SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
División de Estudios de Posgrado e Investigación

"POR MI PATRIA Y POR MI BIEN"

Algoritmo híbrido paralelo para la selección y calendarización de cartera de proyectos de gran escala

Que para obtener el grado de:
Maestro en Ciencias de la Computación

Presenta:
I.S.C Mayra Selene Hernández Guerrero
G10070526

Director:
Dra. Laura Cruz Reyes

Co-director:
Dr. Nelson Rangel Valdez

Cd. Madero, Tamaulipas — Febrero 2018

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Cd. Madero, Tams., a 08 de Diciembre de 2017

OFICIO No.: U5.241/17
ÁREA: DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS.

C. ING. MAYRA SELENE HERNÁNDEZ GUERRERO
No. DE CONTROL G10070526
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestro en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"ALGORTIMO HÍBRIDO PARALELO PARA LA SELECCIÓN Y CALENDARIZACIÓN DE CARTERA DE PROYECTOS DE GRAN ESCALA"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE :	DRA.	CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
SECRETARIO:	DR.	NELSON RANGEL VALDEZ
VOCAL:	DRA.	LAURA CRUZ REYES
SUPLENTE:	DRA.	GUADALUPE CASTILLA VALDEZ
DIRECTOR DE TESIS :	DRA.	LAURA CRUZ REYES
CO-DIRECTOR DE TESIS:	DR.	NELSÓN RANGEL VALDEZ

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
"POR MI PATRIA Y POR MI BIEN"®

DRA. ADRIANA ISABEL REYES DE LA TORRE
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p.- Archivo
Minuta

AIRT 'RAPR 'mapda*

Resumen

La formación de carteras de proyectos, tanto en el ámbito público como el privado, es una actividad periódica, crucial y necesaria para las organizaciones; regularmente éstas requieren maximizar su utilidad en relación a un presupuesto limitado. Para obtener los mejores resultados posibles, es imprescindible realizar un análisis de los aspectos que deben incorporarse en la formulación realista de un problema de carteras, destacando: recursos humanos, recursos financieros, periodos de tiempo, objetivos propuestos, así como restricciones presupuestales, entre otros.

En este trabajo se analizan algunos de estos aspectos y se identifica la importancia de considerar la selección de proyectos cuando existen restricciones temporales, además de las presupuestales. Esta condición lleva al bien conocido problema de programación de proyectos o calendarización, que ha sido ampliamente estudiado en la literatura especializada. Sin embargo, la selección de cartera de proyectos con calendarización ha sido menos tratada, y aún presenta retos abiertos a la investigación; uno de ellos es el tratamiento de problemas de gran escala.

En este trabajo se busca contribuir al reto de la solución de problemas, cuya dificultad crece cuando crece su tamaño, mediante la aplicación de estrategias que permitan resolver eficientemente los problemas que componen a un problema mayor. El algoritmo propuesto hace una hibridación de tres estrategías. La primera es una estrategia de descomposición, basada en relajación lagrangeana, que permite obtener problemas débilmente acoplados. La segunda estrategia parte de la anterior para formar problemas independientes; para ello utiliza un algoritmo aproximado de reordenamiento matricial basado en reducción de ancho de banda. Finalmente, la tercera estrategia es un algoritmo de optimización que resuelve de manera paralela los problemas independientes.

Los resultados experimentales dan evidencia del potencial del algoritmo híbrido propuesto. Se hicieron experimentos para mostrar la aportación de algunas de las estrategias del híbrido, y se contrastó el desempeño integral contra una alternativa de la literatura. La instancia más grande muestra una reducción de una hora, en el tiempo de ejecución, a favor del algoritmo propuesto.

Declaración de originalidad

Declaro que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares.

Además, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado en publicaciones ajenas, indico explícitamente los datos de los autores y publicaciones.

Además, en caso de infracción de los derechos de terceros derivados de este documento de tesis, acepto la responsabilidad de la infracción y relevo de ésta a mi director y codirectores de tesis, así como al Tecnológico Nacional de México y a su Instituto Tecnológico de Ciudad Madero.

Ing. Mayra Selene Hernández Guerrero

Agradecimientos

Dedico esta tesis a:

Mi amado esposo, Juan Javier, quien me brindó paciencia, serenidad y amor en los momentos más difíciles.

Mis padres, Cristina y Raymundo, que con el apoyo, la confianza y el gran aprendizaje de vida que me han brindado desde pequeña, fueron mis motores para concluir este proyecto, gracias por todo.

Mis hermanos, Marlen y Paulo, quienes me prepararon para defenderme en la vida, pero sobre todo me han enseñado que siempre debemos estar unidos sin importar las diferencias.

Mis niñas hermosas, Amy y Angy, mis pequeñas quienes le agregan una chispa a mi vida y por quienes procuro ser una mejor persona esperando ser un buen ejemplo para ellas.

Mi suegros, Paquita y Juan Javier, quienes me han apoyado incondicionalmente como si fueran mis padres, y siempre animándome para ser una persona más preparada.

A mis cuñados, Claudia y Alberto, personas sencillas y de buen corazón, que siempre han estado ahí en el momento en que he necesitado su apoyo.

A mi directora y codirector de tesis, así como a los demás profesores, quienes me brindaron su conocimiento y orientación para la realización de este trabajo.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Antecedentes

Este trabajo busca aportar a la resolución de problemas de selección de cartera de proyectos de gran escala y con restricciones de calendarización. Aquí se propone un algoritmo híbrido, haciendo uso de enfoques evolutivos y estrategias paralelas, así como técnicas de descomposición para mejorar su eficiencia.

El problema de selección de cartera de proyectos es uno de los más importantes que actualmente se están abordando, según la literatura, en diversas disciplinas, tanto en el sector privado como en el sector público (Salo, 2011) (Kleinmuntz, 2011). Llevar a cabo la selección de los proyectos se trata de una decisión crítica, y el problema puede llegar a ser de alta complejidad. Además y de acorde con el numeral 2, al incorporar restricciones de calendarización a la selección de cartera de proyectos se hace aún más complejo (Gutjahr, 2008). Para tratar con esta dificultad, trabajos previos han mostrado que es posible combinar algunas estrategias para mejorar el rendimiento y la calidad de los resultados. Particularmente, los algoritmos híbridos han aportado buenos resultados, por ejemplo, se han combinado algoritmos exactos y meta-heurísticas.

Los factores que pueden afectar a un algoritmo en su desempeño, en la aplicación de este problema, podrían ser:

1. El tamaño del problema.
2. Restricciones duras.
3. Estrategias algorítmicas utilizadas.

Por otra parte, en la literatura existen diversos trabajos de optimización enfocados a cartera de proyectos con calendarización, pero los trabajos que abordan el reto del crecimiento en el tamaño del problema (numeral 1), con estrategias paralelas y de descomposición, son escasos.

Uno de los trabajos que utilizan un enfoque cercano es la versión de cartera con calendarización propuesta por Ghahremani (2015). Los autores proponen calendarizar las tareas que implican la realización de cada proyecto maximizando el beneficio, este trabajo utiliza un algoritmo de optimización de colonia de hormigas y precedencia en las tareas pero no aborda la gran escala.

Los numerales 2 y 3, relativos al tamaño del problema y estrategias de solución, respectivamente, toman una relevancia particular cuando el problema de carteras involucra proyectos con muchos objetivos (numeral 1). Para tratar con estos factores de desempeño, en esta tesis, el algoritmo propuesto incorpora de manera original estrategias de hibridación, paralelización y descomposición con resultados satisfactorios con respecto a algoritmos del estado del arte.

1.2. Objetivos

Los objetivos de este trabajo se encuentran dentro de esta subsección.

1.2.1. Objetivo general

Diseñar un algoritmo evolutivo paralelo que resuelva un problema de selección de cartera de proyectos con calendarización de gran escala.

1.2.2. Objetivos específicos

Los objetivos específicos que favorece el cumplimiento del objetivo general son:

- Analizar el estado del arte para identificar características permitan la descomposición de problemas de selección de cartera con calendarización.
- Analizar y seleccionar una estrategia evolutiva paralela que permita la solución de problemas de selección de cartera con calendarización a gran escala.
- Diseñar una metodología de solución híbrida eficiente basada en la estrategia evolutiva paralela seleccionada.
- Publicar los resultados mediante la escritura de artículos.

1.3. Justificación

Actualmente existen variedad de empresas públicas y privadas que tienen la necesidad de encontrar la mejor cartera de proyectos, y así, hacer uso del presupuesto disponible de la mejor manera posible. Cabe mencionar que, existen algunas heurísticas que aportan buenas carteras, sin embargo, bajo todas las condiciones realistas posibles existe una probabilidad de que esa solución esté muy alejada de la cartera óptima. Particularmente, si en la formación de una cartera de proyectos, se adicionan restricciones de calendarización, es bien conocido que la complejidad del problema aumenta.

Además, el desarrollo tecnológico ha llegado a su límite, esto quiere decir que, la optimización ya no se reflejará en hardware, sino, en los algoritmos, los cuales mediante el uso óptimo de recursos mejoran el desempeño computacional de problemas de alta complejidad, Alcaraz Rodriguez (2015) dicen que:

"Con el fin de aprovechar de manera eficiente los procesadores, los algoritmos secuenciales deben dividir el trabajo a realizar de manera eficiente entre los diferentes núcleos del sistema, surgiendo la necesidad de nuevos modelos de programación."

Lo anterior describe el paralelismo, técnica que favorece el uso eficiente de los recursos de cómputo. Los algoritmos paralelos surgen ante la necesidad de cómputo requerida por problemas de extrema complejidad, cuyo tiempo de ejecución utilizando los tradicionales algoritmos secuenciales es prohibitivo.

En este trabajo se hará uso de algoritmos híbridos paralelo para mejorar la eficiencia y la eficacia del mismo, y explotar al máximo los recursos que aportan los equipos de cómputo actuales. Este algoritmo, dotado de inteligencia evolutiva, constituye una aportación original que hace posible la solución eficiente de ciertos problemas de optimización complejos del área de logística.

1.4. Alcances

- Solución de instancias del problema de selección de cartera con calendarización de al menos 500 variables.
- Solución de un problema de selección de cartera con calendarización cuya definición presente al menos una característica que posibilite la descomposición del problema.
- Este trabajo está soportado en el modelo de calendarización de cartera propuesto por Ghahremani (2015). Mediante adaptación y extensión incorpora características del problema de cartera de proyectos públicos.

- Se utilizó el modelo de Naderi para realizar las primeras evaluaciones del prototipo elaborado de esta tesis.

1.5. Limitaciones

- Se propone un algoritmo híbrido con enfoque paralelo para la solución del problema de cartera de proyectos.
- Las estrategias de paralelización estan basadas en técnicas de descomposición de problemas.
- La implementación solamente se implementó en una arquitectura paralela.

Capítulo 2

Marco Teórico

En esta sección se definen algunos conceptos sobre optimización, heurísticos, estrategias de paralelismo y técnicas de descomposición de problemas. Estos conceptos constituyen el soporte teórico de la investigación desarrollada en el presente trabajo.

2.1. Problemas de optimización

Un problema de optimización consiste en minimizar o maximizar el valor de una variable, buscando una solución que represente el valor óptimo para la función objetivo. Además se deben cumplir ciertas restricciones dadas por el problema, en caso de existir, para evitar soluciones no válidas. Un problema de programación lineal se define según Carranza Purca (2015), de la siguiente manera: Si queremos calcular x tal que:

$$\min_{x} f(x)$$

Sujeto a:

$$g_i(x) \geq 0, \text{ para } i = 1, 2, 3, \ldots, n$$

$$h_j(x) = 0, \text{ para } j = 1, 2, 3, \ldots, n$$

Donde f es un vector de k funciones objetivo $f = (f_1, \ldots, f_n), x$ es el vector de solución $x = (x_1, \ldots, x_r)^t$ de r variables, n es el número de restricciones de desigualdad; y p es el número de restricciones de igualdad.

2.1.1. Modelos de optimización

Un modelo es una interpretación matemática simplificada de una realidad difícil, el cual debe equilibrar la necesidad de visualizar todos los detalles del problema con la finalidad de encontrar técnicas de solución adecuadas (Andrés, 2007). Por lo tanto todo problema de optimización puede ser modelado de forma matemática.

Un modelo de optimización es donde existen múltiples variables de decisión las cuales están sujetas a un conjunto de restricciones imitando cualquier actividad humana. Los pasos para el desarrollo de un modelo de optimización según Ramos (2010) se muestra en la figura 2.1.

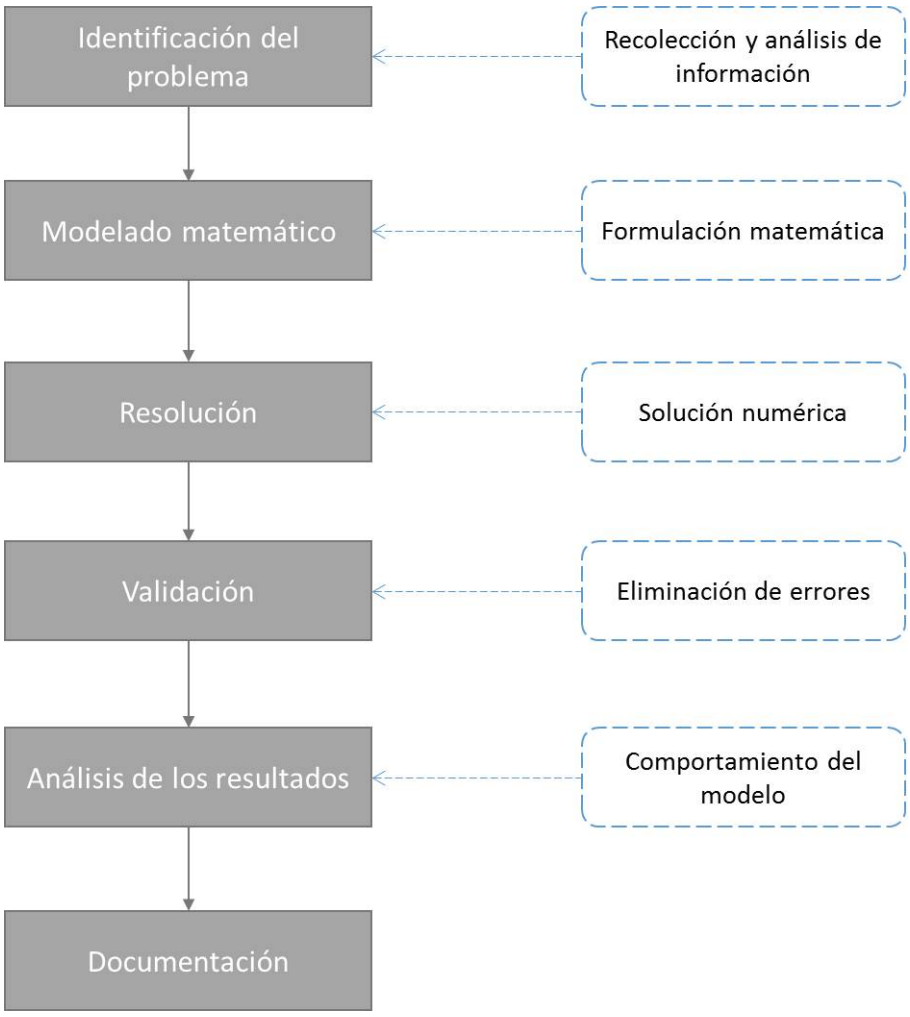


Figura 2.1: Etapas en el desarrollo de un modelo

Programación lineal

La programación matemática es un método de modelado usada en el proceso de toma de decisiones cuando se trata de resolver un problema. Las caracteristicas según Terrazas Pastor (2012) son:

1. La primera etapa consiste en identificar las variables del problema concreto. Comunmente, las variables son de carácter cuantitativo y se buscan los valores que optimizan el objetivo.

2. La segunda etapa implica determinar qué decisiones resultan admisibles; esto conduce a un conjunto de restricciones que se determinan teniendo presente las propiedades del problema

que estamos tratando.

3. En la tercera etapa, se calcula el coste o beneficio asociado a cada opción; esto determina una función objetivo que asigna, a cada conjunto posible de valores para las variables que determinan una decisión, un valor de coste o beneficio.

Todos estos elementos definen el problema de optimización.

2.1.2. Modelado con CPLEX

Existen varios sistemas comerciales de optimización que obtienen la solución exacta de un problema y son muy usados por la comunidad científica; entre ellos destaca el software CPLEX de ILOG. En CPLEX los problemas se expresan usando un lenguaje de modelado basado en programación matemática. Se implementa mediante una librería en distintos lenguajes de programación como son, C, C++, Java, .NET y Python. Esta herramienta, es muy útil cuando tenemos problemas relativamente fáciles o pequeños, debido a que la solución la obtiene mediante branch and bound, simplex y planos de corte, por lo que conforme el tamaño del problema (numero de variables) crece, el tiempo ejecución puede incrementa, por su complejidad, sin embargo, es uno de los mejores en el área de optimización (Cplex, 2010).

