# Formulaciones matemáticas y heurísticos simples para solucionar problemas de programación de proyectos con recursos limitados

*Rene Viveros y Juan Carlos Rivera*
renevive@gmail.com, jrivera6@eafit.edu.co
Departamento de Ciencias Matemáticas, Escuela de Ciencias,
Universidad EAFIT, Medellín, Colombia

13 de junio de 2017

**Resumen**

En este artículo se trata el Problema de Programación de Proyectos con Recursos Limitados, más conocido como RCPSP (sigla de Resource Constrained Project Scheduling Problem). El RCPSP es un problema proveniente del área de programación de producción y de construcción, aunque sus aplicaciones se extienden a diversas áreas del conocimiento. En esta investigación se evalúan cinco formulaciones matemáticas diferentes que se encuentran en la literatura, se presenta un heurístico simple compuesto por nueve procedimientos de solución independientes y se comparan diferentes propuestas para combinar las soluciones heurísticas obtenidas con la formulación que presenta los mejores resultados. Entre los métodos resultantes se encuentran tanto soluciones exactas como heurísticas. El desempeño de los algoritmos es evaluado utilizando las instancias de la libraría PSPLIB de la literatura. Los resultados obtenidos comprueban que tener buenas soluciones heurísticas puede ser útil para acelerar la convergencia de los modelos matemáticos en la búsqueda de soluciones óptimas. Sin embargo, el uso de nuevas restricciones, añadidas de manera heurística, en los modelos matemáticos no es garantía de obtener buenas soluciones ni menores tiempos de cómputo.

**Palabras Clave:** RCPSP, Programación de proyectos, Optimización combinatoria, Algoritmos heurísticos, Programación lineal entera mixta, algoritmos híbridos.

## 1. Introducción

El problema de programación de proyectos o project scheduling puede definirse de manera muy general como el problema de organizar o secuenciar una serie de operaciones y asignarles tiempos de inicio y de finalización de forma que no se viole ninguna de las restricciones, precedencias, recursos u otro tipo, impuestas en el sistema.

En este artículo se estudia el problema de optimización combinatoria denominado programación de proyectos con recursos limitados, más conocido como RCPSP o *resource constrained project scheduling problem*. La programación de proyectos con recursos limitados es un problema en el cual se deben procesar un conjunto de actividades sujetas a restricciones de precedencias y recursos, siendo estos últimos compartidos por varias actividades. Así, el problema consiste en realizar tal asignación optimizando alguna función objetivo.

La importancia de este problema de optimización proviene tanto de sus aplicaciones prácticas como desde el punto de vista teórico. Entre las aplicaciones más comunes del RCPSP se encuentran proyectos civiles de construcción, industriales, investigación, desarrollo de software, operaciones de mantenimiento, entre otras. En el aspecto teórico, dada su pertenencia a la clase de problemas NP-Hard (Blazewicz *et al.*, 1983), no existen algoritmos que resuelvan el problema de manera óptima en tiempo polinomial. Así, el desarrollo de algoritmos eficientes para su solución es un área de investigación de gran interés.

Las estrategias solución para este tipo de problemas se pueden clasificar en dos grandes categorias: a.) los algoritmos exactos, que garantizan la solución óptima, y b.) los algoritmos heurísticos y metaheurísticos que, aunque no garantizan la solución óptima del problema, producen soluciones factibles cercanas a un óptimo en un tiempo de procesamiento razonable.

En esta investigación se propone utilizar ambos enfoques de manera conjunta para obtener soluciones heurísticas de buena calidad o soluciones óptimas en tiempos de ejecución menores. A continuación, este artículo está organizado de la siguiente forma: En la Sección 2 se presenta una revisión de la literatura presentando las diferentes estrategias de solución utilizadas para el RCPSP, mientras que en la Sección 3 se presenta la definición formal del problema. En la Sección 4 se presenta la metodología de solución propuesta. La Sección 5 resume los resultados obtenidos y, finalmente, la Sección 6 presenta las conclusiones finales.

## 2. Revisión de la literatura

Diferentes metodologías han sido propuestas para encontrar soluciones para el RCPSP. Dichos métodos pueden ser clasificados en dos categorías principales: métodos exactos y los métodos heurísticos o metaheurísticos. Una tercera categoría contiene los métodos híbridos que combinan ambas metodologías: matheurísticos. A continuación se mencionan algunos de los principales métodos encontrados en la literatura en cada categoría.

### 2.1. Enfoques exactos

La característica principal este tipo de métodos es que garantizan la obtención de una solución óptima, cuando dicha solución existe. Dichos métodos están basados en técnicas de optimización como programación lineal entera mixta, branch and bound (o ramificación y acotamiento), programación dinámica, entre otras. A continuación se describen algunos de los algoritmos de esta categoría encontrados en la literatura.

Fisher (1973a) y Fisher (1973b) utilizan técnicas basadas en multiplicadores de lagrange en un algoritmo de branch and bound para hallar soluciones óptimas al problema. Patterson & Huber (1974) presentan un método basado en evaluaciones de factibilidad en un modelo de programación binaria. Patterson & Roth (1976) utlizan un algoritmo de enumeración implícita para resolver un modelo de programación binaria. Otro algoritmo de enumeración implícita basado en un modelo de programación entera es propuesto por Patterson & Huber (1974).

Algoritmos de tipo Branch and Bound son propuestos por Willis & Hastings (1976), Christofides *et al.* (1987), Demeulemeester & Herroelen (1992), Demeulemeester & Herroelen (1997), Brucker *et al.* (1998) y Sprecher (2000). Simpson & Patterson (1996) también proponen un algoritmo de búsqueda en árbol, el cual hace uso de procesamiento en paralelo para obtener soluciones en menor tiempo.

Pritsker *et al.* (1969) y Mingozzi *et al.* (1998) proponen modelos de programación lineal entera (MILP) para el RCPSP. Damay *et al.* (2007) también proponen un modelo MILP, el cual resuelve el problema con o sin interrupción de la ejecución de las actividades. Kopanos *et al.* (2014) proponen cuatro modelos de programación lineal entera considerando tiempos discretos y continuos. Koné *et al.* (2011) hace una comparación de diferentes modelos MILP y propone dos nuevas formulaciones basadas en eventos. Artigues *et al.* (2013) presenta una modificación sobre uno de los modelos propuestos por Koné *et al.* (2011), demostrando que dicho modelo, bajo ciertas condiciones, puede presentar soluciones infactibles. Más recientemente Palacio & Larrea (2017) presentan modelos MILP con un enfoque lexicográfico para hallar soluciones para el RCPSP minimizando la robustez sujeto a un makespan óptimo.

Sin embargo, dada la naturaleza combinatoria de estos problemas se hace matemáticamente imposible resolver casos prácticos en tiempos razonables, aún mediante las herramientas computacionales de alto

rendimiento disponibles actualmente.

De acuerdo con investigaciones previas (Valls *et al.*, 2005), se ha encontrado que los métodos exactos actuales solo son capaces de garantizar la optimilidad de una solución para instancias de problemas con 60 actividades o menos. Sin embargo, proyectos con menos de 60 actividades son considerados pequeños comparados con casos reales. Debido a lo anterior, surge la necesidad de diseñar algoritmos eficientes que obtengan soluciones de buena calidad, aunque esta última no sea óptima.

## 2.2. Algoritmos heurísticos y metaheurísticos

La dificultad para encontrar soluciones óptimas en tiempos razonables utilizando métodos exactos ha dado paso a la utilización de métodos heurísticos y metaheurísticos. Éstos, aunque no garantizan la obtención de soluciones óptimas, permiten encontrar soluciones cercanas a un óptimo en tiempos de cómputo razonables. Dichos algoritmos pueden estar basados en la mejora de una solución incumbente, llamados algoritmos basados en búsqueda en vecindarios, o en la evolución de un conjunto de soluciones, llamados algoritmos evolutivos.

Uno de los algoritmos ampliamente usado para el RCPSP es el denominado Justificación. La Justificación es una técnica sencilla y rápida que al aplicarla sobre una solución del RCPSP produce otra de igual o menor duración (Valls *et al.*, 2003a). Wiest (1964) introdujo los conceptos de justificación a la izquierda y a la derecha, aunque el objetivo era extender los conceptos de holgura y ruta crítica del caso de recursos no limitados al caso con limitación de recursos. En la Sección 4 se presenta una descripción del procedimiento.

Algunos de los algoritmos metaheurísticos basados en búsqueda en vecindarios más eficientes se describen a continuación. Pesek *et al.* (2007) presenta algoritmos híbridos basados en procedimientos de Búsqueda Local. El metaheurístico Simulated Annealing (Recocido Simulado) ha sido utilizado para hallar soluciones al RCPSP por Koulamas *et al.* (1994), Boctor (1996), Cho & Kim (1997) y Bouleimen & Lecocq (2003). El método Tabu Search (Búsqueda Tabú) ha sido utilizado por Thomas & Salhi (1998), Baar *et al.* (1999), Nonobe & Ibaraki (2002), Valls *et al.* (2003b), Artigues *et al.* (2003) y Rivera *et al.* (2006). Un Algoritmo de búsqueda con múltiples vecindarios (Variable Neighborhood Search o VNS) es desarrollado por Fleszar & Hindi (2004). Rivera & Celín (2010) combinan los métodos VNS y Recocido Simulado. Un Adaptive Large Neighborhood Search (ALNS) basado en movimientos de destrucción y reconstrucción es propuesto por Muller (2009). Naphade *et al.* (1997) presentan un método llamado Problem Space Search, el cual usa perturbaciones en los datos, un heurístico básico y una estructura de vecindad. Un algoritmo LNS (Large Neighborhood Search) es propuesto por Palpant *et al.* (2004) para el RCPSP, el cual utiliza una fase de diversificación y una fase de generación y solución de subproblemas de optimización de menor escala. He *et al.* (2016) presenta un algoritmo basado en la estrategia Filter-and-fan, llamado filter-and-fan approach with adaptive neighborhood switching (FFANS), el cual usa cuatro vecindarios y un procedimiento de búsqueda local. Moreno *et al.* (2007) comparan dos algoritmos metaheurísticos: Recocido Simulado y Búsqueda tabú.

