Centro de Investigación en Matemáticas, A.C.

Paisaje de búsqueda en el problema de planificación de producción tipo taller

Tesis que presenta

Juan Germán Caltzontzin Rabell

para obtener el Grado de

Maestro en Ciencias con Especialidad en Computación y Matemáticas Industriales

Director de Tesis

Dr. Carlos Segura González

Guanajuato, Gto. Julio de 2021

Resumen

En este trabajo se analiza el efecto de distintas metodologías de modificación del paisaje de búsqueda del problema de planificación de producción tipo taller (JSP por sus siglas en inglés). Existen una gran cantidad de tipos de problemas de planificación, el JSP es un tipo específico de problema que ha atraido considerable atención por su dificultad. En general estos problemas consisten en encontrar una planificación que minimice el tiempo requerido para completar ciertos procesos.

Aunque existen algoritmos exactos para encontrar la solución óptima JSP[4] el tiempo que requieren problemas grandes es demasiado largo como para que sean prácticos o inclusive factibles. La complejidad de este problema lo hizo un candidato ideal para utlizar métodos aproximados para conseguir una solución aceptable en un tiempo razonable. Actualmente los mejores resultados para las instancias de prueba más difíciles se han encontrado mediante metaheurísticas, en especfico la conocida como búsqueda tabú.

Las metaheurísticas son convenientes por su relativa simplicidad algorítmica frente a otros métodos sin embargo, el tiempo requerido para obtener los resultados del estado del arte ha comenzado a crecer también, por lo que es importante dar un paso atrás y replantear un poco la forma de aplicar estos métodos para evitar complicarlos de más.

En el campo de las metahurísticas hay varios conceptos fundamentales que tienen un gran impacto en su desempeño, en este trabajo se analizan tres de ellos que juntos componen lo que se conoce como paisaje de búsqueda el cual está estrechamente relacionado con la dificultad con el desempeño de las metaheurísticas de trayectoria.

Los conceptos analizados en este trabajo son : representación de las soluciones, definición de vecindad de una solución, función de aptitud o fitness. Para evaluar el impacto de cada uno de ellos se utliza una de las metaheurísticas de trayectoria más sencillas; la búsqueda local iterada.

Se tomaron las instancias dmu que son uno de los conjuntos de prueba más utilizados actualmente y que aun no ha sido resuelto por completo para hacer experimentos computacionales y se encontró que de las tres áreas en las que se plantearon modificaciones, la más importante fue la de la representación de soluciones.

Abstract

En este trabajo se analiza el efecto de distintas metodologías de modificación del paisaje de búsqueda del problema de planificación de producción tipo taller (JSSP por sus siglas en inglés) (En inglés)

Índice general

Capítulo 1

Introducción

Sumario

1.1. Antecedentes y Motivación

Los problemas de planificación surgen de manera natural de un sistema de producción o procesamiento en los que la tarea que se quiere completar consiste a su vez de varias subtareas que pueden o deben repartirse entre distintas máquinas o unidades de procesamiento. A grandes rasgos lo que se busca es asignar a cada subtarea una máquina que debe procesarla y un tiempo de inicio y fin. Estos problemas pueden surgir de todo tipo de contextos, desde la producción de algo como una bicicleta hasta la forma en que los sistemas computacionales procesan información o cómo se asignan las clases en una escuela. En muchos de estos contextos no solo se requiere hallar una planificación sino que a demás se quiere encontrar una que sea óptima en algún sentido, habitualmente se quiere la que haga que se complete el trabajo lo más rápido posible aunque puede haber otros criterios.
Desde los década de los 50 se comezaron a formular algoritmos de planificación[9] y el interés en ellos ha crecido en las décadas siguientes hasta la fecha.

En este trabajo se trata un tipo especial de problema de plaficación conocido como el Problema de Planificación de Producción tipo Taller o JSP por sus siglas en inglés. Este problema consiste en hallar una planificación que tome el minimo tiempo posible para realizar un conjunto de trabajos, cada una de ellos consistiendo de una secuencia de operaciones a procesarse en un orden determinado, en un conjunto de máquinas. Este es un problema de optimización combinatoria en

la que se requiere encontrar una secuencia de procesamiento para las operaciones en cada una de las máquinas disponibles y es NP duro cuando el número de máquinas es mayor a dos.

Anteriormente se han propuesto métodos exactos para resolver este problema [4] pero la cantidad de recursos computacionales que requieren conforme el tamaño del problema (número de máquinas y trabajos) aumenta los hace impráticos excepto para instancias pequeñas.

Dado el interés que se tiene en este problema y en que su tamaño puede ser relativamente grande, se comenzaron a proponer otros métodos aproximados para encontrar soluciones buenas en tiempos aceptables. Estós métodos pueden agruparse de la siguiente manera[14]:

- **Métodos Constructivos** Estos métodos construyen una planificación mediante el uso de una relga simple por lo que son muy rápidos y conceptualmente sencillos. En general pueden clasificarse en tres tipos: los que usan reglas de prioridad, los que usan heurísticas de cuello de botella y los que usan algún método de inserción. El primero de estos consiste en establecer una forma de elegir la operación a planificar de varias disponibles. Esto puede hacerse por ejemplo eligiendo la que tome más tiempo o la que pueda procesarse antes. El segundo de estos consiste en replantear el problema como una serie de subproblemas más sencillos que puedan resolverse iterativamente hasta que se tenga una solución completa. El tercer tipo construye una solución partiendo del ordenamiento de solo un subconjunto de operaciones y progresivamente agregando más a partir de las que ya se tienen.