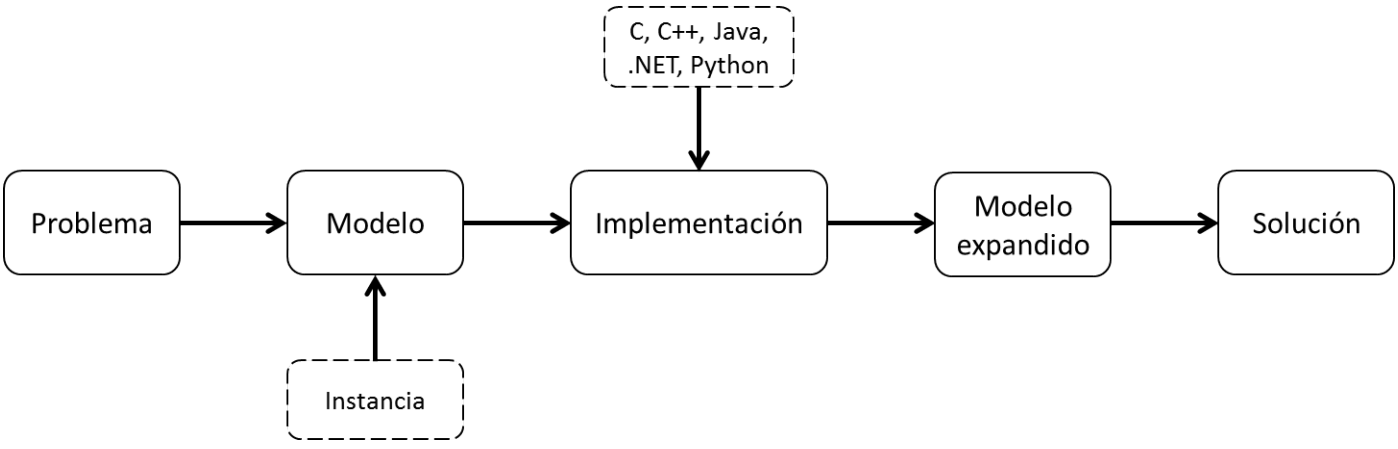


Figura 2.2: Proceso de solución de un problema en CPLEX

En la Figura 2.2, se muestra que inicialmente se contará con un problema, mismo que tiene que ser modelado matemáticamente de acuerdo a las características del problema, con su función objetivo, variables y restricciones, debe ser implementado en algunos lenguajes soportado por CPLEX mediante la librería adecuada, para ejecutar el modelo se debe introducir la instancia que se quiere evaluar, después se generará el modelo expandido (combinación del modelo con todas las posibles variables), por ultimo obtendremos la solución de la instancia dada.

2.1.3. Complejidad de algoritmos y problemas

Complejidad de algoritmos

El número de pasos necesarios para resolver un problema de tamaño n es la complejidad computacional. Esta complejidad se define normalmente en términos de un análisis del peor caso posible.

Existen tres tipos de notaciones generalmente usadas para mostrar la complejidad computacional de los algoritmos:

- Notación O-grande
- Notación Θ-grande
- Notación Ω-grande

Y los algoritmos son clasificados en aquellos de *tiempo polinomial* y de *tiempo exponencial*

Complejidad de problemas

La complejidad de los problemas es equivalente a la complejidad del mejor algoritmo encontrado que da solución al problema; y se considera un problema tratable ó fácil si existe un algoritmo

polinomial que lo resuelva, y se considera intratable o difícil si no existe un algoritmo que lo resuelva en tiempo polinomial.

Los problemas de decisión se dividen principalmente en dos clases: P y NP.

Los problemas de clase P representan todos los problemas de decisión que pueden ser resueltos por una máquina determinista en tiempo polinomial.

Los problemas de clase NP representan todos los problemas de decisión que pueden ser resueltos por un algoritmo no determinista en tiempo polinomial.

2.1.4. Tratamiento de la gran escala

Debido a la magnitud de ciertos problemas optimización, problemas también llamados "difíciles" en donde el numero de variables y/o restricciones es muy elevado, es necesario aplicar algunas técnicas y/o métodos que facilitan la solución del mismo. Estas técnicas dan la oportunidad de adaptar otros métodos de manera natural como es el caso de las técnicas que se utilizan en este trabajo.

Descomposición del problema

Una estrategia usada para solucionar los problemas considerados difíciles son los métodos de descomposición, que nos permite dividir un problemas de gran dimensión, a múltiples problemas de tamaño mas pequeño mas sencillos e independiente, pero que a su vez juntos conformen perfectamente el problema original (Olabe, 2015).

La descomposición basada en modelos de optimización tiene el objetivo de solucionar problemas difíciles (de 100 a 10,000 variables y restricciones) además de que permite la identificación de las estructuras implícitas en el modelo matemático que puede satisfacer los requisitos de diseño de

alto nivel, tales como la concurrencia, modularidad y robustez, así como la disponibilidad de los recursos computacionales.

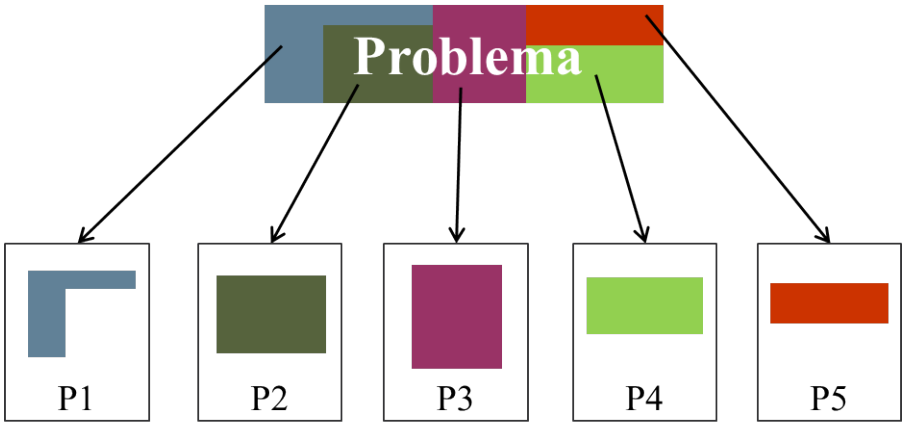


Figura 2.3: Descomposición de problemas

Cuando se trata con problemas de gran tamaño que no pueden ser resueltos en los equipos informáticos disponibles, suele recurrirse a técnicas de descomposición, que permiten fragmentar el problema (como en la Figura 2.3) y coordinar la resolución de los subproblemas para alcanzar la solución del problema completo.

Paralelización

La descomposición es también llamado paralelismo (ya sea de dominio o funcional), y lo que busca es ejecutar tareas pequeñas que son el resultado de la descomposición de los problemas. Esta división puede realizarse seccionando la memoria, donde a cada tarea le corresponde una porción de los datos, o bien se divide el trabajo a realizar en tareas; así, cada una es un conjunto de instrucciones y funciones (Villagra, 2009).

Entonces, de acuerdo a la descomposición de dominio desde la formulación matemática se da de manera natural una separación de tareas o subproblemas que simplifican la transmisión de información entre cada uno de ellos, por lo que pueden ser evaluados de manera simultanea.

2.2. Estrategias de optimización

Una clasificación básica de los algoritmos y las estrategias de optimización, es separarlos por estrategias deterministas (exactas) y no deterministas (aproximadas).

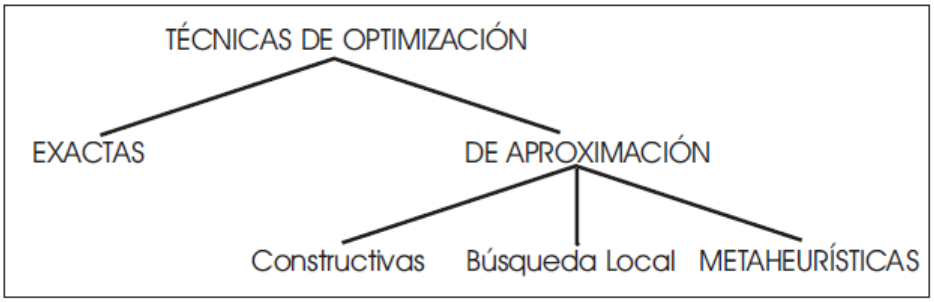


Figura 2.4: Clasificación de las técnicas de optimización
(Díaz, 2007)

Otra forma más concreta de representar una organización en cuanto a los algoritmos es en base a su definición: si es un algoritmo metaheurístico, híbrido o paralelo.

2.2.1. Métodos exactos

Los métodos exactos nos aseguran una solución óptima del problema, sin embargo el tiempo invertido por un método exacto para encontrar la solución puede llegar a ser muy superior en comparación con un método heurístico, el tiempo aumenta exponencialmente con el tamaño del problema, en ciertos casos es inaplicable.

Branch and bound: A partir de una formulación del problema, CPLEX obtiene su solución usando una combinación de métodos: branch and bound, simplex y planos de corte . Generalmente, los problemas de tiempo-polinomial son muy fáciles de resolver con CPLEX, sin embargo, es posible que instancias grandes de la vida real no puedan ser resueltas en un tiempo razonable. Dar solución eficiente a problemas de optimización difíciles, tanto polinomiales como no polinomiales, justifican

la necesidad de usar algoritmos aproximados, como las metaheuristicas, o su acoplamiento con resolvedores de programación matemática (Cplex, 2010).

2.2.2. Métodos aproximados

La existencia de una gran cantidad y variedad de problemas difíciles, que aparecen en la práctica y que necesitan ser resueltos de forma eficiente, impulsó el desarrollo de procedimientos eficientes para encontrar buenas soluciones aunque no fueran óptimas (Cunquero, 2003).

En estos métodos es importante la velocidad del proceso así como la calidad de la solución obtenida, aunque no se sabe con certeza el porcentaje de calidad de la solución, se tiene cierta confianza de que la solución es muy cercana a la óptima y factible. Los métodos heurísticos proporcionan soluciones cercanas al óptimo, en algunas ocasiones puede llegar a ser la solución óptima.

Algoritmo genético

Los algoritmos genéticos, desarrollados por J. Holland en 1970, son un tipo de algoritmos evolutivos muy populares. Estos algoritmos suelen utilizar un operador de cruza entre dos soluciones, y un operador de muta que modifica aleatoriamente la solución actual para promover la diversidad (Talbi, 2009).

Búsqueda Local

La búsqueda local es uno de los métodos heurísticos más simples y viejos. Empieza con una solución inicial, va iterando reemplazando la solución actual por una solución vecina que mejore la función objetivo, y se detiene cuando todos los posibles vecinos son una solución peor que la última actual; es decir, cuando alcanza un óptimo local (Talbi, 2009).

En la búsqueda de la mejora de la solución se implementó una búsqueda local que se basa específicamente del problema a tratar, si se quiere hacer una búsqueda local más general o menos específica de un problema es conveniente hacer uso de modelos matemático, los cuales nos describan el problema implícitamente mediante la función objetivo y sus restricciones. Un algoritmo básico de búsqueda local según Talbi (2009) es el siguiente:

Algorithm 1 Búsqueda Local Talbi (2009)

Salida:s mejor solución

```
1: Generar una solución inicial s = s_0
2: while criterio de terminación do
3:     Generar una solución s'
4:     if s' es mejor que s then
5:         s = s'
6: return s
```

Para la evaluación de las soluciones generadas es necesario hacer uso del conjunto de restricciones del modelo matemático propio de cada problema a evaluar, de la misma forma la función objetivo del problema debe ser evaluado de acuerdo a la solución obtenida, siempre y cuando esta solución no viole ninguna de las restricciones del modelo.

Se le debe dar a conocer la estructura de la solución a la búsqueda local para continuar con la misma nomenclatura y no tener ningún problema en la evaluación de la búsqueda local. En este caso se utilizan soluciones binarias por su facilidad de manipulación y evaluación.

2.3. Descomposición de problemas

De acuerdo a la literatura, la descomposición de un problema favorece el uso eficiente de los recursos de cómputo, mejorando el tiempo de ejecución y alcanzando soluciones competitivas comparadas a las reportadas en la literatura.

2.3.1. Estrategias de descomposición

Las técnicas de descomposición resuelven problemas de gran dimensión con una estructura atípica, que se aprovecha desde un punto de vista teórico y computacional, mediante la solución iterativa de otros problemas, derivados del original, de menor tamaño con estructura similar (Ramos, 2005).

Existen dos técnicas principales de descomposición que pueden considerarse como duales entre sí, ya que realizan la descomposición en dos dimensiones transversales. Los algoritmos de descomposición aplican la metodología de divide y vencerás. Dicha metodología tiene el objetivo de subdividir el problema en partes para su solución. En la literatura podemos encontrar diferentes estrategias de descomposición como:

- **Variables:** Descomposición de Benders.

  Benders en (Baena Mirabete, 2008), propone separar en subproblemas las decisiones tomadas en diferentes etapas. Para ello se necesita que las decisiones de una etapa sólo dependan de las consecuencias de las decisiones tomadas en la etapa anterior. Con esta descomposición se plantea un problema por cada etapa, y en ese problema se incluye tanto la parte correspondiente a la propia etapa como la parte que liga esa etapa a las decisiones tomadas en la etapa anterior.

- **Restricciones:** Descomposición Lagrangiana.

  La descomposición Lagrangiana (Chen, 2005) se refiere a la manera de particionar la matriz de incidencia desordenada originalmente en una matriz bien ordenada conocida como la matriz de bloque angular en la cual cada bloque indica un subproblema del problema original. En general hay dos patrones de descomposición: ideal y basado en la coordinación.

Regularmente una matriz de restricciones es diagonal por bloques con variables, restricciones o ambos que la pueden hacer mas complicada. Por lo tanto se debe realizar un análisis de cual es

la técnica mas conveniente para resolver el problema, debido a que algorítmicamente sólo tienen sentido cuando existe una ventaja computacional en su uso.

2.3.2. Descomposición basada en relajación lagrangeana

La descomposición Lagrangeana cuya idea se basa en descomponer un problema original restringido como en la Figura 2.5, complejo de resolver, de modo de reemplazarlo por otro problema que permita simplificar la resolución. Esto se logra incorporando aquellas restricciones que se consideran difíciles (las que hacen compleja la resolución directa del problema) a la función objetivo, donde cada una de éstas tendrá asociada un parámetro llamado multiplicador de Lagrange u que permitirá (iterativamente) penalizar el incumplimiento de las restricciones difíciles al ser establecidos distintos valores para estos (Fisher, 1985). De esta forma se espera que las restantes restricciones (las que no se incorporan mediante penalizaciones en la función objetivo) permitan verificar un problema cuya resolución sea fácil.

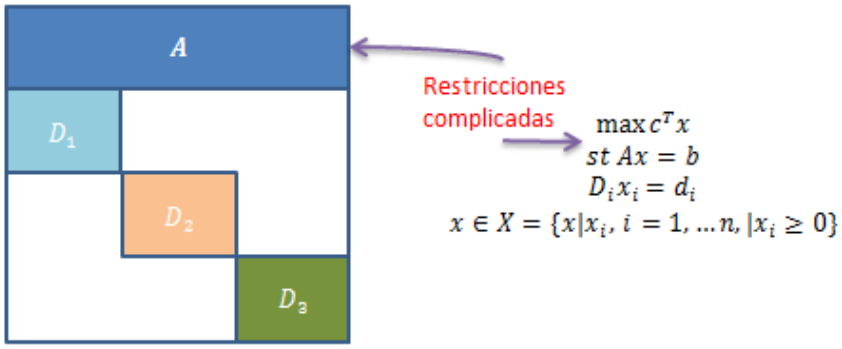


Figura 2.5: Modelo con restricciones difíciles

La descomposición lagrangeana es un caso en particular de la relajación lagrangeana, como se mencionó anteriormente, en este método se definen variables para cada conjunto de restricciones y se agregan restricciones de igualdad consideradas "restricciones difíciles" a la función objetivo con una penalización (Guignard, 1987).