Entre los algoritmos evolutivos se encuentran los Algoritmos Genéticos propuestos por Leon & Balakrishnan (1995), Hartmann (1998), Ozdamar (1999), Alcaraz & Maroto (2001), Hartmann (2002), Kim *et al.* (2003), Wang *et al.* (2005), Li *et al.* (2007), Debels & Vanhoucke (2007), Valls *et al.* (2008), Proon & Jin (2011). El método Scatter Search (Búsqueda Dispersa) combinado con Path Relinking (Reencadenamiento de Trayectorias) ha sido aplicado por Valls *et al.* (2004), Baradaran *et al.* (2010) y Rivera *et al.* (2013). Debels *et al.* (2006) combinan Scatter Search con la metaheurística de electromagnetismo. Merkle *et al.* (2002) y Bautista & Pereira (2002) proponen algoritmos de colonias de hormigas (Ant Colony Optimization – ACO). Tseng & Chen (2006) presentan el metaheurístico híbrido ANGEL, el cual combina Optimización de Colonias de Hormigas, Algoritmos Genéticos y Búsqueda Local. Colak *et al.* (2006) presenta un método híbrido basado en redes neuronales. Por su parte, Chen (2011) y Koulinas *et al.* (2014) utilizan algoritmos del tipo Particle Swarm Optimization (PSO). Wang & Fang (2012) desarrollan un algoritmo híbrido de

estimación de distribuciones para el RCPSP. Un enfoque basado en Programación Evolutiva es presentado por Sebt & Alipouri (2013). Un algoritmo de colonias de abejas (Artificial Bee Colony – ABC) es propuesto por Crawford *et al.* (2015).

De Magalhães *et al.* (2005) propone un Algoritmo Genético con una función objetivo alternativa para el makespan llamada makespan modificado (modified makespan). La idea básica radica en que diferentes soluciones pueden dar como resultado un mismo makespan; sin embargo, algunas pueden ser mejoradas más fácilmente, a lo que los autores llaman potencial de mejoramiento. Lee & Kim (1996) presentan un procedimiento de búsqueda implementado en tres metaheurísticos: Recocido Simulado, Búsqueda Tabú y Algoritmos Genéticos. Kochetov & Stolyar (2003) usan una combinación de Algoritmos Genéticos, GRASP, Búsqueda Tabú y Reencadenamiento de Trayectorias. Debels & Vanhoucke (2005) desarrollan un algoritmo genético que usa dos poblaciones, una justificada a la derecha y la otra justificada a la izquierda. Dichas poblaciones alternan el sentido de su justificación en cada generación.

Yan *et al.* (2009) presenta una aplicación novedosa en rescates marítimos y un heurístico simple para el RCPSP con instancias entre 21 y 50 actividades.

Para mayor información, Kolisch & Hartmann (2006) y Wu *et al.* (2014) presentan revisiones del estado del arte sobre los métodos para hallar soluciones al RCPSP. Lancaster & Ozbayrak (2007) describen el estado del arte de algoritmos evolutivos para el RCPSP. En Hartmann & Briskorn (2010) se presenta un estudio que ofrece un visión general sobre diferentes extensiones de los problemas de programación de proyectos y su clasificación. Morillo *et al.* (2014a) y Morillo *et al.* (2014b) realizan una revisión de técnicas exactas y heurísticas para el RCPSP, así como los índices de complejidad más usados.

## 3. Descripción del problema

El problema de programación de proyectos con recursos limitados (RCPSP) puede ser descrito matemáticamente de la siguiente manera (basado en las definiciones de Mingozzi *et al.* (1998), Tseng & Chen (2006) y Valls *et al.* (2005)):

Se tiene un conjunto $J = \{1, \ldots, n\}$ de $n$ actividades a ser procesadas. Asociada a cada actividad $j \in J$, existe una duración $d_j$. Además, las actividades están relacionadas mediante restricciones de precedencia, siendo $P_j \in J \setminus \{j\}$ el conjunto de todas las actividades predecesoras inmediatas de la actividad $j$, es decir, actividades que deben ser completadas antes de iniciar la ejecución de la actividad $j$. Las restricciones de precedencia pueden estar representadas por un grafo dirigido acíclico $G = (J, H)$ donde $H = \{(i, j) | i \in P_j,\ j \in J\}$. Adicionalmente, existe un conjunto $K = \{1, \ldots, m\}$ de $m$ tipos de recursos renovables, donde cada tipo de recurso $k \in K$ tiene una disponibilidad (capacidad) total $R_k$ durante cada intervalo de tiempo del período de programación, es decir, la suma de la cantidad utilizada del tipo de recurso $k$ en el periodo $t$, $R_k(t)$, no debe exceder $R_k$ para todo $t$. Cada actividad $j$ requiere una cantidad constante de $r_{jk}$ unidades del recurso de tipo $k$ durante todo el intervalo de su duración. Se asume, sin pérdida de generalidad, que $r_{jk} \leq R_k$, lo cual garantiza la existencia de soluciones factibles. Todas las cantidades $d_j$, $r_{jk}$ y $R_k$ son números enteros no negativos para todo $j \in J$ y todo $k \in K$.

No se permite interrumpir el procesamiento de las actividades y se asume que los tiempos de alistamiento están incluidos en los tiempos de procesamiento o no son significativos con respecto a estos últimos. Una vez que un recurso es ocupado por una actividad, este no es liberado hasta que la actividad es terminada.

Las actividades 1 y $n$ son actividades ficticias, usadas con el objetivo de representar el inicio y la finalización del proyecto: la actividad 1 debe ser completada antes de iniciar las actividades $J \setminus \{1\}$ y la actividad $n$ sólo puede comenzar una vez se hayan completado las actividades $J \setminus \{n\}$. Se asume además que $d_1 = d_n = 0$ y $r_{1k} = r_{nk} = 0$, para todo $k \in K$.

El objetivo es encontrar un programa $S$ con tiempos de inicio $s_i$ para toda actividad $i \in J$ de tal forma que se satisfagan las restricciones de precedencia y de recursos, y que el makespan (duración) del proyecto ($Z = s_n$) sea mínimo.

En la Figura 1 se presenta un ejemplo de un grafo que representa un proyecto compuesto por once actividades ($n = 11$), dos de las cuales son ficticias, y tres tipos de recursos ($m = 3$). Cada nodo del grafo corresponde a una actividad y las flechas representan las relaciones de precedencia entre las diferentes actividades. A cada nodo del grafo le corresponde una duración y consumos de cada tipo de recurso, los cuales se encuentran ubicados en la parte superior e inferior respectivamente. La capacidad de cada tipo de recurso es igual a 4 unidades.

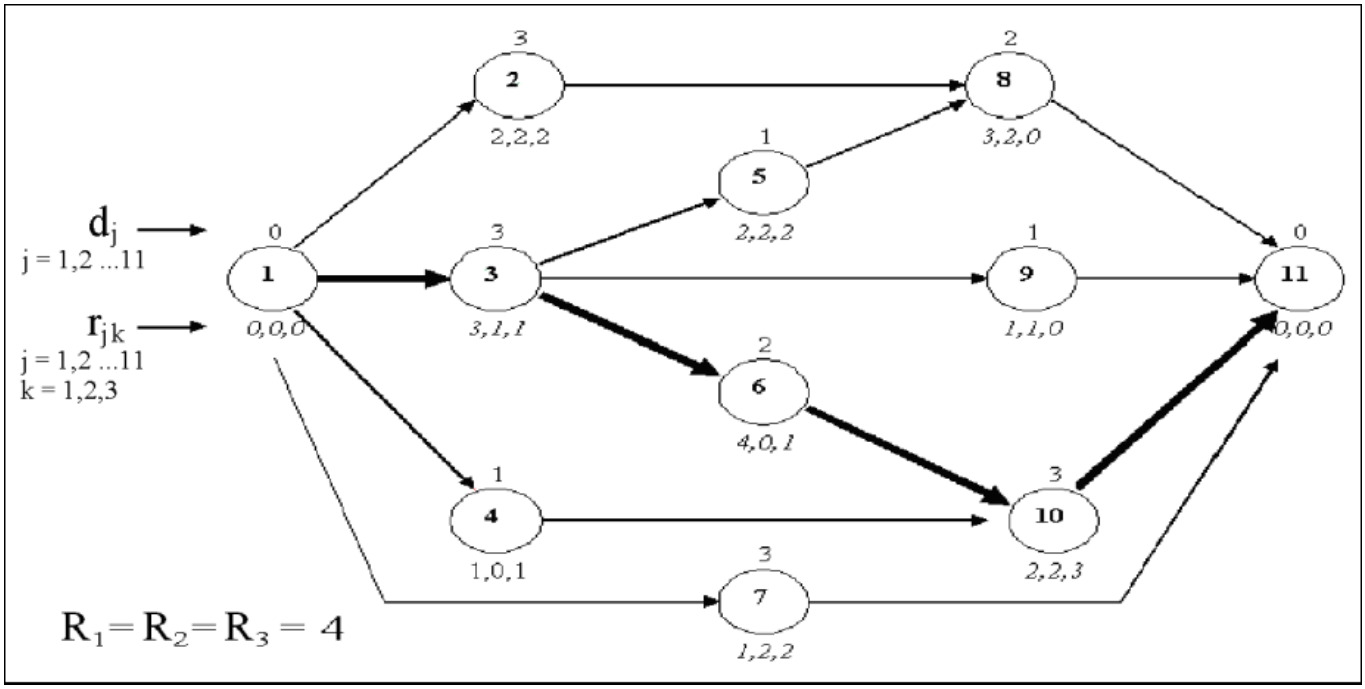


Figura 1: Grafo ejemplo del RCPSP. Tomado de Mingozzi *et al.* (1998).

De acuerdo con Abbasi *et al.* (2006), el RCPSP es el problema más importante en Programación de Proyectos. Además, el RCPSP es una generalización de otros problemas de programación de producción como Flow Shop, Job Shop y Open Shop (Blazewicz *et al.*, 1983; Merkle *et al.*, 2002).

Aunque este problema no es el más general ya que usa tiempos de procesamiento determinísticos para las actividades, los recursos utilizados son renovables (no considera recursos no renovables que involucran el manejo de inventarios y costos), existe una única forma de realizar las actividades (contrario al caso multimodal, en el cual las actividades presentan posibilidades disyuntivas y por lo tanto diferentes consumos de recursos y duraciones), no se permite interrumpir el procesamiento de las actividades, entre otras características, resolverlo eficientemente sigue siendo de gran interés para la comunidad científica ya que su pertenencia a la clase de problemas NP-Hard (Blazewicz *et al.*, 1983) hace que sea un problema difícil de resolver, para el cual no se conocen algoritmos que lo resuelvan exactamente de manera eficiente.