- **Métodos de inteligencia artificial** En estos métodos utlizan redes neuronales para encontrar la planificación. Existen muchos tipos de redes que se han diseñado para atacar este problema aunque en general suelen combinarse con otros métodos para obtener resultados competitivos.

- **Métodos de búsqueda local** En estos métodos se establece una forma de crear soluciones nuevas a partir de una solución dada para reemplazarla una nueva y repetir el proceso hasta que se cumpla algún criterio de paro. Para crear nuevas soluciones se hace un cambio pequeño como, por ejemplo, intercambiar el orden de dos operaciones de modo que sea posible evaluar todas las posibles soluciones generadas al aplicar esta modificación. En general estos métodos se distinguen entre sí por la manera en que se escoge la solución con la cual reemplazar la solución inicial y el criterio de paro.

- **Metaheurísticas** Estos métodos combinan elementos de los previamente mencionados para plantear estrategias que obtengan aun mejores resultados. Es muy común que se planteen metaheurísticas inspiradas en la naturaleza como los algoritmos genéticos basados en la evolución o algoritmos basados en el comportamiento de seres vivos. Al ser de alto nivel pueden adaptarse a una gran cantidad de problemas y obtener buenos resultados.

Actualmente los algoritmos más exitosos para resolver el JSP son algoritmos meméticos que combinan un método de búsqueda local con una metaheurística poblacional que mantiene varias

soluciones. El método de búsqueda local más usado se conoce como búsqueda tabú que mediante el uso de memoria logra escapar de óptimos locales.

Aunque se han obtenido buenos resultados con estos algoritmos se requieren de 24 a 48 horas paralelas de ejecución. La literatura reciente se ha centrado en hacer más eficiente la búsqueda tabú para reducir los tiempos de ejececución, dejando de lado los otros elementos que forman parte de las metaheutísticas. Dentro de estos elementos hay tres qué son comunes a todas las metaheurísticas de búsqueda local. El primero y más básico es cómo representar computacionalmente una solución, actualmente lo más común es que una solución se represente como un grafo dirigido cuyos nodos son las operaciones a procesar y las aristas indican el ordenamiento de éstas. El segundo es la forma de generar una solucion a partir de otra, esto se conoce como una estructura de vecindad y la más exitosa es conocida como N/ y fue propuesta en 2006 [13]. El tercer elemento se conoce como función de fitness o aptitud y nos permite comparar soluciones y decidir si una es mejor que otra.

Los tres elementos anteriormente mencionados conforman lo que se conoce como paisaje de búsqueda.

1.2. Objetivo

El objetivo de este trabajo es plantear modificaciones al paisaje de búsqueda del JSP que permitan el uso de mecanismos más simples y rápidos para encontrar soluciones comparables al estado del arte.

Objetivos específicos

- Proponer una función de fitness que tome en cuenta más que solo el makespan de una planificación.
- Plantear una representación con la que se puedan obtener mejores resultados.
- Plantear una nueva estructura de vecindad.

1.3. Propuestas

Hipótesis

Es posible conseguir resultados comparables al estado del arte con algoritmos sencillos y rápidos si modificamos el paisaje de búsqueda de manera adecuada.
Como se explica mas adelante, el paisaje de búsqueda es la combinación de tres elementos: función de fitness, espacio de búsqueda y estructura de vecindad. Estos tres elementos tienen un efecto muy importante en el éxito que puede llegar a tener una metaheurística Para poner a prueba

la hipótesis se presentan las siguientes propuestas que serán exploradas mediante experimentos computacionales:

1. Utilizar la búsqueda local iterada (ILS por sus siglas en inglés) por ser un algoritmo simple y rápido.
2. Agregar nuevos movimientos a la vecindad N7.
3. Crear una función de fitness que no solo tome en cuenta el makespan sino también otras características de la solución.
4. Utilizar una nueva representación junto con un esquema de decodificación para limitar el espacio de búsqueda.
5. Construir una nueva estructura de vecindad a partir de la nueva representación.

Capítulo 2

Marco teórico

Sumario

2.1. Optimización

La optimización es una herramienta que nos ayuda a encontrar *la mejor* entre diferentes opciones elegibles. En nuestra vida diaria a menudo nos encontramos en este tipo de situaciones, por ejemplo al elegir entre diferentes rutas para llegar a algún lugar.