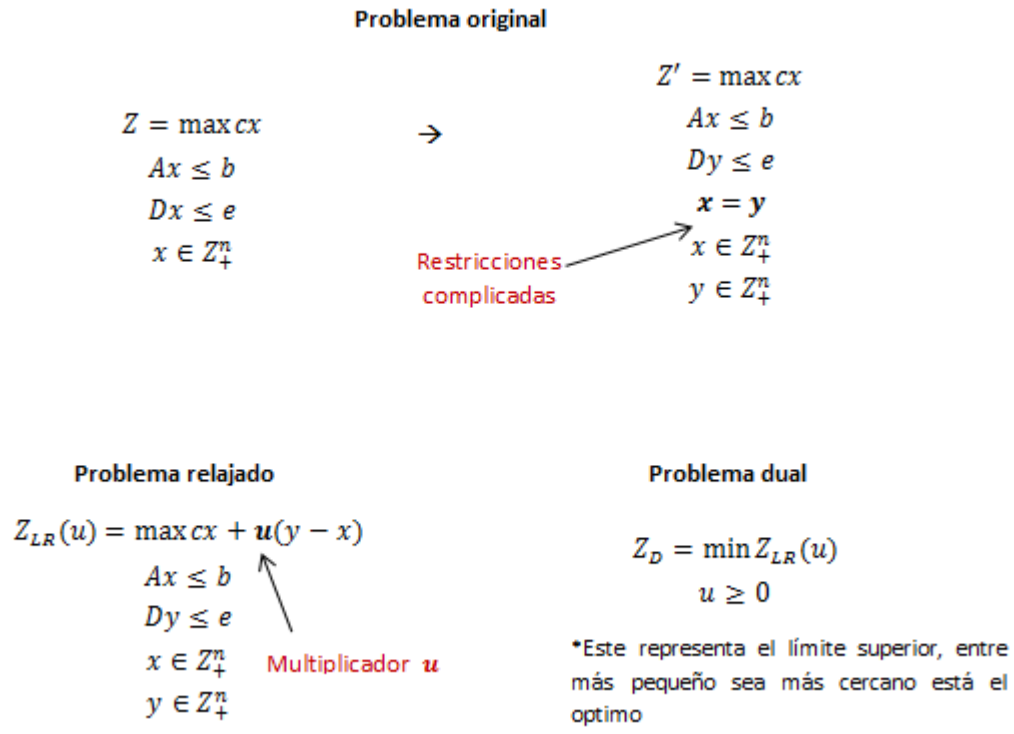


Figura 2.6: Modelo con restricciones difíciles

En la Figura 2.6 se muestra el problema original para después agregarle "restricciones complicadas" que se componen de las variables duplicadas igualadas a otra variable que no existe hasta el momento, posteriormente se forma el problema relajado subiendo todas las "restricciones complicadas" a la función objetivo eliminando la igualdad, cada una de estas restricciones serán multiplicadas por una variable llamada multiplicador, el cual es evaluado de manera independiente al problema pero si usando sus restricciones, al final se forma el problema dual, donde se definirá si el problema se minimizará o maximizará.

Heurístico Lagrangeano

Este algoritmo ayuda a resolver de manera aproximada o exacta los múltiples problemas derivados de un problema original explotando la estructura particular del modelo. Lo anterior es posible debido al cálculo de los multiplicadores Lagrangeanos para obtener la solución de cada subproblema,

al mismo tiempo es posible que nos ayuden a mejorar la solución del problema dual (Fisher, 1985).

El algoritmo es un proceso iterativo, y conforme se avanza se mejoran los resultados, que son el límites inferior y superior de la solución del problema.

Algorithm 2 Heurístico Lagrangeano (Beasly, 1993)

Salida:Z_{LB} limite inferior, Z_{UB} limite superior y mejor solucion

```
(sol, p(sol)) ← getInitialSolution()
bestSol ← sol
Z_LB ← p(sol)
Z_UB ← ∞
Inicializar multiplicadores u
steps = 0
while Z_LB ≠ ⌊Z_UB⌋ o steps ≠ maxsteps do
    (solP1, p(solP1)) ← SolveP1(u)
    (solP2, p(solP2)) ← SolveP2(u)
    UB_aux = solP1 + solP2 ← Actual limite superior
    if UB_aux < Z_UB then
        if UB_aux < ⌊Z_UB⌋ then
            steps = 0
        else
            steps = steps + 1
        Z_UB = UB_aux ← actualizar el limite superior
    else
        steps = steps + 1
    updatemultipliers(u) ← Actualizacion de multiplicadores
```

Como resultado Obtenemos limites superior e inferior, los cuales pueden ser introducidos en algoritmos exactos o aproximados y así disminuir el espacio de soluciones, ayudando a converger con más rapidez al algoritmo.

2.3.3. Necesidad de reordenamiento matricial

En matrices de problemas de optimización de gran tamaño es posible reorganizar filas y columnas para obtener una forma angular por bloques, esto favorece al ahorro de tiempo en cálculos. Esta reorganización permite aplicar métodos de descomposición explotando la estructura por bloques, tomando cada uno de los bloques como una parte del problema de forma independiente.

2.4. Algoritmos de reordenamiento matricial

La descomposición Lagrangiana asume que la matriz de incidencia del modelo matemático del problema de optimización, puede ser perfectamente descompuesta en problemas independientes en bloques desacoplados sin interacciones entre ellos como se observa en la Figura 2.7 , sin embargo esto representa otro problema embebido, debido a que la mayoría de las matrices de cualquier problema de optimización están totalmente desordenados siendo esto una limitante para la aplicación del método.

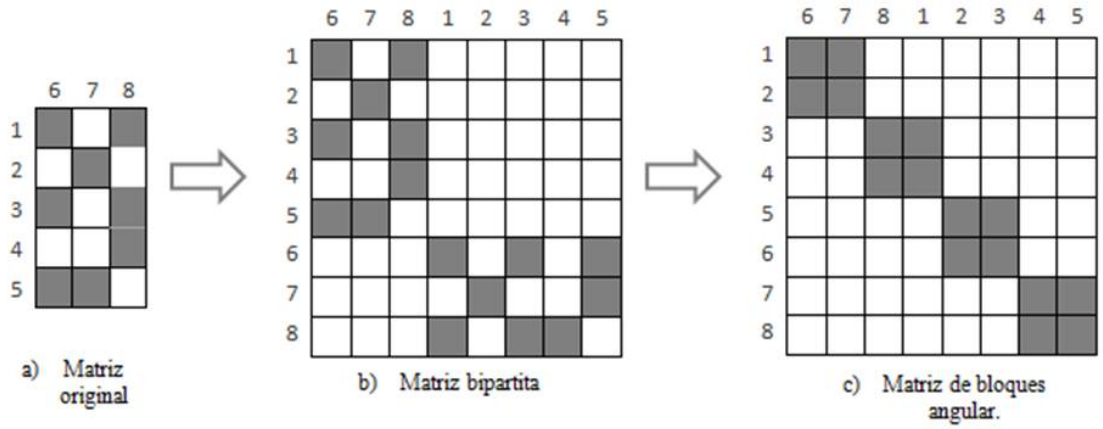


Figura 2.7: Descomposición de Matriz Esparcida.

Sin embargo existen diferentes métodos para minimizar el ancho de banda de matrices esparcidas y contribuir a obtener una matriz de bloques angular débilmente acoplada.

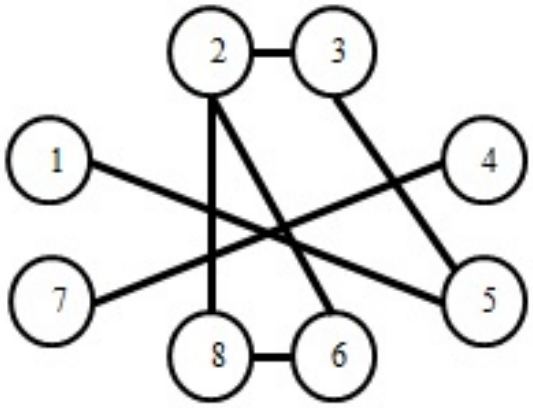
2.4.1. Algoritmo RCM(Reverse Cuthill-McKee)

El algoritmo Reverse Cuthill-McKee, o RCM (Cuthill, 1969), es un algoritmo bien conocido usado para la reducción del ancho de banda de matrices esparcidas, y que hace uso de estructuras de nivel. Una estructura de nivel es una partición de un grafo $G = (V, E)$ en niveles $L_1, L_2, ..., L_k$ sujeto a los vecinos $N(v)$ de un nodo v; éstos son asignados al mismo nivel de v, o en niveles

contiguos. El grado de un nodo en un grafo es el número de nodos adyacentes a la misma. El algoritmo de RCM para un grafo dado $G(n)$ es el siguiente:

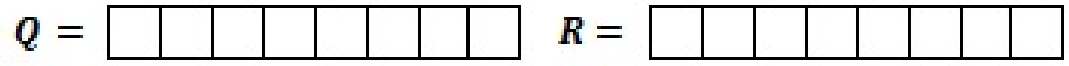
	1	2	3	4	5	6	7	8
1	1	0	0	0	1	0	0	0
2	0	1	1	0	0	1	0	1
3	0	1	1	0	1	0	0	0
4	0	0	0	1	0	0	1	0
5	1	0	1	0	1	0	0	0
6	0	1	0	0	0	1	0	1
7	0	0	0	1	0	0	1	0
8	0	1	0	0	0	1	0	1

Matriz de adyacencia

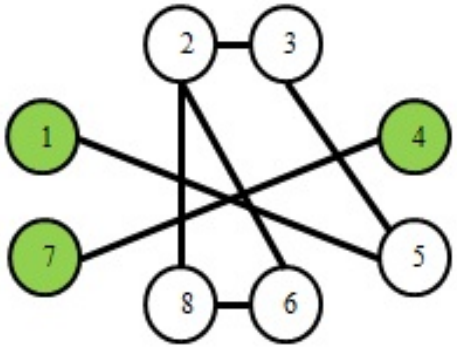


Grafo asociado

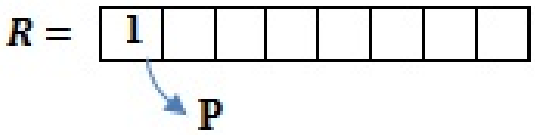
Paso 1: Preparar una cola vacía Q y un arreglo vacío de resultados R.



Paso 2: Los nodos con menor grado, sin explorar, que es 1, son el 1, 7 y 4, de esos elegimos uno arbitrariamente.



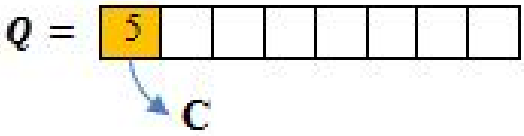
Paso 3: Tomamos el nodo 1 y lo agregamos a la primera posición de R:



Paso 4: Añadimos a Q todos los nodos adyacentes al nodo agregado a R en orden creciente de acuerdo a su grado, en este caso es el nodo 5:

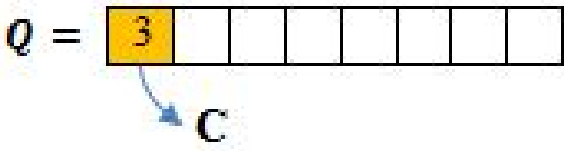
Paso 5: Si Q no está vacía:

Paso 5.1: Extraer el primer nodo de la cola, nodo 5:



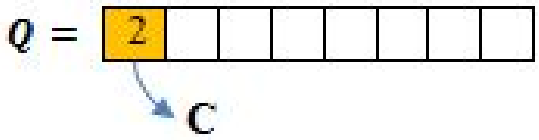
Paso 5.2: El nodo 5 no está en R, por lo tanto se agrega en la primera posición libre de R y se agregan los vecino del nodo 5 a Q, siempre y cuando no estén en R, el único vecino es el nodo 3:

Paso 5.1: Extraer el primer nodo de la cola, nodo 3:



Paso 5.2: El nodo 3 no está en R, por lo tanto se agrega en la primera posición libre de R y se agregan los vecino del nodo 3 a Q, siempre y cuando no estén en R, los vecinos son los nodos 5 (grado 2) y 2 (grado 3), pero el nodo 5 ya está en R, por lo tanto no se agrega a Q:

Paso 5.1: Extraer el primer nodo de la cola, nodo 2:

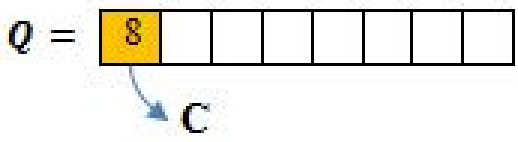


Paso 5.2: El nodo 2 no está en R, por lo tanto se agrega en la primera posición libre de R y se agregan los vecino del nodo 2 a Q, siempre y cuando no estén en R, los vecinos son los nodos 3 (grado 2), 8 (grado 2) y 6 (grado 2), pero el nodo 3 ya está en R, por lo tanto no se agrega a Q, y los nodos restantes se agregan en orden creciente de acuerdo a su grado, como tienen el mismo grado, se agregan arbitrariamente:

Paso 5.1: Extraer el primer nodo de la cola, nodo 6:

Paso 5.2: El nodo 6 no está en R, por lo tanto se agrega en la primera posición libre de R y se agregan los vecino del nodo 6 a Q, siempre y cuando no estén en R, los vecinos son los nodos 2 (grado 3) y 8 (grado 2), pero el nodo 2 ya está en R, por lo tanto no se agrega a Q, y el nodo 8 ya está en Q:

Paso 5.1 Extraer el primer nodo de la cola, nodo 8:

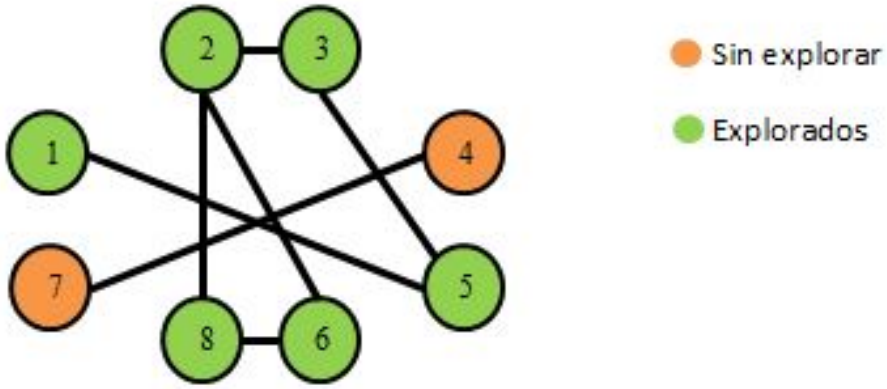


Paso 5.2: El nodo 8 no está en R, por lo tanto se agrega en la primera posición libre de R y se agregan los vecino del nodo 8 a Q, siempre y cuando no estén en R, los vecinos son los

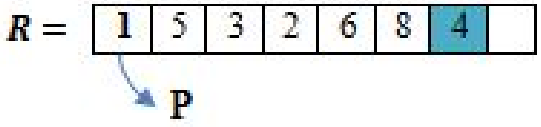
nodos 2 (grado 3) y 6 (grado 2), pero ambos ya están en R, por lo tanto no se agregan a Q, y Q queda vacia:

Paso 6: Si hay nodos sin explorar (el gráfico no está conectado) repita desde el paso 2:

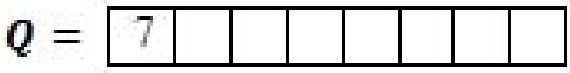
Paso 2: Los nodos con menor grado, sin explorar, que son grado 1, son el 7 y 4, de esos elegimos uno arbitrariamente.



Paso 3: Tomamos el nodo 4 y lo agregamos a la primera posición libre de R:

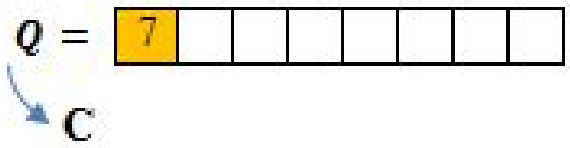


Paso 4: Añadimos a Q todos los nodos adyacentes al nodo agregado a R en orden creciente de acuerdo a su grado, en este caso es el nodo 7:



Paso 5: Si Q no está vacía:

Paso 5.1: Extraer el primer nodo de la cola, nodo 7:



Paso 5.2: El nodo 7 no está en R, por lo tanto se agrega en la primera posición libre de R y se agregan los vecino del nodo 7 a Q, siempre y cuando no estén en R, el único vecino es el nodo 4, el cual ya está en R:

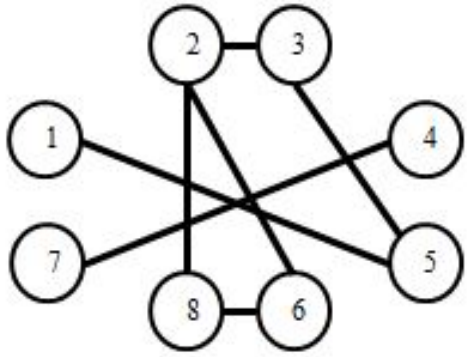
$R =$ | 1 | 5 | 3 | 2 | 6 | 8 | 4 | 7 | $Q =$ | | | | | | | | |

Paso 7: Invertir el orden de los elementos en R. Elemento $R[i]$ se intercambia con el elemento de $R[n+1-i]$.

