Ingeniería y Ciencia
ISSN:1794-9165
ISSN-e: 2256-4314
ing. cienc., vol. 10, no. 19, pp. 247–271, enero-junio. 2014.
http://www.eafit.edu.co/ingciencia

Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1

Daniel Morillo [1], Luis Moreno [2]y Javier Díaz [3]

Recepción: 03-05-2013, Aceptación: 10-09-2013
Disponible en línea: 01-30-2014

MSC:90B35

Resumen

En este artículo se enuncian y describen los métodos de solución más relevantes para el Problema de la Programación de Proyectos con Recursos Restringidos, RCPSP. Se realiza una revisión crítica del estado del arte basado en los trabajos más significativos publicados en la literatura académica sobre el tema. Primero se explican varios métodos de solución exactos y se detallan sus principales ventajas y desventajas, donde se menciona que los mejores algoritmos exactos para la solución de este problema, son los métodos de ramificación y acotamiento o Branch and Bound. Posteriormente, se presentan diversos métodos heurísticos, especialmente aquellos que se han implementado para problemas de secuenciación.

[1] Ph. D.(c). damotor3@posgrado.upv.es, Universitat Politècnica de València, España.
[2] MSc. lfmoreno@unal.edu.co, Universidad Nacional de Colombia, Medellín, Colombia.
[3] Ph. D. javidiaz@unal.edu.co, Universidad Nacional de Colombia, Medellín, Colombia.

Palabras clave: programación de tareas; recursos restringidos; métodos heurísticos; métodos exactos.

Analytic and Heuristic Methodologies for Solving the Resource Constrained Project Scheduling Problem (RCPSP): a review Part 1

Abstract

This paper presents and describes the most relevant methods for the solution of the Resource Constrained Project Scheduling Problem, RCPSP. A critical review of the state of the art, based on the most signicant papers published in the academic literature on this topic is carried out. First, several exact methods of solution are shown and their main advantages and disadvantages are explained; the Branch and Bound methods, considered as the best exact algorithms for solving this problem, are described. Subsequently, several heuristic methods, especially those that have been implemented for sequencing problems, are considered.

Key words: task scheduling, constrained resources, heuristic methods, exact methods.

1 Introducción

Los problemas de secuenciación de actividades tienen como propósito la asignación óptima, en el tiempo, de los recursos escasos. De manera específica el término secuenciación, en investigación operativa, hace referencia a un caso particular de la programación que se entiende como la ordenación de una serie de actividades que guardan alguna relación; la programación, puede llevarse a un caso más general, donde, no sólo se asigne en el tiempo la ejecución de una actividad, sino, por ejemplo se puede asignar el uso de recursos como personas. Uno de los problemas más estudiados, en este contexto, es la programación de tareas con recursos restringidos (Resource Constrained Project Scheduling Problem: RCPSP).

En este trabajo se muestran los orígenes y la evolución de las metodologías de solución reportadas en la literatura más utilizadas para el RCPSP. Además, se propone una clasificación de estas metodologías como respuesta al uso de varios términos ambiguos y a la falta de claridad de algunos conceptos encontrados en la literatura. Se sintetiza la información encontrada,

para brindar al lector una visión conceptual clara y objetiva de los métodos de solución tanto exactos como heurísticos.

El interés en este problema se debe, entre otras razones, a su gran aplicabilidad en la programación de tareas tanto a nivel empresarial como en el ámbito académico. Debido a su complejidad y naturaleza combinatoria, este tipo de problemas es conocido en la literatura como perteneciente a la clase NP-Hard [1], [2]; es decir, el espacio muestral de soluciones factibles crece de manera no polinomial con el número de actividades.

2 Descripción del Problema

De manera formal puede definirse el RCPSP de la siguiente manera [3]: Sea un proyecto compuesto por un conjunto de n actividades $X = (1, \ldots, n)$, cada una de las cuales utiliza una cantidad de recursos r_{ik} para su realización, donde i es la actividad y k es el recurso. Además, b_k es la cantidad total disponible del recurso k y d_i representa la duración de la actividad i. Las actividades 1 y n son actividades ficticias que representan el inicio y la finalización del proyecto [4], con duración y consumo de recursos iguales a cero.

Las actividades están sujetas a dos clases de restricciones, a saber: La primera, las restricciones de precedencia, las cuales consisten en que cada actividad no puede ser iniciada antes de que todas sus actividades predecesoras hayan terminado. La segunda, las restricciones de recursos, las cuales consisten en que para la realización de cada actividad se requiere de unas cantidades de recursos, los cuales son limitados. Mientras una actividad se encuentra activa no se puede disponer, para otra actividad, de esa cantidad de recursos que la primera está utilizando. Sin embargo, si se tuviera mayor disponibilidad de recursos, éstos podrían usarse para hacer otras actividades simultáneas. En este problema, se supone que los recursos usados deben ser renovables; de esta manera, cada vez que se termina una actividad, retorna la cantidad de recursos que utilizó para tener nuevamente estos recursos disponibles.

Se asume, sin pérdida de generalidad, que las actividades se encuentran ordenadas de tal manera que cada actividad predecesora de j se identifica por un valor i numéricamente inferior a j.

La solución del RCPSP está dada por los tiempos de inicio de cada una de las actividades de tal manera que se minimice el tiempo total de terminación del proyecto o makespan.

Sea T_{max} la cota superior del tiempo de terminación del proyecto. Es fundamental hallar cotas superiores e inferiores apropiadas para poder encontrar la solución mediante menor número de iteraciones [5], [6]. Para encontrar cotas inferiores se suele relajar el problema original, es decir, calcular el óptimo para el problema inicial desechando algunas restricciones. De esta manera, se asume que las soluciones del problema relajado son mejores (menor *makespan*) o iguales a las que se pueden encontrar con el problema original. Por lo tanto, si se encuentra una solución para el problema original que sea igual a una cota inferior, se sabe que se llegó al óptimo. La cota inferior LB0, que se estima como la duración total del proyecto calculada mediante el método de la ruta crítica, CPM, sin considerar restricciones de recursos, aparece resaltada en la Figura 1.