## 4. Metodología de solución

En esta sección se describen los algoritmos propuestos así como los modelos matemáticos usados para resolver el RCPSP en este trabajo de investigación. Además, se describen varias formas en que estos

métodos pueden ser combinados para acelerar la búsqueda de buenas soluciones.

Los métodos de solución propuestos están basados en algoritmos heurísticos de construcción, búsqueda local, y formulaciones matemáticas de programación lineal entera (*Mixed Integer Linear Programming* – MILP).

## 4.1. Métodos constructivos

En esta sección se describen dos algoritmos heurísticos constructivos para solucionar el RCPSP. El primer algoritmo, *algoritmo basado en holguras*, se basa en el algoritmo *CPM* (*Critical Path Method*) para el problema sin restricciones de recursos e iterativamente programar las actividades con menor holgura. El segundo algoritmo, *algoritmo basado en adición de precedencias*, también basado en el algoritmo *CPM*, agrega iterativamente precedencias virtuales sobre parejas de actividades hasta satisfacer todas las restricciones de recursos. Adicionalmente, se propone una transformación al problema de modo que con los mismos métodos anteriores se puedan usar para proporcionar nuevas soluciones en espacios de búsqueda diferentes. La combinación de éstos procedimiento permite obtener cuatro soluciones diferentes para el RCPSP.

Las descripciones detalladas de dichos algoritmos se presenta a continuación. Para una descripción detallada del algoritmo *CPM* el lector puede referirse a Demeulemeester & Herroelen (2002).

### Algoritmo basado en holguras

Este algoritmo inicia aplicando el algoritmo *CPM*, el cual incluye el cálculo de los tiempos de inicio más cercano para cada actividad $j$ ($es_j$), tiempo de terminación más cercano ($ef_j$), tiempo de inicio más lejano ($ls_j$), tiempo de terminación más lejano ($lf_j$) y holgura para cada actividad del proyecto ($ss_j$).

Debido a que la programación resultante puede violar las restricciones de disponibilidad de recursos en algún intervalo de tiempo, dicha programación se considera temporal y será modificada en los pasos sucesivos. Las actividades programadas en esta forma se almacenan en el conjunto $B$ (inicialmente $B = J$). Se define el conjunto $A$ como el conjunto de actividades programadas con consideraciones de sus requerimientos de recursos, donde $A = \{\emptyset\}$ inicialmente.

A continuación se busca una actividad $i \in B$ con menor holgura y se programa de tal forma que su tiempo de inicio sea lo menor posible y se respeten todas las restricciones de precedencia y las restricciones de recursos entre las actividades en $A$. Cada vez que se programa una actividad, ésta se remueve del conjunto $B$, se añade al conjunto $A$, y se recalculan los valores $es_j$, $ls_j$, $ef_j$, $lf_j$, $ss_j$ de cada actividad $j \in B$. Las actividades $i \in A$ mantienen los tiempos de inicio y terminación calculados en el paso anterior. El algoritmo termina cuando todas las actividades han sido programadas ($A = J$).

### Algoritmo basado en adición de precedencias

Al igual que el *algoritmo basado en holguras*, este algoritmo se basa en el algoritmo *CPM*. Inicialmente se ejecuta el algoritmo CPM en donde se calculan los tiempos de inicio más cercano y tiempos de finalización más cercanos de cada actividad. Luego, iterativamente se asignan los recursos necesarios para realizar todas las actividades con dichos tiempos de inicio y finalización. Durante el proceso de asignación de recursos se encuentra el primer intervalo de tiempo $t$ en el que se requieren más unidades de algún recurso que las disponibles, se busca el conjunto de actividades $A(t)$ cuyos tiempos de inicio más cercano sea menor o igual a $t$, tiempos de terminación más cercanos sean mayores a $t$ y utilicen al menos una unidad de alguno de los recursos en conflicto. De dicho conjunto $A(t)$ se selecciona la actividad $i$ con menor tiempo de inicio más cercano y la actividad $j$ con mayor tiempo de terminación más lejano, y se agrega la relación de precedencia (ficticia) $(i, j)$. En este punto se ejecuta nuevamente el algoritmo *CPM* considerando el

conjunto de precedencias actualizado. El procedimiento se repite hasta que no se encuentren violaciones a las restricciones de recursos.

#### Grafo invertido

Un procedimiento adicional está relacionado con la construcción de soluciones sobre un grafo $G' = (J, H')$ donde los nodos en $J$ corresponden a los nodos del grafo original $G$ (ver Sección 3), y las relaciones de precedencia en $H'$ equivalen a $H' = \{(i, j) \mid (j, i) \in H\}$. Los requerimientos de recursos así como las duraciones de las actividades son equivalentes a los del problema original. Utilizando este grafo, dos nuevas soluciones pueden ser obtenidas mediante la aplicación de los dos procedimientos de construcción descritos anteriormente.

En la sección de resultados se presenta una comparación del desempeño de los cuatro algoritmos resultantes.

### 4.2. Algoritmos de mejoramiento

Dos algoritmos de mejoramiento son considerados para resolver el problema: El primero es conocido como *Justificación* o *FBI* (Forward-backward Improvement) (Valls *et al.*, 2005). El segundo algoritmo es un procedimiento de búsqueda con múltiples vecindades conocido como *VND* (Variable Neighborhood Descend) (Hansen *et al.*, 2010). Dichos procedimientos de mejoramiento son descritos a continuación.

#### Justificación

La Justificación es una técnica sencilla y rápida que al aplicarla sobre una solución del RCPSP produce otra de igual o menor duración (Valls *et al.*, 2003a). La siguiente descripción del procedimiento está basada en la presentada en Valls *et al.* (2005).

Dado un programa $S$, definido por los tiempos de inicio $s_j$ de cada actividad $j$, justificar una actividad $j \neq n$ a la derecha consiste en obtener un programa $S'$ tal que $s'_j$, para $i \neq j$, se busca un nuevo tiempo de inicio ($s'_j \geq s_j$) de tal forma que sea tan grande como sea posible sin aumentar el makespan y sin violar las restricciones de precedencia. La justificación a la derecha de las actividades $j$ en orden decreciente de su tiempo de terminación ($f_j = s_j + d_j$) genera un programa activo a la derecha $S^R$, que es llamado la justificación a la derecha de $S$. $S^R$ no es único, ya que depende de la(s) regla(s) de desempate usada(s). En esta investigación se utiliza como regla de desempate escoger la actividad con mayor tiempo de finalización en la lista de actividades a programar.

El anterior procedimiento asegura que la nueva solución obtenida $S^R$ tiene un makespan menor o, en el peor de los casos, igual al de la solución $S$ antes de la justificación. Similarmente, el procedimiento anterior se puede realizar en sentido contrario para justificar la solución resultante a la izquierda. Este algoritmo es aplicado a cada una de las cuatro soluciones obtenidas anteriormente.

En Valls *et al.* (2003a) se presenta una generalización de esta técnica en donde el lector puede obtener mayor información.

#### VND

El VND es una metaheurística cuya idea original fue considerar distintas estructuras de vecindarios y cambiarlas sistemáticamente para escapar de los mínimos locales. El VNS básico o *VND* (Variable Neighborhood Descend) obtiene una solución inicial, ejecuta una Búsqueda Local cuyo procedimiento consiste en reemplazar la solución actual si ha habido una mejora o modificar la estructura del vecindario en caso contrario.

VND se basa en tres hechos simples (Hansen & Mladenović, 2005):

- Un óptimo local con respecto a una estructura de vecindad no necesariamente lo es con respecto a otra.
- Un óptimo global es un óptimo local con respecto a todas las posibles estructuras de vecindad.
- En muchos problemas, los óptimos locales con respecto a una o varias estructuras de vecindad están relativamente cerca.

El VND implementado utiliza 4 vecindarios, los cuales se describen a continuacón:

***Inserción hacia adelante:*** El primer vecindario consiste en barrer cada una de las actividades en la solución e insertarla en una posición más adelante. Este vecindario puede ser generado y evaluado en orden $O(n^3)$.

***Inserción hacia atrás:*** El segundo vecindario es similar al anterior, pero inserta las actividades en una posición anterior en el programa. Este vecindario puede ser generado y evaluado en orden $O(n^3)$.

***Intercambio:*** El tercer vecindario consiste en intercambiar las posiciones de dos actividades. Este vecindario puede ser generado y evaluado en orden $O(n^3)$.

***Inserción general:*** El último vecindario consiste en hacer dos inserciones simultáneamente. Se buscan dos actividades $i$ e $j$, y cada una de ellas se inserta hacia adelante o hacia atrás en la solución. Debido al tiempo de cómputo requerido, las posiciones de inserción son limitadas a un máximo $h$ de posiciones adelante o atrás. Así, la complejidad del algoritmos es $O(n^3 \cdot h^2)$.

El VND, debido al tiempo de cómputo requerido solo es aplicado a la mejor de las 8 soluciones obtenidas con los métodos anteriores.

En la sección de resultados se presenta se realiza una evaluación de la eficacia de los procedimientos de mejora anteriores.

## 4.3. Programación lineal entera mixta

El RCPSP ha sido formulado matemáticamente mediante diferentes modelos de programación lineal entera mixta (MILP). A continuación se presentan algunos de ellos con el objetivo de realizar una comparación de sus desempeños.

