Trabajo Final de Ingeniería Industrial

Programación de la producción en un taller textil

Una aplicación del FJSSP

Natalia Soledad Herrada

Mateo Valla

Departamento de Ingeniería Industrial - Facultad de Ingeniería

Universidad Nacional de Mar del Plata

Mar del Plata, Junio de 2011

PROGRAMACIÓN DE LA PRODUCCIÓN EN UN TALLER TEXTIL

Una aplicación del FJSSP

AUTORES:

Natalia Soledad Herrada

Mateo Valla

Departamento de Ingeniería Industrial - Facultad de Ingeniería

Universidad Nacional de Mar del Plata

DIRECTOR:

Mg. Ing. Claudia Zárate

Departamento de Ingeniería Industrial - Facultad de Ingeniería

Universidad Nacional de Mar del Plata

EVALUADORES:

Ing. Alejandra Esteban

Mg. Ing. Claudia Zárate

Mg. Ing. Adolfo E. Onaine

Departamento de Ingeniería Industrial - Facultad de Ingeniería

Universidad Nacional de Mar del Plata

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

GLOSARIO

Cmáx: Mejor valor de makespan conocido.

FJSSP (Flexible Job Shop Scheduling Problem)**:** Problema de secuencuación de operaciones en ambientes de producción flexible.

JSSP (Job Shop Scheduling Problem)**:** Problema clásico de secuenciación de operaciones.

Makespan: tiempo total de producción en la secuenciación de operaciones.

Serie de Lawrence: conjunto de problemas de secuenciación clásicos, utilizados comúnmente para la evaluación de métodos de resolución de la programación de operaciones.

Vecino: condición asignada a una solución o conjunto de soluciones. Se define como el subconjunto de soluciones $S_i \subset S$ tal que todos sus elementos son cercanos a la solución *i* en algún sentido. Se asume que $j \in S_i \leftrightarrow i \in S_j$ (siendo i y j dos soluciones pertenecientes a S).

RESUMEN

La programación de las operaciones es una actividad de suma importancia en toda organización manufacturera. Comprende la asignación de recursos, la secuenciación y la programación detallada de cada período de producción. De ello depende que se puedan llevar a cabo los objetivos establecidos, cumpliendo con la planificación prevista de manera eficiente y en tiempo y forma. En el presente trabajo se desarrolló un programa informático destinado a colaborar con la resolución del problema de secuenciación que enfrenta una empresa textil de la zona. El objetivo fue brindarle a la misma una herramienta que sirviese de apoyo a la toma de decisiones en la fase de programación de las operaciones. Para ello se estudiaron los diferentes tipos de problemas de secuenciación de operaciones y sus métodos de resolución, determinándose el FJSSP como la clase de problemas más representativo del caso presentado. La resolución del mismo se efectuó mediante el algoritmo de recocido simulado. El programa obtenido se evaluó mediante una serie de problemas de prueba, los cuales permitieron verificar la eficacia y confiabilidad de los resultados devueltos por el mismo.

PALABRAS CLAVE: programación de operaciones, secuenciación, FJSSP, recocido simulado.

INTRODUCCIÓN

El presente trabajo se desarrolló a partir de la iniciativa, como futuros ingenieros industriales, de detectar y resolver un problema específico concerniente al ámbito industrial de la región, que permitiera poner en práctica los conocimientos adquiridos durante la carrera y a la vez, significara un aporte sustancial para la industria en cuestión. Se seleccionó una empresa del rubro textil por ser éste uno de los sectores más relevantes dentro de la industria marplatense, y presentar además un potencial importante de mejora en cuanto a sus procesos.

1.1 Descripción de la empresa

La empresa textil mencionada, de ahora en adelante denominada TEXTIL SA, se dedica al diseño y confección de indumentaria. La misma fabrica y comercializa sus productos en dos períodos semestrales correspondientes a las temporadas de verano e invierno, razón por la cual realiza su planificación de la producción dos veces al año, cada una de ellas previa al inicio de temporada.

Para cada temporada se diseñan una serie de artículos que constituyen la colección a comercializar en dicho período. En primer lugar se procede a diseñar las prendas, luego se realizan los prototipos correspondientes para determinar ajustes y mejoras del modelo, y finalmente, una vez aprobados los artículos que comprenderán la colección, se procede a la fabricación de los mismos. La cuantía de la producción se corresponde al pronóstico de la demanda proporcionado por el área de comercialización.

Los lotes de producción se determinan en función del pronóstico mencionado y la estrategia de abastecimiento utilizada. Comúnmente cada artículo constituye un lote, razón por la cual la cantidad de lotes de una temporada es equivalente a la cantidad de artículos a fabricar para esa colección.

Por otra parte, cada artículo, requiere de una serie de tareas necesarias para ser confeccionado, que no necesariamente son iguales para todas las prendas de la colección. Entre estas tareas se encuentran el corte, el estampado, el bordado, la

confección, el lavado, la terminación y el control de calidad. El orden y la cantidad de dichas tareas son determinadas para cada artículo de acuerdo a sus características de diseño.

Para llevar a cabo estas tareas, la empresa cuenta tanto con talleres propios como de terceros, cada uno de los cuales se especializa en alguna tarea en particular (confección, estampado, control de calidad, etc.). A su vez, no todos los talleres confeccionan todo tipo de prendas; por lo general se especializan en un tipo de prenda en particular o un conjunto de éstos (por ejemplo, talleres que sólo confeccionan camperas, o que realizan únicamente pantalones y buzos, etc.). Esto implica que dependiendo del artículo y la etapa del proceso productivo en que se encuentre la prenda, existirá sólo un conjunto del total de talleres disponibles que podrá llevar a cabo la actividad.

Por otra parte, la empresa cuenta con una fecha límite de entrega de la producción, la cual está determinada por el inicio de la temporada. Esta fecha resulta de vital importancia, ya que al tratarse de artículos de moda, pasada la misma la mercadería es rechazada por los clientes (locales de venta de indumentaria) por la imposibilidad de vender la misma fuera de época (Socio Gte. TEXTIL SA, 2010). Teniendo en cuenta que además los clientes no reciben mercadería hasta la fecha de inicio de temporada, la empresa adopta como estrategia tener el total de la producción, o gran parte de ella, lista para esa fecha. Esto genera la necesidad de establecer un plan de producción adecuado que permita llegar al inicio de temporada con el total de los artículos en condiciones de ser entregados.

1.2 Situación actual

Si bien la empresa reconoce la necesidad de llevar a cabo un plan de producción que le permita llegar a tiempo con la entrega de la colección, actualmente no cuenta con el mismo (Gte. de Producción TEXTIL SA., 2010). Existe una planificación de los tiempos previos a la fabricación de las prendas, que comprende toda la etapa de diseño y confección de prototipos, pero no incluye los tiempos posteriores correspondientes a la etapa de producción.

Dada esta imposibilidad de la empresa de establecer un cronograma de producción, llegada la fecha de entrega de la colección sucede que un porcentaje de esta no se ha terminado de confeccionar aún, con las consecuentes pérdidas que esto genera para la misma. Además, a lo largo del proceso, se pueden observar otros inconvenientes tales como retrasos en la confección de determinados artículos, superposición de lotes de producción en un mismo taller, falta de disponibilidad de los talleres, entre otros, que se derivan también de no contar con la previa planificación del proceso.

La asignación de los lotes de producción a los distintos talleres, ya sean propios o de terceros, se realiza, como se mencionó anteriormente, teniendo en cuenta dos factores fundamentales:

- Tarea en la que se especializa el taller (confección, terminación, etc.)
- Tipo de prenda que produce (pantalón, campera, remera, etc.)

Además, se tienen en cuenta la capacidad productiva del taller (cantidad de prendas/tiempo), calidad del trabajo, cumplimiento con los plazos de entrega, costo, entre otros.

Dicha asignación se realiza a medida que se va avanzando en la producción de los artículos, comenzando con aquellos para los cuales se dispone por completo de los insumos y materiales necesarios para su fabricación, y que la empresa considera de mayor prioridad. Terminados dichos artículos se van liberando nuevos lotes de producción, siguiendo el mismo criterio, y así hasta completar el total de la colección.

1.3 Definición del problema

Del análisis de la situación de la empresa y los objetivos que ésta se plantea, se desprende que el problema a resolver consiste en la determinación de la secuencia en que deben fabricarse los lotes de producción en los distintos talleres, de modo tal que el tiempo total de producción sea lo menor posible y permita cumplir con el plazo de entrega de la colección.

Específicamente, lo que se pretende resolver es la programación de los lotes de producción en los diferentes talleres con los que cuenta la empresa. Cada lote dispone de una serie de tareas u operaciones necesarias para ser fabricado, las cuales son determinadas de acuerdo al diseño de la prenda. A su vez, estas tareas tienen definido un orden en que deben ser realizadas, que no necesariamente es el mismo para todos los lotes, y que está determinado también por las características de diseño del artículo. Finalmente, cada una de estas tareas puede ser realizada en uno o más talleres, dependiendo del artículo que se trate.

La descripción realizada, como se verá más adelante, se corresponde con la definición del problema clásico de secuenciación de operaciones, más específicamente con una variante de dicho problema.

1.4 Objetivos

El objetivo de este trabajo es resolver el problema de secuenciación para el caso real presentado, desarrollando una herramienta que sea capaz de trabajar con la gran cantidad de variables que esta situación en estudio posee y que a su vez, proporcione una solución factible para dicho problema. Se pretende con esto ofrecer a la empresa un producto que le sirva de apoyo a la toma de decisiones y como guía en el momento de planificar su período de producción.

Teniendo en cuenta ello, se establecen los siguientes indicadores de éxito:

- La obtención de un programa desarrollado que sea accesible y sencillo de utilizar
- Resultados del mismo que devuelvan soluciones con un margen de error (respecto a la solución óptima) no mayor al 2%.

MARCO TEÓRICO

2.1 Introducción

La Planificación y Control de la Producción constituye, en toda empresa manufacturera, la base para que la misma pueda alcanzar sus objetivos. Implica establecer los parámetros productivos necesarios para que los objetivos se concreten en tiempo y forma, y de manera eficiente. Es por ello que dicha actividad juega un papel fundamental a lo largo del proceso productivo, interviniendo en todos los niveles del subsistema de operaciones.

Dentro de estas actividades se encuentran comprendidas la Planificación Estratégica, que contempla un horizonte temporal a largo plazo, la Planificación Táctica, enfocada al mediano plazo, y la Planificación Operativa, orientada al corto plazo (Machuca et al., 1997). En este trabajo se hará énfasis en este último nivel de planificación, más específicamente en lo que respecta a la Programación de Operaciones.

Dado que el caso en estudio, como se verá más adelante, se enmarca en un contexto de producción por lotes, se profundizará en las actividades de programación para esta configuración en particular, destacando la importancia de su realización.

A continuación se presentará brevemente el esquema de planificación en el subsistema de operaciones, el significado de la Programación de Operaciones y su rol en el proceso de producción por lotes, y un desarrollo más extenso del problema puntual de secuenciación para entornos de producción del tipo Job Shop y Flexible Job Shop, que corresponden al caso particular en estudio.

2.2 Planificación de la producción

El proceso de Planificación y Control de la Producción, como se mencionó anteriormente, se distribuye en tres niveles principales: estratégico, táctico y operativo. Para que exista una adecuada coordinación entre los objetivos, planes y actividades de cada uno de estos niveles, es necesario que dicho proceso adopte un

enfoque jerárquico que permita llevar adelante una gestión integrada de todo el subsistema de operaciones. En la Ilustración 1 : “Diagrama de Planificación y Control de la Producción se puede observar un esquema de dicho proceso.