Formalmente, un problema de optimización consiste en hallar el mínimo o máximo de una función $f : X \to \mathbb{R}x$ (llamada función objetivo) en un conjunto de soluciones X. De modo que el problema consiste en hallar:

$$\min_{x \in X} f(x) \tag{2.1}$$

Es importante mencionar que cualquier problema de maximización puede transformarse en un problema equivalente de minimiación con el reemplazo $f(x) \leftarrow -f(x)$, por lo que se considera solo el caso de minimización sin pérdida de generalidad.

Hallar el mínimo de la función sobre todo el conjunto X recibe el nombre de optimización global, esto puede llegar a ser muy costoso o incluso imposible de obtener por lo que es común optar por obtener un mínimo local, es decir una solución que sea mejor que cualquiera de las soluciones que tiene a la misma distancia de acuerdo con alguna medida.
En la definición anterior no se requiere que el conjunto de soluciones tenga alguna propiedad o alguna estructura adicional. Por ejemplo el conjunto de soluciones puede ser un conjunto numerable (i.e. los enteros) o no numerable (i.e. los números reales). Estos dos casos dividen a la optimización en dos ramas: optimización continua y optimización discreta. En general los problemas de optimización suelen ser más fáciles de abordar[10] porque en muchas ocasiones es posible obtener información del valor de la función objetivo de puntos cercanos a cierto punto conocido mientras que en los problemas discretos esto rara vez puede hacerse.

Dentro de los problemas de optimización discreta se distinguen los problemas de optimización combinatoria. Formalmente un problema de optimización discreta consta de los siguientes elementos[3]:

- Un conjunto de variables $Z = z_1, z_2, ..., z_n$
- Dominio para cada variable $D_1, D_2, ..., D_n$
- Restricciones entre variables

En muchos problemas de optimización combinatoria el conjunto de soluciones no tiene alguna estructura adicional que ayude a buscar el mínimo de la función objetivo; es raro que exista un ordenamiento de las soluciones o una medida de distancia entre ellas que brinde propiedades a la función objetivo tales como continuidad o suavidad las cuales facilitarían la búsqueda del mínimo. Dicho de otro modo, no tenemos una forma eficiente de explorar el espacio de búsqueda. Ante estas limitaciones surgieron técnicas conocidas como metaheurísticas que buscan facilitar la resolución de estos problemas.

2.2. Metaheurísticas

Es muy común que en nuestra cotidianidad nos enfrentemos a problemas tan difíciles o para los que tengamos tan poco tiempo de decisión que no podamos hacer un análisis riguroso, en estos casos es muy común que utilicemos algún método (posiblemente basado en la experiencia) que nos permita hallar una solución aceptable, por ejemplo, es común que reemplacemos el problema por

uno más simple que sí podemos responder y cuya respuesta está relacionada con nuestro problema original.[1]

En el contexto de la optimización una metaheurística es una metodología de alto nivel que combina diferentes heurísticas y puede aplicarse para resolver de manera aproximada una gran cantidad de problemas. En la práctica existen numerosas metaheuristicas que pueden ser muy diferentes entre sí por lo que no hay un sistema de clasificación universalmente aceptado aunque se han propuesto diferentes criterios de clasificación [12] así como características como:

- De trayectoria vs discontinua. Una metaheurística de trayectoria consiste en, dada una solución inicial, mejorarla de manera iterativa mediante algún operador que «mueve» a la solución a través del espacio de búsqueda
- basadas en población vs basadas en una sola solución. En las metaheurísticas basadas en población se mantiene un conjunto de soluciones candidatas.
- basadas en búsqueda local vs constructivas. Como se explicará más adelante, en la búsqueda local, el proceso de mejora implica la evaluación de soluciones muy parecidas a una solución inicial dada mientras que en las constructivas se crean nuevas soluciones de acuerdo a una heurística o algoritmo preestablecido.
- Con uso de memoria vs sin uso de memoria. El uso de memoria consiste en almacenar información que nos ayude a explorar el espacio de búsqueda, por ejemplo una lista de soluciones previamente visitadas.

Los primeros dos de estos criterios están muy relacionados porque casi todas las metaheurísticas discontinuas son poblacionales y muchas de trayectoria son basadas en una sola solución.

Si bien las metaheurísticas son muy diversas existen elementos comunes que tienen un papel determinante en el buen funcionamiento de las mismas. En espécifico para las metaheurísticas de trayectoria existen tres conceptos que determinan cómo se ve el espacio de soluciones: las soluciones en sí, la forma en que las soluciones están conectadas y cómo podemos comparlas entre sí. A continuación se describebe cada uno de ellos

2.2.1. Representación

Puede ser que el problema de optimización en el que estemos interesados surja directamente de las matemáticas aunque si estamos interesados en un problema de nuestro entorno físico es necesario que tengamos una forma de traducirlo a un lenguaje matemático, incluidas las soluciones al mismo. Debemos encontrar una forma de representar de manera útil las soluciones posibles. Por ejemplo si buscamos un ordenamiento óptimo para algún conjunto de n cosas podemos asignarle a cada elemento del conjunto un número entero del 1 al n, en este modelo el espacio de soluciones

[1] No podemos predecir con certeza si lloverá durante el día pero sí podemos responder si el cielo está plagado de nubes oscuras

está constituido por todas las permutaciones de los números del 1 al n. Hay varias maneras de representar permutaciones, podemos usar un arreglo de n entradas o bien una matriz de permutación por mencionar algunos. Algo sumamente importante es que estas dos formas de representar las soluciones son muy distintas y se tiene que trabajar con ellas de manera muy diferente. La primera de ellas puede llegar a representar *soluciones no factibles* por ejemplo que aparezca un número repetido, mientras que la segunda no.

