía.

- Respecto a los métodos de solución, para el RCPSP uni-modal, se podría estudiar e identificar los principales causantes de la redundancia en todos los operadores, y de manera más general, en las reglas de movimiento, para diseñar nuevos procedimientos que eviten la generación de soluciones redundantes. En el caso particular del RCPSP multi-modal, también podrían desarrollarse nuevos métodos de búsqueda, basados en la actual representación que se beneficien de las características de la exploración mediante la modificación de la lista de modos.

De manera similar, las futuras investigaciones podrían enfocarse en el análisis de la topología del espacio de soluciones para el RCPSP uni-modal y multi-modal, así como del MRCPSP-ENERGY, para diseñar nuevas representaciones de soluciones que no se vean afectadas por el fenómeno de redundancia y permitan a su vez, el diseño de nuevas reglas de movimiento que generen vecindarios de soluciones diversos para poder realizar la búsqueda.

- Uno de los principales supuestos de los problemas de programación de proyectos, incluido el MRCPSP-ENERGY, es que el problema se desarrolla en un entorno determinista. Donde todas las actividades tienen una duración constante durante la fase de planificación y la fase de ejecución. Sin embargo, este supuesto no siempre se puede asumir, por este motivo las investigaciones futuras podrían centrarse en el estudio de soluciones robustas para aplicaciones más realistas del MRCPSP-ENERGY, es decir, soluciones poco sensibles a eventos inesperados relacionados con el uso de energía y por tanto en la duración de las actividades.

- Dado que la inclusión de aspectos económicos en los problemas de programación de proyectos es de gran relevancia en la industria, se podría considerar un perfil de coste de energía variable en el tiempo. Esta situación se presenta frecuentemente en problemas aplicados, puesto que las empresas proveedoras de energía varían su coste para regular la demanda. De esta manera, el consumo de energía en horas de la noche, por ejemplo, suele ser más económico. Esta investigación futura también podría tener en cuenta costes de encendido o preparación de máquinas, así como costes de mantener preparada la máquina, sin estar procesando ninguna actividad.

6.3 Publicaciones

En esta sección se presentan las publicaciones fruto de la investigación realizada durante esta tesis doctoral.

6.3.1 Publicaciones en revistas

- **A new model and metaheuristic approach for the energy-based resource-constrained scheduling problem.** D. Morillo, F. Barber and M. Salido. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture. 2017.* doi:10.1177/0954405417711734. (JCR 1.366).

- **Mode-based vs Activity-based search for a non-redundant resolution of the Multi-mode Resource-Constrained Project Scheduling Problem.** D. Morillo, F. Barber and M. Salido. *Mathematical Problems in Engineering. En prensa.* (JCR 0.802).

- **PSPLIB-ENERGY: Una extensión de la librería PSPLIB para la evaluación de la optimización energética en el RCPSP**. D. Morillo, F. Barber and M. Salido. *Iberoamerican Journal of Artificial Intelligence*, 17(54), págs. 35-48. 2014.

- **MRCPSP-ENERGY, an energy-based scheduling problem.** D. Morillo, F. Barber and M. Salido. *Feature Article, ALP newsletter.* 2016.

6.3.2 Congresos

- **Mode List vs Activity List for the Multi-mode Resource Constrained Project Scheduling Problem.** D. Morillo, F. Barber and M. Salido. *The 28th International Conference on Automated Planning and Scheduling (ICAPS 2018).* Delft, The Netherlands, Junio-2018. Artículo invitado a participar en modalidad *fast track*.

- **Mode List vs Activity List for the Multi-mode Resource Constrained Project Scheduling Problem.** D. Morillo, F. Barber and M. Salido. *ICAPS 2017 workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS'17),* págs. 38-47. Pittsburgh, USA, 2017. Este congreso está clasificado como CORE B en ERA Conference Ranking.

- **Una extensión de la librería PSPLIB para la evaluación de la optimización energética en el problema de secuenciación de proyectos con recursos restringidos.** D. Morillo, F. Barber and M. Salido. *XVII Latin-Iberian-American Conference on Operations Research.* Monterrey, Mexico, 2014.

- **MRCPSP-ENERGY, un enfoque meta-heurístico para problemas de programación de actividades basado en el uso de energía.** D. Morillo, F. Barber and M. Salido. *XVIII CLAIO Latin-Iberoamerican Conference on Operations Research.* Santiago, Chile, 2016.