### Modelo DT

El modelo DT (*discrete-time formulation*), utilizado por Pritsker *et al.* (1969) y Mingozzi *et al.* (1998), utiliza la variable de decisión binaria $\varepsilon_{it}$, la cual indica si la actividad $i \in J$ inicia en el periodo de tiempo $t \in T$ ($\varepsilon_{it} = 1$), o no ($\varepsilon_{it} = 0$).

$$\text{mín } Z = \sum_{t=es_n}^{ls_n} t \cdot \varepsilon_{nt} \tag{1}$$

$$\sum_{t=es_j}^{ls_j} \varepsilon_{jt} = 1, \quad \forall\ j \in J \tag{2}$$

$$\sum_{t=es_j}^{ls_j} t \cdot \varepsilon_{jt} - \sum_{t=es_i}^{ls_i} t \cdot \varepsilon_{it} \geq d_i, \quad \forall\ (i,j) \in H \tag{3}$$

$$\sum_{j \in J} \sum_{\tau=\sigma(t,j)}^{t} r_{jk} \cdot \varepsilon_{j\tau} \leq R_k, \quad \forall\ t \in T,\ k \in K \tag{4}$$

$$\varepsilon_{jt} \in \{0,1\}, \quad \forall\ j \in J,\ t \in \{es_j, ..., ls_j\} \tag{5}$$

En el modelo anterior, los parámetros $es_i$ y $ls_i$ indican el tiempo de inicio más cercano (early start) y más lejano (late start), respectivamente. La ecuación (1) representa la función objetivo, minimización del makespan. Las restricciones (2) implican que a cada actividad se le debe asignar exactamente un tiempo de inicio. Las restricciones (3) limitan el tiempo de inicio de una actividad $i \in J$ si ésta es sucesora de otra actividad $j \in J$, es decir, si $(i,j) \in H$. Las restricciones de recursos son expresadas por las ecuaciones (4), donde $\sigma(t,j) = \text{máx}\{0, t - d_j + 1\}$. Finalmente, las restricciones (5) limitan el dominio de las variables de decisión.

Los parámetros $es_i$ y $ls_i$ son usados también en los siguientes modelos, y pueden ser calculados usando el algoritmo CPM y una cota superior para el makespan. En la sección de resultados se compara el desempaño de este modelo utilizando dos cotas superiores diferentes.

**Modelo DDT**

El modelo DDT (*disaggregated discrete-time formulation*) fue propuesto por Christofides *et al.* (1987). Éste está basado en el modelo DT, y modifica la forma en que se formulan las restricciones de precedencia. En este modelo la ecuación (3) del modelo DT es reemplazada por la ecuación (6) presentada a continuación.

$$\sum_{\tau=t}^{ls_i} \varepsilon_{i\tau} + \sum_{\tau=es_j}^{\text{mín}\{ls_j, t+p_i-1\}} \varepsilon_{j\tau} \leq 1, \quad \forall\ (i,j) \in H,\ t \in \{es_i, ..., ls_i\} \tag{6}$$

Así, el modelo DDT puede ser formulado usando las ecuaciones (1) a (2) y (4) a (6).

**Modelo FCT**

Los modelos FCT (*flow-based continuous-time formulation*) involucran variables que indican secuencias en lugar de tiempos de inicio. Así, la variable de decisión binaria $x_{ij}$ indica si la actividad $i \in J$ es procesada antes que la actividad $j \in J$ ($x_{ij} = 1$) o no ($x_{ij} = 0$), es decir, $x_{ij} = 1$ implica que el tiempo de inicio de la actividad $j$ debe ser superior al tiempo de finalización de la actividad $i$. Por otro lado, la variable de decisión $S_i$ representa el tiempo de inicio de la actividad $i \in J$. Finalmente, la variable de decisión $f_{ijk}$ denota el número de unidades del recurso $k$ que son directamente transferidas de la actividad $i$ (al final de su procesamiento) a la actividad $j$ (al principio de su procesamiento). Se define además el parámetro $\tilde{r}_{ik}$ como: $\tilde{r}_{ik} = r_{ik}\ \forall\ i \in J \backslash \{1, n\}$ y $\tilde{r}_{1k} = \tilde{r}_{nk} = R_k$, indicando que la actividad 1 es una fuente de recursos la actividad $n$ es un sumidero. Ejemplos de este tipo de formulaciones son precentados en Balas (1970), Alvarez-Valdés & Tamarit (1993) y Artigues *et al.* (2003). El siguiente modelo está basado en los presentados por Koné *et al.* (2011) y Artigues *et al.* (2003).

$$\text{mín } Z = S_n \tag{7}$$
$$x_{ij} + x_{ji} \leq 1, \quad \forall\ i, j \in J,\ i < j \tag{8}$$
$$S_j - S_i \geq p_i - M_{ij} + M_{ij} \cdot x_{ij}, \quad \forall\ i, j \in J \tag{9}$$
$$f_{ijk} \leq R_k \cdot x_{ij}, \quad \forall\ i, j \in J,\ k \in K \tag{10}$$
$$\sum_{j \in J} f_{ijk} = \tilde{r}_{ik}, \quad \forall\ i \in J,\ k \in K \tag{11}$$
$$\sum_{i \in J} f_{ijk} = \tilde{r}_{jk}, \quad \forall\ j \in J,\ k \in K \tag{12}$$
$$x_{ij} = 1, \quad \forall\ (i, j) \in H \tag{13}$$
$$f_{ijk} \geq 0, \quad \forall\ (i, j) \in J,\ k \in K \tag{14}$$
$$S_1 = 0 \tag{15}$$
$$es_i \leq S_i \leq ls_i, \quad \forall\ i \in J \tag{16}$$
$$x_{ij} \in \{0, 1\}, \quad \forall\ i, j \in J \tag{17}$$

En el modelo FCT, la ecuación (7) representa la función objetivo. Las restricciones (8) indican que para cada par de actividades $(i, j)$, la actividad $i$ inicia primero que $j$, o la actividad $j$ inicia primero que $i$, o ninguna de las dos inicia antes que la otra. Las ecuaciones (9) implican que si la actividad $i$ inicia antes que la actividad $j$, entonces el tiempo de inicio de la actividad $j$ tiene que ser mayor que el tiempo de inicio de la actividad $i$ más su tiempo de procesamiento. $M_{ij}$ es un valor constante adecuadamente grande. En esta investigación se usa $M_{ij} = ls_n$. Las restricciones (10) a (12) modelan el flujo de recursos de una actividad a otra, mientras que las restricciones (13) representan las restricciones de precedencia. Finalmente, las ecuaciones (14) a (17) limitan el dominio de las variables de decisión.

**Modelo SEE**

Koné *et al.* (2011) propone una formulación basada en eventos denominada SEE (*start/end event-based formulation*). Dicha formulación está basada en la formulación propuesta por Zapata *et al.* (2008) para la versión multimodal del problema (*multimode resource constrained multiproject scheduling problem*). El término 'evento' corresponde a los tiempos de inicio o terminación de las actividades. En este modelo las variables de decisión binarias $y_{ie}$ y $y'_{ie}$ indican si la actividad $i \in J$ inicia y finaliza, respectivamente, en el evento $e \in E$ ($y_{ie} = 1$ y $y'_{ie} = 1$), o no ($y_{ie} = 0$ y $y'_{ie} = 0$). La variable de decisión $t_e$ expresa el tiempo en que ocurre el evento $e \in E$, y la variable de decisión $a_{ek}$ expresa la cantidad de recursos de tipo $k \in K$ requeridos después del evento $e \in E$. Nótese que un proyecto puede tener a lo sumo $n + 1$ eventos, por lo tanto el conjunto de eventos $E$ tiene $n + 1$ elementos.

La siguiente es la formulación propuesta en Koné *et al.* (2011).

$$\text{mín } Z = t_n \tag{18}$$
$$t_0 = 0 \tag{19}$$
$$t_{e+1} \geq t_e, \quad \forall\ e \in E \setminus \{n\} \tag{20}$$
$$t_f \geq t_e + p_i \cdot y_{ie} - p_i \cdot (1 - y'_{if}), \quad \forall\ e, f \in E,\ i \in J \tag{21}$$
$$\sum_{e \in E} y_{ie} = 1, \quad \forall\ i \in J \tag{22}$$
$$\sum_{e \in E} y'_{ie} = 1, \quad \forall\ i \in J \tag{23}$$
$$\sum_{e'=e}^{n} y'_{ie'} + \sum_{e'=0}^{e-1} y_{je'} \leq 1, \quad \forall\ (i, j) \in H,\ e \in E \tag{24}$$

$$a_{0k} - \sum_{i \in J} b_{ik} \cdot y_{i0} = 0, \quad \forall\ k \in K \tag{25}$$

$$a_{ek} - a_{e-1,k} + \sum_{i \in J} r_{ik} \cdot (y'_{ie} - y_{ie}) = 0, \quad \forall\ e \in E \setminus \{0\},\ k \in K \tag{26}$$

$$a_{ek} \leq R_k, \quad \forall\ e \in E,\ k \in K \tag{27}$$

$$\sum_{v=0}^{e} y'_{iv} + \sum_{v=e}^{n} y_{iv} \leq 1, \quad \forall\ i \in J,\ e \in E \tag{28}$$

$$\sum_{f \in E} f \cdot y'_{if} - \sum_{e \in E} e \cdot y_{ie} \geq 1, \quad \forall\ i \in J \tag{29}$$

$$es_i \cdot y_{ie} \leq t_e \leq ls_i \cdot y_{ie} + ls_{n+1} \cdot (1 - y_{ie}), \quad \forall\ i \in J,\ e \in E \tag{30}$$

$$(es_i + p_i) \cdot y'_{ie} \leq t_e \leq (ls_i + p_i) \cdot y'_{ie} + ls_{n+1} \cdot (1 - y'_{ie}), \quad \forall\ i \in J,\ e \in E \tag{31}$$

$$es_{n+1} \leq t_n \tag{32}$$

$$y_{ie}, y'_{ie} \in \{0, 1\}, \quad \forall\ i \in J,\ e \in E \tag{33}$$

$$t_e \geq 0, \quad \forall\ e \in E \tag{34}$$

$$a_{ek} \geq 0, \quad \forall\ e \in E,\ k \in K \tag{35}$$

La ecuación (18) reresenta la función objetivo. La ecuación ((19)) fija el tiempo de inicio (primer evento) en cero. Las restricciones (20) aseguran que el tiempo de ocurrencia de los eventos sea creciente. Las ecuaciones (21) garantizan que el tiempo entre el inicio y el fin de una actividad sea igual a su duración. Las restricciones (22) y (23) garantizan que a cada actividad sea asignado un inicio y un fin, respectivamente. Las restricciones de precedencia son representadas por las ecuaciones (24). Las restricciones (25) a (27) expresan y limitan la cantidad de recursos requeridos después de cada evento. Las ecuaciones (28) indican que el evento de finalización de una actividad debe ser posterior al de su inicio, mientras que la restricción (29) expresa que la difirencia entre los índices de dichos eventos debe ser superior a 1. Las restricciones (30) a (32) limitan los tiempos de inicio y finalización de cada evento de acuerdo a los tiempos de inicio más cercanos y de finalización más lejanos de cada actividad. Finalmente, la ecuaciones (33) a (35) definen el dominio de las variables de decisión.