La primera etapa de este proceso está constituida por el nivel estratégico, el cual se establece para un horizonte temporal a largo plazo. En esta fase se determinan el Plan de Ventas, que indica el nivel de demanda a satisfacer en el largo plazo, el Plan de Producción, que se desprende de éste y establece las cantidades a producir en términos anuales por tipo de producto, y el Plan Financiero, que determina las necesidades de recursos en base a los ingresos previstos en el Plan de Ventas, que permitirán llevar a cabo los objetivos planteados en ambos planes.

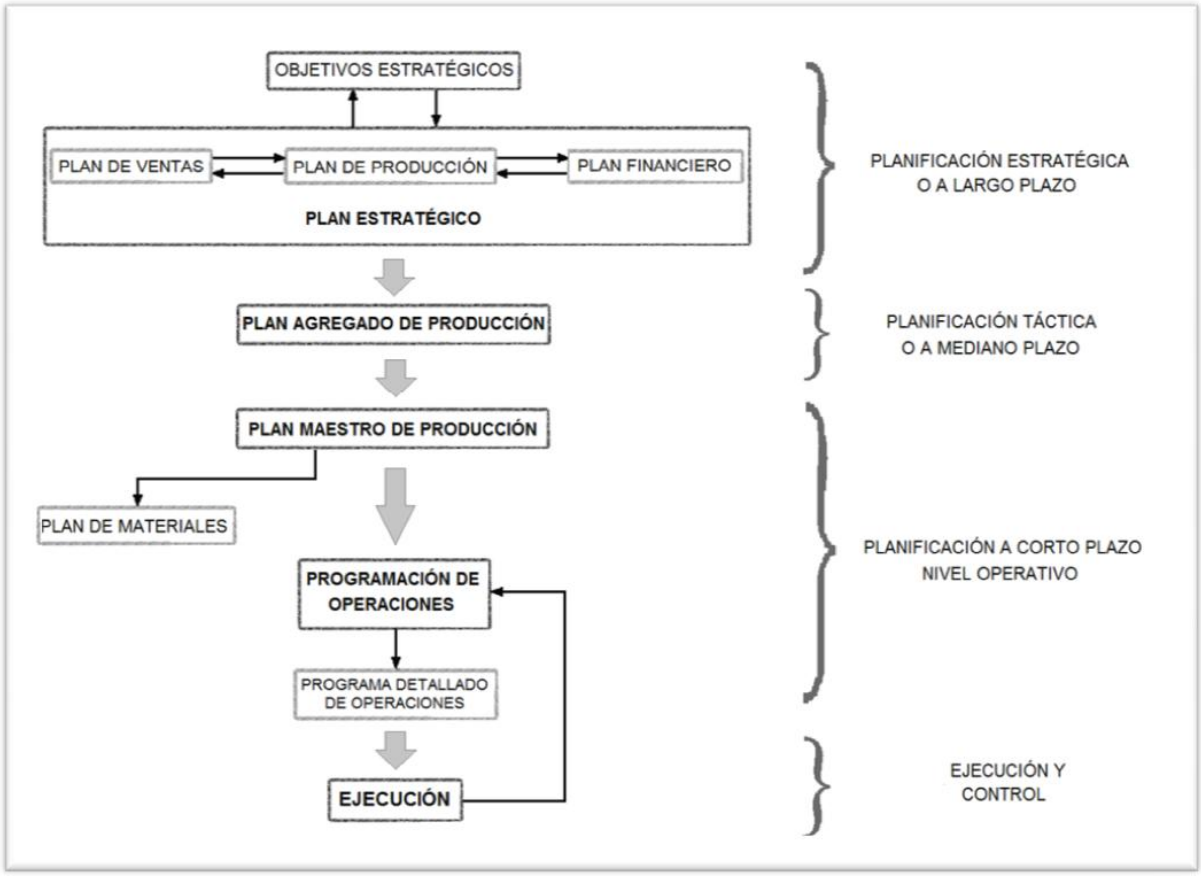


Ilustración 1 : “Diagrama de Planificación y Control de la Producción”

La siguiente etapa está asociada al nivel táctico o de planificación a mediano plazo. Consiste en fijar las principales variables productivas, aún en términos agregados de producto, para períodos de tiempo menor a los establecidos en el Plan Estratégico. De esta etapa surge la Planificación Agregada, compuesta por los planes agregados de producción y de capacidad. El propósito de la misma es lograr una coordinación entre el nivel estratégico y operativo, de modo que se concreten los objetivos fijados por la empresa en el Plan Estratégico.

Por último, a nivel operativo se lleva a cabo una planificación más detallada, descomponiendo las familias de producto en unidades de producto y reduciendo los períodos de planificación (por lo general a semanas). En esta fase se desarrollan, por un lado, el Plan Maestro de Producción, el cual se realiza para un período de tiempo no mayor a un año, el Plan de Materiales, que refleja las necesidades surgidas del anterior, y finalmente la Programación de Operaciones, que define cómo se llevará a cabo la producción en los distintos talleres o centros de trabajo, de modo que se cumpla la planificación establecida.

Todo lo mencionado anteriormente puede verse más claramente en la Ilustración 1, en donde se refleja la relación jerárquica entre los distintos niveles de planificación, a la vez que se distingue el importante lugar que ocupa la Programación de Operaciones en este proceso.

2.3 Programación de las Operaciones

La Programación de Operaciones es una función muy importante dentro del esquema operativo, dado que es a través de ella, básicamente, que se concreta la planificación de la producción. Su función es determinar para cada pedido, qué trabajos deberán realizarse, en qué forma, cómo se distribuirán los mismos en los distintos talleres o centros de trabajo según la capacidad disponible en cada momento y qué prioridades tendrá cada pedido, de modo tal que se cumplan las fechas de entrega planificadas, utilizando la menor cantidad de recursos posible y

manteniendo un mínimo inventario. También es útil como *feedback*[1] para ajustar la planificación maestra prevista, tal como lo señala Pinedo (2008), en el caso de no poder alcanzar los objetivos mencionados anteriormente.

De acuerdo al tipo de proceso que se realice, la programación de las operaciones puede resultar más o menos compleja. Las configuraciones por tipo de proceso que pueden presentarse son las siguientes (Dominguez Machuca et al., 1997).

- Producción en línea: se utiliza por lo general para grandes volúmenes de producción, con poca variedad de productos y un alto grado de estandarización. Se caracteriza por una disposición de los equipos de trabajo según el orden de los procesos o etapas de fabricación del producto.

- Producción por lotes: se recurre a esta configuración en ambientes productivos donde se dispone de la fabricación de varios productos, con distintos volúmenes de producción, menor estandarización y diferentes secuencias de fabricación. En estos casos, los equipos de trabajo se organizan en talleres o centros de trabajo, agrupados por tipo de actividad.

- Producción por proyectos: en estos casos la producción suele desarrollarse a pedido, disponiéndose los recursos de la empresa en función de cada proyecto en particular. Se caracteriza por un alto grado de detalle del producto a fabricar.

En el caso más completo, que corresponde a una configuración por lotes, se pueden distinguir tres etapas o actividades a realizar para obtener el Programa de Operaciones:

- La Asignación de Recursos, que indica qué operaciones de los distintos pedidos se realizarán en cada taller o centro de trabajo.

[1] En castellano *Retroalimentación* suele ser la traducción más utilizada. La palabra hace referencia al proceso de ajuste de las variables de entrada de un sistema en función de las variables de salida del mismo.

- La Secuenciación, que constituye la actividad fundamental de la programación de operaciones en cualquier configuración, y consiste en establecer las prioridades de fabricación de los distintos pedidos en los centros de trabajo.
- La Programación Detallada, que determina las fechas de comienzo y fin de cada actividad en cada centro de trabajo, así como de las operaciones de cada pedido para la secuenciación realizada.

Cada una de estas etapas tiene asociada a su realización distintos métodos operativos de resolución.

En el caso de la Asignación de Recursos, o carga a talleres, se utilizan métodos de prueba y error, como lo son las gráficas de carga; métodos optimizadores, como la asignación por métodos de transporte o mediante modelos de programación lineal, y métodos heurísticos, como por ejemplo el método de los índices.

La Secuenciación es un tanto más compleja que la anterior y presenta diversos métodos de resolución que dependen del tipo de configuración en particular. Existen métodos de secuenciación para procesos de producción en los que existe una única máquina o instalación (por ejemplo, el algoritmo de Kauffman), métodos para la secuenciación en varias máquinas (como las reglas de Johnson), y métodos, por lo general heurísticos o de simulación, que permiten resolver casos más generales y complejos. La fabricación en Job Shop, la cual se explicará más adelante, resulta uno de los casos más interesantes a resolver, tanto por su complejidad como por su aplicación práctica. Para estos casos, no existen métodos exactos que permitan alcanzar una solución óptima, sino que se resuelve mediante técnicas de simulación o métodos heurísticos, que encuentran soluciones aceptables más o menos cercanas al óptimo. Muchas de estas técnicas se encuentran apoyadas por el uso de sistemas informáticos que agilizan su resolución.

En cuanto a la Programación Detallada, muchas veces se encuentra resuelta directamente en la fase de secuenciación, al utilizar , por ejemplo, los diagramas de

Gantt como método de resolución. Otras veces sólo se dispone del ordenamiento de los distintos procesos en los centros de trabajo, y la determinación de las fechas de inicio y fin de cada actividad se establece a medida que se van ejecutando las mismas.

2.4 El flujo en los talleres

Como se mencionó anteriormente, la Programación de Operaciones, y más específicamente, la Secuenciación, dependen en muchos casos de la configuración con que disponga el sistema de producción.

Pueden distinguirse básicamente tres modelos distintos de flujos de trabajo en una organización (Salazar González, 2001):

- Flow-shop: La producción se encuentra comprendida por diferentes lotes, sin embargo, todos y cada uno de ellos deben respetar el mismo orden en el recorrido productivo dentro del taller. Cada etapa o proceso se encuentra compuesto por una única máquina capaz de realizar dicha tarea y la misma sólo puede realizar un único trabajo a la vez y de forma ininterrumpida. Existe la posibilidad de que algunos lotes no necesiten atravesar todos los procesos pero cada proceso podrá ser efectuado, para un mismo lote, una única vez a lo largo de todo el proceso productivo. Se trata de una configuración de tipo lineal.

- Job-shop: Al igual que en el caso del flow-shop, la producción, conformada por diversos lotes, debe respetar cierto orden predefinido. La diferencia radica en que cada lote puede tener un orden de procesamiento distinto y existe la posibilidad de que el mismo lote requiera atravesar por el mismo proceso en más de una ocasión. De la misma forma que el caso anterior, por cada proceso existe una única máquina disponible para realizar el trabajo, la cual sólo puede realizar un único trabajo a la vez y de forma ininterrumpida. Se trata entonces de determinar el orden o secuencia en que ingresarán los distintos lotes de producción a las diferentes máquinas, de modo tal que respetando el orden predefinido de operaciones para cada lote y la capacidad

disponible en cada taller o centro de trabajo, se cumplan las fechas de entrega de los pedidos, empleando el menor tiempo total en la obtención de todos los lotes a procesar.

- Open-shop: la diferencia fundamental de este modelo respecto a los anteriores, radica en que el orden de los procesos dentro de la secuencia productiva carece de importancia. Es decir, si bien cada lote productivo debe atravesar una determinada cantidad de procesos para ser completado, el orden en que se efectúen dichas operaciones es indistinto. De la misma forma que en los casos anteriores, por cada proceso existe una única máquina disponible para realizar el trabajo, y ésta sólo puede realizar un único trabajo a la vez y de forma ininterrumpida.

El caso a analizar en este trabajo corresponde a la Programación de Operaciones para una configuración del tipo Job Shop. Se explicará con más detalle en los puntos subsiguientes en qué consiste la Programación de Operaciones para ambientes de trabajo del tipo Job Shop y Flexible Job Shop (que es una generalización del anterior).

2.5 El problema clásico de secuenciación JSSP

La Programación de las Operaciones para entornos de producción del tipo Job Shop constituye uno de los casos más interesantes de resolver. Tal es así que varios autores han dedicado particularmente sus estudios a ello, dando origen al problema clásico de secuenciación JSSP (Job Shop Scheduling Problem).