Con el ejemplo anterior también podemos ver que cuando queramos establecer algunos operadores que, por ejemplo, perturben la solución tendremos que definirlos de maneras completamente distinta. Tambien es importante notar que es posible que las soluciones no factibles que podamos representar sean muchas más que las factibles.

Formalmente una represntación es un mapa que asocia elementos entre el conjunto de soluciones y el conjunto de las representaciones. Pueden distinguirse tres tipos de representaciones[5] de acuerdo a como asocian los elementos de estos dos conjuntos.

- 1 a 1
- 1 a n
- n a 1

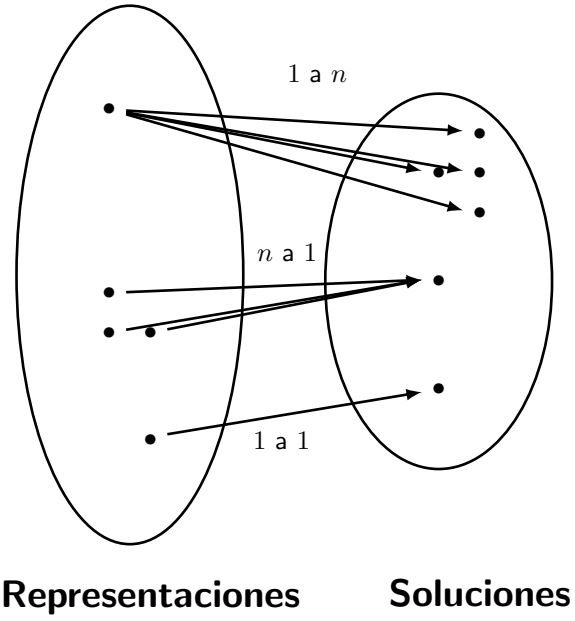


Figura 2.1: Tipos de representaciones

Lo más conveniente suele ser un mapeo 1 a 1 y lo menos deseable es tener un mapeo n a 1.

2.2.2. Vecindad

La definicion de vecindad es crucial para las metaheurísticas de trayectoria y las basadas en una sola solución. Formalmente, una vecindad es un mapeo $N : X \leftarrow 2^X$ que le asigna a cada solución $x \in X$ un subconjunto de soluciones en X. Intuitivamente podemos pensar que es una forma de definir a las soluciones que «rodean» a otra. Se dice que la solución y es un vecino de x si $y \in N(x)$.

A partir de la definición de vecindad podemos también definir un operador de movimiento cuyo efecto al aplicarlo a una solución sea transformarla en una que pertenezca a su vecindad, i.e. este operador selecciona a un vecino de la solución inicial.

2.2.3. Función de aptitud o fitness

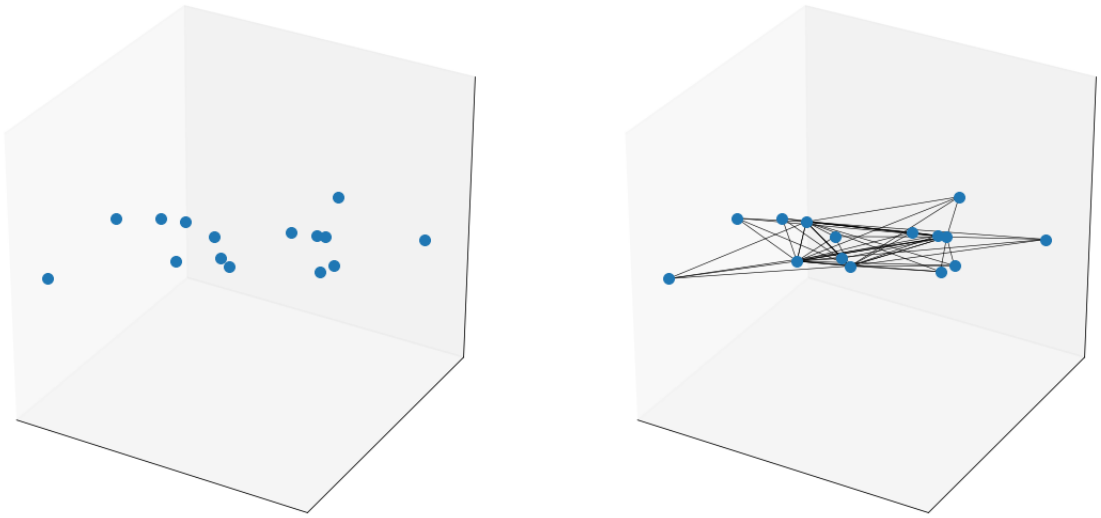
Aunque para un problema de optimización ya se tiene definida una función objetivo que se quiere minimizar, no siempre tenderemos el mejor desempeño de las metaheurísticas con solo esta función por lo que resulta benéfico plantear una nueva función a minimizar con la que tengamos mejor desempeño. Por ejemplo puede suceder que aunque dos soluciones tengan asociado el mismo valor de la función objetivo una de ellas posee características que la hacen un mejor punto de partida para alguna metaheuristica.