$R =$ | 7 | 4 | 8 | 6 | 2 | 3 | 5 | 1 |

	1	2	3	4	5	6	7	8
1	1	0	0	0	1	0	0	0
2	0	1	1	0	0	1	0	1
3	0	1	1	0	1	0	0	0
4	0	0	0	1	0	0	1	0
5	1	0	1	0	1	0	0	0
6	0	1	0	0	0	1	0	1
7	0	0	0	1	0	0	1	0
8	0	1	0	0	0	1	0	1

Matriz de adyacencia



Grafo asociado

	7	4	8	6	2	3	5	1
7	1	1	0	0	0	0	0	0
4	1	1	0	0	0	0	0	0
8	0	0	1	1	1	0	0	0
6	0	0	1	1	1	0	0	0
2	0	0	1	1	1	1	0	0
3	0	0	0	0	1	1	1	0
5	0	0	0	0	0	1	1	1
1	0	0	0	0	0	0	1	1

Matriz de adyacencia

En resumen, el objetivo del RCM es reducir el ancho de banda de un gráfo reordenando los índices asignados a cada vértice. A los vértices se les asigna básicamente una orden de búsqueda

de amplitud, excepto que en cada paso, los vértices adyacentes se colocan en la cola en orden de grado creciente.

2.4.2. Algoritmo GPS(Gibbs, Poole and Stockmeyer)

Gibbs, Poole and Stockmeyer desarrollaron un algoritmo, llamado GPS, para reducir el ancho de banda en una matriz esparcida, este algoritmo también está basado en estructuras de nivel. El algoritmo fue creado inicialmente como una estructura de dos niveles usando el camino que describe el diámetro de la instancia del grafo. El nodo inicial y final, que definen el diámetro del grafo, fueron asignados al primer nivel L_1 de cada estructura. El resto de los nodos son organizados de tal manera que los vértices adyacentes están al mismo nivel o en un nivel contiguo. La combinación de los niveles de la estructura resulta en un nuevo nivel en el cual los nodos son etiquetados acorde a su nivel y su grado (Garey, 1976a).

2.5. Estrategias de paralelización

La necesidad que surge para resolver problemas que requieren tiempo elevado de cómputo ha dado lugar a lo que hoy se conoce como computación paralela. Mediante el uso concurrente de varios procesadores se resuelven problemas de manera más rápida que lo que se puede realizar con un solo procesador.

Una computadora paralela es un conjunto de procesadores que son capaces de trabajar cooperativamente para solucionar un problema computacional. El paralelismo está determinado por las estrategias de paralelización, el compilador y el sistema operativo (Pousa, 2016)

2.5.1. Procesamiento en Paralelo

La razón de ser del procesamiento en paralelo es acelerar la resolución de un problema, la aceleración que puede alcanzarse depende tanto del problema en sí como de la arquitectura de la computadora y el tipo de paralelismo que se quiere implementar.

Tipos de paralelismo

- **Paralelismo de datos.** La explotación del paralelismo de datos proviene de la constatación de que ciertas aplicaciones actúan sobre estructuras de datos regulares (vectores, matrices, etc.), repitiendo un mismo cálculo sobre cada elemento de la estructura. La idea es explotar la regularidad de los datos, realizando en paralelo un mismo cálculo sobre datos distintos (Pérez, 2014).

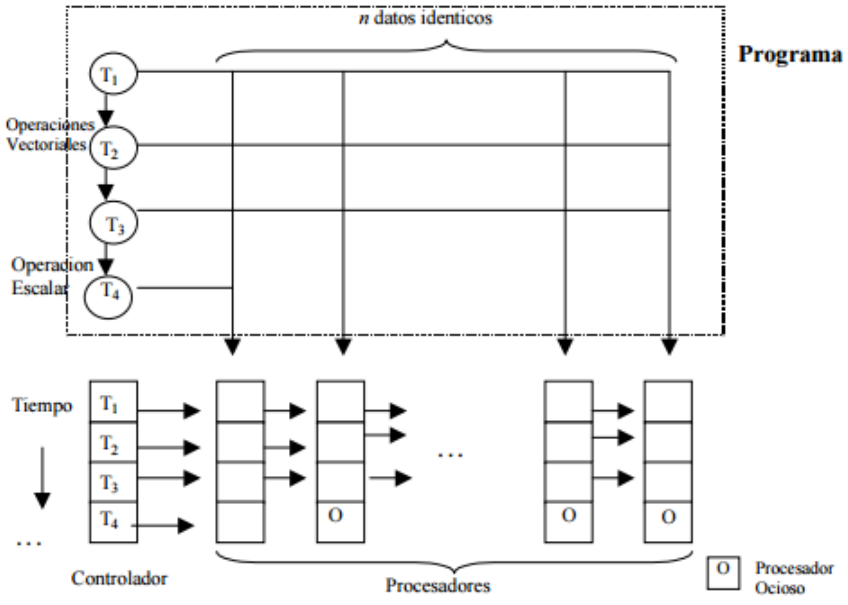


Figura 2.8: Paralelismo de datos.
(Aguilar, 2004)

- **Paralelismo de tareas.** En este enfoque se requiere una primera fase de descomposición del programa en tareas, para después definir cuáles se pueden ejecutar simultáneamente o

se deben ejecutar secuencialmente. Los criterios importantes a considerar son el número de tareas, el grano de paralelismo, y las dependencias entre las diferentes tareas (Pérez, 2014).

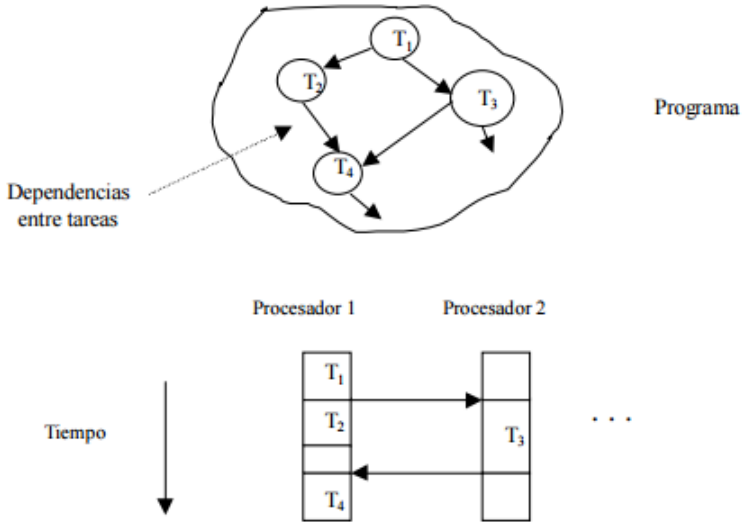


Figura 2.9: Paralelismo de tareas.
(Aguilar, 2004)

Arquitecturas

La clasificación de los sistemas paralelos de acuerdo a su acoplamiento:

- Los sistemas fuertemente acoplados son aquellos en los que los procesadores dependen unos de otros.
- Los sistemas débilmente acoplados son aquellos en los que existe poca interacción entre los diferentes procesadores que forman el sistema.

De acuerdo a estas características la clasificación que se les da se divide en dos tipos (Súarez, 2007).

Equipo Paralelo de Memoria Compartida. Un multiprocesador puede verse como una computadora paralela compuesta por varios procesadores interconectados que comparten un mismo sistema de

memoria. Cada procesador puede ejecutar una instrucción diferente y el acceso a la memoria se hace a través de una red de interconexión. La memoria se puede dividir en varios bancos, y en un momento dado, un banco puede solamente ser accesado por un procesador.

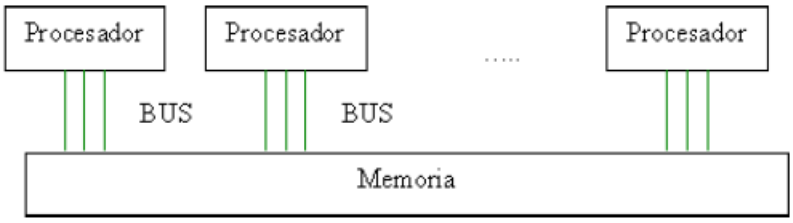


Figura 2.10: Memoria compartida.
(Revilla)

Equipo paralelo de memoria distribuida. Los sistemas multicomputadoras se pueden ver como una computadora paralela en el cual cada procesador tiene su propia memoria local. En esta arquitectura, cada procesador tiene una memoria local pequeña y recibe las instrucciones de una unidad de control central para ejecutar la misma instrucción, conjuntamente con el resto de procesadores, de manera sincronizada

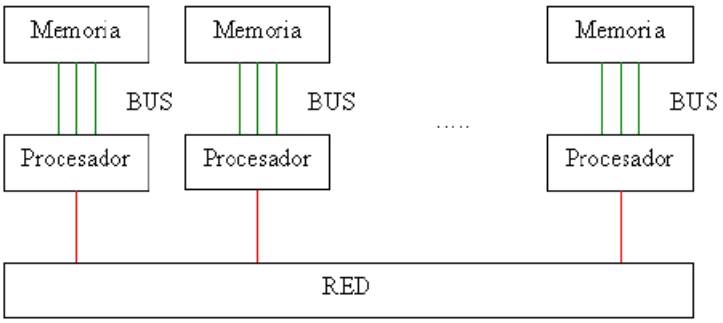


Figura 2.11: Memoria distribuida.
(Revilla)

El procesamiento paralelo puede ser aplicado en diferentes secciones de un algoritmo, como se muestra en los siguientes ejemplos:

- Solución de problemas. Cuando los problemas son totalmente independiente, éstos pueden ser evaluados al mismo tiempo:

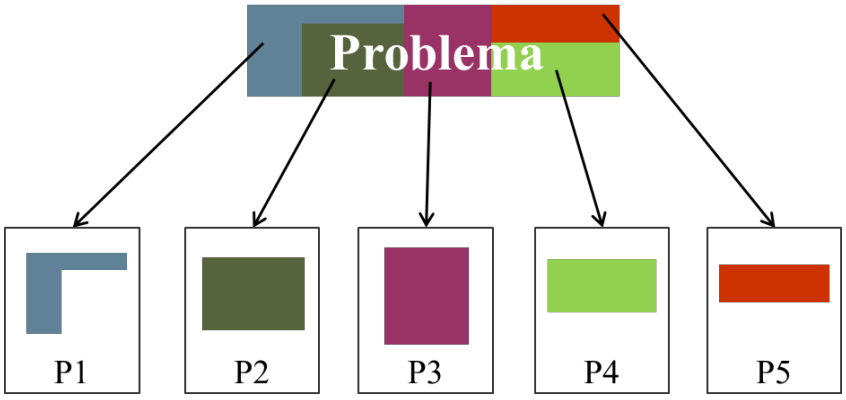


Figura 2.12: Problemas independientes.
(Revilla)

- Calculo de los multiplicadores. Por ejemplo, en la descomposición lagrangeana, para cada subproblema se requiere calcular de forma independiente y paralela cada uno de estos.

2.5.2. Ejemplos de tareas paralelas

2.5.3. Interfaz de programación paralela

Los supercomputadores o clusters en general, son capaces de realizar tareas de una forma muy rápida no solo por la calidad de los procesadores, sino especialmente por el reparto de trabajo que se hace de manera que a cada uno le corresponda una o varias subtareas. Mediante la comunicación entre los procesadores, es posible obtener el resultado final. El paralelismo se puede conseguir mediante varias interfaces o tecnologías que actualmente se están posicionando en área de la programación paralela.

Como se muestra en la Tabla 2.1 cada una de estas tecnologías tiene sus propias características, algunos funcionan mediante memoria compartida, otras con memoria distribuida o con ambas, sin

Tabla 2.1: Tecnologías para programación paralela

Tecnología	**Memoria compartida**	**Memoria distribuida**	**Diseño Escalable**	**Lenguaje de programación**
OpenMP	Si	No	Si	C, C++, Fortran
MPI	No	Si	Si	C, Fortran
JP	Si	Si	Si	Java
JOMP	Si	No	No	Java
CUDA	Si	No	Si	C

embargo lo que hace la diferencias entre estas es el o los lenguaje(s) de programación en los que funcionan. La disponibilidad de código, manuales y tutoriales son un punto importante, en el caso de OpenMP es una de las tecnologías para paralelizar de las cuales ha sido mas documentada, por lo que es mas sencillo implementarla.

OpenMP

Para implementar paralelismo con OpenMP se debe hacer referencia a la unidad más básica para la ejecución de programas paralelos, que es lo que comúnmente le llamamos "hilo" o "thread" por su nombre en ingles. La implementacion paralela sobre esta interfaz es mucho mas sencilla debido a que la memoria es compartida y la información podrá ser guardada en cualquier sector y el programa tendrá acceso a ella.

La definición de "hilo" es definida como un flujo de control secuencial simple (Castellanos, 2005), como es el caso de un programa secuencial como en la Figura 2.13.

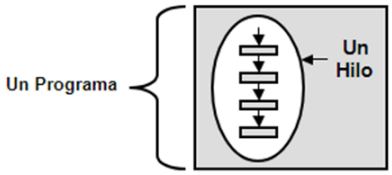


Figura 2.13: Programa secuencial.
(Castellanos, 2005)

Cuando hablamos de dos o mas hilos que componen un solo programa, entonces se hace referencia

a un programa paralelo (ver Figura 2.14).

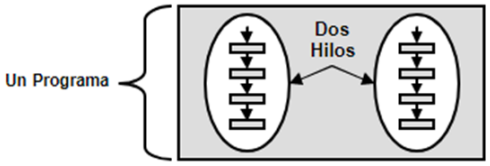


Figura 2.14: Programa paralelo.
(Castellanos, 2005)

En caso el caso de este trabajo, se utilizaron otros métodos para simplificar la aplicación de tecnologías paralelas, como por ejemplo descomposición de problemas, en donde un problema se dividió en múltiples problemas de tamaño mas pequeño y que juntos forman el problema original.

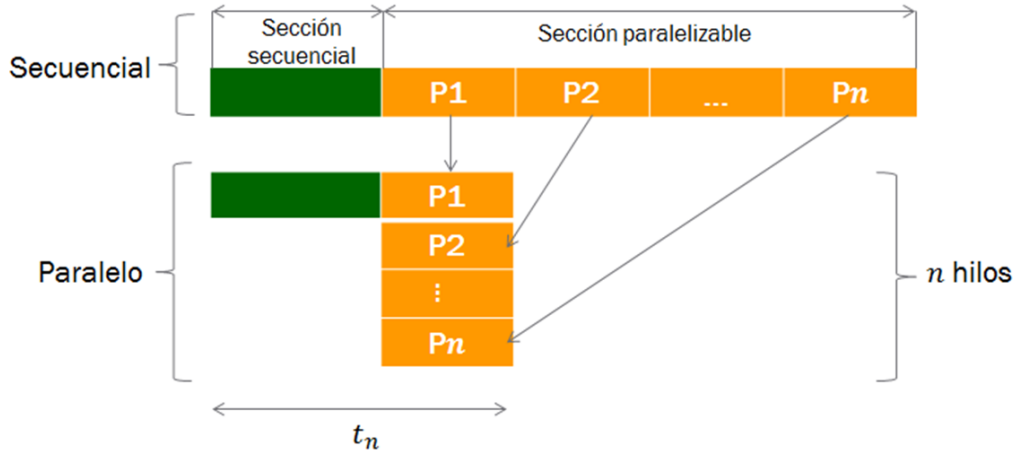


Figura 2.15: Programa de secuencial a paralelo.

Un ejemplo de la forma en como se representa el paralelismo es la Figura 2.15 en donde, si un programa secuencial tiene secciones de código independientes es posible ejecutarlas en el mismo periodo de tiempo (t_n) y en múltiples hilos (n), de esta manera es probable acortar el tiempo de ejecución.

Para la implementación de “threads” o “hilos” es necesario hacer uso de una directiva la cual es una forma de sincronización de un bloque de código que se ejecuta de forma paralela, donde se crean hilos según sea necesario como en la Figura 2.16.