Bibliografía

Aggarwal, Aman y col. (2008). «Optimization of multiple quality characteristics for CNC turning under cryogenic cutting environment using desirability function». *Journal of Materials Processing Technology* 205.1, págs. 42-50 (vid. pág. 59).

Agha, Mujtaba H. y col. (2010). «Integrated production and utility system approach for optimizing industrial unit operations». *Energy* 35.2, págs. 611-627. ISSN: 0360-5442 (vid. pág. 63).

Alcaraz, J., C. Maroto y R. Ruiz (2003). «Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with Genetic Algorithms». *The Journal of the Operational Research Society* 54.6, págs. 614-626 (vid. pág. 46).

Artigues, Christian, Pierre Lopez y Alain Haït (2013). «The energy scheduling problem: Industrial case-study and constraint propagation techniques». *International Journal of Production Economics* 143.1, págs. 13-23. ISSN: 0925-5273 (vid. pág. 62).

Artigues, Christian, Philippe Michelon y Stéphane Reusser (2003). «Insertion techniques for static and dynamic resource-constrained project scheduling». *European Journal of Operational Research* 149.2, págs. 249-267. ISSN: 0377-2217 (vid. pág. 48).

Ballestín, Francisco y Rosa Blanco (2011). «Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems». *Computers & Operations Research* 38.1, págs. 51-62. ISSN: 03050548 (vid. pág. 95).

Bein, Wolfgang W., Jersy Kamburowski y Matthias F. M. Stallmann (1992). «Optimal reduction of two-terminal directed acyclic graphs». *SIAM Journal on Computing* 21.6, págs. 1112-1129 (vid. págs. 25, 26).

Bianco, L., P. Dell'Olmo y M. Grazia Speranza (1998). «Heuristics for multimode scheduling problems with dedicated resources». *European Journal of Operational Research* 107.2, págs. 260-271. ISSN: 0377-2217 (vid. pág. 6).

Blazewicz, J., J.K. Lenstra y A.H.G.Rinnooy Kan (1983). «Scheduling subject to resource constraints: classification and complexity». *Discrete Applied Mathematics* 5.1, págs. 11-24. ISSN: 0166-218X (vid. págs. 9, 20, 22).

Błażewicz, J. y col. (2007). *Handbook on Scheduling*. International Handbook on Information Systems. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-28046-0 (vid. págs. 2, 30).

Boctor, F. F. (1993). «Heuristics for scheduling projects with resource restrictions and several resource-duration modes». *International Journal of Production Research* 31.11, págs. 2547-2558. ISSN: 0020-7543 (vid. pág. 55).

Boctor, F. F. (1996). «Resource-constrained project scheduling by simulated annealing». *International Journal of Production Research* 34.8, págs. 2335-2351. ISSN: 0020-7543 (vid. págs. 47, 76, 91).

Boglietti, A. y col. (2004). «International Standards for the Induction Motor Efficiency Evaluation: A Critical Analysis of the Stray-Load Loss Determination». *IEEE Transactions on Industry Applications* 40.5, págs. 1294-1301. ISSN: 0093-9994 (vid. pág. 97).

Böttcher, Jan y col. (1999). «Project Scheduling Under Partially Renewable Resource Constraints». *Management Science* 45.4, págs. 543-559. ISSN: 0025-1909 (vid. pág. 6).

Bouleimen, K. y H. Lecocq (2003). «A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode

version». *European Journal of Operational Research* 149.2, págs. 268-281. ISSN: 0377-2217 (vid. pág. 47).

Brinkmann, Klaus y Klaus Neumann (1996). «Heuristic procedures for resource-constrained project scheduling with minimal and maximal time lags: the resource-levelling and minimum project-duration problems». *Journal of Decision Systems* 5.1-2, págs. 129-155 (vid. pág. 8).

Bruzzone, A.A.G. y col. (2012). «Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops». *CIRP Annals - Manufacturing Technology* 61.1, págs. 459-462 (vid. pág. 62).

Cervantes, Mariamar y col. (2008). «A Dynamic Population Steady-State Genetic Algorithm for the Resource-Constrained Project Scheduling Problem». *New Frontiers in Applied Artificial Intelligence*. Ed. por Ngoc Thanh Nguyen y col. Volume 502. Springer Berlin Heidelberg, pp 611-620. ISBN: 978-3-540-69045-0 (vid. pág. 74).