Esta función debe asociar a cada solución un elemento de un conjunto donde esté definido un ordenamiento total. En esencia esta función define un operador de comparación entre soluciones de modo que podemos elegir la mejor de dos soluciones.

2.2.4. Paisaje de búsqueda

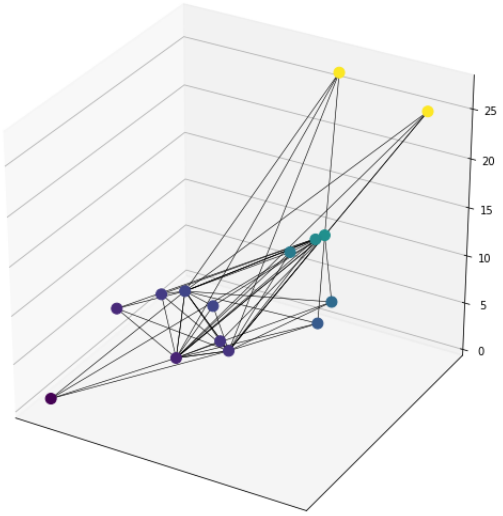
Una vez que tenemos el espacio de búsqueda y operadores de cambio para generar nuevas soluciones a partir de otras, se define el espacio de búsqueda como un grafo dirigido en el que los nodos son las soluciones al problema y una solución x está conectada a otra y si podemos generar a y aplicando los operadores de cambio a x.

Podemos asociar a cada solución en el espacio un valor de aptitud o fitness que mide la calidad de dicha solución. La adición de esta función de aptitud al espacio de búsqueda genera al paisaje de búsqueda.



(a) Soluciones

(b) Relaciones inducidas por los operadores de cambio



(c) Adición de la función de fitness

Figura 2.2: Creación del paisaje de búsqueda

El paisaje de búsqueda es el «terreno» a explorar y puede cambiar si cambiamos cualquiera de sus componentes, podría ser que alguna representación nos centre en un subconjunto de soluciones convenientes o que alguna estructura de vecindad se proponga de modo que las mejores soluciones nunca estan muy lejos del resto, o que la función de fitness nos ayude a atravesar cúmulos de soluciones que serían iguales sin ella. Las metaheurísticas sirven como una estrategía para explorar el paisaje de búsqueda. Una de las más sencillas e intuitivas es conocida como escalada estocástica y simplemente consiste en reemplazar la solución actual por algún vecino mejor escogido al azar

hasta que la solución en la que estemos sea mejor que todos sus vecinos, es decir, un ótimo local. Esta es una metaheurística de trayectoria y traza un camino entre las soluciones inicial y final.

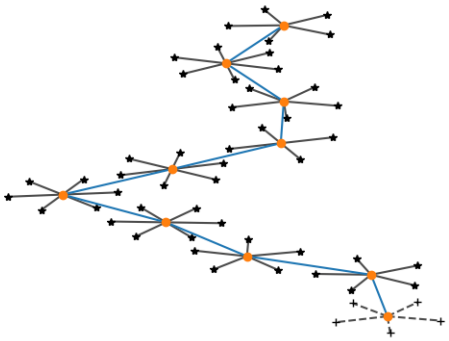


Figura 2.3: Ilustración de una escalada estocástica.
En cada paso se selecciona un vecino mejor que la solución actual hasta que la solución actual sea mejor que todos sus vecinos

2.3. Problema del planificación de producción tipo taller (JSSP)

Una instancia del JSSP consiste en n trabajos diferentes constituidos cada uno por m operaciones que deben procesarse por un tiempo determinado en m máquinas en una secuencia predeterminada.
El objetivo es hallar la planificación que minimiza el tiempo que toma terminar todos los trabajos dado que cada máquina puede procesar solo un trabajo a la vez.

Una planificación consiste en asignar tiempos de inicio y fin a cada operación respetando el orden requerido para cada trabajo. El tiempo que toma terminar todos los trabajos se conoce como makespan y la secuencia de trabajos que toma el mayor tiempo en completarse se conoce como ruta crítica. La ruta crítica puede verse como una serie de bloques críticos que consisten en las secuencias de operaciones de la ruta crítica que se ejecutan de forma adyacente en la misma máquina. Una planificación puede tener una o varias rutas críticas.

En general se considera que el tiempo requerido para procesar cada operación puede expresarse como un entero.