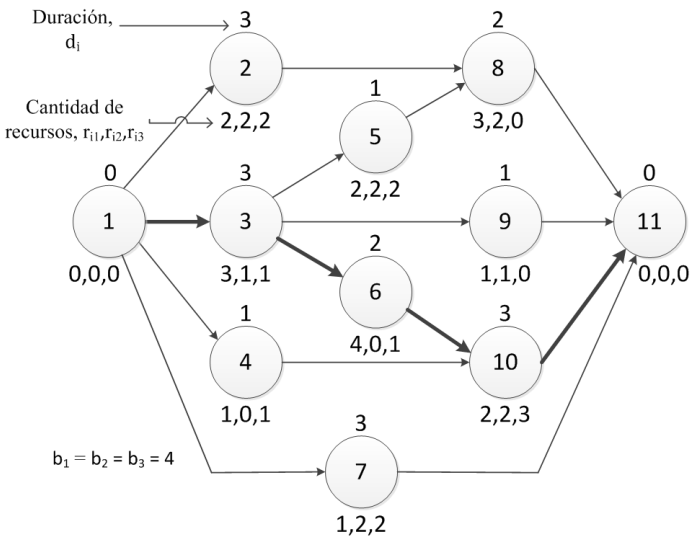


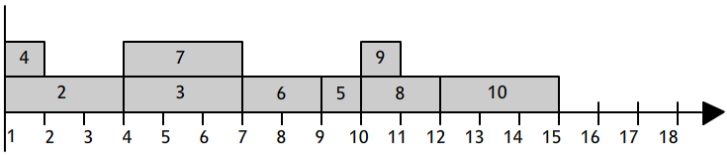
Figura 1: Ejemplo de un RCPSP. Con 11 tareas y 3 recursos [3].

Una forma simple de estimar una cota superior, T_{max}, es mediante la siguiente expresión: $T_{max} = \Sigma_{(i=1)}^{n} d_i$, en la cual se considera la realización secuencial de cada una de las actividades.

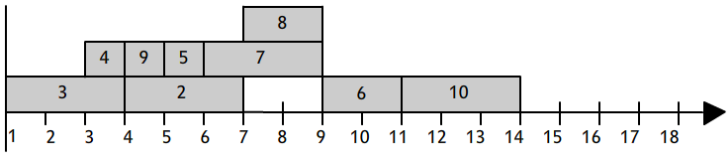
Con base en una cota superior del proyecto es posible estimar, para cada actividad i, intervalos de tiempo (es_i, ls_i), donde es_i es el punto en el tiempo más temprano en donde se puede iniciar la actividad y ls_i es el periodo más tardío en que se puede empezar sin que se retrase la duración total del proyecto. Estos valores pueden calcularse mediante una revisión de la red hacia adelante y hacia atrás, como se hace en el CPM. En la Figura 1 se ilustra un ejemplo del RCPSP adaptado de [3].

En la Figura 1 se muestra un ejemplo que consta de 11 actividades, nueve reales y dos ficticias. Dentro de cada nodo se observa el número de la actividad; en su parte superior, la duración de dicha actividad, y en su parte inferior, el consumo de cada recurso. Cada flecha indica la existencia de una relación de precedencia entre las actividades que conecta.

En la Figura 2, se aprecian los diagramas de Gannt para dos soluciones del problema de la Figura 1.



(a) Solución factible no óptima.



(b) Solución factible óptima.

Figura 2: Soluciones del RCPSP de la Figura 1.

En la Figura 2, el eje horizontal representa unidades de tiempo. Los rectángulos enumerados son las actividades cuya longitud representa su duración. Por ejemplo la actividad 2 se inicia en el punto del tiempo 4, finaliza en 7 y su duración es de 3 unidades. El eje vertical permite la representación de las actividades y la visualización se aquellas que se encuentran superpuestas en el tiempo, es decir que comparten el tiempo de

ejecución con otras actividades.

3 Metodologías de Solución Utilizados para el RSPCP

El origen del RCPSP se remonta a finales de los años 50 del siglo pasado cuando hubo una extensiva investigación acerca de la planeación de proyectos en el área de la investigación operativa, la cual se centra en la administración de tiempos, recursos y actividades para conseguir un objetivo a mediano o corto plazo. Como resultado, se desarrollaron la técnica de revisión y evaluación de programas (Program Evaluation and Review Technique: PERT) y el método la ruta crítica (Critical Path Method: CPM). A partir de estos trabajos se modelaron problemas de secuenciación complejos, los cuales debido a su naturaleza combinatoria no era posible resolver con las metodologías clásicas de programación, en un tiempo razonable. Así, posteriores investigaciones se orientaron al desarrollo de algoritmos heurísticos para la solución de este tipo de problemas.

Luego se definió el RCPSP, para cuya solución se han desarrollado innumerables metodologías usando optimización clásica y métodos heurísticos. Para un estudio más detallado, se remite al lector a los artículos tipo *survey* [7], [8]. Dentro de los métodos exactos pueden resaltarse los eficientes algoritmos basados en ramificación y acotamiento o *branch and bound* [9], [10].

En [11] se presenta un primer algoritmo de ramificación y acotamiento basado en la metodología planteada por [12], donde se transformaba el RCPSP en un caso de encontrar la ruta más corta de un grafo en el cual cada arco representa un conjunto de soluciones factibles. Este algoritmo solo podía resolver problemas pequeños.

Posteriormente, los esfuerzos para resolver el RCPSP se centraron en el desarrollo de algoritmos y metodologías que establecieran mejores cotas inferiores y superiores, con la finalidad de reducir el espacio muestral y las restricciones. El primero en derivar una cota inferior basado en simplificaciones lagrangianas de las restricciones de recursos fue [13]. En [14] se propone un nuevo método para estimar cotas inferiores para el RCPSP basadas en el camino más largo de un modelo con simplificaciones en las restricciones de recursos. Luego, en [15] se muestra que las cotas de Fisher

y similares son mejores que las propuestas por [14]. Sin embargo, el éxito de encontrar unas cotas muy buenas es un problema complejo por la determinación de un buen conjunto de multiplicadores de Lagrange.

En [15] se proponen dos nuevas cotas inferiores, la primera, basada en una relajación de la programación entera clásica del RCPSP; la segunda, basada en un grafo disyuntivo obtenido a partir de las precedencias, adicionándoles arcos que parcialmente representan restricciones de recursos.

En [16] se generó un conjunto de problemas que sirvieron de punto de partida para evaluar el rendimiento (*benchmark*) de los algoritmos que se desarrollaron para solucionar el RCPSP.