Chen, Wang y col. (2010). «An efficient hybrid algorithm for resource-constrained project scheduling». *Information Sciences* 180.6, págs. 1031-1039. ISSN: 0020-0255 (vid. págs. 51, 88-90).

Christofides, Nicos, R. Alvarez-Valdes y J.M. Tamarit (1987). «Project scheduling with resource constraints: A branch and bound approach». *European Journal of Operational Research* 29.3, págs. 262-273 (vid. pág. 33).

Cucala, a. P. y col. (2012). «Fuzzy optimal schedule of high speed train operation to minimize energy consumption with uncertain delays and drivers behavioral response». *Engineering Applications of Artificial Intelligence* 25.8, págs. 1548-1557. ISSN: 0952-1976 (vid. pág. 94).

De Reyck, Bert, Erik Demeulemeester y Willy Herroelen (1998). «Local search methods for the discrete time/resource trade-off problem in project networks». *Naval Research Logistics (NRL)* 45.6, págs. 553-578. ISSN: 1520-6750 (vid. pág. 30).

De Reyck, Bert y Willy Herroelen (1996). «On the use of the complexity index as a measure of complexity in activity networks». *European Journal of Operational Research* 91.2, págs. 347-366. ISSN: 0377-2217 (vid. págs. 25, 26, 28).

Debels, Dieter y Mario Vanhoucke (2007). «A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem». *Operations Research* 55.3, págs. 457-469. ISSN: 0030-364X (vid. págs. 74, 88-90).

Debels, Dieter y col. (2006). «A hybrid scatter search/electromagnetism metaheuristic for project scheduling». *European Journal of Operational Research* 169.2, págs. 638-653. ISSN: 0377-2217 (vid. págs. 71, 88-90).

Demeulemeester, E. y W. Herroelen (1992). «A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem». *Management Science* 38.12, págs. 1803-1818. ISSN: 0025-1909 (vid. pág. 33).

Demeulemeester, Erik, Bert De reyck y Willy Herroelen (2000). «The discrete time/resource trade-off problem in project networks: a branch-and-bound approach». *IIE Transactions* 32.11, págs. 1059-1069. ISSN: 1573-9724 (vid. pág. 30).

Demeulemeester, Erik L. y Willy. Herroelen (2002). *Project scheduling: a research handbook*. 1.ª ed. Springer US, pág. 686. ISBN: 978-1-4020-7051-8 (vid. pág. 30).

Dorigo, Marco (1992). «Optimization, Learning and Natural Algorithms». Tesis doct. Politecnico di Milano, Italy, in Italian (vid. pág. 49).

Drexl, Andreas (1991). «Scheduling of Project Networks by Job Assignment». *Management Science* 37.12, págs. 1590-1602 (vid. pág. 73).

Duflou, Joost R. y col. (2012). «Towards energy and resource efficient manufacturing: A processes and systems approach». *CIRP Annals - Manufacturing Technology* 61.2, págs. 587-609 (vid. pág. 59).

Erenguc, S. Selcuk, Suleyman Tufekci y Christopher J. Zappe (1993). «Solving time/cost trade-off problems with discounted cash flows using generalized benders decomposition». *Naval Research Logistics* 40.1, págs. 25-50. ISSN: 1520-6750 (vid. pág. 29).

Fahmy, Amer, Tarek M. Hassan y Hesham Bassioni (2014). «Improving RCPSP solutions quality with Stacking Justification - Application with particle swarm optimization». *Expert Systems with Applications* 41.13, págs. 5870-5881. ISSN: 0957-4174 (vid. pág. 53).

Fang, Kan y col. (2011). «A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction». *Journal of Manufacturing Systems* 30.4, págs. 234-240 (vid. págs. 62, 95).

Gahm, Christian y col. (2016). «Energy-efficient scheduling in manufacturing companies: A review and research framework». *European Journal of Operational Research* 248.3, págs. 744-757. ISSN: 0377-2217 (vid. págs. 60, 63).

Glover, Fred (1977). «Heuristics for Integer Programming Using Surrogate Constraints». *Decision Sciences* 8.1, págs. 156-166 (vid. pág. 50).

Glover, Fred y M. Laguna (1997). *Tabu search*. 1.ª ed. Boston, MA: Springer US, pág. 382. ISBN: 978-0-7923-9965-0 (vid. pág. 48).

Gonzalez, Teofilo F. (2007). *Handbook of Approximation Algorithms and Metaheuristics*. 1.ª ed. Chapman y Hall/CRC, pág. 1432. ISBN: 978-1-58488-550-4 (vid. pág. 45).