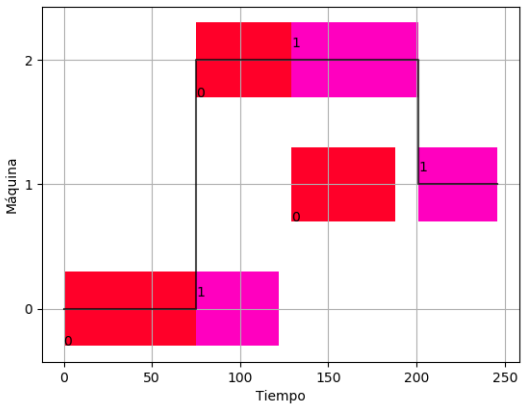
Ejemplo

Se muestra un ejemplo de una instancia con 3 máquinas y 2 trabajos.

Tabla 2.1: Instancia simple con 3 maquinas y 2 trabajos

Trabajo	Secuencia de procesamiento (máquina, tiempo)		
0	0, 75	2, 54	1, 59
1	0, 47	2, 72	1, 45

La siguiente es una posible planificación para la instancia de ejemplo, visualizada mediante un diagrama de gantt. En negro se marca los trabajos que conforman la ruta crítica.



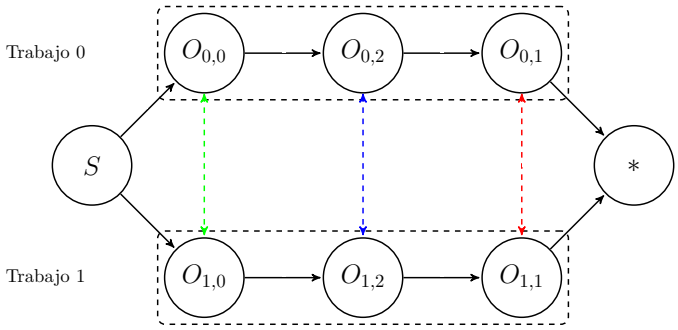
2.4. Representación de planificaciones

Existen varias formas de representar las planificaciones, en este trabajo se utilizaron dos: el grafo disyuntivo y las reglas de prioridad.

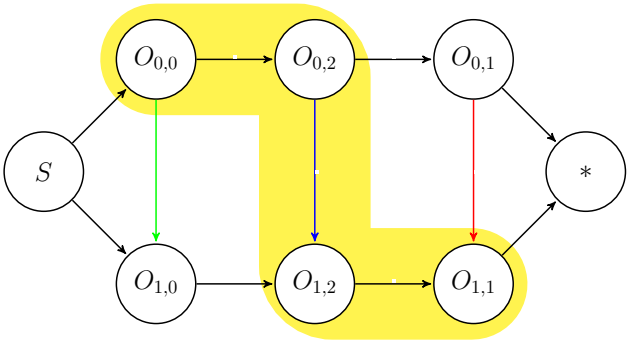
Modelo de grafo disyuntivo

En este modelo las planificaciones se representan con un grafo dirigido $G = (V, A, E)$ en el que V es un conjunto de nodos que representa las operaciones, las aristas A representan la secuencia que deben seguir las operaciones dentro de un mismo trabajo y E es otro conjunto de aristas que indica el orden de procesamiento en cada una de las máquinas. Es importante mencionar que con este modelo podemos representar planificaciones no factibles, esto se da cuando el grafo G contiene un ciclo.

Formalmente en una instancia del JSP se representa cada operación como un nodo, se agregan dos nodos de control que sirven como el nodo inicial (del que dependen todos los trabajos) y final (que depdende de todos los trabajos), las restricciones de precedencia dentro de cada trabajo se representan como aristas dirigidas fijas llamadas aristas conjuntivas y las operaciones que deben procesarse en una misma máquina se unen mediante aristas llamadas disyuntivas, una solución



(a) Representación de una instancia



(b) Planificación obtenida al fijar las aristas disyuntivas.
La ruta critica se resalta en amarillo

Figura 2.4: Modelo de grafo disyuntivo para la instancia de ejemplo 2.1

factible o planificación se obtiene al elegir la dirección para cada arista disyuntiva de modo que no se generen ciclos.

Reglas de prioridad

En esta representación una planificación se construye al aplicar un proceso de simulación en el que para cada maquina se construye una cola con las operaciones cuyas dependencias ya han sido procesadas. Inicialmente se tienen en las colas solo las operaciones iniciales de cada trabajo. Una vez que se tiene esto se utiliza una regla de prioridad para elegir qué operación debe planificarse en qué máquina. Se actualizan las colas para las máquinas que lo requieran y se continua con este proceso hasta completar la planificación (vaciar las colas)

Tipos de planificaciones

Dentro de el conjunto de planificaciones factibles se pueden distinguir dos subconjuntos de interés para el presente trabajo: el conjunto de las planificaciones óptimas que está conformado por las planificaciones con el menor makespan posible y el conjunto de las planificaciones activas. Estas últimas se definen como las planificaciones en las que no es posible disminuir el tiempo

de inicio de ninguna operación sin aumentar el tiempo de inicio de otra. Es conocido que las planificaciones óptimas representan un subconjunto de las activas.

