Algoritmo Memético Para Resolver el Problema de Secuenciación con Recursos Limitados Modo Múltiple, MRCPSP.

Por

Luis Fernando Machado Dominguez. Mat.

Trabajo presentado como requisito parcial para
optar al título de Maestría en Matemáticas

Director
Agustín Barrios S., Dr.

Universidad del Norte

División de Ciencias Básicas

Departamento de Matemáticas y Estadística

Julio de 2014

Agradecimientos

A Dios por brindarme salud, paciencia y la sabiduría necesaria para realizar esté trabajo.

A mi madre que siempre me ha apoyado incondicionalmente, que con su amor y consejos me han llevado a mejorar cada día más.

A mi novia que siempre estuvo a mi lado compartiendo alegrías y tristezas durante todo este proceso.

Al grupo de docentes de Matemáticas de las Universidades de Córdoba y del Norte, en especial a mi director de tesis, Agustín Barrios Sarmiento, por todas sus contribuciones, sus consejos, su paciencia, motivación y supervisión durante el tiempo que dediqué a la realización de este trabajo.

A mi madre María Eugenia y
a Lissette.

Índice general

Índice de figuras

Índice de cuadros

Resumen

La secuenciación de proyectos es un proceso de toma de decisiones que se utiliza de forma regular en la industria. Describe la asignación de recursos a las actividades durante períodos de tiempo determinados y su principal propósito es optimizar uno o más objetivos, entre ellos están minimizar la duración de un proyecto, minimizar el coste del proyecto o maximizar la calidad del proyecto. La secuenciación de proyectos cubre una amplia gama de problemas, entre ellos se encuentra el Problema de Secuenciación de Proyectos con Recursos Limitados Modo Múltiple (MRCPSP). El MRCPSP es un modelo de secuenciación muy general, que contiene problemas con la característica que las actividades se pueden realizar de distintas maneras, cambiando factores como son: la duración y el consumo de recursos.

En los últimos años ha aumentado notablemente la cantidad de trabajos de investigación relacionados con el MRCPSP, los cuales abordan métodos de solución exactos y aproximados. Entre los métodos de solución aproximados se encuentran las metaheurísticas: algoritmos genéticos (GA), búsqueda tabu (TS), búsqueda de vecindarios variables (VNS), algoritmos meméticos (MA), etc. Esté tipo de procedimientos se utiliza por la necesidad de encontrar soluciones satisfactorias u óptimas, puesto que los procedimientos exactos para este tipo de problemas no son tan eficientes cuando el espacio de búsqueda es muy grande.

El objetivo de esté documento es desarrollar e implementar en C++ un algoritmo memético para resolver el MRCPSP. Este algoritmo utiliza componentes de los algoritmos genéticos y búsqueda de vecindarios variables. Se implementa una adaptación del operador de cruce uniforme para el MRCPSP y una búsqueda local VNS de manera que genere mejores soluciones. Se utiliza una función de evaluación de los agentes que guía adecuadamente la evolución del algoritmo. La propuesta realizada para resolver el MRCPSP se ha implementado y sus resultados se han evaluado mediante la solución de las instancias de la librería estándar PSPLIB [34].

Capítulo 1
Introducción y revisión bibliográfica

1.1. Introducción

El campo de la secuenciación de proyectos ha tenido un desarrollo notable en las últimas décadas, esto ha sido propiciado por el entorno competitivo actual para ofrecer productos de calidad a tiempo y dentro del presupuesto. Además, juega un papel importante en la toma de decisiones dentro de los campos de la construcción, transporte, distribución y comunicación.

En este capítulo se pretende ofrecer una visión general de los conceptos de secuenciación de proyectos. Se proporciona un concepto de proyecto, sus atributos y el ciclo de vida. La gestión del proyecto abarca funciones administrativas básicas de planificación, programación y control. Además, se aborda un problema de secuenciación llamado *problema de secuenciación de proyectos con recursos limitados modo multiple (MRCPSP).*

Los procedimientos para solucionar el MRCPSP están basados en técnicas matemáticas y métodos heurísticos los cuales asignan limitaciones en la disponibilidad de recursos para las distintas actividades a realizar. Esta asignación tiene que ser hecha de tal manera que se optimicen y se logren los objetivos del proyecto.

El MRCPSP cuenta con la característica de que las actividades pueden ser ejecutadas de varias formas, en la cual pueden cambiar parámetros como su duración, relaciones temporales con otras actividades y los recursos requeridos. Esto hace que la secuenciación de las actividades sea un problema muy complejo. El MRCPSP es una extensión del RCPSP (problema de secuenciación de proyectos con recursos limitados) y tienen por objetivo la selección de una combinación entre *tiempo/recursos* de manera que se minimice la duración del proyecto al completar todas las actividades y satisfaga todas las restricciones de recursos.

1.1.1. Objetivos

La secuenciación de proyectos con recursos limitados modo multiple, es un problema muy complejo. Se necesitan algoritmos eficientes desde el punto de vista computacional para resolver problemas específicos en este campo. Por ello, se plantean los siguientes objetivos.

1.1.1.1. Objetivo general

Diseñar e implementar un nuevo método metaheurístico eficaz, para resolver el problema de secuenciación de proyectos con recursos limitados modo multiple.

1.1.1.2. Objetivos específicos

- Adquirir una perspectiva del estado del arte en este campo científico, esto es, cuáles son los trabajos teórico-prácticos más relevantes.
- Proponer un nuevo operador de cruce de elementos de la población.
- Introducir técnicas que puedan ser empleadas en otros algoritmos para el mismo problema u otros similares.

Este trabajo está organizado de la siguiente forma:

- En lo que resta del **primer capítulo** se realizará una revisión bibliografica, definiendo las componentes de un proyecto, las etapas y sus objetivos. Se realiza una descripción general de los problemas de secuenciación, los tipos de recursos, las actividades, relaciones de precedencia, funciones de evaluación, etc.

- En el **segundo capítulo** se hace una introducción al problema de secuenciación de proyectos con recursos limitados modo múltiple MRCPSP, el cual es el objeto de estudio de este trabajo. Se realiza un estado del arte de los métodos mas relevantes en la literatura para resolver el MRCPSP.

- En el **tercer capítulo** se propone un algoritmo memético para resolver el MRCPSP, detallando cada una de sus componentes y presentando sus resultados computacionales.

- En el **cuarto capítulo**, se destacan los principales aportes de este trabajo y futuras lineas de investigación.

1.2. Secuenciación de proyectos

En la literatura encontramos diversos conceptos de lo que se debe entender por proyecto, pero según Demeulemeester [15] la siguiente definición, ISO 8402 (1990), se ha ganado la aceptación de una amplia gama de usuarios.

Definición 1.1 (Proyecto). *Es un proceso único, que consta de un conjunto de actividades controladas y coordinadas con las fechas de comienzo y final, emprendiendo para alcanzar un objetivo de acuerdo con requerimientos específicos, incluyendo restricciones de tiempo y recursos.*

Ejemplo 1.2. Los siguientes son algunos proyectos [49]:

- Instalación de sistemas: considere la posibilidad de la adquisición, instalación y pruebas de un sistema informático de gran tamaño. El proyecto consiste en una serie de tareas diferentes, incluyendo la evaluación y la selección de hardware, desarrollo de software, capacitación de personal, pruebas de sistema, sistema de depuración, etc.

- Planta de fabricación de semiconductores: los semiconductores se fabrican en instalaciones altamente especializadas. Este es el caso de los chips de memoria, así como con los microprocesadores. El proceso de producción en estas instalaciones por lo general consta de cuatro fases: la fabricación de obleas, prueba de obleas, montaje o embalaje y la prueba final. La fabricación de obleas es tecnológicamente la fase más compleja. Las capas de metal y material de obleas se construye en patrones en las obleas de silicio o arseniuro de galio para producir los circuitos. Cada capa requiere una serie de operaciones, las cuales típicamente incluyen:

  1. La limpieza.
  2. Oxidación, deposición y metalización.
  3. La litografía.
  4. Aguafuerte.
  5. La implantación de iones.
  6. Fotorresistencia.
  7. Inspección y medición

  Debido a que consiste en muchas capas, cada oblea se somete a estas operaciones varias veces. Por lo tanto, hay una cantidad significativa de recirculación en el proceso. Las obleas se mueven a través de la instalación en lotes de 24. Algunas máquinas pueden requerir ajustes para prepararlos para trabajos entrantes. El tiempo de preparación a menudo depende de la configuración del lote recién terminado y del lote a punto de comenzar. El número de pedidos en el sistema es a menudo en cientos y cada uno tiene su propia fecha de lanzamiento y fecha de vencimiento. El objetivo del planificador es satisfacer la mayor cantidad de envíos comprometidos con las fechas como sea posible,

al tiempo que se maximiza el rendimiento. Este último objetivo se consigue mediante la maximización de la utilización del equipo, especialmente de las máquinas de cuello de botella. Por lo tanto la minimización de los tiempos muertos y tiempos de preparación también son necesarios. En muchos entornos de fabricación, a sistemas de manipulación de materiales se automatiza el flujo de productos a través del sistema. Los sistemas flexibles de montaje se incluyen en esta categoría. El trabajo del planificador en este tipo de ambiente es desarrollar la mejor secuencia de tiempo que satisfaga de cierto modo las restricciones.

- Una cadena de montaje de automóviles: una línea de montaje de automóviles produce típicamente muchos modelos diferentes, todos pertenecen a un pequeño número de familias de coches. Por ejemplo, los diferentes modelos dentro de una familia pueden incluir un cupé de dos puertas, un sedan de cuatro puertas. También hay un número de diferentes colores y paquetes de opciones. Algunos vehículos tienen transmisiones automáticas, mientras que otros son manuales y algunos coches tienen techos solares, mientras que otros coches tienen techos sólidos.
  En una línea de montaje hay típicamente varios cuellos de botella, donde el rendimiento de una máquina o proceso en particular determina la tasa de producción general. El taller de pinturas es a menudo un cuello de botella, cada vez que cambia el color de las pistolas de pintura se tienen que limpiar y este es un proceso que consume tiempo. Uno de los objetivos es el de maximizar el rendimiento mediante la secuenciación de los coches de tal manera que la carga de trabajo en cada estación se equilibra con el tiempo.

- Secuenciación de la producción en una fábrica de papel: todo comienza con la preparación de la madera y la pulpa. El proceso finaliza con rollos de papel bobinados y cortados a las medidas requeridas. El corazón de la fábrica de papel son sus máquinas de papel, las cuales son muy grandes y representan una importante inversión de capital (entre 50 y 100 millones de dólares cada uno). Cada máquina produce varios tipos de papel que se caracterizan por sus pesos básicos, grados y colores. Los planes maestros de producción de estas máquinas suelen ser elaborados sobre una base anual. Los horarios son proyectos cíclicos con tiempos de ciclo de dos semanas o más. Un tipo particular de papel producido durante un cambio no cumple con cualquiera de las normas establecidas, o bien se vende con una gran descuento o considerados como desechos y se alimenta de nuevo en el sistema de producción. El plan de producción trata de maximizar la producción y minimizar los costos de inventario. Maximización de la producción implica reducir al mínimo los tiempos de cambio. Esto significa más largos los ciclos de producción, lo cual a su vez genera mayores costos de inventario. El plan de producción global es un libre comercio entre costos de instalación y los costos de inventario.

Entre otros ejemplos de proyectos se encuentran:

- Asignación de horarios a enfermeras en un hospital.
- Programación de un Torneo deportivo.
- Construcción de grandes edificaciones.
- Manufactura y ensamblaje de grandes productos (como barcos, generadores, etc.)
- Reparación y mantenimiento (de plantas nucleares, refinerías de petróleo, etc.)
- Diseño, desarrollo, y mercadeo de un nuevo producto.

En todo proyecto existen factores fundamentales como son: los objetivos, la unicidad, los caracteres temporales, al incertidumbre y sobre todo el ciclo de vida de un proyecto.

Objetivo: El objetivo del proyecto se refiere a la situación final que la gestión del proyecto está tratando de lograr y que puede ser usado para monitorear el progreso e identificar cuándo un proyecto se ha completado con éxito. Los tres objetivos fundamentales de la secuenciación de proyectos son el tiempo, costo y calidad. Todo proyecto está sujeto al tiempo, es decir tienen una fecha límite en la que debe concluirse. El coste está asociado al factor humano, maquinaria, material, uso de instalaciones u otros elementos que al final se traducen en un presupuesto económico. Para todos los proyectos el coste supone una limitación. La cantidad de tiempo dedicado a las tareas individuales determina la calidad global del proyecto. Algunas tareas pueden requerir una cantidad dada de tiempo para ser completadas adecuadamente, pero con más tiempo podrían ser completadas excepcionalmente. A lo largo de un proyecto la calidad puede tener un impacto significativo en el tiempo y en el coste (o viceversa).

Unicidad. Un proyecto suele ser único, no es un proceso repetitivo. Incluso los *proyectos repetitivos*, tales como la construcción de plantas químicas con las mismas especificaciones pueden tener diferencias distintivas en términos de recursos utilizados y el entorno real en el que el proyecto se realice.

Carácter temporal. Los proyectos tienen un comienzo y un final definidos, que por lo general conlleva a un uso concentrado de los recursos que se necesitan para llevar a cabo el proyecto.

Incertidumbre. Los proyectos se planifican antes de su ejecución, por lo que llevan un elemento de riesgo.

Ciclo de vida. Un proyecto pasa a través de un ciclo de vida que consta de diferentes fases. Para empezar, existe la fase de diseño conceptual durante el cual la organización se da cuenta de la necesidad del proyecto o recibe una solicitud de un cliente para proponer un plan para llevar a cabo un proyecto. A continuación, está la fase de definición del proyecto en el que se definen los objetivos del proyecto, el contenido de trabajo y se decide sobre los diferentes rendimientos. Una vez que el proyecto está bien definido y aprobado, la fase de

planificación del proyecto puede comenzar, lo que implica dividir el proyecto en paquetes de trabajo manejables que consisten en actividades específicas que deben llevarse a cabo con el fin de lograr los objetivos del proyecto. Las duraciones de las actividades deben ser estimadas y las necesidades de los recursos y disponibilidad, así como las relaciones de precedencia entre las actividades que deben ser determinadas con suficientes cuidado y detalle. Posteriormente, el proyecto entra en la fase de planificación que contempla la construcción de una secuencia que identifica el inicio y hora de finalización de las actividades. El cronógrama del proyecto entonces debe convertirse en realidad. Durante la ejecución del proyecto, su progreso debe ser monitoreado y se deben realizar acciones correctivas cuando sea necesario. Finalmente, el proyecto entra en su fase de terminación que consiste en la entrega de los resultados del proyecto.

En Demeulemeester y Herroelen [15] se realiza una revisión detallada del ciclo de vida del proyecto. El ciclo de vida de un proyecto enmarca sobre todo la gestión de un proyecto que consiste básicamente en la planificación, programación y control de las actividades del proyecto para alcanzar los objetivos de rendimiento, costo y tiempo para un ámbito determinado de trabajo, utilizando los recursos de manera eficiente y eficaz. Está gestión consiste en tres diferentes pasos:

1. **Datos:** Indispensable en un proyecto es saber, los tipos y número de recursos disponibles, las actividades y sus duraciones, requerimientos de recursos, y las relaciones tecnológicas de precedencia, además, la especificación de la función objetivo.

2. **Secuenciación de las actividades:** Dada la función objetivo, debemos conocer la mejor manera posible de realizar las actividades considerando las restricciones impuestas por las relaciones de precedencia y de recursos.

3. **Realización de un plan:** La construcción de un plan para el proyecto en el cual se especifican para cada actividad, las relaciones de precedencia y de recursos factibles, proporciona mayor efectividad en la realización de los objetivos del proyecto.

El trabajo desarrollado en este documento está enfocado a la etapa de la secuenciación del proyecto la cual consiste en definir los objetivos, el presupuesto, las actividades y en estimar sus duraciones aproximadas y los recursos que se requieren.

Los proyectos pueden tener elementos probabilísticos o elementos determinísticos. Los primeros se originaron con el programa de misiles nucleares para submarinos de la armada de los Estados Unidos de Norte América en 1958, y recibieron el nombre de PERT. Los segundos, CPM, método del camino crítico, surgió de los esfuerzos de la compañía DuPond para gestionar mejor la construcción y reparación de sus plantas químicas, entre 1956 y 1959.

No obstante, los métodos antes mencionados no tienen en cuenta las restricciones de recursos. Debido a la necesidad de disminuir los costes de un proyecto surge la necesidad de administrar de forma óptima los recursos. Con ese objetivo se han propuesto numerosas técnicas de solución, tanto heurísticas como metaheurísticas. Una de las fuentes de motivación de trabajos de investigación en esta linea han sido los Problemas de Secuenciación de Proyectos PSP. En la tabla 1.1 se muestran algunos de los más importantes PSP encontrados en literatura [51].

	Un Modo		Multi Modo		
	Sin recursos	Multiples recursos renovables	1 recurso no-renovable	1 recurso renovable	Multiples recursos renovables y no-renovables
	Sin intercambios	Sin intercambios	Intercambio tiempo/costo	Intercambio tiempo/costo	Intercambio tiempo/costo, tiempo/recurso, recurso/recurso
ZERO-LAG FS	CPM/PERT $cpm\|C_{\text{máx}}$	RCPSP $m,1\|cpm\|C_{\text{máx}}$	DTCTP $1,T\|cpm,disc,mu\|C_{\text{máx}}$	DTRTP $1,1\|cpm,disc,mu\|C_{\text{máx}}$	MRCPSP $m,1T\|cpm,disc,mu\|C_{\text{máx}}$
MIN SS, SF, FS,FF	PDM $min\|C_{\text{máx}}$	GRCPSP $m,1,va\|min,\rho_i,\delta_i\|C_{\text{máx}}$	GDTCTP $1,T\|min,\rho_i,disc,mu\|C_{\text{máx}}$	GDTRTP $1,1\|min,\rho_i,disc,mu\|C_{\text{máx}}$	GMRCPSP $m,1T\|min,\rho_i,disc,mu\|C_{\text{máx}}$
MIN+MAX SS, SF,FS,FF	MPM $gpr\|C_{\text{máx}}$	RCSP-GPR $m,1,va\|gpr,\rho_i,\delta_i,vr\|C_{\text{máx}}$	DTCTP-GPR $1,T\|gpr,\rho_i,\delta_i,disc,mu\|C_{\text{máx}}$	DTRTP-GPR $1,1,va\|gpr,\rho_i,\delta_i,disc,mu\|C_{\text{máx}}$	MRCPSP-GPR $m,1T,va\|gpr,\rho_i,disc,\mu\|C_{\text{máx}}$

Cuadro 1.1: Clasificación de los problemas de secuenciación

Los problemas son clasificados con respecto a relaciones de precedencia generalizada (CPM restricciones de precedencia, relaciones de tipo mínimo y máximo), multiples modos, tipos de recursos renovables y tipos de recursos no-renovables. La abreviación de cada tipo de problema se encuentra en la tabla 1.2 [51].

Abreviaciones

Clase del problema	Descripción
CPM/PERT	Metodo del camino critico
PDM	Precedence Diagramming Method
MPM	Metra Potential Method
RCPSP	Problema de secuenciación de proyectos con recursos limitados
GRCPSP	Problema de secuenciación de proyecto con recursos limitados generalizado
RCPSP-GPR	Problema de secuenciación de proyectos con recursos limitados con relaciones de precedencia generalizada
DTCTP	Problema sin intercambio tiempo/costo discreto
GDTCTP	Problema sin intercambio tiempo/costo discreto generalizada
DTCTP-GPR	Problema sin intercambio tiempo/costo discreto con relaciones de precedencia generalizada
DTRTP	Problema sin intercambio tiempo/recurso discreto
GDTRTP	Problema sin intercambio tiempo/recurso discreto generalizada
DTRTP-GPR	Problema sin intercambio tiempo/recurso discreto con relaciones de precedencia generalizada
MRCPSP	Problema de secuenciación de proyectos con recursos limitados multi-modo
GMRCPSP	Problema de secuenciación de proyectos con recursos limitados multi-modo generalizado
MRCPSP-GPR	Problema de secuenciación de proyectos con recursos limitados multimodo con relaciones de precedencia generalizada

Todo PSP consiste en actividades, recursos, relaciones de precedencia y funciones de evaluación. En lo que sigue consideraremos que cada dato necesariamente es determinístico, es viable y toma valores enteros.

1.3. Recursos

Los recursos son clasificados por categoría, tipos y números. Los recursos de categorías son distinguidos por recursos renovables, no-renovables, doblemente limitados y recursos parcialmente (no) renovables.

Recursos renovables: Es un tipo de recurso que puede restaurarse a una velocidad similar o superior a la de consumo estando disponible periodo a periodo, es decir, la cantidad disponible se renueva de un periodo a otro. Un ejemplo de recurso renovable es el caso del número de trabajadores cualificados disponibles para trabajar en el proyecto, en los cuales cada día es limitado aunque no se pone restricción en el número de días hábiles en que la mano de obra es utilizada.

Los recursos renovables también incluyen maquinaria, herramientas, equipo, espacio, etc. El conjunto de recursos renovables se denota por $\mathcal{R}^{\tau}$. Se usara $\mathcal{R}_k^{\tau}$ para denotar la disponibilidad (unidades) del tipo de recurso renovable $k \in \mathcal{R}^{\tau}$.

Recursos no-renovables: Los recursos no-renovables están limitados sobre la duración completa del proyecto, sin restricciones sobre cada periodo. Un ejemplo de recursos no-renovable es el presupuesto total para la realización de un proyecto. El conjunto de recursos no-renovables es denotado por $\mathcal{R}^{\eta}$. La disponibilidad de un tipo de recurso no-renovable $l \in \mathcal{R}^{\eta}$, es denotada por $\mathcal{R}_l^{\eta}$.

Recursos doblemente restringidos: Los recursos doblemente restringidos están limitados tanto sobre el horizonte de planificación como sobre cada período de tiempo. En Talbot [58] se demostró formalmente que cada recurso de este tipo se puede descomponer en uno renovable y uno no renovable. Un ejemplo de recurso doblemente restringido es el presupuesto, si éste está restringido globalmente y además existen restricciones diarias.

Recursos parcialmente renovables: Definen la disponibilidad de un recurso para subconjuntos de periodos. Estos mismos autores [58] demostraron que tanto los recursos renovables como los no renovables y los doblemente restringidos se pueden representar por recursos de ese tipo.

Dentro de cada categoría, podemos necesitar o utilizar diferentes recursos, cada uno con diferentes funciones a realizar. Se habla entonces de los diferentes tipos de recursos. La clasificación de tipo distingue, por tanto, dentro de cada categoría a los distintos recursos con respecto a la función que realizan.

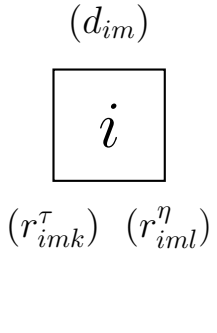
1.4. Actividades

Un proyecto consiste de $n \geq 1$ actividades reales $1, \ldots, n$, donde cada actividad es realizada sin interrupción, las actividades también son conocidas como trabajos, operaciones y tareas. Se introducen las actividades ficticias 0 y $n+1$, las cuales representan el inicio y la finalización del proyecto, respectivamente. El conjunto de actividades lo denotaremos por $V = \{0, 1, \ldots, n, n+1\}$.

Para completar el proyecto de forma satisfactoria es necesario procesar (ejecutar, secuenciar) cada actividad en uno de diversos modos, el conjunto de modos para una actividad $i \in V$, es denotado por $\mathcal{M}_i$, donde cada modo $m \in \mathcal{M}_i$ representa una forma distinta de realizar la actividad i. Un modo $m \in \mathcal{M}_i$ determina la duración $d_{im} \geq 0$ de la actividad $i \in V$, medida

en número de periodos o unidades de tiempo, que indica el tiempo necesario para completar la actividad, y si existe la posibilidad de interrumpir el proceso de esa actividad. Para las actividades ficticias $d_{01} = d_{(n+1)1} = 0$. Si el modo de ejecución de la actividad $i \in V$ es $m \in \mathcal{M}_i$, $r^{\tau}_{imk} \geq 0$ (donde $r^{\tau}_{imk} \leq \mathcal{R}^{\tau}_k$ y $r^{\tau}_{01k} = r^{\tau}_{n+11k} = 0$) son las unidades del recurso renovable k requeridas por la actividad i para su realización, mientras que $r^{\eta}_{iml} \geq 0$ son las unidades que se requieren del recurso no-renovable l con $r^{\eta}_{iml} \leq \mathcal{R}^{\eta}_l$ y $r^{\eta}_{01l} = r^{\eta}_{(n+1)1l} = 0$.

Sin perdida de generalidad, asumamos que los modos de cada actividad están organizados en orden no-decreciente en su duración, es decir, $d_{i1} \leq d_{i2} \leq \ldots \leq d_{i|\mathcal{M}_i|}$. Además, cada actividad lleva asociada un tiempo de inicio (marca el instante de tiempo a partir del cual se puede procesar la actividad) y un tiempo de finalización (indica el tiempo máximo en el que se debe haber completado la actividad). En la Figura (1.1) se representa una actividad, el vector de duraciones con respecto a cada modo y el consumo de recurso.



$$(d_{im})_{m\in\mathcal{M}_i} = \begin{pmatrix} d_{i1} \\ \vdots \\ d_{i|M_i|} \end{pmatrix}$$

$$(r^{\tau}_{imk}) = \begin{pmatrix} r^{\tau}_{i11} & \cdots & r^{\tau}_{i1|\mathcal{R}^{\tau}|} \\ \vdots & \ldots & \vdots \\ r^{\tau}_{iM_i1} & \cdots & r^{\tau}_{iM_i|\mathcal{R}^{\tau}|} \end{pmatrix}$$

$$(r^{\eta}_{iml}) = \begin{pmatrix} r^{\eta}_{i11} & \cdots & r^{\eta}_{i1|\mathcal{R}^{\eta}|} \\ \vdots & \ldots & \vdots \\ r^{\eta}_{iM_i1} & \cdots & r^{\eta}_{iM_i|\mathcal{R}^{\eta}|} \end{pmatrix}$$

Figura 1.1: Representación de los datos de una Actividad

1.5. Relaciones de precedencia

Las razones tecnológicas derivan en la secuenciación de una actividad que dependa de la ejecución de otra. Estas relaciones entre actividades se denominan **relaciones de precedencia**, porque la forma habitual (y la más sencilla) en que aparecen, se basa en que una actividad no puede comenzar antes de que otra(s) finalice(n).

Las relaciones de precedencia entre actividades se pueden visualizar representando el proyecto mediante un grafo dirigido donde cada actividad se representa por un nodo o vértice y una relación de precedencia entre dos actividades, por una arista dirigida. Esta representación se conoce como RAN.

Definición 1.3. *Una RAN (red con actividad en los nodos) es un par ordenado* (V, E)*, donde* V *es el conjunto de nodos y denota las actividades. Es decir* $V = \{0, 1, 2, \ldots, n, n+1\}$*, mientras que* $E = \{(i, j) : i, j \in V, i \to j\}$ *es el conjunto de arcos que representa las relaciones de precedencia.*

Para cada actividad i definimos

$$\mathcal{P}_i := \{j : (j, i) \in E\} \quad \text{y} \quad \mathcal{S}_i := \{j : (i, j) \in E\}$$

el conjunto de predecesores y sucesores inmediatos de i, respectivamente. Si $(i, j) \in E$, es decir $i \to j$: j no puede comenzar hasta que i haya terminado (i es predecesora inmediata de j y j es sucesora inmediata de i), el tipo de relación es FS (final-inicio, el final de la actividad i restringe el comienzo de la actividad j). Aquí la duración entre las actividades predecesoras es 0, sin embargo, es posible utilizar un parámetro distinto de cero, de modo que una actividad tenga que esperar un tiempo mínimo (máximo) después de haber terminado su predecesora para poder iniciar.

De acuerdo con Elmaghraby [14], también existen otro tipo de relaciones: Inicio-Inicio (SS), Inicio-Final (SF) y Final-Final (FF).

1.5.1. Relaciones de tipo mínimo

Relación final-inicio

La relación $FS_{ij}^{\text{mín}}(0)$ no requiere mayor explicación: una actividad j (por ejemplo la instalación de una grúa en un sitio) puede ser iniciada inmediatamente después que la actividad i (por ejemplo la preparación del lugar) se ha terminado. Esta relación estricta *final-inicio* es la tradicional PERT/CPM relación de precedencia. Si en un cierto número de unidades de tiempo, debe transcurrir entre el final de la actividad j, la relación final-inicio recibe un factor positivo adelanto o atraso. Es decir, si $FS_{ij}^{\text{mín}}(\pi)$ significa que el inicio de la actividad j no puede ser antes de π unidades de tiempo, después que la actividad i termine, ver figura 1.2

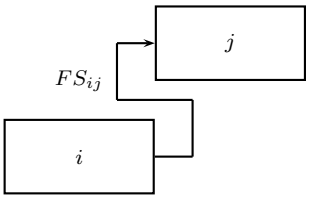


Figura 1.2: Relación Final-Inicio

Relación inicio-inicio

La relaciones inicio-inicio denota que entre el inicio de dos actividades puede haber un lapso de tiempo o que una actividad sucesora puede iniciar tan pronto como una parte especifica de otra actividad es finalizada. La relación $SS_{ij}^{\text{mín}}(3)$, denota que el inicio de la actividad j (por ejemplo un lugar para instalar la tubería) debe ir a 3 unidades de tiempo tras el inicio de la actividad i (nivelar el suelo). $SS_{ij}^{\text{mín}}(0)$ denota que la actividad j (nivelación del concreto) puede iniciar tan pronto como la actividad i (vertido del concreto) es iniciada. Una vez listo para la actividad i puede ser fácilmente modelado bajo la imposición de una relación de tipo mínimo inicio-inicio entre la actividad 0 (el nodo inicial ficticio en la red del proyecto) y la actividad i. Las relaciones $SS_{ij}^{\text{mín}}(x)$ son generalmente representadas con la figura 1.3

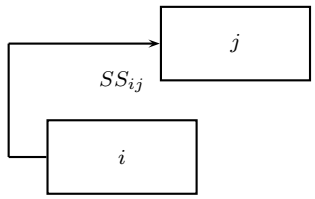


Figura 1.3: Relación Inicio-Inicio

Relación final-final

Se utiliza muy a menudo. Representa el requisito de que el final de una actividad j (para las paredes de acabado) debe ir al final de la actividad i (por ejemplo instalar la electricidad) por unidad de tiempo x, porque x unidades de la actividad j se necesitan para hacer frente a la salida de una unidad de tiempo de la actividad i. $FF_{ij}^{\text{mín}}(x)$ es generalmente representada en la figura 1.4

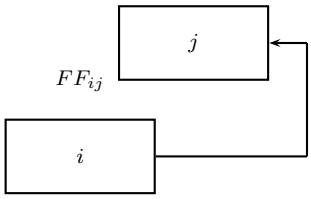


Figura 1.4: Relación Final-Final

Relación inicio-final

La relación inicio-final con relaciones de tipo mínimo son usadas con menos frecuencia. El siguiente ejemplo en Moder [44] ilustra un posible uso. $SF_{ij}^{\text{mín}}(45)$ denota que puede ocurrir entre el inicio de la actividad i (transmisión de diseño) porque al menos 2 unidades de tiempo de la actividad j (designar chasis) la cual tiene una duración de 30 días, depende de los resultados de los primeros 25 días de trabajo de la actividad i los cuales tienen una duración de 40 días $SF_{ij}^{\text{mín}}(x)$ es generalmente representada con la figura 1.5

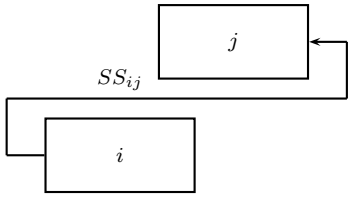


Figura 1.5: Relación Inicio-Final

1.5.2. Relaciones de tipo máximo

Relaciones de tipo máximo tienen interesantes aplicaciones. Una relación de tipo maximal $SS_{ij}^{\text{máx}}(x)$, por ejemplo, establecer que puede haber un máximo de unidades de tiempo x entre el inicio de la actividad i y la actividad j. Esto es muy útil para asegurar una demora máxima entre el inicio de 2 actividades. Supongamos por ejemplo que el plazo es puesto en el inicio de los trabajos de cimentación: el trabajo de cimentación debe comenzar, por ejemplo, antes del día 50 del proyecto. Esto puede ser representado por una relación de tipo máximo inicio-inicio $SS_{ij}^{\text{máx}}(50)$. Entre la representación del proyecto inicial representada por una actividad ficticia 1 y la correspondiente actividad j (sentar las bases).
Una relación de tipo máximo inicial-final establece que solo puede haber un máximo de unidades de tiempo x entre el inicio de una actividad y el fin de otra actividad. Por ejemplo, un plazo para la actividad i puede ser representado por una relación de tipo máximo inicial-final entre las actividades 1 y la i. Como un ejemplo similar, la entrega y el montaje de una estructura de acero prefabricada debe ser terminado por el mismo subcontratista, que debido a sus propios compromisos tiene a lo sumo 100 días para hacer eso, esta relación puede ser indicada por una relación $SF_{ij}^{\text{máx}}(100)$ entre la actividad i (entrega de acero prefabricado) y la actividad j (montaje de acero fabricado).

Las relaciones de precedencia que solo se utilizaran en este documento son las de fin-inicio con duración entre las actividades igual a cero, es decir, que la actividad sucesora no inicia hasta que no finaliza la predecesora.

Ejemplo 1.4. Mostremos un ejemplo de un PSP utilizando una RAN, con 5 actividades reales, un recurso renovable $\mathcal{R}^{\tau} = \{1\}$, un recurso no-renovable $\mathcal{R}^{\eta} = \{1\}$ tal que $\mathcal{R}^{\tau}_1 = 4$ y $\mathcal{R}^{\eta}_1 = 8$.

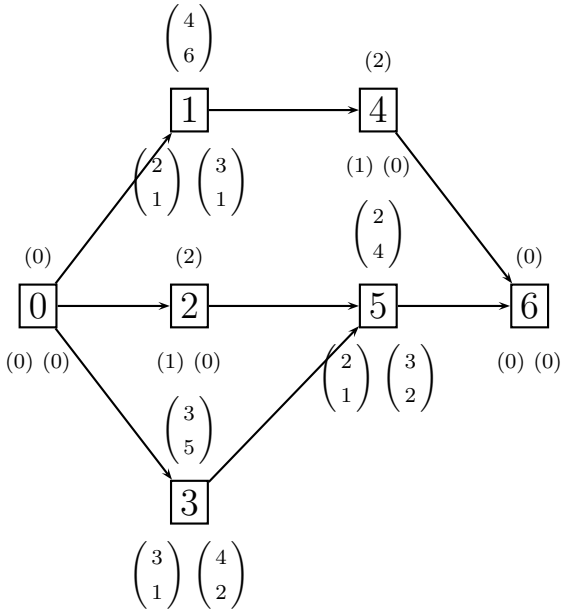


Figura 1.6: RAN del ejemplo PSP

j	0	1	2	3	4	5	6
m	1	1 2	1	1 2	1	1 2	1
d_{im}	0	4 6	2	3 5	2	2 4	0
r^{τ}_{im1}	0	2 1	1	3 1	1	2 1	0
r^{η}_{im1}	0	3 1	0	4 2	0	3 2	0

Cuadro 1.2: Datos del ejemplo PSP

1.6. Funciones de evaluación

Todos los problemas de optimización consisten en encontrar una *mejor* solución respecto a un criterio determinado. Generalmente una función evaluación o función objetivo es el criterio por el cual se cuantifica la calidad de una solución. Las funciones objetivos mostradas en Kolish [33], más estudiados son las siguientes:

- Minimizar la duración del proyecto: es, sin lugar a dudas, la medida más aplicada en el dominio de la secuenciación de proyectos. La duración está definida como el intervalo

de tiempo entre el comienzo y el fin del proyecto. Como el comienzo del proyecto es usualmente asumido en el tiempo $t = 0$, minimizar la duración es equivalente a minimizar el máximo de los tiempos de finalización de todas las actividades.

- Maximizar el valor actual neto del proyecto: cuando en proyectos de gran envergadura y a largo plazo están presentes cantidades significativas de flujo de dinero, en forma de gastos para iniciar las actividades y pagos para completar las partes del proyecto; el valor actual neto (VAN) es un criterio adecuado para medir la optimalidad del proyecto. Este criterio genera una ruta crítica de coste y no la ruta crítica de tiempos generada cuando se minimiza la duración.
- Maximizar la calidad del proyecto: este objetivo es muy importante para los directores de proyecto. La calidad de un proyecto está dada porque éste se haga dentro de los plazos planeados, cumpla con el presupuesto y que el cliente quede satisfecho con el producto. La formulación de este problema se ha centrado en minimizar la desviación de los plazos y el presupuesto establecido debido a actividades que deben ser preprocesadas.
- Minimizar el coste del proyecto: Este objetivo ha atraído mucha atención de los investigadores debido a su relevancia práctica. Se puede minimizar el coste de actividades ya que distintas maneras de desarrollar una actividad resulta en distintos costes directos, los cuales deben ser minimizados (intercambio coste-duración). Por otra parte puede plantearse minimizar el coste de los recursos que está determinado por la secuenciación de las actividades, la cual influencia el coste indirectamente a través de los recursos.

Cada función de evaluación define un problema distinto, aunque el conjunto de soluciones posibles sea el mismo. En este trabajo se aborda el MRCPSP cuyo objetivo es minimizar la duración del proyecto.

Definición 1.5. *En un PSP se define el espacio de modos por*

$$\mathcal{M} := \mathcal{M}_0 \times \mathcal{M}_1 \times \cdots \times \mathcal{M}_{n+1}$$

donde $M = (\mu_0, \mu_1, \ldots, \mu_{n+1}) \in \mathcal{M}$ *asigna a cada actividad* $i \in V$*, un único modo* $\mu_i \in \mathcal{M}_i$*.*

Ejemplo 1.6. En la tabla 1.2 se muestran los datos de un PSP, en el cual se puede definir el espacio de modos por:

$$\mathcal{M} = \left\{ \begin{array}{l} (1,1,1,1,1,1,1),(1,1,1,1,1,2,1),(1,1,1,2,1,1,1),(1,1,1,2,1,2,1), \\ (1,2,1,1,1,1,1),(1,2,1,1,1,2,1),(1,2,1,2,1,1,1),(1,2,1,2,1,2,1) \end{array} \right\}$$

Definición 1.7. *En un PSP se define el espacio de secuencias por*

$$\mathcal{S} := \mathcal{S}_0 \times \mathcal{S}_1 \times \cdots \times \mathcal{S}_{n+1}$$

donde $S = (s_0, s_1, \ldots, s_{n+1}) \in \mathcal{S}$ *asigna a cada actividad* $i \in V$*, un único tiempo de inicio* $s_i \in \mathcal{S}_i$*.*

Definición 1.8. *Para $i \in V$, sean ES_i, EF_i, LS_i y $LF_i \in \mathbb{Z}$ no negativos, llamados:*

ES_i ***(Earliest Start)**: Es el instante más temprano en que puede comenzar la actividad $i \in V$, respetando las relaciones de precedencia.*

EF_i ***(Earliest Finish)**: Es el instante más temprano en que puede finalizar la actividad $i \in V$, respetando las relaciones de precedencia.*

LS_i ***(Latest Start)**: Es el instante más tardío en que puede comenzar la actividad $i \in V$ si se exige que el proyecto termine en el instante T, respetando las relaciones de precedencias.*

LF_i ***(Latest Finish)**: Es el instante más tardío en que puede finalizar la actividad $i \in V$ si se exige que el proyecto termine en el instante T, respetando las relaciones de precedencias.*

Se utilizan los siguientes algoritmos para calcular los tiempos anteriores:

Algoritmo 1: Tiempos más tempranos

begin $ES_0 = EF_0 = 0$
  Para $j = 1$ *hasta* $n + 1$ **hacer**
    $ES_j := \text{máx}\{EF_i : i \in \mathcal{P}_j\}$;
    $EF_j := ES_j + d_j$

Algoritmo 2: Tiempos más tardíos

begin $LF_{n+1} = LS_{n+1} = EF_{n+1}$
  Para $j = n$ *hasta* 0 **hacer**
    $LF_j := \text{mín}\{LS_i : i \in \mathcal{S}_j\}$;
    $LS_j := LF_j - d_j$

Lo anterior, define las secuencias ES, EF, LS y $LF \in \mathcal{S}$ tal que

- $ES = (ES_0, ES_1, \ldots, ES_{n+1})$ llamada la secuencia de inicio más temprana.
- $EF = (EF_0, EF_1, \ldots, EF_{n+1})$ llamada la secuencia de finalización más temprana.
- $LS = (LS_0, LS_1, \ldots, LS_{n+1})$ llamada la secuencia de inicio más tardía.
- $LF = (LF_0, LF_1, \ldots, LF_{n+1})$ llamada la secuencia de finalización más tardía.

Ejemplo 1.9. Considere el proyecto del ejemplo (1.4), tomando solo el vector de modos $M = (1, 1, 1, 1, 1, 1, 1)$ y $\mathcal{R}^\eta = \emptyset$, entonces la red del proyecto es la siguiente:

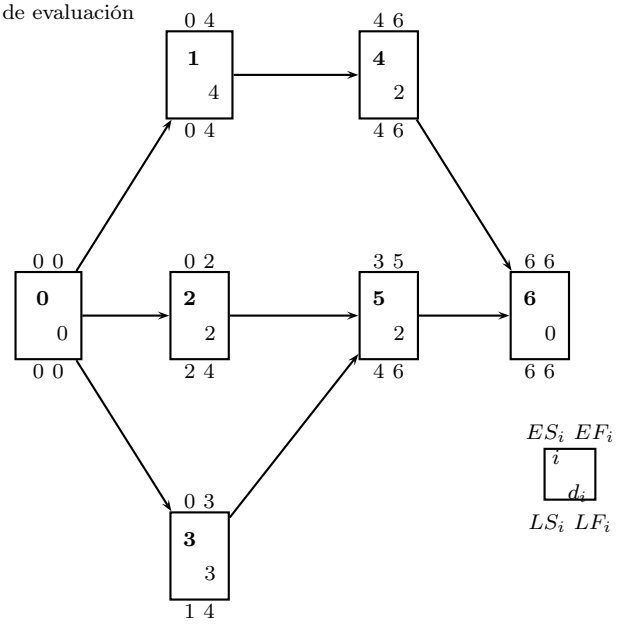


Figura 1.7: RAN asociada a los tiempos ES, EF, LS y LF.

La tabla 1.3 proviene de la secuencia de finalización más temprana (EF_i) y más tardía (LF_i), obtenidas de los algoritmos 1 y 2.

i	0	1	2	3	4	5	6
EF_i	0	4	2	3	6	5	6
LF_i	0	4	4	4	6	6	6

Cuadro 1.3: Vector EF y LF.

La representación gráfica de una secuencia S se da utilizando el diagrama de Gantt, donde el eje horizontal representa el tiempo y el vertical representa la utilización del único recurso, es necesario dibujar un diagrama para cada tipo de recursos.

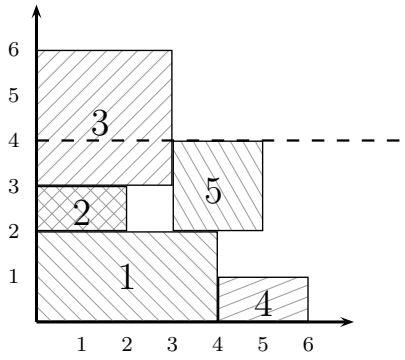


Figura 1.8: Representación de la secuencia ES

Teorema 1.10. *El problema de comprobar si existe una secuencia factible para una instancia dada de $PS|temp|C_{max}$ es NP-completo.*

Definición 1.11. *Una función objetivo $f : \mathcal{M} \times \mathcal{S} \to \mathbb{R}^+$, asigna a cada $(M, S) \in \mathcal{M} \times \mathcal{S}$ un valor $f(M, S) \geq 0$, donde $\mathbb{R}^+ := \{x \in \mathbb{R} : x \geq 0\}$. Una función objetivo es regular, si dado $M \in \mathcal{M}$ fijo, $f(M, S) \leq f(M, S')$ para toda $S, S' \in \mathcal{S}$ con $S \leq S'$ ($s_i \leq s'_i$ para todo $i \in V$)*

La minimización de la duración del proyecto (makespan) es, probablemente la función objetivo, más estudiada y aplicada en el dominio de la secuenciación de proyectos. La duración del proyecto se define como el tiempo transcurrido entre el principio y el final del proyecto. Dado que el inicio del proyecto se sitúa habitualmente en $t = 0$, minimizar la duración del proyecto se reduce a minimizar el máximo del inicio de la actividad final $n+1 \in V$, es decir, definiendo la función objetivo $\hat{f}(M, S) = \max s_{n+1}$, la minimización del makespan es:

$$\min \hat{f}(M, S)$$

1.7. Elementos aleatorios y deterministas

En los apartados anteriores no se ha comentado nada acerca del conocimiento incertidumbre de los datos iniciales de los problemas. Dependiendo de si todos los datos concernientes están determinados de forma exacta a priori o no, los problemas se dividen en deterministas y estocásticos.

El ejemplo más antiguo, y uno de los más conocidos y estudiados de los problemas estocásticos de secuenciación de proyectos, es el PERT, donde las relaciones de precedencia son del tipo FS con valor 0 y no hay restricciones de recursos, pero las duraciones de las actividades no son conocidas a priori, sino que únicamente se dispone de tres estimaciones de cada duración, la optimista, la pesimista, y la más probable.

Capítulo 2
El MRCPSP.

Introducción

El problema de secuenciación de proyectos con recursos limitados modo múltiple, al que se hace referencia en la literatura como MRCPSP (Multi-Mode Project Scheduling Problem), generaliza el problema tradicional (RCPSP), el cual consiste en realizar un proyecto o conjunto de actividades sujetas a tan solo dos tipos de restricciones. La primera restricción condiciona el orden en que se deben desarrollar las actividades, esto es, las restricciones de precedencia obligan a algunas actividades a comenzar después de la finalización de otras y la segunda es en el uso de los recursos limitados. En cambio el MRCPSP, además de las restricciones del RCPSP, tiene que cada actividad se puede secuenciar de varias maneras o modos, en los cuales cada actividad necesita de diferentes unidades de cada tipo de recurso y la duración también varia. Una restricción adicional es que se incluye otros tipos de recursos, los no renovables, cuya disponibilidad no está limitada en cada unidad de tiempo, sino a lo largo del proyecto completo.
El MRCPSP es motivo de interés por muchos científicos alrededor del mundo, de ahí el elevado número de trabajos de investigación desarrollados entorno a esté tema. Además del interés académico que despierta, también es atractivo por permitir modelar problemas de la vida real, por ejemplo: procesos de producción, mantenimiento de sistemas complejos, lanzamiento al mercado de nuevos productos, asignación de personal según su grado de cualificación, asignación de médicos y enfermeros, construcción de edificios o casas, o la renovación de un aeropuerto.

El objetivo del MRCPSP es secuenciar las actividades durante un tiempo en el que los recursos no escaseen y se optimice la función objetivo (makespan).

2.1. Formulación del modelo

El MRCPSP estándar involucra la selección de un modo de ejecución para cada actividad y la asignación de un tiempo de inicialización o finalización de las actividades, de tal manera que se cumplan las relaciones de precedencia, no se sobrepase la disponibilidad de los recursos y la duración del proyecto se minimice.

El problema es denotado por $m, 1T|cpm, disc, mu|C_{\text{máx}}$, según la clasificación de Herroelen [25] donde m proporciona el número de recursos, $1T$ la disponibilidad de los recursos renovables y no renovables de los cuales se especifica el tiempo de duración de cada uno y una base total del horizonte del proyecto, *cpm* reglamenta las relaciones de precedencia final-inicio sin retrasos, tal como se utiliza en el modelo básico de PERT/CPM, *disc* los requerimientos de recursos por parte de las actividades y son una función discreta de la duración de la actividad, *mu* las actividades tienen múltiples modos de ejecución preestablecidos y $C_{\text{máx}}$ el objetivo es minimizar la duración del proyecto.

La formulación de modelos matemáticos ha sido presentada por diversos autores. Los más difundidos en la literatura son Pritsker [50], Talbot [58], Kaplan [27], Alvarez-Valdes [59] y Mingozzi [43]. La representación formal dada por Talbot [58], el cual introdujo un modelo de programación binaria, donde las variables de decisión representan el tiempo de inicio de cada actividad y el modo en que será ejecutada:

$$x_{imt} = \begin{cases} 1, & \text{si la actividad } i \text{ se ejecuta en el modo } m \text{ y finaliza en el instante } t. \\ 0, & \text{en otro caso} \end{cases}$$

El MRCPSP puede ser formulado como sigue:

$$\text{Minimizar} \quad \sum_{t=ES_{n+1}}^{LS_{n+1}} t x_{(n+1)1t} \tag{1}$$

$$\text{sujeta a} \quad \sum_{m=1}^{M_i} \sum_{t=ES_i}^{LS_i} (t + d_{im}) x_{imt} \leq \sum_{m=1}^{M_i} \sum_{t=ES_j}^{LS_j} t x_{jmt} \quad \forall (i,j) \in E, \tag{2}$$

$$\sum_{m=1}^{M_i} \sum_{t=ES_i}^{LS_i} x_{imt} = 1 \quad \forall i \in N, \tag{3}$$

$$\sum_{i=1}^{n} \sum_{m=1}^{M_i} r_{imk}^{\tau} \sum_{s=\text{máx}\{t-d_{im,ES_i}\}}^{\text{mín}\{t-1,LS_i\}} x_{ims} \leq \mathcal{R}_k^{\tau} \tag{4}$$

$$\forall k \in R^{\tau} \text{ y } t = 1, ..., T,$$

$$\sum_{i=1}^{n} \sum_{m=1}^{M_i} r_{iml}^{\eta} \sum_{t=ES_i}^{LS_i} x_{imt} \leq \mathcal{R}_l^{\eta} \quad \forall l \in \mathcal{R}^{\eta}, \tag{5}$$

$$x_{imt} \in \{0, 1\} \quad \forall i \in N; m = 1, ..., M_i; \; t = 1, ..., T, \tag{6}$$

donde $ES_i(LS_i)$ es el tiempo más temprano (tardío) de inicio de la actividad j basado en los modos de menor duración y la cota superior del proyecto T, se obtiene sumando las duraciones máximas de todas las actividades del proyecto:

$$T = \sum_{j=1}^{n} \max_{m \in \mathcal{M}_j} d_{jm}$$

La función objetivo (1) minimiza la duración del proyecto. Se presupone que las actividades ficticias de inicio y fin solo se ejecutan en un único modo de duración cero y no consumen recursos. La ecuación (2) describe las relaciones de precedencia entre las actividades. El conjunto de restricciones (3) asegura que a cada actividad sólo se asigna un modo y un único tiempo de inicio. Las restricciones representadas mediante la ecuación (4) aseguran que no se sobrepase la disponibilidad de los recursos renovables en cada periodo mientras que la ecuación (5) vela por los recursos no renovables a lo largo del proyecto. Finalmente, la ecuación (6) impone valores binarios a las variables de decisión. Si en este problema solo se define un único modo de ejecución para cada actividad y no se especifican recursos no renovables, obtenemos el problema RCPSP.

Dada una secuencia $S = (s_i)_{i \in V}$ y un vector de modos $M = (m_i)_{i \in V}$, sea

$$\mathcal{A}(S, M, t) = \{i \in V : s_i \leq t < s_i + d_{i,m_i}\}$$

el conjunto de actividades reales las cuales serán secuenciadas en el tiempo t, también llamado conjunto activo.
El número de unidades del recurso no-renovable $\ell \in \mathcal{R}^{\eta}$ requerido por el conjunto de actividades V, dado el vector de modos M es

$$r_\ell^\eta(M) = \sum_{i=1}^{n} r_{i,m_i,\ell}^\eta, \qquad \ell \in \mathcal{R}^\eta.$$

El vector M es llamado vector de modos recursos factible si satisface las restricciones de recursos no-renovables

$$r_\ell^\eta(M) \leq \mathcal{R}_\ell^\eta \qquad \ell \in \mathcal{R}^\eta.$$

Por otra parte, el número de unidades del recurso renovable $k \in \mathcal{R}^\tau$ requerida por las actividades que se realizan en el instante t se define por:

$$r_k^\tau(S, M, t) = \sum_{i \in \mathcal{A}(S,M,t)} r_{i,m_i,k}^\tau \quad (k \in \mathcal{R}^\tau,\ t \geq 0).$$

La secuencia S es llamada de recurso factible con respecto al modo M si satisface las restricciones de recurso

$$r_k^\tau(S, M, t) \leq \mathcal{R}_k^\tau \qquad 0 \leq t \leq \overline{d} = \sum_{i \in V} d_{i,m_i}.$$

Además, la secuencia S es llamada de tiempo factible con respecto al vector de modos M si satisface las relaciones de precedencia $s_i + d_{i,m_i} \leq s_j$ para todo $(i, j) \in E$.

Definición 2.1. *Si el vector de modos M es recurso factible y la secuencia S es recurso y tiempo factible, entonces diremos que M es un vector de modos factibles y S es una secuencia factible.*

El problema de buscar una secuencia factible S con respecto a un vector de modo M y que minimize la duración del proyecto S_{n+1} puede modelarse de la siguiente forma dado por Christofides [8]:

$$\begin{array}{lll}
\text{mín} & s_{n+1} & (1)\\
\text{sujeta a} & r_k^\tau(S, M, t) \leq \mathcal{R}_k^\tau, \quad k \in \mathcal{R}^\tau, 0 \leq t \leq \overline{d}, & (2)\\
& r_\ell^\eta(M) \leq \mathcal{R}_\ell^\eta, \quad \ell \in \mathcal{R}^\eta, & (3)\\
& s_i + d_{i,m_i} \leq s_j \quad \forall (i,j) \in E, & (4)\\
& m_i \in \mathcal{M}_i, \quad i \in V & (5)\\
& s_i \geq 0, \quad i \in V & (6)\\
& s_0 = 0 & (7)
\end{array}$$

Definición 2.2. *Una solución factible al problema anterior, es una pareja $(M, S) \in \mathcal{M} \times \mathcal{S}$, donde M es un vector de modos factible y S es una secuencia factible con respecto al modo M. Una solución óptima (M, S) es aquella que es factible y minimiza la duración del proyecto. El conjunto de soluciones factibles lo denotaremos por $\mathcal{M}_T \times \mathcal{S}_T$. $\mathcal{OS}$ designa el conjunto de soluciones óptimas.*

Teniendo en cuenta que el RCPSP pertenece a la clase de problemas NP-Hard (Blazewics [5]) , implica que el MRCPSP siendo una generalización del RCPSP, también es un problema

de la clase NP-Hard. De hecho, para comprobar el problema factibilidad del MRCPSP, ya es un problema NP-completo. Para probar esto, Kolish [29] dividió el MRCPSP en dos subproblemas: el Problema de Asignación de Modos (MAP), en el cual a cada actividad i se le asigna un modo de ejecución $m \in \mathcal{M}_i$ de manera que la demanda de cada tipo de recurso no renovable sea menor que la disponibilidad. Cuando se han fijado los modos se obtiene el subproblema RCPSP.

Introduciendo las variables de decisión

$$x_{im} = \begin{cases} 1, & \text{si el modo } m \text{ es asignado a la actividad } i \\ 0, & \text{en otro caso} \end{cases}$$

El MAP puede ser formulado de la siguiente forma: si existe una vector de modos factible $M = (\mu_1, \dots \mu_n)$, i.e. un vector de variables x_{im} tal que las siguientes restricciones se cumplan:

$$\begin{aligned} &\sum_{m=1}^{M_i} x_{im} = 1 && i = 1, \dots, n \\ &\sum_{i=1}^{n} \sum_{m=1}^{M_i} r_{iml}^{\eta} x_{im} \leq \mathcal{R}_l^{\eta} && l \in \mathcal{R}^{\eta} \\ &x_{im} \in \{0, 1\} && i = 1, \dots, n;\ m = 1, \dots, M_i \end{aligned} \tag{2.1}$$

La primera ecuación asegura que cada actividad solo puede realizarse de un solo modo, en cambio la segunda no permite que la disponibilidad de los recursos no-renovables sea violada.

Teorema 2.3. *El MAP es NP-completo para* $|\mathcal{R}^{\eta}| \geq 2$ *y* $M_i \geq 2$, $1 \leq i \leq n$.

La prueba se puede leer en Kolish [29].

2.2. Métodos exactos

En esta sección vamos a discutir diferentes métodos exactos para resolver el MRCPSP, mostrados por Hartmann y Drexl [22]. La mayor parte de estos métodos se basan en los procedimientos de *Branch and Bound* (Ramificación y Poda), es decir, dividir el problema en subproblemas y tratar de eliminar los subproblemas mediante el cálculo de cotas inferiores o usando reglas de dominancia. Los métodos exactos aseguran la solución óptima del problema, aunque en general requieren un gran esfuerzo computacional, por tratarse de problemas NP-hard (Blazewicz [5]).

Los algoritmos de ramificación y poda construyen un árbol de búsqueda con el fin de explorar implícitamente el espacio de búsqueda. En cada nodo del árbol de búsqueda, dos situaciones pueden ocurrir:

1. En cada etapa se construye una secuencia parcial factible, en la cual están aquellas actividades a las que se les ha asignado un tiempo de inicio (secuenciadas) y una solución viable posible.
2. Cuando el calculo de las cotas inferiores, cotas superiores y, posiblemente, los ajustes de duración están realizados, el espacio de búsqueda correspondiente al nodo actual se divide en subconjuntos tales que la unión de estos subconjuntos corresponde al conjunto de soluciones de dicho nodo. Esta operación de partición del área del espacio de búsqueda en sub-zonas es llamado ramificación.

Existen diversos métodos para la ramificación, como el basado en el árbol de precedencia, en las alternativas de retraso y en las alternativas de extensión.

Definición 2.4. *Sea V el conjunto de actividades. Una secuencia parcial S_p, es una secuencia donde se le asignan los tiempos de inicio a tan solo (no necesariamente propio) un subconjunto de V. Diremos que S es una secuencia completa si a cada actividad en V, se le ha asignado un tiempo de inicio.*

2.2.1. Árbol de precedencia

Presentamos una formulación simplificada del algoritmo del árbol de precedencia. El procedimiento comienza con la actividad ficticia de partida en el tiempo $t = 0$. En cada nivel $g \in \{0, 1, \ldots, n+1\}$ del árbol de búsqueda, se determina el conjunto Sec_g de las actividades secuenciadas actualmente y el conjunto D_g de las actividades elegibles, una actividad elegible es aquella que aun no ha sido secuenciada y todas sus predecesoras sí lo han sido, por lo tanto

$$D_g := \{j \in V \setminus Sec_g : \mathcal{P}_j \subseteq Sec_g\}$$

Entonces seleccionamos una actividad elegible j_g y, posteriormente, un modo m_{j_g} de esta actividad. Ahora se busca el tiempo inicial s_{j_g} que no viole ninguna relación de precedencia y mucho menos la restricción de recursos, que no es menos que el tiempo inicial asignado en el nivel previo del árbol de búsqueda. Se escoge una actividad elegible y se programa tan pronto como sea posible, expandiendo así una rama del árbol (ramificación). Si dicha actividad es la ficticia final hemos encontrado una secuencia completa. En este caso, se retrocede al nivel anterior. Aquí, seleccionamos el proximo modo no probado. Si no existe ninguno, seleccionamos la proxima actividad elegible no probada. Si se han escogido todas las actividades elegibles y todos los modos viables, hacemos un retroceso al nivel anterior. Más formalmente, el algoritmo (3) muestra paso a paso el procedimiento.

Note que cada combinación de una actividad elegible y un modo relacionado corresponde a un descendiente del nodo actual en el árbol de búsqueda. Cada rama desde la raíz (actividad ficticia inicial) hasta una hoja del árbol de precedencia corresponde a una permutación del

Algoritmo 3: Árbol de precedencia

begin

$g := 0$; $j_0 := 0$; $m_{j_0} := 1$; $Sec_0 := \emptyset$;

1. Calcular actividades elegible;

   $g := g + 1$; $Sec_g := Sec_{g-1} \cup \{j_{g-1}\}$;

   $D_g := \{j \in \{1, \ldots, J+1\} \setminus D_g : P_j \subseteq Sec_g\}$;

   Si $J + 1 \in D_g$ **entonces** guardar solución actual e ir 4;

2. Selección de la proxima actividad;

   Si *no existe actividad elegible no probada, se deja en* D_g **entonces** ir a 4;

   Si no seleccionamos la actividad no probada $j_g \in D_g$;

3. Selección del proximo modo y calculo del tiempo de inicio;

   Si *no hay modo no probado, se deja* $\{1, \ldots, M_{j_g}\}$ **entonces** ir a 2;

   Si no seleccionamos el modo no probado $m_{j_g} \in \{1, \ldots, M_{j_g}\}$;

   Si *un conflicto w.r.t de recurso no-renovable ocurre* **entonces** ir a 3;

   Calcular precedencias más tempranas y recursos factible en tiempo de inicio s_{j_g} con $s_{j_g} \geq s_{j_{g-1}}$;

   ir a 1;

4. Regresar;

   $g := g - 1$; **Si** $g = 0$ **entonces** Stop **Si no** ir a 3;

conjunto de actividades $j_1, \ldots, j_n$ que es necesariamente factible en el sentido de que cada predecesor de una actividad j_g tiene un índice más pequeño en la secuencia que j_g. Además la unión en cada paso g, de los conjunto Sec_g y D_g no es el conjunto V en general, ya que puede haber actividades no secuenciadas no elegibles.

2.2.2. Alternativas de retraso

En esta sección se resume el enfoque de ramificación y poda propuesta por Sprecher y Hartmann [55]. Introduciendo la noción de un modo alternativo, extendiendo el concepto de alternativas de retraso usado por (Christofides [8]) y (Demeulemeester y Herroelen [12]) con un instante fijo t_g (punto de decisión) en el cual las actividades pueden iniciar. Consecuentemente, usamos una definición diferente de actividades elegibles en este algoritmo: Una actividad no secuenciada j es llamada elegible en el tiempo t_g si todos sus predecesores $i \in P_j$ son secuenciadas con un tiempo de finalización $f_i \leq t_g$. Además, una actividad j es secuenciada en un modo m_j con tiempo de inicio s_j se dice que está en proceso en el momento t_g si se tiene $s_j \leq t_g < s_j + d_{jm_j}$.

El procedimiento en el nivel actual g del árbol de búsqueda es el siguiente: Se determina el nuevo punto de decisión t_g como el tiempo de finalización más temprano de las actividades actualmente en proceso. Note que, debido a los niveles de disponibilidad constante de los recursos renovables, sólo el tiempo de finalización de las actividades secuenciadas deben ser

consideradas para iniciar otras no secuenciadas. Usando el conjunto F_g de las actividades que están finalizando en o antes del punto de decisión, se busca el conjunto D_g de las actividades elegibles. Entonces (temporalmente) comienzan las actividades elegibles en el punto de decisión que ya se han asignado un modo a un nivel anterior del árbol de búsqueda. Si hay actividades elegibles que aún no han sido asignados a un modo, es decir, si $D_g \setminus D_{g-1}$ es no vació, entonces calculamos el conjunto $\mathcal{SOMA}_g$ de modos alternativos: Un modo alternativo es un mapeo $\mathcal{MA}_g$ el cual asigna a cada actividad $j \in D_g \setminus D_{g-1}$ un modo $\mathcal{MA}_g(j) = m_j \in \{1, \ldots, M_j\}$. Seleccionando un modo alternativo, se puede (temporalmente) iniciar las actividades elegibles restantes en el punto de decisión también. Después de haber iniciado todas las actividades elegibles por adición entonces el conjunto IP_g de las actividades en proceso, puede haber violado las restricciones de recurso. Por lo tanto, calculamos el conjunto $\mathcal{SODA}_g$ de las alternativas de retraso mínimas de acuerdo con la siguiente definición: Una alternativa de retraso $\mathcal{DA}_g$ es un subconjunto de IP_g tal que para cada recurso renovable $k \in \mathcal{R}^\tau$

$$\sum_{i \in IP_g \setminus \mathcal{DA}_g} r^\tau_{im_ik} \leq \mathcal{R}^\tau_k$$

Una alternativa de retraso $\mathcal{DA}_g$ es llamada *minimal* si ningún subconjunto propio de $\mathcal{DA}_g$ es una alternativa de retraso. Se selecciona una alternativa de retraso minima y eliminamos las actividades que se retrasan en la secuencia parcial actual. Tenga en cuenta, si no se produce ningún problema con los recursos, la alternativa de retraso mínima es el conjunto vacío. Se guarda el tiempo inicial t_g de una actividad j que se retrasa en s^{old}_{gj} porque se tiene que restaurar la información durante el proceso. Luego se pasa a un nivel superior y se calcula el punto de decisión siguiente. Si tenemos una secuencia completa, se realiza un paso hacia atrás para probar la siguiente alternativa de retraso mínima o, si todos han sido probados, el siguiente modo alternativo. Formalmente, el algoritmo puede describirse en el algoritmo (4).

Observe que cada combinación de un modo alternativo y una alternativa de retraso mínima corresponde a nodo actual descendiente en el árbol de búsqueda. Evidentemente, este procedimiento es diferente del algoritmo (3) en que los conjuntos de actividades en lugar de (una sola) actividad se inicia en cada nivel del árbol de búsqueda. Además, aquí el instante de tiempo en el que las actividades pueden ser iniciadas se determina antes de que las actividades se estén seleccionando. Finalmente, en contraste con el algoritmo (3), este enfoque permite retirar las decisiones de programación en el nivel actual que se han hecho en un nivel inferior.

Algoritmo 4: Modo y alternativas de retraso.

begin

$g := 0$; $t_0 := 0$; $IP_0 = \{0\}$; $F_0 := \emptyset$; $m_0 := 1$; $s_0 := 0$; $D_0 := \emptyset$; $\mathcal{DA}_0 := \emptyset$;

1. Calcular un nuevo punto de decision y las actividades elegibles;

$g := g + 1$; $t_g := \min\{s_j + d_{jm_j} : j \in IP_{g-1}\}$;

$F_g := F_{g-1} \cup \{j \in IP_{g-1} : s_j + d_{jm_j} = t_g\}$;

$D_g := \{j \in \{1, \ldots, n+1\} \setminus (F_g \cup IP_{g-1}) : P_j \subseteq F_g\}$;

$IP_g := IP_{g-1} \setminus F_g \cup D_g$;

Si $n + 1 \in D_g$ **entonces** guardar solución actual e ir 6;

foreach $j \in \mathcal{DA}_{g-1}$ **do**
  actualizar $s_j := t_g$

;

2. Calcular los modos alternativos;

Si $D_g \setminus D_{g-1} = \emptyset$ **entonces** $\mathcal{SOMA}_g := \emptyset$ e ir a 4;

Si no $\mathcal{SOMA}_g := SetofModeAlternatives(D_g \setminus D_{g-1})$;

3. Selección del proximo modo alternativo;

Si *no hay modo no probado se deja* $\mathcal{SOMA}_g$ **entonces** ir a 6;

Si no seleccionamos modo no probado $\mathcal{MA}_g \in \mathcal{SOMA}_g$;

foreach $j \in D_g \setminus D_{g-1}$ **do**
  actualizar $m_j := \mathcal{MA}_g(j)$ y $s_j := t_g$

Si *un conflicto w.r.t un recurso no-renovable ocurre* **entonces** ir a 3;

4. Calcular alternativas de retraso;

$\mathcal{SODA}_g := SetOfMinimalDelayAlterantives(IP_g)$;

5. Seleccionar las alternativas de retraso próximas;

Si *la minima alternativa de retraso en* $\mathcal{SODA}_g$ *no se ha probado* **entonces** ir a 3;

Si no seleccionamos $\mathcal{DA}_g \in \mathcal{SODA}_g$; $IP_g := IP_g \setminus \mathcal{DA}_g$;

foreach $j \in \mathcal{DA}_g$ **do**
  guardar $s_{gj}^{old} := s_j$; ir a 1

6. Regresar;

$g := g - 1$; **Si** $g = 0$ **entonces** Stop, ;

Si no para cada $j \in \mathcal{DA}_g$ restaurar $s_j := s_{gj}^{old}$; $IP_g := IP_g \cup \mathcal{DA}_g$; ir a paso 5.;

2.2.3. Alternativas de extensión

Este apartado está dedicado a un nuevo enfoque de branch-bound para resolver el MRCPSP. Utilizando nuevamente el concepto de modo de alternativas desarrollados por Sprecher [55], presentamos las alternativas de extensión para la construcción de secuencias factibles. Una forma similar de extender las secuencias parciales ha sido propuesto pot Stinson [57] , para el caso de un solo modo.

Como en el algoritmo (4), cada nivel g del árbol de búsqueda esta asociado con un punto de decisión t_g, un conjunto IP_g de actividades en proceso, un conjunto F_g de las actividades finalizadas, y un conjunto D_g de actividades elegibles. Nuevamente se utiliza un modo de alternativas para fijar los modos de dichas actividades elegibles que aún no se les ha sido

asignado un modo. Luego, ampliamos la secuencia parcial actual, a partir de un subconjunto de las actividades elegibles en el punto de decisión, sin violar las restricciones de recursos renovables. Más precisamente, una *extensión alternativa* $\mathcal{EA}_g$ es un subconjunto del conjunto de las actividades elegibles para las cuales se tiene

$$\sum_{i \in IP_g \cup \mathcal{EA}_g} r^{\tau}_{im_i k} \leq \mathcal{R}^{\tau}_k$$

para cada recurso renovable $k \in \mathcal{R}^{\tau}$ y, por otra parte, $\mathcal{EA}_g \neq \emptyset$ si $IP_g \neq \emptyset$. Tenga en cuenta, a fin de asegurar que el algoritmo finalize, sólo podemos contar con alternativas de extensión no vacías si no hay actividades en proceso. Sin embargo, si existen en la actualidad las actividades en proceso, el conjunto vacío es siempre una alternativa de extensión, que debe ser probada para garantizar optimalidad.
En el actual nivel g del árbol de búsqueda se procede de la siguiente forma: Se determina un nuevo punto de decisión y calculamos el conjunto de actividades elegibles. Entonces determinamos el conjunto de modos alternativos $\mathcal{SOMA}_g$ para fijar los modos de las actividades elegibles que no han sido elegibles antes, es decir, las actividades de los modos de los cuales aún no han sido fijados. Después de seleccionar un modo alternativo $\mathcal{MA}_g$, calculamos el conjunto de alternativas de extensión $\mathcal{SOEA}_g$. Finalmente, seleccionamos una alternativa de extensión $\mathcal{EA}_g$ e iniciamos las actividades correspondientes antes de la ramificación al siguiente nivel. El mecanismo de desecuenciación es igual a la del algoritmo (4). Formalmente, se describe en el algoritmo (5).

Cada combinación de un modo alternativo y una alternativa de extensión corresponde a un descendiente del nodo actual en el árbol de búsqueda. Note que este procedimiento es diferente del algoritmo (4). Mientras que este ultimo incluye la posibilidad de retrasar las actividades que se han iniciado en un nivel más abajo que el actual, el nuevo enfoque no permite retirar una decisión de secuenciación de un nivel inferior. Como una consecuencia de ello, no puede restringir la búsqueda a una alternativa *maximal* de extensión, mientras que no perdamos la optimalidad al considerar sólo las alternativas de retraso minimal.

2.2.4. Reglas de dominancia

Esta sección resume los criterios de limitación mostrados en Hartmann [22] que agilizan los procedimientos de enumeración de la sección anterior. Aunque la mayoría de las reglas se conocen en la literatura, Hartmann [22] presenta dichas reglas y transfiere algunas más conocidas a los esquemas de enumeración que aún no habían sido definidos. En aras de la brevedad se han omitido las pruebas de esas reglas que son conocidas en la literatura.

Algoritmo 5: Modo y alternativas de extensión.

begin

$g := 0$; $t_0 := 0$; $IP_0 = \{0\}$; $F_0 := \emptyset$; $m_0 := 1$; $s_0 := 0$; $D_0 := \emptyset$; $\mathcal{DA}_0 := \emptyset$;

1. Calcular un nuevo punto de decision y las actividades elegibles;

$g := g + 1$; $t_g := \text{mín}\{s_j + d_{jm_j} : j \in IP_{g-1}\}$;

$F_g := F_{g-1} \cup \{j \in IP_{g-1} : s_j + d_{jm_j} = t_g\}$;

$D_g := \{j \in \{1, \ldots, n+1\} \setminus (F_g \cup IP_{g-1}) : P_j \subseteq F_g\}$;

$IP_g := IP_{g-1} \setminus F_g \cup D_g$;

Si $n + 1 \in D_g$ **entonces** guardar solución actual e ir 6;

2. Calcular los modos alternativos;

Si $D_g \setminus D_{g-1} = \emptyset$ **entonces** $\mathcal{SOMA}_g := \emptyset$ e ir a 4;

Si no $\mathcal{SOMA}_g := SetofModeAlternatives(D_g \setminus D_{g-1})$;

3. Selección del proximo modo alternativo;

Si *no hay modo no probado se deja* $\mathcal{SOMA}_g$ **entonces** ir a 6;

Si no seleccionamos modo no probado $\mathcal{MA}_g \in \mathcal{SOMA}_g$;

foreach $j \in D_g \setminus D_{g-1}$ **do**
  actualizar $m_j := \mathcal{MA}_g(j)$ y $s_j := t_g$

Si *un conflicto w.r.t un recurso no-renovable ocurre* **entonces** ir a 3;

4. Calcular alternativas de extensión;

$\mathcal{SOEA}_g := SetOfExtensionAlterantives(D_g, IP_g)$;

5. Seleccionar las alternativas de extensión próximas;

Si *las alternativas de retraso en* $\mathcal{SOEA}_g$ *no se ha probado* **entonces** ir a 3;

Si no seleccionamos $\mathcal{EA}_g \in \mathcal{SOEA}_g$; $IP_g := IP_g \cup \mathcal{EA}_g$;

foreach $j \in \mathcal{EA}_g$ **do**
  guardar $s_j := t_g$; ir a 1

6. Regresar;

$g := g - 1$; **Si** $g = 0$ **entonces** Stop ;

Si no $IP_g := IP_g \setminus \mathcal{EA}_g$; ir a paso 5;

2.2.4.1. Reglas basadas en lapsos de tiempo

El primer criterio delimitador hace uso de lapsos de tiempo determinado por MPM. Sprecher [53] and Sprecher [55], han utilizado esta regla en sus algoritmos para resolver el MRCPSP. Teniendo en cuenta las relaciones de precedencia y una cota superior en el makespan del proyecto (por ejemplo, que está dada por la suma de las duraciones máximas de las actividades), se utilizan los modos de duración menor y obtenemos el tiempo de finalización más tardío LF_j para cada actividad j por la recursión hacia atrás tradicional. Si un procedimiento ha encontrado la primera o una secuencia de mejora con un makespan T, los tiempos de finalización más tardíos se recalculan por $LF_j := LF_j - (LF_{n+1} - T + 1)$ para $j = 1, \ldots, n + 1$. De la definición de tiempo de finalización más tardío se deriva la siguiente regla de dominancia:

Regla de dominancia 1 *(Regla basada en lapsos de tiempo) Si existe una actividad secuenciada en la cual, el tiempo de finalización asignado excede el tiempo de finalización más tardío, entonces la secuencia parcial actual no puede ser completada con un makespan menor que el mejor conocido actualmente.*

Usando la definición de lapsos de tiempo y teniendo en cuenta explícitamente los modos múltiples, Sprecher se ha desarrollado la siguiente regla para el algoritmo del árbol de precedencia:

Regla de dominancia 2 *(Regla sin retraso para el Algoritmo 3) Si existe una actividad elegible que no puede ser secuenciada factiblemente en cualquier modo de la secuencia parcial actual, sin exceder el tiempo de finalización más tardío, entonces no hay otra actividad elegible que deba ser examinada en este nivel.*

Teniendo en cuenta las diferencias entre el procedimiento del árbol de preferencias por un lado y los algoritmos basados en modo de alternativas en los otros, podemos adaptar la regla de dominancia 2 como sigue:

Regla de dominancia 2' *(Regla sin retraso para los Algoritmo (4) y (5)) Si una actividad elegible el modo de que no se ha fijado aún no se puede iniciar en el modo en el menor tiempo posible en el punto de decisión actual, sin exceder su tiempo de finalización más tardío, entonces el modo alternativo necesita ser examinado en el nivel actual.*

2.2.4.2. Preproceso

El preproceso es utilizado en el MRCPSP para reducir el espacio de búsqueda y se basa en dos reglas de dominancia que pueden ser implementadas por el preprocesamiento (procedimiento que se ejecuta antes de cualquier método de resolución). La primera ha sido originalmente propuesta por Sprecher [55]. Utiliza las siguientes definiciones:

Definición 2.5. *Sea $i \in V$ una actividad. Un modo $m \in \mathcal{M}_i$ es llamada no ejecutable si su ejecución viola las restricciones de recursos renovables o de no-renovables en cualquier secuencia $S \in \mathcal{S}$, es decir*

$$r^{\tau}_{i,m,k} > \mathcal{R}^{\tau}_{k} \quad (k \in \mathcal{R}^{\tau}) \quad o \quad r^{\eta}_{i,m,\ell} + \sum_{\substack{j=1 \\ j \neq i}}^{n} \min_{m \in \mathcal{M}_j} \{r^{\eta}_{j,m,\ell}\} > \mathcal{R}^{\eta}_{\ell} \quad (\ell \in \mathcal{R}^{\eta})$$

Definición 2.6. *Sea $i \in V$ una actividad. Un modo $m \in \mathcal{M}_i$ es llamado ineficiente, si existe un modo $\mu \in \mathcal{M}_i$ con $d_{i,\mu} \leq d_{i,m}$, $r^{\tau}_{i,\mu,k} \leq r^{\tau}_{i,m,k}$ para todo $k \in \mathcal{R}^{\tau}$ y $r^{\eta}_{i,\mu,\ell} \leq r^{\eta}_{i,m,\ell}$ para todo $\ell \in \mathcal{R}^{\eta}$.*

Definición 2.7. *Un recurso no renovable $\ell \in \mathcal{R}^{\eta}$ es llamado redundante si la suma de los requerimientos máximos de las actividades de dicho recurso no excede su disponibilidad. Es decir,*

$$\sum_{j=1}^{n} \max_{m \in \mathcal{M}_j} \{r^{\eta}_{j,m,\ell}\} \leq \mathcal{R}^{\eta}_{\ell}$$

Es evidente que los modos no ejecutables e ineficientes, así como los recursos redundantes no renovables pueden ser excluido de los datos del proyecto, sin perder la optimalidad. Sprecher [55]. Se describen varios efectos de interacción que aparecen cuando los modos o recursos no renovables, se eliminan, por ejemplo, la eliminación de un recurso no renovable redundante puede provocar la ineficiencia de un modo. Por lo tanto, proponen la siguiente manera de preparar los datos de entrada:

Regla de dominancia 3 *(Reducción de datos)*

- ***Paso 1**: Remover todos los modos no ejecutables de los datos del proyecto*
- ***Paso 2**: Eliminar todos los recursos no renovables redundantes*
- ***Paso 3**: Eliminar todos los modos ineficientes*
- ***Paso 4**: Si se ha eliminado algún modo en el paso 3, volver al paso 2.*

Ejemplo 2.8. Consideremos el siguiente ejemplo, utilizado en [54]. El cual es un proyecto multi-modo con $n = 4$ actividades, donde cada actividad puede ser procesada en dos diferentes modos $m = 1, 2$. Se dan 3 recursos: un recurso renovable $\mathcal{R}^{\tau} = \{1\}$ con capacidad $\mathcal{R}^{\tau}_1 = 4$ y dos recursos no-renovables $\mathcal{R}^{\eta} = \{1, 2\}$ con $\mathcal{R}^{\eta}_1 = 13$ y $\mathcal{R}^{\eta}_2 = 14$. Por otra parte, las actividades $i = 1, \ldots, 4$ en los respectivos modos m tiene tiempo de procesamiento $d_{i,m}$ y requerimientos de recursos $r^{\tau}_{i,m,1}$ y $r^{\eta}_{i,m,\ell}$ para $\ell = 1, 2$:

j	0	1	2	3	4	5
m	1	1 2	1 2	1 2	1 2	1
d_{im}	0	2 4	3 5	2 3	3 4	0
r^{τ}_{im1}	0	5 2	3 1	2 1	2 2	0
r^{η}_{im1}	0	2 4	3 2	8 2	3 1	0
r^{η}_{im2}	0	1 1	3 4	3 3	2 7	0

De hecho $r^{\tau}_{111} = 5 > \mathcal{R}^{\tau}_1 = 4$ el modo 1 de la actividad 1 es no ejecutable con respecto al recurso renovable 1 y puede ser eliminado de los datos iniciales del proyecto.

j	0	1	2	3	4	5
m	1	2	1 2	1 2	1 2	1
d_{im}	0	4	3 5	2 3	3 4	0
r^{τ}_{im1}	0	2	3 1	2 1	2 2	0
r^{η}_{im1}	0	4	3 2	8 2	3 1	0
r^{η}_{im2}	0	1	3 4	3 3	2 7	0

Esto induce que el modo 1 de la actividad 3 sea no ejecutable con respecto al recurso no-renovable 1 dado que

$$r^{\eta}_{3,1,1} + \sum_{\substack{j=1 \\ j\neq 3}}^{4} \min_{m\in\mathcal{M}_j} \{r^{\eta}_{j,m,1}\} = 15 > \mathcal{R}^{\eta}_1 = 13$$

Después removiendo este modo,

j	0	1	2	3	4	5
m	1	2	1 2	2	1 2	1
d_{im}	0	4	3 5	3	3 4	0
r^{τ}_{im1}	0	2	3 1	1	2 2	0
r^{η}_{im1}	0	4	3 2	2	3 1	0
r^{η}_{im2}	0	1	3 4	3	2 7	0

el recurso no-renovable 1 es redundante y puede ser eliminado, puesto que

$$\sum_{j=1}^{4} \max_{m\in\mathcal{M}_j} \{r^{\eta}_{j,m,1}\} = 12 < 13 = \mathcal{R}^{\eta}_1$$

Entonces el modo 2 de la actividad 4 es ineficiente y eliminando esté modo causa que el recurso 2 también sea redundante. Dada la instancia inicial, por la eliminación anterior de modos y recursos no-renovables, se transforma en una instancia con 1 recurso renovable $\mathcal{R}^{\tau} = \{1\}$ con $\mathcal{R}^{\tau}_1 = 4$ y ningún recurso no-renovable (el número de modos es ajustado)

j	0	1	2	3	4	5
m	1	2	1 2	2	1	1
d_{im}	0	4	3 5	3	3	0
r^{τ}_{im1}	0	2	3 1	1	2	0

La proxima regla de dominancia está designada para los casos con recursos no renovables.

Regla de dominancia 4 *(Regla recursos no renovables) Si programando cada actividad actual no secuenciada en el modo con el menor requerimiento de un recurso no renovable que pueda exceder la capacidad de este recurso no renovable, entonces la secuencia parcial actual no puede ser factible completada.*

Sprecher [53] adapto la regla al MRCPSP y mejora los efectos reformulado como una regla estática. Antes de que un algoritmo se ejecuta, los datos del proyecto se ajusta de la siguiente manera: Definiendo $r^{\eta,\min}_{j,\ell}$ como el requerimiento mínimo de la actividad j del recurso no-renovable ℓ, actualizamos $r^{\eta}_{j,m,\ell} := r^{\eta}_{j,m,\ell} - r^{\eta,\min}_{j,\ell}$ para $j = 1,\ldots,n$, $m = 1,\ldots,M_j$,

$\ell \in \mathcal{R}^\eta$ y

$$\mathcal{R}^\eta_\ell := \mathcal{R}^\eta_\ell - \sum_{j=1}^{n} r^{\eta,\text{mín}}_{j,\ell}$$

Este método es ampliamente usado en la solución del MRCPSP por métodos exactos y aproximados.

2.2.4.3. Justificación

Se proporcionan tres reglas de dominancia que hacen uso de una clasificación del conjunto de secuencias. La noción de actividad semi-activa y activa como formalmente define Sprecher [56] para el caso del RCPSP puede extenderse al MRCPSP.

Definición 2.9. *Una justificación a la izquierda de una actividad $i \in V$ es una operación $JI_i : \mathcal{S}_T \to \mathcal{S}_T$ definida por $JI_i(S) = S'$ tal que $s'_i < s_i$ y $s'_j = s_j$ para todo $j \in V$, $j \neq i$.*

Observemos que en realidad JI_i no es una función y su objetivo es mejorar(minimizar) el tiempo de finalización de la secuencia factible o en el peor de los casos dejarla con la misma duración.

Definición 2.10. *Sean $i \in V$ una actividad y JI_i una justificación a la izquierda de $i \in V$.*

1. *JI_i se dice que es de 1-periodo, si para todo $S \in \mathcal{S}_T$ con $JI_i(S) = S'$, $s_i - s'_i = 1$.*
2. *JI_i se dice que es local, si se obtiene por uno o más justificaciones a la izquierda de 1-periodo.*
3. *JI_i se dice que es global, si no es local.*

Observación 2.11.

1. Considerando una medida de rendimiento regular u y una secuencia S' tal que $JI_i(S) = S'$, entonces S es dominada por S' bajo u.
2. Cada secuencia intermedia derivada de una justificación a la izquierda es factible por definición.
3. Si JI_i es global, entonces $\forall S \in \mathcal{FS}$ con $JI_i(S) = S'$ se tiene que $s_i - s'_i > 1$.
4. Si $S' \in \mathcal{S}_T$ con $JI_i(S) = S'$ para algún $S \in \mathcal{FS}$ y JI_i global, entonces S' no se obtiene de S por una serie de justificaciones a la izquierda locales, donde al menos una secuencia intermedia no es factible con respecto a la restricción de recursos.

Definición 2.12. *Sea S una secuencia factible sobre un conjunto de actividades V.*

1. *S es una secuencia semi-activa si ninguna de las actividades en V pueden ser transformadas bajo una justificación a la izquierda local.*
2. *S es una secuencia activa si ninguna de las actividades en V pueden ser transformadas bajo una justificación a la izquierda local o global.*

3. *S es una secuencia sin retraso si ninguna de las actividades puede empezar antes sin retrasar alguna otra actividad, aunque se permita la interrupción de actividades (suponiendo que cada unidad de duración de cada actividad se secuencia sin interrupción).*

Teorema 2.13. *Sea $\mathcal{S}$ el conjunto de secuencias, $\mathcal{S}_T$ el conjunto de secuencias factibles, $\mathcal{SAS}$ el conjunto de secuencias semi-activas, $\mathcal{AS}$ el conjunto de secuencias activas y $\mathcal{NDS}$ el conjunto de secuencias sin retraso, entonces se cumple lo siguiente:*

$$\mathcal{NDS} \subseteq \mathcal{AS} \subseteq \mathcal{SAS} \subseteq \mathcal{S}_T \subseteq \mathcal{S}$$

Una justificación a la izquierda de una actividad dentro de una secuencia dada es una reducción de su tiempo de finalización sin cambiar su modo y sin cambiar los tiempos de finalización de las otras actividades, de tal manera que la secuencia resultante no viole las restricciones de precedencia y de recursos. Una justificación a la izquierda local es una justificación a la izquierda que se obtiene por una o más justificaciones a la izquierda sucesivas de un periodo. Una secuencia se llama semi-activa, si ninguna de las actividades puede ser justificada a la izquierda local. Siguiendo de French [16], se puede afirmar que si existe una secuencia optima para una instancia determinada, entonces no hay una secuencia semi-activa óptima. Este resultado es utilizado para la siguiente regla de dominancia, que ha sido empleada por Sprecher [53] y Sprecher [55] para el caso multi-modo.

Regla de dominancia 5 *(Regla de Justificación a la Izquierda) Si una actividad que se ha iniciado en el nivel actual del árbol de búsqueda se puede justificar a la izquierda local sin cambiar su modo, entonces la secuencia parcial no es completa.*

Está regla permite un cambio de modo de la actividad que se va a justificar, sin embargo Sprecher [55] define la noción de una justificación a la izquierda multi-modo: Dentro de una secuencia dada, una justificación a la izquierda multi-modo es una reducción del tiempo de finalización de una actividad sin tener que cambiar los modos o los tiempos de finalización de las otras actividades, de manera que la secuencia resultante sea factible. Una secuencia es llamada fuerte si no se puede realizar una justificación a la izquierda multi-modo. La noción de una secuencia fuerte ha sido introducida por Speranza y Varcellis [52].

Otra operación en una secuencia de una instancia del MRCPSP es presentada por Sprecher [55]. Una reducción de modo en una actividad es una reducción del número de modos sin cambiar el tiempo de finalización y sin violar las restricciones o cambiando los modos y tiempos de finalización de las otras actividades. Una secuencia es llamada modo-minimal si existe una actividad en modo reducción que pueda ser realizada.

Note que existen secuencias fuertes que no son modo-minimal y vice versa. Obviamente, si existe una secuencia óptima para un caso dado, entonces existe una secuencia optima la cual

es fuerte y modo-minimal. Consecuentemente, la siguiente regla propuesta por Sprecher [55] provoca una marcha atrás cuando esta cierta secuencia no fuerte o modo-minimal puede ser obtenida de la secuencia parcial actual.

Regla de dominancia 6 *(Regla Multi-Modo) Asumamos que una actividad actualmente no secuenciada puede ser iniciada antes de la finalización de una actividad secuenciada j donde la secuencia parcial actual es completada. Si una justificación a la izquierda multi-modo o una reducción de modo de la actividad j con resultado modo $m'_j \in \mathcal{M}_j$, puede ser representada en la secuencia parcial actual y, asimismo, si $r^{\eta}_{jm'_j l} \leq r^{\eta}_{jm_j l}$ se cumple para cada recurso no-renovable $l\mathcal{R}^{\eta}$, entonces la secuencia parcial necesariamente no es completada.*

Claramente, si la justificación a la izquierda multimodo no puede aplicarse, entonces una justificación local a la izquierda tampoco puede aplicarse. Sin embargo, es útil para comprobar si los dos tipos de justificación hacia la izquierda por separado coinciden según las dos reglas de dominancia anteriores. Tenga en cuentas si una justificación local a la izquierda cuando la actividad correspondiente se ha iniciado. Sin embargo, sólo podemos comprobar si hay una justificación a la izquierda multimodo si la actividad correspondiente ya ha terminado. De lo contrario, como se ha indicado por Hartmann y Sprecher [23], perderíamos el óptimo. En consecuencia, la regla de justificación local a la izquierda no es superfluo, como la exclusión de una secuencia parcial a una posible justificación local a la izquierda, se puede detectar un nivel inferior de la rama del árbol de búsqueda y el limite de la misma (el modo de preservación) justificación local a la izquierda.

La proxima operación y la categoría relacionada de secuenciación son nuevas: Denotando el tiempo de finalización de una secuencia de la actividad j con $f_j = s_j + d_{jm_j}$, consideraremos dos actividades i y j con $i > j$ que sucesivamente es procesada con una secuencia, es decir, $f_i = s_j$. Ahora bien, un *orden de intercambio* es definido como el intercambio de estas dos actividades asignándoles un nuevo tiempo de inicio y un tiempo de finalización $s'_j := s_i$ y $f'_i := f_j$, respectivamente. De está manera, las restricciones de precedencias y de recursos pueden no ser violadas, y los modos y tiempos de inicio de las otras actividades que no pueden ser modificados. Una secuencia en la cual no se puede realizar un orden de intercambio es llamada de orden monótono. Claramente, es suficiente para enumerar solo secuencias de orden monótonas. Cabe señalar que hay secuencias que son ajustadas y de modo-minimal, pero no de orden monótono y vice-versa. Aplicamos la siguiente criterio de dominancia:

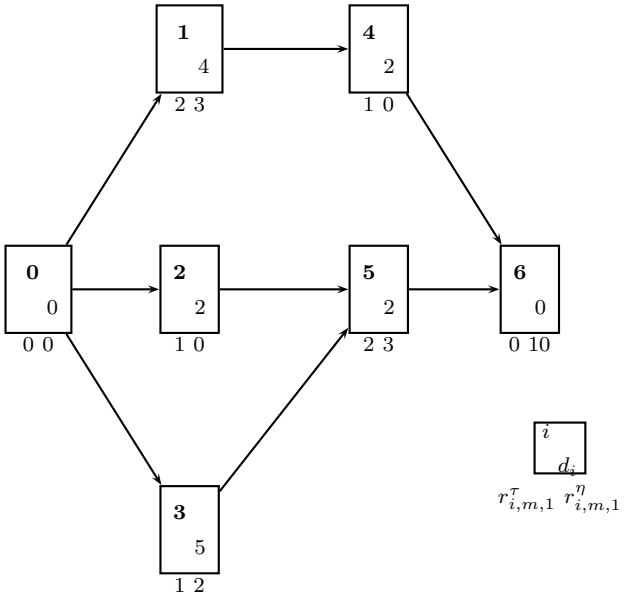
Regla de dominancia 7 *(Regla del orden de intercambio) Considere una actividad secuenciada, con tiempo de finalización menor o igual a cualquier tiempo de inicio que puede ser asignado al completar la secuencia parcial actual. Si un intercambio de orden en*

esta actividad junto con cualquiera de las actividades que finalicen en su tiempo de inicio, puede llevarse a cabo, a continuación, la secuencia parcial no necesita ser completada

Ejemplo 2.14. Consideremos el modo $M = (1, 1, 1, 2, 1, 1, 1)$ para el problema multi-modo propuesto en el ejemplo 1.4, con duración y recursos presentado en la siguiente tabla

j	0	1	2	3	4	5	6
m	1	1	1	2	1	1	1
d_{im}	0	4	2	5	2	2	0
r^{τ}_{im1}	0	2	1	1	1	2	0
r^{η}_{im1}	0	3	0	2	0	3	0

Cuadro 2.1: Datos del ejemplo PSP fijando el vector de modos M



Una cota superior de la duración del proyecto para el siguiente RCPSP es $T = 15$. Una secuencia factible $S = (0, 0, 0, 0, 8, 6, 10)$ para el problema anterior es mostrada en la figura

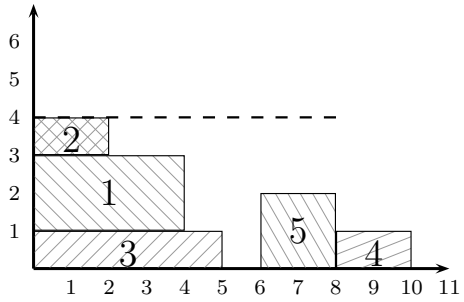
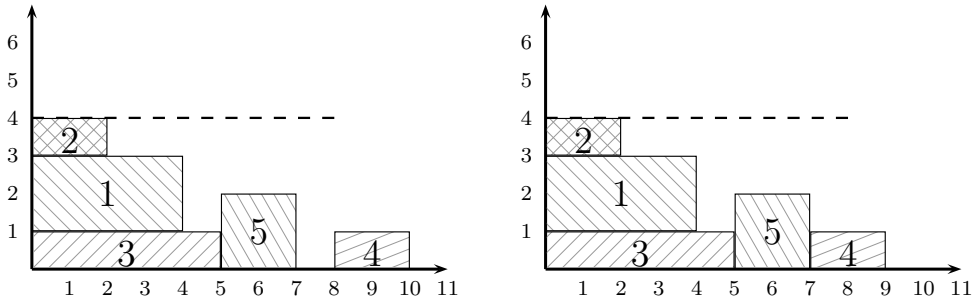


Figura 2.1: Secuencia factible.

Para representar una justificación local (justificación a la izquierda de 1-periodo) de la actividad 4 y 5, respectivamente, produciendo la secuencia semi-activa. $S' = (0, 0, 0, 0, 7, 5, 9)$ mostrado en la figura. Note que la secuencia intermedia $\hat{S} = (0, 0, 0, 0, 8, 5, 10)$



(a) Secuencia intermedia (b) Secuencia semi-activa

referente a la secuencia semi-activa $S = (0, 0, 0, 0, 7, 5, 9)$ mostrada en la figura. Claramente ninguna de las actividades puede ser localmente justificada a la izquierda. Sin embargo la actividad 4 puede ser justificada a la izquierda globalmente por la realización de una justificación a la izquierda de periodo 3. De aquí se obtiene la secuencia activa $S = (0, 0, 0, 0, 4, 5, 7)$, la cual es optima, mostrada en la figura

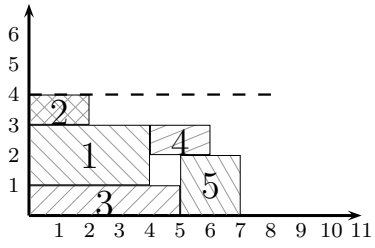


Figura 2.2: Secuencia óptima.

2.3. Métodos aproximados

El MRCPSP es un problema de la clase NP-Hard, el cual no es posible resolver todas las instancias de manera óptima. Sin embargo es posible obtener secuencias de buena calidad utilizando métodos aproximados. En este capitulo se describen algunos procedimientos heurísticos y metaheurísticas propuestos para solucionar el MRCPSP.

2.3.1. Reglas de prioridad

Las reglas de prioridad son para determinar el orden en el que se seleccionan las actividades durante el proceso de búsqueda heurístico [Kolish [30], Kelley [28]]. Existen numerosas reglas de prioridad propuestos en la literatura Conway [10].

Las reglas de prioridad generan una lista de actividades, que es una permutación de las actividades $\lambda = (i_0, i_1, \ldots, i_{n+1})$ las cuales quedan ordenadas según su prioridad, pero respetando las relaciones de precedencia. En particular $i_0 = 0$ y $i_{n+1} = n + 1$ para todo λ lista de actividades. Además está lista de actividades proporciona una secuencia, ya sea cuando se utiliza una sola regla de prioridad (los llamados métodos de una sola pasada), o cuando se utiliza una combinación de reglas de prioridad varias veces (los llamados métodos multi-pasada).

Definición 2.15. *Una regla de prioridad está conformada por tres componentes:*

1. *Una función $v : D_g \to \mathbb{R}^+$ la cual asigna a cada actividad $j \in D_g$ (conjunto de elegibles) un valor $v(j) \geq 0$.*
2. *Se determina el extremo (*mín *o* máx*) del conjunto $v(D_g)$. Es decir, se selecciona la actividad del conjunto de elegibles con el mínimo o máximo valor.*
3. *La regla que permite desempatar entre las actividades de igual valor.*

Entonces

$$j^* := \min_{j \in D_g} \{j : v(j) = \underset{i \in D_g}{ext}\, v(i)\} \qquad ext \in \{\text{mín}, \text{máx}\}$$

Una de las maneras más habituales de desempate es la de elegir, de entre las actividades con mismo valor, aquella con etiqueta menor. Se dice entonces que se está utilizando una regla *pura*. Las reglas de prioridad pueden ser clasificadas en 5 categorías según Lawrence [37], basadas en el tipo de información que se requiere para calcular la lista de prioridades. Estas cinco categorías son detalladas en Kolish [29]:

- Reglas de prioridad basada en actividades, las cuales consideran la información relacionada con las actividades, en el resto del proyecto.

1. Menor tiempo de procesamiento **SPT**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{mín}}\, d_i$$

2. Mayor tiempo de procesamiento **LPT**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{máx}} d_i$$

- Reglas de prioridad basadas en redes, las cuales solo consideran la información que está contenida en la red del proyecto.

1. Mayor número de sucesores inmediatos **MIS**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{máx}}\{|S_i|\}, \qquad S_i = \{k|(i,k) \in E\}$$

2. Mayor número total de sucesores **MTS**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{máx}}\{|\bar{S}_i|\}, \qquad \bar{S}_i = \{k|\text{existe un camino entre } k \text{ e } i\}$$

3. Mayor rango posicional de peso **GRPW**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{máx}}(d_i + \sum_{h \in S_i} d_h)$$

- Reglas basadas en camino crítico, las cuales están basadas en hallar el camino crítico hacia adelante y hacia atrás.

1. Tiempo de inicio más temprano **ES**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{mín}}\, ES_i$$

2. Tiempo de finalización más temprano **EF**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{mín}}\, EF_i$$

3. Tiempo de inicio más tardío **LS**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{mín}}\, LS_i$$

4. Tiempo de finalización más tardío **LF**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{mín}}\, LF_i$$

5. Holgura minima **MSLK**

$$\underset{i \in D_g}{ext}\, v(i) = \underset{i \in D_g}{\text{mín}}\, LS_i - ES_i$$

- Reglas de prioridad basadas en recursos, las cuales consideran los recursos usados en las diferentes actividades.

  1. Mayor demanda de recursos **GRD**

$$\underset{i\in D_g}{ext}\, v(i) = \underset{i\in D_g}{\text{máx}} d_i \sum_{k\in\mathcal{R}} r_{ik}$$

- Reglas de prioridad compuestas, las cuales se obtienen sumando los pesos de los valores de la prioridad obtenida por las reglas de prioridad de las 3 categorías anteriores.

  1. Utilización de recursos y la prioridad ponderada **WRUP**

$$\underset{i\in D_g}{ext}\, v(i) = \underset{i\in D_g}{\text{mín}}\, w_1|S_i| + w_2 \sum_{k\in\mathcal{R}} \frac{r_{ik}}{\mathcal{R}_k}$$

  2. Método de secuenciación de recursos **RSM**

$$\underset{i\in D_g}{ext}\, v(i) = \underset{i\in D_g}{\text{mín}}\ \underset{(j,i)\in AP}{\text{máx}} \{0, t_g + d_i - LS_j\}, \qquad AP = \{(j,i)|j,i \in D_g, i \neq j\}$$

2.3.2. Esquema de generación de secuencias, SGS

Los esquemas generadores de secuencias o esquemas de secuenciación SGS (Schedule generation scheme) son una parte indispensable en muchos procedimientos para el MRCPSP. Los SGS son procedimientos iterativos que permiten fijar el tiempo de comienzo s_i de cada actividad i del proyecto, hasta que una secuencia factible es alcanzada.

Se distinguen dos esquemas de secuenciación: el esquema serial y el esquema paralelo. Los dos esquemas generan secuencias factibles extendiendo por etapas una secuencia parcial. En cada etapa el SGS construye el conjunto de todas las actividades elegibles, también llamado conjunto de decisión. Únicamente se detalla el esquema en serie, puesto que este trabaja en el incremento temporal.

2.3.2.1. Esquema en serie

Propuesto por Kelley [28] para el RCPSP, generalizado para el MRCPSP por Lova [40]. Serie consiste en $g = 1, \ldots, n$ etapas. En cada una de las cuales se selecciona una actividad de acuerdo con una regla de prioridad y se le asigna un modo con respecto a la regla de selección de modos de tal forma que cumpla las restricciones de las relaciones de precedencia y sin sobrepasar la disponibilidad de recursos. Existen dos conjuntos disjuntos, asociados a cada etapa g: el conjunto de actividades secuenciadas Sec_g el cual comprende las actividades que ya han sido secuenciadas, mientras que el conjunto decisión D_g comprende todas las

actividades que son elegibles para secuenciar de acuerdo al orden establecido por las reglas de prioridad y la regla de selección de modos.

En adición a las reglas de prioridad mostradas anteriormente, la siguiente regla puede ser usada para la selección de las actividades para el MRCPSP Kolisch [35]:

- Mayor consumo de recursos **GRC**

$$\underset{i\in D_g}{ext}\, v(i) = \max_{i\in D_g} \min_{m\in\mathcal{M}_j} \sum_{\ell\in\mathcal{R}^\eta} \frac{r^\eta_{j,m,\ell}}{\mathcal{R}^\eta_\ell - \sum\limits_{j\in\mathcal{A}_m} r^\eta_{j,m_j,\ell}}$$

La regla GRC selecciona la mayor actividad elegible tal que el consumo de recurso sea menor (en términos de la disponibilidad de recursos residuales). La idea básica es que las actividades sean secuenciadas lo más rápido posible en el curso del algoritmo tal que exista una probabilidad alta que el modo (barato) de ejecución se pueda seleccionar para conseguir un tiempo de inicio factible para la actividad j.

En Heilmann [24] propone la siguiente regla de selección de modos:

- Mínimo consumo de recurso **MRC**

$$\underset{m_j\in\mathcal{M}_j}{ext}\, u(m_j) = \min_{m\in\mathcal{M}_j} \sum_{\ell\in\mathcal{R}^\eta} \frac{r^\eta_{j,m,\ell}}{\mathcal{R}^\eta_\ell - \sum\limits_{j\in\mathcal{A}_m} r^\eta_{j,m_j,\ell}}$$

Entonces

$$m^*_j := \min_{m_j\in\mathcal{M}_j} \{m_j : u(m_j) = \underset{m'_j\in\mathcal{M}_j}{ext}\, u(m'_j)\}$$

Para dar una descripción formal del esquema generacional de secuencias en serie, introducimos algunas notaciones importantes: en cada paso $g : 1 \dots n$ sea:

Representación	Notación
Sec_g	Conjunto de actividades secuenciadas
D_g	Conjunto de decisión.
	$D_g := \{j \in V \setminus Sec_g : \mathcal{P}_j \subseteq Sec_g\}$
$\mathcal{A}_m$	Conjunto de actividades a las cuales se les ha asignado un modo
$(\mathcal{R}^\eta_\ell)^{res}$	Disponibilidad residual del recurso no-renovable
	$(\mathcal{R}^\eta_\ell)^{res} = \mathcal{R}^\eta_\ell - \sum\limits_{j\in\mathcal{A}_m} r^\eta_{j,m_j,\ell}$
$(r^\eta_{j,m})^{avg}$	El consumo relativo de los recursos no-renovables por la actividad j en el modo m
	$(r^\eta_{j,m})^{avg} = \sum\limits_{\ell\in\mathcal{R}^\eta} \frac{r^\eta_{j,m,\ell}}{(\mathcal{R}^\eta_\ell)^{res}}$

El esquema generador de secuencias en serie puede describirse utilizando el algoritmo :

Algoritmo 6: SGS Serial

1. Inicialización: $s_0 = f_0 = 0$, $Sec_0 = \{0\}$. $\mathcal{A}_m = \emptyset$, $\tilde{\mathcal{M}} = (1)$, $(\mathcal{R}_\ell^\eta)^{res} = \mathcal{R}_\ell^\eta \quad (\ell \in \mathcal{R}^\eta)$;
2. Para $g = 1$ hasta n hacer ;
2.1. Calcular D_g, F_g, $\tilde{\mathcal{R}}_k^\tau(S, M, t) \quad (k \in \mathcal{R}^\tau, \ \ t \in F_g)$;
2.2. Para todo $j \in D_g$ hacer ;
2.2.1. Calcular $v(j)$;
2.2.2. Elegir $j^* := \text{mín}\{j \in D_g : v(j) = \underset{h \in D_g}{ext}\ v(h)\}$;
2.3. Para todo $m_{j^*} \in \mathcal{M}_{j^*}$;
2.3.1. Calcular $u(m_{j^*})$;
2.3.2. Elegir $m_{j^*}^* := \text{mín}\{m_{j^*} \in \mathcal{M}_{j^*} : u(m_{j^*}) = \underset{m'_{j^*} \in \mathcal{M}_{j^*}}{ext} u(m'_{j^*})\}$;
2.3. $es_{j^*} = \underset{h \in \mathcal{P}_{j^*}}{\text{máx}}\ \{f_h\}$;
2.4. $s_j = \text{mín}\{t : \ t \geq es_{j^*};\ t \in F_g;\ r^\eta_{j^*,m^*_{j^*},k} \leq \tilde{R}^\tau_k(\kappa)\ \forall k \in \mathcal{R}^\tau,\ \kappa \in [t, t + d_{j^*,m^*_{j^*}}) \cap F_g\}$;
2.5 $f_{j^*} = s_{j^*} + d_{j^*,m^*_{j^*}}$;
2.6. $Sec_g = Sec_{g-1} \cup \{j\}$;
3. $f_n = \underset{h \in \mathcal{P}_h}{\text{máx}}\{f_h\}$

Precisamente, el SGS serial trabaja de la siguiente forma: en el inicio a ninguna de las actividades se le ha asignado un modo. Consecuentemente el conjunto $\mathcal{A}_m$ es vació y la disponibilidad residual de todos los recursos no renovables es igual a la disponibilidad inicial, respectivamente. En cada paso g con $1 \leq g \leq n$, toda actividad que no se le ha asignado un modo es calculado y se selecciona la actividad de acuerdo a la regla de prioridad. A esta actividad se ha asignado un modo de acuerdo a la regla de selección de modo, después ya seleccionada la actividad y su modo de ejecución, se calcula el tiempo de inicio más temprano y luego el tiempo de inicio tal que cumpla las restricciones de recursos renovables. Esta actividad se secuencia formando una secuencia parcial.

Ejemplo 2.16. Apliquemos el SGS serie al ejemplo siguiente

j	0	1	2	3	4	5	6
m	1	1 2	1	1 2	1	1 2	1
d_{im}	0	4 6	2	3 5	2	2 4	0
r^τ_{im1}	0	2 1	1	3 1	1	2 1	0
r^η_{im1}	0	3 1	0	4 2	0	3 2	0

						j	0	1			2	3		4	5		6			
n	A_m	$(\mathcal{R}_\ell^\eta)^{res}$	Sec_g	D_g	F_g	m	1	1	2	1	1	2	1	1	2	1	GRC	j^*	m_{j*}^*
0	∅	8	0	1,2,3	0			0.375	0.125	0	0.5	0.25					0.25	3	2
1	3	6	0,3	1,2	0,5			0.5	0,167	0							0,167	1	2
2	1,3	5	0,1,3	2,4	0,5,6					0			0				0	2	1
3	1,2,3	5	0,1,2,3	4,5	0,2,5,6								0	0,6	0,4		0,4	5	2
4	1,2,3,5	3	0,1,2,3,5	4	0,2,5,6,9								0		0		0	4	1

El SGS serie produce un vector de modos factibles $M = (1, 2, 1, 2, 1, 2, 1)$ y una secuencia factible $S = (0, 0, 0, 0, 6, 5, 9)$ con un makespan de 9 periodos.

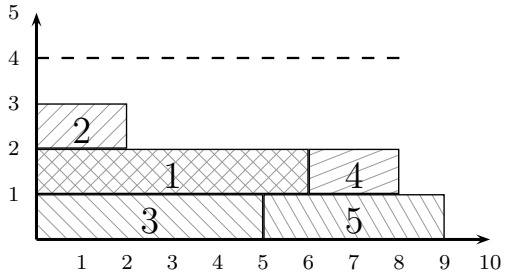


Figura 2.3: Secuencia generada por SGS serial

2.3.3. Métodos metaheurísticos

En esta sección se describen algunos trabajos recientes para resolver el MRCPSP, utilizando procedimientos metaheurísticos, los cuales son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos con un alto desempeño. Dichos trabajos se toman como referentes dado que se destacan por sus buenos resultados computacionales. Weglarz [61] proporciona un estado del arte muy completo de los trabajos más importantes que resuelven el MRCPSP, así como las direcciones para las investigaciones futuras.

Kolish y Drexl [31] aplican un procedimiento de búsqueda local, donde una solución se representa por medio de dos listas, una es la asignación de modos y la otra es la lista de los tiempos de finalización de cada actividad. El método propuesto consiste en tres fases. En primer lugar, en la fase de construcción, se genera un modo de asignación inicial y luego, si se ajustan a las restricciones de recursos no renovables, una heurística rápida se utiliza para el RCPSP resultante. En la segunda fase, un número determinado de iteraciones de una búsqueda local se lleva acabo. Por último, se realiza una fase de intensificación. En esta fase, una secuencia con un valor dado por la función objetivo es

mejorado buscando en la base de la mejor asignación de los modos obtenidos durante la fase anterior. El funcionamiento de este algoritmo se comprueba sobre la base de experimentos computacionales de dos conjunto de instancias con 10 y 30 actividades, generadas usando ProGen en Kolish [35]. Cada conjunto se compone de 640 casos, en los que en cada caso se consideran dos recursos no renovables y dos renovables, así con tres modos por actividad. Los resultados obtenidos por el algoritmo propuesto se comparan con los resultados obtenidos por un $B\&B$ truncado en Talbot [58] y STOCOM por Drexcel y Grunewald [13]. La heurística propuesta es el único de los tres enfoques probados que encuentra soluciones viables para todas las instancias compuestas por 10-30 actividades.

Ozdamar [47] propone dos versiones de los algoritmos genéticos: puro (PGA) e híbridos (HGA). Una solución en el HGA está representada por dos listas: una lista de asignación de modos y una lista de reglas de prioridad. La posición $i-$ésima en la segunda lista indica una regla de prioridad que es usada para la elección de la $i-$ésima secuencia. Las siguientes reglas de prioridad que se utilizan son: la holgura total minima (MIN SLK), MIN LFT, el tiempo de procesamiento más corto (SP), RAND, utilización de recursos y prioridad ponderada (WRUP), tiempo de inicio más tardio (MIN LS), el tiempo de inicio más temprano (MIN ES). Como una regla de decodificación, secuenciación hacia delante y hacia atrás (ver, por ejemplo , Li y Willis [38]) empleando SGS paralelo. Se proponen el cruce de dos puntos y el uniforme. Un operador de mutación que cambia aleatoriamente una posición en la lista de reglas de prioridad. En el PGA una solución es representada por una lista de asignación de modos y una lista de actividades. SGS en serie (véase, por ejemplo, Kelley [28]; Kolish [30]) se emplea como regla de decodificación. Un cruce lineal de dos puntos se utiliza para generar soluciones descendientes. Si una lista de actividades obtenida no satisface las relaciones de precedencia, se ejecuta un procedimiento de reparación. Un operador de mutación para el PGA cambia aleatoriamente una posición en la lista de modos y mueve en saltos parejas de actividades de la lista de actividades, pero no se permite cambiar las que no cumplan las relaciones de precedencia. En ambos casos, los algoritmos miran si la descendencia no es factible con respecto a los recursos no renovables entonces no se evalúa. En otras palabras, sólo se permiten soluciones con las asignaciones de modos que sean factibles son respecto a los recursos no renovables. El desempeño de los enfoque propuestos se evaluó utilizando 536 instancias con 10 actividades, 3 modos por actividad, 2 recursos renovables y 2 recursos no renovables, así como 32 instancias con 90 actividades, de dos modos por actividad, dos recursos renovables y dos no renovables. Todos los casos son generados por ProGen [35] y disponible por internet en la librería PSPLIB [34]. Los resultados obtenidos se comparan con los reportados por Kolisch y Drexl [31] y muestran que el SAG supera a todos los otros algoritmos probados en ese experimento.

Hartmann [20] propone un algoritmo genético. Antes de ejecutar el algoritmo, se aplica el preprocesamiento con el fin de reducir el espacio de búsqueda mediante la adaptación de los

datos del proyecto. Una solución está representada por una lista de actividades que satisface las relaciones de precedencia y una lista de asignación de modos. Esta representación permite generar secuencias factibles con respecto a los recursos no renovables. En tal caso, una función de penalización se utiliza para calcular la aptitud de una solución factible. Se genera la población inicial y se cruzan los individuos usando un operador de cruce (se utilizan dos puntos de corte diferentes: uno para la lista de actividades y otro para la lista de asignación de modos). El operador de mutación cambia dos actividades adyacentes en la lista de actividades siempre que satisfaga las relaciones de precedencia y además cambia al azar un modo en la lista de asignación de modos. El operador de selección es el de ranking. una secuencia se construye aplicando el SGS serie, y una búsqueda local de un solo paso o de varios pasos basándose en una justificación a la izquierda cambiando el modo. A continuación si se mejora la secuencia, una regla de codificación es aplicada para transformar inversamente la secuencia sobre su lista de actividades y la lista de asignación de modos. Esta operación, denominada herencia, por desgracia, no mejora de manera significativa los resultados obtenidos. El autor también muestra que el enfoque de repetición y el modelo de isla de GA, así como otros operadores de selección, no mejoran el rendimiento del enfoque propuesto. Conjunto de datos con 10, 12, 14, 16, 18, 20, 30 y las actividades de PSPLIB se utilizan en un experimento computacional en el que se eligen experimentalmente los mejores ajustes del algoritmo genético desarrollado, y el rendimiento de la versión final del GA se compara con el rendimiento de otros enfoques, a saber búsqueda local de Kolisch y Drexl [31], el algoritmo genético de Ozdamar [47], el recocido simulado por Bouleimen y Lecocq [6], y el $B\&B$ truncado por Hartmann y Drexl [22]. Los resultados obtenidos muestran que el nuevo algoritmo genético es mejor que otras heurísticas. La única excepción es el $B\&B$ truncado es ligeramente mejor para los casos con un pequeño número de actividades.

Józefowska [26] propone dos versiones de recocido simulado: con y sin función de penalización, donde una solución se representa por una lista de asignación de modos y una lista de actividades. El SGS serie se utiliza como regla de decodificación. Una pena para la violación de las limitaciones de recurso no renovable es añadir a la a la cota superior del makespan en la función de penalización. Soluciones vecinas son generadas por un cambio aleatorio de una actividad elegida y/o cambiando al azar el modo elegido. El proceso de búsqueda es controlado por el sistema de refrigeración adaptable. El preprocesamiento se aplica al comienzo de los algoritmos. El mejoramiento de ambos enfoque es examinado durante un experimento computacional en donde los conjuntos de datos a partir de la librería PSPLIB con 10, 12, 14, 16, 18, 20 y 30 actividades de 3 modos por actividad, dos recursos renovables y dos recursos no renovables. Los resultados obtenidos indican que la versión uno con función de penalización es la mejor y comparable con GA de Hartmann [20].

El primer enfoque de búsqueda tabú aplicado al MRCPSP se presenta en (Nonobe e Ibaraki [46]). En esta implementación de una solución es representada por una lista de

actividades y una lista de asignación de modos. Soluciones vecinas son generadas, ya sea por un cambio en la lista de asignación de modos, o cambiando alguna actividad en la lista de actividades. La secuencia se genera mediante la aplicación de un esquema propuesto por los autores, llamado CONSTRUCT, pero nuevas observaciones muestran que en algunos casos una solución óptima no se puede obtener por este procedimiento. Los resultados computacionales son sólo reportados para el conjunto de datos con 30 actividades de PSPLIB, pero no son comparados con cualquier otro enfoque presentado.

Alcaraz [2] propone otro algoritmo genético. Al igual que en los otros enfoques, se aplica el preprocesamiento antes que el algoritmo se ejecute. La solución se codifica utilizando una lista de actividades, una lista de asignación de modos, y un gen(bit) adicional (forward/backward) que indica la dirección en la que el SGS en serie genera la secuencia: hacia delante o hacia atrás. La función de aptitud se calcula ya sea simplemente como el makespan, si la secuencia que se obtiene es factible, o $C_{\text{máx}} + MFC_{\text{máx}} - CP_{\text{mín}} + PF$ si la secuencia no es factible, donde $C_{\text{máx}}$ es el makespan de la solución actual, $MFC_{\text{máx}}$ es el máximo makespan posible de la población actual, $CP_{\text{mín}}$ es la longitud del camino critico, y $PF = \sum_{l=1}^{N} \left(\text{mín}\{R_l^{\eta} - \sum_{j=1}^{n} r_{jml}^{\eta}\} \right)$ para $l = 1, 2, \ldots, N$ es una penalización por violar las restricciones de recursos no renovables. Se demostró que esta función de aptitud es mejor que la propuesta por Hartmann [20]. Se propone un operador de cruce de dos puntos hacia delante-hacia atrás en el que se construye la descendencia ya sea de la cabeza o la cola en la dependiendo del valor del gen (forward/backward) de los padres. El operador de mutación se compone de dos fases: en la primera de ellas, las actividades se reordenan en la lista de actividades con una probabilidad dada en la forma en que la lista de actividades después de esta operación es aún de precedencia factible, y en la segunda fase, los modos de la lista de asignación de modos de cambian con una probabilidad determinada. El gen hacia adelante/ hacia atrás puede ser cambiado en la primera fase. Por último, un procedimiento de sustitución aleatoria se aplica para remplazar con soluciones con una probabilidad dada, por otras soluciones generadas aleatoriamente. El experimento computacional se lleva a cabo utilizando conjunto de datos de PSPLIB con instancias de 10, 12, 14, 16, 18, 20, y 30 actividades. La comparación con la búsqueda local por Kolisch y Drexl [31] se hace sólo para el conjunto de datos con 10 actividades después de generar 6000 soluciones, y muestran que la propuesta GA supera a la búsqueda local y GA de Ozdamar, pero funciona un poco peor que el GA de Hartmann. Se presentan más resultados de la comparación con el recocido simulado por Józefowska [26]. En este caso, todos los conjuntos de datos, excepto el que tiene 30 actividades, y un criterio de parada, tanto para los algoritmos analizados se fijan 5000 soluciones generadas. Los resultados presentados muestran que la propuesta supera el SA y GA.

Chyu [9] propone un algoritmo de optimización de colonia de hormigas híbrido (H-ACO) en el que tanto el mecanismo de construcción de soluciones de $B\&B$, y los algoritmos de

optimización basada en colonia de hormigas (ACO) se híbrida. En primer lugar, $B\&B$ se utiliza para encontrar un conjunto de listas de asignación de modos. Para cada asignación de modos factible se aplica el siguiente procedimiento. En primer lugar, la RPC se utiliza para calcular la longitud de las secuencias cuando no cumple las restricciones de recursos. En segundo lugar, una regla disyuntiva se aplica para añadir algunos arcos AoN con el fin de eliminar los conflictos de recursos. En tercer lugar, un nuevo makespan del proyecto se calcula para la nueva red del proyecto. A continuación, un número determinado de listas de modo con los mejores valores posibles (el makespan obtenido durante el tercer paso del procedimiento antes mencionado) son elegidos para la segunda fase de la H-ACO donde se aplica el algoritmo ACO con el mejoramiento forward-backward a cada lista de modos. Finalmente, la mejor secuencia obtenida por ACo para cada elección de modos es comparada, y el mejor se elige como una solución de la instancia del problema considerado. H-ACO se compara con SA por Józefowska [26] utilizando los conjuntos de datos estándar de PSPLIB. Los resultados obtenidos muestran que la H-ACO supera SA.

2.3.4. Librería de pruebas PSPLIB

La librería PSPLIB cuenta con una gran variedad de problemas para diferentes variantes del RCPSP y MRCPSP, así como soluciones óptimas y heurísticas. Los conjuntos de datos pueden ser utilizados para la evaluación de las soluciones del RCPSP y del MRCPSP. Las instancias se generan con ProGen (Kolish y Sprecher [29]) el cual las genera de forma aleatoria. Los investigadores pueden descargar las instancias para evaluar sus algoritmos, y pueden enviar sus resultados para ser agregados a la librería.

La librería en sí, es decir, los tipos de modelos que representan, los detalles de la generación de las instancias, el diseño experimental para la generación de los problemas, parámetros del problema, etc, se pueden encontrar en (Kolish y Sprecher [34]). En Kolish y Sprecher [32] se publica un informe sobre los casos adicionales que se han añadido a la PSPLIB, que se encuentra disponible en internet en *http://www.om-db.wi.tum.de/psplib/*. Esta librería contiene las instancias referentes al MRCPSP: J10, J12, J14, J16, J18, J20 y J30. Existen otros conjuntos de problemas conocidos como las instancias Boctor (Boctor50 y Boctor100). Estas instancias muestran algunas deficiencias, dada la evolución reciente en el desarrollo de los procedimientos metaheurísticos. Por tanto, un nuevo conjunto de datos se propone en Van Peteghem [48], los cuales se pueden descargar desde la página web *http://www.projectmanagement.ugent.be*. Este conjunto se denota por J50 y J100.

- J10 → 10 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.
- J12 → 12 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.
- J14 → 14 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.

- J16 → 16 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.
- J18 → 18 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.
- J20 → 20 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.
- J30 → 30 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.
- J50 → 50 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.
- J100 → 100 actividades, 3 modos por actividad, 2 recursos renovables y 2 no renovables.

Capítulo 3
Algoritmos meméticos

En esta capítulo se presenta un algoritmo memético (MA) para resolver el MRCPSP. Diseñado por Moscato [45], un MA se define como una estrategia metaheurísticas que combina sinérgicamente conceptos tomados de los algoritmos evolutivos (EA) y de búsqueda local (LS). En los algoritmos meméticos se utiliza el término de agentes en lugar de individuos, puesto que el término agente implica la existencia de un comportamiento activo en el que aprende y transmite información. Para una introducción sobre MA se hace referencia a Moscato.

3.1. Esquema básico

Iniciamos el MA seleccionando una población inicial POB, compuesta por Pob agentes, ordenados de acuerdo a su evaluación en orden no decreciente. Cada agente se denota por $I = (\lambda, \mu)$ donde μ es un vector de modos y λ una lista de actividades. Aplicamos una búsqueda local a todos los miembros de la población, con el objetivo de mejorar su evaluación y aplicamos el operador de mutación.

La POB se particiona en pares de agentes. Para cada par de agentes resultantes, se aplica el o los operadores de recombinación, obteniendo así dos nuevos agentes a los cuales se implementa la búsqueda local para mejorar su evaluación. Posteriormente se aplica el operador de mutación y de nuevo se aplica búsqueda local. Estos agentes se adhieren a POB. A continuación se aplica el operador de selección para reducir POB a su tamaño original Pob y obtener la siguiente generación a la que volvemos aplicar el mismo procedimiento. Este proceso se repite para un número preestablecido de las generaciones que se denota por N o alternativamente hasta un limite de tiempo sea alcanzado.

Algoritmo 7: Algoritmo memético

Función: $Memetico(Pob, N)$

Generar POB ;
for *cada* $I \in POB$ **do**
  $I := VND(I, 2)$;
  $I := Mutacion(I)$;
  $POB = POB$;
Repetir
  for $n = 1$ *hasta* K *recombinaciones* **do**
    Seleccionar dos agentes I_M e I_F;
    Producir dos nuevos agentes I_D e I_G por recombinación ;
    $I^D = VND(I^D, 2)$ y $I^G = VND(I^G, 2)$;
    $I^D := Mutacion(I^D)$ y $I^G := Mutacion(I^G)$;
    $I^D = VND(I^D, 2)$ y $I^G = VND(I^G, 2)$;
    Adherir I_D e I_G a POB;
  POB= {*mejores Pob agentes de* POB *por selección*};
  $N++$;
Hasta *Se cumple criterio de terminación*;
Return POB

3.2. Representación

Los MA igualmente que los AG operan sobre una representación codificada de las soluciones. Una adecuada codificación de las soluciones es crucial para el éxito del algoritmo y es la base de la que dependen los demás operadores, que deben ser diseñados para que utilicen, de manera eficiente, las propiedades de cada representación.

Kolish y Hartmann [32] distinguen cinco diferentes tipos de representación de secuencias. Las más utilizadas son la lista de actividades (AL) y la representación random-Key (RK).

En este trabajo, se usará la representación (AL) de una secuencia. Cada individuo es representado por una doble lista: una lista de actividades AL y un vector de modos. En la AL cada actividad es ubicada después de sus predecesores, por lo tanto, es de precedencia factible. El vector de modos representa la asignación de modos de cada actividad. Esta representación es usada por varios autores tales como Hartmann [20], Józefowska [26] y Bouleimen y Lecocq [6] entre otros. Alcaraz [2] adiciona a la representación AL un gen forward/backward (f/b) que indica la dirección en la cual el esquema de generación en serie construye la secuencia, sin embargo, en este trabajo la dirección en la que se construye la secuencia siempre es forward. El objetivo de utilizar el gen (f/b) es simplemente para la diversidad en el operador de cruce.

Se utilizará la representación de un agente $I = (\lambda, \mu, f/b)$, donde λ es una lista de actividades ordenadas según la relación de precedencia factible, μ es un vector de modos para las correspondientes actividades y el gen (f/b) es forward/backward. Para cada actividad $j \in V$, tomamos uno de sus modos $\mu(j) \in \mathcal{M}_j$, entonces

$$I = \begin{pmatrix} j_1 & \cdots & j_k & \cdots & j_n & (f/b) \\ \mu(j_1) & \cdots & \mu(j_k) & \cdots & \mu(j_n) & \end{pmatrix}$$

donde cada agente I es relacionado con una determinada secuencia S, la cual se obtiene a partir de la fijación de los modos μ, se busca el RCPSP asociado a dicho vector de modos y se aplica el esquema de generación en serie. La adaptación del SGS del RCPSP para transformar lista de actividades se muestra en el algoritmo 8.

Algoritmo 8: SGS serial

Función: $SGS(I)$

Inicialización: $s_0 = f_0 = 0$, $Sec_0 = \{0\}$.;

Para $g = 1$ *hasta* n **to hacer**

- Calcular F_g, $\tilde{R}_k(t) \quad (k \in \mathcal{R}; \quad t \in F_g)$;
- Seleccionar $j = j_g$;
- $es_j = \max\limits_{h \in \mathcal{P}_j} \{f_h\}$;
- $s_j = \min\{t : \ t \geq es_j; \ t \in F_g; \ r_{ik} \leq \tilde{R}_k(\tau) \ \forall k \in \mathcal{R}, \ \tau \in [t, t + p_j) \cap F_g\}$;
- $f_j = s_j + p_j$;
- $Sec_g = Sec_{g-1} \cup \{j\}$;

$f_{n+1} = \max\limits_{h \in \mathcal{P}_h} \{f_h\}$;

Return S

3.3. Población inicial

Una parte fundamental del MA es la generación de la población inicial, POB, de tamaño Pob, a partir de la cual se originarán las siguientes generaciones. Para crear la población inicial se ejecuta el algoritmo 9.

Algoritmo 9: Población inicial

Función: $PoblacionInicial(Pob)$
$POB = \emptyset$;
Mientras $|POB| < Pob$ **hacer**
  $I = GenerarAgente(I)$;
  $POB = POB \cup \{I\}$
Return POB

Cada agente $I = (\lambda, \mu, (f/b))$ es calculado de la siguiente forma. Primero, cada actividad $j \in V$, se le asigna un modo $\mu_j \in \mathcal{M}_j$ de manera aleatoria, dando como resultado un vector de modo μ. Verificamos si μ es un vector de modos factibles. Si no es así, se aplica un procedimiento de búsqueda local, previamente usado por Hartmann [20] en su algoritmo genético, para transformar el vector de modos en un vector de modos factibles.

Para generar la lista de actividades de precedencia factible λ, se fijan los modos de las actividades con respecto a μ. Entonces, se calculan Pob soluciones con la regla de prioridad LST, y $Pob/4$ con las reglas SPT, MSLK, RSM, LFT. Luego el gen (f/b) es seleccionado de manera aleatoria. La población inicial se forma con las Pob mejores, así obtenidas, de este modo se aúnan calidad, factibilidad y diversidad. El algoritmo 10 muestra la forma en la que se construyen los agentes.

Algoritmo 10: Generador de Agentes

Función: $GenerarAgente(I)$
Para $j = 1$ *hasta* n **hacer**
  Seleccionar $\mu_j \in M_j$ aleatoriamente
Definir $\mu := (\mu_1, \mu_2, \ldots, \mu_n)$;
Si μ *es factible* **entonces**
  $GenerarLista(\lambda)$
Si no
  $\mu := BusquedaLocal(\mu)$;
  $GenerarLista(\lambda)$
$Generar(f/b)$;
Return $I = (\lambda, \mu, f/b)$

Para generar cada una de las listas de actividades, se utiliza el algoritmo (11). Observe que, solo se debe cambiar la regla de prioridad para calcular el resto de agentes.

Algoritmo 11: Generador de la lista LST

Función: $GenerarListaLST(\mu)$

$Sec_0 := \{0\}$, $\lambda[0] = 0$;

Para $g = 1$ *hasta* n **hacer**

Definir $D_g := \{j \in V : \mathcal{P}_j \subseteq Sec_{g-1}\}$;

seleccionar $j^* = \text{mín}\{j \in D_g : LS_j = \text{mín}\{LS_i : i \in D_g\}\}$;

$Sec_g = Sec_{g-1} \cup \{j^*\}$;

$\lambda[g] = j^*$

$\lambda[n+1] = n+1$;

Return λ

Algoritmo 12: Generador de vectores de modos factibles

Función: $BusquedaLocal(\mu)$

Repetir

Seleccionar aleatoriamente $j \in V$ con $\mathcal{M}_j > 1$;

seleccionamos aleatoriamente $m_j \in \mathcal{M}_j \setminus \{\mu_j\}$;

definimos $\mu' = (\mu_1, \ldots, m_j, \ldots, \mu_n)$;

calcular $L^\eta(\mu)$ y $L^\eta(\mu')$;

Si $L^\eta(\mu') \leq L^\eta(\mu)$ **entonces**

$\mu = \mu'$

Si no

ir a la selección aleatoria de la actividad.

Hasta n *veces o* $L^\eta(\mu) = 0$;

Return μ

3.4. Función de evaluación

Debido a la complejidad del problema, el hecho de generar una secuencia que sea factible, ya es en sí mismo, en un problema complejo. Por ello, los algoritmos deben admitir soluciones no factibles y por tanto las funciones objetivo incluyen factores de penalización.

Un aspecto importante en los algoritmos genéticos y meméticos es la determinación de la evaluación o fortaleza de los agentes en la población, esta función evalúa la calidad de las soluciones. Para obtener la evaluación de un agente se aplica el esquema de generación en serie, obteniendo una secuencia S factible. Si resulta imposible transformar una lista de actividades en una secuencia factible, se permite que alguna actividad $i \in V$ sea secuenciada teniendo en cuenta sólo su tiempo de inicio más temprano y las restricciones de recursos, con lo cual $S_i > LS_i$. Dada la naturaleza de nuestro problema, el fitness de los agentes será

el tiempo total de ejecución del proyecto (makespan), y cuanto más pequeño sea este valor, más fuerte será el agente. Concretamente, el fitness de un agente es:

$$F(I) = \begin{cases} S_{n+1}, & \text{si } S \text{ es factible.} \\ \text{máx}\{S_{n+1}, F(I^*)\} + \sum_{i \in V} (S_i - LS_i)^+ & \text{en otro caso.} \end{cases} \tag{3.1}$$

Donde $F(I^*)$ es el fitness de la mejor secuencia S factible encontrada hasta ahora, correspondiente al individuo I. El algoritmo (13), muestra la forma de calcular la evaluación de un agente.

Algoritmo 13: Fitness de los agentes

Función: $Fitness(I)$
$SGS(I) = S$;
Si S *es factible* **entonces**
  $Fitness(I) = S_{n+1}$
Si no
  $Fitness(I) = \text{máx}\{S_{n+1}, Fitness(I^*)\} + \sum_{i \in V} (S_i - LS_i)^+$
Return $Fitness(I)$

A continuación se realiza una revisión del estado del arte de algunas funciones de evaluación, que se caracterizan por su desempeño en los algoritmos. Hartmann [20] define la siguiente función de evaluación:

$$F(I) = \begin{cases} mak(I), & \text{si } L^\eta(\mu) = 0. \\ T + L^\eta(\mu) & \text{en otro caso.} \end{cases} \tag{3.2}$$

donde $mak(I)$ es el makespan del individuo I y T es una cota superior en el makespan del proyecto dado por la suma de las duraciones máximas de las actividades. $L^\eta(\mu)$ es el exceso de recurso no-renovable definido por

$$L^\eta(\mu) = \sum_{k \in \mathcal{R}^\eta} |\,\text{mín}\{0, \mathcal{R}_k^\eta - \sum_{j=1}^{n} r^\eta_{j\mu(j)k}\}|$$

Sin embargo, un individuo representa una secuencia factible si, y solo si, $L^\eta(\mu) = 0$.

Al hacer un estudio de la función de evaluación de Hartmann 3.2, Alcaraz [2] hace las siguientes observaciones:

- El valor de la función de evaluación de un individuo no factible depende únicamente del exceso de requerimientos con respecto a los recursos no-renovables, sin embargo la duración de la secuencia no se tiene en cuenta. Por tanto, dos individuos con el mismo

exceso en el uso de recursos pero con diferentes duraciones tendrán el mismo valor de la función objetivo.

- El limite superior T es mucho más alto que el valor de la duración de cualquier proyecto factible de tal manera que la penalización es tan alta que los individuos no factibles tienen muy poca probabilidad de ser seleccionados para la siguiente generación.

para superar estas debilidades Alcaraz [2] propuso la siguiente función de evaluación:

$$F(I) = \begin{cases} mak(I), & \text{si } L^{\eta}(\mu) = 0. \\ \text{máx}\_mak(P) + mak(I) - \text{mín}\_CP + L^{\eta}(\mu) & \text{en otro caso.} \end{cases} \tag{3.3}$$

donde $mak(I)$ es el makespan del individuo I y máx $\_mak(P)$ la mayor duración de los individuos factibles de la población actual. Dicha duración se utiliza como cota superior de la duración de todos los individuos de la población actual que son factibles. $mak(I) - \text{mín}\_CP$ es el incremento sobre la duración del proyecto determinado por la mínima ruta crítica del proyecto y $L^{\eta}(\mu)$ el exceso de recurso no-renovables, que representa el nivel de infactibilidad del individuo.
Con la función de evaluación de Alcaraz 3.4, dos soluciones con L^{η} idéntico, pero diferente makespan puede tener diferentes valores de evaluación y la penalidad de un individuo no-factible proporciona una probabilidad razonable de participar en el proceso genético.

Sin embargo, está función de evaluación se construye adicionando unidades de tiempo al makespan y unidades de los excesos de recursos no-renovables. La magnitud de ambos aspectos de la solución puede perturbar el significado de la función de evaluación. Para solucionar este punto, Lova [39] propuso una nueva función de evaluación donde ambos aspectos de la solución son conjuntamente considerados. Esta función de evaluación es calculada para cada individuo acorde a la siguiente expresión dependiendo si el individuo es factible ($L^{\eta}(\mu) = 0$) o no-factible.

$$F(I) = \begin{cases} 1 - \dfrac{\text{máx}\_mak(P) - mak(I)}{\text{máx}\_mak(P)}, & \text{si } L^{\eta}(\mu) = 0. \\ 1 + \dfrac{\text{máx}\_mak(P) - \text{mín}\_CP}{mak(I)} + \displaystyle\sum_{k \in \mathcal{R}^{\eta}} \text{máx}\left\{0, \dfrac{\sum_{j=1}^{n}(r^{\eta}_{j\mu(j)k} - \mathcal{R}^{\eta}_{k})}{\mathcal{R}^{\eta}_{k}}\right\} + L^{\eta}(\mu), & \text{en otro caso.} \end{cases} \tag{3.4}$$

Los individuos factibles con makespan grande pueden tener un valor en su evaluación igual a 1, aunque uno de los mejores puede tener un valor de evaluación igual a cero.

3.5. Operador de cruce

El objetivo del operador de recombinación es generar nuevos agentes utilizando principalmente la información extraída de los agentes combinados, por lo tanto, tiene

un gran impacto en el resultado del algoritmo a la hora de buscar soluciones de calidad.

Cervantes [7] referenció los operadores que se utilizan con más frecuencia en la secuenciación de proyectos:

1. *Cruce de un punto.* Esta técnica fue propuesta por Davis [36]. En un cromosoma de longitud n existen $n-1$ puntos de cruce k será un número aleatorio en el intervalo $[0, n-1]$. El hijo tendrá los genes del 0 al k iguales a los de su padre y los genes desde $k+1$ hasta n heredados en el orden relativo de su madre. Este procedimiento asegura que la secuencia generada sea factible. La hija se genera de manera análoga.
   Esta técnica ha sido utilizada por Hartmann [22]. Una versión modificada de esta técnica es empleada por Alcaraz y Maroto [1], quienes utilizan el gen que representa la SD para generar el hijo de inicio a fin si el gen es Forward o de fin a inicio si el gen es Backward. Este operador fue extendido por Hartmann [20] para la versión multimodo y también es usado en el trabajo de Van Petghem [60].
2. *Cruce de Dos puntos.* Esta técnica fue propuesta por Goldbert y Lingle (1985) y puede verse como una generalización del operador anterior. Para implementar este operador, se generan dos puntos de cruce de manera aleatoria k_1 y k_2 con $k_1 < k_2$. El modo más común de realizar este cruce consiste en que el hijo hereda directamente del padre los genes del intervalo inicial $[0, k_1]$ y del intervalo final $(k_2, n]$. Los genes del intervalo $(k_1, k_2]$ se heredan en el orden relativo de la madre. La hija se genera de manera análoga.
   Este operador es utilizado por Devels y Vanhoucke [11], Hartmann [18], Alcaraz y Maroto [1],[3]. Adicionalmente han modificado el operador para incluir la información codificada en el gen de la dirección de secuenciación: si el gen es Forward el hijo se genera de inicio a fin y si es Backward se genera de fin a inicio. En el caso de múltiples modos este operador es utilizado por Özdamar [47] y por Alcaraz [2].
3. *Cruce Uniforme.* Esta técnica se trata de un cruce de n puntos, en el cual el número n no se fija previamente. Se genera una lista de números aleatorios $p_i \in \{0, 1\}$, $i = 1, \ldots, n$. De manera sucesiva se llenan las posiciones $i = 1, \ldots, n$ en el hijo. Si $p_i = 1$ se toma la actividad de la lista del padre que tenga el menor índice entre las actividades no seleccionadas y si $p_i = 0$ la actividad se toma de la lista de la madre. La hija se genera de manera análoga heredando de la madre si $p_i = 1$ y del padre si $p_i = 0$.
   Este tipo de cruce es utilizado por Hartmann [22] y Méndez [42] y en el MRCPSP es utilizado por Wall [41] para la lista de modos.
4. *Peak Crossover.* Este operador fue diseñado específicamente para el RCPSP por Valls [21] y permite combinar partes especificas de la solución que contribuyen con la calidad del individuo. En el caso del RCPSP es deseable que los hijos hereden los periodos donde se utilizan intensivamente los recursos. Este operador selecciona las secuencias parciales donde el uso de recursos es superior a un límite determinado y las hereda directamente el hijo.

5. *Blend Crossover*. Utilizado por Wall [41] en la solución del MRCPSP y fue diseñado específicamente para el arreglo que contiene los tiempos entre actividades. Este operador genera un nuevo valor dependiendo de la distancia (similitud) de los padres. Es un operador adaptativo y no tiene necesidad de ajustar ningún parámetro. Si p_1 es el valor del gen del padre y p_2 el de la madre y suponemos que $p_1 > p_2$, se calcula la distancia entre los padres como $I = |p_1 - p_2|$ y $\alpha = d/2$. El valor del hijo es un valor aleatorio en el intervalo $[p_1 - \alpha I, p_2 + \alpha I]$. Los valores se truncan según los datos del problema para mantener la factibilidad de los tiempos de inicio de las actividades.

3.5.1. Descripción formal

Consideremos el espacio de búsqueda $\mathcal{I}$ conformado por agentes y el espacio de secuencias $\mathcal{S}$. Sea $\rho : \mathcal{I} \to \mathcal{S}$ es una función de representación, tal que para cualquier agente devuelve una secuencia de $\mathcal{S}$. Asumamos que ρ es inyectiva (tal que para todo $\lambda \in \mathcal{I}$, existe una única secuencia $\rho(\lambda) \in \mathcal{S}$ que la representa), pero no es sobreyectiva, es decir, puede haber secuencias en $\mathcal{S}$ que no tienen asignado una agente $\mathcal{I}$. Sea f la función de evaluación, la cual puede considerarse como una función

$$f : \mathcal{S} \to \mathbb{R}^+$$

Se asume que el objetivo es minimizar la evaluación y el conjunto de óptimos globales se denota por $\mathcal{S}^* \subset \mathcal{S}$.
Sea σ un operador de movimiento unario estocástico sobre $\mathcal{S}$. Asignamos al operador σ un conjunto $\mathcal{K}_\sigma$(no vació) llamado conjunto de control, el cual proporciona los posibles movimientos que pueden ocurrir. La forma funcional de σ puede ser

$$\sigma : \mathcal{I} \times \mathcal{K}_\sigma \to \mathcal{I}$$

Una secuencia $S \in \mathcal{S}$ se dice que es un óptimo local con respecto a σ si no existe una secuencia S^* de menor aptitud que S, tal que se pueda generar por una sola aplicación de σ, esto es, si y solo si

$$\forall \kappa \in \mathcal{K}_\sigma : \quad f(S) \leq f(\sigma(S, \kappa))$$

Definición 3.1. *Dado un conjunto de control $\mathcal{K}_\sigma$ y un espacio de búsqueda $\mathcal{I}$. A una función*

$$\mathcal{X} : \mathcal{I} \times \mathcal{I} \times \mathcal{K}_\sigma \to \mathcal{I}$$

es llamado operador de cruce o recombinación sobre $\mathcal{I}$.

Definición 3.2. *Sea $\mathbb{B} = \{0, 1\}$ tal que $\mathbb{B}^n$ es el conjunto de vectores binarios de n componentes. A veces $\mathbb{B}$ es interpretado como el conjunto de valores de verdad, donde 0 corresponde a falso y 1 a verdadero.*

A continuación se presenta un nuevo un operador de cruce uniforme el cual es una generalización del operador de cruce de dos puntos, presentado por Hartmann [19] para el RCPSP. En este trabajo se redefine para el MRCPSP.

Definición 3.3 (Operador de cruce uniforme). *El operador de cruce uniforme*

$$\mathcal{X}_U : \mathcal{I} \times \mathcal{I} \times \mathbb{B}^{n+1} \to \mathcal{I}$$

se inicia seleccionando dos agentes I^M *e* I^F *del espacio de búsqueda* $\mathcal{I}$ *y un vector* $v \in \mathbb{B}^{n+1}$. *Se aplica el operador de cruce* $\mathcal{X}_U(I^M, I^F, v) = I^D = (\lambda^D, \mu^D, (f/b))$. *La lista de actividades* λ^D *de* I^D *se define de la siguiente forma:*

Si $v_{n+1} = 0$, *entonces para* $i = 1$ *hasta* n,

$$j_i^D = \begin{cases} j_k^M & si \quad v_i = 0 \text{ con } k = \text{mín}\{\ell \in N : j_\ell^M \notin \{j_1^D, \ldots, j_{i-1}^D\}\} \\ j_k^F & si \quad v_i = 1 \text{ con } k = \text{mín}\{\ell \in N : j_\ell^F \notin \{j_1^D, \ldots, j_{i-1}^D\}\} \end{cases}$$

Si $v_{n+1} = 1$, *entonces para* $i = n$ *hasta 1,*

$$j_i^D = \begin{cases} j_k^M & si \quad v_i = 0 \text{ con } k = \text{máx}\{\ell \in N : j_\ell^M \notin \{j_1^D, \ldots, j_{i-1}^D\}\} \\ j_k^F & si \quad v_i = 1 \text{ con } k = \text{máx}\{\ell \in N : j_\ell^F \notin \{j_1^D, \ldots, j_{i-1}^D\}\} \end{cases}$$

para el vector de modos μ^D *de* I^D *se define por:*

$$\mu(j_i^D) = \begin{cases} \mu^M(j_i^D) & si \quad v_i = 0 \\ \mu^F(j_i^D) & si \quad v_i = 1 \end{cases}$$

Una manera de generar un nuevo agente es alternando la posición de I^M *e* I^F.

Esté procedimiento es mostrado en el algoritmo (15) y la generación de un vector $v \in \mathbb{B}^{n+1}$ en el algoritmo (14). De la definición anterior se puede observar que existen cuatro diferentes alternativas para la generación de agentes, ampliando la exploración de diferentes regiones en el espacio de búsqueda.

Algoritmo 14: Generador de vectores binarios

Función: $GenerarVectorBinario(n)$ `// n: número de actividades no ficticias.`

Para $i = 1$ *hasta* $n+1$ **hacer**

  Si $0{,}5 > RND$ **entonces** `//` RND`: número elegido uniformemente entre 0 y 1.`

    $v_i = 1$

  Si no

    $v_i = 0$

Return $v = (v_1, \ldots, v_{n+1})$

Algoritmo 15: Operador de cruce uniforme

Función: $Crossover(I^M, I^F, v)$

Si $v_{n+1} = 0$ **entonces**

  Para $i = 1$ *hasta* n **hacer**

    Si $v_i = 0$ **entonces**

      Calcular $k = \text{mín}\{\ell \in N : j_\ell^M \notin \{j_1^D, \ldots, j_{i-1}^D\}\}$;

      Definir $j_i^D = j_k^M$

    Si no

      Calcular $k = \text{mín}\{\ell \in N : j_\ell^F \notin \{j_1^D, \ldots, j_{i-1}^D\}\}$;

      Definir $j_i^D = j_k^F$

Si no

  Para $i = n$ *hasta* 1 **hacer**

    Si $v_i = 0$ **entonces**

      Calcular $k = \text{máx}\{\ell \in N : j_\ell^M \notin \{j_1^D, \ldots, j_{i-1}^D\}\}$;

      Definir $j_i^D = j_k^M$

    Si no

      Calcular $k = \text{máx}\{\ell \in N : j_\ell^F \notin \{j_1^D, \ldots, j_{i-1}^D\}\}$;

      Definir $j_i^D = j_k^F$

Para $i = 1$ *hasta* n **hacer**

  Si $v_i = 0$ **entonces**

    $\mu(j_i^D) = \mu^M(j_i^D)$

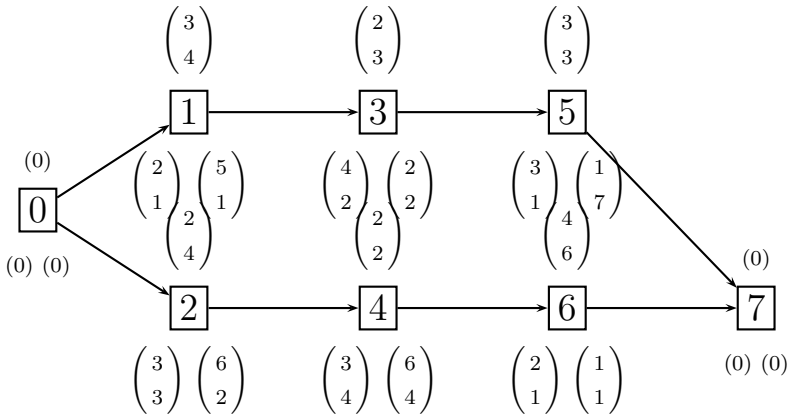
  Si no

    $\mu(j_i^D) = \mu^F(j_i^D)$

Return I^D

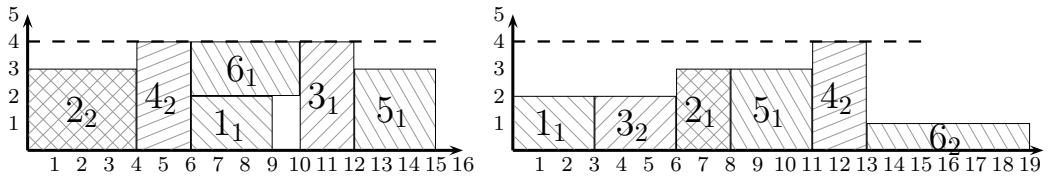
Para generar otro agente I^S a partir de los agentes I^M e I^F, simplemente cambiamos el orden de I^M e I^F, es decir $Crossover(I^F, I^M, v) = I^S$

Ejemplo 3.4. Mostremos un PSP de Hartmann [1] utilizando una RAN, con 6 actividades reales, un recurso renovable $\mathcal{R}^{\tau} = \{1\}$, un recurso no-renovable $\mathcal{R}^{\eta} = \{1\}$ tal que $\mathcal{R}^{\tau}_1 = 4$ y $\mathcal{R}^{\eta}_1 = 15$.



Considere dos agentes I^M de recurso factible y I^F que no es de recurso factible.

$$I^M = \begin{pmatrix} 2\ 4\ 1\ 6\ 3\ 5\ 0 \\ 2\ 2\ 1\ 1\ 1\ 1 \end{pmatrix} \qquad I^F = \begin{pmatrix} 1\ 3\ 2\ 5\ 4\ 6\ 0 \\ 1\ 2\ 1\ 1\ 2\ 2 \end{pmatrix}$$



(a) Secuencia asociada a I^M (b) Secuencia asociada a I^F

Figura 3.1: Secuencias asociadas a I_M e I_F

Suponga que se eligió aleatoriamente el vector $v = (0, 1, 1, 0, 1, 0, (f/b))$ entonces

$$I^M = \begin{pmatrix} 2\ 4\ 1\ 6\ 3\ 5\ 0 \\ 2\ 2\ 1\ 1\ 1\ 1 \end{pmatrix}$$

si $v_7 = 0$

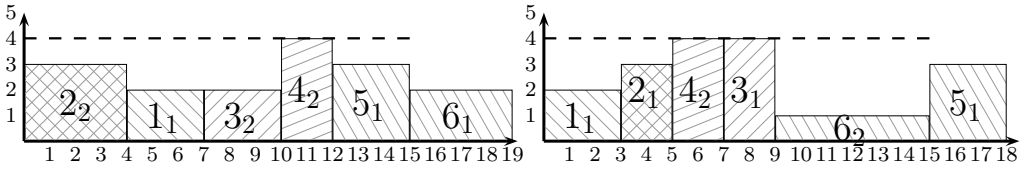
$$I^{D_1} = \begin{pmatrix} 2\ 1\ 3\ 4\ 5\ 6\ 0 \\ 2\ 1\ 2\ 2\ 1\ 1 \end{pmatrix}$$

$$I^{D_2} = \begin{pmatrix} 1\ 2\ 4\ 3\ 6\ 5\ 1 \\ 1\ 1\ 2\ 1\ 2\ 1 \end{pmatrix}$$

si $v_7 = 1$

$$I^F = \begin{pmatrix} 1\ 3\ 2\ 5\ 4\ 6\ 0 \\ 1\ 2\ 1\ 1\ 2\ 2 \end{pmatrix}$$

Se observa que si $v_7 = 0$, entonces $crossover(I^M, I^F, v) = I^{D_1}$. Además, si $v_7 = 1$, entonces $crossover(I^M, I^F, v) = I^{D_2}$. Las secuencias asociadas son:



(a) Secuencia asociada a I^{D_1} (b) Secuencia asociada a I^{D_2}

Figura 3.2: Secuencias generadas por el operador de cruce.

De manera análoga, se pueden conseguir agentes alternando la posición de I^M e I^F, es decir,

$$crossover(I^F, I^M, v) = \begin{pmatrix} 2\,1\,3\,4\,5\,6\,0 \\ 1\,1\,1\,2\,1\,2 \end{pmatrix} \qquad crossover(I^F, I^M, v) = \begin{pmatrix} 1\,2\,4\,3\,6\,5\,1 \\ 1\,2\,2\,2\,1\,1 \end{pmatrix}$$

3.6. Operador de mutación

El operador de mutación es aplicado en dos etapas del algoritmo, después que se realiza la búsqueda local a la población POB y cuando un agente es generado por el operador de cruce. El operador de mutación que hemos utilizado es una adaptación al operador usado por Lova y Tormos [39] para el MRCPSP.

Una vez que el operador de cruce se ha aplicado y la población de agentes generados ha reemplazado a la población original, el operador de mutación es aplicado a la población de agentes. La mutación altera uno o más genes (posiciones) de un cromosoma seleccionado, esto es, para reintroducir material perdido genéticamente y así tener cierta variabilidad extra en la población. De hecho, la mutación puede dar lugar a valores de genes totalmente nuevos, que a veces permiten que el algoritmo memético llegue a mejores soluciones que versiones anteriores de algoritmos genéticos. Además, la mutación ayuda a prevenir el estancamiento de la población en cualquier óptimo local.

La mutación se aplica tanto a las componentes de cada representación individual que son la lista de actividades y el vector de modos. En el primer caso, el procedimiento de mutación utilizado es el de inserción que funciona de la siguiente manera: para cada actividad en la lista de actividades una nueva posición es elegida al azar entre las posiciones más alta de sus predecesores y la posición más baja de sus sucesores. La actividad se inserta en la nueva posición con una probabilidad $p_{mut} = 0{,}05$.

En cuanto a la lista de asignación de modos, la aplicación de los cambios de mutación del operador cambian dependiendo de si el individuo tiene o no un vector de modos recurso factible.

1. Si el agente tiene un vector de modos factible, se deja dicho vector de modos.
2. Si el vector de modos no es factible, usamos la función $Arreglarrec(\mu)$.

Algoritmo 16: Operador de mutación

Función: $Mutacion(I)$
Elegir aleatoriamente i en $\{1, \ldots, n\}$;
Si $RND \leq P_{mut}$ **entonces**
  Calcular $j_\mu = \max \mathcal{P}(j_i)$ y $j_\nu = \min \mathcal{S}(j_i)$;
  Elegir $j_r \in (j_\mu, j_\nu)$;
  $\hat{j}_r = j_i$;
  $\hat{j}_i = j_r$
Si no
  $\hat{j}_i = j_i$
Si μ *es recurso factible* **entonces**
  Para $i = 1$ *hasta* n **hacer**
    $\hat{\mu}(j_i) = \mu(j_i)$
Si no
  $Arreglarrec(\mu)$
Return $\hat{I}$

Algoritmo 17: Arreglar recursos

Función: $Arreglarrec(I)$
Repetir
  foreach $\ell \in \mathcal{R}^\eta$ **do**
    calcular $r^\eta_\ell(\mu)$;
    Mientras $r^\eta_\ell(\mu) > \mathcal{R}^\eta_\ell$ **hacer**
      Para $j = 1$ *hasta* n **hacer**
        $P(j) = \frac{r^\ell_{j\mu(j)\ell}}{r^\eta_\ell(\mu)}$;
        Elegir aleatoriamente una actividad i con probabilidad de ser elegida $P(i)$;
        elegir $\hat{\mu}_i = \min\{\mu_i \in \mathcal{M}_i : r^\eta_{i\mu_i\ell} = \min_{m \in \mathcal{M}_i} r^\eta_{im\ell}\}$;
        $\mu_i = \hat{\mu}_i$;
        redefinir I
Hasta μ *recurso factible o* 100 *veces*;
Return I

Ejemplo 3.5. Consideremos la lista I^F. Asumiendo que la actividad que se eligió es 5 al azar por mutación y que su ultimo predecesor es 3 y su primer sucesor es 7, un número aleatorio entre 2 y 7 (la posición de la actividad) que se ha generado, el valor obtenido es igual a 6 y por lo tanto la actividad 5 se inserta en esa posición, dando como resultado el agente $I^{F'}$. Ahora, el agente $I^{F'}$ no es recurso factible puesto que tiene un exceso de recurso no renovable de 4 unidades, por lo tanto aplicando el algoritmo (17), lo transforma en $I^{F''}$ el cual es recurso factible.

$$I^F = \begin{pmatrix} 1\,3\,2\,5\,4\,6\,0 \\ 1\,2\,1\,1\,2\,2 \end{pmatrix}$$

$$I^{F'} = \begin{pmatrix} 1\,3\,2\,4\,6\,5\,0 \\ 1\,2\,1\,1\,2\,2 \end{pmatrix}$$

$$I^{F''} = \begin{pmatrix} 1\,3\,2\,4\,6\,5\,0 \\ 1\,2\,2\,1\,2\,2 \end{pmatrix}$$

3.7. Selección

Los algoritmos meméticos combinan conceptos de los algoritmos genéticos, por lo tanto el operador de selección es uno de los componentes principales de la búsqueda. El principal objetivo del operador de selección es escoger *los mejores individuos, que tienen mayor probabilidad de ser padres (reproducirse)* frente a los individuos de menor calidad. Esta idea nos define la **presión selectiva** que determina en qué grado la reproducción está dirigida por los mejores individuos. Sin embargo, se debe tener cuidado para dar una oportunidad de reproducirse a los agentes menos buenos. Estos pueden incluir información útil en el proceso de reproducción.

Cervantes [7], describe tres métodos los cuales son los más utilizados en la secuenciación de proyectos:

1. Ruleta: El método de la ruleta ha sido el método más comúnmente utilizado desde el origen de los AG. Los padres se seleccionan de acuerdo con su aptitud. Los individuos mejores son los que tienen mayores posibilidades de ser elegidos, siendo la probabilidad de ser seleccionados proporcional a su aptitud. Este método es utilizado por Hartmann [22].
2. Torneo: La idea básica de este método es seleccionar con base en comparaciones directas de los individuos. En la manera más común de implementarlo se comparan p individuos de la población escogidos al azar y se selecciona el que tenga mejor aptitud. El tamaño del torneo es p, y usualmente $p = 2$. Cuando $p \geq 10$ la selección se considera dura y se considera blanda cuando $2 \leq p \leq 5$. Este tipo de selección ha sido utilizada por Hartmann [22], Alcaraz y Maroto [1],[3]. En Debels y Vanhoucke [11] un padre es escogido por torneo

y el otro es el $i-$ésimo individuo de la población. En cuanto al MRCPSP Van Petghem [60] utiliza el método de 2-torneo para seleccionar cada uno de los padres a los que se les aplicarán los operadores genéticos.

3. Ranking: Este método fue diseñado para subsanar el tema de la convergencia prematura que puede surgir cuando existan grandes diferencias entre los fitness de los individuos de la población. Por ejemplo, si un cromosoma ocupa el 90 % de la ruleta el resto de los cromosomas tienen muy pocas posibilidades de ser elegidos. La selección por ranking da solución a este problema, los individuos son ordenados de acuerdo a su ranking de fitness, de esta manera si tenemos K cromosomas, al individuo con peor fitness se le asignará un 1 y el que tenga el mejor fitness se le asignará la K. Una vez más, en el caso de minimizar (como en el RCPSP) se debe transformar la aptitud de los individuos. Este método es utilizado por Alcaraz y Maroto [1] para resolver el RCPSP y por Hartmann para resolver el RCPSP [18],[21] y el MRCPSP [20].

El algoritmo memético presentado en este documento, está basado en la selección por ranking, donde se ordenan los agentes de mayor a menor según su evaluación y se forman parejas tomando el primer individuo con el último, luego el segundo con el penúltimo y así sucesivamente. La ventaja de esta selección de agentes radica en que permite aprovechar ciertas características beneficiosas que seguramente tiene el individuo más débil y que los fuertes no han de poseer, permitiendo que se repita en las próximas generaciones. Este procedimiento se muestra en el siguiente algoritmo (18):

Algoritmo 18: Selección

Función: $Seleccion(POB)$
foreach $I \in POB$ **do** $F(I)$;
Ordenar POB de mayor a menor según el fitness;
Formar parejas $\{(I_1, I_{pob}), (I_2, I_{pob-1}), \ldots, (I_{pob/2}, I_{pob/2+1})\}$;
Seleccionar los mejores $Pob/2$ agentes de POB

3.8. Búsqueda local

En una gran cantidad de problemas de optimización, existen formas sencilla de construir soluciones. Sin embargo, las soluciones obtenidas mediante estos métodos constructivos suelen ser en general de una calidad moderada, lo que lleva a desarrollar algoritmos que actúen sobre una solución inicial dada e intenten mejorarla.

Una búsqueda local es un proceso que, dada la solución actual en la que se encuentra el recorrido, selecciona iterativamente una solución de su entorno, reflejando el concepto de proximidad o vecindad entre las soluciones alternativas del problema.

3.8.1. Conceptos básicos

La función ρ no necesariamente es sobreyectiva, es decir, pueden haber secuencias en $\mathcal{S}$ que no tienen asignado un agente en $\mathcal{I}$. El objetivo de definir la función ρ es garantizar la existencia de una secuencia, siempre que se encuentre un agente.

Definición 3.6. *Una estructura de vecindario $\mathcal{N}$ es una función $\mathcal{N} : \mathcal{I} \to 2^{\mathcal{I}}$, que asigna a cada solución $I \in \mathcal{I}$ un conjunto solución $\mathcal{N}(I) \subseteq \mathcal{I}$. El conjunto $\mathcal{I}$ se denomina el vecindario de la solución I.*

Las metaheurísticas de búsqueda local aplican una transformación o movimiento a la solución de búsqueda y por tanto utilizan, explícita o implícitamente una estructura de vecindario. La vecindad de una solución $I \in \mathcal{I}$ estaría constituido por todos aquellas soluciones que se pueden obtener desde I mediante una de las transformaciones o movimientos contemplados.

Definición 3.7. *La vecindad $\mathcal{N}(I)$ de una solución $I \in \mathcal{I}$ será representada por el conjunto*

$$\mathcal{N}_{\mathcal{O}}(I) := \{I' \in \mathcal{I} : \mathcal{O}_i(I) = I'\}$$

donde el operador $\mathcal{O}$ se denomina movimiento, o, que modifica, en algún sentido, la solución actual I para obtener nuevas soluciones. Además, $\mathcal{O}_i(I)$ representa la i-ésima aplicación de $\mathcal{O}$ sobre I.

Una vez definido de alguna manera apropiada el vecindario de una solución I como $\mathcal{N}(I)$ y dada una solución inicial $I = I_0$, es posible definir un esquema de mejoramiento iterativo que comience buscando alguna solución $I_1 \in \mathcal{N}(I_0)$, tal que I_1 verifica alguna propiedad. Posteriormente, I_1 pasa a ser la solución actual y el proceso se repite hasta que se satisfaga algún criterio de parada, o no sea posible obtener ninguna solución $I_{k+1} \in \mathcal{N}(I_k)$ que satisfaga los requerimientos establecidos. En este momento, se establece que I_k es un óptimo local.

Definición 3.8. *Dada una función objetivo $f : \mathcal{I} \to \mathbb{R}^+$, una solución $I \in \mathcal{I}$ es un mínimo global si*

$$f(I) := \min_{\hat{I} \in \mathcal{I}} f(\hat{I})$$

Diremos que la solución $I^* \in \mathcal{I}$ es un mínimo local con respecto a una vecindad $\mathcal{N}(I)$ si

$$f(I^*) := \min_{\hat{I} \in \mathcal{N}(I)} f(\hat{I})$$

Un problema de los procedimientos de búsqueda local, es que suelen caer en óptimos locales, que a veces están bastante alejados del óptimo global del problema. Por está razón implementamos la metaheurística búsqueda de vecindarios variables (VNS, Variable Neighborhood Search) descrita en Hansen [17], consiste en cambiar la estructura de vecindarios de forma sistemática con el fin de continuar la búsqueda después de encontrar un óptimo local.

La metaheurística VNS se basa en aprovechar sistemáticamente tres hechos simples:

1. Un mínimo local en una estructura de vecindades no lo es necesariamente con otra.
2. Un mínimo global es un mínimo local con todas las posibles estructuras de vecindades.
3. Para muchos problemas, los mínimos locales están relativamente próximos entre sí.

Esta última observación, que es empírica, implica que los óptimos locales proporcionan información acerca del óptimo global. Por ejemplo, puede ser que ambas soluciones tengan características comunes. Sin embargo, generalmente no se conoce cuáles son esas características. Es procedente, por tanto, realizar un estudio organizado en las proximidades de este óptimo local, hasta que se encuentre uno mejor.

El procedimiento implementado en VNS utiliza tan solo dos vecindarios, definidos de la siguiente forma:

$\mathcal{N}_1$ Inserción hacia adelante: Dado un agente $I \in \mathcal{I}$, se define

$$\mathcal{N}_1(I) := \{I' \in \mathcal{I} : \mathcal{O}_1(I) = I'\}$$

seleccionando aleatoriamente a $\lambda, \mu \in \{1, 2, \ldots, n-1\}$ con $\mu < \lambda$, el nuevo agente será generado al colocar la actividad i_λ en la posición μ y las actividades en las posiciones intermedias se mueven una posición hacia la derecha para garantizar la factibilidad, además para todo $v \in \{\mu, \ldots, \lambda - 1\}$ no existe relación de precedencia $i_v \to i_\lambda$.

$\mathcal{N}_2$ Inserción hacia atrás: Dado un agente $I \in \mathcal{I}$, se define

$$\mathcal{N}_2(I) := \{I' \in \mathcal{I} : \mathcal{O}_2(I) = I'\}$$

seleccionando aleatoriamente $\lambda, \mu \in \{1, 2, \ldots, n-1\}$ con $\lambda < \mu$, el nuevo agente será generado al colocar la actividad i_λ en la posición μ y las actividades en las posiciones intermedias se mueven una posición hacia la izquierda para garantizar la factibilidad, además para todo $v \in \{\lambda + 1, \ldots, \mu\}$ no existe relación de precedencia $i_\lambda \to i_v$.

Las vecindades $\mathcal{N}_1$ y $\mathcal{N}_2$ están acotadas por $O(n^2)$.
Dado un agente I, se definen las vecindades de I, es decir $\mathcal{N}_1(I)$ y $\mathcal{N}_2(I)$. Se inicia la búsqueda, en la vecindad $\mathcal{N}_1(I)$, si se encuentra un agente mejor que I, se reemplaza dicho agente por I y se inicia la búsqueda predefiniendo las vecindades $\mathcal{N}_k$ $(k = 1, 2)$. Si no se

encuentra un individuo mejor que I, se cambia de vecindad $\mathcal{N}_1$ por $\mathcal{N}_2$ y se realiza el paso anterior, esté termina hasta que se hayan evaluado todas las vecindades o hasta que un criterio de parada se cumpla.
El primer algoritmo que se utiliza, proporciona la solución, cambiando la vecindad:

Algoritmo 19: Cambio de vecindad

Función: $CambiodeVecindad(I, I', k)$
Si $f(I') < f(I)$ **entonces**
    $I := I'$;
    $k := 1$;
Si no
    $k := k + 1$;
Return I, k

La función $CambiodeVecindad(\cdot)$ compara los valores de $f(I)$ con el nuevo valor $f(I')$ obtenido de la $k-$ésima vecindad. Si se obtiene una mejora, el nuevo dato es actualizado y k es retornado a su valor inicial. Si no, la proxima vecindad es considerada.
El cambio de vecindad se puede realizar de forma determinística, estocástica o determinística y estocástica a la vez.

La búsqueda que se implementa sa basa en el cambio de vecindades de forma determinística, llamada búsqueda descendente por entornos variables, el cual aplica una búsqueda monótona, es decir que mejora por entornos, cambiando de forma sistemática las vecindades cada vez que se alcance un mínimo local.

Algoritmo 20: VND

Función: $VND(I, 2)$
$k := 1$;
Repetir
    $I' := ObtenerVecino(I, k)$;
    $CambiodeVecindad(I, I', k)$;
Hasta $k = 2$ *o un número prefijado de veces*;
Return I

La función $obtenerVecino(I, k)$ genera un nuevo agente I' aleatorio de la $k-$ésima vecindad de I, esto es, $I' \in \mathcal{N}_k(I)$ para $k = 1, 2$. Se definen los algoritmos 21 y 22 para cada vecindad.

Algoritmo 21: Obtener vecino de $\mathcal{N}_1$

Función: $ObtenerVecino(I, 1)$

$\lambda = random(2, n)$, Elegir un entero uniforme de 2 hasta n ;

$\mu = random(1, \lambda - 1)$, Elegir un entero uniforme de 1 hasta $\lambda - 1$;

Definir $\mathcal{P}_\lambda$;

Si *para* $v \in \{\mu, \ldots, \lambda - 1\}$ *no existe relación de precedencia* $i_v \to i_\lambda$ **entonces**

  $i_\mu = i_\lambda$;

  Para t *de* μ *hasta* λ **hacer**

    $i_{t+1} = i_t$

Si no

  ir al paso inicial eligiendo nuevamente λ y μ

Return I'

Algoritmo 22: Obtener vecino de $\mathcal{N}_2$

Función: $ObtenerVecino(I, 2)$

$\lambda = random(1, n - 1)$, Elegir un entero uniforme de 1 hasta $n - 1$;

$\mu = random(\lambda + 1, n)$, Elegir un entero uniforme de $\lambda + 1$ hasta n ;

Definir $\mathcal{S}_\lambda$;

Si *para todo* $v \in \{\lambda + 1, \ldots, \mu\}$ *no existe relación de precedencia* $i_\lambda \to i_v$ **entonces**

  $i_\mu = i_\lambda$;

  Para t *de* $\lambda + 1$ *hasta* μ **hacer**

    $i_{t-1} = i_t$

Si no

  ir al paso inicial eligiendo nuevamente λ y μ

Return I'

3.9. Resultados computacionales

En esta sección se muestran los resultados de las pruebas computacionales concernientes al MA introducido en las secciones anteriores. Los métodos desarrollados se implementaron con un sistema operativo Microsoft Windows 7 Starter, Procesador: Pentium P6200, 2.13 GHz y 2.00 GB de RAM. El algoritmo se implemento en lenguaje C++ y evaluado en los problemas de PSPLIB.

Ballestín [4], explica algunas notaciones que se emplean en las tablas de resultados y a lo largo del documento. Si $\mathcal{A}$ es un algoritmo, $\mathcal{A}(i)$ es la duración de la solución proporcionada por $\mathcal{A}$ en la instancia i. Del mismo modo, $UB(i)$ ($CPM(i)$) es el valor de la solución contenida en PSPLIB (el valor del CPM) para esa instancia. Teniendo en cuenta esto, y que $\mathcal{K}$ es el conjunto de PSPLIB con el que se está trabajando, definimos:

$\sum \mathcal{A} := \sum_{i \in \mathcal{K}} \mathcal{A}(i)$	La suma de las duraciones de las secuencias obtenidas por $\mathcal{A}$
$\sum PSPLIB := \sum_{i \in \mathcal{K}} UB(i)$	La suma de las mejores soluciones conocidas
$desv\_UB(\mathcal{A}) := \frac{1}{\lvert\mathcal{K}\rvert} \sum_{i \in \mathcal{K}} \frac{\mathcal{A}(i) - UB(i)}{UB(i)}$	La desviación media con respecto a las mejores soluciones de PSPLIB
$desv\_CPM(\mathcal{A}) := \frac{1}{\lvert\mathcal{K}\rvert} \sum_{i \in \mathcal{K}} \frac{\mathcal{A}(i) - CPM(i)}{CPM(i)}$	La desviación media con respecto al CPM
$\text{máx}\_desv(\mathcal{K}) := \text{máx}\{\frac{\mathcal{A}(i) - UB(i)}{UB(i)} : i \in \mathcal{K}\}$	La máxima desviación con respecto a las cotas superiores
$mejor\_sol(\mathcal{A}) := \lvert\{i \in \mathcal{K} : \mathcal{A}(i) = UB(i)\}\rvert$	Número de soluciones donde $\mathcal{A}$ obtiene la misma duración que la mejor solución conocida.

La $desv\_UB(\mathcal{A})$ como $desv\_CPM(\mathcal{A})$, así como la suma, sirven para comparar la eficacia de los algoritmos. La desviación respecto del CPM es la medida de eficiencia más comúnmente empleada por los investigadores, dado que las cotas superiores cambian con el paso del tiempo. La desviación con respecto a la mejor solución es una medida intuitiva que proporciona una mejor estimación de la desviación con respecto del óptimo.

Para calibrar los parámetros del MA se implementaron tres instancias de PSPLIB. Las tablas 3.1 y 3.2 muestran los resultados obtenidos a la solución encontrada y la desviación relativa, respectivamente, de acuerdo al óptimo, cambiando el tamaño de la población y el número de generaciones.

- La primera instancia es J1037_2 perteneciente al conjunto J10 generado por Kolish y Sprecher [35]. Este ejemplo es utilizado por Cervantes y Lova [39]. Es un proyecto con 10 actividades no ficticias que requieren para su ejecución dos tipos de recursos renovables y dos no renovables.
  Uno de los agentes asociado con la secuencia optima es:

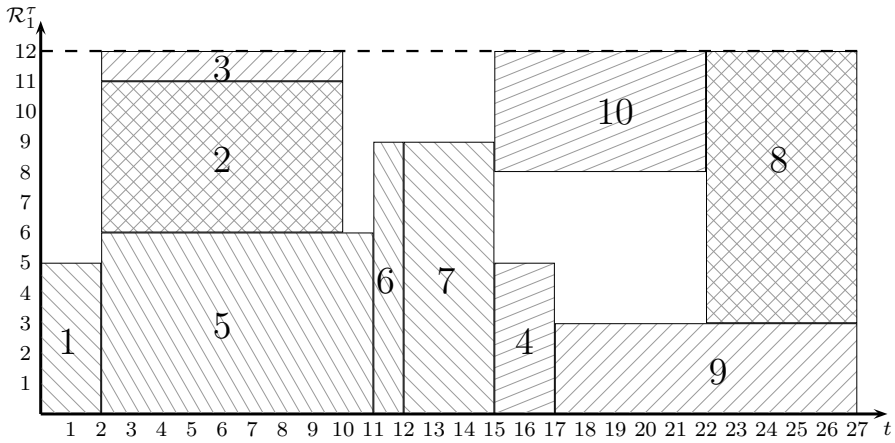
$$I = \begin{pmatrix} 0\ 1\ 2\ 3\ 5\ 6\ 7\ 4\ 9\ 10\ 8\ 11 \\ 1\ 1\ 2\ 3\ 3\ 1\ 3\ 3\ 3\ \ 3\ \ 2\ \ 1 \end{pmatrix}$$

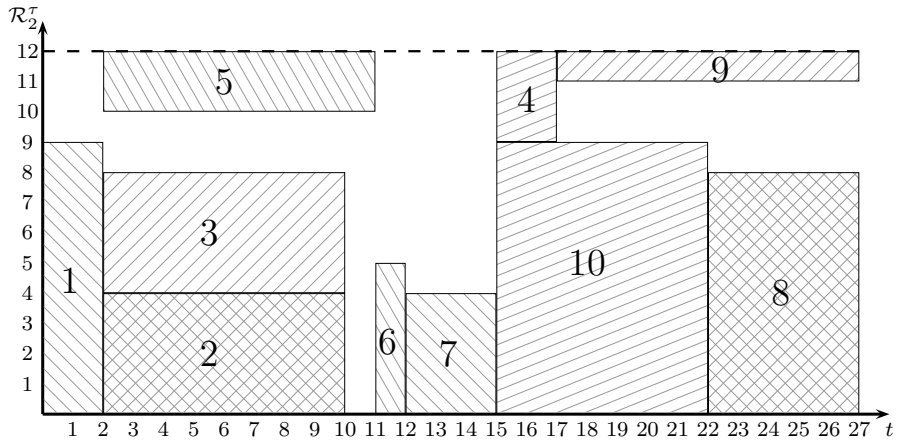
Problema J1037_2										Óptimo 27	
$n = 50$	Tamaño de la población										
		20		40		60		80		100	
		Sol	tiempo	Sol	tiempo	Sol	tiempo	Sol	tiempo	Sol	tiempo
Gene	25	29	0.082	29	0.149	27	0.217	27	0.288	27	0.358
	50	27	0.141	27	0.268	27	0.405	27	0.537	27	0.657
	75	27	0.2	27	0.386	27	0.568	27	0.768	27	0.942
	100	27	0.257	27	0.528	27	0.759	27	1	27	1.33

Cuadro 3.1: Resultado de la instancia J1037_2

Problema J1037_2						Óptimo 27
$n = 10$	Tamaño de la población					
		20	40	60	80	100
		Sol	Sol	Sol	Sol	Sol
Gene	25	0.07	0.07	0	0	0
	50	0	0	0	0	0
	75	0	0	0	0	0
	100	0	0	0	0	0

Cuadro 3.2: Desviación relativa de J1037_2





- La segundo instancia es J501_1 perteneciente al conjunto J50 de Boctor. Es un proyecto con 50 actividades no ficticias que requieren para su ejecución dos tipos de recurso renovables y dos no renovables y además cada actividad se puede realizar de 3 modos. El mismo procedimiento se realiza para la instancia J501_2. Los resultados obtenidos se muestran en los cuadros 3.14, 3.4, 3.15 y 3.6.

Problema J501_1											Óptimo 31
$n = 50$	Tamaño de la población										
		20		40		60		80		100	
		Sol	tiempo	Sol	tiempo	Sol	tiempo	Sol	tiempo	Sol	tiempo
Gen	25	37	1.6	35	3.11	33	5.019	33	6.349	33	7.47
	50	34	2.85	33	5.411	33	8.191	33	11.11	33	14.33
	75	34	4.2	33	7.99	33	11.9	33	16.06	33	19.8
	100	34	5.16	34	10.12	33	15.17	33	20.13	32	27.384

Cuadro 3.3: Resultado de la instancia J501_1

Problema J501_1						Óptimo 31
$n = 50$	Tamaño de la población					
		20	40	60	80	100
		Sol	Sol	Sol	Sol	Sol
Gen	25	0.193	0.129	0.064	0.064	0.064
	50	0.096	0.064	0.064	0.064	0.064
	75	0.096	0.064	0.064	0.064	0.064
	100	0.096	0.096	0.064	0.064	0.032

Cuadro 3.4: Desviación relativa de J501_1

- J501_2

Problema J501_2										Óptimo 26
$n = 50$	Tamaño de la población									
		20		40		60		80	100	
		Sol	tiempo	Sol	tiempo	Sol	tiempo	Sol	tiempo	Sol	tiempo
Generación	25	29	1.73	30	3.06	28	4.5	28	6.05	28	7.49
	50	30	2.83	30	5.6	30	8.51	28	11.08	27	14.216
	75	28	4.06	27	8.42	28	12.1	27	16.6	27	20.3
	100	30	5.45	28	10.55	28	16.2	27	20.97	27	27.31

Cuadro 3.5: Resultado de la instancia J501_2

Problema J501_1						Óptimo 26
$n = 50$	Tamaño de la población					
		20	40	60	80	100
		Sol	Sol	Sol	Sol	Sol
Generación	25	0.115	0.153	0.076	0.076	0.076
	50	0.153	0.153	0.153	0.076	0.038
	75	0.076	0.038	0.0769	0.038	0.038
	100	0.153	0.076	0.076	0.038	0.038

Cuadro 3.6: Desviación relativa de J501_2

Los resultados obtenidos nos permiten deducir que el MA obtiene soluciones factibles de excelente calidad en todas las instancias con un numero pequeño de actividades, tanto así que alcanza el óptimo en las instancias J10, J12, J14, J16, J18, J20 y J30, mientras que en las de mayor tamaño obtiene soluciones factibles, de muy buena calidad.
Presentamos a continuación los resultados obtenidos al ejecutar el MA en 10 instancias de cada conjunto J10, J12, J14, J16, J18, J20, J30 y J50. Los parámetros son de 10000 secuencias y los resultados se muestran en los cuadros 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14 y 3.15.

Problema J1010				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	17	17	0	1.262
2	24	24	0	1.242
3	21	21	0	1.242
4	15	15	0	1.226
5	24	24	0	1.24
6	18	18	0	1.222
7	15	15	0	1.232
8	15	15	0	1.302
9	10	10	0	1.21
10	17	17	0	1.246

Cuadro 3.7: Resultado de la instancia J1010

Problema J1210				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	20	20	0	1.782
2	21	21	0	1.636
3	15	15	0	1.622
4	20	20	0	1.632
5	16	16	0	1.732
6	27	27	0	1.647
7	15	15	0	1.642
8	11	11	0	1.671
9	22	22	0	1.846
10	17	17	0	1.644

Cuadro 3.8: Resultado de la instancia J1210

Problema J1410				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	16	16	0	2.027
3	24	24	0	1.996
4	19	19	0	2.044
5	19	19	0	1.981
6	21	21	0	2.075
7	30	30	0	2.059
8	19	19	0	2.106
9	16	16	0	2.059
10	24	24	0	2.013
11	23	23	0	1.966

Cuadro 3.9: Resultado de la Instancia J1410

Problema J1610				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	22	22	0	2.418
2	21	21	0	2.621
3	26	26	0	2.465
4	20	20	0	2.418
5	28	28	0	2.433
6	29	29	0	2.465
7	21	21	0	2.449
8	20	20	0	2.418
9	24	24	0	2.402
10	17	17	0	2.434

Cuadro 3.10: Resultado de la instancia J1610

Problema J1810				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	25	25	0	2.948
2	21	21	0	2.901
3	20	20	0	2.917
4	20	20	0	2.979
5	23	23	0	2.917
6	25	25	0	2.839
7	20	20	0	2.839
8	30	30	0	3.026
9	33	33	0	3.026
10	30	30	0	3.136

Cuadro 3.11: Resultado de la instancia J1810

Problema J2010				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	18	18	0	3.401
2	24	24	0	3.526
3	20	20	0	3.447
4	19	19	0	3.541
5	22	22	0	3.416
6	25	25	0	3.588
7	23	23	0	3.525
8	17	17	0	3.526
9	22	22	0	3.463
10	28	28	0	3.51

Cuadro 3.12: Resultado de la instancia J2010

Problema J3010				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	26	26	0	6.786
2	28	28	0	6.77
3	24	24	0	6.598
4	36	36	0	6.646
5	33	33	0	6.708
6	25	25	0	6.77
7	22	22	0	6.677
8	31	31	0	6.77
9	38	38	0	6.677
10	32	32	0	6.583

Cuadro 3.13: Resultado de la instancia J3010

Problema J501				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	33	31	0.0645	33.603
2	27	26	0.0384	36.363
3	27	26	0.0384	53.928
4	35	29	0.2068	61.716
5	28	26	0.0769	72.489

Cuadro 3.14: Resultado de la instancia J501

Problema J502				
Instancia	Sol MA	Sol Optima	Desviación	Tiempo
1	26	25	0.04	26.153
2	27	26	0.038	52.273
3	25	23	0.086	55.382
4	28	25	0.012	24.212
5	25	24	0.041	66.454

Cuadro 3.15: Resultado de la instancia J502

Capítulo 4
Conclusiones y futuras líneas

En esta tesis se ha abordado el problema de secuenciación de proyectos con recursos limitados modo multiple. Se muestran los resultados de nuestra investigación de tal forma que se puedan evidenciar las contribuciones, alcances y futuras líneas de investigación. A continuación se presenta un resumen de los aportes y conclusiones, en concordancia de los objetivos propuestos.

4.1. Aportes

En primer lugar se ha realizado una revisión bibliográfica de los problema de secuenciación de proyectos con recursos limitados modo multiple y de los trabajos más relevantes teórico-prácticos publicados hasta el momento para resolver el MRCPSP. Estos trabajos están basados en métodos metaheurísticos que garantizan soluciones de muy buena calidad para espacios de búsqueda muy grandes.

El enfoque desarrollado en nuestra investigación está basada en los algoritmos meméticos debido a que estos son combinaciones de los algoritmos evolutivos y de búsqueda local, los cuales por separado han contribuido a dar soluciones de alta calidad a problemas de grandes dimensiones.

Se diseñó, codificó y contrastó computacionalmente un algoritmo memético para resolver el MRCPSP, el cual utiliza tres procedimientos de búsqueda local. El primer procedimiento mejora la calidad de las soluciones, el segundo está ligado al operador de mutación de los agentes logrando minimizar el consumo de recursos no renovables y el tercer procedimiento está basado en el cambio de vecindad, realizando movimientos en el espacio de búsqueda para mejorar las soluciones del problema.

El algoritmo memético emplea una generalización del operador de cruce uniforme, basado, principalmente, en propiciar en la población diversidad en el espacio de elección, es decir, seleccionando un padre y una madre, generando cuatro soluciones diferentes, escogiendo las mejores.

Los resultados computacionales del algoritmo memético fueron analizados con las soluciones presentadas por cada conjunto de problemas. El tiempo promedio de computo por instancia factible permite concluir que el algoritmo memético es muy eficiente. El algoritmo memético resuelve las instancias propuestas de manera factible, con cincuenta o menos actividades.

4.2. Líneas futuras de investigación

Basándose en la investigación realizada, se proponen las siguientes líneas para desarrollar futuros trabajos:

- Realizar ajustes en el algoritmo para resolver instancias de manera factible con cien o más actividades.
- Mejorar la búsqueda local implementada en la mutación, la cual permitiría mejorar la calidad de las soluciones.
- Realizar adaptaciones a la búsqueda local VNS, con componentes asociadas al mejoramiento de las soluciones, ya sea en el tiempo de computo o el de las soluciones factibles.
- Implementar la justificación en el algoritmo memético.
- Implementar el preproceso.

Referencias

1. J. Alcaraz and C. Maroto. A robust genetic algorithm for resource allocation in project scheduling. *Ann. Oper. Res.*, 102:83–109, 2001.
2. J. Alcaraz and C. Maroto. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, 54(6):614–626, 2003.
3. J. Alcaraz and C. Maroto. A hybrid genetic algorithm based on intelligent encoding for project scheduling. In *Perspectives in modern project scheduling*, volume 92 of *Internat. Ser. Oper. Res. Management Sci.*, pages 249–274. Springer, New York, 2006.
4. F. Ballestín. *Nuevos métodos de resolución del problema de secuenciación de proyectos con recursos limitados.* Tesis doctoral, Universidad de Valencia., 2002.
5. J. Blazewicz, J. Lenstra, and A. Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11 – 24, 1983.
6. K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268 – 281, 2003.
7. M. Cervantes. *Nuevos métodos metaheurísticos para la asignación eficiente, optimizada y robusta de recursos limitados.* PhD thesis, Universitat Politécnica de Valéncia. Departamento de Sistemas Informáticos y Computación., 2010.
8. N. Christofides, R. Alvarez-Valdes, and J. Tamarit. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, 29(3):262 – 273, 1987.
9. C. Chyu, A. Chen, and X. Lin. Hybrid ant colony approach to multi-mode resource-constrained project scheduling problems with nonrenewable types. *In Proceedings of First International Conference on Operations and Supply Chain Management, Bali*, 2005.
10. R. Conway, W. Maxwell, and L. Miller. *Theory of Scheduling.* Addison-Wesley Publishing Company, 1967.
11. D. Debels and M. Vanhoucke. A bi-population based genetic algorithm for the resource-constrained project scheduling problem. In *Computational Science and Its Applications - ICCSA 2005*, volume 3483 of *Lecture Notes in Computer Science*, pages 378–387. Springer Berlin Heidelberg, 2005.
12. E. Demeulemeester and W. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12):1803–1818, 1992.
13. A. Drexl and J. Gruenewald. Nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 25(5):74–81, 1993.
14. S. E. Elmaghraby and J. Kamburowski. The analysis of activity networks under generalized precedence relations (gprs). *Manage. Sci.*, 38(9):1245–1263, Sept. 1992.
15. W. S. H. Erik L. Demeulemeester. *Project Scheduling - A Research Handbook*, volume 49. Kluwer Academic Publishers, Boston, 2002.

16. S. French. Scheduling and sequencing an introduction to the mathematics of the job-shop. *John Wiley and Sons, New York*, 1982.
17. P. Hansen, N. Mladenovic, J. Brimberg, and J. Pérez. Variable neighborhood search. In *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 61–86. Springer US, 2010.
18. S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Res. Logist.*, 45(7):733–750, 1998.
19. S. Hartmann. *Project scheduling under limited resources: Models, methods, and applications*, volume 478 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 1999.
20. S. Hartmann. Project scheduling with multiple modes: A genetic algorithm. *Ann. Oper. Res.*, 102:111–135, 2001. Project scheduling.
21. S. Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Res. Logist.*, 49(5):433–448, 2002.
22. S. Hartmann and A. Drexl. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32(4):283–297, 1998.
23. S. Hartmann and A. Sprecher. A note on hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, 94(2):377 – 383, 1996.
24. R. Heilmann. A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. *European Journal of Operational Research*, 144(2):348 – 365, 2003.
25. W. Herroelen, E. Demeulemeester, and B. De Reyck. A classification scheme for project scheduling problems. 1998.
26. J. Józefowska, M. Mika, R. Rózycki, G. Waligóra, and J. Weglarz. Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 102(1-4):137–155, 2001.
27. L. Kaplan. *Resource-Constrained Project Scheduling with Preemption of Jobs*. Unpublished ph.d. thesis, University of Michigan., 1988.
28. J. J. Kelley. *The Critical-Path Method: Resources Planning and Scheduling, in Muth, J.F. and G.L.* Thompson (Eds.), 1963.
29. R. Kolisch. *Project Scheduling under Resource Constraints-Efficient Heuristics for Several Problem Classes,*. Physica-Verlag, 1995.
30. R. Kolisch. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14(3):179 – 192, 1996.
31. R. Kolisch and A. Drexl. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29(11):987–999, 1997.
32. R. Kolisch and S. Hartmann. Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In *Project Scheduling*, volume 14 of *International Series in Operations Research & Management Science*, pages 147–178. Springer US, 1999.
33. R. Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, 29(3):249 – 272, 2001.
34. R. Kolisch and A. Sprecher. PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, 96(1):205 – 216, 1996.
35. R. Kolisch, A. Sprecher, and A. Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Manage. Sci.*, 41(10):1693–1703, Oct. 1995.
36. D. L. Job shop scheduling with genetic algorithms. *In Proceedings of the First International Conference on Genetic Algorithms and their applications*, 2005.
37. S. Lawrence. *Resource Constrained Project Scheduling - A Computational Comparison of Heuristic Scheduling Techniques*. Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, 1988.

38. K. Li and R. Willis. An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, 56(3):370 – 379, 1992.
39. A. Lova, P. Tormos, M. Cervantes, and F. Barber. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117(2):302 – 316, 2009.
40. A. L. Lova, M. P. Tormos, and F. Barber. Multi-mode resource constrained project scheduling: scheduling schemes, priority rules and mode selection rules. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 10(30):69–86, 2006.
41. W. M. *A genetic algorithm for resource-constrained scheduling.* Phd thesis, Massachusetts Institute of Technology, 1996.
42. J. Mendes, J. Gonçalves, and M. Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36(1):92 – 109, 2009.
43. 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*, 44(5):714–729, 1998.
44. J. Moder, C. Phillips, and E. Davis. *Project Management with CPM, PERT and Precedence Diagramming.* Van Nostrand Reinhold Company Inc., New York, 3 edition, 1983.
45. P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms. Technical report caltech concurrent computation program, California Institute of Technology, Pasadena, California, USA, 1989.
46. K. Nonobe and T. Ibaraki. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In *Essays and Surveys in Metaheuristics*, volume 15 of *Operations Research/Computer Science Interfaces Series*, pages 557–588. Springer US, 2002.
47. L. Ozdamar. A genetic algorithm approach to a general category project scheduling problem. *Trans. Sys. Man Cyber Part C*, 29(1):44–59, Feb. 1999.
48. V. V. Peteghem and M. Vanhoucke. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. (11/758), Nov. 2011.
49. M. Pinedo. *Planning and scheduling in manufacturing and services*, volume 24. Springer, 2005.
50. A. Pritsker, L. Waiters, and P. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108, 1969.
51. B. D. Reyck and W. Herroelen. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119(2):538 – 556, 1999.
52. M. Speranza and C. Vercellis. Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, 64(2):312 – 325, 1993.
53. A. Sprecher. *Resource-Constrained Project Scheduling-Exact Methods for the Multi-Mode Case.* Lecture Notes in Economics and Mathematics N° 409, Springer, Berlin, Germany., 1994.
54. A. Sprecher, S. Hartmann, and A. Drexl. Project scheduling with discrete time-resource and resource-resource tradeoffs, 1994.
55. A. Sprecher, S. Hartmann, and A. Drexl. An exact algorithm for project scheduling with multiple modes. *OR Spektrum*, 19(3):195–203, 1997.
56. A. Sprecher, R. Kolisch, and A. Drexl. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 80(1):94 – 102, 1995.
57. J. P. Stinson, E. W. Davis, and B. M. Khumawala. Multiple resource-constrained scheduling using branch and bound. *A I I E Transactions*, 10(3):252–259, 1978.
58. F. B. Talbot. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, 28(10):1197–1210, 1982.
59. R. A. Valdés and J. T. Goerlich. The project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Operational Research*, 67(2):204 – 220, 1993.

60. V. M. Van Petghem V. Technical report, faculty of economics, ghent university, A genetic algorithm for the multi-mode resource-constrained project scheduling problem., 2008.
61. J. Weglarz, J. Józefowska, M. Mika, and G. Waligóra. Project scheduling with finite or infinite number of activity processing modes-a survey. *European Journal of Operational Research*, 208(3):177 – 205, 2011.