**Modelo OOE**

Koné *et al.* (2011) proponen también el modelo OOE (*on/off event-based formulation*) basado en eventos. En este modelo, la variable de decisión binaria $z_{ie}$ indica si la actividad $i \in J$ inicia o continua en ejecución en el evento $e \in E$ ($z_{ie} = 1$), o no ($z_{ie} = 0$).

$$\text{mín}\ \ Z = t_n \tag{36}$$

$$\sum_{e \in E} z_{ie} \geq 1, \quad \forall\ i \in J \tag{37}$$

$$t_n \geq t_e + (z_{ie} - z_{i,e-1}) \cdot p_i, \quad \forall\ e \in E,\ i \in J \tag{38}$$

$$t_0 = 0 \tag{39}$$

$$t_{e+1} \geq t_e, \quad \forall\ e \in E \setminus \{n-1\} \tag{40}$$

$$t_f \geq t_e + ((z_{ie} - z_{i,e-1}) - (z_{if} - z_{i,f-1})) \cdot p_i, \quad \forall\ e, f \in E,\ f > e,\ i \in J \tag{41}$$

$$\sum_{e'=0}^{e-1} z_{ie'} \leq e \cdot (1 - (z_{ie} - z_{i,e-1})), \quad \forall\ e \in E \setminus \{0\} \tag{42}$$

$$\sum_{e'=e}^{n-1} z_{ie'} \leq (n - e) \cdot (1 + (z_{ie} - z_{i,e-1})), \quad \forall\ e \in E \setminus \{0\} \tag{43}$$

$$z_{ie} + \sum_{e'=0}^{e} z_{je'} \leq 1 + (1 - z_{ie}) \cdot e, \quad \forall\ e \in E,\ (i,j) \in P \tag{44}$$

$$\sum_{i=0}^{n-1} r_{ik} \cdot z_{ie} \leq R_k, \quad \forall\ e \in \psi,\ k \in K \tag{45}$$

$$es_i \cdot z_{ie} \leq t_e \leq ls_i \cdot (z_{ie} - z_{i,e-1}) + ls_n \cdot (1 - (z_{ie} - z_{i,e-1})), \quad \forall\ i \in J,\ e \in E \tag{46}$$

$$es_n \leq t_n \leq ls_n, \quad \forall\ i \in J,\ e \in E \tag{47}$$

$$z_{ie} \in \{0,1\}, \quad \forall\ i \in J,\ e \in E \tag{48}$$

$$t_e \geq 0, \quad \forall\ e \in E \tag{49}$$

La ecuación (36) representa la función objetivo, minimizar el makespan. Las restricciones (37) aseguran que cada actividad este en ejecución después de al menos un evento. Las restricciones (38) limitan el tiempo de inicio del último evento, el cuál es equivalente al makespan. La restricción (39) fija el tiempo del primer evento al instante cero. Las ecuaciones (40) aseguran que el tiempo de inicio de los diferentes eventos sea no decreciente. Las ecuaciones (41) relacionan las variables binarias ($z_{ie}$) y las continuas ($t_e$) y garantizan que los eventos correspondan al inicio o fin de una actividad. Las ecuaciones (42) y (43), llamadas de contiguidad, aseguran la ininterrupción de las actividades. Las restricciones (44) describen cada relación de precedencia entre las actividades. Las restricciones de disponibilidad de recursos están dadas por las ecuaciones (45). Las restricciones (46) y (47) garantizan que el inicio de cada actividades se encuentre entre su tiempo de inicio más cercano y el más lejano. Finalmente, las ecuaciones (48) y (49) definen el dominio de las variables de decisión.

### 4.4. Estrategias para acelerar convergencia de métodos exactos

En esta sección se describen procedimientos desarrollados con el fin de utilizar las soluciones halladas con los métodos heurísticos para aumentar el número de instancias en las que se obtiene rápidamente soluciones con los procedimientos exactos. Estos procedimientos se basan en el principio de que la información sobre soluciones heurísticas deberían ayudar a mejorar el desempeño de las formulaciones matemáticas.

Se emplearon seis estrategias diferentes para combinar soluciones heurísticas con formulaciones matemáticas. Debido a sus diferentes características, tres de ellas garantizan la obtención de una solución óptima (cuando dicha solución existe), mientras que las otras tres son consideradas como estrategias heurísticas ya que pueden eliminar la solución óptima al reducir la región factible. A continuación se describen las estrategias utilizadas:

- **Usar $Z^H$ para calcular $ls_j$.** Esta estrategia consiste en utilizar la función objetivo de la solución heurística, $Z^H$, como cota superior para calcular $ls_j$ para cada actividad $j$. Dicha estrategia garantiza que la solución obtenida es óptima.
- **Restringir la función objetivo.** Esta estrategia consiste en añadir una restricción al modelo matemático indicando que la función objetivo debe ser menor o igual a $Z^H$. Dicha estrategia también garantiza que la solución obtenida es óptima.
- **Solución inicial.** Algunos de los software (solvers) comerciales permiten iniciar el procedimiento de solución con una solución inicial. Dicho procedimiento es utilizado considerando la solución heurística como solución inicial del método exacto. Dicha estrategia también garantiza que la solución obtenida es óptima.
- **Fijar variables.** Esta estrategia consiste en utilizar el algoritmo constructivo basado en holguras para fijar algunas variables en el modelo matemático. Así, dicho algoritmo indica algunas de las variables que se deben fijar, y el modelo matemático optimiza sobre las variables de decisión restantes. Al fijar algunas variables se puede estar eliminando la solución óptima, por lo que este procedimiento se considera heurístico.

- **Agregar relaciones de precedencia.** De manera similar al procedimiento anterior, aquí se agregan restricciones de precedencia con el algoritmo constructivo basado en precedencias. Al igual que en el procedimiento anterior, la solución obtenida se considera heurística.
- **Reducción de límites para los tiempos de inicio ($es_j$ y $ls_j$).** Esta estrategia utiliza los movimientos de búsqueda local utilizados por el algoritmo VND para aproximar los menores y mayores tiempos de inicio que puede tener cada una de las actividades.

En la sección de resultados se comparan dichas estrategias utilizando una de las formulaciones matemáticas descritas anteriormente.

# 5. Experimentación computacional

En esta sección de describen los experimentos computacionales orientados a evaluar el desempeño de las diferentes metodologías expuestas en las secciones anteriores.

Los algoritmos heurísticos están desarrollados en MATLAB 2014, mientras que los modelos matemáticos son resueltos con el software IBM ILOG CPLEX Optimization Studio 12.6. Las instancias de prueba fueron tomadas de la librería PSPLIB (Kolisch & Sprecher, 1997), disponible en internet (http://www.om-db.wi.tum.de/psplib/). Las pruebas han sido ejecutadas en un computador con procesador Intel Core i5, 3.10 GHz, 4GB de memoria RAM.

A continuación, la Subsección 5.1 se comparan los cuatro métodos constructivos. En la Subsección 5.2 se discute el uso del método Justificación. El algoritmo VND es evaluado en la Subsección 5.3. La Subsección 5.4 compara el desempeño de la formulación DT usando dos cotas superiores diferentes. En la Subsección 5.5 se comparan los diferentes modelos matemáticos presentados en la Sección 4.3. Finalmente, la Subsección 5.6 evalúa las diferentes estrategias para combinar los métodos heurísticos y los métodos exactos.

## 5.1. Evaluación de métodos constructivos

A continuación se presentan los resultados obtenidos con los cuatro métodos constructivos utilizando las (480) instancias de 32 actividades y 4 tipos de recursos disponibles en la libraría PSPLIB (Kolisch & Sprecher, 1997). Es importante resaltar que, para estas instancias, todas las soluciones óptimas son conocidas.

En la Tabla 1 se resumen los resultados obtenidos. La primera columna indica el tipo de resultado a observar para cada método. Las columnas C1, C2, C3 y C4 representan los resultados obtenidos con cada uno de los cuatro métodos constructivos. La última columna representa el mejor resultado obtenido considerando la mejor solución obtenida por los cuatro métodos constructivos. Las filas Dopt $= 0\,\%$, $0\,\% <$ Dopt $\leq 5\,\%$, $5\,\% <$ Dopt $\leq 10\,\%$, $10\,\% <$ Dopt $\leq 20\,\%$ y Dopt $> 20\,\%$ indican el número de instancias con desviación con respecto a la solución óptima igual a cero, entre $0\,\%$ y $5\,\%$, entre $5\,\%$ y $10\,\%$, entre $10\,\%$ y $20\,\%$, y mayor que $20\,\%$ respectivamente. La fila promedio presenta la desvación promedio con respecto a la solución óptima. Finalmente, la columna Tiempo indica el tiempo de cómputo en segundos.

Tabla 1: Comparación métodos de constructivos

| | C1 | C2 | C3 | C4 | Cmin |
|---|---|---|---|---|---|
| Dopt $= 0\,\%$ | 195 | 199 | 153 | 180 | 253 |
| $0\,\% <$ Dopt $\leq 5\,\%$ | 47 | 36 | 56 | 43 | 58 |
| $5\,\% <$ Dopt $\leq 10\,\%$ | 76 | 49 | 60 | 57 | 73 |
| $10\,\% <$ Dopt $\leq 20\,\%$ | 110 | 98 | 104 | 106 | 76 |
| Dopt $>20\,\%$ | 52 | 98 | 107 | 94 | 20 |
| Promedio | 7,86 % | 9,84 % | 11,59 % | 10,05 % | 4,77 % |
| Tiempo (s) | 0,004 | 0,001 | 0,004 | 0,001 | 0,010 |

Se puede observar que de los cuatro métodos, el método C2 es el que más soluciones óptimas obtiene, seguido por el método C1. En promedio, las desviaciones obtenidas por el método C1 son las menores entre los cuatro métodos. El tiempo de ejecución de cada método es inferior a 0,004 segundos. A pesar de la aparente superioridad del método C1, ejecutar los cuatro métodos mejora significativamente los resultados con un tiempo de ejecución total de 0,01 segundos.