Hartmann, Söke (2002). «A self-adapting genetic algorithm for project scheduling under resource constraints». *Naval Research Logistics* 49.5, págs. 433-448. ISSN: 0894-069X (vid. págs. 46, 76, 88-90).

Hartmann, Söke y Rainer Kolisch (2000). «Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem». *European Journal of Operational Research* 127.2, págs. 394-407. ISSN: 0377-2217 (vid. págs. 27, 28, 45, 72, 116).

Hartmann, Sönke (1998). «A competitive genetic algorithm for resource-constrained project scheduling». *Naval Research Logistics* 45.7, págs. 733-750. ISSN: 0894-069X (vid. pág. 76).

Hartmann, Sönke (2001). «Project Scheduling with Multiple Modes: A Genetic Algorithm». *Annals of Operations Research* 102.1-4, págs. 111-135. ISSN: 1572-9338 (vid. págs. 48, 49, 53).

He, Yan y Fei Liu (2010). «Methods for Integrating Energy Consumption and Environmental Impact Considerations into the Production Operation of Machining Processes». *Chinese Journal of Mechanical Engineering* 23.4, págs. 428-435 (vid. pág. 59).

Herroelen, Willy, Bert De Reyck y Erik Demeulemeester (1998). «Resource-constrained project scheduling: A survey of recent developments». *Computers and Operations Research* 25.4, págs. 279-302 (vid. págs. 3, 55).

Holland, H. J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, pág. 183 (vid. pág. 46).

Jarboui, B. y col. (2008). «A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems». *Applied Mathematics and Computation* 195.1, págs. 299-308. ISSN: 0096-3003 (vid. págs. 49, 50).

Józefowska, Joanna y col. (2001). «Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling». *Annals of Operations Research* 102.1, págs. 137-155. ISSN: 1572-9338 (vid. págs. 47, 48).

Kennedy, J. y R. Eberhart (1995). «Particle swarm optimization». *Proceedings of ICNN'95 -International Conference on Neural Networks*. Vol. 4. IEEE, págs. 1942-1948. ISBN: 0-7803-2768-3 (vid. pág. 49).

Kim, Jin-Lee (2009). «Proposed methodology for comparing schedule generation schemes in construction resource scheduling». *Proceedings of the 2009 Winter Simulation Conference (WSC)*, págs. 2745-2750 (vid. pág. 40).

Kirkpatrick, S., C. D. Gelatt y M. P. Vecchi (1983). «Optimization by simulated annealing.» *Science (New York, N.Y.)* 220.4598, págs. 671-80. ISSN: 0036-8075 (vid. pág. 47).

Klein, Robert (2000). «Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects». *European Journal of Operational Research* 127.3, págs. 619-638. ISSN: 0377-2217 (vid. págs. 42, 43).

Kobya, M. y col. (2008). «Study on the treatment of waste metal cutting fluids using electrocoagulation». *Separation and Purification Technology* 60.3, págs. 285-291. ISSN: 1383-5866 (vid. pág. 59).

Kochetov, Yu. A. y A. A. Stolyar (2003). «Evolutionary Local Search with Variable Neighborhood for the Resource Constrained Project Scheduling Problem».

Workshop on Computer Science and Information Technologies CSIT 2003, Ufa, Russia, 2003. (Vid. págs. 51, 88-90).

Kolisch, Rainer (1995). «Efficient Heuristics for Several Problem Classes». *Project Scheduling under Resource Constraints*. Physica-Verlag HD, pág. 212. ISBN: 978-3-7908-0829-2 (vid. pág. 22).

Kolisch, Rainer y Sönke Hartmann (1999). «Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis». *Project Scheduling*. Ed. por Jan Weglarz. Springer US. Cap. 7, págs. 147-178. ISBN: 978-1-4615-5533-9 (vid. págs. 10, 35, 42, 73).

Kolisch, Rainer y Sönke Hartmann (2006). «Experimental investigation of heuristics for resource-constrained project scheduling: An update». *European Journal of Operational Research* 174.1, págs. 23-37. ISSN: 0377-2217 (vid. págs. 10, 85, 87).

Kolisch, Rainer y R Padman (2001). «An integrated survey of deterministic project scheduling». *Omega* 29.3, págs. 249-272. ISSN: 0305-0483 (vid. pág. 4).