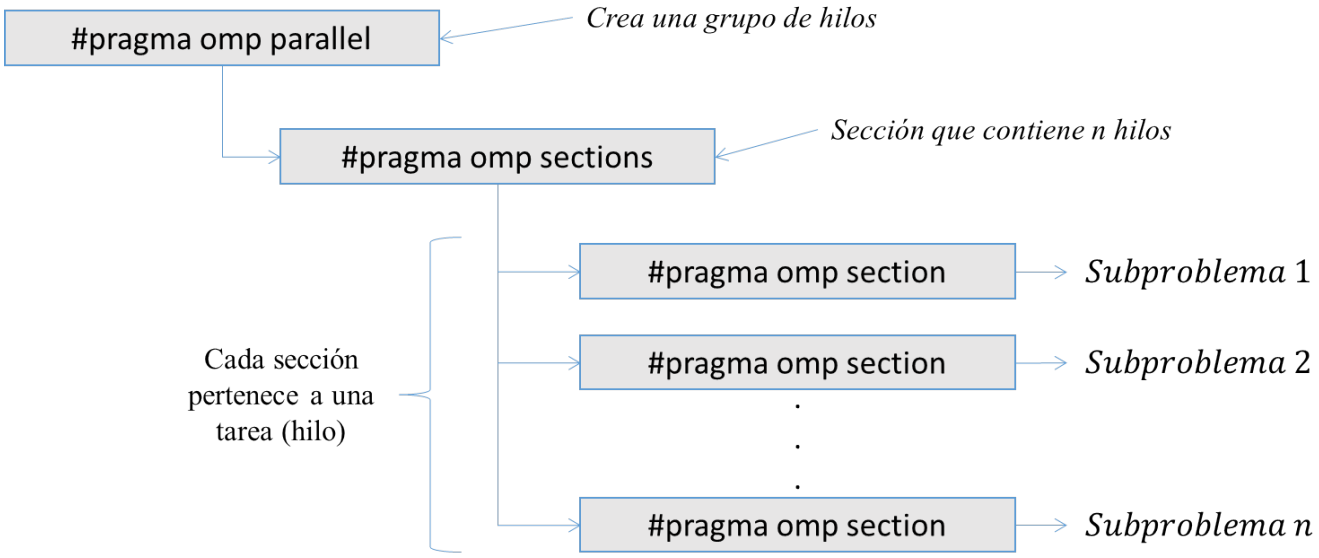


Figura 2.16: Ejemplo de generación de hilos.

2.5.4. Paralelización de algoritmos aproximados

En los algoritmos aproximados se encuentran los algoritmos evolutivos que son técnicas de optimización que trabajan sobre un conjunto (población) de potenciales soluciones (individuos) para que mediante la aplicación de un conjunto de operadores(muta y cruza) puedan progresar en la búsqueda de la solución óptima a un dado problema. En el especifico caso de los algoritmos genéticos existen unas estrategias que son llamadas “Paralelas”, los algoritmos genéticos celulares (Nesmachnow, 2002).

Un algoritmo genético Celular es un algoritmo, basado en una clase de población descentralizada en el que las soluciones evolucionan en barrios superpuestos (Soria, 2014). Dos de las estrategias donde fácilmente se puede observar el paralelismo son:

Estrategia maestro-esclavo. El esquema más simple de paralelización es la paralización global.

Un procesador central es el encargado de la selección, mientras que los procesadores esclavos son los encargados de hacer mutación y/o evaluación de la aptitud, por ejemplo. Aparentemente se ve como un algoritmo secuencial, sin embargo, su tiempo de ejecución puede ser menor.

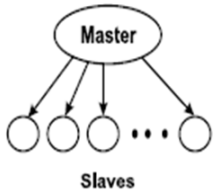


Figura 2.17: Estrategia maestro-esclavo
(Minetti, 2012)

Estrategia grano grueso. En este esquema se divide la población en islas, las cuales se ejecutaran en paralelo. Entre estas islas se hacen intercambios para introducir cierta diversidad, evitando que las soluciones queden en óptimos locales.

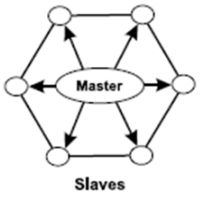


Figura 2.18: Estrategia grano grueso
(Minetti, 2012)

Este paralelismo implícito se consigue sin ningún dispositivo o memoria adicionales, sólo con la propia población.

Capítulo 3

Descripción del Problema

El capítulo inicia con la descripción del problema de investigación, objeto de estudio de la presente tesis, a partir de enunciarlo como un problema de optimización de carteras que implica la selección y calendarización de los proyectos que la conforman. Finalmente, para mostrar la relevancia científica del problema estudiado, se describe su complejidad computacional, y se ubica como un problema complejo que es al menos tan difícil como los que están en la clase NP-duro.

3.1. Problema de investigación

En esta tesis se aborda el problema de selección de cartera de proyectos con calendarización, por lo tanto, en los siguientes puntos, se describe en forma breve los problemas que lo componen, planteados de manera separada, selección de cartera de proyectos y calendarización, para después definir los dos problemas como uno solo.

De acuerdo con la literatura los problemas de calendarización (Chen, 1998) y selección de cartera de proyectos (Lin, 2001) son NP-Duro, por lo que su integración es al menos tan compleja como los problemas de esta clase.

3.1.1. Selección de Cartera de Proyectos

Según Nebro (2007) podemos definir el problema de selección de cartera de proyectos como:

Instancia:

Dado un conjunto finito C con n proyectos, denotados por c, donde $c \in C$, también cada proyecto se le asocia un valor $v(c) \in Z^+$ y un beneficio $b(c) \in Z^+$ y un $P \in Z^+$ que se refiere al presupuesto.

Pregunta:

Deseamos una asignación no negativa $u(c) = 0, 1$ tal que $\sum_{c \in C} v(c)u(c) \leq P$.

Una de las principales tareas de dirección en las organizaciones del sector público, fundaciones, centros de investigación y empresas que realizan investigación y desarrollo, consiste en evaluar un conjunto de proyectos que compiten por apoyo financiero, con el fin de seleccionar aquellos que aporten el máximo beneficio a la organización. Este subconjunto constituye una cartera de proyectos (Nebro, 2007). El problema de Selección de cartera de proyectos es un problema de programación lineal, la representación básica general del modelo matemático es la siguiente:

$$\begin{aligned} \text{máx } Z &= \sum_{i=0}^{n} b_i x_i \\ &\text{Sujeto a:} \\ \sum_{i=0}^{n} c_i x_i &\leq P \end{aligned} \tag{3.1}$$

$$x_i = \{0, 1\} \quad \text{para } i = 1, 2, 3, ..., n \tag{3.2}$$

Dado un conjunto de n proyectos donde cada proyecto tiene su propio beneficio y costo, además de un presupuesto definido, se pretende encontrar una cartera que maximice el beneficio total y respete el presupuesto determinado. Los parámetros e índices del modelo se encuentran en la Tabla 3.1.

Tabla 3.1: Parámetros del problema de cartera de proyectos.

Parámetro	**Definición**
n	Número de proyectos
i	Índice de proyectos $i \in 1, 2, 3, \ldots, n$
b_i	Beneficio del proyecto i.
C_i	Costo del proyecto i.
P	Presupuesto total disponible.

3.1.2. Calendarización de Tareas

De acuerdo con Salazar-Hornig (2013), la calendarización (JSSP) la podemos definir:

Instancia:

Dado un conjunto de máquinas M, un conjunto de trabajos J y un conjunto de operaciones O. Para cada operación $i \in O$ tenemos ligados un trabajo $J_i \in O$ al que pertenece y una máquina $m_i \in M$ en la que debe efectuarse en un tiempo $p_i \in R$ ininterrumpido y positivo. Además, es dada una relación de precedencia binaria $\prec$ que descompone O en cadenas, una para cada trabajo.

Pregunta:

Encontrar un tiempo inicial s_i para cada operación $i \in O$ tal que se minimiza el makespan, definido como $max_{i \in O}(s_i + p_i)$ tal que $s_i \geq 0 \forall i \in O$, y $s_j \geq s_i + p_i \forall i, j \in O$ con $i \prec j y s_j \geq s_i + p_i$ o $s_i \geq s_j + p_j \forall i, j \in O$ con $m_i = m_j$.

La calendarización es la planificación de tareas o actividades en un tiempo determinado, respetando restricciones de precedencia y recursos. La definición dada por Wren (1995), es la siguiente: "La asignación, sujeta a restricciones de los recursos otorgados con el propósito de ser establecidos en un espacio de tiempo, de tal manera que satisfaga lo más cercanamente posible el conjunto de objetivos deseados".

Uno de los objetivos que se considera en este problema es el makespan (C_{max}), que consiste en minimizar el intervalo de tiempo entre el inicio del procesamiento del primer trabajo (tiempo de

referencia 0) y el tiempo de terminación del procesamiento del último trabajo, es decir, el intervalo de tiempo en el que se procesa completamente la totalidad de los trabajos (Salazar-Hornig, 2013).

EL modelo matemático formulado por Rogers (1991) es el siguiente:

$$\text{mín}\, c_{max} \tag{3.3}$$

Sujeto a:

$$s_{j,1} \geq 0 \qquad j = 1, ..., n \tag{3.4}$$

$$s_{j,i+1} - s_{j,1} \geq p_{j,1} \qquad j = 1, ..., n;\ i = 1, ..., m-1 \tag{3.5}$$

$$(-s_{j,,r} + s_{l,,r} \geq p_{j,,r}) \vee (-s_{l,,r} + s_{j,,r} \geq p_{l,,r}) \forall (o_{j,,r}, o_{l,,r}) \in (M_r \times M_r), r = 1, ..., m \tag{3.6}$$

$$c_{max} - s_{j,m} \geq p_{j,m} \quad j = 1, ..., n \tag{3.7}$$

La restricción 3.4 define el tiempo de mínimo de inicio para la primera operación en cada trabajo, estos son 0 y no negativos. En la restricción 3.5 se definen las secuencias tecnológicas para cada trabajo. Las restricciones disyuntivas 3.6 se aseguran que no se realicen dos operaciones simultáneamente en la misma máquina. Cuando se minimiza, las restricciones de makespan 3.7 definen C_{max}. Los parámetros e índices del modelo se encuentran en la Tabla 3.2.

Tabla 3.2: Parámetros del problema Job Shop Scheduling.

Parámetro	**Definición**
n	Número de trabajos o tareas
m	Numero de máquinas o procesadores
C_{max}	Makespan
j	Índice de trabajo $j \in 1, 2, 3, \ldots, n$
r	Índice de maquina $r \in 1, 2, 3, \ldots, m$
i	Indica la posición de la operación $i \in 1, 2, 3, \ldots, k_j$
M_r	Indica la maquina con índice r
$p_{j,i,r}$	Indica la duración del trabajo j en la maquina r con posición en operación i.
k_i	Valor de m
$s_{j,i,r}$	Tiempo de inicio del trabajo j en la maquina r con posición en operación i.

3.1.3. Selección de Cartera de Proyectos con Calendarización

En cuanto a la definición de selección de cartera de proyectos con calendarización de acuerdo con Bahman (2012), es:

Instancia:

Dado un conjunto de n proyectos donde cada proyecto tiene su propio beneficio. Para llevar a cabo cada proyecto se deben realizar un conjunto de m tareas. Para realizar cada tarea, se requieren algunos tipos de recursos. Todos los recursos disponibles de un tipo son limitados. Hay algunas relaciones de precedencia entre las tareas de un proyecto.

Pregunta:

La meta es seleccionar entre los proyectos, un subconjunto de proyectos con un beneficio total máximo, los cuales deben ser calendarizados con el fin de terminar los proyectos en un horizonte de tiempo determinado. El proceso debe hacerse de tal manera que se cumplan las restricciones de los recursos.

Consideramos un conjunto de proyectos n con cierto número de actividades m, con un horizonte de tiempo determinado. A excepción de la primera actividad, cualquier actividad de un proyecto se debe realizar después de la finalización de la actividad anterior. Para todos los proyectos, la i-ésima actividad de los proyectos necesita el mismo tipo de recurso. Por lo tanto tenemos L diferentes recursos. Debido a la limitación de recursos, solo un tipo de actividad puede llevarse a cabo a la vez. El objetivo es maximizar el valor presente neto del rendimiento de los proyectos seleccionados que se pueden completar en el tiempo dado. El retorno es sensible al tiempo de finalización de los proyectos. Por lo tanto, se utiliza un factor de descuento a lo largo del horizonte de planificación (Ghahremani, 2015). El modelo matemático es como sigue:

$$\text{máx} \sum_{i=1}^{n} P_i X_i (1+r)^{-t_{i,l}} \tag{3.8}$$

Sujeto a:

$$C_{il} \geq C_{i,l-1} + t_{il}X_i \quad \forall_{i,l} \tag{3.9}$$

$$C_{il} \geq C_{k,l} + t_{il} - M(1 - Y_{ki}) - M(2 - X_i - X_k) \quad \forall_{i,l,i<n,k>i} \tag{3.10}$$

$$C_{kl} \geq C_{i,l} + t_{kl} - M(Y_{ki}) - M(2 - X_i - X_k) \quad \forall_{i,l,i<n,k>i} \tag{3.11}$$

$$C_{i,m} \leq T \quad \forall_i \tag{3.12}$$

$$C_{il} \geq 0 \quad \forall_{i,l} \tag{3.13}$$

$$X_i, Y_{ki} \in \{0,1\} \forall_{i,i<n,k>i} \tag{3.14}$$

Los parámetros e índices del modelo propuesto por Ghahremani y Naderi (2015), se encuentran en la Tabla 3.3

Tabla 3.3: Parámetros del problema de cartera de proyectos.

Parámetro	**Definición**
n	Número de proyectos
m	Número de actividades
i, k	Índices de los proyectos donde $i, k = 1, 2, \ldots, n$
l	Índices de los actividades donde $l = 1, 2, \ldots, m$
p_i	Beneficio del proyecto i
r	Porcentaje de descuento
$t_{i,l}$	Duración de la $i - esima$ actividad del proyecto i
T	Horizonte de tiempo de planificación
$C_{i,l}$	Variables continúa para el tiempo de finalización de la actividad l del proyecto i

La función objetivo 3.8 calcula la maximización del beneficio total descontado de los proyectos seleccionados. La restricción 3.9 asegura que para todos los proyectos, la $i - esima$ actividad no puede comenzar antes de que la $(i-1) - esima$ actividad sea finalizada. Las restricciones 3.10 y 3.11 se consideran para cada par de proyectos. Se aseguran de que si el proyecto i es programado después del proyecto j, para cualquier recurso, la actividad il comienza después de la finalización de la actividad kl. La restricción 3.12 hace cumplir todas las actividades de los proyectos seleccionados en el periodo de tiempo T. Por último, definen las variables de decisión la restricción 3.13 que toma

Tabla 3.4: Duración de los cuatro proyectos con sus tres actividades.

Actividad	**Proyecto**			
	1	**2**	**3**	**4**
1	4	2	4	6
2	4	6	4	2
3	4	2	2	4

el valor de 1 si el proyecto es seleccionado y 0 si ocurre lo contrario y la restricción 3.14 toma el valor de 1 si el proyecto i es programado después del proyecto k, y 0 en caso contrario.

Ejemplo ilustrativo

Como ejemplo, tenemos el siguiente: dado una cartera con cuatro proyectos y tres actividades, los detalles de las actividades se muestran en la Tabla 3.4, y un horizonte de tiempo de 18 unidades. Se manejan dos casos, donde inicialmente se elige al proyecto 4 y posteriormente se elige el proyecto 2.

Con respecto a la Figura 3.1 podemos observar que cuando iniciamos con el proyecto 4 (a) cubrimos la menor parte de las actividades, dejando inconcluso el proyecto 1 y 2, a diferencia que iniciando con el proyecto 2 (b), donde solo queda incompleto el proyecto 1.

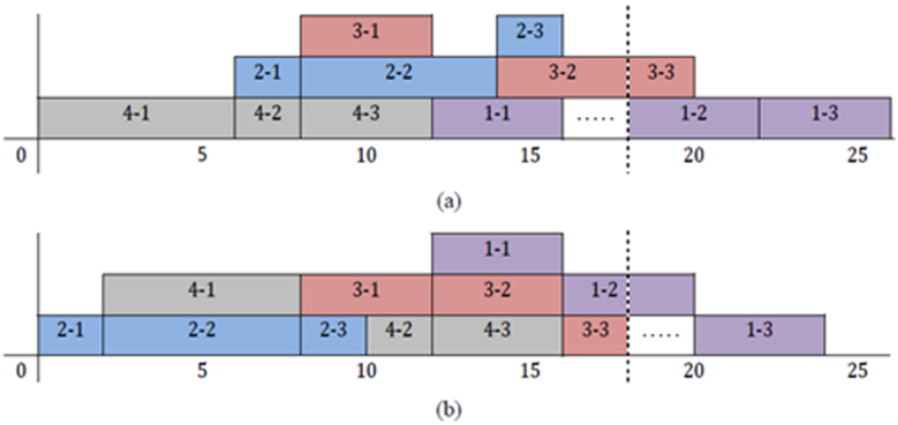


Figura 3.1: Gráfica de Gantt de las soluciones para el ejemplo de calendarización de actividades. (Ghahremani, 2015)

3.2. Complejidad del problema de investigación

Debido a su complejidad, el problema de calendarización de tareas de manufactura, mejor conocido como JSSP (Job-Shop Scheduling Problem) ha sido uno de los problemas más estudiados durante las últimas cuatro décadas, motivo por lo que es el problema que ha obtenido más avances en el área de la Calendarización. Además de estar clasificado dentro de la Teoría de la Complejidad como NP-Completo (Garey, 1976b), también se le conoce como uno de los problemas más difíciles de resolver en ésta clasificación. Debido a que este trabajo tiene embebido el JSSP en su definición, este trabajo es por lo menos tan complejo como el JSSP.