Este tipo de problemas representa básicamente la programación a nivel de operaciones de la utilización de los talleres o centros de trabajo. Es una actividad que resulta por demás importante, particularmente en aquellos casos donde las instalaciones de la organización resultan ser un cuello de botella, es decir, donde la demanda supera a la capacidad productiva y es menester minimizar los tiempos de máquina parada, como así también, en aquellos casos en donde los tiempos de entrega resultan ser una ventaja competitiva.

Para entender en qué consiste este problema, supongamos un conjunto de N lotes que representan la totalidad de la producción a procesar. Cada lote $n \in N$ se encuentra compuesto por una suma de tareas O_n y cada uno de estos conjuntos posee un orden entre sí, de manera que cada tarea $o \in O$, salvo la primer actividad de cada lote, posee otra tarea *o'* que la precede. Esto implica, por consiguiente, que la actividad *o* no puede llevarse adelante hasta que la tarea predecesora *o'* no ha sido finalizada y por carácter transitivo, hasta que la totalidad de las actividades que la preceden no se hayan llevado a cabo.

A su vez cada operación *o* debe ser procesada en una única máquina o taller $m \in M$ (de aquí en adelante utilizaremos la palabra máquina para hacer referencia a cualquiera de los dos[2]), durante un tiempo conocido t_o. Cada máquina *m*, puede llevar adelante únicamente una actividad a la vez y de forma ininterrumpida.

El problema de secuenciar todas las tareas del conjunto *O* en sus respectivas máquinas del conjunto *M* es conocido como JSSP. L a resolución del problema consiste en encontrar una secuencia que minimice el tiempo total de producción, *makespan,* calculado como el tiempo transcurrido desde que se comienza a procesar la primera tarea del primer lote hasta que se finaliza con la última tarea del último lote.

Este tipo de problemas es un problema de combinatoria del tipo *NP-hard*[3], y dentro de esta categoría es considerado como uno de los más difíciles de resolver. Es ampliamente conocido tanto en el mundo empresarial como en el académico debido a su frecuencia de aparición en distintos ámbitos organizacionales. Dada su complejidad, rara vez se utilizan métodos exactos (Peña y Zumelzu, 2006) para su resolución, salvo para casos excepcionales (Nahmias, 2007); por lo cual generalmente se suele resolver mediante algún método heurístico o de simulación a través del uso de sistemas informáticos (López de Haro et al., 2004).

[2]Si bien en la teoría estudiada se denomina taller al espacio físico en donde se encuentran las máquinas, en el caso en estudio cada tarea puede que deba realizarse en un taller externo a la empresa y por consiguiente, a fines del problema, tomaría el carácter de máquina.

[3]NP (*Non-deterministic Polynomial-time*) es una categoría utilizada en complejidad algorítmica, la misma hace referencia a la imposibilidad de determinar a priori un tiempo de procesamiento necesario para la resolución del problema.

2.6 El FJSSP

Una variante muy extendida del problema tradicional del JSSP es el FJSSP, que tiene que ver con la secuenciación en ambientes de trabajo flexibles, es decir, configuraciones productivas para las cuales existe la posibilidad de que todas o algunas de las operaciones a realizar puedan efectuarse en más de una máquina. A diferencia del JSSP, para el cual se establece en su enunciado que cada tarea $o \in O$ debe ser realizada únicamente en una máquina $m \in M$, en el FJSSP existe un conjunto de máquinas $M_o \subset M$ que son capaces de realizar la misma tarea o.

El problema a resolver en este caso consiste en secuenciar las tareas del conjunto O en las máquinas del conjunto M, estableciendo para cada *tarea* o la máquina del conjunto M_o que realizará dicha actividad, de manera de minimizar el makespan.

Este tipo de problemas, dependiendo de la cantidad de posibilidades existentes para cada tarea, puede resultar considerablemente más complejo de resolver que el JSSP tradicional, debido a que la cantidad de combinaciones existentes aumenta de forma exponencial en función de la cantidad de tareas que poseen más de una alternativa. Al igual que para el JSSP, se suelen utilizar para su resolución, métodos heurísticos o de simulación apoyados por el uso de sistemas informáticos.

En el siguiente ejemplo, se pueden observar las características detalladas para ambos tipos de problemas, así como también identificarse claramente las diferencias existentes entre uno y otro. En la Tabla 1 se muestra un caso de JSSP y en la Tabla 2, uno de FJSSP.

En el primer caso, puede verse que para cada proceso de cada lote, existe una única máquina que puede efectuar ese trabajo. La cantidad de trabajos o procesos de cada lote no necesariamente tienen que ser iguales.

En el segundo caso, se observa que, a diferencia del anterior, un mismo proceso o trabajo puede ser realizado por más de una máquina (por ejemplo, el proceso A del lote 1 puede ser procesado en la máquina 1 o en la máquina 5

indistintamente). Es decir, existen procesos para los cuales se tienen varias alternativas de procesamiento. El hecho de que para al menos un proceso se tenga más de una alternativa es la diferencia fundamental entre el JSSP y el FJSSP.

JSSP				
		PROCESO A	**PROCESO B**	**PROCESO C**
Máquinas que pueden realizar la actividad	**LOTE 1**	Máquina 1	Máquina 2	-
	LOTE 2	Máquina 1	Máquina 3	Máquina 2

Tabla 1: Ejemplo de problema tipo JSSP

FJSSP				
		PROCESO A	**PROCESO B**	**PROCESO C**
Máquinas que pueden realizar la actividad	**LOTE 1**	Máq. 1/ Máq. 5	Máquina 2	-
	LOTE 2	Máquina 1	Máq. 3/ Máq. 6/ Máq. 7	Máq. 2/ Máq. 4

Tabla 2: Ejemplo de problema tipo FJJSP

2.7 Medidas de eficiencia de la secuenciación

Si bien la secuenciación puede tener como objetivo primordial alcanzar una secuencia de ordenamiento de los pedidos que permita reducir lo más posible el tiempo total de producción, con frecuencia se tendrán en cuenta otros objetivos adicionales que dependerán del tipo de proceso que se realice y de cada empresa en particular.

Dichos objetivos podrán buscarse en forma conjunta, o podrá seleccionarse de entre ellos aquel que resulte de mayor relevancia para el caso en particular. Es importante aclarar que si bien la incorporación de varios criterios de optimización generará soluciones más integrales a nivel organizacional, cuantos más criterios se seleccionen, tanto más compleja se volverá la secuenciación.

A continuación se detalla una serie de posibles criterios de optimización, que servirán para medir la eficiencia del proceso de secuenciación una vez realizado, y que dependerán de cada caso en particular (Nahmias, 2007):

- Cantidad de pedidos atrasados: Sobre el total de la producción, minimizar la totalidad de pedidos que se entregaron posterior al plazo estipulado.
- Tardanza: Se define como el tiempo transcurrido entre la fecha de entrega establecida y la fecha real de entrega del pedido más atrasado.
- Retardo promedio (tardanza media): Se calcula como la suma de los días de atraso de cada pedido sobre el total de pedidos.
- Anticipo medio: A la inversa del caso anterior, se calcula como la suma de los días de anticipo (tiempo transcurrido desde que se finaliza la producción de un pedido y la fecha de entrega estipulada) de cada pedido sobre el total de pedidos.
- Trabajo en proceso (WIP[4]): Se calcula como la suma de pedidos que se encuentran siendo procesados.
- Reducir tiempos de preparación: Minimizar el total de tiempo empleado de preparación de la maquinaria entre el proceso de distintos lotes.
- Makespan[5]: Es el tiempo transcurrido desde que se comienza a procesar el primer proceso del primer lote hasta que se termina el último proceso del último lote.

Lo anteriormente enumerado representa sólo un extracto de los principales criterios empleados a nivel organizacional para la medición de la eficiencia en la secuenciación.

[4]Del ingles Work-In-Process: *WIP*
[5] Esta terminología es ampliamente utilizada en el ámbito académico internacional.

2.8 Métodos de resolución

La planificación de las operaciones para los casos presentados resulta de gran complejidad debido a la cantidad de variables que se encuentran involucradas en la misma. La cantidad de lotes a fabricar, el número de procesos a realizar por cada uno, la cantidad de máquinas disponibles para llevarlos a cabo, así como el número de objetivos que se pretenda analizar, resultan de por sí factores determinantes de las dimensiones del problema a resolver.

La búsqueda de una solución óptima por medio de la simple enumeración de posibles soluciones es una tarea que se torna prácticamente imposible por el tiempo que esto implica. Hay que tener en cuenta que, por ejemplo, para problemas del tipo JSSP, con m máquinas disponibles y l lotes a fabricar, existen $(l!)^m$ soluciones posibles a comparar. Es evidente que cuánto mayor es la dimensión del problema, tanto más difícil se torna poder resolverlo.

Existe una gran cantidad de modelos para resolver este tipo de problemas (Vicentini y Puddu, 2003), basados algunos en métodos exactos y otros en métodos heurísticos o de simulación. El inconveniente con ellos es que muchas veces su aplicación no permite llegar a un óptimo, o si lo hacen, no contempla éste todos los objetivos del problema. En la mayoría de los casos, resultan imposibles de llevar a la práctica, ya sea por lo complejo del método en sí o por la cantidad de lotes y máquinas disponibles. No existe en verdad técnica alguna que permita obtener el óptimo del problema de secuenciación para cualquier caso posible. Es por todo esto que las empresas optan por utilizar métodos que, si bien no permitan llegar a una solución óptima, sí proporcionen soluciones satisfactorias a su problema de secuenciación. Este tipo de problemas se encuentran enmarcados en la denominada Optimización Combinatoria.

2.9 Optimización combinatoria

La programación matemática es un campo comprendido dentro de las matemáticas aplicadas orientado al diseño de metodologías para resolver problemas de optimización con recursos limitados (Salazar González, J.J. 2001).

La optimización combinatoria es una parte de la programación matemática y se encuentra destinada a resolver problemas del tipo:

$$\min\{ f(x): x \in S\}, \qquad |S| < \infty$$

Para enfrentar este tipo de problemas se han desarrollado metodologías de resolución, considerablemente exitosas, basadas en algoritmos de cálculo que requieren la utilización de la computadora. Estos métodos tienen la particularidad que si bien no siempre permiten encontrar una solución óptima al problema planteado, sí devuelven soluciones factibles, muy cercanas a ella, ahorrando significativamente el tiempo computacional empleado en la resolución. A este tipo de metodología se la denomina metaheurística. El prefijo “meta” significa “más allá”, y tiene que ver con que estas técnicas están diseñadas para mejorar los procedimientos heurísticos a fin de obtener un rendimiento mayor.

Los métodos metaheurísticos utilizan algoritmos aproximados de optimización y búsqueda de propósito general. Parten de una solución factible inicial (habitualmente obtenida mediante alguna otra técnica heurística) y mediante alteraciones de la solución, van iterando a otras soluciones factibles cercanas a ellas. De este modo, se va almacenando la mejor solución encontrada hasta que se cumple un determinado criterio de terminación.

Entre los principales métodos pertenecientes a esta categoría se encuentran los algoritmos genéticos, el método de la colonia de hormigas, la búsqueda tabú y el recocido simulado. Este último es el método seleccionado en este trabajo para la resolución del problema de secuenciación, con lo cual se desarrollará más extensamente en el siguiente apartado. El resto de los métodos se encuentran descriptos en el Anexo A, basados en lo desarrollado por Duarte Muñoz et al. (2007).

2.10 Recocido simulado

Este método fue introducido por Kirkpatrick et. al. (1980) como una técnica no determinista de optimización de problemas combinatorios. Debe su nombre a la

similitud que guarda el fenómeno físico de recocido de los metales con la lógica empleada por el algoritmo para la búsqueda de una solución.