Kolisch, Rainer y Arno Sprecher (1996). «PSPLIB - A project scheduling library». *European Journal of Operational Research* 96, págs. 205-216 (vid. págs. 10, 55).

Kolisch, Rainer, Arno Sprecher y Andreas Drexl (1992). «Characterization and generation of a general class of resource-constrained project scheduling problems: Easy and hard instances». *Research report No. 301, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Univerität zu Kiel.Germany.* (Vid. págs. 25, 27).

Li, Heng y Hong Zhang (2013). «Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints». *Automation in Construction* 35, págs. 431-438. ISSN: 0926-5805 (vid. pág. 50).

Li, Lin, Jihong Yan y Zhongwen Xing (2013). «Energy requirements evaluation of milling machines based on thermalÂ equilibrium and empirical modelling». *Journal of Cleaner Production* 52, págs. 113-121 (vid. págs. 59, 95).

Li, S., F. Liu y X. Zhou (2016). «Multi-objective energy-saving scheduling for a permutation flow line». *Proceedings of the Institution of Mechanical Engineers,*

Part B: Journal of Engineering Manufacture, pág. 0954405416657583 (vid. pág. 63).

Lobo, Fernando G. y Cláudio F. Lima (2005). «A review of adaptive population sizing schemes in genetic algorithms». *Proceedings of the 2005 workshops on Genetic and evolutionary computation - GECCO '05*. Washington, D.C.: ACM Press, págs. 228-234 (vid. pág. 73).

Lova, Antonio y col. (2009). «An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes». *International Journal of Production Economics* 117.2, págs. 302-316. ISSN: 0925-5273 (vid. pág. 53).

Luo, Hao y col. (2013). «Hybrid flow shop scheduling considering machine electricity consumption cost». *International Journal of Production Economics* 146.2, págs. 423-439 (vid. pág. 63).

Mahdi Mobini, M. D. y col. (2008). «Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem». *Soft Computing* 13.6, págs. 597-610. ISSN: 1432-7643 (vid. págs. 50, 88-90).

Mendes, J.J.M., J.F. Gonçalves y M.G.C. Resende (2009). «A random key based genetic algorithm for the resource constrained project scheduling problem». *Computers & Operations Research* 36.1, págs. 92-109. ISSN: 0305-0548 (vid. págs. 46, 88-90).

Merkert, Lennart y col. (2015). «Scheduling and energy - Industrial challenges and opportunities». *Computers & Chemical Engineering* 72, págs. 183-198. ISSN: 0098-1354 (vid. pág. 61).

Merkle, D., M. Middendorf y H. Schmeck (2002). «Ant colony optimization for resource-constrained project scheduling». *IEEE Transactions on Evolutionary Computation* 6.4, págs. 333-346. ISSN: 1089-778X (vid. pág. 50).

Michalewicz, Zbigniew y David B. Fogel (2000). *How to Solve It: Modern Heuristics*. Springer, pág. 570. ISBN: 3540224947 (vid. pág. 31).

Mingozzi, Aristide y col. (1995). «An Exact Algorithm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation». *Management Science* 44.5, págs. 714-729 (vid. pág. 17).

Möhring, Rolf H. (1984). «Minimizing Costs of Resource Requirements in Project Networks Subject to a Fixed Completion Time». *Operations Research* 32.1, págs. 89-120. ISSN: 0030-364X (vid. pág. 8).

Moradnazhad, M. y H. O. Unver (2016). «Energy efficiency of machining operations: A review». *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, pág. 0954405415619345. ISSN: 0954-4054 (vid. pág. 59).

Morillo, Daniel, Federico Barber y Miguel A. Salido (2017). «Mode List vs Activity List for the Multi-mode Resource Constrained Project Scheduling Problem». *ICAPS 2017 workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS'17)*, págs. 38-47 (vid. pág. 119).

Morillo, Daniel, Federico Barber y Miguel A. Salido (en prensa). «Mode-based vs Activity-based search for a non-redundant resolution of the Multi-mode Resource-Constrained Project Scheduling Problem.» *Mathematical Problems in Engineering.* (Vid. pág. 125).

Morillo, Daniel, Luis Moreno y Javier Díaz (2014a). «Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión Parte 1». *Ingeniería y Ciencia - ing.cienc.* 10.19, págs. 247-271. ISSN: 2256-4314 (vid. pág. 32).