Capítulo 4

Estado del Arte

En esta sección se describe brevemente los trabajos relacionados al tema de selección de cartera de proyectos con calendarización, mostrándose algunas de sus características, con enfoque en el tipo de metaheurísticas empleadas, instancias y si se utilizan tecnologías y/o estrategias paralelas.

4.1. Revisión de trabajos relacionados con selección y calendarización

Modelado y solución de la selección y calendarización de proyectos

Tofighian, Ali Asghar and Naderi, B

Este trabajo resuelve la selección y calendarización de proyectos de forma bi-objetiva para optimizar tanto la variación del uso de recursos como el beneficio total esperado. Propone un algoritmo de optimización de colonia de hormigas el cual se evalúa comparándolo con un Algoritmo Genético y uno de Búsqueda Dispersa. El tipo de instancias utilizadas se generaron en base a instancias del problema Knapsack y no se utilizan estrategias paralelas Tofighian (2015).

Tabla 4.1: Comparación de trabajos relacionados con calendarización de tareas

Autor/ Año	**Problema**	**Metaheurísticas**	**Disponibilidad de instancias**	**Paralelo**	**Descomposición**
Tofighian (2015)	Cartera de proyectos y calendarización	Ant Colony Optimization (ACO)	Si (Basadas en instancias del problema knapsack.)	No	No
Summerville (2015)	Cartera de proyectos y calendarización	NSGA-II	N/D	No	No
Krzeszowska (2013b)	Cartera de proyectos y calendarización	Strength Pareto Evolutionary Approach 2 (SPEA 2)	N/D	No	No
Naderi (2013)	Cartera de proyectos y calendarización	Imperialist Competitive Algorithm (ICA). Genetic Algorithm (GA) Simulated Annealing	Si (Generadas mediante código)	No	No
Karunakaran (2016)	Calendarización	Algoritmo genético paralelo (modelo de islas).	Taillard	Si	No
Nesmachnow (2013)	Calendarización	Memetic Algorithm (MA) and TS , and a cellular MA	HSCP	Si	No
Propuesta: Algoritmo Híbrido Paralelo	Cartera de proyectos y calendarización	Algoritmo Híbrido paralelo	Si	Si	Si

Un algoritmo genético con llaves aleatorias para un problema de calendarización y selección de proyectos bicriterio

Natalia Summerville, Reha Uzsoy, Juan Gaytán

La selección de los proyectos y la calendarización está relacionado con la asignación de recursos limitados para proyectos que compiten con el tiempo para optimizar una función objetivo. Las actividades están sujetas a restricciones de precedencia, así como un presupuesto que limita el capital disponible en cada período de planificación. En este trabajo se implementa un algoritmo genetico y un procedimiento con NSGA-II, no aplican técnicas paralelas Summerville (2015).

Programación de la cartera de proyectos como un problema de toma de decisiones de criterios múltiples

Krzeszowska, B.

Este es un trabajo donde se aborda el problema de calendarización de la cartera de proyectos como un problema de toma de decisiones multiobjetivo, y le da solución utilizando una metaheurística Strength Pareto Evolutionary Approach 2 (SPEA 2). Lo realiza mediante tres pasos, primero formula el problema como un modelo matemático con minimización de penalizaciones por demoras en proyectos y maximización del uso de recursos, después detecta la soluciones no dominadas mediante un algoritmo elitista, y tercero aplica un procedimiento para elegir la solución Krzeszowska (2013a).

Problema de selección y calendarización de cartera de proyectos: modelo matemático y algoritmos

Naderi, Bahman.

Naderi investiga el problema de selección y calendarización de un conjunto de proyectos. Cada proyecto consiste de varias tareas y para desempeñar cada una requiere algún recurso. El objetivo es maximizar el beneficio total. Utiliza tres metaheurísticas: SPEA 2, GA y Simulated Annealing.

Las instancias utilizadas en la experimentación son generadas mediante un código generador de instancias del autor Naderi (2013). Cabe señalar que tampoco utiliza métodos paralelos.

Parallel multiobjective job shop scheduling using genetic programming

Karunakaran, D., Chen, G., and Zhang, M.

Desarrollaron un algoritmo evolutivo paralelo buscando reducir el consume de energía y el makespan usando un modelo de islas. Es aplicado para problemas multiobjetivo y utiliza instancias de Taillard, utilizan la metodología de islas para realizar su paralelización, sin embargo no utiliza tecnologías paralelas (OpenMP, MPI, CUDA, etc.) ni aborda el problema de selección de proyectos Karunakaran (2016) .

Algoritmos evolutivos multiobjetivos paralelos para la programación de lotes en sistemas heterogéneos de computación y grids

Sergio Nesmachnow

Se diseñan MOEAs paralelas, el más importante es el algoritmo celular el cual por su naturaleza es más fácil de paralelizar es un método eficaz para la optimización simultánea de la makespan y los objetivos tiempo de flujo, sin embargo no utiliza tecnologías paralelas ni aborda el problema de selección de proyectos. Se utilizaron las instancias de HSCP Nesmachnow (2013).

4.2. Revisión de trabajos relacionados con descomposición Lagrangeana

Como parte de la justificación del uso y énfasis de descomposición Lagrangeana, en la Tabla 4.2 se enlistan los artículos analizados en donde el uso de este tipo de métodos ha mejorado los resultados considerablemente en problemas de gran escala, y al mismo tiempo han ayudado a integrar otras estrategias para la solución de diferentes problemas de optimización.

Tabla 4.2: Relación de trabajos de descomposición de problemas revisados

Trabajo	**Autor/Año**
Hybrid Bilevel-Lagrangean Decomposition Scheme for the Integration of Planning and Scheduling of a Network of Batch Plants	"Bruno A. Calfa Anshul Agarwal, Ignacio E. Grossmann, and John M. Wassick (2013)"
Heurística lagrangiana para el problema de localización capacitado en dos etapas	Edith Lucero Ozuna Espinosa(2011)
Relajación lagrangiana paralela en la optimización de generación de un sistema hidroeléctrico.	"González F. Fariña L. Martínez E.Vargas & Arce A. (2014)"
Lagrangian decomposition to solve the problem of localized generalized p-median	"Giancarlo Montes Oblitas Jenny Rojas Gerónimo(2009)"
Lagrangian relaxation algorithm for a single machine scheduling with release dates	"Shujun Jiang Lixin Tang(2008)"

Hybrid Bilevel-Lagrangean Decomposition Scheme for the Integration of Planning and Scheduling of a Network of Batch Plants

Bruno A. Calfa, Anshul Agarwal, Ignacio E. Grossmann and John M. Wassic

Trata de la integración de planificación y programación en el funcionamiento de una red de plantas de procesos por lotes, en donde existe un tiempo horizonte dividido por periodos de tiempo los cuales deben ser satisfechos con éxito. Se utliza un modelo de planificación derivada de una planificación basada en precedencia. Resuelve problemas de gran tamaño mediante las siguientes estrategias de descomposición Bilevel y Temporal Lagrangean.

Heurística lagrangiana para el problema de localización capacitado en dos etapas

Edith Lucero Ozuna Espinoza

Este trabajo trata el problema de localización de instalaciones haciendo uso de estrategias de descomposicion como es la relajacion lagrangeana mediante dos etapas con la finalidad de econtrar las mejores cotas para el problema. Este caso no usa estrategias o metodos de paralelizacion. Se utilizan las instancias de Wildbore.

Relajación lagrangiana paralela en la optimización de generación de un sistema hidroeléctrico

González, F., Fariña, L., Martínez, E., Vargas, E., Arce, A.

Para el desarrollo de este trabajo se utilizaron estrategias de descomposicion lagrangeana y paralelización, para el problema de despacho optimo de unidades generadoras de un sistema hidroeléctrico. Fue ejecutado sobre un "cluster" contruido con unas computadoras personal, por lo que hablamos de poco poder de procesamiento. Se utilizó el método de maestro-esclavo, y la tecnologias para parelizar fue MPI, no se hace mencion de las instancias utilizadas.

Lagrangian decomposition to solve the problem of localized generalized p-median

C. Beltran C. Tadonki J. Ph. Vial

Se usa un método de relajación lagrangiana parcial para el problema de P-median, esto haciendo algunos ajustes en el método para obtener límites para instancias de gran escala. Se usan instancias de Avelle para el Problema de TSP. Los métodos se implementan sobre MATLAB, en C y en CPLEX.

Lagrangian relaxation algorithm for a single machine scheduling with release dates

Shujun Jiang, Lixin Tang

Por ultimo, este trabajo se enfoca al problema de calendarización de trabajos, tratando de minimizar el tiempo total de finalización de un conjunto de actividades. El problema pertenece a la clase NP-hard. El problema original es un problema de asignación donde las decisiones del modelo principal definen la asignación del intervalo de tiempo en el que se procesa cada actividad en una sola máquina. Propone implementar relajación lagrangiana (LR) para resolver el problema. La heurística basada en lagrangiana consta de dos fases: procedimiento de construcción y procedimiento de mejora. Como comparación utiliza el mismo problema implementado en CPLEX, por lo que deduce que en instancias relativamente pequeñas, da mejores resultados. No usa estrategias de paralelización.

4.3. Análisis comparativo

Las Tablas 4.1 y 4.2 contrastan las características relevantes de la presente tesis con las de los trabajos revisados de la literatura. Como conclusión se puede decir que no existe algún algoritmo paralelo que solucione el problema de cartera de proyectos con calendarización mediante estrategias de descomposición de problemas.

Como podemos observar existen algoritmos para calendarización que utilizan estrategias paralelas, sin embargo, no cuentan con alguna técnica descomposición, y las que si implementan las dos estartegias trabajan sobre otro conjunto de problemas de optimizacion. Este trabajo diseña un algoritmo híbrido paralela eficiente que hace uso de métodos de descomposición y paralelización.

En la Tabla 4.3 se muestra una comparativa entre todos los trabajos revisados en donde se especifica el tipo de paralelismo y estrategia de descomposición usada, en esta comparativa se demuestra que, al menos en los trabajos revisados, no se ha hecho uso de estrategias de descomposición incorporadas con tecnologías paralelas como son OpenMP y MPI, por ejemplo, como en el caso de este trabajo en el cual se implementa la descomposición lagrangeana aprovechando la naturaleza del método aplicando tecnologías paralelas como OpenMP.

Tabla 4.3: Comparativa de trabajos relevantes con la propuesta

Autor/ año	**Problema**	**Estrategias Paralelas**	**Tecnología Paralela**	**Descomposición**
Tofighian (2015)	Cartera de proyectos y calendarización	No	No	No
Summerville, N., Uzsoy, R., & Gaytán, J. (2015).	Cartera de proyectos y calendarización	No	No	No
Krzeszowska (2014)	Cartera de proyectos y calendarización	No	No	No
Bahman (2012).	Cartera de proyectos y calendarización	No	No	No
Karunakaran, D., Chen, G., & Zhang, M. (2016)	Calendarización	Si (paralelismo por islas)	No	No
Sergio Nesmachnow (2013)	Calendarización	Si (algoritmo celular)	No	No
Propuesta: Algoritmo Híbrido Paralelo	Cartera de proyectos y calendarización	Si (descomposición de problemas)	Si (OpenMP)	Si (Relajación lagrangeana)
Bruno A. Calfa Anshul Agarwal, Ignacio E. Grossmann, and John M. Wassick (2013)	Planificación y Programación de una Red de Plantas	Si (descomposición de problemas)	No	Si (Relajación lagrangeana)
Edith Lucero Ozuna Espinosa(2011)	Localización de capacitado en dos etapas	No	No	Si (Relajación lagrangeana)
González F., Martínez E.Vargas y Arce A. (2014)	Optimización de generación de un sistema hidroeléctrico	Si (descomposición de problemas)	Si (MPI)	Si (Relajación lagrangeana)
Giancarlo Montes Oblitas Jenny Rojas Gerónimo (2009)	p-mediana generalizada	No	No	Si (Relajación lagrangeana)
Shujun Jiang Lixin Tang (2008)	Calendarización	No	No	Si (Relajación lagrangeana)

Capítulo 5

Metodología de Solución

La selección y calendarización de proyectos, integrados en un solo modelo matemático, son el objeto de estudio de esta tesis. En esta sección se describe la metodología propuesta para dar solución a este problema en condiciones de gran escala. El enfoque de solución central se basa en la descomposición del problema original en varios problemas independientes que pueden ser resueltos de manera paralela para reducir el tiempo de procesamiento.

5.1. Adaptaciones al modelo matemático de selección de cartera de proyectos con calendarización

Chen (2005) proponen un modelo para la solución del problema de selección y calendarización de cartera de proyectos que es el siguiente:

$$Max \sum_{i=1}^{N} \sum_{t=EF_{iJ_I}}^{LF_{iJ_i}} P(i,t) \cdot C_{i,J_i,t}$$

Sujeto a:

$$\sum_{t=EF_{ij}}^{LF_{ij}} C_{ijt} = Y_i \qquad \forall i = 1,2,\dots,N;\ \ j = 2,\dots,J_i$$

$$0 \leq \sum_{t=EF_{ih}}^{LF_{ih}} (t - d_{ih}) \cdot C_{iht} \qquad \forall i = 1,2,\dots,N;\ \ ih \in S(i1)$$

$$\sum_{t=EF_{ij}}^{LF_{ij}} t \cdot C_{ijt} \leq \sum_{t=E_{ih}}^{LF_{ih}} (t - d_{ih}) \cdot C_{iht} \qquad \forall i = 1,2,\dots,N;\ \ j = 2,\dots,J_i - 1;\ \ ih \in S(ij)$$

$$\sum_{i=1}^{N} \sum_{j=2}^{J_i-1} r_{ijk} \sum_{q=\max\{t,EF_{ij}\}}^{\min[t+d_{ij}-1,LF_{ij}]} C_{ijq} \leq R_{kt} \qquad \forall k = 1,2,\dots,K; \quad t = 1,2,\dots,T$$

Donde,

Tabla 5.1: Parámetros del modelo de Chen

Parámetro	Definición
N	Número de proyectos candidatos
K	Numero de tipo de recursos
J_i	Ultima tarea por proyecto i
T	Tiempo horizonte
EF_{ij}	Tiempo para la finalización temprana de la tarea ij
LF_{ij}	Tiempo para la finalización tardía de la tarea ij
$P(i,t)$	Ganancia o descuento si el proyecto i es completado en el periodo t
d_{ij}	Duración de la actividad ij
$S(ij)$	Set de sucesores de tareas inmediatas ij
r_{ijk}	Recurso k requerido para la actividad ij
R_{kt}	Recurso k disponible en tiempo t

Se realizó una adaptación del modelo anterior para ser usado con las instancias de Naderi (2015), esto también para ser implementado en CPLEX. La adaptación incluye los siguientes cambios:

- Agregar un nodo cero como nodo inicial en cada uno de los proyectos (instancias).
- Se linealizó la función objetivo del modelo de Chen.
- Se eliminó la precedencia en el modelo de Chen, ejecutando todas las actividades de cada proyecto en forma consecutiva.