Básicamente, este algoritmo simula los cambios energéticos que sufre un sistema de partículas conforme decrece la temperatura, hasta que el mismo converge a un estado de mínima energía en donde se vuelve estable. En el caso de los problemas de optimización combinatoria el nivel de energía viene dado por la función de costos asociada al problema, la cual se trata de minimizar.

Desde su aparición, el recocido simulado ha sido empleado para resolver una gran variedad de problemas de optimización complejos, tanto basados en variables continuas como discretas. Entre ellos pueden mencionarse el diseño de bases de datos distribuidas, la configuración de aplicaciones multiprocesador, el diseño de filtros digitales e inclusive problemas de secuenciación tales como el JSSP.

El recocido simulado pertenece a una clase de algoritmos de aproximación denominados de búsqueda local, cuyo funcionamiento está basado en la exploración de vecinos. Esto significa que siendo S el conjunto finito de todas las posibles soluciones del problema, se asume una estructura de vecindad que define para cada solución $i \in S$ un conjunto $S_i \subset S$ tal que todos sus elementos son cercanos a i en algún sentido. Se asume que $j \in S_i \leftrightarrow i \in S_j$. El mecanismo de generación de vecinos se define como la forma de seleccionar una solución j en la vecindad S_i de la solución i.

Su funcionamiento comienza entonces con la generación de una solución aleatoria para el problema y el cálculo del costo de dicha solución. A partir de allí, mediante algún mecanismo de generación establece una nueva solución vecina a la anterior, la cual es aceptada si su costo es inferior. En el caso de que no lo sea, la nueva solución es aceptada con una probabilidad que decrece exponencialmente con el cociente de la diferencia de costos de ambas soluciones y un parámetro T que representa la temperatura. El hecho de que la probabilidad de aceptar soluciones peores no sea nula permite que el algoritmo no quede atrapado en

mínimos locales. Típicamente, si la nueva solución no fuera aceptada, se probaría un nuevo movimiento desde la solución anterior.

A lo largo de la ejecución del algoritmo, la temperatura *T* va disminuyendo en forma geométrica, reduciéndose así la probabilidad de que se generen soluciones de mayor costo. El algoritmo suele terminar cuando no se ha obtenido ninguna mejora en un número determinado de desplazamientos, aunque también pueden emplearse criterios de finalización más complicados.

2.10.1 Algoritmo de Metrópolis

Este algoritmo fue desarrollado por Metrópolis et. al. (1953) para describir el comportamiento de las moléculas en los metales, al someterlos al proceso de recocido. Kirkpatrick años después, se inspiró en él para diseñar el recocido simulado.

El algoritmo de Metrópolis, tal como él lo diseñara, está basado en las leyes de la Termodinámica, y establece lo siguiente: dado un estado i con energía E_i (correspondiente a un estado físico del metal en un instante determinado), se genera un nuevo estado j mediante un mecanismo de perturbación. Se calcula la energía del nuevo estado Ej, y si (Ej – Ei) < 0 entonces se acepta el nuevo estado j. Caso contrario, se acepta con una probabilidad exp ($\frac{E_i - E_j}{K_b * T}$), donde T representa la temperatura y K_b es la constante de Boltzmann.

El algoritmo de Metrópolis, por lo tanto, es la base que estructura el comportamiento del recocido simulado. A través del establecimiento de sus parámetros es que se puede demostrar la convergencia de este método a la solución óptima.

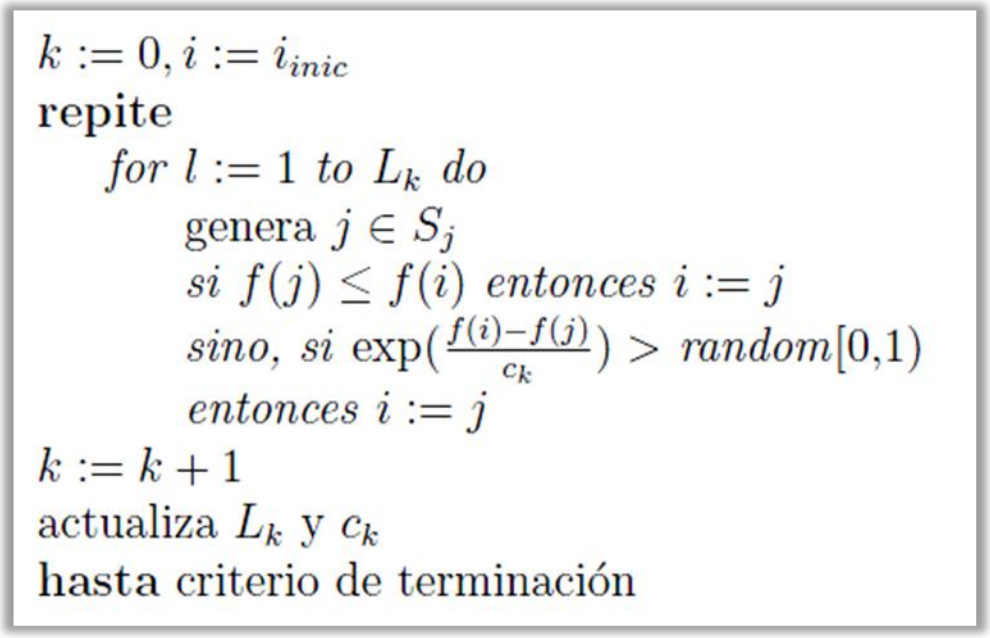


Ilustración 2: "Algoritmo de Metrópolis"

En la Ilustración 2 se puede visualizar la estructura de dicho algoritmo. Las variables que intervienen en él son (Valenzuela, 2004):

- *k*: Trabaja como contador, inicia en 0 y su valor se incrementa en una unidad cada vez que el algoritmo termina con un ciclo completo.
- *i; j*: Son posibles soluciones, combinaciones que evaluadas en la función objetivo permiten conocer su costo. En el caso del esquema, *i* es la solución inicial, mientras que *j* es una solución vecina de la misma.
- *l*: Es un contador de ciclos, comienza en un 1 y su valor se va incrementando a medida que los ciclos transcurren. Una vez que el ciclo se completa, el contador vuelve a su valor inicial.
- L_k: Es una variable que determina la cantidad de ciclos que el algoritmo debe realizar antes de disminuir la temperatura de comparación.

- C_k: Es la constante de temperatura. En los casos en que la solución vecina posee un costo operativo mayor la probabilidad de aceptación de la nueva solución depende directamente de este valor. A medida que el proceso del algoritmo avanza, el valor de la temperatura disminuye y por ende también lo hace la probabilidad de aceptación de soluciones que posean un costo mayor que la original.

El algoritmo finaliza su ejecución cuando se cumple el criterio de terminación adoptado. Comúnmente el criterio suele ser que el programa haya recorrido todos los ciclos determinados por la variable L_k sin haber podido realizar ningún cambio en la solución.

Es importante aclarar, que el algoritmo no utiliza la constante de Boltzmann, tal como lo hiciese Metrópolis en su algoritmo original, puesto que este valor sólo tiene significado en la representación del problema físico.

MATERIALES Y MÉTODOS

3.1. Materiales

3.1.1 Lazarus

Lazarus constituye el recurso principal utilizado en este trabajo. Es una herramienta de programación que posee un alto grado de compatibilidad con *Delphi* y sus librerías (Free Pascal Lazarus Project, 2010). Esta herramienta ha sido desarrollada sobre el compilador *Free Pascal*, con la diferencia que el primero brinda la posibilidad de realizar los desarrollos bajo el paradigma de la *Programación Orientada a Objetos* (POO). Dentro de las características principales que cabe resaltar de esta útil herramienta podemos mencionar que es un compilador de código abierto y se encuentra disponible para su utilización bajo distintas plataformas, tales como *Windows*, *Linux*, *Mac*, etc. (Ver Ilustración 3: "Captura de pantalla de Lazarus")

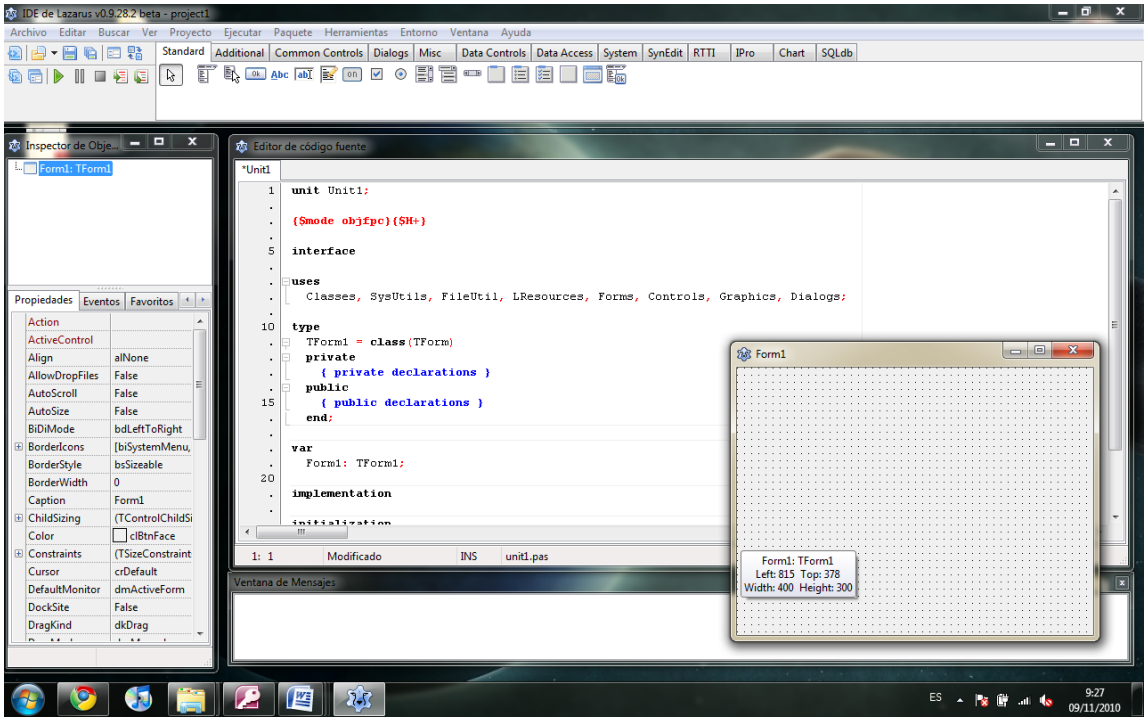


Ilustración 3: "Captura de pantalla de Lazarus"

3.2. Métodos

Los métodos que se presentarán a continuación fueron desarrollados particularmente para la resolución del problema planteado hasta aquí. Dichos métodos se adaptan tanto a las necesidades del problema enunciado (FJJSP) como a las del algoritmo seleccionado para su resolución. Cabe destacar que para la elaboración de estos métodos se ha tomado como guía los desarrollos presentados por Cortés Rivera (2004).

3.2.1 Representación del problema

Como punto de partida para la resolución del problema, es necesario establecer el método a emplear para la representación de una posible solución. Existen numerosas formas llevar a cabo la misma, pero todas están basadas en los mismos principios básicos. En primer lugar para toda solución factible debe existir una representación en el sistema elegido y la misma debe ser unívoca, es decir, que represente una y solo una solución del sistema; y en segundo lugar, no puede existir una representación de una solución no factible (o inviable).

Se puede presentar el caso de que una misma solución posea varios esquemas que la representen, pero esto puede resultar en detrimento del proceso de optimización, ya que a lo largo de la resolución existiría la posibilidad de una revaluación de la misma solución bajo dos combinaciones diferentes.