Morillo, Daniel, Luis Moreno y Javier Díaz (2014b). «Metodologías analíticas y heurísticas para la solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 2». *Ingeniería y Ciencia - ing.cienc.* 10.20, págs. 203-227. ISSN: 2256-4314 (vid. pág. 34).

Morillo Torres, Daniel, Federico Barber y Miguel A. Salido (2014). «PSPLIB-ENERGY: Una extensión de la librería PSPLIB para la evaluación de la optimización energética en el RCPSP». *Iberoamerican Journal of Artificial Intelligence* 17.54, págs. 35-48 (vid. pág. 100).

Morillo Torres, Daniel, Federico Barber y Miguel A. Salido (2017). «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*, pág. 095440541771173. ISSN: 0954-4054 (vid. pág. 95).

Morillo Torres, Daniel, Luis Fernando Moreno Velásquez y Francisco javier Diaz Serna (2015). «A branch and bound hybrid algorithm with four deterministic heuristics for the resource constrained project scheduling problem (RCPSP)». *DYNA* 82.190, págs. 198-207 (vid. pág. 52).

Mouzon, Gilles, Mehmet B. Yildirim y Janet Twomey (2007). «Operational methods for minimization of energy consumption of manufacturing equipment». *International Journal of Production Research* 45.18-19, págs. 4247-4271. ISSN: 0020-7543 (vid. págs. 11, 62, 94).

Neumann, Klaus., Christoph. Schwindt y Jürgen. Zimmermann (2003). *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions*. Springer Berlin Heidelberg, pág. 385. ISBN: 978-3-540-24800-2 (vid. pág. 6).

Nolde, Kristian y Manfred Morari (2010). «Electrical load tracking scheduling of a steel plant». *Computers & Chemical Engineering* 34.11, págs. 1899-1903. ISSN: 00981354 (vid. pág. 63).

Nonobe, Koji y Toshihide Baraki (2002). «Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem». *Essays and Surveys in Metaheuristics*. Springer US, págs. 557-588. ISBN: 978-1-4615-1507-4 (vid. págs. 48, 88-90).

Oberg, Erik y col. (2004). *Machinery's Handbook*. Ed. por Christopher J. McCauley, Riccardo M. Heald y Muhammed Iqbal Hussain. 27.ª ed. New York, USA: Industrial Press Inc., pág. 3056. ISBN: 0-8311-2737-6 (vid. págs. 95, 101).

Okubo, Hironori y col. (2015). «Project scheduling under partially renewable resources and resource consumption during setup operations». *Computers & Industrial Engineering* 83, págs. 91-99. ISSN: 0360-8352 (vid. pág. 63).

Ozdamar, L. (1999). «A genetic algorithm approach to a general category project scheduling problem». *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 29.1, págs. 44-59. ISSN: 1094-6977 (vid. pág. 46).

Paraskevopoulos, D.C., C.D. Tarantilis y G. Ioannou (2012). «Solving project scheduling problems with resource constraints via an event list-based evolu-

tionary algorithm». *Expert Systems with Applications* 39.4, págs. 3983-3994. ISSN: 09574174 (vid. págs. 50, 71, 87-90).

Patterson, James H. (1984). «A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem». *Management Science* 30.7, págs. 854-867 (vid. pág. 55).

Paulus, Jacob Jan y Johann Hurink (2006). «Adjacent-Resource Scheduling: Why spatial resources are so hard to incorporate». *Electronic Notes in Discrete Mathematics* 25, págs. 113-116 (vid. pág. 6).

Peng, Wuliang, Wang y Chengen (2009). «A multi-mode resource-constrained discrete time-cost tradeoff problem and its genetic algorithm based solution». *International Journal of Project Management* 27.6, págs. 600-609. ISSN: 0263-7863 (vid. pág. 29).

Peteghem, Vincent Van y Mario Vanhoucke (2010). «A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem». *European Journal of Operational Research* 201.2, págs. 409-418. ISSN: 0377-2217 (vid. pág. 46).

Rahimifard, S., Y. Seow y T. Childs (2010). «Minimising Embodied Product Energy to support energy efficient manufacturing». *CIRP Annals - Manufacturing Technology* 59.1, págs. 25-28 (vid. pág. 94).

Ranjbar, Mohammad, Bert De Reyck y Fereydoon Kianfar (2009). «A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling». *European Journal of Operational Research* 193.1, págs. 35-48. ISSN: 0377-2217 (vid. pág. 31).