Entonces el modelo queda de la siguiente manera:

$$Max \sum_{i=1}^{N} \sum_{t=EF_{iJ_i}}^{LF_{iJ_i}} P(i,t) \cdot C_{i,J_i,t} \rightarrow Max \sum_{i=1}^{N} \sum_{t=EF_{im}}^{LF_{im}} (1+r)^{-t} \cdot C_{i,m,t}$$

Sujeto a:

$$\sum_{t=EF_{ij}}^{LF_{ij}} C_{ijt} - Y_i = 0 \qquad \forall i = 1,2,\dots,N;\ \ j = 2,\dots,m$$

$$\sum_{t=EF_{ih}}^{LF_{ih}} (t - d_{ih}) \cdot C_{iht} \geq 0 \qquad \forall i = 1,2,\dots,N;\ \ ih \in S(i1)$$

$$\sum_{t=EF_{ij}}^{LF_{ij}} t \cdot C_{ijt} - \sum_{t=E_{ih}}^{LF_{ih}} (t - d_{ih}) \cdot C_{iht} \geq 0 \qquad \forall i = 1,2,\dots,N;\ \ j = 2,\dots,J_i - 1;\ \ ih \in S(ij)$$

$$\sum_{i=1}^{N} \sum_{j=2}^{m-1} r_{ijk} \sum_{q=\max\{t,EF_{ij}\}}^{\min[t+d_{ij}-1,LF_{ij}]} C_{ijq} \leq R_{kt} \qquad \forall k = 1,2,\dots,m;\quad t = 1,2,\dots,T$$

Donde,

Tabla 5.2: Parámetros del modelo en CPLEX

Parámetro	Definición
N	Número de proyectos candidatos
m	Número de tareas
T	Tiempo horizonte
EF_{ij}	Tiempo para la finalización temprana de la tarea ij
LF_{ij}	Tiempo para la finalización tardía de la tarea ij
$P(i,t)$	Ganancia o descuento si el proyecto i es completado en el periodo t
d_{ij}	Duración de la actividad ij
$S(ij)$	Set de sucesores de tareas inmediatas ij
r_{ijk}	Recurso k requerido para la actividad ij
R_{kt}	Recurso k disponible en tiempo t

5.2. Algoritmo Híbrido Paralelo Basado en descomposición Lagrangeana (HPDL)

Mediante la aplicación de métodos de descomposición y paralelización, así como la aplicación de otros tecnicas y tecnologías, como CPLEX, algoritmos genéticos, algoritmos de reordenamiento matricial, entre otros, se da solución al problema de selección y calendarización de proyectos. El método propuesto es un algoritmo híbrido que integran de manera sinérgica los diversos enfoques antes mencionados.

5.2.1. Propuesta general

De manera general, para iniciar el proceso necesitamos una instancia específica del problema o acoplada al mismo. Después, la instancia se incorpora al modelo matemático de Naderi (2013), por ejemplo con la ayuda de CPLEX, de forma que queda especificado el modelo matemático de la instancia. Posteriormente se genera una matriz binaria del modelo instanciado, la cual será procesada por un algoritmo de reorganización matricial, ya dea recocido dimulado o RCM, para reordenarla de en forma escalonada generando subproblemas débilmente dependientes y entonces, mediante la adición de variables adicionales obtendremos subproblemas independientes, comunmente referidos

como bloques en la literatura. Por último, obtendremos los límites y/o el conjunto de soluciones, mediante la aplicación del método de descomposición lagrangeana, el cual esta basado en el uso de multiplicadores.

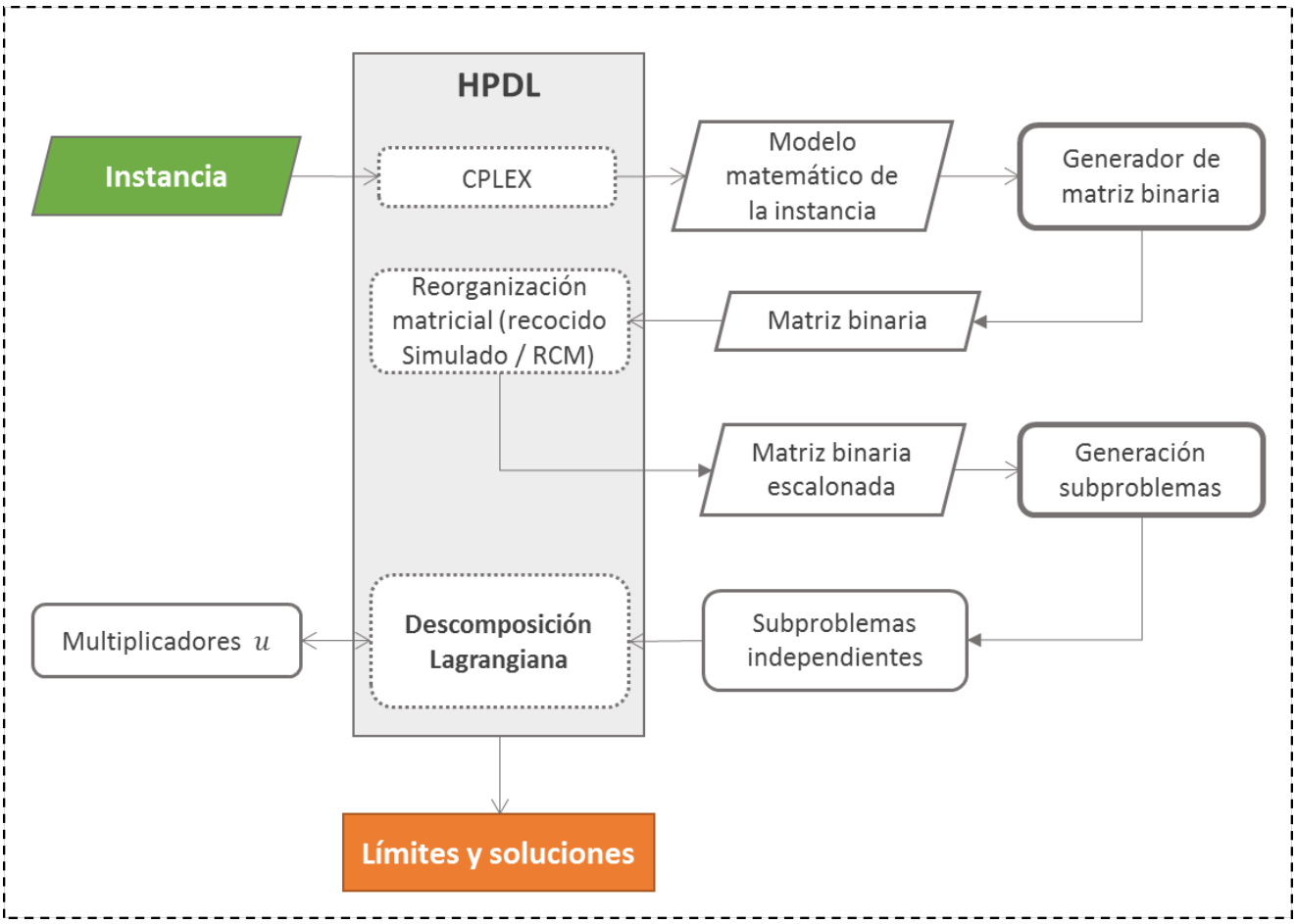


Figura 5.1: Diseño de la propuesta.

5.2.2. Generación de subproblemas

Para la generación de subproblemas, inicialmente se tienen bloques dependientes, mediante la identificación de variables dependientes, las cuales se encuentran en dos o mas bloques, se realiza una duplicación de las mismas, y adicionalmente se agrega la ecuación de asignación, de la variable anterior a la nueva (restricciones difíciles). Finalmente se obtendrán los bloques totalmente independientes.

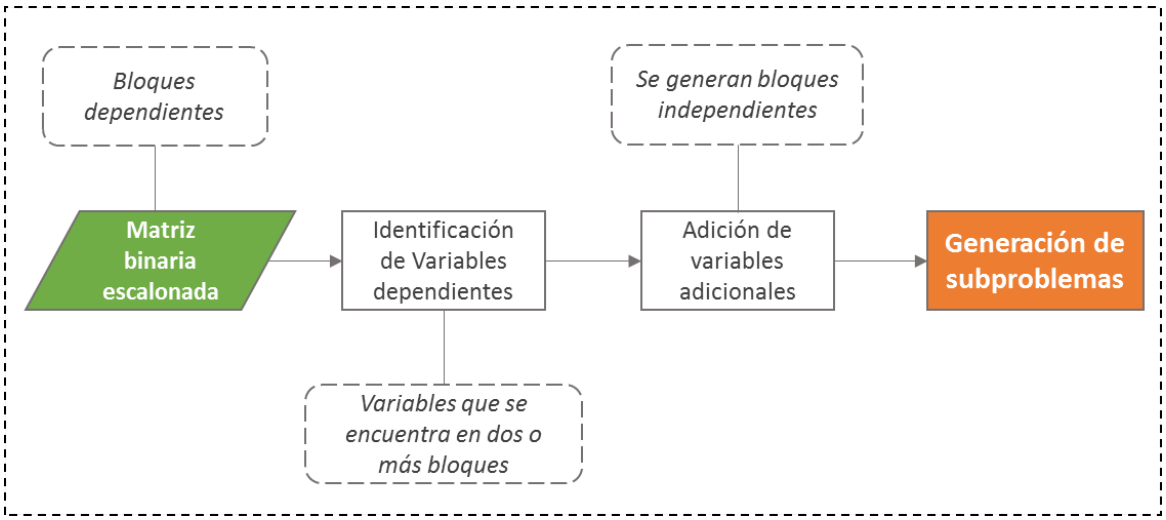


Figura 5.2: Generación de subproblemas.

5.3. HPDL como generador de soluciones iniciales de un método exacto

El algoritmo propuesto HPDL por si mismo ofrece una solución al problema de selección y calendarización de carteras. Adicionalmente, para mejora el resultado, HPDL puede ser usado en combinación con otros métodos de solución. Para un método exacto, las soluciones y los límites generados por el HPDL sirven para delimitar el espacio de búsqueda del algoritmo, en el caso de CPLEX es posible inyectarle un conjunto de soluciones, y así diminuir su tiempo de ejecución, disminuyendo su trayectoria de búsqueda y/o dándole una ruta alternativa para obtener la solución optima.

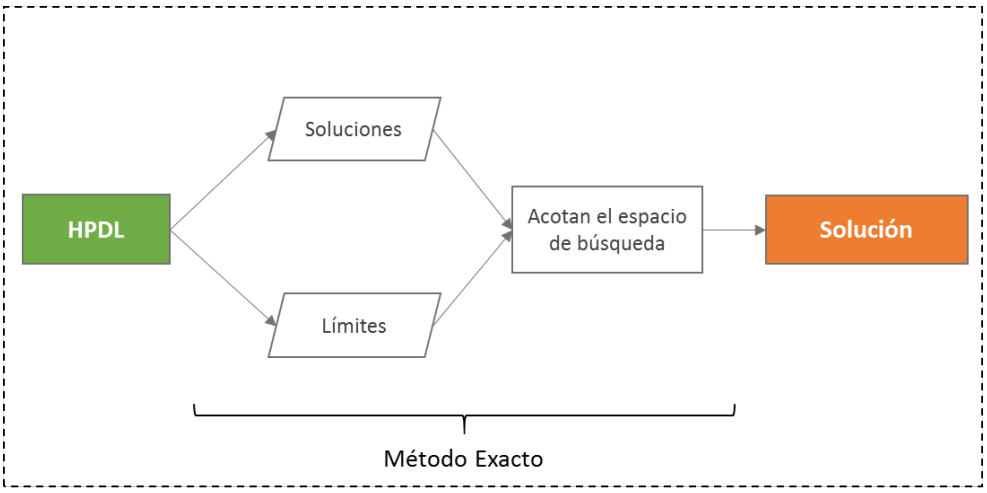


Figura 5.3: Generador de soluciones iniciales para un método exacto.

5.4. HPDL como generador de soluciones iniciales de un método aproximado

El método propuesto HPDL genera un conjunto de soluciones y límites, mismos que pueden ser usados en algoritmos evolutivos, como son los genéticos, ya que el conjunto de soluciones obtenidas pueden convertirse en la población inicial del algoritmo evolutivo, y/o también se puede hacer uso de los limites para acortar el espacio de búsqueda. Los límites pueden incorporarse como restricciones de valor solución.

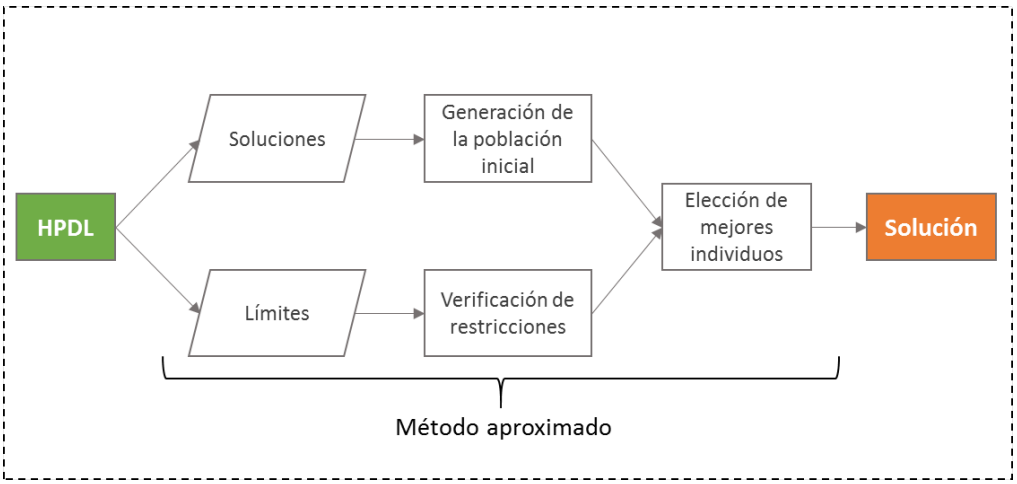


Figura 5.4: Generador de soluciones y límites para un método aproximado.

Capítulo 6

Experimentación y Resultados

Este capítulo muestra los resultados obtenidos por las diferentes experimentaciones así como el respectivo análisis de los mismos.

6.1. Experimento 1: Identificación de instancias de gran escala

Unas de las metas de este trabajo es utilizar instancias de gran tamaño o instancias difíciles de resolver, para esto se hizo una experimentación con el fin de identificar esas instancias que representan un reto para el modelo, esto para facilitar la evaluación positiva o negativa del algoritmo propuesto.

Para este experimento se corrieron combinaciones diferentes de instancias, con el fin de identificar alguna que contenga cierta dificultad para ser resuelta para el modelo de Chen (2009) y esta resolverla con el algoritmo propuesto para facilitar la validación del mismo.

Objetivo: El objetivo es obtener un conjunto instancias dada por el generador de Naderi (2015) que sea difícil de resolver por el modelo de Chen (2009).

Instancias: Las instancias utilizadas son las creadas mediante el generador de Naderi (2015), las cuales se observan en la Figura 6.1

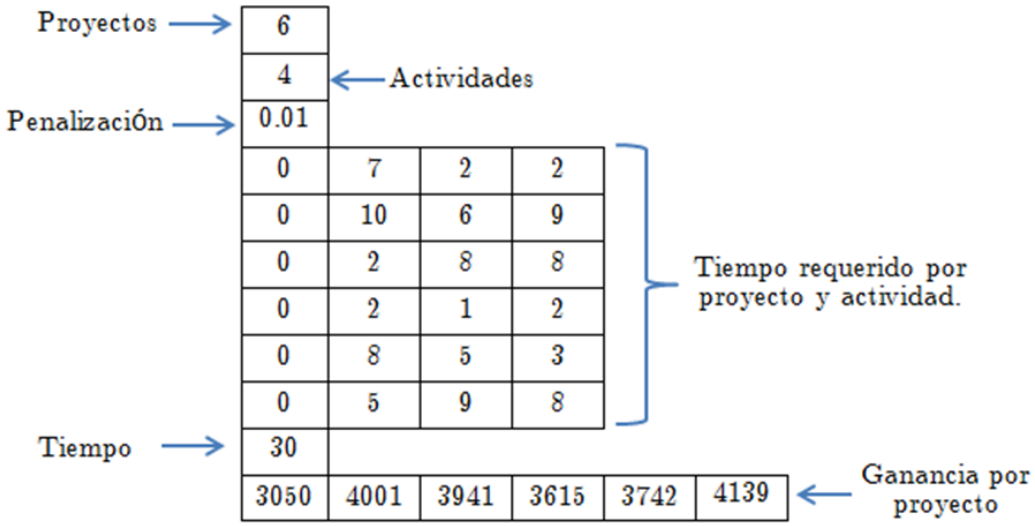


Figura 6.1: Instancia.
Ghahremani (2015)

Estas instancias fueron ejecutadas en un equipo con las siguientes características, procesador Core i5 a 1.8 Ghz con 4GB de RAM con sistema operativo Windows 7 sobre un entorno de desarrollo de entorno libre llamado NetBeans, en lenguaje de programación Java utilizando librerías de Cplex para solucionar problemas de programación lineal.

Análisis de los resultados: Se ejecutó código del modelo de Chen (2009), obteniendo la solución del modelo, en algunos casos, para determinar algunas instancias difíciles (con más de 500 variables). La definición de difícil se basó en el tiempo de ejecución para obtener la respuesta óptima y en el número de variables que contiene la instancia. Los resultados de esta experimentación se muestran en la Tabla 6.1. Este análisis sirvio para determinar que instancias eran adecuadas para la aplicación del método de descomposición.