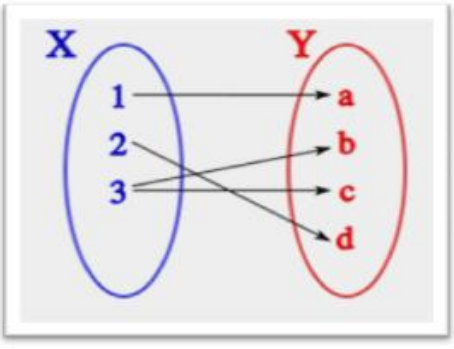


Ilustración 4: "Representación unívoca y no biunívoca"

Lo explicado anteriormente se vislumbra con mayor claridad en la Ilustración 4 en donde se esquematiza la relación existente entre el conjunto de soluciones reales, el “X”, con el conjunto de las soluciones representadas por el sistema elegido, el “Y”.

De todas formas en el presente trabajo se desarrolló una representación que, además de cumplir con lo hasta aquí mencionado, también es biunívoca y por consiguiente cada solución del sistema se encuentra ligada a una única solución real y viceversa, tal cual se puede observar en la Ilustración 5.

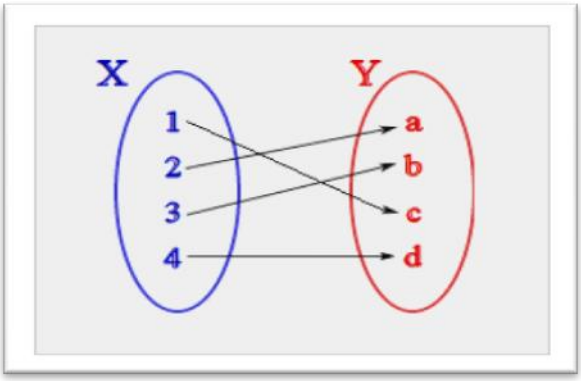


Ilustración 5: "Representación biunívoca"

Concretamente el esquema que se diseñó es el que figura en la Ilustración 6. La solución se encuentra conformada por *n+1* vectores filas, donde *n* es el número de lotes. Uno de dichos vectores es el vector principal, el cual define el orden en el cual se van a ir procesando los lotes, de manera de respetar el orden de asignación. Cuando el número de un lote se repite, debe continuarse con el siguiente proceso del mismo.

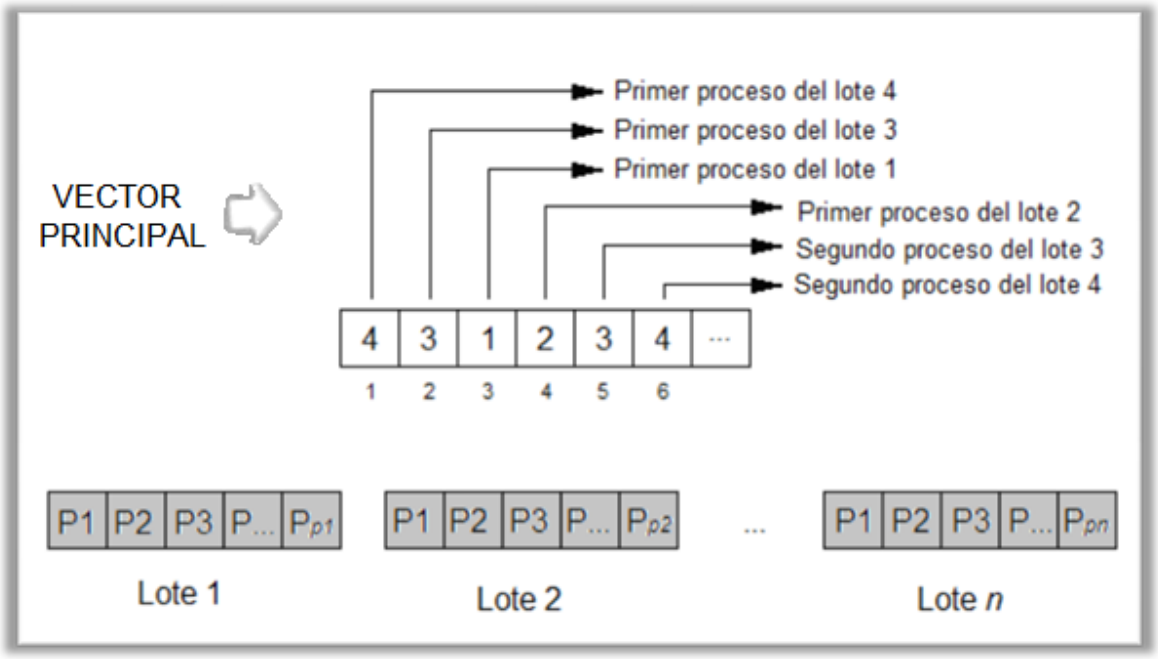


Ilustración 6: "Representación de la solución"

En el caso del ejemplo, se comienza procesando la primera actividad del *lote 4*, luego se continúa con el *lote 3* y así sucesivamente. En la quinta posición del vector puede verse que se repite el *lote 3*, lo cual indica que se debe continuar con la segunda tarea de este lote. El largo del vector principal varía según el problema y es igual a la suma del total de procesos de todos los lotes.

Los *n* vectores restantes corresponden uno a cada lote, definiendo cada posición del vector el puesto de trabajo en el cual se realizará la actividad asignada. La cantidad de elementos que cada vector posee es igual a la cantidad de procesos o actividades que se requieren para fabricar el lote que representa. Estos vectores carecen de importancia en el caso de los problemas tipo JSSP, ya que en ellos, cada proceso sólo puede ser llevado a cabo en un único puesto de trabajo, lo cual significa que el puesto de trabajo queda definido directamente por la actividad, en el vector principal.

En el caso de la Ilustración 6, el vector correspondiente al *Lote 1* posee P_{p1} procesos y para cada uno de ellos se encuentra aclarado, en el elemento del vector,

el centro de trabajo que realizará la actividad. Lo mismo se repite para cada uno de los lotes que conforman el problema.

3.2.2 Generación de vecinos

La generación de vecinos se lleva a cabo en el programa mediante tres métodos distintos, permitiendo de esta forma lograr una mayor facilidad para escapar de óptimos locales. Vale la pena aclarar que cuanto mayor sea la complejidad de la estructura vecinal, menor será la probabilidad de que el algoritmo quede atrapado en una solución ineficiente. A continuación se detalla el funcionamiento de cada uno de ellos:

1. *Intercambio de procesos de la solución*: a partir de la alternativa de solución anterior se realiza un intercambio aleatorio de dos procesos dando como resultado una nueva alternativa. Para ello es necesario que los procesos a intercambiar correspondan a diferentes lotes, puesto que el orden de los procesos para un mismo lote es inalterable de acuerdo a la definición del problema. Luego la condición necesaria para realizar este proceso es que $alternativa_i$ <> $alternativa_j$.(Ver Ilustración 7)

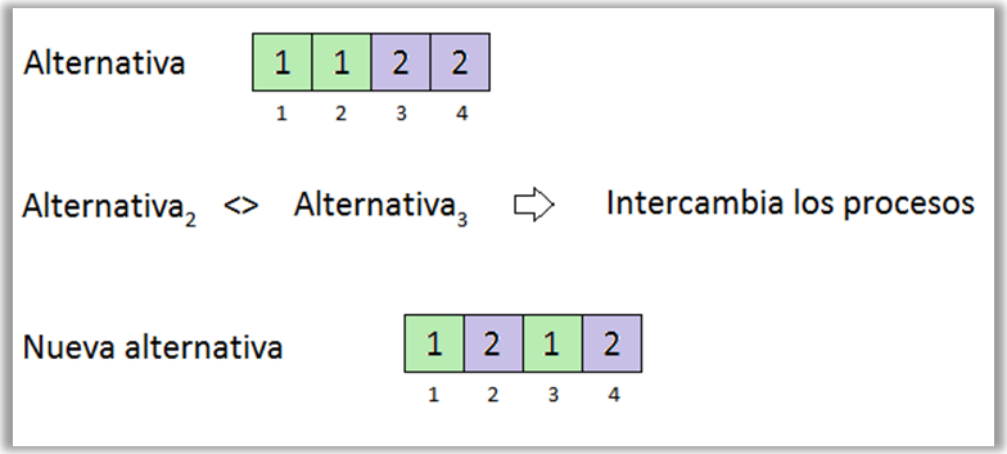


Ilustración 7: "Primer generador de vecino"

2. *Desplazamiento de los procesos*: consiste en elegir un proceso de la alternativa anterior al azar y cambiarlo de posición, desplazando todos los procesos intermedios un lugar entre la posición original y la nueva posición de dicho proceso. Es decir, se elige un proceso al azar $alternativa_i$, se lo desplaza a otra posición aleatoria *j*, y todos los procesos intermedios se desplazan una posición. (Ver Ilustración 8)

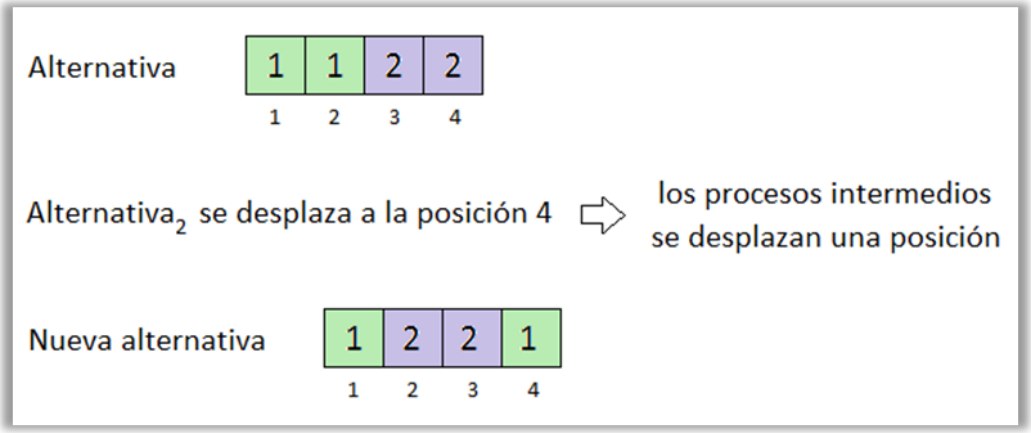


Ilustración 8: "Segundo generador de vecino"

3. *Cambio de máquina*: este método se utiliza sólo cuando el programa resuelve un problema del tipo FJSSP, puesto que en este caso un mismo proceso puede ser realizado indistintamente en más de una máquina. El método consiste en elegir un proceso al azar, y cambiar aleatoriamente la máquina en que se realiza, de modo que el vector alternativa continúa siendo el mismo, pero el makespan correspondiente varía. La condición para efectuar este cambio es que el proceso elegido $alternativa_i$ pueda realizarse en más de una máquina. (Ver Ilustración 9)

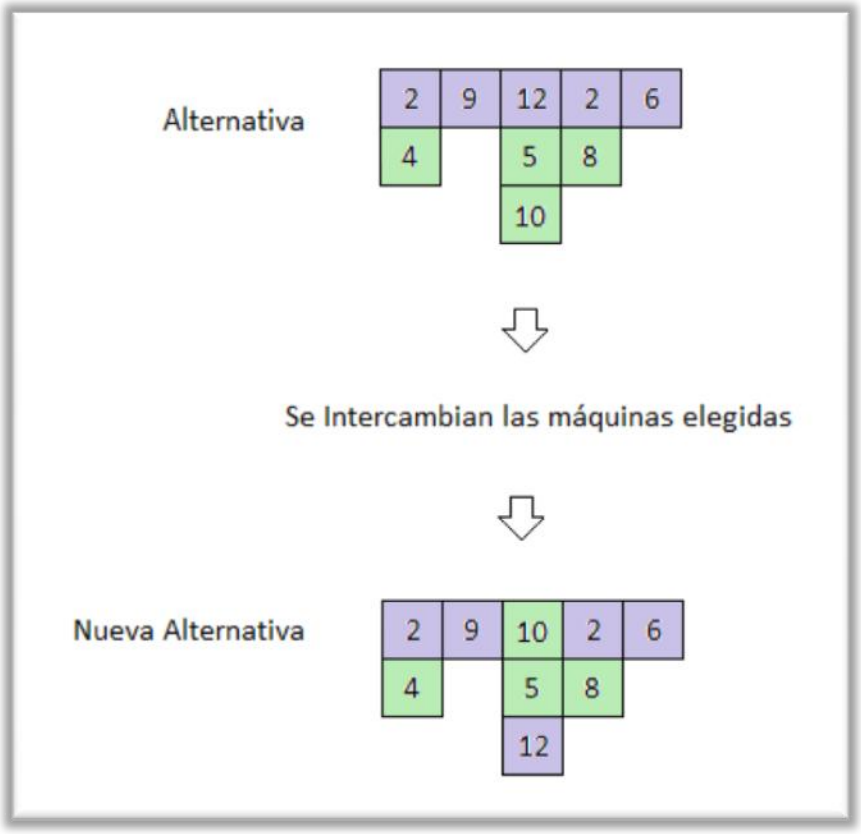


Ilustración 9: "Tercer generador de vecino"

La combinación de estos tres métodos descriptos permite obtener un algoritmo de mayor eficiencia a la hora de encontrar la solución.