Ranjbar, Mohammad R y Fereydoon Kianfar (2007). «Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms». *Applied Mathematics and Computation* 191.2, págs. 451-456. ISSN: 0096-3003 (vid. pág. 31).

Saidur, R. (2010). «A review on electrical motors energy use and energy savings». *Renewable and Sustainable Energy Reviews* 14.3, págs. 877-898. ISSN: 1364-0321 (vid. pág. 97).

Salonitis, Konstantinos y Peter Ball (2013). «Energy Efficient Manufacturing from Machine Tools to Manufacturing Systems». *Procedia CIRP* 7, págs. 634-639 (vid. págs. 11, 59).

Schwindt, C. y N. Trautmann (2003). «Scheduling the production of rolling ingots: industrial context, model, and solution method». *International Transactions in Operational Research* 10.6, págs. 547-563. ISSN: 0969-6016 (vid. pág. 6).

Seow, Y. y S. Rahimifard (2011). «A framework for modelling energy consumption within manufacturing systems». *CIRP Journal of Manufacturing Science and Technology* 4.3, págs. 258-264 (vid. pág. 59).

Słowiński, Roman (1980). «Two Approaches to Problems of Resource Allocation Among Project Activities - A Comparative Study». *Journal of the Operational Research Society* 31.8, págs. 711-723. ISSN: 0160-5682 (vid. pág. 5).

Sprecher, Arno, Sönke Hartmann y Andreas Drexl (1997). «An exact algorithm for project scheduling with multiple modes». *OR Spektrum* 19.3, págs. 195-203. ISSN: 0171-6468 (vid. págs. 33, 45).

Talbi, E. G. (2002). «A Taxonomy of Hybrid Metaheuristics». *Journal of Heuristics, Kluwer Academic Publishers.* 8.1, págs. 541-564 (vid. pág. 51).

Talbot, F. Brian (1982). «Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case». *Management Science* 28.10, págs. 1197-1210. ISSN: 0025-1909 (vid. págs. 6, 21, 22, 32).

Tareghian, Hamed R y Seyyed Hassan Taheri (2006). «On the discrete time, cost and quality trade-off problem». *Applied Mathematics and Computation* 181.2, págs. 1305-1312. ISSN: 0096-3003 (vid. pág. 30).

Thomas, Paul R. y Said Salhi (1998). «A Tabu Search Approach for the Resource Constrained Project Scheduling Problem». *Journal of Heuristics* 4.2, págs. 123-139. ISSN: 1572-9397 (vid. pág. 48).

Tormos, Pilar y Antonio Lova (2001). «A Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling». *Annals of Operations Research* 102.1-4, págs. 65-81. ISSN: 1572-9338 (vid. pág. 53).

Tseng, Lin-Yu y Shih-Chieh Chen (2006). «A hybrid metaheuristic for the resource-constrained project scheduling problem». *European Journal of Operational Research* 175.2, págs. 707-721. ISSN: 0377-2217 (vid. págs. 51, 88-90).

Tseng, Lin-Yu y Shih-Chieh Chen (2009). «Two-Phase Genetic Local Search Algorithm for the Multimode Resource-Constrained Project Scheduling Problem». *IEEE Transactions on Evolutionary Computation* 13.4, págs. 848-857 (vid. pág. 46).

U.S. Energy Information Administration (EIA) (2016). *Monthly Energy Review - October 2016*. Inf. téc. Washington: Office of Energy Statistics, pág. 228 (vid. pág. 10).

Valls, Vicente, Francisco Ballestín y Sacramento Quintanilla (2005). «Justification and RCPSP: A technique that pays». *European Journal of Operational Research* 165.2, págs. 375-386. ISSN: 0377-2217 (vid. pág. 48).

Valls, Vicente, Francisco Ballestin y Sacramento Quintanilla (2008). «A hybrid genetic algorithm for the resource-constrained project scheduling problem». *European Journal of Operational Research* 185.2, págs. 495-508. ISSN: 0377-2217 (vid. págs. 51, 88-90).

Valls, Vicente, Sacramento Quintanilla y Francisco Ballestín (2003). «Resource-constrained project scheduling: A critical activity reordering heuristic». *European Journal of Operational Research* 149.2, págs. 282-301. ISSN: 0377-2217 (vid. pág. 51).

Van Peteghem, Vincent y Mario Vanhoucke (2014). «An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances». *European Journal of Operational Research* 235.1, págs. 62-72. ISSN: 0377-2217 (vid. págs. 34, 55).