2.5. Conjuntos de prueba

Existesn

2.6. Metaheurísticas aplicadas al JSP

Las metaheurísticas han conseguido hallar buenas soluciones para los conjuntos de prueba U

Vecindades previamente propuestas

Se han propuesto varias estructuras de vecindad al JSP, a continuación se describen las más importantes a la fecha:

- N1 [2] Consiste en considerar todas las soluciones que se crean al intercambiar cualquier par de operaciones adyacentes que pertenecen a un bloque crítico. Esta vecindad es muy grande y considera muchos cambios que no mejoran el makespan.

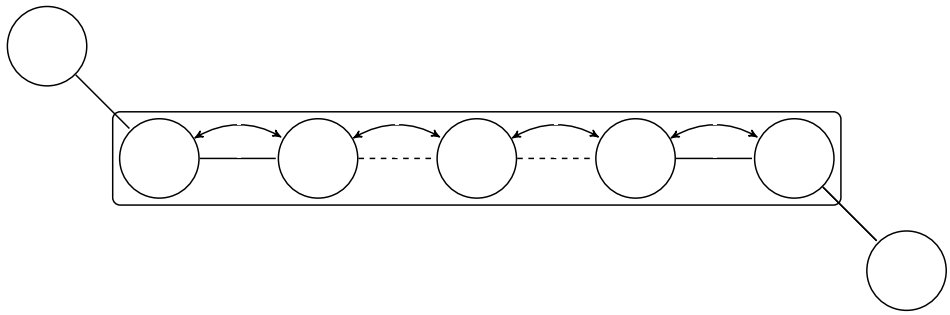


Figura 2.5: Movimientos de la vecindad N1

- N4 [6] Esta vecindad se propuso como un refinamiento y extensión de la vecindad N1 y toma como base el concepto de bloque crítico. Consiste en llevar operaciones internas del bloque crítico al inicio o final.

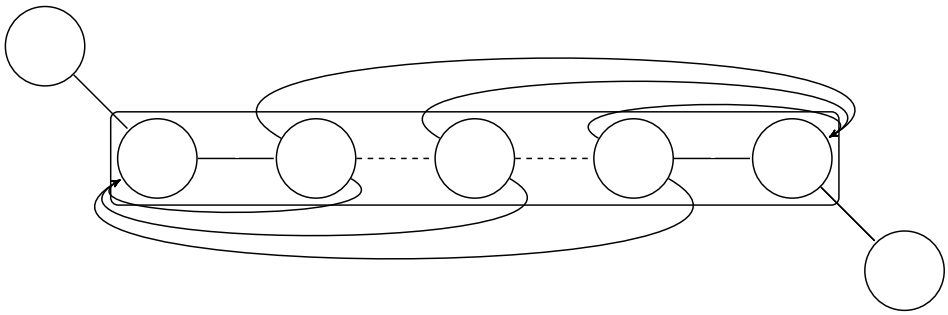


Figura 2.6: Movimientos de la vecindad N4

- N5 [7] Consiste en intercambiar solo las operaciones adyacentes a la final o inicial de un bloque crítico.

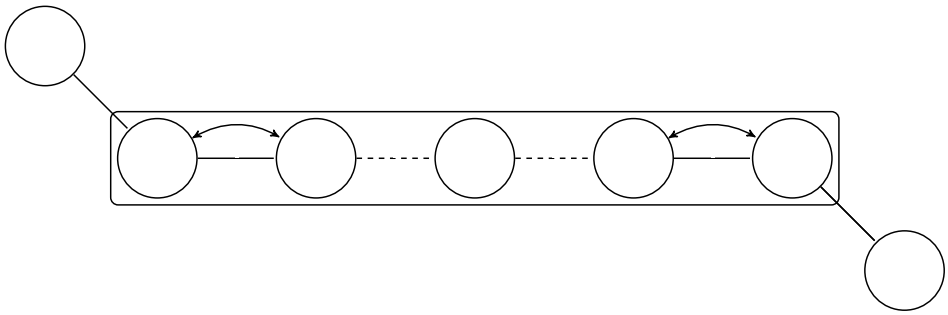


Figura 2.7: Movimientos de la vecindad N5

- N6 [1] Los autores utilizan varios teoremas para identificar pares (u, v) de operaciones dentro de un bloque crítico que puedan llevar a mejorar la solución y a su vez identificar si se tiene que mover a u justo después de v(forward) o bien a v justo antes de u (backward).

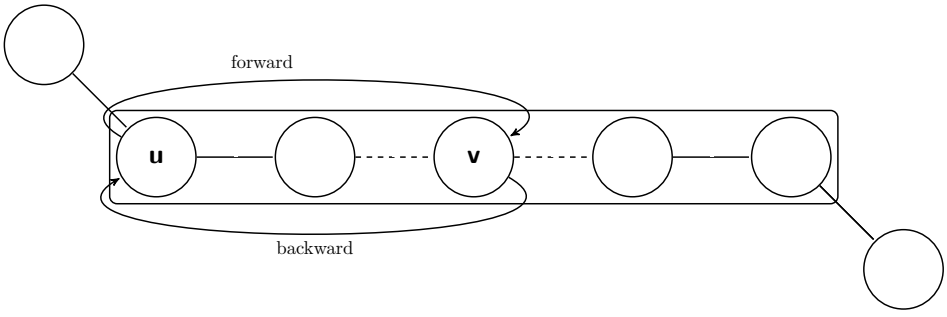


Figura 2.8: Los dos tipos de movimientos para un par (u, v)

- N7 [13] Esta vecindad se plantea como una extensión de la N6 en la cual se toma la idea de los movimientos entre pares de operaciones de un bloque crítico. Los autores toman en cuenta

todos los cambios posibles entre el inicio o fin del bloque crítico con todas las operaciones internas.