3.2.3 Cálculo del makespan

El cálculo del makespan es el proceso mediante el cual se le asigna un valor a la solución alcanzada. El mismo está dado por el tiempo total de producción de la solución en cuestión. Para su obtención, el programa almacena una serie de datos que le permiten determinar el tiempo que requiere la configuración en evaluación para cumplir con el programa de producción necesario.

Dicha actividad se realiza mediante dos vectores de almacenamiento – como puede observarse en la Ilustración 10 - uno para el tiempo de los lotes y otro para el tiempo de los talleres o centros de trabajo. Cada elemento del vector Lotes corresponde a un lote determinado, almacenándose allí el tiempo total de

procesamiento del mismo. De igual forma, cada elemento del vector Centros de trabajo corresponde a un taller en particular, y se almacena en él el tiempo de procesamiento total correspondiente. El largo de ambos vectores está determinado por la cantidad de lotes a procesar y la cantidad de talleres disponibles, respectivamente.

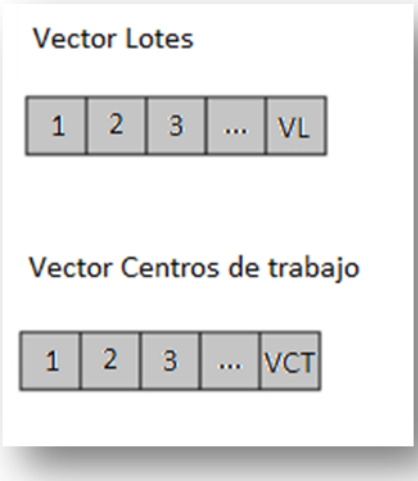


Ilustración 10: "Vectores para el cálculo del Makespan"

Para calcular el makespan el programa toma como tiempo cero (t_0) el inicio de la primera actividad, es decir, el momento en que comienza la producción del primer proceso del primer lote. Para ello toma como referencia el vector solución, el cual indica el orden de procesamiento de los distintos lotes. En base a ello, y teniendo en cuenta el tiempo de procesamiento de cada taller para cada proceso, el programa va calculando el tiempo de inicio y fin de cada actividad, siempre en referencia a t_0.

El programa comienza consultando el primer ítem del vector solución, analizando qué lote inicia la secuencia, en qué máquina será procesado y cuánto tiempo demorará. Una vez realizada dicha actividad actualiza los valores de los vectores lote y centros de trabajo, en aquellos ítems que corresponda, con el tiempo de procesamiento.

A medida que el programa avanza en el proceso de cálculo del makespan se encuentra con la repetición del procesamiento de un lote o la utilización de una máquina que posee una asignación previa, para lo cual consulta cuál valor es mayor, si el correspondiente al centro de trabajo o al lote, y le suma el tiempo de procesamiento correspondiente. Finalmente actualiza ambos ítems, el del vector centro de trabajo y el del vector lote, con este nuevo valor.

Este proceso es repetido hasta terminar de recorrer el vector solución; lo que se obtiene es el tiempo o fecha final de procesamiento de cada lote y de cada máquina. Dado que todos están calculados en función de t_0, aquel que presente el mayor valor es el que corresponde al momento de finalización de la última actividad procesada. Este tiempo es, precisamente, el tiempo total de procesamiento de toda la producción, es decir, el valor de makespan.

Vale aclarar que este cálculo permite tener en cuenta los tiempos de espera que pudieran existir durante el procesamiento de un lote, así como también los tiempos muertos o de ocio de cada taller, puesto que los mismos si bien no representan un tiempo productivo, existen e influyen en el valor total de procesamiento de la producción.

3.2.4 Aceptación de la solución

El programa almacena tres configuraciones de soluciones distintas. Por un lado, las dos básicas del algoritmo de recocido, que son la solución actualmente aceptada y la vecina (o alternativa), la cual podrá pasar a ser la aceptada dependiendo de su evaluación. Por otra parte, almacena también una tercera solución, que es la mejor hallada hasta el momento.

La decisión de almacenar esta última es para evitar la pérdida de la mejor solución encontrada a lo largo de la ejecución del algoritmo, como consecuencia de aceptar en el camino una solución inferior. Hay que recordar que esta posibilidad existe debido a la naturaleza del método.

En cada ciclo la solución vecina se evalúa y compara con la solución actualmente aceptada, determinando si mejora o no a esta última, en función de sus

makespan. En caso de mejorarla, pasa a ser automáticamente la nueva solución aceptada. Caso contrario, se la acepta con una cierta probabilidad, que depende de los parámetros del problema. La nueva solución aceptada se compara a su vez con la mejor solución, y en caso de superarla, se guarda también como mejor solución. La solución vecina vuelve a generarse a través de alguno de los mecanismos de generación explicados anteriormente, y a partir de allí, se repite el ciclo.

3.2.5 Velocidad de enfriamiento

La velocidad de enfriamiento (α) es una constante que multiplica a la constante de comparación a medida que el proceso de resolución avanza para que este valor disminuya y así resulte más difícil aceptar configuraciones con un costo de implementación (makespan) mayor.

Dicho parámetro resulta clave en la ejecución del problema, ya que un enfriamiento muy acelerado implicaría la obtención de un resultado rápido y probablemente de baja calidad, mientras que un valor elevado implicaría a su vez correr el riesgo de aumentar considerablemente los tiempos de resolución.

Por tal motivo se decidió recurrir a la teoría para establecer un valor adecuado para esta variable. Kuik y Salomon (1990) establecieron que el rango más conveniente para el enfriamiento en el recocido simulado es entre [0.8,0.99]. Debido a que este análisis se realizó hace 20 años y el poder de procesamiento de las computadoras aumentó exponencialmente con el paso del tiempo, se consideró oportuno privilegiar al resultado final frente al tiempo de ejecución y por ende, tomar el valor de $\alpha = 0.99$.

RESULTADOS Y DISCUSIÓN

Como resultado del análisis y estudio del caso presentado, se desarrolló un software capaz de resolver el problema del FJSSP, que sirve a su vez como herramienta de apoyo para la toma de decisiones en la fase de programación de operaciones.

El mismo está diseñado de modo tal que su aplicación es independiente de la cantidad de lotes a fabricar en cada temporada y de los tiempos de producción correspondientes. Esta característica le otorga al programa una gran versatilidad, ya que puede ser utilizado reiteradamente sin importar que varíen los parámetros productivos. Hay que tener en cuenta que cada temporada de producción implica la fabricación de artículos exclusivos de esa colección, con lo cual la cantidad de lotes de una temporada a otra puede variar.

Al no contar con una base histórica de datos de la empresa, no se pudo realizar una programación a posteriori de las operaciones en temporadas anteriores. Es por ello que la eficiencia del software se evaluó en función de los resultados obtenidos en la resolución de problemas estándares (para los cuales se conocen sus resultados) y no en función de los valores reales de producción de la empresa en particular.

A continuación se explicará más detalladamente en qué consiste la herramienta desarrollada y se presentarán los resultados obtenidos con la misma.

4.1 Programa desarrollado

El software está diseñado en forma genérica, de modo tal que puedan resolverse distintos casos del FJSSP, variando para cada uno de ellos la cantidad de lotes y sus respectivos tiempos de producción. La resolución del problema clásico de secuenciación JSSP queda incluida dentro de éstos como un caso particular del anterior.

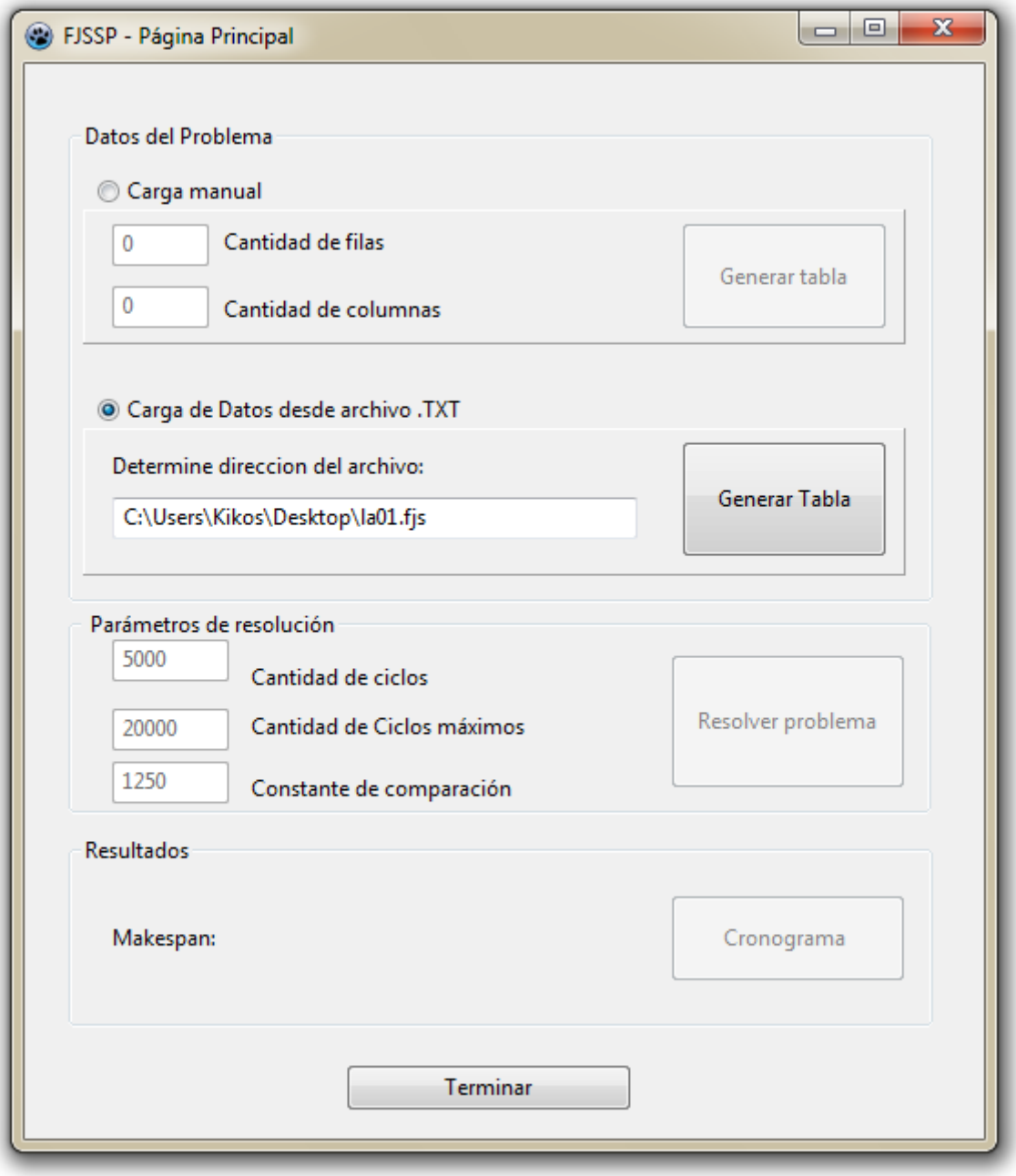


Ilustración 11: "Página Principal"

Tal como puede observarse en la Ilustración 11, el programa está dividido en tres partes principales: ingreso de datos, ajuste de parámetros de resolución y ejecución, y muestra de resultados. Los mismos, a excepción del primero, se encuentran deshabilitados al inicio del programa, y se van liberando

automáticamente a medida que se cumplimenta con las actividades solicitadas para cada etapa. Seguidamente se explicará por separado en qué consiste cada una de ellas.