El enfoque de ramificación y acotamiento es la mejor técnica exacta propuesta, combinada con el uso de otros métodos, para encontrar buenas cotas inferiores y superiores. Además, su eficiencia aumentó cuando se incorporaron reglas de estrategias dominadas para eliminar ramas del árbol que no contenían el óptimo [16].

En [17] se introducen ciertos parámetros que permiten identificar si el caso del RCPSP es "difícil" o "fácil", conceptos que se explicarán más adelante en este trabajo. Además, se propone un nuevo conjunto de problemas que sirven para realizar un *benchmark* de los nuevos algoritmos [18]. Los resultados muestran que el algoritmo de ramificación y acotamiento propuesto en [9] no logró resolver estos nuevos conjuntos de problemas "difíciles" aunque utilizó un gran tiempo de computo.

En la actualidad, el estudio del RCPSP se centra en la búsqueda de soluciones prácticas para la industria; por lo tanto, se utilizan métodos heurísticos que obtienen resultados satisfactorios, en el sentido de lograr buenas aproximaciones a la solución óptima sin un esfuerzo computacional demasiado elevado [19]. Dentro de los algoritmos heurísticos, que se mencionan repetidamente en la literatura, están los algoritmos genéticos y los de la colonia de hormigas. Además, se observa la propensión a la utilización de algoritmos híbridos. En este ámbito se pueden citar los trabajos [20], [21], [22], [23], [24], [25], [26].

A continuación se presenta una clasificación propuesta por los autores de este trabajo de diversos métodos de solución que se han considerado relevantes para la revisión del estado del arte.

3.1 Métodos Exactos

Dentro de la categoría de métodos exactos se agrupan los algoritmos que tienen como característica el uso de técnicas analíticas o matemáticas, que aseguran la convergencia a una solución óptima, si ésta existe. Estos métodos son diseñados bajo supuestos y características específicas tales como continuidad, diferenciabilidad, espacio de búsqueda pequeño o linealidad, entre otros. Con base en teoremas matemáticos desarrollan procedimientos que garantizan una solución óptima [27].

Por supuesto los métodos exactos no son siempre la respuesta adecuada, ya que presentan varias desventajas que impiden su uso en muchos problemas aplicados. En [27] se afirma que la razón por la cual existen muchos métodos exactos puede deberse a 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 procedimiento de encontrar la solución óptima. Esta problemática suele ser ocasionada por las características inherentes de un problema, ya que éstas pueden impedir el uso de ciertos métodos exactos y crear la necesidad de elaborar otros más apropiados. Sin embargo, existen problemas que bajo su enfoque no pueden ser resueltos debido a su complejidad o al gran tamaño de su espacio de búsqueda.

A continuación se citan algunos de los métodos exactos más representativos en la literatura para la solución del RCPSP.

3.1.1 Búsqueda Exhaustiva (Exhaustive Search) La búsqueda exhaustiva es tal vez uno de los enfoques más antiguos para solucionar un problema y al mismo tiempo el más robusto de los métodos exactos, ya que tiene la ventaja de poderse aplicar a muchos problemas. Sin embargo tiene la desventaja de consumir excesivo tiempo de cómputo. Este método requiere generar y evaluar todas las posibles soluciones dentro del espacio de búsqueda factible.

Es una técnica simple y, además, eficiente en algunos problemas pequeños. Se considera útil para los denominados problemas P, cuyo tiempo de cómputo crece de manera polinomial [28]. Cabe mencionar que en problemas grandes o complejos es necesario tener una clara metodología de la generación de las soluciones y la forma de avanzar en el espacio de búsqueda. A continuación se muestra el pseudocódigo para un algoritmo de

búsqueda exhaustiva 1.

Sea $f(\vec{x})$ una función de aptitud, donde $\vec{x}$ pertenece al espacio factible.

Algoritmo 1 : Algoritmo de Búsqueda Exhaustiva para un problema de minimización

```
Inicio
x_i =primer valor del espacio factible
   mejor= f(x_i), solución=x_i
Hacer hasta x_i = ultimo valor del espacio factible
    x_i = siguiente valor del espacio factible
   prueba = f(x_i)
         si prueba <mejor entonces
      mejor=prueba, solución= x_i
         Fin si
Repetir
Fin
```

Las aplicaciones de este método para la solución del RCPSP son útiles en problemas muy pequeños; sin embargo, por su naturaleza combinatoria, la búsqueda exhaustiva no es una alternativa viable desde el punto de vista práctico, debido a que problemas pequeños, por ejemplo de 30 actividades, según la experiencia lograda en esta investigación, pueden dar millones de soluciones factibles que conforman el espacio de búsqueda de la solución óptima.

3.1.2 Programación Lineal Entera Mixta (Mixed Integer Programming) La Programación Lineal es una de las herramientas más útiles y poderosas de la investigación de operaciones, sobre todo, luego de 1947, cuando George Dantzig desarrolló el conocido método símplex para la solución de problemas lineales tanto en la función objetivo como en sus restricciones. En este contexto, se ha estudiado el desarrollo y la aplicación de nuevos métodos más eficientes de solución.

Esta técnica es robusta y flexible, en el sentido de que ha servido para una gran variedad de problemas (incluido el RCPSP) que han sido formulados y resueltos con éxito. La hipótesis fundamental de la programación lineal es que todas las funciones deben ser combinaciones lineales de las

variables de decisión del problema. La función objetivo es la expresión que se desea minimizar o maximizar.

Las restricciones son las relaciones que delimitan el espacio de búsqueda, es decir, son condiciones que deben cumplirse mientras se busca una combinación de valores de las variables de decisión para optimizar el objetivo [27]. El esquema de un problema de programación lineal se muestra en la ecuación 1, para un problema de maximización.

$$\begin{aligned} Maximizarz(\vec{x}) &= \sum_{j=1}^{n} c_j * x_J \\ sujeto\ a & \\ g_i(\vec{x}) &= \sum_{j=1}^{n} a_{ij} * x_j \leq b_i \text{ donde } i = 1 \cdots m \end{aligned} \tag{1}$$

Para encontrar la solución a estos problemas, el teorema fundamental de la Programación Lineal asegura que si un problema de este tipo define un politopo no vacío P como espacio de búsqueda y si existe una solución óptima finita, ésta necesariamente se encuentra en un punto extremo P, es decir, uno de los vértices del citado politopo.

En los modelos de Programación Lineal se supone que las variables son continuas. Cuando, además, se usan variables enteras o binarias, el modelo es llamado de Programación Lineal Entera Mixta (Mixed Integer Linear Programming). Estos modelos son más complicados de resolver que los de programación lineal, pero existen diversos algoritmos, especialmente diseñados para encontrar sus soluciones.

En [3] se presenta una formulación eficiente del RCPSP como un problema de Programación Lineal Entera Mixta. Sin embargo, ningún método de solución exacta para resolver el modelo matemático asociado ha logrado encontrar el óptimo en un tiempo razonable, aun usando los computadores de más alta tecnología. Algunos autores afirman que sólo es posible encontrar soluciones óptimas mediante métodos exactos para problemas de menos de 60 actividades [29].

Si el RCPSP se modelara mediante una de las formas de programación lineal entera mixta [3], el problema de la Figura 1 tendría un total de 160

variables binarias, 74 restricciones y 39.804 soluciones factibles. Se necesitarían alrededor de 1500 iteraciones con un fuerte algoritmo de optimización exacto para llegar a un óptimo global con un valor en la función objetivo de 13 unidades de tiempo, que representan la duración total del proyecto.

3.1.3 Divide y Vencerás (Divide and Conquer) La esencia del enfoque Divide y Vencerás plantea que un problema aparentemente complicado, podría partirse en sub-problemas más fáciles de resolver. Posteriormente, podría aplicarse una forma previamente diseñada para ensamblar o unir las soluciones de los sub-problemas para construir una solución global óptima [27].

Esta metodología es eficiente sólo cuando el tiempo y el esfuerzo requeridos tanto para la partición del problema como para hallar las soluciones de cada sub-problema y para el ensamble final de la solución, son menores que los usados para la solución del problema original. Sin embargo, es necesario aclarar que no siempre la unión de las soluciones parciales es la solución del problema completo e incluso no garantiza que sea una solución factible.

En el RCPSP puede partirse el problema de muchas formas. Una de ellas es dividir el tiempo en intervalos donde en cada uno se representa un problema de secuenciación de las actividades elegibles no programadas. En este caso la programación final de actividades que se obtiene al ensamblar cada solución parcial sólo será una respuesta factible pero no necesariamente óptima para el problema original. Además, la complejidad del problema haría necesario dividir muchas veces con lo que podría perderse calidad en la solución del problema original.

Un esquema de esta metodología de divide y vencerás puede apreciarse en el Algoritmo 2.

El Algoritmo 2 muestra que el problema principal se remplaza por una colección de sub-problemas que se dividen otra vez y así sucesivamente, de manera recursiva, hasta que los sub-problemas obtenidos finales sean triviales o fáciles de resolver. Luego, el algoritmo debe escalar al siguiente sub-problema hasta llegar al principal.

Algoritmo 2 : Pseudocódigo Divide y Vencerás

```
Inicio
Dividir el problema P en subproblemas
      p_1, p_2, ··· , p_k
para i = 1 hasta k hacer
    si el tamaño de p_i < ε entonces
      Resolver p_i)
      Guardar solución en s_i
      De lo contrario
      Resolver p_i con D&V
    Fin si
Fin para
Combinar las soluciones s_i
Fin
```

3.1.4 Programación Dinámica (Dynamic Programming) La Programación Dinámica es una metodología apropiada para problemas donde deben tomarse decisiones de manera secuencial, por ejemplo, en diferentes periodos de tiempo y en los cuales el orden de las operaciones es crucial [30]. Este enfoque trata de encontrar la solución global para un problema complejo, a partir las soluciones secuenciales de etapas ya resueltas, mediante un procedimiento recursivo. Un problema se puede resolver usando programación dinámica si cumple las siguientes características [27]:

- El problema puede descomponerse en una secuencia de decisiones que deben tomarse en varias etapas.
- Cada etapa tiene un número finito de posibles estados (también existe la programación dinámica continua, aunque no aplica para el RCPSP, en este contexto).
- La decisión tomada en cada estado de la etapa actual lleva a algún estado de la etapa siguiente.
- La mejor decisión asociada a una etapa es independiente de las decisiones tomadas en etapas anteriores.
- Debe estar bien definido el costo asociado por pasar de un estado a otro, a través de las etapas; además, esta función de costos debe ser recursiva.

Existen dos enfoques de la programación dinámica: hacia adelante y hacia atrás. En este último, se inicia a partir del objetivo deseado y se hace un análisis hacia atrás; es decir, se toma la mejor decisión de la última etapa para cada valor de la variable de estado; luego, se toma la mejor decisión de la penúltima etapa, teniendo en cuenta los costos agregados de las dos etapas consideradas. Posteriormente, se retrocede una etapa y se repite el proceso hasta llegar a la primera.

A pesar de que el RCPSP podría cumplir los requisitos de la programación dinámica, haciendo algunos supuestos, en la práctica este método es difícil de aplicar y además puede no llegar a la solución óptima aunque si a una solución factible. Esto se debe a que es necesario tomar decisiones en cada etapa, pero la determinación de estas etapas es una tarea compleja ya que estas varían de manera dinámica durante la ejecución del algoritmo.

3.1.5 Ramificación y Acotamiento (Branch and Bound) Dentro de los algoritmos exactos, el de ramificación y acotamiento es el que mejores resultados ha mostrado para resolver el RCPSP. En [9] se desarrolla un algoritmo, basado en esta técnica, que demostró ser robusto y relativamente eficiente, ya que disminuyó de manera significativa el tiempo de solución del conjunto de 110 problemas del RCPSP propuesto por [16]. Por esta razón, los enfoques que desean enfrentar este problema con metodologías exactas se dedican más a mejorar este método que a buscar uno nuevo. En [16] se publica la librería PSPLIB, desarrollada para evaluar la eficacia y eficiencia de los algoritmos para la solución de problemas de secuenciación y por lo tanto contiene todos los grados de dificultad, basados en los indicadores de complejidad que se mostrarán más adelante. En este trabajo se demuestra la existencia de muchos problemas relativamente pequeños, 30 y 60 actividades, donde el algoritmo de ramificación y acotamiento se demora muchas horas en hallar la solución.

El *Branch and Bound* se considera un método tan robusto como la búsqueda exhaustiva, pero más eficiente. La idea principal es dividir el espacio factible y buscar sólo en donde se sabe que puede estar el óptimo, desechando los espacios de soluciones factibles que no mejoran la solución actual. Esta división puede realizase de una manera particular truncando dicho espacio mientras se está construyendo una solución. A medida que se asigna un conjunto de actividad a la secuencia, se genera una rama que

delimita una gran cantidad de soluciones que comparten las actividades ya programadas en la secuencia. Por tanto, si se desecha esta rama, por medio de una regla de dominancia, concepto que se explicará más adelante, el algoritmo no tendrá que evaluar esa gran cantidad de soluciones delimitadas. Por ejemplo si en la construcción de la solución de la Figura 1 se han asignado las actividades 2 y 4 en el periodo 1, existen muchas secuencias completas de actividades que empiezan con este conjunto. Si se desecha esta rama, se evita buscar todas las secuencias sucesoras de este conjunto.

Adicionalmente, el *Branch and Bound* realiza la búsqueda de manera organizada y sistemática, creando niveles en los puntos del tiempo donde se liberan recursos, en cada uno de los cuales se genera el conjunto de actividades elegibles, es decir, aquellas actividades no programadas cuyas predecesoras ya terminaron su ejecución. De este conjunto de elegibles se generan todos los subconjuntos de actividades factibles por recursos, de las cuales se selecciona uno, de manera descendente por el número de actividades, para generar soluciones completas. Por esta razón, el algoritmo nunca regresa en el árbol y por lo tanto, la búsqueda nunca queda estancada.

La manera como se acota ese gran conjunto de soluciones factibles es mediante reglas de dominancia, en las cuales se usan artificios matemáticos para conocer a priori, sin tener que construir toda la solución, información suficiente para acotar la búsqueda; por ejemplo, puede encontrarse el tiempo mínimo que deberá demorarse para terminar el proyecto a partir de cualquiera de las actividades de una solución parcial; si ese tiempo es mayor que la mejor solución actual, se podrá eliminar esta rama del árbol de soluciones.

En la Figura 3 se muestra un esquema de la expansión organizada de las ramas en el algoritmo de *Branch and Bound*. En los nodos o niveles se muestra el número total de conjuntos válidos (ramas que parten de ese nodo o nivel). En cada uno de los arcos se muestra el conjunto válido programado. Por ejemplo, en el nivel 1 del tiempo existen tres conjuntos válidos; si se selecciona el conjunto 1 se da origen al nivel 2 que tiene cinco conjuntos validos; si en nivel 1 se selecciona el conjunto 2, se da origen a otro nivel 2 completamente diferente (que puede tener un punto en el tiempo distinto). Estos niveles 2, a su vez, generan los niveles 3 y así sucesivamente.

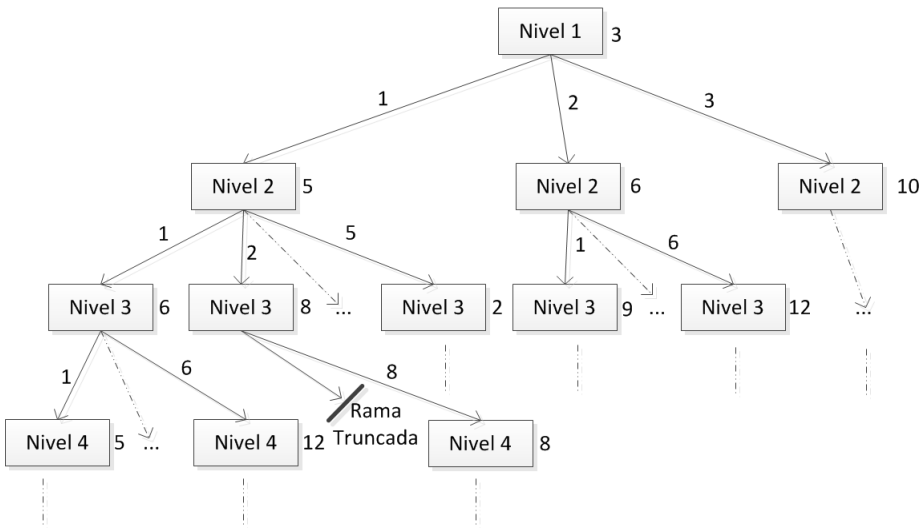


Figura 3: Esquema de le expansión de las ramas en el algoritmo de Branch and Bound.

3.2 Métodos Iterativos de Aproximación

Dentro de la categoría de los Métodos Iterativos de Aproximación se encuentran aquellos que incorporan un proceso iterativo, donde en cada iteración, mediante algún razonamiento, se realiza una búsqueda que permite la construcción o mejora de la solución actual. A veces, el razonamiento es simple pero los resultados no son muy buenos para problemas complejos. Estos métodos se conocen como heurísticas primitivas. Otras veces el razonamiento es tan ingenioso que da origen a las heurísticas o las metaheurísticas cuya implementación en diversos problemas complejos ha mostrado su gran versatilidad y eficiencia. Estos métodos heurísticos comparten la característica de que no pueden garantizar una solución óptima (aunque a veces llegan a obtenerla). A pesar de ello, la práctica ha mostrado que suelen llegar a soluciones muy buenas. En este apartado se mostrarán los algoritmos iterativos de aproximación más utilizados para la solución del RCPSP.

3.2.1 Heurísticos Primitivos (Primitive Heuristics) El término heurístico, en general, puede referirse a aquel procedimiento lógico por medio del cual se pretende solucionar un problema de manera eficiente contando con el conocimiento disponible acerca de él. Este mismo principio puede aplicarse a los algoritmos. De esta manera, en la investigación operativa se entiende por algoritmo heurístico aquel método donde se aplica este principio a un procedimiento de solución de un problema de optimización, del cual se espera encontrar soluciones de alta calidad en un tiempo de cómputo razonable, aunque muchas veces no se puede estimar que tan cerca se encuentra de la solución óptima.

Por otra parte, el razonamiento subyacente en los heurísticos o metaheurísticos es inspirado en algún proceso natural, artificial o físico. Por ejemplo, en el primer caso se tienen el movimiento de los enjambres y la evolución de las especies; en el segundo, la búsqueda tabú; en el tercero, el temple simulado. Los algoritmos metaheurísticos poseen como características fundamentales ser ingeniosos y sencillos pero muy eficientes. Su clave es que incorporan una búsqueda local o fase de explotación, con todas sus bondades, pero cuentan con estrategias que les permiten escapar de los óptimos locales para, de esta manera, darle a la búsqueda mayor poder de exploración.

Debido a que la idea de un heurístico es realmente una estrategia basada en ingenio para resolver problemas de optimización, estos métodos tienen un perfil muy general que permite su aplicación a gran diversidad de problemas, continuos, discretos, lineales, no lineales o combinatorios, entre otros, convirtiendo sus algoritmos en procedimientos robustos y flexibles. Sin embargo, se debe adaptar la idea original usando la información disponible y las características intrínsecas de cada problema particular [31].

En [31] se afirma que las metodologías exactas y heurísticas no deben entrar en conflicto o competencia puesto que estas últimas se usan para resolver problemas con características particulares, las cuales podrían hacer imposible usar los métodos exactos o podrían ser muy costosos en relación al esfuerzo computacional. En particular se recomienda usar heurísticos para problemas que tienen las siguientes características [31]:

- Que no se conozca un método exacto para resolver el problema.
- A pesar de que existan métodos exactos, éstos son incapaces de re-

solverlo en un tiempo razonable. Esta problemática es debida a la complejidad del problema.
- El problema es lo suficientemente complejo y el modelo necesita incorporar numerosas expresiones o complicadas ecuaciones.
- Cuando no se requiere una solución óptima sino que se busca una buena alternativa.
- Cuando encontrar soluciones factibles es muy difícil.
- Si no es posible formular el problema o bien la formulación es demasiado grande para manipularla.
- Si no se cuenta con los recursos necesarios (recursos computacionales o de conocimiento) para implementar un método exacto.

A pesar del estudio permanente de estos métodos, los términos heurísticos y metaheurístico no están del todo dilucidados pues están basados en interpretaciones de lo que es una forma inteligente de resolver un problema. Sin embargo, de manera general, puede decirse que las metaheurísticas son estrategias inteligentes que mejoran y hacen más eficientes las heurísticas primitivas [32].

Las heurísticas primitivas fueron las primeras metodologías iterativas de aproximación que se usaron para solucionar problemas. Se caracterizan por utilizar una simple regla empírica o sentido común que tiene como único objetivo mejorar la solución actual siempre que sea posible. Estas primeras heurísticas se conocen como algoritmos de búsqueda monótona o algoritmos escaladores. Tienen alto riesgo de quedar atrapados en óptimos locales.

A continuación se describen, de manera general, algunos de los algoritmos primitivos más utilizados.

3.2.1.1 Búsqueda Local (Local Search)

La Búsqueda Local es conocida también como *hill climbing* (escalar la montaña). Este método se centra únicamente en una parte de todo el espacio factible, la cual está localizada dentro de una vecindad de una solución en particular. El procedimiento general puede explicarse en cuatro pasos, así: [27]

1. Tomar o encontrar una solución del espacio factible, evaluarla en la

función objetivo y definirla como la solución actual.

2. Aplicar algún método para transformar la solución actual y generar una nueva solución para evaluarla en la función objetivo.
3. Si la nueva solución es mejor que la solución actual, se actualiza esta última como la mejor solución; de lo contario, si la nueva solución no es mejor que la actual, se descarta.
4. Repetir los pasos 2 y 3 hasta que la transformación no obtenga una mejor solución actual.

La clave, para entender cómo funciona este algoritmo, se encuentra contenida en el paso 2, es decir, en la transformación de la solución actual, pues de ésta depende si el algoritmo es eficiente o no. Esta transformación puede ser simple o compleja, la mayoría de las veces fundamentada en alguna característica matemática.

El alcance de la transformación puede abarcar una vecindad grande o pequeña. De esta manera, en un extremo, la transformación podría devolver una solución elegida aleatoriamente en el espacio de búsqueda, sin considerar las soluciones encontradas previamente, permitiendo una exploración con gran amplitud de búsqueda. En el otro extremo, la transformación retornaría una solución basada en la solución actual y próxima a ésta, en cuyo caso tendría poca amplitud de búsqueda pero mayor explotación en regiones vecinas a la solución actual.

Si se considera una vecindad pequeña, la búsqueda evolucionará y convergerá rápidamente, pero con un alto riesgo de quedar atrapada en un óptimo local; por el contrario, si se toma una vecindad grande, el riesgo de quedar estancado es mucho menor pero la búsqueda será lenta y la mejora de la solución será pobre.

Este procedimiento se basa en el principio d optimalidad próxima [33], el cual afirma que las soluciones buenas se encuentran cercanas entre sí y, en consecuencia, comparten una estructura común. Aunque este principio no puede generalizarse para todos los problemas, es muy importante a la hora de diseñar un algoritmo de búsqueda local eficiente. A continuación se muestra el Algoritmo 3, donde se toma $f(\vec{x})$ como una función de aptitud, $\vec{x}$ perteneciente a la región factible.

Algoritmo 3 : Pseudocódigo de Búsqueda Local para minimización

```
1: Inicio
2:    iteración=1, x_0 =solución inicial
3:    mejor=f(x_0), solución= x_0
4:    Hacer hasta iteración= #max
5:       x_i =Transformación ϵ vencidario de (x_0)
6:       prueba= f(x_i)
7:       si prueba<mejor entonces
8:        mejor=prueba, solución= x_i
9:       Fin si
10:    Repetir
11: Fin
```

La gran ventaja de estos algoritmos es que pueden encontrar una solución rápidamente aunque tienen una gran probabilidad de quedar atrapados en un óptimo local. Ésta ha sido la principal dificultad que se encuentra cuando se desea solucionar el RCPSP mediante búsqueda local. Como respuesta a esta dificultad, en algunos trabajos como en [34], se han mejorado estos algoritmo con la utilización de vecindarios grandes; sin embargo, no han sido demasiado eficientes.

3.2.1.2 Algoritmos Codiciosos o Voraces (Greedy Algorithms)

Si bien este tipo de algoritmos suelen usarse en combinación con otros, es conveniente describir su esencia: los algoritmos codiciosos procuran resolver un problema a través de la construcción de una solución en una serie de pasos. La idea principal consiste en asignar valores a todas las variables de decisión, de una en una, seleccionando en cada paso la mejor variable disponible y su valor óptimo.

Por supuesto que este enfoque no es muy eficiente debido a que la toma de decisiones en cada paso, de manera separada, no garantiza obtener la solución óptima global. Por esta razón, los algoritmos codiciosos suelen mezclarse con otros algoritmos para evitar los atrapamientos en óptimos locales.

La manera de obtener las diferentes alternativas para seleccionar la mejor, en cada paso del algoritmo, suele obtenerse mediante una regla heu-

rística dependiendo del problema. Por ejemplo, en el caso del RCPSP, se programan las actividades de menor duración o las de mayor consumo de recursos, entre otros. La razón de la popularidad de este método es su simplicidad y su poco esfuerzo computacional. Aunque sus resultados no siempre son buenos, pueden ser útiles.

3.2.1.3 Métodos Truncados de Branch and Bound

En [35] se plantea un algoritmo de *Branch and Bound* que usa soluciones parciales, construidas heurísticamente, de secuencia en paralelo. Esta heurística consta de dos estrategias, así: la primera, secuencia la actividad con mayor prioridad; la segunda, se activa ocasionalmente programando, además, la actividad con la segunda mayor prioridad. La manera de generar soluciones es en serie.

En [36] se usa un algoritmo de *Branch and Bound*, en el cual se listan las actividades elegibles por precedencias que se ramifican según un subconjunto de ellas; luego, a través del método de vuelta atrás (*backtracking*) se listan todas las actividades elegibles, de las cuales se seleccionan, según reglas de prioridad, las más prometedoras para truncar el árbol de soluciones.

3.2.1.4 Métodos Basados en Arcos Disyuntivos

Un arco conjuntivo se define como la relación de precedencia existente entre dos actividades en la red del proyecto. Un arco disyuntivo es un arco adicional, definido de manera artificial, para representar otro tipo de restricciones (de recursos). Estos métodos tienen como idea principal la adición de nuevas precedencias (arcos disyuntivos) sobre la red ya existente del proyecto, de manera que se creen ciertas precedencias (artificiales) que impidan la existencia de conjuntos de actividades no factibles por recursos.

La creación de estas nuevas precedencias que reemplazan las restricciones de recursos tiene como propósito resolver el problema mediante el método de la ruta crítica. En [37] se presenta un ejemplo de esta metodología.

3.2.1.5 Algoritmos Basados en Secuenciación por Bloques

En [38] se usa una estructura de secuenciación por bloques para obtener una disminución en el tiempo de ejecución del proyecto. Inicialmente, se parte de una solución factible que se construye mediante un esquema generador de secuencias en paralelo, que se describirá detalladamente más adelante en el presente trabajo. Luego, se identifican bloques o intervalos de tiempo que contienen varias actividades desde su inicio hasta su finalización. Posteriormente, se analiza cada bloque de manera independiente y se intenta re-secuenciar las actividades que lo conforman, usando diversos esquemas generadores de nuevas secuencias, tratando, de esta manera, de minimizar el tiempo total del proyecto.

4 Conclusiones

En este trabajo, inicialmente se realiza una revisión crítica del estado del arte en la literatura científica referente al RCPSP, incluyendo definiciones, características, métodos de solución y complejidad. Además, se propone una clasificación de los métodos encontrados, mediante la cual se pretende agruparlos de una manera lógica y organizada, debido a que en varios de los métodos de solución que se han desarrollado durante los últimos años muchas veces se mezclan conceptos que no están clasificados de una manera clara.

Para el RCPSP existen diversos métodos de solución tanto exactos como heurísticos. Cabe resaltar que el mejor de los métodos exactos es el Branch and Bound, que resuelve el problema garantizando optimalidad aunque no es aplicable a problemas demasiado complejos, donde la alternativa eficiente es resolverlos a través de algoritmos heurísticos. Actualmente los más estudiados son los heurísticos basados en poblaciones como los algoritmos genéticos.

Luego de una extensiva búsqueda de información acerca del RCPSP, puede concluirse que el interés de la comunidad científica en esta línea de investigación y específicamente en encontrar nuevos métodos o mezclas de ellos que solucionen el RCPSP, se ha incrementado a través de los años. Además, su solución es de gran interés para industrias de diferentes tipos tanto a nivel local como mundial, debido a su amplio aporte en la progra-

mación y planificación de tareas. Esta última se refiere a toda la gestión relacionada con actividades en el mediano o largo plazo.

Referencias

[1] M. W. Schäffter, "Scheduling with forbidden sets," *Discrete Applied Mathematics*, vol. 72, no. 1-2, pp. 155–166, Jan. 1997. [Online]. Available: http://dx.doi.org/10.1016/S0166-218X(96)00042-X 249

[2] J. Blazewicz, J. K. Lenstra, and K. Rinooy, "Scheduling subject to resource constraints: classification and complexity," *Discrete Applied Mathematics - DAM*, vol. 5, no. 1, pp. 11–24, 1983. 249

[3] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco, "An Exact Algorithm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation," *Management Science*, vol. 44, no. 5, pp. 714–729, 1995. 249, 250, 251, 256

[4] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999. 249

[5] P. Brucker and S. Knust, "Lower bounds for resource-constrained project scheduling problems," *European Journal of Operational Research*, vol. 149, no. 2, pp. 302–313, Sep. 2003. 250

[6] S. E. Elmaghraby, *Activity Networks: Project Planning and Control by Network Models*. John Wiley & Sons Inc, 1977. 250

[7] W. Herroelen, B. De Reyck, and E. Demeulemeester, "Resource-constrained project scheduling: A survey of recent developments," *Computers and Operations Research*, vol. 25, no. 4, pp. 279–302, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.95.1904 252

[8] R. Kolisch and R. Padman, "An integrated survey of deterministic project scheduling," *Omega*, vol. 29, no. 3, pp. 249–272, Jun. 2001. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0305048300000463 252

[9] E. Demeulemeester and W. Herroelen, "A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem," *Management Science*, vol. 38, no. 12, pp. 1803–1818, Dec. 1992. [Online]. Available: http://mansci.journal.informs.org/content/38/12/1803.full.pdf 252, 253, 259

[10] P. Brucker, S. Knust, A. Schoo, and O. Thiele, "A branch and bound algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 107, no. 2, pp. 272–288, Jun. 1998. [Online]. Available: http://dx.doi.org/10.1016/S0377-2217(97)00335-4 252

[11] E. W. Davis and G. E. Heidorn, "An Algorithm for Optimal Project Scheduling under Multiple Resource Constraints," *Management Science*, vol. 17, no. 12, pp. B803–B816, 1971. 252

[12] A. L. Gutjahr and G. L. Nemhauser, "An Algorithm for the Line Balancing Problem," *Management Science*, vol. 11, no. 2, pp. 308–315, 1964. 252

[13] M. Fisher, "Optimal Solution of Scheduling Problems Using Lagrange Multipliers Part1," *Operations Research*, vol. 21, no. 5, pp. 1114–1127, 1973. 252

[14] J. P. Stinson, E. W. Davis, and B. M. Khumawala, "Multiple Resource-Constrained Scheduling Using Branch and Bound," *A I I E Transactions*, vol. 10, no. 3, pp. 252–259, Sep. 1978. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/05695557808975212 252, 253

[15] N. Christofides, R. Alvarez-Valdes, and J. M. Tamarit, "Project scheduling with resource constraints: A branch and bound approach," *European Journal of Operational Research*, vol. 29, no. 3, pp. 262–273, 1987. [Online]. Available: http://ideas.repec.org/a/eee/ejores/v29y1987i3p262-273.html 252, 253

[16] J. H. Patterson, "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem," *Management Science*, vol. 30, no. 7, pp. 854–867, 1984. 253, 259

[17] R. Kolisch, A. Sprecher, and A. Drexl, "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.*, 1992. 253

[18] R. Kolisch and A. Sprecher, "PSPLIB - A project scheduling library," *European Journal of Operational Research*, vol. 96, pp. 205–216, 1996. 253

[19] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 127, no. 2, pp. 394–407, Dec. 2000. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0377221799004853 253

[20] H. Wang, T. Li, and D. Lin, "Efficient genetic algorithm for resource-constrained project scheduling problem," *Transactions of Tianjin University*,

vol. 16, no. 5, pp. 376–382, Oct. 2010. [Online]. Available: http://www.springerlink.com/index/10.1007/s12209-010-1495-y 253

[21] J. F. Gonçalves, M. Resende, and J. Mendes, "A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem," *Journal of Heuristics*, vol. 17, no. 5, pp. 1–20, 2011. [Online]. Available: http://www.ingentaconnect.com/content/klu/heur/2011/00000017/00000005/00009142?crawler=true 253

[22] X.-G. Wang, Y.-W. W. Bai, C.-L. L. Cai, and X. Yan, "Application of Resource-Constrained Project Scheduling with a Critical Chain Method on organizational Project Management," in *2010 International Conference On Computer Design and Applications*, vol. 2. IEEE, Jun. 2010, pp. V2–51–V2–54. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5541070'escapeXml='false'/> 253

[23] S. Elloumi and P. Fortemps, "A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 31–41, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.ejor.2009.12.014 253

[24] J. R. Montoya-Torres, E. Gutierrez-Franco, and C. Pirachicán-Mayorga, "Project scheduling with limited resources using a genetic algorithm," *International Journal of Project Management*, vol. 28, no. 6, pp. 619–628, Aug. 2010. 253

[25] V. Valls, F. Ballestín, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 185, no. 2, pp. 495–508, Mar. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.ejor.2006.12.033 253

[26] L. Deng, V. Lin, and M. Chen, "Hybrid ant colony optimization for the resource-constrained project scheduling problem," *Journal of Systems Engineering and Electronics*, vol. 21, no. 1, pp. 67–71, 2010. 253

[27] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Springer, 2000. [Online]. Available: http://www.amazon.com/How-Solve-It-Modern-Heuristics/dp/3540224947 254, 256, 257, 258, 263

[28] J. Nievergelt, "Exhaustive Search , Combinatorial Optimization and Enumeration : Exploring the Potential of Raww Computing Power," *Proceedings of the 27th Conference on Current Trends in Theory and Practice of Informatics*, pp. 18–35, 2000. 254

[29] V. Valls, F. Ballestín, and S. Quintanilla, "Justification and RCPSP: A technique that pays," *European Journal of Operational Research*, vol. 165, no. 2, pp. 375–386, Sep. 2005. 256

[30] R. Bellman, "Dynamic Programming," Ph.D. dissertation, Princeton University Press, Princeton, N. J., 1957. 258

[31] R. Martí, "Procedimientos Metaheurísticos en Optimización Combinatoria," *Departament d'Estadística i Investigació Operativa, Facultat de Matemàtiques. Universitat de València*, vol. 1, pp. 1–60, 2003. 262

[32] J. B. Santana, C. C. Rodríguez, F. C. García López, M. García Torres, B. M. Batista, J. A. Moreno Pérez, and J. M. Moreno Vega, "Metaheurísticas: Una revisión actualizada." *Universidad de La Laguna, España: Departamento de Estadística, Investigación Operativa y Computación*, 2004. 263

[33] F. Glover and M. Laguna, "Tabu search," *Boston: Kluwer Academic Publishers*, 1997. 264

[34] M. Palpant, C. Artigues, and P. Michelon, "LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search," *Annals of Operations Research*, vol. 131, no. 1-4, pp. 237–257, Oct. 2004. 265

[35] B. Pollack-Johnson, "Hybrid structures and improving forecasting and scheduling in project management," *Journal of Operations Management*, vol. 12, no. 2, pp. 101–117, Feb. 1995. [Online]. Available: http://dx.doi.org/10.1016/0272-6963(94)00008-3 266

[36] A. Sprecher, "Solving the RCPSP Efficiently at Modest Memory Requirements," *Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel.*, vol. 425, p. 27, 1996. 266

[37] C. E. Bell and J. Han, "A new heuristic solution method in resource-constrained project scheduling," *Naval Research Logistics*, vol. 38, no. 3, pp. 315–331, Jun. 1991. [Online]. Available: http://doi.wiley.com/10.1002/1520-6750(199106)38:3<315::AID-NAV3220380304>3.0.CO;2-7 266

[38] H. E. Mausser and S. R. Lawrence, "Exploiting Block Structure to Improve Resource-Constrained Project Schedules," in *Meta-Heuristics*, I. H. Osman and J. P. Kelly, Eds., 1996, pp. 203–217. 267