Tabla 6.1: Evaluación de las instancias de Naderi.

Instancia	Solución	Estado de la solución	GAP	Tiempo
6-3	12024.21014	óptima	0	3 segundos
6-5	8861.005676	óptima	0	5 segundos
6-10	8077.077	óptima	0	30 minutos 14 segundos
8-3	18666.86208	óptima	0	3 segundos
8-5	13152.0779	óptima	0	19 segundos
8-10	11008.501	-	5.10 %	2 hora 30 minutos
10-3	19360.61847	óptima	0	16 segundos
10-5	17185.6083	óptima	0	6 min
12-3	23899.27521	óptima	0	6 minutos 13 segundos
15-3	34567.3023	óptima	0	15 minutos 2 segundos
15-5	30809.9653	-	1.67 %	10 horas 36 minutos

6.2. Experimento 5: Evaluación de multiplicadores iniciales

Para esta experimentación se utilizaron dos métodos de inicialización de multiplicadores para el método de descomposición lagrangeana.

- Método 1: Mediante la fórmula proporcionada por Fisher Fisher (1985), la cual es la siguiente:

$$u^{k+1} = \text{máx}\{0, u^k - t_k(b - Ax^k)\}$$

Donde,

$$t_k = \frac{\lambda_k(Z_D(u^k) - Z^*)}{(b - Ax^k)^2}$$

En donde,

u^k = multiplicador

t_k = escalar positivo

$(b - Ax^k)$ = dirección (restricción)

$Z_D(u^k)$ = óptimo del problema dual

Z^* = óptimo del problema dual

Tabla 6.2: Tiempo de convergencia de métodos para la generación de multiplicadores iniciales.

Instancia	**Metodo 1**	**Metodo 2**
6-3	4 segundos	7 segundos
8-5	21 segundos	28 segundos
10-5	8 minutos 13 segundos	9 minutos 7 segundos
15-3	20 minutos 55 segundos	26 minutos 34 segundos

- Método 2: La segunda forma y la más sencilla es inicializando todos los valores de los multiplicadores a cero.

Objetivo: El objetivo es encontrar el método de multiplicadores iniciales que ayude a converger de forma mas rápida al algoritmo.

Instancias: Las instancias utilizadas son las creadas mediante el generador de Naderi (2015), las cuales se observan en la Figura 6.1.

Estas instancias fueron ejecutadas en un equipo con las siguientes características, procesador Core i5 a 1.8 Ghz con 4GB de RAM con sistema operativo Windows 7, utilizando el lenguaje de programación de bajo nivel, C.

Análisis de los resultados: Se implementaron los dos métodos propuestos, haciendo la evaluación en cuanto el tiempo que tardó el algoritmo en converger, como resultado se obtuvo que al inicializar en cero los multiplicadores el tiempo de ejecución incrementa debido a la poca información que se le proporciona, esto afecta en que la exploración del multiplicador es más exhaustiva, por lo que se identifico que la mejor opcion es inicializar el multiplicador mediante el metodo 1.

Tabla 6.3: Tiempo de ejecución por métodos aplicados.

Instancia	Descomposición	Búsqueda local	Paralelismo
6-10	39 minutos 17 segundos	36 minutos 2 segundos	31 minutos 45 segundos
8-10	10 horas 38 minutos	4 horas 27 minutos	2 horas 51 minutos
12-3	11 minutos 43 segundos	7 minutos 38 segundo	7 minutos 5 segundos
15-3	20 minutos 55 segundos	16 minutos 3 segundos	14 minutos 23 segundos
15-5	21 horas 12 minutos	15 horas 19 minutos	10 horas 10 minutos

6.3. Experimento 4: Comparación por métodos y técnicas aplicadas

En esta experimentación se ejecutó la implementación múltiples veces, añadiendo cada vez uno de los métodos y/o técnicas planteadas, con el fin de evaluar el algoritmo con los diferentes métodos aplicados.

Objetivo: Visualizar el cambio y la mejora en terminos de tiempo de ejecucion al ser aplicado cada uno de los metodos propuestos para la mejora de la implementacion.

Instancias: Las instancias utilizadas son las creadas mediante el generador de Naderi (2015), las cuales se observan en la Figura 6.1

Estas instancias fueron ejecutadas en un equipo con las siguientes características, procesador Core i5 a 1.8 Ghz con 4GB de RAM con sistema operativo Windows 7, utilizando el lenguaje de programación de bajo nivel, C.

Análisis de los resultados: Se ejecutó el algoritmo conforme se iban implementando los métodos aplicados, en la segunda columna se muestran los tiempos de ejecución utilizando la técnica de descomposición, en la tercera columna se aplica la descomposición y una búsqueda local simple aleatoria, y su mejoría en algunas instancias es bastante evidente, finalmente en el caso de la aplicación de los tres métodos mejora aun más el resultado. Como se muestra en la Tabla 6.3

6.4. Experimento 6: CPLEX vs HPDL

Por último, se contrarresta el algoritmo propuesto HPDL contra la implementación en CPLEX, el cual hasta el momento es una herramienta que otorga los mejores resultados para múltiples problemas de optimización.

Objetivo: Visualizar los tiempos de ejecución de cada una de las implementaciones, con el fin de visualizar si fue posible o no una mejoría después de la aplicación de los métodos de descomposición y paralelización.

Instancias: Las instancias utilizadas son las creadas mediante el generador de Naderi (2015), las cuales se observan en la Figura 6.1

Estas instancias fueron ejecutadas en un equipo con las siguientes características, procesador Core i5 a 1.8 Ghz con 4GB de RAM con sistema operativo Windows 7, utilizando el lenguaje de programación de bajo nivel, C y de alto nivel Java.

Análisis de los resultados: Se ejecutaron cada uno de las implementaciones, la herramienta CPLEX sobre Java y el HPDL en lenguaje C. Como se comento antes, para resolver problemas de programación lineal, CPLEX, es de los mejores, sin embargo al ser aplicado con problemas con un gran numero de variables, por su complejidad, puede llegar a ser sumamente exhaustivo, por lo que en las instancias grandes (más de mil variables) podemos ver que el HPDL obtuvo el resultado en menor tiempo, sin embargo en los problemas mas pequeños, aun sigue logrando los mejores resultados. Esto muestra el potencial del HPDL para los problemas de gran escala.

Tabla 6.4: Comparacion de tiempos de ejecucion entre CPLEX y HPDL

Instancia	**Solucion**	**Estado**	**CPLEX**	**HPDL**
6-3	12024.21014	óptima	3 segundos	6 segundos
6-5	8861.005676	óptima	5 segundos	9 segundos
6-10	8077.077	óptima	30 minutos 14 segundos	29 minutos 45 segundos
8-3	18666.86208	óptima	3 segundos	5 segundos
8-5	13152.0779	óptima	19 segundos	30 segundos
8-10	11008.501	-	2 hora 30 minutos	2 horas 18 minutos
10-3	19360.61847	óptima	16 segundos	28 segundos
10-5	17185.6083	óptima	6 minutos 13 segundos	8 minutos
12-3	23899.27521	óptima	6 minutos 13 segundos	6 minutos 5 segundos
15-3	34567.3023	óptima	15 minutos 2 segundos	14 minutos 23 segundos
15-5	30809.9653	-	10 horas 36 minutos	9 horas 10 minutos

Capítulo 7

Conclusión

En este capítulo se presentan las conclusiones, así como el trabajo futuro que se propone para mejorar las estrategias para la selección y calendarización de proyectos.

7.1. Conclusiones

Debido a la naturaleza de los problemas analizados, calendarización y selección de cartera de proyectos, se determinó que la aplicación de métodos de descomposición es factible. Se inició analizando a nivel de modelo matemático, conociendo los diferentes métodos y evaluando la aplicación de cada uno de ellos, por los que se determinó por implementar la relajación lagrangeana.

de acuerdo al análisis del problema, y que se aplicaron métodos de descomposición de problemas, no fue necesario implementar estrategias evolutivas para incluir el paralelismo, sin embargo es una opción, el paralelismo se incluyó debido a la separación del problema original en subproblemas, evaluando cada uno de ellos simultáneamente mediante tecnologías paralelas como OpenMP.

La metodología de solución híbrida, se logro mediante la incorporación de diferentes métodos y estrategias, como son la herramienta para solución de problemas de optimización CPLEX, el

algoritmo para reordenamiento matricial, la descomposición lagrangeana y la búsqueda local simple aleatoria.

Para finalizar, el empleo de múltiples técnicas mejoró en ciertas condiciones algunos resultados obtenido por la herramienta exacta (CPLEX), por lo que es conveniente continuar con la implementación de este tipo de estrategias.

7.2. Trabajos futuros

De este trabajo se puede continuar con la realización de diferentes trabajos futuros que aquí se proponen:

- Analizar y estudiar sobre la implementación de otros métodos de descomposición como es el método de Benders.
- Mejorar la estrategia paralela mediante la implementación de tecnologías como MPI y/o CUDA.
- Generar instancias más grandes.
- Implementar una estrategia evolutiva para evaluar el conjunto de soluciones obtenidas por el HPDL.
- Hacer uso de instancias con un número de actividades/tareas mayor a diez y/o instancias con número de variables mayor a 2000.

Bibliografía

José Aguilar, Ernst Leiss. Introducción a la computación paralela. *Editorial Venezolana, Universidad de Los Andes, Mérida*, 2004.

Juan Carlos Alcaraz Rodriguez, Jordi y Moure López. Paralelización del problema de satisfacción de restricciones utilizando arco consistencia. 2015.

Cerisola Andrés, Ramos y Santiago. Optimización estocástica. *Universidad Pontificia Comillas. Madrid-Espana*, 2007.

Daniel Baena Mirabete. Aplicación del método de benders para ajuste controlado de tablas (cta). Master's thesis, Universitat Politècnica de Catalunya, 2008.

Marlo Carranza Purca. Media proximal y regularización. 2015.

René Castellanos, Jorge A y Dorta. Implementación de la interfaz para programación paralela de modula-3 bajo el sistema operativo windows. *Revista INGENIERÍA UC*, 12(1), 2005.

Chris N y Woeginger Gerhard J Chen, Bo y Potts. A review of machine scheduling: Complexity, algorithms and approximability. In *Handbook of combinatorial optimization*, pages 1493–1641. Springer, 1998.

Zhendong y Li Simon Chen, Li y Ding. A formal two-phase method for decomposition of complex design problems. *Journal of Mechanical Design*, 127(2):184–195, 2005.

IBM ILOG Cplex. 12.2 user's manual. *Book 12.2 User's Manual, Series 12.2 User's Manual*, 2010.

Rafael Martí Cunquero. Algoritmos heurísticos en optimización combinatoria. *Universidad de Valencia, Facultad de Ciencias Matemáticas*, 2003.

James Cuthill, Elizabeth y McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172. ACM, 1969.

Bernabé Dorronsoro Díaz. *Diseño e implementación de algoritmos genéticos celulares para problemas complejos*. PhD thesis, PhD thesis, Universidad de Málaga, 2007.

Marshall L Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15(2): 10–21, 1985.

David S Garey, MR y Johnson. Scheduling tasks with nonuniform deadlines on two processors. *Journal of the ACM (JACM)*, 23(3):461–467, 1976a.

David S y Sethi Ravi Garey, Michael R y Johnson. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976b.

Bahman Ghahremani, Pezhman y Naderi. Solution algorithms for the project selection and scheduling problem with resource constraints and time dependent returns. *International Journal of Industrial and Systems Engineering*, 19(3):348–363, 2015.

Siwhan Guignard, Monique y Kim. Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Mathematical programming*, 39(2):215–228, 1987.

Stefan y Reiter Peter y Stummer Christian y Denk Michaela Gutjahr, Walter J y Katzensteiner. Competence-driven project portfolio selection, scheduling and staff assignment. *Central European Journal of Operations Research*, 16(3):281–306, 2008.

Gang y Zhang Mengjie Karunakaran, Deepak y Chen. Parallel multi-objective job shop scheduling using genetic programming. In *Australasian Conference on Artificial Life and Computational Intelligence*, pages 234–245. Springer, 2016.

DN Kleinmuntz. Portfolio decision analysis: Improved methods for resource allocation, chapter foreword, pages v–vii, 2011.

Bogumiła Krzeszowska. Project portfolio scheduling as a multiple-criteria decision making problem. *Studia Ekonomiczne*, (137):69–82, 2013a.

Bogumiła Krzeszowska. Three step procedure for a multiple criteria problem of project portfolio scheduling. *Operations Research and Decisions*, 23, 2013b.

Shouyang y Yan Hong Lin, Dan y Wang. A multiobjective genetic algorithm for portfolio selection problem. 2001.

Gabriela F Minetti. Problema de ensamblado de fragmentos de adn resuelto mediante metaheurísticas y paralelismo. In *XIV Workshop de Investigadores en Ciencias de la Computación*, 2012.

Bahman Naderi. The project portfolio selection and scheduling problem: mathematical model and algorithms. *Journal of Optimization in Industrial Engineering*, 6(13):65–72, 2013.

Francisco y Alba Enrique y Dorronsoro Bernabé y Durillo Juan J Nebro, Antonio J y Luna. Un algoritmo multiobjetivo basado en búsqueda dispersa. 2007.

Sergio Nesmachnow. Evolución en el diseño y clasificación de algoritmos genéticos paralelos. In *Actas de la XXVIII Conferencia Latinoamericana de Informática*, pages 1933–1944, 2002.

Sergio Nesmachnow. Parallel multiobjective evolutionary algorithms for batch scheduling in heterogeneous computing and grid systems. *Computational Optimization y Applications*, 55(2): 515–544, 2013.

Miguel Ángel Olabe y Basogain Juan Carlos Olabe Olabe, Xabier Basogain y Basogain. Pensamiento computacional a través de la programación: Paradigma de aprendizaje. *Revista de Educación a Distancia*, (46), 2015.

Alberto Martínez Pérez. Paralelismo de tareas aplicado a algoritmos matriciales en memoria compartida. 2014.

Victoria María y De Giusti Armando Eduardo Pousa, Adrián y Sanz. Estructurando código paralelo para clusters heterogéneos de cpus/gpus. In *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016).*, 2016.

Pedro y Ferrer José María y Barquín Julián y Linares Pedro Ramos, Andrés y Sánchez. Modelos matemáticos de optimización. *Publicación Técnica*, 1, 2010.

Santiago Ramos, Andrés y Cerisola. Optimización estocástica. *Pontificia Universidad COMILLAS de Madrid*, 2005.

Antonio Carrillo y García Guillermo Hernández Revilla, Ismael Herrera y Ledesma. Aplicaciones del cómputo en paralelo a sistemas continuos.

KP Rogers, Ralph V y White. Algebraic, mathematical programming, and network models of the deterministic job-shop scheduling problem. *IEEE transactions on systems, man, and cybernetics*, 21(3):693–697, 1991.

SJC Salazar-Hornig, E y Medina. Minimizatión del makespan en máquinas paralelas idénticas con tiempos de preparación dependientes de la secuencia utilizando un algoritmo genético. *Ingeniería, investigación y tecnología*, 14(1):43–51, 2013.

Jeffrey y Morton Alec Salo, Ahti y Keisler. An invitation to portfolio decision analysis. In *Portfolio decision analysis*, pages 3–27. Springer, 2011.

Daniel Raul y Villagra Silvia Myriam Soria, Christian Luis y Pandolfi. Algoritmos genéticos celulares con operadores de recombinación aplicados a problemas de optimización discretos. *Informes Científicos-Técnicos UNPA*, 6(3):1–21, 2014.

S y Zárate F y Botello S y Moreles M y Pérez J y Rodríguez M y Domínguez F Súarez, MC y Gallegos. Aplicación del cómputo paralelo a la modelación de sistemas continuos en ciencias e ingeniería. 2007.

Reha y Gaytán Juan Summerville, Natalia y Uzsoy. A random keys genetic algorithm for a bicriterion project selection and scheduling problem. *International Journal of Planning and Scheduling*, 2(2):110–133, 2015.

El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

Rafael Terrazas Pastor. Aplicación de la programación matemática a la localización de proyectos. *Revista Perspectivas*, (29):69–92, 2012.

B Tofighian, Ali Asghar y Naderi. Modeling and solving the project selection and scheduling. *Computers & Industrial Engineering*, 83:30–38, 2015.

Bárbara Venegas y Gómez Jaime Bustos Villagra, Armin Lüer y Quintrileo. Estrategias de paralelización de metaheurísticas aplicadas a problemas de localización de instalaciones. *Revista Ingeniería Industrial*, 8(2), 2009.

Anthony Wren. Scheduling, timetabling and rostering—a special relationship? In *International Conference on the Practice and Theory of Automated Timetabling*, pages 46–75. Springer, 1995.