4.2 Ingreso de Datos

En la pantalla del sistema, la primera opción habilitada solicita al usuario el ingreso de la información necesaria para identificar el problema a resolver. Para ello se ofrecen dos modalidades de ingreso distintas, la carga manual y mediante archivos del tipo ".txt".

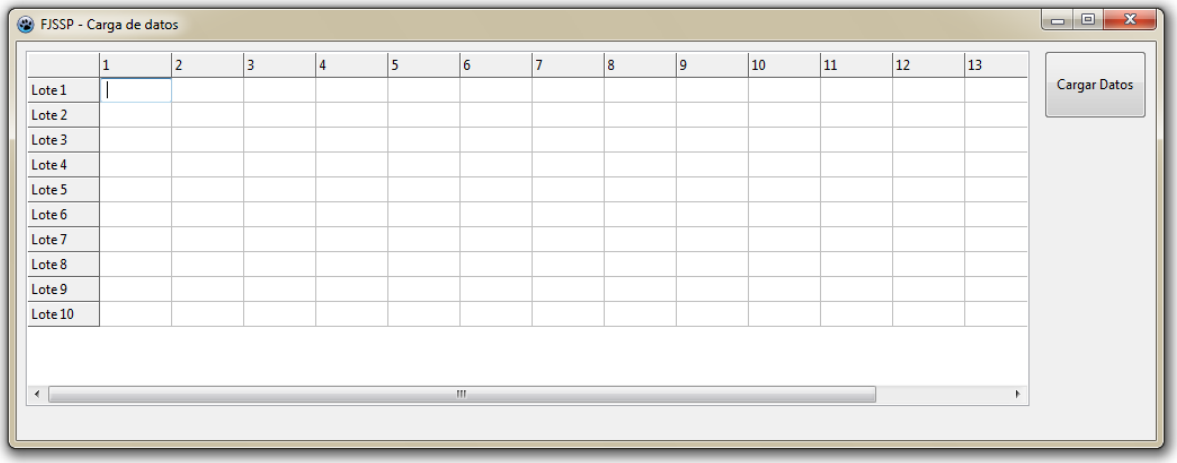


Ilustración 12: "Tabla manual"

La primera alternativa consiste en generar una tabla con la cantidad de filas y columnas que el usuario desee, las cuales representan, respectivamente, la cantidad de lotes a fabricar y la descripción de los procesos correspondientes. De seleccionar esta modalidad, y luego de especificar las dimensiones deseadas, se tendrá una tabla como la que se puede observar en la Ilustración 12. En dicha tabla el usuario podrá ingresar uno a uno los datos del problema, seleccionando la celda correspondiente.

Para que el programa sea capaz de interpretar estos datos, la disposición de la información deberá respetar un formato establecido. A continuación se detalla los lineamientos que deben seguirse (la explicación es acompañada por las ilustraciones 12 y 13):

- El programa lee la tabla de datos de arriba hacia abajo y de izquierda a derecha.
- Cada fila representará un lote de producción, estableciéndose en cada una de ellas la secuencia productiva del lote en cuestión, las operaciones necesarias, alternativas, tiempos, etc.
- En la primera celda de cada fila se indicará la cantidad de procesos que integran la secuencia productiva. En este ejemplo, este dato se encuentra recuadrado en rojo.
- A partir de la segunda fila se detalla para cada proceso la cantidad de alternativas. En el ejemplo estos datos se encuentran recuadrados en verde.
- Para cada alternativa se detallan las máquinas que pueden llevar adelante el proceso.
- Y finalmente, para cada máquina, se indica un tiempo de procesamiento. Cada pareja máquina-tiempo se encuentra recuadrada en azul.

El ejemplo numérico de la Ilustración 13, es acompañado por la Ilustración 14, la cual señaliza claramente lo que cada número representa. Se puede observar que el ejemplo respeta la lógica desarrollada anteriormente. Si al cargar los datos se registran incongruencias, el programa no será capaz de resolver el problema.

	1	2	3	4	5	6	7	8	9	10	11	12
Lote 1	3	2	3	12	4	10	1	2	6	1	1	9

Ilustración 13: "Problema ejemplo: disposición"

	1	2	3	4	5	6	7	8	9	10	11	12
Lote 1	Procesos	Alternativa	Máquina	Tiempo	Máquina	Tiempo	Alternativa	Maquina	Tiempo	Alternativa	Maquina	Tiempo

Ilustración 14: "Disposición de los datos"

En la Ilustración 15 se visualiza el ejemplo en un cuadro, que resulta de ayuda para interpretar la configuración recientemente descripta.

L OTE 1					
Proceso 1		**Proceso 2**		**Proceso 3**	
Alternativa	**Tiempo**	**Alternativa**	**Tiempo**	**Alternativa**	**Tiempo**
Máquina 3	12	Máquina 2	6	Máquina 1	9
Máquina 4	10	-	-	-	-

Ilustración 15: "Problema ejemplo"

La segunda modalidad para el ingreso de datos es a través de un archivo del tipo ".txt", en el cual se encuentren cargados con anterioridad los datos del problema. Esto de realizará ingresando en la celda correspondiente la dirección en la cual se encuentra el archivo, tal cual puede verse en la Ilustración 16.

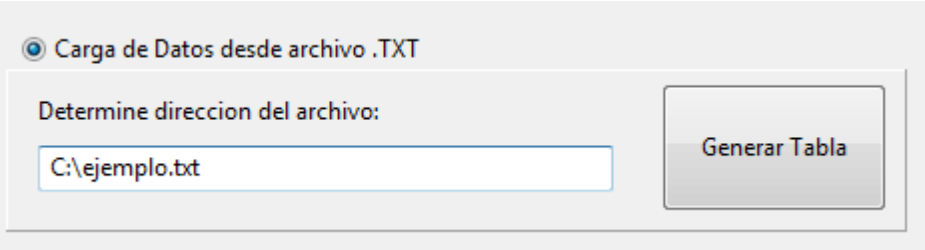


Ilustración 16: "Carga de datos desde archivo .txt"

Para leer los datos que se encuentran dentro del archivo de texto el programa construye, de forma automática, una tabla con los datos provenientes del archivo. Cada fila de éste corresponde a una fila de la tabla y a su vez, a un lote de producción en particular. El programa asigna 4 caracteres para cada dato, contemplando éstos valores numéricos y espacios en blanco. De esta forma, si se desea ingresar, por ejemplo, un dato de un solo dígito, supongamos "x", los cuatro

caracteres estarán compuestos por el número “x” seguido de tres espacios en blanco. A continuación de éstos seguiría el siguiente dato, y así sucesivamente. Esto implica que como máximo podrán ingresarse valores hasta 9999, teniendo que recurrir en caso de existir valores mayores a éste, al ingreso mediante tabla (ingreso manual).

4.3 Establecimiento de parámetros de resolución

El establecimiento de los parámetros de resolución tiene un carácter particular en la resolución de la problemática enfrentada, dado que es el único paso en donde se requiere que el usuario tome decisiones que impactarán en la solución encontrada.

La sección de “Parámetros de resolución” del programa incluye tres variables a definir. Las dos primeras apuntan al establecimiento de la cantidad de ciclos o reiteraciones que el programa deberá realizar antes de modificar la variable de comparación, mientras que la tercera es la constate de comparación misma. En la Ilustración 17 se visualiza la sección a la que se hace referencia.

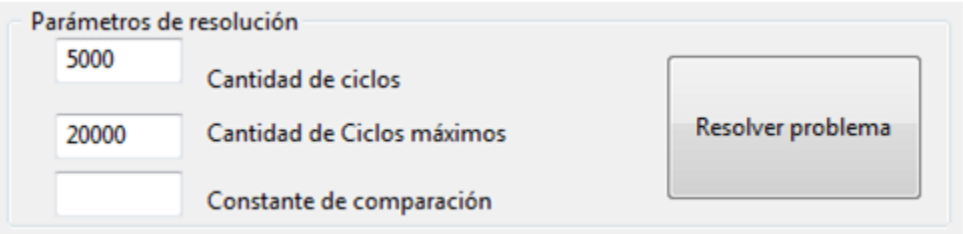


Ilustración 17: "Parámetros de resolución"

La “Cantidad de ciclos”, como su nombre lo indica, establece la cantidad inicial de ciclos que el programa realizará, es decir, cuántas modificaciones y evaluaciones se efectuarán antes de disminuir el valor de la constaste de comparación. Un valor pequeño de este parámetro implicará que el “recalentamiento” requerido para escapar de la solución inicial sea poco eficiente,

mientras que un valor alto generará un tiempo de procesamiento innecesariamente elevado.

Como el procedimiento que realiza la heurística elegida, desarrollada en la sección 2.10 Recocido simulado, incrementa la cantidad de ciclos a medida que el proceso de resolución avanza, resulta necesario establecer un valor tope que evite que el valor de los ciclos resulte inadecuadamente elevado incrementado el tiempo de procesamiento a niveles indeseados. Por tal motivo el segundo parámetro que se pide establecer es el valor máximo de ciclos que se realizarán en la resolución.

El último parámetro que debe establecerse es el valor de la constante de comparación, llamada *temperatura inicial* por algunos autores (Diaz, 1996). Este valor influye en la cantidad de soluciones inferiores que se aceptarán a lo largo del proceso de resolución. Lo que se debe tener en cuenta aquí es que un valor pequeño no permitirá escapar de mínimos locales, mientras que un valor elevado demorará la convergencia del programa a una solución óptima o próxima a la misma.

4.4 Resultado y salida del programa

Una vez resuelto el problema ingresado en el programa, el mismo informa el valor de la solución alcanzada en el área de resultado (Ilustración 18). Este valor, como se explicó anteriormente, corresponde al tiempo necesario para realizar todas las actividades, comprendido entre el momento en el cual se inician las actividades del primer proceso del primer lote hasta finalizada la última tarea del último lote.