### 5.2. Evaluación de la Justificación

A continuación se presentan los resultados obtenidos al aplicar el procedmiento de Justificacón a cada uno de los los cuatro métodos constructivos utilizando las (480) instancias de 32 actividades y 4 tipos de recursos disponibles en la libraría PSPLIB (Kolisch & Sprecher, 1997).

En la Tabla 2 se resumen los resultados obtenidos, utilizando los mismos indicadores utilizados en la Tabla 1.

Tabla 2: Comparación métodos de justificación

| | J1 | J2 | J3 | J4 | Jmin |
|---|---|---|---|---|---|
| Dopt = 0 % | 260 | 229 | 260 | 236 | 308 |
| 0 % < Dopt ≤ 5 % | 67 | 63 | 75 | 68 | 88 |
| 5 % < Dopt ≤ 10 % | 66 | 67 | 67 | 67 | 52 |
| 10 % < Dopt ≤ 20 % | 72 | 83 | 63 | 73 | 32 |
| Dopt > 20 % | 15 | 38 | 15 | 36 | 0 |
| Promedio | 4,24 % | 5,97 % | 4,06 % | 5,52 % | 2,24 % |
| Tiempo (s) | 0,006 | 0,004 | 0,006 | 0,003 | 0,012 |

Es importante notar que, aunque el método constructivo C3 es el que menos soluciones óptimas obtiene y el que presenta la mayor desviación promedio respecto a las soluciones óptimas, al aplicar el método de Justificación (J3) se obtienen mejores soluciones que con los otros métodos constructivos y sus combinaciones con la Justificación. Además, todos los métodos son ejecutados en menos de 0,006 segundos. Es importante mencionar que dichos tiempos incluyen el tiempo necesario para obtener las soluciones iniciales con los métodos constructivos.

Al combinar todos los métodos constructivos y aplicar a cada uno el método de Justificación se obtiene una desviación promedio de 2,24 % con respecto a las soluciones óptimas en un tiempo de ejecución promedio de 0,012 segundos. Adicionalmente, todas las soluciones obtenidas se encuentran a una desviación con respecto a la solución óptima inferior o igual al 20 %.

Al utilizar el método de Justificación sobre los cuatro métodos constructivos, la calidad de la solución mejora en un 53 % mientras que el tiempo de ejecución solo aumenta el 20 %.

### 5.3. Evaluación del VND

En esta sección se presentan los resultados del algoritmo VND utilizando como solución inicial la mejor solución obtenida con los 8 métodos anteriores: métodos constructivos y justificación.

El algoritmo VND propuesto utiliza 4 estructuras de vecindarios: a.) Inserción hacia adelante, b.) Inserción hacia atrás, c.) Intercambio, y d.) Inserción General. El orden de de ejecución de los vecindarios se ha fijado de acuerdo a su complejidad computacional (teórica) y a su eficiencia en la práctica.

En la Tabla 3 se resumen los resultados obtenidos. La columna SI representa los resultados con la solución inicial, es decir, los mejores resultados en la Tabla 2. La columna V1 indica los resultados obtenidos

al aplicar el vecindario 1, la columan V2 inidica los resultados obtenidos al aplicar los vecindarios 1 y 2, y la columan V3 inidica los resultados obtenidos al aplicar los vecindarios 1, 2 y 3. Las siguientes columnas indican los resultados obtenidos a aplicar los resultados 1, 2, 3 y 4, donde el vecindario 4 tiene como límite $h$ posiciones entre el punto de origen y el punto de inserción de las actividades. En los tests, el parámetro $h$ varía entre los siguientes valores: $h \in \{3, 5, 10, n\}$. Nótese que, de acuerdo con la estrategia del VND, la calidad de la solución nunca empeora al agregar nuevos vecindarios o al aumentar el valor de $h$.

Tabla 3: Evaluación del desempeño del algoritmo VND

| | SI | V1 | V2 | V3 | V4h3 | V4h5 | V4h10 | V4h$n$ |
|---|---|---|---|---|---|---|---|---|
| Dopt(0 %) | 308 | 319 | 330 | 331 | 348 | 358 | 367 | 373 |
| Dopt($\leq$5 %) | 88 | 106 | 111 | 111 | 106 | 102 | 95 | 92 |
| Dopt($\leq$10 %) | 52 | 42 | 32 | 32 | 24 | 18 | 16 | 14 |
| Dopt($\leq$20 %) | 32 | 13 | 7 | 6 | 2 | 2 | 2 | 1 |
| Dopt(>20 %) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| Promedio | 2,24 % | 1,55 % | 1,30 % | 1,26 % | 0,96 % | 0,85 % | 0,76 % | 0,68 % |
| Tiempo (s) | 0,012 | 0,031 | 0,042 | 0,046 | 0,387 | 0,747 | 1,292 | 1,629 |

De manera general, cada vez que se aplica un nuevo vecindario, los resultados en términos de número de soluciones óptimas y desviación promedio con respecto a la solución óptima son mejoradas. De la misma manera, cada vez que se agrega un vecindario, el tiempo promedio aumenta de manera significativa.

Al usar los primeros tres vecindarios, el tiempo de ejecución aumenta 158 %, 35 % y 9 %, respectivamente. Cuando se usa el vecindario 4 el tiempo de ejecución presenta un aumento superior a 740 %, incluso con $h = 3$.

### 5.4. Comparación de modelos matemáticos usando diferentes cotas superiores

En esta sección se usa el modelo DT para comparar su desempeño cuando se emplea un método heurístico para obtener información preliminar.

Así, se pueden obtener valores apropiados para los tiempos de inicio más cercanos de una actividad $i$, $es_i$, teniendo en cuenta las relaciones de precedencia utilizando la Ecuación (50).

$$ES_j = \max_{i \in J}\{(ES_i + p_i) \cdot P_{ij}\}, \qquad \forall\ j \in J \tag{50}$$

Para calcular los tiempo de inicio más lejanos hace falta conocer una cota superior $UB$, de manera que el tiempo de inicio más lejano de la última actividad sea igual a dicho valor ($LS_n = UB$). Así, conocer una buena cota superior podría ayudar a acotar dichos tiempos de inicio más lejanos, los cuales se calculan utilizando la Ecuación (51).

$$LS_i = \min_{j \in J}\{LS_j + UB \cdot (1 - P_{ij}))\} - p_i, \qquad \forall\ i \in J \tag{51}$$

A continuación se presentan los resultados para dos valores diferentes de dichas cotas superiores. En el primer caso se calcula dicha cota como la suma de las duraciones de todas las actividades (Ecuación 52), mientas que en el segundo caso se utilizan las soluciones halladas con los métodos heurísticos como cota superior para la función objetivo ($UB_2 = Z^H$).

$$UB_1 = \sum_{i \in J} p_i \tag{52}$$

La Tabla 4 resume los resultados obtenidos. En la columna DT1 se presentan los resultados con la primera cota superior, mientras que la columna DT2 presenta los resultados con la segunda. Las dos primeras filas

indican el número de instancias en las cuales el solucionador CPLEX garantiza la solución óptima y aquellas en las que no garantiza dicha solución. Las filas 3 a 7 indican el número de soluciones con desviación porcentual con respecto a la solución óptima igual a cero, menor o igual a 5 %, 10 %, 20 % y superior al 20 %. La fila 8 (Sin solución) indica el número de instancias en la que no se obtuvo una solución factible con el modelo matemático. La fila 9 (Promedio) indica la desviación promedio con respecto a la solución óptima. Finalmente, la fila 10 (Tiempo) indica en tiempo promedio requerido para hallar dichas soluciones.

Tabla 4: Comparación del desempeño del modelo DT con diferentes cotas superiores

| | DT1 | DT2 |
|---|---|---|
| Óptimos | 426 | 436 |
| No Óptimos | 54 | 44 |
| Dopt(0 %) | 430 | 443 |
| Dopt($\leq$5 %) | 15 | 32 |
| Dopt($\leq$10 %) | 26 | 5 |
| Dopt($\leq$20 %) | 8 | 0 |
| Dopt(>20 %) | 0 | 0 |
| Sin solución | 1 | 24 |
| Promedio | 0,69 % | 0,24 % |
| Tiempo (s) | 920,06 | 785,16 |

Los resultados anteriores indican que el uso de las soluciones obtenidas con los algoritmos heurísticos para calcular los tiempos de inicio cercanos y lejanos (Modelo DT2) permiten obtener 10 soluciones óptimas más con garantía de optimalidad.

Por otro lado, se obtienen 13 soluciones más con desviación igual a 0 %. Además, todas las instancias tienen una desviación inferior al 10 %. El tiempo de cómputo promedio también se reduce en 134,9 segundos.

Aunque el modelo DT2 obtiene menos soluciones factibles que el modelo DT1, el uso del algoritmo heurístico en la segunda estrategia permite tener como respuesta una solución factible.

Los valores obtenidos de $es_j$ y $ls_j$ en esta subsección son utilizados en las siguientes con todos los modelos matemáticos.

### 5.5. Comparación de todos los modelos matemáticos

En esta sección se comparan los 5 modelos matemáticos presentados en la Sección 4.3. En todos los modelos matemáticos se usa la información de los algoritmos heurísticos para calcular los tiempos de inicio cercanos y lejanos. La Tabla 5 resumen los resultados de los 5 modelos utilizando los mismos indicadores que en la Tabla 4.

Los resultados en esta tabla indican que los modelos DT y DTT tienen un desempeño superior a los otros tres, tanto en número de soluciones óptima obtenidas, desviación promedio con respecto a la solución óptima y tiempo de cómputo. Para la combinación de los métodos exactos con las soluciones heurísticas en las siguientes secciones se utiliza el modelo DTT, ya que obtiene más soluciones factibles que el modelo DT y es el que registra el menor tiempo de cómputo.

### 5.6. Combinación de heurísticos con modelos matemáticos

En esta sección se evalúan cuatro estrategias para combinar la información de las soluciones heurísticas con los modelos matemáticos. Dichas estrategias se dividen en dos tipos, aquellas que garantizan la obtención de soluciones óptimas y aquellas que sacrifican la optimalidad de la solución.

Tabla 5: Comparación del desempeño de todos los modelos matemáticos

| | DT | DTT | FCT | SEE | OOE |
|---|---|---|---|---|---|
| Óptimos | 436 | 436 | 373 | 203 | 211 |
| No Óptimos | 44 | 44 | 107 | 277 | 269 |
| Dopt(0 %) | 443 | 445 | 421 | 353 | 367 |
| Dopt(≤5 %) | 32 | 28 | 43 | 105 | 96 |
| Dopt(≤10 %) | 5 | 7 | 15 | 20 | 17 |
| Dopt(≤20 %) | 0 | 0 | 1 | 2 | 0 |
| Dopt(>20 %) | 0 | 0 | 0 | 0 | 0 |
| Sin solución | 24 | 23 | 64 | 222 | 144 |
| Promedio | 0,24 % | 0,26 % | 0,46 % | 0,88 % | 0,74 % |
| Tiempo (s) | 785,16 | 745,81 | 1 704,64 | 4 313,30 | 4 084,30 |

La Tabla 6 presenta los resultados de las estrategias óptimas. Dicha tabla utiliza los mismos criterios que la Tabla 5. La columna DTT corresponde con los resultados obtenidos con el modelo matemático presentados en la Tabla 5. La columna HE1 (Híbrido Exacto 1) presenta los resultados para la estrategia consistente en restringir el valor de la función objetivo de manera que sea inferior o igual al valor de la función objetivo obtenido por los algoritmos heurísticos. La columna HE2 (Híbrido Exacto 2) indica los resultados cuando se introduce una solución inicial al modelo matemático.

Tabla 6: Comparación del desempeño de métodos híbridos exactos

| | DTT | HE1 | HE2 |
|---|---|---|---|
| Óptimos | 436 | 463 | 464 |
| No Óptimos | 44 | 17 | 16 |
| Dopt(0 %) | 445 | 466 | 467 |
| Dopt(≤5 %) | 28 | 11 | 10 |
| Dopt(≤10 %) | 7 | 3 | 2 |
| Dopt(≤20 %) | 0 | 0 | 1 |
| Dopt(>20 %) | 0 | 0 | 0 |
| Sin solución | 23 | 0 | 0 |
| Promedio | 0,26 % | 0,10 % | 0,11 % |
| Tiempo (s) | 745,81 | 434,32 | 427,00 |

Los resultados en esta tabla muestran que ambos métodos mejoran significativamente los resultados obtenidos por el modelo DTT. Sin embargo no se encuentran diferencias significativas entre ambos métodos híbridos.

La Tabla 7 presenta los resultados para los métodos híbridos heurísticos. En dicha tabla los resultados HH1 corresponden al método en el cual se fijan los tiempos de inicio de algunas variables para disminuir el tamaño de la región factible; entre paréntesis se indica el número de variables fijas en cada caso. Aquí se usa el método C1 para fijar la mitad de las variables de decisión y el método C3 (con el grafo invertido) para fijar la otra mitad de las variables. Similarmente, en el método HH2 se añade la mitad de las relaciones de precedencia ficticias con el método C2 y la otra mitad con el método C4. El método HH3 limita el intervalo en que puede variar el tiempo de inicio de una actividad a los intervalos hallados durante el procedimiento VND.

Los resultados muestran que aunque los tiempos de cómputo se disminuyen considerablemente, la calidad es inferior (desviaciones mayores) a la obtenida con el procedimiento VND. Como era de esperarse, aumentar el número de variables fijas o el número de relaciones de precedencia añadidas impacta positivamente el tiempo de cómputo, pero disminuyendo la calidad promedio de las soluciones.

Tabla 7: Comparación del desempeño de métodos híbridos heurísticos

| | DTT | HH1(6) | HH1(10) | HH2(6) | HH2(10) | HH3 |
|---|---|---|---|---|---|---|
| Dopt(0 %) | 445 | 404 | 371 | 375 | 361 | 363 |
| Dopt(≤5 %) | 28 | 68 | 90 | 94 | 102 | 101 |
| Dopt(≤10 %) | 7 | 7 | 17 | 10 | 16 | 14 |
| Dopt(≤20 %) | 0 | 1 | 2 | 1 | 1 | 2 |
| Dopt(>20 %) | 0 | 0 | 0 | 0 | 0 | 0 |
| Sin solución | 23 | 0 | 0 | 0 | 0 | 0 |
| Promedio | 0,26 % | 0,45 % | 0,74 % | 0,62 % | 0,76 % | 0,76 % |
| Tiempo promedio (s) | 745,81 | 176,46 | 29,74 | 323,65 | 284,81 | 16,80 |
| Tiempo máximo (s) | 7 200,00 | 7 200,00 | 2 244,24 | 7 200,00 | 801,04 | 7 200,00 |

A pesar de que en algunos casos el tiempo de cómputo promedio es relativamente bajo (ver por ejemplo HH3 y HH1(10)), el tiempo de ejecución máximo continúa siendo el límite de 2 horas impuesto al modelo matemático. Lo anterior es una muestra de que los métodos resultantes no son robustos.

## 6. Conclusiones

En este artículo se presentan técnicas de solución, tanto exactas como heurísticas, para resolver el problema de programación de proyectos con recursos limitados (RCPSP). En cuanto a los métodos exactos, se comprar cinco formulaciones diferentes, lo cual permite identificar el modelo DTT como el más eficiente de los cinco. El método heurístico utilizado combina estrategias constructivas, justificación y VND, el cual permite obtener buenas soluciones en tiempos de cómputos muy bajos.

El aporte principal de este trabajo consiste en combinar las estrategias anteriores para mejorar algunas de las falencias de cada una: reducir el tiempo de cómputo de los métodos exactos y mejorar la calidad de las soluciones halladas por lo métodos heurísticos. Dichas estrategias son clasificadas entre estrategias exactas y heurísticas. Las estrategias exactas resultados exitosas en cuanto permiten hallar mejores soluciones en promedio, con tiempos de cómputo también inferiores. Sin embargo, las estrategias heurísticas, si bien presentan algunas buenas propiedades, no mejorar las soluciones halladas por el métodos heurístico inicial.

Como trabajo futuro, se propone aplicar estrategia similares a otros problemas combinatorios como Flow Shop Scheduling y Job Shop Scheduling. Otra fuente de investigación consiste en clasificar las diferentes instancias de acuerdo a su complejidad y evaluar la efectividad de diferentes estrategias de acuerdo a dicha clasificación.

## Referencias


Abbasi, Babak, Shadrokh, Shahram, & Arkat, Jamal. 2006. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation*, **180**(1), 146–152.

Alcaraz, J., & Maroto, C. 2001. A Robust Genetic Algorithm for Resource Allocation in Project Scheduling. *Annals of Operations Research*, **102**(1), 83–109.

Alvarez-Valdés, Ramón, & Tamarit , José Manuel. 1993. The project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Operational Research*, **67**(2), 204–220.

Artigues, Christian, Michelon, Philippe, & Reusser, Stéphane. 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, **149**(2), 249–267.

Artigues, Christian, Brucker, Peter, Knust, Sigrid, Koné, Oumar, Lopez, Pierre, & Mongeau, Marcel. 2013. A note on "event-based MILP models for resource-constrained project scheduling problems". *Computers & Operations Research*, **40**(4), 1060–1063.

Baar, Tonius, Brucker, Peter, & Knust, Sigrid. 1999. *Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project Scheduling Problem.* Boston, MA: Springer US. Pages 1–18.

Balas, E. 1970. *Applications of mathematical Prograprogr techniques.* American Elsevier. Chap. Project scheduling with resource constraints, pages 187–200.

Baradaran, Siamak, Ghomi, S.M.T. Fatemi, Mobini, Mahdi, & Hashemin, S.S. 2010. A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks. *Advances in Engineering Software*, **41**(7–8), 966–975. Advances in Structural Optimization.

Bautista, Joaquín, & Pereira, Jordi. 2002. *Ant Colonies for the RCPS Problem.* Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 257–268.

Blazewicz, J., Lenstra, J.K., & Kan, A.H.G.Rinnooy. 1983. Scheduling subject to resource constraints classification and complexity. *Discrete Applied Mathematics*, **5**(1), 11–24.

Boctor, F. F. 1996. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, **34**(8), 2335–2351.

Bouleimen, K., & Lecocq, H. 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), 268–281. Sequencing and Scheduling.

Brucker, Peter, Knust, Sigrid, Schoo, Arno, & Thiele, Olaf. 1998. A branch and bound algorithm for the resource-constrained project scheduling problem1. *European Journal of Operational Research*, **107**(2), 272–288.

Chen, Ruey-Maw. 2011. Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. *Expert Systems with Applications*, **38**(6), 7102–7111.

Cho, J.-H., & Kim, Y.-D. 1997. A Simulated Annealing Algorithm for Resource Constrained Project Scheduling Problems. *The Journal of the Operational Research Society*, **48**(7), 736–744.

Christofides, Nicos, Alvarez-Valdes, R., & Tamarit, J.M. 1987. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, **29**(3), 262–273.

Colak, Selcuk, Agarwal, Anurag, & Erenguc, Selcuk S. 2006. *Resource Constrained Project Scheduling: a Hybrid Neural Approach.* Boston, MA: Springer US. Pages 297–318.

Crawford, Broderick, Soto, Ricardo, Johnson, Franklin, Norero, Enrique, & Olguín, Eduardo. 2015. *An Artificial Bee Colony Algorithm for the Resource Contrained Project Scheduling Problem.* Cham: Springer International Publishing. Pages 582–586.

Damay, Jean, Quilliot, Alain, & Sanlaville, Eric. 2007. Linear programming based algorithms for preemptive and non-preemptive RCPSP. *European Journal of Operational Research*, **182**(3), 1012–1022.

De Magalhães, Jorge José, Gonçalves, José Fernando, & Resende, Maurício G.C. 2005. *A Random Key Based Genetic Algorithm for the Resource Constrained Project Scheduling Problem.* Tech. rept. AT&T Labs Research.

Debels, Dieter, & Vanhoucke, Mario. 2005. *A Bi-population Based Genetic Algorithm for the Resource-Constrained Project Scheduling Problem.* Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 378–387.

Debels, Dieter, & Vanhoucke, Mario. 2007. A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. *Operations Research*, **55**(3), 457–469.

Debels, Dieter, Reyck, Bert De, Leus, Roel, & Vanhoucke, Mario. 2006. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, **169**(2), 638–653. Feature Cluster on Scatter Search Methods for Optimization.