Vanhoucke, M, E Demeulemeester y W Herroelen (2002). «Discrete time/cost trade-offs in project scheduling with time-switch constraints». *Journal of the Operational Research Society* 53.7, págs. 741-751. ISSN: 0160-5682 (vid. pág. 29).

Vanhoucke, Mario y Dieter Debels (2007). «The discrete time/cost trade-off problem: extensions and heuristic procedures». *Journal of Scheduling* 10.4, págs. 311-326. ISSN: 1099-1425 (vid. pág. 30).

Vanhoucke, Mario y Dieter Debels (2008). «The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects». *Computers & Industrial Engineering* 54.1, págs. 140-154. ISSN: 0360-8352 (vid. pág. 31).

Węglarz, Jan (1981). «Project Scheduling with Continuously-Divisible, Doubly Constrained Resources». *Management Science* 27.9, págs. 1040-1053. ISSN: 0025-1909 (vid. pág. 5).

Węglarz, Jan y col. (2011). «Project scheduling with finite or infinite number of activity processing modes - A survey». *European Journal of Operational Research* 208.3, págs. 177-205. ISSN: 0377-2217 (vid. págs. 7, 10).

WWF World Wildlife Fund (2010). *Living Planet Report*. Inf. téc., págs. 1-28 (vid. pág. 58).

Zhang, Hong, C. M. Tam y Heng Li (2006). «Multimode Project Scheduling Based on Particle Swarm Optimization». *Computer-Aided Civil and Infrastructure Engineering* 21.2, págs. 93-103 (vid. pág. 49).

Zhang, Zhongwei y col. (2016). «A method for minimizing the energy consumption of machining system: integration of process planning and scheduling». *Journal of Cleaner Production* 137.20, págs. 1647-1662. ISSN: 0959-6526 (vid. págs. 11, 12, 63).

Zhu, Guidong, Jonathan F. Bard y Gang Yu (2006). «A Branch-and-Cut Procedure for the Multimode Resource-Constrained Project-Scheduling Problem». *INFORMS Journal on Computing* 18.3, págs. 377-390. ISSN: 1091-9856 (vid. pág. 33).

A partir de la segunda mitad del siglo XVIII, con la llegada de la revolución industrial, el mundo fue testigo del mayor crecimiento económico, tecnológico y científico en la historia de la humanidad. Los pequeños negocios se transformaron en las grandes multinacionales de hoy en día. Estos avances trajeron consigo la generación de nuevas necesidades en las empresas que buscaban ser competitivas y productivas. La piedra angular de esas organizaciones fue la división de las responsabilidades administrativas, en la cual una empresa es fraccionada en diversos núcleos donde cada uno se especializa en un componente específico. De esta manera, el gran desafío, que las empresas aún comparten, es la asignación óptima de recursos restringidos a las diferentes actividades de cada componente de la organización.

Dentro de este conjunto de problemas, destaca el denominado Problema Muti-modal de Programación de Tareas con Recursos Restringidos (Multi-mode Resource Constrained Project Scheduling Problem, MRCPSP), el cual tiene una gran aplicabilidad en la vida real pero una enrome complejidad de resolución. De manera general, el MRCPSP se define como un problema de asignación de recursos limitados en el tiempo, con el objetivo de secuenciar el conjunto de actividades que componen un proyecto, donde dichas actividades requieren de estos recursos, se relacionan entre ellas, y pueden ser ejecutadas de diferentes modos.

A pesar de que el MRCPSP ha sido ampliamente estudiado a lo largo de las últimas décadas, actualmente han surgido nuevos desafíos, pues las empresas se enfrentan no solo a la gestión de sus recursos sino también a una problemática medioambiental que crece a pasos agigantados. Uno de los principales factores que contribuyen a esta problemática es el alto consumo de energía del sector industrial en el mundo, la cual es producida en su mayoría con recursos no renovables. En este contexto, esta tesis se centra en el estudio y resolución del MRCPSP bajo un enfoque de eficiencia energética. Para ello, se propone una extensión al problema original que tiene en cuenta el consumo de energía y cuyo objetivo es maximizar la eficiencia energética. Se propone un modelo original subyacente y nuevos métodos de resolución basados en una novedosa estrategia de búsqueda. Finalmente, se propone una librería de casos de prueba para evaluar el rendimiento de los métodos de solución. Los resultados obtenidos demuestran la viabilidad de las propuestas realizadas.

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN