Universidad de las Ciencias Informáticas

Facultad III

Implementación de un algoritmo híbrido basado en optimización por Enjambre de Partículas para el problema de planificación de proyecto con múltiples modos de procesamiento

TRABAJO PARA OPTAR POR EL TÍTULO DE:

INGENIERO EN CIENCIAS INFORMÁTICAS

Autor:

Jarsey Capetillo Corbea

Tutor:

Lic. Bolivar Ernesto Medrano Broche

Co-Tutora:

Lic. Lytyet Fernández Capestany

Declaración de Autoría

Declaro ser el único autor del presente trabajo y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____del año _____.

_____

Firma del Autor

Jarsey Capetillo Corbea

_____

Firma de la Tutora

Lic. Bolivar E. Medrano Broche

_____

Firma del Co-Tutor

Lic. Lytyet Fernández Capestany

Dedicatoria:

Les dedico mi tesis a mi familia chiquita, antes de nada a mis padres que hacen de mi sueño el suyo también, a mi hermano por darme las fuerzas para esforzarme para ser su inspiración, a Claudia por todo su amor y el mío, y a mis abuelos que sin ellos nada de esto fuera posible.

Agradecimientos

Agradezco a:

A mis padres sobre todas las cosas por haberme dado la vida, por estar siempre a mi lado, por apoyarme en los instantes de angustia, ser mis confidentes, mis amigos, por tener paciencia por mis travesuras y más que nada por haber confiado en mí y tener esa seguridad de que mi prioridad en la vida es no defraudarlos.

A mi hermano por sus interminables preocupaciones, por ser mi parte contra puesta en el transcurso de mi vida, mi alegría diaria, por ser eso: Un Gran Hermano.

A mi novia Claudia por todo, por ayudarme en el transcurso de mi carrera y darme la fe que necesito en los momentos de vacío, por ser más que una novia, una amiga, una compañera de tesis, por sufrir mis tristezas, llorar mis alegrías, y por su apoyo incondicional en todo momento.

A mis tutores por toda la ayuda brindada y las horas dedicadas a mi tesis, se que se esforzaron y dieron todo de sí para poder brindarme sus conocimientos y guiarme durante toda mi investigación.

A mis abuelos por ser mis segundos padres, por ser tan especiales y constantes desde que nací. A mis amigos Héctor, Reynier y Gustavo por su apoyo incondicional en todo momento, ya que sin su granito de arena todo hubiera sido más difícil, gracias por estar siempre ahí sin importar nada, por ser muy buenos amigos.

A mis tíos Haydee, Tony (mi padrino), Puchito y Alexander por estar siempre preocupados por todo y por quererme como otro hijo más.

A mi segunda familia Marielena, Pulido y Renecito por, hacerme la vida más amena y por quererme tanto. A Yamo, David, Danis y Pino que de una forma u otra forman parte de mi vida y han sido incondicionales conmigo. A los trabajadores de la EICMA que me brindaron su ayuda.

A todos muchas gracias por hacer posible este día.

Abstract

The Particle Swarm Optimization (PSO) is a very competent bio-inspired technique for solving optimization problems in continuous search spaces. In recent years PSO has been used successfully to solve combinatorial optimization problems. This work proposes a hybrid algorithm based on PSO and Simulated Annealing for solve to the resources constrained multi mode projects scheduling problem. This algorithm uses PSO strategy for the movement in the search space. The solution space is represented by two sets. On the one hand a set of priority activity vectors. In other hand a set that contains the resource allocations. The particle displacement is performed in both set simultaneously. The changes in the particle position update are accepted through the SA acceptance form. The algorithm performance is tested in instances . The results are compared with variants of PSO and SA without hybridize.

Keyword -: **Particle Swarm Optimization, Hybrid Algorithm, Multi Mode Project Scheduling**

Resumen

En este trabajo se propone un algoritmo híbrido basado en la optimización por Enjambre de Partículas para la solución del problema de planificación de proyecto con múltiples modos de procesamiento. Las partículas en el algoritmo se mueven simultáneamente en dos conjuntos que forman el espacio de solución del problema, el conjunto de los vectores reales que almacenan los valores de las prioridades de las actividades y el conjunto de vectores enteros que contienen la asignación de recursos. El algoritmo emplea el mecanismo de aceptación del Recocido Simulado (SA), para aceptar los rasgos que adquieren las partículas en su actualización. Se emplean instancias de prueba de dieciocho actividades para estudiar el comportamiento del algoritmo propuesto. Los resultados son comparados con los mejores algoritmos reportados en la literatura.

Índice general

Índice de algoritmos

Índice de figuras

Índice de tablas

0.1. Introducción

"La gestión de proyectos se remonta por lo menos 4500 años atrás. Los constructores de las pirámides de Egipto y los templos mayas en América Central a menudo se citan como los primeros gestores de proyecto del mundo" [1]. No tenían computadoras, ni software para auxiliarse en la planificación, ni siquiera conocían PERT (Program Evaluation Review Technique [2]) o CPM (Critical Path Method [3]), que datan de finales de los años cincuenta. Sin embargo, se las arreglaron para ejecutar proyectos excepcionalmente complejos, utilizando las herramientas más simples. En la actualidad en casi todas las ingenierías se gestionan proyectos y su planificación es objeto de atención por parte de administradores e investigadores.

La confección de un cronograma para un proyecto es una tarea sumamente compleja, pues se debe definir la fecha de inicio de cada una de las actividades respetando sus características tecnológicas, que obligan ha que se considere las relaciones de precedencia. También se le deben asignar recursos que en muchas ocasiones están restringidos. Esta es una de las razones por lo que la planificación de un proyecto se trate como un problema de optimización.

El problema de planificación de proyecto con recursos limitados (RCPSP[1]), es un problema de Optimización Combinatoria, que ha sido muy investigado en los últimos cincuenta años, cuenta con gran desarrollo teórico y múltiples aplicaciones en casi todas las ingenierías. Por ello, poder disponer de soluciones algorítmicas eficientes y flexibles supone un alto valor añadido en diferentes entornos de aplicación [1].

El objetivo del RCPSP es hallar el momento de inicio de cada actividad de tal forma que se minimice el tiempo de duración del proyecto, teniendo en cuenta dos tipos de restricciones [4, 5]. Por una parte las relaciones de precedencia, fuerzan a cada actividad a comenzar después de que sus antecesoras hayan concluido. Por otra parte, para procesar una actividad se requieren utilizar recursos, los cuales están disponibles en una cantidad fija y limitada en cada espacio de tiempo. Existen otras variantes más sofisticadas de este problema, que representan situaciones más realistas de la planificación en los proyectos [6].

[1] Resource Constraint Scheduling Problem

El problema de planificación de proyecto con múltiples modos de procesamiento (MRCPSP[2]) es una de las variantes del RCPSP más difíciles de resolver. En este problema se considera la posibilidad de que cada actividad se ejecute en uno de varios modos posibles. Esta característica provoca que la obtención de una asignación de modos factibles sea un problema de la clase NP-duro[7]. Lo anterior es uno de los principales motivos que ha generado gran cantidad de trabajos, que proponen métodos exactos, heurísticos y metaheurísticos para la solución de este problema. Los métodos exactos tienen limitaciones ante instancias grandes, cuando la cantidad de actividades y de recursos es considerable resultan impracticables[8]. Esta situación ha propiciado que prevalezcan los enfoques heurísticos en la solución de este tipo de problema.

En Medrano y colaboradores [9], se propone un algoritmo híbrido basado en Optimización por Enjambre de Partículas (H-PSOSA) para la solución de un problema de planificación de múltiples proyectos de desarrollo de software (SDMPSP). Este problema guarda gran similitud con el MRCPSP porque considera que las actividades se pueden ejecutar de varios modos. Esta es la razón que hace pensar que el algoritmo H-PSOSA, resulta una estrategia eficaz para la búsqueda de soluciones del problema MRCPSP.

Por tanto el ***problema*** de esta investigación radica en como comprobar que el algoritmo H-PSOSA, resulta un método de solución eficaz para el MRCPSP.

El ***objetivo general*** que se ha trazado consiste en: implementar el algoritmo H-PSOSA para el MRCPSP. El ***objeto de la investigación*** es la aplicación de algoritmos metaheurísticos híbridos a problemas de planificación de proyecto y como ***campo de acción*** se tiene la implementación de una metaheurística para la solución de un problema de planificación de proyecto.

El objetivo general fue desglosado en los ***objetivos*** **específicos** siguientes:

1. Elaborar el marco teórico de la investigación.
2. Diseñar el algoritmo H-PSOSA para buscar soluciones del MRCPSP.
3. Evaluar los resultados obtenidos y establecer una configuración de los parámetros del algoritmo.

Se defenderá la idea siguiente: Si se implementa el algoritmo H-PSOSA se obtendrá un método de

[2]Multi Mode Resource Constraint Scheduling Problem

solución eficaz para resolver el MRCPSP.

En esta investigación se emplean los métodos científicos siguientes:

Métodos lógicos: el método *analítico-sintético*, al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta; el método *inductivo-deductivo*, como vía de la constatación teórica durante el desarrollo del trabajo y el método de *modelación* para el diseño y desarrollo del algoritmo.

Métodos empíricos: el método *coloquial* para la presentación y discusión de los resultados en sesiones científicas y el método *experimental* para comprobar la calidad de los resultados obtenidos.

Métodos matemáticos: los métodos *estadísticos* para evaluar la calidad de los resultados obtenidos.

Este trabajo esta organizado de la manera siguiente: En el capítulo 1 se abordan los elementos fundamentales de los problemas de planificación de proyecto (PSP) y se describen las técnicas que se serán empleadas para su solución. En el capítulo 2 se expone el algoritmo diseñado y los detalles tenidos en cuenta en la implementación. En el capítulo 3 se exponen los resultados computacionales de la investigación. Por último se incluyen las conclusiones y las recomendaciones para el trabajo futuro.

1 Marco teórico de la investigación

1.1. Introducción

Los problemas de planificación de proyecto (PSP) por su importancia han suscitado numerosas investigaciones desde la aparición de los trabajos [2, 3, 10, 11] en los años cincuenta. El problema de planificación de proyecto con recursos limitados, también considerado como RCPSP modo simple por [5], es el más tratado de los PSP [6]. Gran parte de los resultados que se han obtenido en el estudio de este problema han sido extendido a las demás variantes, por tanto se hace obligado detallar sus aspectos esenciales.

1.2. Elementos de los problemas de planificación de proyecto.

Los PSP están formados por actividades, recursos, relaciones de precedencia y una función objetivo [12]. A continuación se describen estos elementos.

1.2.1. Actividades

Los proyectos cuentan con un número finito de actividades (tareas, trabajos u operaciones[1]) que hay que ejecutar. Para terminar un proyecto es necesario ejecutar cada actividad de uno o de diversos

[1]Términos generalmente usados para problemas de planificación en máquinas (Job Shop, Flow Shop ,etc.)[13]

modos, donde cada modo representa una forma distinta de realizar las actividades. Un modo determina el tiempo de duración de la actividad, medida en número de unidades de tiempo, que indica el tiempo necesario para completar la actividad. También el modo de procesar una actividad determina los requerimientos de recursos para su ejecución. Cada actividad puede llevar asociada una fecha de entrega, que indica el tiempo máximo en el que se debe haber completado la actividad o parte de ella y una fecha de disponibilidad, que marca el instante de tiempo a partir del cual se puede procesar la actividad. Lo que se debe hallar en un *PSP* es cuándo y de qué modo se va a procesar cada actividad.

1.2.2. Relaciones de precedencia

En los proyectos por diversas razones, la planificación de una actividad está condicionada a la ejecución de otra actividad. Esta relación entre las actividades se denomina relación de precedencia. La forma más simple en que pueden aparecer las relaciones de precedencia es cuando una actividad no puede comenzar hasta que otra(s) finalice(n). Por ejemplo la actividad j sólo se puede ejecutar a partir de que finalice la actividad i, este tipo de relación es de *fin-inicio,* el final de la actividad i restringe el inicio de la actividad j. Si la actividad j comienza inmediatamente después de la terminación de i, se dice que el lapso de tiempo entre actividades es de cero y se denota por $FS_{ij} = 0$. En el caso que $FS_{ij} \neq 0$ se dice que la relación de precedencia es generalizada de tipo *fin-inicio* [1].

Las relaciones de precedencia se pueden visualizar, representando el proyecto mediante un grafo dirigido y sin ciclos, en formato actividad en nodo *AON*[2] donde cada actividad representa un nodo del grafo y cada arco dirigido una relación de precedencia entre dos actividades. Cada arco tiene asociado el tipo de relación de precedencia que existe entre las dos actividades que separa.

1.2.3. Recursos

Para realizar una actividad, generalmente es necesario tener algún tipo de consumo. En los *PSP* se modela ese consumo a través de los recursos que utiliza cada actividad. Los recursos se clasifican según su categoría, tipo y valor [4]. Las categorías más generales de los recursos son las de recursos

[2] *Activity-On-Node*

renovables y no renovables.

1.2.3.1. Recursos renovables

Recursos que sólo están limitados, en un espacio de tiempo del ciclo de vida del proyecto, se dice que están temporalmente disponibles en cada espacio de tiempo. Ejemplo: Mano de obra, herramientas, espacio, etc.

1.2.3.2. Recursos no renovables

Recursos que están limitados a lo largo del ciclo de vida del proyecto, y una vez usados en el procesamiento de una actividad no pueden ser asignados a otra. Ejemplo: presupuesto, materia prima, energía, etc.

1.2.4. Funciones objetivos

Todo problema de optimización requiere de un criterio (función objetivo) para evaluar la calidad de sus soluciones. Según las clasificaciones dadas en [5, 14], cada función objetivo define un modelo distinto, aunque el conjunto de solución sea el mismo.

La minimización de la duración del proyecto (tiempo transcurrido entre el inicio y la terminación del proyecto) es probablemente la función objetivo más tratada y aplicada en el dominio de la planificación de proyecto [4, 6, 15]. Minimizar la duración del proyecto se reduce a minimizar el máximo de los tiempos de finalización de las actividades que forman el proyecto.

Las funciones objetivos que dependen del tiempo en los PSP, son clasificadas como regulares. Una función objetivo *regular,* es aquella que al comparar dos secuencias S y $\hat{S}$ para un problema dado, si S difiere de $\widehat{S}$ únicamente en el tiempo de finalización de una actividad j y se cumple que $f_j < \widehat{f}_j$ para $f_j \in S$ y $\widehat{f}_j \in \widehat{S}$, se puede asegurar que la secuencia con el menor tiempo de finalización en esa actividad es al menos tan buena como la otra secuencia (tomando la evaluación de la función objetivo como criterio) y se dice que S domina a $\hat{S}$. La minimización de la duración del proyecto y la

minimización de las tardanzas son funciones objetivos regulares que dependen del tiempo.

1.3. Problemas de planificación de proyecto

Los PSP cuentan con variantes, que van desde la simple planificación de actividades sin tener en cuenta los recursos, hasta variantes más sofisticadas que consideran varios modos de procesamiento de las actividades, generalización de las relaciones de precedencia, múltiples proyectos de forma simultánea, proyectos con recursos variables, etc [6]. El (RCPSP) modo simple (denotado como $PS|prec|Cmax$ según la nomenclatura de [5] o como $m,1|cpm|Cmax$ según [14]) es uno de los problemas básicos y claves en la planificación de proyecto. Pues a partir del estudio de este problema se han podido desarrollar métodos que han sido extendidos a otras variantes de problemas de Planificación.

1.3.1. Problema planificación de proyecto con recursos limitados

El RCPSP según Brucker y colaboradores[5], consiste en establecer la secuencia de un conjunto $J=\{1,...,n\}$ de actividades de un proyecto sujetas a dos tipos de restricciones, las relaciones de precedencia y la cantidad de recurso disponible para ejecutar las actividades en cada instante de tiempo. Se asumen que los recursos son renovables de diferentes tipos. Como función objetivo se tiene la minimización del tiempo de terminación del proyecto. Todas las magnitudes son asumidas enteras.

1.3.1.1. Representación del RCPSP

Un proyecto es representado por un grafo dirigido sin ciclo $G = (J,E)$ donde J es el conjunto de los nodos (actividades) y $E = \{(i,j) : i \in A_j\}$ el conjunto de los arcos (relaciones de precedencia), donde $A_j \subset J$ es el conjunto de las actividades antecesoras de j y por ende tienen que ejecutarse antes que esta, el conjunto de las actividades sucesoras inmediatas de j denotado por Suc_j Cada actividad j tiene una duración d_j (peso del arco (i,j)) con $d_0, = d_n = 0$ ya que $j = 1$ y $j = n$ son nodos ficticios que son tomados para representar donde comienza y termina el proyecto. Se cuenta con un conjunto de recursos renovables , cada actividad j para procesarse requiere de una cantidad q_{kj} de recursos de

tipo k.

En la figura 1.3.1 se muestra un ejemplo que se denota como $(E1)$ (tomado de [16]) que consta de un proyecto con $n = 7$ actividades y dos recursos del mismo tipo $R_1 = 2$.

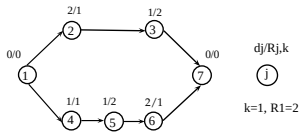

Figura 1.3.1: Red de proyecto (E1)

Dos soluciones de este problema se pueden ver en la figura 1.3.2, la solución representada mediante el diagrama de Gantt (A) tiene duración 7 unidades de tiempo y la solución (B) sólo tiene 5 unidades de tiempo (óptimo para el ejemplo).

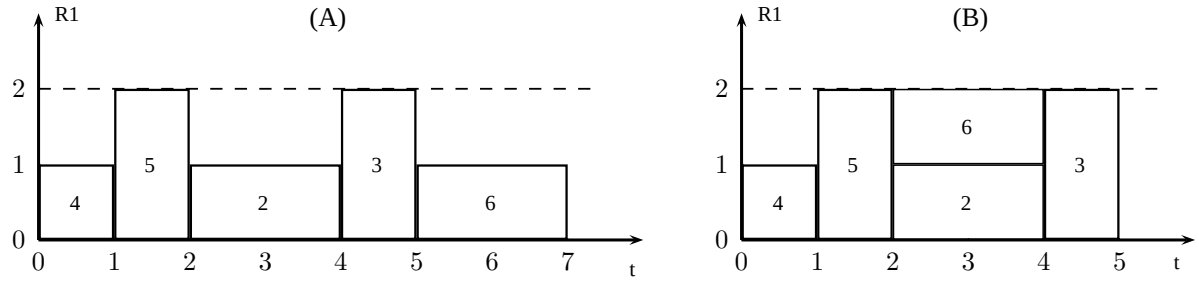

Figura 1.3.2: Soluciones

A lo largo de este trabajo se denota por s_j $\left(f_j\right)$ al tiempo de inicio (fin) de la actividad j en una secuencia o solución $S = (s_1, s_2, ..., s_n)$ y por R_k al conjunto de recursos disponibles del tipo $k \in K$.

Sea $P_t = \left\{ j \in J : s_j \leq t < s_j + d_j \right\}$ el conjunto de las actividades que se están procesando en el instante de tiempo t y $U(S,t) = \sum_{j \in P} q_{jk}$ la cantidad de recursos que consumen las actividades $j \in P_t$ donde q_{jk} es la cantidad de recurso que consume la actividad j. Un modelo conceptual para el *RCPSP* queda de la manera siguiente:

$$min\, s_n = \max_{j \in J} \left\{ f_{j-1} \right\} \tag{1.3.1}$$

Sujeto a:

$$s_j - s_h \geq d_j \ \ \forall h \in A_j, j \in J \tag{1.3.2}$$

$$U(S,t) \leq |R_k|, \ t = 0,1,2,...,T-1, \forall k \in K \tag{1.3.3}$$

$$s_j \geq 0 \ \ \forall j \in J \tag{1.3.4}$$

La expresión (1.3.1) define la minimización de la duración del proyecto, que esta dada por el máximo del tiempo de finalización de la actividad $n-1$ del proyecto. El grupo de restricciones (1.3.2) controla las relaciones de precedencia en la solución S y las restricciones (1.3.3) controlan que no se exceda la cantidad de recurso disponible para cada momento t . Finalmente (1.3.4) define la no negatividad de los tiempos de inicio.

El RCPSP pertenece a la clase de problemas de optimización NP-Hard pues es una generalización del problema clásico Job Shop [1]. Sin embargo, el RCPSP se convierte en un problema resoluble en un tiempo polinomial si se eliminan las restricciones (1.3.3). El problema relajado (denotado $cpm|Cmax$ según [14]) se resuelve utilizando el método de la recursión hacia delante.

Algoritmo 1.1 Recursión hacia delante

1. Inicializar:$s_1 = 0$
2. Para cada actividad $j \in J \setminus \{1\}$
2.1 Calcular $s_j = \underset{h \in A_j}{max}\{s_h + d_h\}$

La solución que aporta este algoritmo es denominada ES, donde cada actividad está planificada lo más temprano posible, teniendo las relaciones de precedencia. En ES el tiempo de inicio de una actividad j se denota como es_j y como $ef_j = es_j + d_j$ al tiempo de finalización más temprano.

Tomando el problema relajado se puede proceder de forma opuesta y obtener una secuencia denominada LS con la misma longitud del proyecto (la de ES), de forma tal que cada actividad se planifica lo más tarde posible . En este sentido se utiliza el método se recursión hacia atrás.

Algoritmo 1.2 Recursión hacia detrás

1. Inicializar:$s_n = \max_{j \in J} \{es_j\}$
2. Para cada actividad $j \in J \setminus \{1\}$
2.1 Calcular $s_j = \min_{h \in P_j} \{s_h\} - d_j, j = n-1, .., 1$

En *LS* el tiempo de inicio de una actividad j se denota como ls_j y como $lf_j = ls_j + d_j$ al tiempo finalización más tardío. A partir de *ES* y *LS* se obtienen las llamadas actividades críticas, que son las que cumplen con que $es_j = ls_j$. Se denomina camino crítico, a todos los caminos del grafo cuya longitud (*CP*) es igual a la de *ES* (y *LS*) medida sumando las duraciones de las actividades (nodos) que lo componen. La longitud de los caminos críticos es una cota inferior de la duración de un proyecto para el RCPSP. La desviación respecto al *CP* se emplea a menudo en la comparación entre métodos heurísticos, dado que para instancias grandes no se conocen la solución óptima.

1.3.2. Problema de planificación de proyecto con recursos limitados con múltiples modos

El problema de planificación de proyecto con recursos limitados y múltiples modos de procesamiento (MRCPSP), puede verse como un problema RCPSP donde las actividades puede ser ejecutadas de acuerdo a la asignación de determinadas cantidades de recursos (modos de procesamiento). Por esta razón se dice que cada actividad $j \in J$ tiene un conjunto $M_j = \{1, ..., m\}$ de modos de procesamiento, de forma tal que una actividad ejecutada en el modo $m \in M_j$ consume r_{jkm} unidades del recurso renovable de tipo $k \in K$ y r^N_{jkm} de recurso no renovable $k \in K^N$. La disponibilidad de recursos renovables de tipo $k \in K$ está dada por R_k y la de recursos no renovables de tipo $k \in K^N$ está dada por R^N_k. Se adopta que la cantidad de recursos de cualquiera de las dos categorías que se consume en el modo m es mayor que la que se consume en el modo $m+1$ $\left(r_{jkm} > r_{jkm+1}\right)$ lo que se traduce en que la duración d_j de cualquier actividad j cumple con $d_{jm} < d_{jm+1}$.Una cota superior $\hat{t}$ de la duración del proyecto en el MRCPSP puede ser calculada usando el algoritmo (1.1) para el problema relajado tomando como tiempo de duración de las actividades el que proporciona el modo con mayor tiempo de procesamiento.

El MRCPSP [1], puede ser formulado de la siguiente manera definiendo la variable binaria $x_{jmt} = \{0,1\}$, tal que $x_{jmt} = 1$ si la actividad j es planificada en el modo m en el instante $t \in T$, con $T_j = \left[ES_j, LS_j\right]$, donde ES_j es el tiempo de inicio más temprano de la actividad j, el cual es calculado resolviendo el problema relajado mediante el algoritmo 1.1, tomando como duración de las actividades la duración del modo que aporta menor duración. El valor LS_j es el tiempo de inicio más tardío en que puede iniciar una actividad j y se calcula de forma análoga pero esta vez se toma como duración de cada actividad j el del modo de mayor de duración. La variable de decisión $x_{jmt} = 0$ en otro caso.

El modelo lineal en enteros quedaría:

$$min \sum_{m \in M_j} \sum_{t \in T_j} t x_{jmt} \quad j = n \tag{1.3.5}$$

sujeto a:

$$\sum_{m \in M_j} \sum_{t \in T_j} x_{jmt} = 1 \quad \forall j \in J \tag{1.3.6}$$

$$\sum_{m \in M_j} \sum_{t \in T_j} x_{jmt} \left(t + d_{jm}\right) x_{jmt} \leq \sum_{m \in M_h} \sum_{t \in T_j} t x_{hmt} \quad \forall j \in A_h, \forall h \in J \tag{1.3.7}$$

$$\sum_{j \in J} \sum_{m \in M_j} r_{jmk} \sum_{q \in Q} x_{jmq} \leq R_k \quad k \in K \tag{1.3.8}$$

$$\sum_{j \in J} \sum_{m \in M_j} r_{jkm} \sum_{t \in T_j} x_{jmt} \leq R_k^N \quad k \in K \tag{1.3.9}$$

$$x_{jmt} \in \{0,1\} \tag{1.3.10}$$

La expresión (1.3.5) representa la minimización de la duración del proyecto. El grupo de restricciones (1.3.6) controla que todas las actividades sean ejecutadas en algún modo, el grupo (1.3.7) de

restricciones garantizan que no se violen las relaciones de precedencia y (1.3.8) son las restricciones que controlan que no se exceda la cantidad de recurso renovable de tipo $k \in K$ disponible para cada momento $t \in Q$ donde $Q = \left[max\left\{t-d_{jm}, ES_j\right\}, min\left\{t-1, LS_j\right\}\right]$ y $t = 1,2,...,\hat{t}$. El valor de $\hat{t}$ es una cota superior del proyecto y puede ser calculada por el valor de la ruta crítica del problema relajado tomando como tiempo de procesamiento de las actividades el modo que mayor duración imprime a las actividades. El grupo de restricciones (1.3.9) es la que controla que no se exceda la cantidad de recurso no renovable disponible. El grupo (1.3.10) fuerza a la variable de decisión a tomar valores binarios $\{0,1\}$.

Ahora una solución del MRCPSP esta dada por dos vectores un $S = (s_1, s_2, ..., s_n)$ con el tiempo de inicio de cada actividad y otro vector $\mu(S) = (\mu_1, \mu_2, ..., \mu_n)$, μ_j es el modo en que se procesa la actividad j.

1.3.3. Esquema de generación de soluciones

Los esquemas de generación de soluciones (SGS) son el núcleo de la mayoría de los procedimientos heurísticos para la solución del RCPSP [17] pues permiten construir una solución factible. Los SGS construyen una solución factible extendiendo paso a paso una secuencia parcial de actividades, que inicialmente asigna el tiempo de inicio $s_1 = 0$ a la actividad 1 de la secuencia. Una secuencia parcial es una secuencia donde únicamente un subconjunto de las n actividades de un proyecto ha sido planificada. Existen dos SGS diferentes, Serie (SSGS) y Paralelo (PSGS). En ambos casos se permiten encontrar soluciones factibles para el RCPSP. Según Kolisch en [18], SSGS construye soluciones activas y tiene mejor ejecución ante instancias grandes[3], en cambio PSGS genera soluciones sin retraso, el conjunto de soluciones sin retraso es un subconjunto de las soluciones activas.

1.3.3.1. Tipos de soluciones

Las soluciones *activas* son aquellas donde una actividad no puede comenzar a ejecutarse más tarde de lo que lo hace, sin que retrase el inicio de otra. En soluciones *sin retraso*, ninguna de las actividades

[3]Un instancia es considerada grande cuando excede las 30 actividades.

puede comenzar antes de lo que lo hace sin retrasar otra actividad. El conjunto de las soluciones sin retraso, tiene un cardinal menor que el de las soluciones activas. En [16] se demuestra que cuando se tiene un RCPSP donde se minimiza la duración del proyecto, la solución óptima del problema es una solución activa, lo que representa una ventaja importante al usar SSGS.

1.3.3.2. Esquema de generación de soluciones en serie

El esquema de generación de soluciones en serie (SSGS) según [17], consiste en $g = 0, 1, 2, ...$ estados en los cuales las actividades de un proyecto son planificadas sin violar las relaciones de precedencia y las restricciones de recursos. Asociado a cada estado g existen dos conjuntos disjuntos de actividades. Un conjunto $D_g = \left\{ j \in J \setminus S_g : A_j(t) \subseteq S_g \right\}$ de las actividades que pueden ser seleccionadas para planificarse, donde $A_j(t)$ es el conjunto de las actividades antecesoras de la actividad $j \in J$ planificadas en el instante $t = 1, ..., T$. El otro conjunto Sec_g, es el de las actividades que ya han sido planificadas en el estado g. Se tiene que $\hat{R}_k(t) = |R_k| - U_k(S,t)$ es el número de recursos del tipo $k \in K$, disponibles en el instante t, y que $F_g = \{f_j : j \in S_g\}$ es el conjunto de los tiempo de fin de todas las actividades que se han planificado.

Algoritmo 1.3 Esquema de Generación Soluciones en Serie (*SSGS*)

1. Inicializar:$g = 0$,$Sec_0 = \{1\}$
2. Mientras $S_g \neq J$
2.1 $g = g + 1$
2.2 Calcular: D_g, F_g, $\hat{R}_k(t)$ tal que $t \in F_g, k \in K$.
2.3 Seleccionar: $j \in D_g$
2.4 Calcular: $ES_j = \underset{h \in A_j(t)}{max} \{f_h\}$
2.5 Calcular: $s_j = \underset{h \in A_j}{min} \left\{ t : t \geq ES_j, r_{jk} \leq \hat{R}_k(\tau), \forall k \in K, \tau \in \left[t, t + d_j\right] \cap F_g \right\}$
2.6 Calcular: $f_j = s_j + d_j$
2.7 Insertar: $Sec_g = Sec_{g-1} \cup \{j\}$
2.8 $C_{max} = \underset{h \in A_n(t)}{max} \{f_h\}$Fin

El SSGS posee una complejidad $\mathcal{O}(n^2 k)$ [18], a continuación se muestra el funcionamiento de SSGS para el ejemplo $E1$ (Tabla 1.1).

g	1	2	3	4	5
D_g	$\{2,4\}$	$\{2,5\}$	$\{2,6\}$	$\{6\}$	$\{3\}$
j	4	5	2	6	3
F_g	$\{0\}$	$\{0,1\}$	$\{0,1,2\}$	$\{0,1,2,4\}$	$\{0,4,1,2,4\}$
Sec_g	$\{1\}$	$\{1,4\}$	$\{1,4,5\}$	$\{1,4,5,2\}$	$\{1,2,4,5,6\}$

Tabla 1.1: Funcionamiento de SSGS

En el estado inicial $(g = 0)$, se inserta la actividad ficticia $j = 1$ en el conjunto Sec_g y se le asigna tiempo de inicio y tiempo de finalización 0. Al principio de cada estado se calcula el conjunto de las actividades elegibles D_g, el conjunto de los tiempos de finalización F_g y las disponibilidades de recursos restantes $\hat{R}_k(t)$ (paso 2.2). Después de ello se selecciona la actividad $j \in D_g$ (paso 2.3). El tiempo de inicio de la actividad j se calcula determinando primero su tiempo de inicio más temprano ES_j que es igual al mayor tiempo de finalización de sus antecesoras (paso 2.4), luego se calcula el tiempo de inicio más temprano según la disponibilidad de los recursos, finalmente el tiempo de inicio de j está dado por el máximo entre ambos tiempos de inicio. El tiempo de finalización de la actividad j es hallado sumando al tiempo de inicio s_j la duración d_j (paso 2.6). Finalmente se inserta la actividad j en la secuencia parcial.

1.3.4. Representación de soluciones

En[17], aparecen varias propuestas para representar las soluciones del RCPSP. Las diferentes representaciones condicionan las técnicas que se van a poder emplear dentro de un algoritmo, que esté basado en alguna de estas representaciones. Al usar una representación es imprescindible tener en cuenta los elementos siguientes:

- decodificador (en la mayoría de los método heurísticos el decodificador es SSGS),
- la manera de obtener una solución a partir de la representación, el costo computacional de obtenerla, el subconjunto de soluciones factibles que se puede alcanzar,
- la calidad de ese subconjunto y si siempre contiene algún óptimo,
- la redundancia (número de representaciones diferentes que codifican la misma solución).

1.3.4.1. Lista de actividades

Una lista de actividades $\mathscr{L}$ es una permutación del conjunto $J = \{1,...,n\}$ de actividades de un proyecto de forma tal que se cumplan las relaciones de precedencia, es decir que a cada actividad le corresponde una posición u orden en la lista, mayor que el de cualquiera de sus antecesoras. En particular el primer y último elemento de la lista están dados por $\ell_1 = 0$ y $\ell_n = n$ respectivamente. Una secuencia activa admite más de una representación como lista de actividades, a priori, no es controlable la redundancia porque, dada una lista de actividades $\mathscr{L}$, no se sabe que solución codifica antes de secuenciarla.

1.3.4.2. Vector de prioridades

Esta representación está basada en un vector de números (habitualmente reales y en múltiples ocasiones en $[0,1]$), $\Pi = (\pi_1, \pi_2, ..., \pi_n)$, donde se asigna un número π_j a una actividad $j \in J$. La obtención de una solución a partir de un vector de prioridades se realiza, empleando un cualquier *SGS*, seleccionando en el conjunto de las elegibles (D_j) en cada etapa g aquella actividad j con el π_j mayor (o menor según el caso). Los valores de las componentes de Π, se calculan a través de alguna regla de prioridad (ver sección (2.3)). Por tanto, el orden en que se secuencian las actividades se determina por la importancia relativa de las actividades en el proyecto indicada por π_j.

1.3.5. Reglas de prioridad

Una regla de prioridad no es más que una estrategia para asignar prioridades entre las actividades de un proyecto. La regla de prioridad define un procedimiento para seleccionar cada actividad del conjunto de actividades elegibles D_g en cada estado g del algoritmo (1.3).

Una regla de prioridad esta formada por tres componentes, el primero es la función que asigna la prioridad π_j a cada $j \in D_g$, el segundo es la forma de seleccionar el extremo del conjunto D_g, bien puede ser la que posea menor o mayor valor π_j, el tercero sería la forma de desempatar dos actividades con igual valor π_j , se suele desempatar eligiendo la de menor etiqueta primero o aleatoriamente.

Las reglas de prioridad pueden ser clasificadas de dos maneras, según la información que se emplea para calcular el valor π_j de una actividad (reglas basadas en la red del proyecto, en los recursos, en las actividades, en la ruta crítica) o según las veces que actualizan el valor de π_j en la ejecución de SGS empleado (reglas dinámicas si el valor de π_j cambia durante las iteraciones del SGS, estática si no cambia el valor π_j) [1]. Las reglas, actividad con menor tiempo de procesamiento (SPT), actividad con mayor tiempo de procesamiento (LPT), actividad con mayor cantidad de sucesores (MTS), actividad con mayor rango y peso posicional (GRPW) y actividad con la mayor razón de utilización de los recursos y precedencia ponderadas (WRUP), son reglas de las más empleadas en la literatura [19].

La regla MTS consiste en elegir primero aquella actividad $j \in D_g$ que posee un mayor número de actividades sucesoras, ya que el retraso de esta retrasaría a todos sus sucesores. El valor de la prioridad se calcula por $\pi_j = |Suc_j|$, donde Suc_j es el número total de sucesores inmediatos de la actividad j. Otra regla que es un caso particular de MTS es la regla NIS la cual solo considera el número de sucesores inmediatos de la actividad j el valor de $\pi_j = |SI_j|$. La regla GRPW consiste en elegir primero a la actividad $j \in D_g$ con mayor peso posicional, el cual se obtiene sumando la duración de esta actividad y la duración de todas sus antecesoras. La prioridad queda calculada de $\pi_j = d_j + \sum_{h \in Suc_j} d_h$.

1.4. Algoritmo Recocido Simulado

El algoritmo Recocido Simulado (SA), fue introducido por Kirkpatrick, Gelatt y Vecchi en 1983 [20]. Este procedimiento se basa en la analogía del comportamiento del proceso de recocido físico de un sólido, que consiste en el calentamiento del sólido hasta su fundición, seguido de un enfriamiento hasta que cristalice en un estado de cristal perfecto [21, 22].

1.4.1. Estructura del SA

Sea $min\{F(x) : x \in X\}$ un problema de optimización, donde X es el conjunto de soluciones factibles del problema y $F : x \to \mathbb{R}$ es una función objetivo. Comenzando por una solución inicial s se genera

una solución $x_1 \in N(x_0)$ donde $N(x)$ es el entorno de cada solución x . Si $F(x_1) < F(x_0)$ se acepta x_1 y se continua el proceso con x_1. En otro caso, si $F(x_1) > F(x_0)$, se acepta s' según cierta probabilidad. Esta probabilidad, depende de la diferencia $\Delta = F(x_1) - F(x_0)$, y de un parámetro denominado temperatura T, que varía durante la ejecución del algoritmo. Cuando mayor es Δ, menor es la probabilidad de aceptación, y cuando menor es la temperatura más difícil es aceptar una solución que no mejore. Este mecanismo permite de cierta manera escapar de óptimos locales etapas tempranas respecto al entorno definido. La temperatura general se inicializa en valor alto, al comenzar el algoritmo de forma tal que se permita la aceptación de soluciones que empeoren con cierta frecuencia. A medida que transcurre la ejecución del algoritmo se reduce la temperatura paulatinamente, lo que permite que la probabilidad de aceptar soluciones que empeoren decrezca. Como criterio de parada habitualmente está el definir cierto valor mínimo de temperatura y detener el algoritmo cuando este sea alcanzado, o simplemente establecer un número de iteraciones donde se garantice que la temperatura tenga un valor suficientemente bajo. Un esquema general del algoritmo sería el siguiente:

Algoritmo 1.4 Algoritmo Recocido Simulado

1. Inicialización:
1.1 Inicializar: CI, T , $n = 0$
1.2 Generar una solución inicial: $x_n \in X$
1.3 Evaluar *:* $F(x_n)$
1.4 Mejor solución*:* $\hat{x} = x_n$
2.1 Mientras no se cumpla condición de parada ($n < CI$ o $T < \varepsilon$) :
2.1 $n = n + 1$
2.2 Generar una solución: $x_n = N(x_{n-1})$
2.3 Evaluar $f(x_n)$
2.4 Calcular $\Delta F = F(x_n) - F(x_{n-1})$
2.5 Si $\Delta F < 0$ o $exp(-\frac{\Delta F}{T}) > U(0,1)$
2.5.1 Aceptar: x_n
2.5.2 Si $F(x_n) < F(\hat{x})$
2.5.3 Actualizar mejor solución: $\hat{x} = x_n$
2.5.4En otro caso: $x_n = x_{n-1}$
2.6 Actualizar: $T = \alpha T$
3. Devolver :$\hat{x}$ FIN

En el algoritmo CI representa la cantidad de iteraciones. La expresión $T_n = \alpha T_{n-1}$, donde $\alpha \in (0,1)$ y $n = 1, ..., CI$ permite que la temperatura decrezca con cada iteración del algoritmo. La condición de parada del algoritmo puede ser la cantidad de iteraciones o que la temperatura alcance un valor

predeterminado (suficientemente bajo).

1.4.2. SA para problemas de planificación de proyecto

El SA ha demostrado ser una herramienta exitosa para resolver Problemas de Optimización Combinatoria de variadas estructuras. Los problemas de Scheduling son de los problemas que han sido resueltos con SA . En [23], se aplica SA al Job Shop teniendo en cuenta la minimización del makespan, donde se demuestra la convergencia en probabilidad al óptimo global, en este trabajo se consideró como generador de entornos, el intercambio de dos actividades adyacentes en la ruta crítica. La primera adaptación del SA al RCPSP y al MRCPSP[4] fue la realizada en [24], los resultados computacionales de este trabajo no fueron comparados con los métodos más competitivos, ni se realizaron experimentos con instancias grandes. Otras aplicaciones importantes de SA a problemas de PSP son [25, 26, 27]. Los mejores resultados alcanzados hasta el momento por el SA en RCPSP y MRCPSP se reportan en [28], en este trabajo los autores tomaron en cuenta las características del espacio de solución para guiar la búsqueda, la configuración de los parámetros es dependiente del problema, el mecanismo de enfriamiento es dinámico, compuesto por múltiples cadenas de enfriamiento. La condición de parada del algoritmo es la cantidad de iteraciones, que es proporcional con la dimensión del problema.

1.5. Optimización por Enjambre de Partículas.

La Optimización por Enjambre de Partículas (PSO) es una metaheurística poblacional que fue introducida por Kennedy y Eberhart en 1995 [29]. PSO desde su surgimiento cuenta con múltiples aplicaciones en la solución de problemas de optimización de una variada naturaleza [30, 31] . El algoritmo original de PSO está inspirado en el comportamiento social de los individuos de algunas especies de animales, cuando buscan en conjunto posiciones favorables en un área (por ejemplo: una bandada de aves buscando alimento o un sitio para anidar) [32]. PSO simula el movimiento realizado por los individuos (partículas) de la bandada, donde cada uno es un agente simple que se mueve in-

[4]Multi-Mode Resource Constrained Project Scheduling Problem

fluenciado por una componente personal (el mejor lugar visitado) y una componente social (el mejor lugar visitado por algún vecino).

1.5.1. Descripción de PSO para problemas continuos

Sea $min\{F(x) : x \in X\}$ un problema de optimización, donde $X \subset \mathbb{R}^n$ es el conjunto de soluciones factibles del problema y $F : x \to \mathbb{R}$ es una función objetivo. Al establecer la analogía con un problema de optimización en un espacio continuo, una partícula i del enjambre, en cada instante de tiempo $t = 1, 2, ..., CI$ (iteración) está formada por tres vectores. Un primer vector $x_i^t \in X$ representa la posición de la partícula en el conjunto de solución, un segundo vector $x_p^t \in X$ guarda la mejor posición visitada por la partícula i hasta el momento t y el otro vector $v_i^t \in X$ es de dominado velocidad en el momento t .

La velocidad es calculada empleando información sobre la velocidad histórica de la partícula, la posición con mejor posición que ha sido visitada por la partícula (información personal) y la mejor posición que ha visitado alguna de las partículas de su vecindad (información social). La vecindad de una partícula i, es el grupo de partículas, con que la partícula tiene conexión para intercambiar información. La posición de cada partícula es actualizada por las expresiones:

$$v_i^t = \varpi \cdot v_i^{t-1} + \underbrace{U(0,\phi_1) \cdot \left(x_p^t - x_i^{t-1}\right)}_{Influencia\ personal} + \underbrace{U(0,\phi_2) \cdot \left(x_g^t - x_i^{t-1}\right)}_{Influencia\ social} \tag{1.5.1}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{1.5.2}$$

En (1.5.1) se tiene que x_g^t representa la mejor posición que ha alcanzado una partícula en el enjambre las expresiones $U(0,\phi_a), a = 1,2$ son números aleatorios uniformemente distribuidos en el intervalo $[0,\phi_a]$ generados en cada iteración para cada partícula. Los parámetros ϕ_1 y ϕ_2 determinan la magnitud de las fuerzas aleatorias en la dirección de x_p^t y x_g^t respectivamente . A menudo estos parámetros se denominan coeficientes de aceleración y tienen gran influencia en el movimiento de la partícula. El parámetro ϖ, denominado la ponderación de la inercia (*inertia weigth*) se introduce para limitar la

velocidad histórica de la partícula, fue propuesto por Shi y Eberhart en [33], pues no estaba incluido en la primera versión de PSO. Usualmente se toma entre los valores de 0,4 y 0,9 [34, 35], cumpliendo con $\varpi > (\phi_1 + \phi_2) - 1$ para asegurar trayectorias convergentes de las partículas [36]. La expresión (1.5.2) permite actualizar la posición de la partícula en cada momento t. Un esquema general del algoritmo de la versión original de PSO quedaría de la forma siguiente:

Algoritmo 1.5 Optimización por Enjambre de Partículas

1. Inicializar: $t = 0$, SS, CI, ϖ, ϕ_1,ϕ_2
1.1 Para la cantidad de partículas*:* SS
1.2 Generar posición aleatoria: $x_i^t \in X$
2. Mientras no se cumpla la condición de parada: $t < CI$ o $f\left(x_g^t\right) \leq \varepsilon$
2.1$t = t + 1$
2.2 Calcular: mejor posición obtenida por la partícula: x_p^t
2.3 Calcular la mejor posición obtenida por el enjambre:x_g^t
2.4 Para la cantidad de partículas*:* $i = 1, ..., SS$
2.5 Calcular la velocidad y actualizar la posición según (1.5.1) , (1.5.2)
3. La mejor solución es*:* x_g^t FIN

En el algoritmo, SS y CI son el número de partículas y la cantidad de iteraciones respectivamente, a menudo SS se elige empíricamente teniendo en cuenta la dimensión del problema, generalmente se encuentra en el rango de $20 - 50$ y CI es definida según la naturaleza del problema [37]. En [38, 39], se propone el uso del factor de constricción denotado por χ el cual transforma la ecuación (1.5.1) en :

$$v_i^t = \chi \cdot \left(v_i^{t-1} + U(0, \phi_1) \cdot \left(x_p^t - x_i^{t-1}\right) + U(0, \phi_2) \cdot \left(x_g^t - x_i^{t-1}\right)\right) \tag{1.5.3}$$

El factor de constricción χ está definido en función de ϕ_1,ϕ_2 y es calculado por la expresión

$$\chi = \frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}} \tag{1.5.4}$$

Donde, $\varphi = \phi_1 + \phi_2$ y debe cumplir con $\varphi > 4$. Este resultado puede ser considerado como una regla práctica de limitar v_i^t y x_i^t [36, 37].

1.5.1.1. Topologías

Las partículas en el enjambre forman una estructura para el intercambio de información, que se conoce como topología del enjambre. Las más usuales son las denominadas Gbest y Lbest [40].

En la topología Gbest todas las partículas del enjambre intercambian información entre sí, se mantiene una sola partícula x_g^t como la mejor del enjambre por esta razón la vecindad de una partícula es todo el enjambre. Esta estructura procura una rápida convergencia [40, 34]. Eventualmente la atracción que ejerce la partícula x_g^t a al resto puede traer consigo convergencia prematura a óptimos locales si no se actualiza regularmente la partícula x_g^t [41]. Se tiene que la mejor partícula del enjambre es aquella que cumple con:

$$x_g^t = \left\{ x_i^t : F\left(x_i^t\right) = \min_i \left\{ F\left(x_i^t\right) \right\} \right\} \tag{1.5.5}$$

En la topología Gbest cada partícula intercambia información con l partículas, cada partícula tiene su vecindad compuesta por l partículas (usualmente se toma $l = 3$ [34]) y cada vecindad su mejor partícula x_g^t. Esta estructura previene la convergencia prematura y la atracción ejercida por los los óptimos locales . Tiene una convergencia más lenta que Gbest [41]. La vecindad de una partícula puede ser definida como:

$$N_i = \left\{ x_{i-l}^t, x_{i-l+1}^t, ..., x_{i-1}^t, x_i^t, x_{i+1}^t, ..., x_{i+l-1}^t, x_{i+l}^t \right\} \tag{1.5.6}$$

La mejor partícula de la vecindad N_i se define como se define como:

$$x_g^t = \left\{ x_i^t : F\left(x_i^t\right) = \min_i \left\{ F\left(x_i^t\right) : x_i^t \in N_i \right\} \right\} \tag{1.5.7}$$

Los algoritmos PSO que consideran Gbest presenta mejores desempeños en problemas con funciones objetivos unimodales. En el caso que la función objetivo sea multimodal, Gbest proporciona mejores resultados a pesar de su convergencia lenta [32].

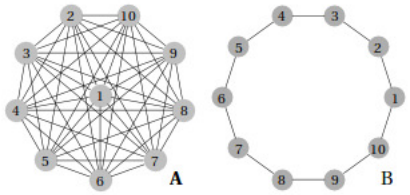

Figura 1.5.1: Topologías del Enjambre Gbest (A) y Lbest (B)

1.5.1.2. Descripción de PSO para el problema binario

El algoritmo PSO binario fue propuesto por Kennedy y Eberhart en [42]. En este algoritmo es concebido para el problema de optimización de la forma $min\{F(x) : x \in \{0,1\}^n\}$. Una partícula al igual que la versión de PSO para el problema continuo es representada por una terna de vectores, posición $x_i^t \in \{0,1\}^n$, mejor solución alcanzada por la partícula $x_p^t \in \{0,1\}^n$ y velocidad $v_i^t \in \{0,1\}^n$. La velocidad es calculada usando (1.5.1) y la posición es actualizada de la manera siguiente:

$$x_{ij}^t = \begin{cases} 1 & si\, sig(v_{ij}^t) > U_j(0,1) \\ 0 & si\, sig(v_{ij}^t) \leq U_j(0,1) \end{cases} \tag{1.5.8}$$

Donde $sig(v_{ij}^t) = \frac{1}{1+e^{(v_{ij}^t)}}$, puede ser vista como la probabilidad de que la $j-esima$ componente el valor 0 o 1. En (1.5.4) para evitar que $x_{ij}^t = 0$ o $x_{ij}^t = 1$ en todas las componentes de la posición, caso posible si $v_{ij}^t \leq -10$ o $v_{ij}^t \geq 10$ respectivamente, se recomienda restringir la velocidad al intervalo $[-4,4]$ [36].

1.5.2. PSO en problemas de planificación de proyecto

Kolisch y Hartman en [43] realizan un amplio estudio de los métodos heurísticos para el RCPSP y no aparecen incluidos trabajos sobre PSO, por tanto se puede afirmar que son recientes las incursiones en este campo.

Las primeras publicaciones sobre la aplicación de PSO al RCPSP estuvieron a cargo de Zhang y co-

laboradores en [44, 45, 46]. En estos trabajos las soluciones del RCPSP, se representaron mediante un vector de prioridades, el cual es tomado como la posición de la partícula. Inicialmente se genera aleatoriamente varias soluciones (partículas) del RCPSP, y se buscan x_p^t y x_g^t**.** En cada iteración t de PSO se toma cada vector de prioridades y se actualiza mediante (1.5.1), luego cada posición nueva (vector de prioridades) es codificado en una solución factible del RCPSP mediante PSGS, se actualizan los x_p^t y x_g^t, la condición parada es la cantidad de iteraciones o una cantidad determinada sin actualizar la mejor partícula. No se refieren de forma explícita a la topología de enjambre usada aunque se asume que sea *Gbest* por descripción del procedimiento.

Recientemente Ruey y colaboradores [47, 48], han propuesto un procedimiento muy competitivo, que utiliza PSO como estrategia para la generación de prioridades en una heurística que se basa en SSGS mejora de búsqueda local. Un uso similar de PSO fue reportado en [49].

En [50], adaptan al RCPSP el procedimiento propuesto para TSP por Clerc en [51], para las instancias de PSPLIB[5] obtiene resultados comparables con los algoritmos más competentes de la literatura . No hace alusión a la topología usada.

Deng y colaboradores en [52], usaron un enfoque PSO para el problema RCPSP, para representar las soluciones usan independientemente, lista de prioridad, vector de reglas de prioridad y vector de prioridades. Al mismo tiempo diseñan una variante de PSO para cada uno de de las representaciones empleadas. Su algoritmo para el caso de la representación de la solución como lista de actividades consiste en usar los coeficientes $U(0,\phi_a)$, $a = 1,2$ presentes en (1.5.1) para definir el punto de cruzamiento entre x_i^t y las partícula $x_{p_i}^t$ y x_g^t respectivamente. Tomando la mejor partícula resultante como x_i^{t+1}, luego utilizan la búsqueda local para mejorar la posición de esta partícula. Al usar el vector de reglas de prioridades y el vector de prioridades, emplean un procedimiento similar al presentado por Zhang y colaboradores, para actualizar velocidad y las partículas. En todos las variantes de PSO usan como codificador SSGS y aplican el proceso de justificación de la solución (ver en [53]). La variante de PSO con operador de cruzamiento fue extendida también al RCMPSP en [52, 54].

Otro resultado interesante es el que proponen Czogalla and Fink en [55], donde emplean la lista

[5]Librería de instancias, públicas en :129.187.106.231/psplib/

de actividades para representar la solución y algoritmo genético con operadores de cruzamiento de dos puntos (aleatoriamente elegidos) y mutación combinado con estrategia PSO en los cruzamientos. Utilizan x_i^t , x_p^t y x_g^t como padres, luego eligen el orden de cruzamiento aleatoriamente, y toman los coeficientes de $U(0,\phi_a)$, $a = 1,2$ para determinar el impacto de x_p^t y x_g^t la solución hija, posteriormente con un procedimiento de permutación mutan el hijo que es elegido como x_i^{t+1}. Como codificador de las soluciones se utiliza SSGS. En este trabajo se combinan las topología Gbest y Lbest, dando lugar a una topología jerárquica. Los resultados obtenido por estos autores figuran entre los mejores de obtenidos en las instancias de PSPLIB.

De forma general la mayoría de las aplicaciones de PSO al RCPSP aparecen como algoritmos híbridos donde se combina las estrategias de PSO y algoritmos genéticos y estrategias de mejora de las soluciones.

1.6. Microsoft Visual Studio 2010

El Microsoft Visual Studio 2010 es un entorno de desarrollo integrado (IDE por sus siglas en inglés) para sistemas operativos Windows, que ha sido empleado en este trabajo. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Esta herramienta permite realizar tareas de modelado, codificado, prueba o depuración, sin salir del entorno, brindando alta eficiencia y productividad. Brinda además la capacidad de usar múltiples monitores, así como desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo. Posee una edición que compila las características de todas las ediciones comunes de Visual Studio: Professional, Team Studio, Test, conocida como Visual Studio Ultimate. Microsoft Visual Studio 2010 incluye nuevos productos para entornos de Prueba y Control de Calidad, como son Visual Studio Test Professional 2010, conjunto especializado de herramientas que aseguran una calidad profesional en la planificación y ejecución

de pruebas. Las funcionalidades para la Gestión del Ciclo de Vida de las Aplicaciones (ALM), garantizan crear soluciones de calidad y a la medida, a la vez que se reducen el costo y el tiempo de su desarrollo, independientemente del tamaño del equipo.

1.6.1. Lenguaje C# 4.0

El C# 4.0 es el lenguaje de programación usado para implementar el algoritmo en esta investigación. Es simple, con seguridad de tipos y orientado a objetos. Una diferencia de este enfoque orientado a objetos respecto otros lenguajes como C++ es que C# no admite funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código [56]. A través de sus diversas innovaciones, se pueden desarrollar aplicaciones rápidamente, manteniendo la elegancia de los lenguajes de tipo C. Entre otras de sus principales características se encuentra la portabilidad de su código, al contar con tipos de datos fijos e independientes del sistema operativo, o máquina para la cual se compile, incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas.

2 Métodos de solución para MRCPSP

En este captítulo se incluyen los elementos que conforman el algoritmo H-PSOSA. Se describen la estructura de la solución y el algoritmo para construir las soluciones factibles. También se expone la estrategia de desplazamiento del algoritmo y la hibridación del PSO con el SA.

2.1. Representación de las soluciones del MRCPSP

Las soluciones del MRCPSP son representadas a partir de dos vectores. Un vector de prioridad denotado por $\Pi = (\pi_1, \pi_2, ..., \pi_n)$, donde cada una de sus componentes π_j contiene el valor de prioridad de una actividad asignado por alguna regla. El otro vector denotado por $M = (\mu_1, \mu_2, ..., \mu_n)$ contiene en sus componentes μ_j el modo de procesamiento asignado a cada actividad j del proyecto. Se decidió emplear el vector de prioridades con el objetivo de tener las soluciones representadas por un vector real y poder diseñar un algoritmo que emplee el desplazamiento usual de PSO en $\mathbb{R}^n$. El vector de prioridad puede ser construido de acuerdo a las reglas siguientes adaptadas al MRCPSP:

Regla de prioridad (rand) el valor π_j de la prioridad de cada actividad es generado aleatoriamente en el intervalo $\pi_j = U(0,1)$. Se planifica primero la actividad que posea menor valor π_j.

Regla de prioridad NIS el valor de la prioridad es calculado por $\pi_j = |Suc_j|$ donde Suc_j es el conjunto de las actividades sucesoras inmediatas de la actividad j. Se planifica primero la actividad con mayor valor de π_j.

Regla de prioridad GRPW el valor de la prioridad es calculado por la expresión:

$$\pi_j = \frac{1}{|M_j|} \sum_{m \in M_j} d_{jm} + \sum_{h \in Suc_j} \frac{1}{|M_j|} \sum_{m \in M_j} d_{ihm} \tag{2.1.1}$$

Donde M_j es el conjunto de los modos de procesamiento de la actividad j. Se planifica primero la actividad con mayor valor de π_j

La regla de prioridad SPT y el valor de la prioridad es calculado por la expresión $\pi_j = \sum_{m \in M_j} d_{ijm}$. En el caso de SPT se planifica primero la actividad con menor valor π_j y en la regla LPT se planifica primero la actividad con mayor valor de π_j.

2.2. Heurística para la generación de soluciones del MRCPSP

La representación (Π, M) es codificada en una solución factible mediante una heurística basada en el procedimiento SSGS descrito en (1.3.5.2) y propuesta en [7]. A continuación se muestran todos los detalles de este algoritmo.

Sea:

S: conjunto de actividades que tienen un modo de procesamiento asignado.

D: conjunto de actividades que no tienen asignado modo de procesamiento.

cR_k^N: cantidad de recurso no renovable de tipo k disponible $cR_k^N = R_k^N - \sum_{j \in J} r_{j\mu k}$

Γ_{jm}: consumo relativo de recursos no renovables de la actividad j en modo m ,$\Gamma_{jm} = \sum_{k \in K^N} \frac{r_{jmk}}{cR_k^N}$

Γ_j^{min}: menor consumo relativo de recurso de la actividad j

Γ^{max}: máximo consumo relativo más grande.

$\Delta r_{jmr} = r_{jmk} - r_{j\mu k}$ tal que $\mu_j \in M_j$ $m \neq \mu_j$: diferencia entre el consumo de recursos de los modos en que se pueden asignar a una actividad y el modo que tiene asignado en un instante de tiempo.

En el algoritmo 2.1 se construye una asignación de modos factibles, mediante un procedimiento divi-

dido en cuatro fases, a continuación se muestra su estructura:

Algoritmo 2.1 Heurística para el MRCPSP (H1)

Fase 1: Inicialización:

1. $S = \{1\}$,$D = \{j : j \in J \setminus \{1\}\}$, $cR_k^N = R_k^N \forall k \in K^N$

1.1 Calcular : cR_k^N, $k \in K^N$,Γ_{jm}, $j \in D$

Fase 2: Asignación de modos de procesamiento:

2. FOR $j = 2, ..., |J| - 1$ calcular

2.1. $\Gamma_j^{min} = min\left\{\Gamma_{jm} : m \in M_j,\ j \in D\right\}$

2.2. $\Gamma^{max} = max\left\{\Gamma_j^{min} : j \in D\right\}$

2.3. $\hat{j} = min\left\{j : \Gamma_j^{min} = \Gamma^{max},\ j \in D\right\}$,$\mu_{\hat{j}} = min\left\{m : \Gamma_{\hat{j}m} = \Gamma_{\hat{j}}^{min},\ m \in M_{\hat{j}}\right\}$

2.4.$S = S \cup \{\hat{j}\}$, $D = D \setminus \{\hat{j}\}$

2.5. Actualizar: $cR_k^N \forall k \in K^N$,Γ_{jm},$j \in D$, $m \in M_j$

Fase 3: Cálculo de tiempos de inicio y fin

3. Si$\exists k \in K^N$, $cR_k^N < 0$: entonces parar.

En otro caso: $S = \{1\}$, $D = \{j : j \in J \setminus \{1\}\}$

3.1. Seleccionar $j^* = \min_j \left\{\pi_j : j \in S\right\}$ y calcular

3.2 $ES_{j^*} = max\left\{f_h : h \in A_{j^*}(t)\right\}$

3.4. $s_{j^*} = min\{t\}$ tal que $t \geq ES_{j^*}$, $R_k(\tau) \geq r_{j^*\mu k}$,$\tau \in \left[t, t + d_{j\mu}\right]$,$k \in K$

3.5. $f_{j^*} = s_{j^*} + d_{j^*\mu}$

Fase 4: Asignación de recursos excedentes:

4. For $j = 2, ..., |J| - 1$

4.1. $m = \mu_j$

4.2. Mientras $m \geq 1$

4.2.1 Si $cR_k^N - \Delta r_{jmk} \geq 0\ \forall k \in K^N$ entonces $\mu_j = m$

4.2.2 $m = m - 1$

4.2.3 Actualizar: $cR_k^N \forall k \in K^N$

4.3. Si $cR_k^N \geq 0$, $\forall k \in K^N$ entonces se tiene una solución factible.

En la primera fase se inicializa todos lo elementos que intervienen en el procedimiento. En la segunda fase, se procede a realizar la asignación de modos de procesamiento a todas las actividades, siempre asignando los modos que menos recursos no renovables consumen. Una vez terminado la asignación inicial de modos se controla si no se ha excedido la cantidad de recursos no renovables disponibles (si se sobrepasa no existe asignación de modo factible). La fase tres consiste en calcular los tiempos de inicio y de fin de cada actividad j, respetando las relaciones de precedencia y sin violar la dispo-

nibilidad de recursos renovables en cada instante de tiempo. En la cuarta fase se procede a reasignar los recursos renovables excedentes de la asignación inicial.

2.3. PSO para el MRCPSP

Una de las variantes de PSO para resolver un PSP que más se ajusta a la idea original de la versión para problemas binarios propuesta por Kennedy y Eberhart en [42], es la propuesta por Jarboui y colaboradores [57] para resolver MRCPSP. Se emplea la idea de este trabajo para realizar el desplazamiento en un conjunto discreto $\widehat{M}$ formado por los vectores $M = (\mu_1, \mu_2, ..., \mu_n)$ de las asignaciones de los modos de procesamiento, por esta razón se muestra en detalle su funcionamiento.

2.3.1. Desplazamiento en el conjunto de los modos de procesamiento

A cada vector M se tiene un vector $Y_i^t \in \{-1, 0, 1\}^n$ el cual representa la posición de la partícula i de un enjambre de dimensión SS. Cada una de las componentes y_{ij}^t toman valores de acuerdo con el estado de la componente j del vector M_i^t en la iteración t. El estado de una componente y_{ij}^t no es más que la relación que existe entre la componente μ_{ij}^t de M_i^t y las mismas componentes μ_{gj}^t del vector M_g^t (x_g^t mejor partícula del enjambre en el PSO continuo (1.5.1)) y $\mu_{p_ij}^t$ del vector $M_{p_i}^t$ ($x_{p_i}^t$mejor posición alcanzada por la partícula i en el PSO continuo (1.5.1)) .

Las componentes de Y_i^t toman valor según la expresión siguiente:

$$y_{ij}^t = \begin{cases} 1 & si\ \mu_{ij}^t = \mu_{gj}^t \\ -1 & si\ \mu_{ij}^t = \mu_{p_ij}^t \\ alea\{-1,1\} & si\ \mu_{ij}^t = \mu_{p_ij}^t = \mu_{gj}^t \\ 0 & e.o.c \end{cases}$$

La componente j de la velocidad de una partícula i se actualiza de acuerdo con la ecuación siguiente:

$$v\mu_{ij}^{t} = \chi \cdot \left(v_{ij}^{t-1} + U\left(0, \phi_1\right) \cdot \delta_1 + U\left(0, \phi_2\right) \cdot \delta_2 \right) \tag{2.3.1}$$

Donde: $\delta_1 = -1 - y_{ij}^{t-1}$ y $\delta_2 = 1 - y_{ij}^{t-1}$

Note que, si $\mu_{ij}^{t} = \mu_{gj}^{t}$ entonces $y_{ij}^{t-1} = 1$, lo que implica que $\delta_1 = -2$ y $\delta_2 = 0$, lo que impone un sentido negativo en la velocidad. Si $\mu_{ij}^{t} = \mu_{pj}^{t}$ entonces $y_{ij}^{t-1} = -1$ en este caso se tiene que $\delta_1 = 0$ y $\delta_2 = -2$, tomando un sentido positivo la velocidad. Al coincidir $\mu_{ij}^{t} = \mu_{pj}^{t} = \mu_{gj}^{t}$, la componente y_{ij}^{t-1} toma aleatoriamente los valores $\{-1, 1\}$ y la velocidad adquiere sentido inverso al valor tomado por y_{ij}^{t-1} . Cuando $\mu_{ij}^{t} \neq \mu_{ig}^{t}$ y $\mu_{ij}^{t} \neq \mu_{p_i j}^{t}$,se tiene que $\delta_1 = -1$ y $\delta_2 = 1$, en esta situación el sentido de la velocidad está determinado por $U(0, \phi_a)$, $a = 1, 2$. En [57], no se emplea el factor de constricción χ para limitar la velocidad, sino el peso de inercia (ver expresión (1.5.1)).

Para actualizar la posición de la partícula se emplea expresión:

$$\rho_{ij}^{t} = y_{ij}^{t-1} + v\mu_{ij}^{t} \tag{2.3.2}$$

que sirve para se actualizar y_{ij}^{t} :

$$y_{ij}^{t} = \begin{cases} 1 & si\ \rho_{ij}^{t} > \alpha \\ -1 & si\ \rho_{ij}^{t} < -\alpha \\ 0 & e.o.c \end{cases}$$

El valor de α es denominado parámetro de intensificación o diversificación. Note que para valores pequeños de α, μ_{ij}^{t} toma valores preferentemente de μ_{gj}^{t} o μ_{pj}^{t} (intensificación). En otro caso μ_{ij}^{t} toma valores distintos de μ_{gj}^{t} o $\mu_{p_i j}^{t}$ (diversificación).

Finalmente la nueva posición de la partícula queda actualizada según:

$$\mu_{ij}^{t} = \begin{cases} \mu_{gj}^{t-1} & si\ y_{ij}^{t} = 1 \\ \mu_{pj}^{t-1} & si\ y_{ij}^{t} = -1 \\ alea. & si\ y_{ij}^{t} = 0 \end{cases} \tag{2.3.3}$$

2.3.2. Descripción del algoritmo propuesto

La idea del algoritmo PSO que se propone para el MRCPSP consiste en definir el espacio de búsqueda como $X = \left\{\widehat{\Pi}, \widehat{M}\right\}$ donde $\widehat{\Pi}$ es el conjunto de los vectores que contienen las prioridades de cada una de las actividades del proyecto. El conjunto $\widehat{M}$ es el conjunto de los vectores que contienen la asignación de modos de cada solución.

Una partícula queda representada por la tupla $(\Pi_i, M_i, v\pi_i, v\mu_i, \Pi_{p_i}, \mu_{p_i})$ donde:

- Π_i: posición en el conjunto $\widehat{\Pi}$.
- M_i: posición en el conjunto $\widehat{M}$.
- $v\pi_i = \chi \cdot \left(v\pi_i^{t-1} + U(0,\phi_1) \cdot \left(\Pi_{p_i}^t - \Pi_i^{t-1}\right) + U(0,\phi_2) \cdot \left(\Pi_g^t - \Pi_i^{t-1}\right)\right)$: es la velocidad de una partícula i en $\widehat{\Pi}$.
- $v\mu_i$: es velocidad del desplazamiento de la partícula i en el conjunto $\widehat{M}$ calculada a partir de (2.3.1).
- Π_{p_i}: mejor posición que ha visitado cada partícula i en $\widehat{\Pi}$.
- M_{p_i}: mejor posición que ha visitado cada partícula i en $\widehat{M}$.

En el paso (1) del algoritmo 2.4 se inicializa el enjambre generando SS soluciones factibles (número de partículas) del MRCPSP, mediante la heurística H1. En el paso (2) se realiza el movimiento del enjambre en el espacio de búsqueda $X = \left\{\Pi, \widehat{M}\right\}$ lo que se traduce en movimientos en los conjuntos Π y $\widehat{M}$. La condición de parada que se tendrá en cuenta es el número de iteraciones CI que realiza el algoritmo. A continuación se muestra un seudocódigo del algoritmo PSO con topología Gbest para el MRCPSP.

Algoritmo 2.2 Algoritmo *HPSOSA* para el MRCPSP

1. Inicialización : $t = 0$, SS, CI, ϕ_1, ϕ_2 ,ε

   1.1 Para $i = 1, ..., SS$:

   1.2 Inicializar posición $\left(\Pi_i^0, M_i^0\right) \leftarrow H1$

2. Mientras $(t \leq CI + 1)$ (Movimiento del Enjambre)

   2.1 $t = t + 1$

   2.2 Calcular $\left(\Pi_g^t, M_g^t\right)$ tal que $g = \left\{ i : \min_i \left(F\left(\Pi_i^t, M_i^t\right)\right) \right\}$

   2.3 Para la cantidad de partículas: $i = 1, ..., np$

   2.4 Actualizar mejor posición de i :$\left(\Pi_{p_i}^t, M_{p_i}^t\right)$

   2.4.1 Calcular velocidad en Π: $v\pi_i^t$

   2.4.2 Desplazamiento en Π: $\Pi_i^t = \Pi_i^{t-1} + v\pi_i^t$

   2.4.3 Calcular velocidad: $v\mu_i^t$

   2.4.3 Desplazamiento en $\widehat{M}$: actualizar μ_{ij}^t según (2.3.3)

   2.4.3.1. Codificación de: (Π_i^t, M_i^t)

3. La mejor solución es: $\left(\Pi_g^t, M_g^t\right)$ Fin

2.3.3. Hibridación en el enfoque PSO

Los cambios que experimentan las partículas al realizar el desplazamiento en el conjunto $\widehat{M}$ que no son más que los rasgos que adquieren provenientes de las asignaciones de modo de M_g^t , $M_{p_i}^t$ o de la asignación aleatoria de un recurso que no está presente en estas asignaciones (ver expresión 2.3.3). Se puede dar el caso de que los nuevos modos asignados en M_i^t no permitan mejorar la calidad de la posición de las partículas, inclusive puede que no sean factibles en cuanto al consumo de recursos no renovables. Para atenuar la posible no factibilidad en la asignación de modos M_i^t, se propone una estrategia para aceptar los cambios en la asignación de modos que se realiza en la expresión (2.3.3).

En la los pasos 2.3.4 y 2.3.4.1 del algoritmo 2.2, se procede a actualizar los valores μ_{ij}^t según la expresión (2.3.3). El siguiente algoritmo describe como se realiza este proceso:

Algoritmo 2.3 Aceptación del modo de procesamiento

1. **Inicialización** $j = 2,...,|J|-1$, $t = 1,...,CI$, $\Gamma^N_{j\mu}(t-1)$, $T^l_0 = \frac{300*|J|}{SS*CI}$, $T^g_0 = \frac{300}{SS}$, $cR^N_k(t) = R^N_k$ $\forall k \in K^N$, $c = 0, S = \{1\}, D = \{j : j \in J \setminus \{1\}\}$

   Seleccionar $j^* \in D$ según regla de prioridad MTS.

2. **Aceptación del modo**

   2.1 $c = c+1$

   2.2 $T^l(j^*) = \frac{T^l_0}{c\times\chi}$, $T^g(t) = \frac{T^g_0}{t\times\chi}$

   2.3 $\Delta_{j^*} = \Gamma^N_{j\mu}(t) - \Gamma^N_{j\mu}(t-1)$

   2.4 Si $\Delta < 0$ o $P\left(\mu^t_{ij^*} = \mu^t_{ij^*}, \Delta > 0\right) > U(0,1)$

   2.4.1 $\mu^t_{ij^*} = \mu^t_{ij^*}$

   2.4.2 En otro caso $\mu^t_{ij^*} = \mu^{t-1}_{ij^*}$

3. **Secuenciación**

3.1 $ES_{j^*} = max\left\{f_h : h \in A_{j^*}(t)\right\}$
3.2. $s_{j^*} = min\{t\}$ tal que $t \geq ES_{j^*}$, $R_k(\tau) \geq r_{j^*\mu k}, \tau \in \left[t, t+d_{j\mu}\right], k \in K$
3.3. $f_{j^*} = s_{j^*} + d_{j^*\mu}$

La aceptación del modo μ^t_{ij} en el algoritmo 2.3 se realiza a partir de un procedimiento similar al de SA. La probabilidad de aceptación se calcula a partir de la expresión siguiente:

$$P\left(\mu^t_{ij} = \mu^t_{ij}, \Delta > 0\right) = \frac{1}{j}\left(p_t + p_j\right),\ j = 2,...,|J|, t = 1,...,CI \tag{2.3.4}$$

Donde:

Δ: es la diferencia entre el consumo relativo de los modos asignados μ^t_{ij} , μ^{t-1}_{ij} en las iteraciones t y $t-1$ respectivamente.

$p_t = \exp\left(-\frac{\Delta_j}{T^g(t)}\right)$: es el valor de la probabilidad de aceptar un incremento en el consumo relativo en la iteración t

$p_j = \exp\left(-\frac{\Delta_j}{T^l(j)}\right)$: es el valor de la probabilidad de aceptar un incremento en el consumo relativo en la actividad j del proyecto.

El valor de la expresión (2.3.4) siempre es menor que 1 debido a que la suma de las dos probabilidades

es dividida por el índice de la actividad j que recibe la asignación del modo en la iteración t. Esta expresión tiene la propiedad que en las primeras iteraciones y en las primeras actividades que deben ser las de mayor prioridad existe libertad para aceptar un incremento del consumo relativo de recursos. El paso 3 del algoritmo 2.3 tiene como objetivo secuenciar las actividades sin violar las relaciones de precedencia y los niveles de recursos disponibles.

3 Resultados computacionales.

3.1. Instancias de prueba

Una vez diseñado un algoritmo es necesario implementarlo y realizar experimentos para estudiar su comportamiento ante las intancias de prueba. En este sentido se tomaron las intancias de la PSPLIB[1] [58], que son las más utilizadas por la comunidad científica que trabaja en el tema. Las instancias son construidas por un generador denominado ProGen el cual genera proyectos de forma aleatoria utilizando un diseño de experimentos factorial basado en dos conjuntos de parámetros. El primer conjunto consta de parámetros base que son iguales para cada instancia de prueba:

- Número de actividades
- Número de modos en que cada actividad puede ejecutarse
- Número de recursos renovables existentes en el problema
- Disponibilidad máxima de cada recurso renovable
- Número de recursos no renovables existentes en el problema
- Disponibilidad máxima de cada recurso no renovable
- Número de sucesores de la actividad ficticia inicio
- Número de predecesores de la actividad ficticia fin
- Número de sucesores y predecesores de las actividades no ficticias
- Duración de las actividades

[1] Project Scheduling Problem Library

El segundo conjunto está conformado por los parámetros variables, es decir cambian de instancia a instancia:

- Complejidad de la Red (NC): Define la cantidad promedio de relaciones de precedencia no redundantes por cada actividad.
- Factor de Recurso (RF): Refleja la proporción media del número de recursos diferentes que cada actividad no ficticia utiliza. Se aplica tanto a recursos renovables (RFR) como no renovables (RFNR). Si $RF = 1$ quiere decir que la actividad utiliza todos los recursos renovables, por el contrario $RF = 0$ significa que la actividad no utiliza ningún recurso renovable.
- Grado de restricción de los recursos (RS): Este parámetro expresa la relación entre la demanda de recursos de las actividades y la disponibilidad de los mismos; es decir mide la fortaleza de las restricciones o la escasez de los recursos. Este parámetro se calcula tanto para recursos renovables y no renovables.

3.2. Selección de los parámetros

Los parámetros de un algoritmo heurístico son los que conducen la búsqueda. Por esta razón su configuración es de suma importancia. A continuación se muestran los parámetros que se han definido para el algoritmo H-PSOSA.

Optimización por Enjambre de Partículas	
Parámetro	Valores
Factores de aceleración: ϕ_1, ϕ_2	$\phi_2 = \phi_1 = 2{,}05$
intensificación - diversificación: ε	$\{1; 1{,}5; 2; 2{,}5, 3\}$
Cantidad de iteraciones: *CI*	250
Dimensión del Enjambre: *SS*	20
Hibridación Recocido Simulado	
Parámetro	Valores
Temperatura Inicial: T_0	17,5
Factor de Enfriamiento: α	0,72

Tabla 3.1: Parámetros del algoritmo

La cantidad de iteraciones (*CI*) del algoritmos se eligió de acuerdo a la dimensión del enjambre (*SS*)

de forma tal que $SS \times CI = 5000$, y teniendo en cuenta que Medrano en [59], prueba estadísticamente que para enjambre de 20 y 25 partículas no es significativa la calidad de las soluciones que aporta el algoritmo H-PSOSA. Los valores de los coeficientes de aceleración se tomaron como $\{\phi_1 = \phi_2 = 2{,}05\}$ de acuerdo con [9]. Con esta elección se compensa la influencia de la mejor posición $\left(\Pi_p^t, \mu_p^t\right)$ visitada por la partícula y de la mejor partícula del enjambre $\left(\Pi_g^t, \mu_g^t\right)$ sobre la velocidad de las partículas. Para todos los casos el valor del factor de constricción se mantiene constante $\chi = 0{,}7298$ (ver expresión (1.5.4). Los parámetros correspondientes a la hibridación con SA se eligieron de acuerdo con [60]. Se analiza solamente el efecto que tiene los diferentes niveles del parámetro factor de intensificación-diversificación de la búsqueda (ε).

3.3. Análisis de los resultados

No existe un procedimiento establecido y aceptado para comparar el desempeño de los algoritmos sobre múltiples conjuntos de datos de entrada, esto se debe al comportamiento no determinista de estos. Por este motivo, la diferencia detectada entre los resultados de dos algoritmos podría deberse a factores aleatorios, y no a una mejora real [61]. Para cada configuración de los parámetros de un algoritmo se realizaron 25 corridas. Por lo general los resultados obtenidos por las metaheurísticas no cumplen las condiciones requeridas para poder usar de forma correcta las comparaciones paramétricas [62]. Esta es la razón por la que en este trabajo se usan las técnicas no paramétricas para la comparación de las medias del valor de la función objetivo de las soluciones encontradas.

3.3.1. Resultados computacionales

A continuación se muestran los resultados alcanzados por el algoritmo en un conjuntos de instancias de la PSPLIB, con proyectos de 18 actividades. La medida empleada para analizar el comportamiento del algoritmo es la desviación respecto al óptimo *DRO* que se define a continuación:

$DRO = \frac{C_{max} - OP}{OP} \times 100$

Donde:

Cmax : es la duración del proyectos (valor de la función objetivo).

OP : valor del óptimo de la instancia.

El experimento computacional llevado a cabo en esta investigación consiste en 25 corridas para cada una de las 5 configuraciones de los parámetros del algoritmo y cada una de las instancias del conjunto $J18$[2] de la PSPLIB. En la tabla 3.2 aparecen las estadísticas que se tomaron a partir del experimento computacional, donde Max, significa el máximo de DRO para una configuración, Std es la desviación estándar para cada una de las configuraciones y %OP es el porcentaje de las veces que el algoritmo alcanza el valor óptimo de una instancia.

Estadísticas *DRO*				
Valores de ε	Max.	Media.	Std.	%OP
1.0	53.30	5.65	8.71	58,00
1.5	50.00	5.55	8.64	58,30
2.0	50.00	5.54	8.50	57,80
2.5	58.33	5.75	8.61	56,10
3.0	50.00	6.20	9.06	54,50

Tabla 3.2: Resultados del algoritmo H-PSOSA instancias $J18$

En tabla 3.3 se muestra una comparación del algoritmo H-PSOSA con 8 de los mejores algoritmos para resolver MRCPSP reportados en la literatura. Solamente se han incluido los trabajos relacionados con el empleo de alguna técnica metaheurística. Es

Algoritmo	*DRO* (5000 soluciones)	%OP
H-PSOSA	**5,55**	**58.30**
Josefowska y colaboradores. [63]	5,52	-
Bouleimen y colaboradores. [28]	1.85	69.4
Alcaraz y colaboradores. [64]	1,43	-
Zhang y colaboradores. [65]	1.33	74.5
Jarboui y colaboradores. [57]	0.89	79.8
Lova y colaboradores. [66]	0.63	84.9
Peteghem and Vanhoucke [67]	0.52	86.2
Vanhoucke and Peteghem [68]	0.42	88.9

Tabla 3.3: Comparación del algoritmo H-PSOSA con algunos trabajos similares en instancias $J18$

Se puede apreciar que el promedio de la desviación respecto al óptimo del algoritmo H-PSOSA es

[2]Instancias con 18 actividades

ligeramente superior (0,03 unidades) a la del trabajo de Josefowska y colaboradores [63]. En el caso del porcentaje de óptimos alcanzados es aproximadamente un 11 % menor que la alcanzada por Bouleimen y colaboradores. [28][3].

Con el objetivo de determinar la influencia del factor de intensificación-diversificación de las configuraciones de la tabla 3.1, se realizó una comparación de medias. Para este propósito se empleó la prueba de Friedman[69] (Anexo 1), con un nivel de significación del 99 %. Se detecta que existen diferencias significativas entre los diferentes niveles del factor de intensificación-diversificación tal que $\varepsilon \in \{1{,}0; 1{,}5; 2{,}0; 2{,}5; 3{,}0\}$. En el Anexo 2 aparecen las salidas de la aplicación del test de Friedman realizado a los datos de las corridas.

[3] [28] Es el mejor algoritmo basado en Recocido Simulado reportado en la literatura para resolver el MRCPSP.

4 Conclusiones

En esta investigación se implementó un algoritmo metaheurístico híbrido basado en PSO y SA para el MRCPSP, dando cumplimiento a los objetivos trazados. Con el desarrollo de la investigación se arriban a las conclusiones siguientes:

- A partir del estudio que se realiza sobre la teoría de los problemas de planificación de proyecto con recursos limitados, se logró establecer la representación de la solución y el espacio de búsqueda del problema MRCPSP combinando el vector de prioridades y el vector de asignación de modos de procesamiento.
- Se realizó un estudio de la metaheurística Optimización por Enjambre de Partículas y Recocido Simulado, lo que permitió diseñar un algoritmo híbrido para la obtención de soluciones del MRCPSP.
- Se implementó una variante del algoritmo H-PSOSA diseñado en C#, que se descompone en tres partes fundamentales: el procedimiento empleado para la generación de asignación de modos factibles, el mecanismo de movimiento en el espacio de búsqueda y el procedimiento de aceptación de modos.
- Se realizó un estudio computacional del algoritmo implementado a partir de 25 corridas para cada configuración de los parámetros en las instancias $J18$, el cual permitó comprobar que el algoritmo H-PSOSA obtiene resultados ligeramente inferiores, si se compara con los mejores de su tipo reportados en la literatura hasta el año 2011. Esto se debe a que los algoritmos empleados en la comparación utilizan mecanismos de mejoras para las secuencias y el H-PSOSA no los considera. A pesar de esto se puede considerar que el H-PSOSA es un algoritmo eficaz para

resolver el MRCPSP.

- Se probaron estadísticamente los resultados del algoritmo H-PSOSA para las diferentes configuraciones, determinándose que es significativo el efecto del factor de intesificación-diversificación para los valores $\{1;1,5;2;2,5;3\}$ cuando se mantienen constantes los coeficientes de aceleración $\{\phi_1 = \phi_2 = 2{,}05\}$, la cantidad de iteraciones $CI = 250$ y la dimensión del enjambre $SS = 20$. Para $\varepsilon = 1,5$ se obtienen resultados ligeramente mejores que para los demás valores del factor.

4.1. Recomendaciones

La investigación realizada no agota las alternativas de modelación y solución del problema en estudio y por cuestiones de tiempo no se pudieron realizar experimentos computacionales más extensos. Por tanto se proponen las siguientes recomendaciones:

- Realizar experimentos computacionales más extensos empleando otras configuraciones de los parámetros y otras instancias de la PSPLIB.
- Implementar un algoritmo que emplee otra topología de enjambre.
- Incorporar algún mecanismo de mejora para las secuencias.
- Desarrollar una variante adaptativa del algoritmo implementado.

Bibliografía

[1] E. L. Demeulemeester and W. S. Herroelen, *PROJECT SCHEDULING A Research Handbook*, 1st ed. New York, Boston, Dordrecht, London, Moscow: KLUWER ACADEMIC PUBLISHERS, 2002. 12, 16, 20, 22, 27

[2] D. Malcolm, J. Roseboom, C. Clark, and W. Fazar, "Applications of a technique for r and d program evaluation pert," *Operations Research*, vol. 5, no. 7, pp. 646–669, 1959. 12, 15

[3] J. J. E. Kelley and M. R. Walker, "Critical-path planning and scheduling," in *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, ser. IRE-AIEE-ACM '59 (Eastern). New York, NY, USA: ACM, 1959, pp. 160–173. [Online]. Available: http://doi.acm.org/10.1145/1460299.1460318 12, 15

[4] R. Kolisch and R. Padman, "An integrated survey of deterministic project scheduling," *Omega*, vol. 29, no. 3, pp. 249–272, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/B6VC4-433PB5J-4/2/2438e3d60326edbde74a7078f62e85e9 12, 16, 17

[5] P. Brucker, A. Drexl, R. Mohring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-3YSXJP1-P/2/7006a62859d7b4d83e72b0ea50c8b88b 12, 15, 17, 18

[6] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 207, no. 1, pp. 1–14, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-4XNN5G6-2/2/2500b7b287428c1b2f4823dd73df7d17 12, 15, 17, 18

[7] R. KOLISCH and A. DREXL, "Local search for nonpreemptive multi-mode resource-constrained project scheduling," *IIE Transactions*, vol. 29, pp. 987–999, 1997, 10.1023/A:1018552303415. [Online]. Available: http://dx.doi.org/10.1023/A:1018552303415 13, 38

[8] S. Hartmann and S. Hartmann, "Project scheduling with multiple modes: A genetic algorithm," *Annals of Operations Research*, vol. 102, pp. 111–135, 1997. 13

[9] *Hybrid Algorithm Based on Particle Swarm Optimization and Simulated Annealing for Solving Resources Constrained Multi-Projects Scheduling Problem*. International Conference on Operation Research, Havana, April 2012. 13, 48

[10] A. A. B. Pritsker and G. E. Whitehouse, "Gert, graphical evaluation and review technique; part i development part ii probabilistic and industrial engineering applications," *Journal of Industrial Engineering*, vol. 17, no. 5, pp. 45–50, 1966. 15

[11] ——, "Gert: Graphical evaluation and review technique, part ii probabilistic and industrial engineering applications," *Journal of Industrial Engineering*, vol. 17, no. 6, pp. 45–50, 1966. 15

[12] R. Slowinski, B. Soniewicki, and J. Weglarz, "Dss for multiobjective project scheduling," *European Journal of Operational Research*, vol. 79, no. 2, pp. 220–229, 1994. 15

[13] R. Kolish, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource constrained project scheduling problems," *Management Science*, vol. 41, pp. 1693–1703, 1995. 15

[14] W. Herroelen, E. Demeulemeester, and B. De Reyck, "A clasification scheme for project problem," in *Project Scheduling: Recente Models Algorithms and Applications*, ser. Operation Research and Management Science, J. Weglarz, Ed. Kluwer Academic Publishers, 1999, pp. 1–26. 17, 18, 20

[15] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: An update," *European Journal of Operational Research*, vol. 174, no. 1, pp. 23–37, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-4G7DY5D-7/2/6b51533fd69866a5f3a3a556e13ea72a 17

[16] 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*, vol. 80, no. 1, pp. 94–102, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377221793E02948 19, 24

[17] R. Kolish and S. Hartman, "Heuristic algorithms for solving the resource constrained project scheduling problem classification and computational analysis," pp. 147–178, 1999. 23, 24, 25

[18] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation," *European Journal of Operational Research*, vol. 90, no. 2, pp. 320–333, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-3VVVR6Y-13/2/0ed4cf8c4903422c10de168f349b71ec 23, 24

[19] B. Fransisco, "Nuevos métodos de resolucion del problema de secuenciacion de proyectos con recursos limitados," Ph.D. dissertation, Departamento de Investigacion de Operaciones y Estadistica Universidad de Politecnica de Valencia, 2002. 27

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: http://www.sciencemag.org/content/220/4598/671.abstract 27

[21] D. Henderson, S. Jacobson, and A. Johnson, "The theory and practice of simulated annealing," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science, F. Glover and G. Kochenberger, Eds. Springer New York, 2003, vol. 57, pp. 287–319, 10.1007/0-306-48056-5_10. [Online]. Available: http://dx.doi.org/10.1007/0-306-48056-5_10 27

[22] L. Ingber, "Simulated annealing: Practice versus theory," *Mathematical and Computer Modelling*, vol. 18, no. 11, pp. 29–57, 1993. 27

[23] P. J. M. V. LAARHOVEN, E. H. L. AARTS, and J. K. LENSTRA, "Job shop schedulingb y simulateda nnealing," *Operations Research*, vol. 40, no. 1, pp. 113–125, Jan. - Feb 1992. [Online]. Available: http://www.jstor.org/stable/171189 29

[24] F. F. Boctor, "Resource-constrained project scheduling by simulated annealing," *International Journal of Production Research*, vol. 34, no. 8, pp. 2335–2351, 1996. 29

[25] J.-H. Cho and Y.-D. Kim, "A simulated annealing algorithm for resource constrained project scheduling problems," *The Journal of the Operational Research Society*, vol. 48, no. 7, pp. 736–744, 1997. 29

[26] J. Jozefowska, M. Mika, R. Rozycki, G. Waligora, and J. Węglarz, "Simulated annealing for

multi-mode resource-constrained project scheduling," *Annals of Operations Research*, vol. 102, pp. 137–155, 2001, 10.1023/A:1010954031930. [Online]. Available: http://dx.doi.org/10.1023/A:1010954031930 29

[27] M. Mika, G. Waligóra, and J. Weglarz, "Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models," *European Journal of Operational Research*, vol. 164, no. 3, pp. 639–668, 2005, recent Advances in Scheduling in Computer and manufacturing Systems. 29

[28] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version," *European Journal of Operational Research*, vol. 149, no. 2, pp. 268–281, September 2003. 29, 49, 50

[29] J. Kennedy and R. Eberhart, "Particle swarm optimization." Proceedings of the IEEE international conference on neural networks: Piscataway, 1995, pp. 1942–1948. 29

[30] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Computing*, vol. 7, pp. 109–124, 2008, 10.1007/s11047-007-9050-z. [Online]. Available: http://dx.doi.org/10.1007/s11047-007-9050-z 29

[31] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, 2008. [Online]. Available: http://downloads.hindawi.com/archive/2008/685175.pdf 29

[32] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, april 2007, pp. 120–127. 29, 32

[33] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, may 1998, pp. 69–73. 31

[34] M. Clerc, *Particle Swarm Optimization*. ISTE Ltd London. UK, 2006. 31, 32

[35] M. R. Rapaic and Z. Kanovic, "Time-varying pso convergence analysis, convergence-related parameterization and new parameter adjustment schemes," *Information Processing Letters*, vol. 109, no. 11, pp. 548–552, 2009. [Online]. Available: http://hinari-gw.who.int/whalecomwww.sciencedirect.com/whalecom0/science/article/pii/S0020019009000350 31

[36] F. van den Bergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025505000630 31, 33

[37] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007, 10.1007/s11721-007-0002-0. [Online]. Available: http://dx.doi.org/10.1007/s11721-007-0002-0 31

[38] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization," in *Evolutionary Computation, 1999.*, ser. CEC 99., jul 1999 1999. 31

[39] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 1, pp. 58–73, feb 2002. 31

[40] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, vol. 2, 2002, pp. 1671–1676. 32

[41] F. van den Bergh, "An analysis of particle swarm optimizers." Ph.D. dissertation, Department of Computer Science, University of Pretoria,, 2001. [Online]. Available: http://www.cs.up.ac.za/cs/fvdbergh/publications/phd_thesis.ps.gz 32

[42] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, oct 1997, pp. 4104–4108 vol.5. 33, 40

[43] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: An update," *European Journal of Operational Research*, vol. 174, no. 1, pp. 23–37, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-4G7DY5D-7/2/6b51533fd69866a5f3a3a556e13ea72a 33

[44] H. Zhang, X. Li, H. Li, and F. Huang, "Particle swarm optimization-based schemes for resource-constrained project scheduling," *Automation in Construction*, vol. 14, no. 3, pp. 393–404, 2005, international Conference for Construction Information Technology 2004. [Online]. Available: http://www.sciencedirect.com/science/article/B6V20-4DJ4J6C-1/2/e1e8316b796dd095ed98b27aba6d5927 34

[45] H. Zhang, H. Li, and C. Tam, "Particle swarm optimization for resource-constrained project scheduling," *International Journal of Project Management*, vol. 24, no. 1, pp. 83–92, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/B6V9V-4GSTPKN-1/2/6bc1478dd42fc849d70ae648400764ef 34

[46] H. Zhang, H. Li, and C. M. Tam, "Permutation-based particle swarm optimization for resource-constrained project scheduling," *Journal of Computing in Civil Engineering*, vol. 20, no. 2, pp. 141–149, 2006. [Online]. Available: http://link.aip.org/link/?QCP/20/141/1 34

[47] R.-M. Chen, C.-L. Wu, C.-M. Wang, and S.-T. Lo, "Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in psplib," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1899–1910, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6V03-4WXSJY7-4/2/d1fe16b6370b1a2cdbd15031805794cb 34

[48] R.-M. Chen, "Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7102–7111, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/B6V03-51S25CW-8/2/3576438aa5e85da720b6fbf05d936c8f 34

[49] S.-T. Lo, R.-M. Chen, D.-F. Shiau, and C.-L. Wu, "Using particle swarm optimization to solve resource-constrained scheduling problems," in *Soft Computing in Industrial Applications, 2008. SMCia '08. IEEE Conference on*, june 2008, pp. 38–43. 34

[50] S. K. Tchomte and M. Gourgand, "Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems," *International Journal of Production Economics*, vol. 121, no. 1, pp. 57–67, 2009, modelling and Control of Productive Systems: Concepts and Applications. [Online]. Available: http://hinari-gw.who.int/whalecomwww.sciencedirect.com/whalecom0/science/article/pii/S0925527309001042 34

[51] C. M., "A discrete particle swarm optimization, illustraded by the traveling salesman problem," in *New optimization techniques in engineering*, ser. Fuzziness and Softcomputing, C. Godfrey, B. Onwubolu, and V. Babu, Eds. Springer, 2004, pp. 219–238. 34

[52] L. Deng, Y. Lin, and W. Zheng, "Incorporating justification in the particle swarm optimization for the rcpsp," *International Journal of Innovative Computing, Information and Control*, vol. 4,

no. 9, September 2008. 34

[53] V. Valls, F. Ballestín, and S. Quintanilla, "Justification and rcpsp: A technique that pays," *European Journal of Operational Research*, vol. 165, no. 2, pp. 375–386, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221704002462 34

[54] D. Linyi and L. Yan, "A particle swarm optimization for resource-constrained multi-project scheduling problem," in *Proceedings of the 2007 International Conference on Computational Intelligence and Security*, ser. CIS '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1010–1014. [Online]. Available: http://dx.doi.org/10.1109/CIS.2007.58 34

[55] J. Czogalla and A. Fink, "Particle swarm topologies for resource constrained project scheduling," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, ser. Studies in Computational Intelligence, N. Krasnogor, M. Melian-Batista, J. Perez, J. Moreno-Vega, and D. Pelta, Eds. Springer Berlin / Heidelberg, 2009, vol. 236, pp. 61–73, 10.1007/978-3-642-03211-0_6. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03211-0_6 34

[56] I. Marteens, *Intuitive CShap: Introducción a la plataforma .Net*, 2008. [Online]. Available: http://www.marteens.com/pdfs/csharp_intsight.pdf 36

[57] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299–308, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6TY8-4NMC869-2/2/849c1eacd45855659b4e865e5736d44a 40, 41, 49

[58] R. Kolisch and A. Sprecher, "Psplib - a project scheduling problem library : Or software - orsep operations research software exchange program," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-3T7HK9P-1F/2/f3cc7f46a925673bfab16f6be5a4de4b 46

[59] B. E. Medrano, "Planificación de múltiples proyectos de desarrollo software utilizando métodos heurísticos," Master's thesis, Universidad de La Habana, por aparecer. 48

[60] B. E. Medrano, A. Ruiz, and R. R. Tauler, "Algoritmos basados en recocido simulado para la solución de un problema de planificación de múltiples proyectos de desorrollo de software," vol. En CD. IV Taller de Inteligencia Artificial. UCIENCIA, 2012. [Online]. Available: http://uciencia.uci.cu/es/node/1477?destination=node/1477 48

[61] A. Y. P. Cáceres, "Desarrollo de meta-heurísticas poblacionales para la solución de problemas complejos," Ph.D. dissertation, UCLV CEI, 2009. 48

[62] S. Garcia, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec 2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, pp. 617–644, 2009, 10.1007/s10732-008-9080-4. [Online]. Available: http://dx.doi.org/10.1007/s10732-008-9080-4 48

[63] J. Józefowska, M. Mika, R. RóÅŒycki, G. Waligóra, and J. Węglarz, "Simulated annealing for multi-mode resource-constrained project scheduling," *Annals of Operations Research*, vol. 102, pp. 137–155, 2001, 10.1023/A:1010954031930. [Online]. Available: http://dx.doi.org/10.1023/A:1010954031930 49, 50

[64] A. J., M. C., and R. R., "Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms," *The Journal of the Operational Research Society*, vol. 54, no. 6,

pp. 614–626, 2003. 49

[65] H. Zhang, C. M. Tam, and H. Li, "Multimode project scheduling based on particle swarm optimization," *Computer-Aided Civil and Infrastructure Engineering*, vol. 21, no. 2, pp. 93–103, 2006. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8667.2005.00420.x 49

[66] A. Lova, P. Tormos, M. Cervantes, and F. Barber, "An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes," *International Journal of Production Economics*, vol. 117, no. 2, pp. 302–316, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925527308003654 49

[67] M. Vanhoucke and V. Van Peteghem, "An artificial immune system for the multi-mode resource-constrained project scheduling problem," *Technology Management*, vol. 5482, pp. 85–96, 2009. [Online]. Available: http://discovery.ucl.ac.uk/1312142/ 49

[68] V. V. Peteghem and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 201, no. 2, pp. 409–418, 2010. [Online]. Available: http://hinari-gw.who.int/whalecomwww.sciencedirect.com/whalecom0/science/article/pii/S037722170900191X 49

[69] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference.*, fourth edition, revised and expanded ed. MARCEL DEKKER, INC., 2003. 50, 60

ANEXOS

Anexo 1 Test de Friedman

Test de Friedman:

Es una prueba no paramétrica de Friedman para comparar los efectos de las columnas en un diseño de dos factores (two-way). Esta prueba es similar al ANOVA clásico de dos factores, pero sólo para determinar los efectos de columnas (configuraciones) después de ajustar los efectos posibles de filas(corridas). No prueba los efectos de las filas ni el de la interacción fila-columnas. El test de Friedman es apropiado cuando las columnas de la matriz de datos representan los tratamientos que están bajo estudio, y las filas representan los efectos molestos (bloques) que se deben tomar en cuenta, pero no son de ningún interés para el estudio [69].

	configuraciones					
corridas		1	$\cdots$	j	$\cdots$	n
	1					
	$\vdots$		$\ddots$	$\vdots$		
	i		$\cdots$	DRO_{ij}	$\cdots$	
	$\vdots$			$\vdots$	$\ddots$	
	m					

Se asume, el modelo

$y = \mu + \alpha_j + \beta_i + \varepsilon_{ij}$

donde:

μ: es un parámetro de localización global.

α_j: representa el efecto de las columnas.

β_i: representa el efecto de las filas.

Hipótesis:

$H_0 : \alpha_1 = \alpha_2 =, ..., = \alpha_n$

H_1: al menos uno difiere.

Anexo 2 Resultados del Test de Friedman

Tabla Test de Friedman:

'Source'	'SS'	'df'	'MS'	'Chi-sq'	'Prob>Chi-sq'
'Columns'	[157.1639]	[4]	[39.2910]	[143.3980]	[0.00000]
'Error'	[3.4704e+04]	[31804]	[1.0912]	[]	[]
'Total'	[3.4862e+04]	[39759]	[]	[]	[]

stats:

source: 'friedman'

n: 7952

meanranks: [2.9796 2.9476 2.9450 3.0118 3.1159]

sigma: 1.0469

Comparación múltiple (test de Dunn-Sidak):

Conf1	Conf2	-0.0145	0.0319	0.0784
Conf1	Conf3	-0.0120	0.0345	0.0810
Conf1	Conf4	-0.0787	-0.0323	0.0142
Conf1	Conf5	-0.1829	-0.1364	-0.0899
Conf2	Conf3	-0.0439	0.0026	0.0491
Conf2	Conf4	-0.1107	-0.0642	-0.0177
Conf2	Conf5	-0.2148	-0.1683	-0.1218
Conf3	Conf4	-0.1133	-0.0668	-0.0203
Conf3	Conf5	-0.2174	-0.1709	-0.1244
Conf4	Conf5	-0.1506	-0.1041	-0.0576