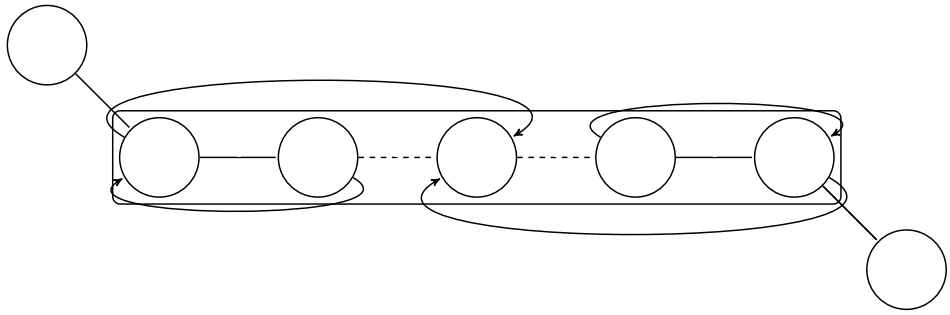


Figura 2.9: Movimientos de la vecindad N7

Capítulo 3

Propuestas

3.1. Extensión a vecindad N7

Como punto de partida se planteó agregar movimientos a la vecindad N7 con la cual se han obtenido los resultados del estado del arte. Los movimientos que plantea esta vecindad solo tienen que ver con pares de operaciones en la ruta crítica por lo que una extensión sencilla consiste en considerar movimientos de operaciones que pueden no pertenecer a la ruta crítica. Es importante resaltar que si no se planteara también una función de fitness que no tome solo en cuenta el makespan estos movimientos nunca llevarían a una mejora [2]. Los movimientos planteados se basan en observar que en alguna solución encontrada por una búsqueda local para cada maquina pueden existir periodos de tiempo en la que está inactiva pero existe una operación que podría comenzar a procesarse en este periodo y que se procesa después.

3.2. Vecindad basada en soluciones activas

Este problema

3.3. Llaves aleatorias

La representación propuesta se basa en asignar a cada operación una número real entre 0 y 1 el cual sirve para definir un orden entre operaciones en una misma máquina mediante un proceso de decodificación un planteamiento similar puede encontrarse en [11].

Para decodificar la solución a partir de las llaves para cada operación se utiliza el algoritmo de Giffler & Thompson [8] con el cual se generan soluciones activas.

Capítulo 4

Validación experimental

Capítulo 5

Conclusiones y Trabajos a Futuro

Conclusiones

Bibliografía

[1] Balas, E., and Vazacopoulos, A. Guided local search with shifting bottleneck for job shop scheduling. *Management Science 44*, 2 (1998), 262–275.

[2] Błażewicz, J., Domschke, W., and Pesch, E. The job shop scheduling problem: Conventional and new solution techniques. *European journal of operational research 93*, 1 (1996), 1–33.

[3] Blum, C., and Roli, A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys 35*, 3 (2003), 268–308.

[4] Brucker, P., Jurisch, B., and Sievers, B. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics 49*, 1-3 (1994), 107–127.

[5] Cheng, R., Gen, M., and Tsujimura, Y. A tutorial survey of job-shop scheduling problems using genetic algorithms - I. Representation. *Computers and Industrial Engineering 30*, 4 (1996), 983–997.

[6] Dell'Amico, M., and Trubian, M. Applying tabu search to the job-shop scheduling problem. *Annals of Operations research 41*, 3 (1993), 231–252.

[7] Eugeniusz Nowicki, C. S., and To. A Fast Taboo Search Algorithm for the Job Shop Problem. *Manage. Sci. 42*, 6 (2003), 797–813.

[8] Giffler, B., and Thompson, G. L. Algorithms for Solving Production-Scheduling Problems, 1960.

[9] Johnson, S. M. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly 1*, 1 (1954), 61–68.

[10] Nocedal, J., and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

[11] Norman, B. A., and Bean, J. C. A random keys genetic algorithm for job shop scheduling. Tech. rep., 1996.

[12] Stegherr, H., Heider, M., and Hähner, J. Classifying Metaheuristics: Towards a unified multi-level classification system. *Natural Computing 0* (2020).

[13] Zhang, C. Y., Li, P. G., Guan, Z. L., and Rao, Y. Q. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers and Operations Research 34*, 11 (2007), 3229–3242.

[14] Zhang, J., Ding, G., Zou, Y., Qin, S., and Fu, J. Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing 30*, 4 (2019), 1809–1830.