Demeulemeester, E., & Herroelen, W. 2002. *Temporal Analysis: The Basic Deterministic Case*. Boston, MA: Springer US. Pages 95–129.

Demeulemeester, Erik, & Herroelen, Willy. 1992. A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. *Management Science*, **38**(12), 1803–1818.

Demeulemeester, Erik L., & Herroelen, Willy S. 1997. New Benchmark Results for the Resource-Constrained Project Scheduling Problem. *Management Science*, **43**(11), 1485–1492.

Fisher, Marshall L. 1973a. Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part I. *Operations Research*, **21**(5), 1114–1127.

Fisher, Marshall L. 1973b. *Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part II*. Springer Berlin Heidelberg. Pages 294–318.

Fleszar, Krzysztof, & Hindi, Khalil S. 2004. Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, **155**(2), 402–413. Financial Risk in Open Economies.

Hansen, Pierre, & Mladenović, Nenad. 2005. *Variable Neighborhood Search*. Boston, MA: Springer US. Pages 211–238.

Hansen, Pierre, Mladenović, Nenad, Brimberg, Jack, & Pérez, José A. Moreno. 2010. *Variable Neighborhood Search*. Boston, MA: Springer US. Pages 61–86.

Hartmann, Sönke. 1998. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, **45**(7), 733–750.

Hartmann, Sönke. 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*, **49**(5), 433–448.

Hartmann, Sönke, & Briskorn, Dirk. 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, **207**(1), 1–14.

He, Jieguang, Chen, Xindu, & Chen, Xin. 2016. A filter-and-fan approach with adaptive neighborhood switching for resource-constrained project scheduling. *Computers & Operations Research*, **71**, 71–81.

Kim, Kwan Woo, Gen, Mitsuo, & Yamazaki, Genji. 2003. Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling. *Applied Soft Computing*, **2**(3), 174–188. Soft Computing in Manufacturing Enterprise Systems.

Kochetov, Y. A., & Stolyar, A. A. 2003. Evolutionary Local Search with Variable Neighborhood for the Resource Constrained Project Scheduling Problem. *In: Proceedings of the 3rd International Workshop of Computer Science and Information Technologies*.

Kolisch, Rainer, & Hartmann, Sönke. 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, **174**(1), 23–37.

Kolisch, Rainer, & Sprecher, Arno. 1997. PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, **96**(1), 205–216.

Koné, Oumar, Artigues, Christian, Lopez, Pierre, & Mongeau, Marcel. 2011. Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, **38**(1), 3–13.

Kopanos, Georgios M., Kyriakidis, Thomas S., & Georgiadis, Michael C. 2014. New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems. *Computers & Chemical Engineering*, **68**, 96–106.

Koulamas, C, Antony, SR, & Jaen, R. 1994. A survey of simulated annealing applications to operations research problems. *Omega*, **22**(1), 41–56.

Koulinas, Georgios, Kotsikas, Lazaros, & Anagnostopoulos, Konstantinos. 2014. A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences*, **277**, 680–693.

Lancaster, John, & Ozbayrak, Mustafa. 2007. Evolutionary algorithms applied to project scheduling problems – a survey of the state-of-the-art. *International Journal of Production Research*, **45**(2), 425–450.

Lee, Jae-Kwan, & Kim, Yeong-Dae. 1996. Search Heuristics for Resource Constrained Project Scheduling. *The Journal of the Operational Research Society*, **47**(5), 678–689.

Leon, V. Jorge, & Balakrishnan, Ramamoorthy. 1995. Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *Operations-Research-Spektrum*, **17**(2), 173–182.

Li, Xiang, Kang, Lishan, & Tan, Wei. 2007. *Optimized Research of Resource Constrained Project Scheduling Problem Based on Genetic Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 177–186.

Merkle, D., Middendorf, M., & Schmeck, H. 2002. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, **6**(4), 333–346.

Mingozzi, Aristide, Maniezzo, Vittorio, Ricciardelli, Salvatore, & Bianco, Lucio. 1998. An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation. *Management Science*, **44**(5), 714–729.

Moreno, Luis Fernando, Díaz, Francisco Javier, na, Gloria Elena Pe & Rivera, Juan Carlos. 2007. Análisis comparativo entre dos algoritmos heurísticos para resolver el problema de planeación de tareas con restricción de recursos (RCPSP). *Dyna*, **74**(151), 171–183.

Morillo, Daniel, Moreno, Luis, & Díaz, Javier. 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), 247–271.

Morillo, Daniel, Moreno, Luis, & Díaz, Javier. 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), 203–227.

Muller, Laurent Flindt. 2009. An Adaptive Large Neighborhood Search Algorithm for the Resource-constrained Project Scheduling Problem. *In: MIC 2009: The VIII Metaheuristics International Conference*.

Naphade, Kedar S., David Wu, S., & Storer, Robert H. 1997. Problem space search algorithms for resource-constrained project scheduling. *Annals of Operations Research*, **70**(0), 307–326.

Nonobe, Koji, & Ibaraki, Toshihide. 2002. *Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem*. Boston, MA: Springer US. Pages 557–588.

Ozdamar, L. 1999. A Genetic Algorithm Approach to a General Category Project Scheduling Problem. *Trans. Sys. Man Cyber Part C*, **29**(1), 44–59.

Palacio, Juan D., & Larrea, Olga L. 2017. A lexicographic approach to the robust resource-constrained project scheduling problem. *International Transactions in Operational Research*, **24**(1-2), 143–157.

Palpant, Mireille, Artigues, Christian, & Michelon, Philippe. 2004. LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search. *Annals of Operations Research*, **131**(1), 237–257.

Patterson, James H., & Huber, Walter D. 1974. A Horizon-Varying, Zero-One Approach to Project Scheduling. *Management Science*, **20**(6), 990–998.

Patterson, James H., & Roth, Glenn W. 1976. Scheduling a Project Under Multiple Resource Constraints: A Zero-One Programming Approach. *A I I E Transactions*, **8**(4), 449–455.

Pesek, Igor, Schaerf, Andrea, & Žerovnik, Janez. 2007. *Hybrid Local Search Techniques for the Resource-Constrained Project Scheduling Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 57–68.

Pritsker, A. Alan B., Watters, Lawrence J., & Wolfe, Philip M. 1969. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science*, **16**(1), 93–108.

Proon, Sepehr, & Jin, Mingzhou. 2011. A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. *Naval Research Logistics (NRL)*, **58**(2), 73–82.

Rivera, Juan Carlos, & Celín, Ana Josefina. 2010. Algoritmo heurístico híbrido con múltiples vecindarios y recocido simulado para resolver el RCPSP. *Revista Facultad de Ingeniería Universidad de Antioquia*, 12, 255–267.

Rivera, Juan Carlos, Moreno, Luis Fernando, Díaz, Javier, & Peña, Gloria Elena. 2006. Un heurístico para planeación de proyectos con restricción de recursos. *Revista Colombiana de Tecnologías de Avanzada*, **1**(6), 120–125.

Rivera, Juan Carlos, Moreno, Luis Fernando, Díaz, Francisco Javier, & na, Gloria Elena Pe2013. A hybrid heuristic algorithm for solving the resource constrained project scheduling problem (RCPSP). *Revista EIA*, **10**(20), 87–100.

Sebt, MH, & Alipouri, Y. 2013. Solving resource-constrained project scheduling problem with evolutionary programming. *The Journal of the Operational Research Society*, **64**(9), 1327–1335.

Simpson, Wendell P., & Patterson, James H. 1996. A multiple-tree search procedure for the resource-constrained project scheduling problem. *European Journal of Operational Research*, **89**(3), 525–542.

Sprecher, Arno. 2000. Scheduling Resource-Constrained Projects Competitively at Modest Memory Requirements. *Management Science*, **46**(5), 710–723.

Thomas, Paul R., & Salhi, Said. 1998. A Tabu Search Approach for the Resource Constrained Project Scheduling Problem. *Journal of Heuristics*, **4**(2), 123–139.

Tseng, Lin-Yu, & Chen, Shih-Chieh. 2006. A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research*, **175**(2), 707–721.

Valls, V., Ballestín, F., & Quintanilla, S. 2003a. Generalizaciones de la técnica de la Justificación. *In: 27 Congreso Nacional de Estadística e Investigación Operativa*.

Valls, Vicente, Quintanilla, Sacramento, & Ballestín, Francisco. 2003b. Resource-constrained project scheduling: A critical activity reordering heuristic. *European Journal of Operational Research*, **149**(2), 282–301. Sequencing and Scheduling.

Valls, Vicente, Ballestín, Francisco, & Quintanilla, Sacramento. 2004. A Population-Based Approach to the Resource-Constrained Project Scheduling Problem. *Annals of Operations Research*, **131**(1), 305–324.

Valls, Vicente, Ballestín, Francisco, & Quintanilla, Sacramento. 2005. Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, **165**(2), 375–386. Project Management and Scheduling.

Valls, Vicente, Ballestín, Francisco, & Quintanilla, Sacramento. 2008. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, **185**(2), 495–508.

Wang, Hong, Lin, Dan, & Li, Minqiang. 2005. *A Genetic Algorithm for Solving Resource-Constrained Project Scheduling Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 185–193.

Wang, Ling, & Fang, Chen. 2012. A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem. *Expert Systems with Applications*, **39**(3), 2451–2460.

Wiest, Jerome D. 1964. Some Properties of Schedules for Large Projects with Limited Resources. *Operations Research*, **12**(3), 395–418.

Willis, R. J., & Hastings, N. A. J. 1976. Project Scheduling with Resource Constraints Using Branch and Bound Methods. *Operational Research Quarterly (1970-1977)*, **27**(2), 341–349.

Wu, Changzhi, Wang, Xiangyu, & Lin, Jiang. 2014. *Optimizations in Project Scheduling: A State-of-Art Survey*. Dordrecht: Springer Netherlands. Pages 161–177.

Yan, Liang, Jinsong, Bao, Xiaofeng, Hu, & Ye, Jin. 2009. A heuristic project scheduling approach for quick response to maritime disaster rescue. *International Journal of Project Management*, **27**(6), 620–628.

Zapata, Juan Camilo, Hodge, Bri Mathias, & Reklaitis, Gintaras V. 2008. The multimode resource constrained multiproject scheduling problem: Alternative formulations. *AIChE Journal*, **54**(8), 2101–2119.