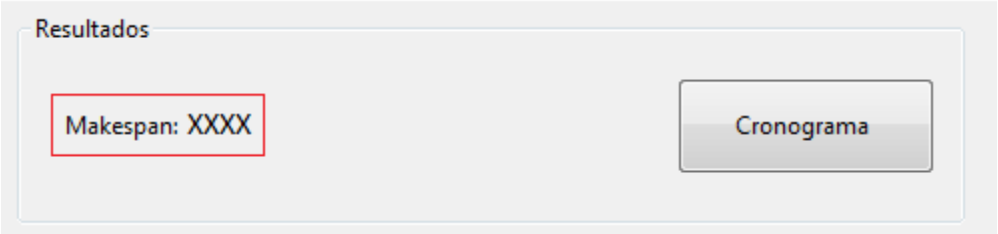
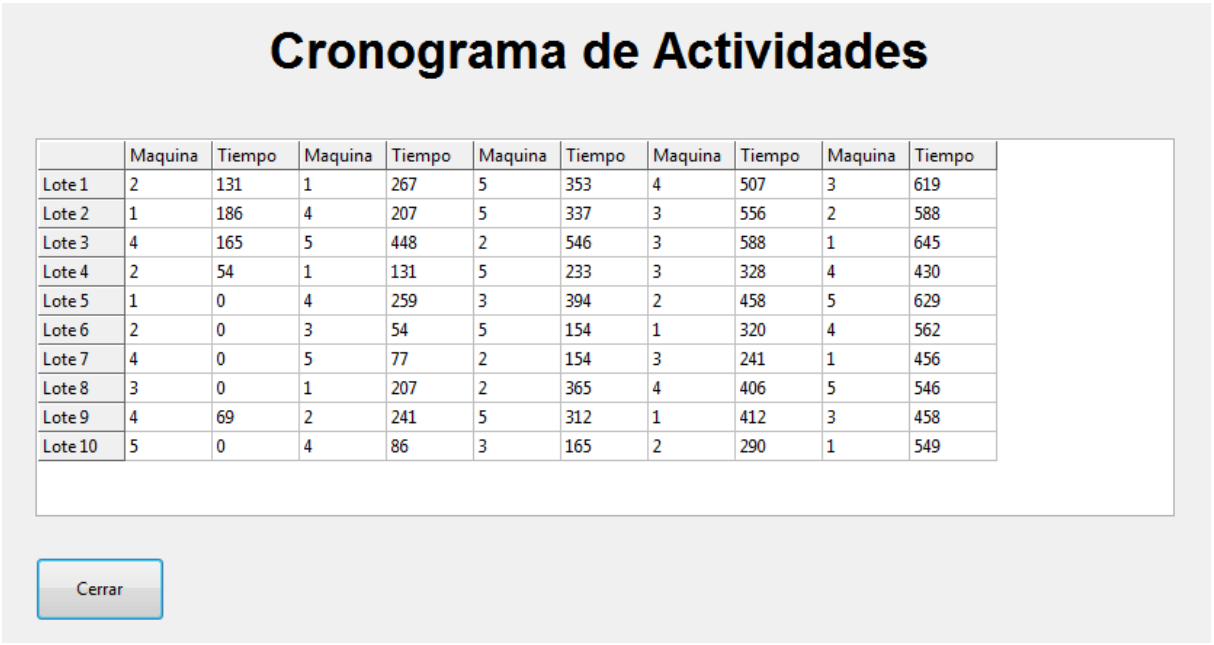


Ilustración 18: "Valor de la solución hallada"

La configuración de la solución hallada, que es en definitiva la información necesaria para realizar la planificación, se encuentra representada en la tabla de “Cronograma de actividades”. A dicha tabla es posible acceder, una vez resuelto el problema, haciendo click en el botón “Cronograma” que se encuentra ubicado en el área de resultados, como figura en la Ilustración 18.

El cronograma que genera el programa es una descripción de actividades para cada uno de los lotes, indicando para cada uno de ellos el comienzo de las actividades y el centro de trabajo, señalizado como “Máquina”, responsable de llevar a cabo cada uno de los procesos. En la Ilustración 19 se visualiza un posible cronograma de actividades para el problema LA01 de la serie de Lawrence para la modalidad JSSP.



	Maquina	Tiempo	Maquina	Tiempo	Maquina	Tiempo	Maquina	Tiempo	Maquina	Tiempo
Lote 1	2	131	1	267	5	353	4	507	3	619
Lote 2	1	186	4	207	5	337	3	556	2	588
Lote 3	4	165	5	448	2	546	3	588	1	645
Lote 4	2	54	1	131	5	233	3	328	4	430
Lote 5	1	0	4	259	3	394	2	458	5	629
Lote 6	2	0	3	54	5	154	1	320	4	562
Lote 7	4	0	5	77	2	154	3	241	1	456
Lote 8	3	0	1	207	2	365	4	406	5	546
Lote 9	4	69	2	241	5	312	1	412	3	458
Lote 10	5	0	4	86	3	165	2	290	1	549

Ilustración 19: "Cronograma de Actividades"

En el ejemplo visualizado en la ilustración superior se puede observar, para cada uno de los 10 lotes a fabricar y para cada una de las 5 tareas que deben realizarse, la máquina que debe llevar adelante la tarea y el momento en el tiempo en el cual debe comenzar a trabajarse.

Así, por ejemplo, se ve que sólo los lotes 5,6,7,8 y 10 son los que comienzan a procesarse en el momento 0, dado que únicamente hay 5 máquinas que pueden realizar los trabajos, y cada una de ellas se ocupa con dichos lotes. También puede observarse que la última tarea que comienza a procesarse es la correspondiente al lote 3, ya que posee la fecha de inicio más tardía (645).

Para visualizar mejor y comprender completamente el ejemplo, se presenta a continuación un diagrama de Gantt, ilustrando los resultados del cronograma obtenido mediante el programa. (Ilustración 20)

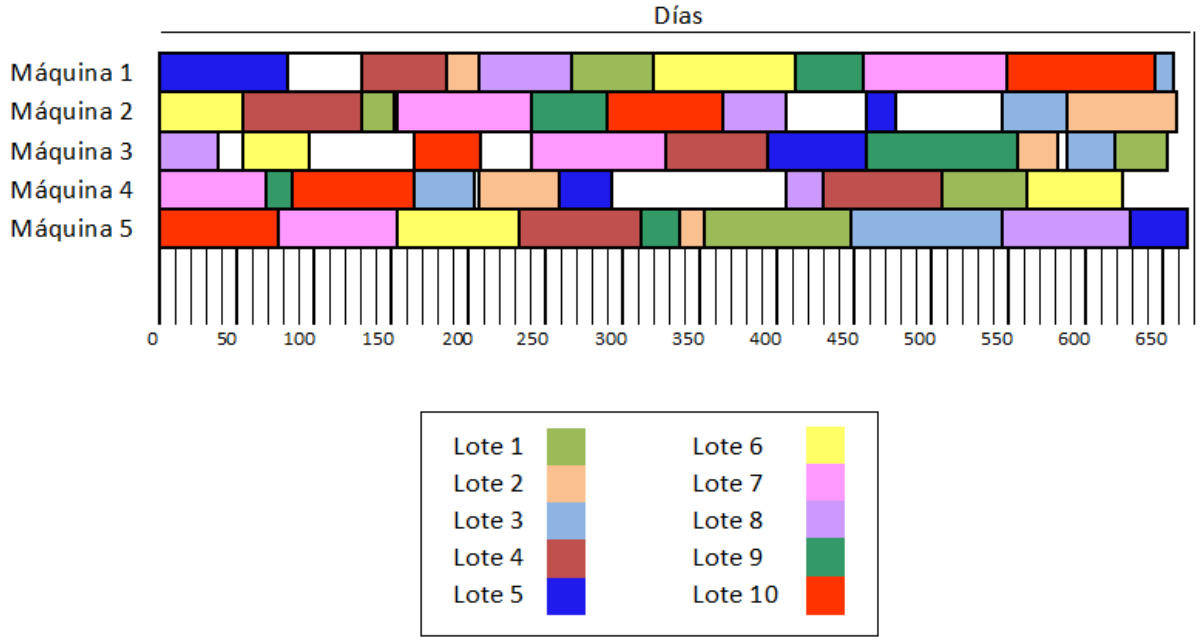


Ilustración 20: "Diagrama de Gantt"

4.5 Evaluación de resultados

Al resolver problemas del tipo metaheurísticos, es importante medir la calidad de los resultados obtenidos ya que la optimalidad de la solución no está garantizada. Para ello es necesario contrastar los resultados del algoritmo desarrollado frente a otros algoritmos existentes o los resultados teóricos correspondientes.

Existen diversos procedimientos que permiten llevar adelante esta comparación, entre los más habituales se encuentran (Duarte Muñoz, 2007):

- Comparación con la solución óptima, si se conoce, o con la mejor solución disponible
- Comparación con una cota
- Comparación con un método exacto truncado
- Comparación con los resultados de otras heurísticas
- Análisis del peor caso

En este trabajo se optó por la primera, ya que se cuenta con una serie de problemas de prueba para los cuales se conocen sus resultados (no necesariamente los óptimos, sino los mejores conocidos hasta el momento).

4.6 Problemas de prueba

Para evaluar la eficacia del software, se recurrió a la base de datos OR-Library (Beasley, 1990), la cual contiene una gran variedad de problemas clásicos de prueba para distintas aplicaciones en Investigación Operativa, entre ellas, la secuenciación de operaciones del tipo JSSP y FJSSP. Estos problemas son ampliamente utilizados en la bibliografía, ya que abarcan un amplio espectro en cuanto a dimensiones y complejidad, y permiten al mismo tiempo contrastar certeramente los resultados, ya que cuentan con sus correspondientes soluciones óptimas o en su defecto, las mejores conocidas hasta el momento (Jin, 2005).

De dicha base de datos se seleccionaron las series de problemas para JSSP y FJSSP denominados LA01, LA02,…, LA40, los cuales fueron desarrollados originalmente por Lawrence (1984). Cada uno de estos problemas está representado por un conjunto de máquinas y operaciones para las cuales se conocen sus tiempos de procesamiento. Los mismos abarcan instancias de 5 problemas cada una, en las siguientes dimensiones: 5x10, 5x15, 5x20, 10x10, 10x15, 10x20, 10x30 y 15x15.

4.7 Resultados del problema

En las tablas que se presentarán en esta sección, se muestran los resultados obtenidos con el programa desarrollado, para la resolución de los problemas de Lawrence.

Los mismos se encuentran agrupados por tamaño del problema y se indica, en cada caso, el mejor valor de makespan conocido hasta el momento (Cmáx), el promedio y los valores extremos del programa, la desviación obtenida (en base al promedio) y el error porcentual.

Se decidió indicar el error porcentual y no el absoluto, puesto que cada problema posee una configuración diferente y a medida que lo cantidad de procesos y lotes aumenta también lo suele hacer el valor makespan, con lo cual el error porcentual resulta la medida más adecuada para conocer el desempeño del programa desarrollado. Este mismo se calcula como el cociente de la diferencia entre la mejor solución encontrada por el programa y el Cmáx, con el Cmáx.

En la resolución de los problemas se tomaron en cuenta los siguientes criterios para el establecimiento de los parámetros:

- El valor máximo de ciclos para todos los problemas es 20000; este valor implica que el programa no terminará su ejecución hasta que no se hayan rechazado dicha cantidad de posibles nuevas soluciones para una misma constante de comparación.
- La cantidad inicial de ciclos para todos los problemas fue establecido en función del valor máximo de ciclos, tomando el 25% de dicho valor, es decir, 5000 ciclos. Se consideró que este valor es suficientemente grande como para que el programa pueda “recalentar” la solución y escapar de los mínimos locales iniciales en las primeras rondas de ciclos sin que demore excesivamente el tiempo de procesamiento.
- La constante de comparación se estableció para cada problema en particular. De todas formas, para todos los problemas se respetó el mismo criterio, considerándose apropiado que una transición a una

solución un 10% inferior, con respecto al costo de las distribuciones, otorgarle una probabilidad del 20% de ser aceptada.

Como el valor de las distribuciones al inicio de ejecución del programa puede variar se determinó tomar como valor aproximado de los costos de las distribuciones como el doble del valor de la mejor solución encontrada hasta la fecha.

Este método presentado en forma similar por Adenso Díaz (1996), utiliza la siguiente fórmula:

$$To = \frac{\mu}{-\ln(\emptyset)} C(Sact)$$

Donde:

$\emptyset$: Probabilidad de aceptar la solución inferior. En nuestro caso 20%

μ : Diferencia porcentual entre la solucion oroginal y la nueva solución inferior. En nuestro caso 10%.

C(Sact) : Costo de la solución original. En nuestro caso 2 veces el valor de Cmáx.

Obteniéndose finalmente:

$$To = 0{,}124 \text{ x } Cmáx$$

Para la obtención de los valores reflejados en las tablas se ejecutó el programa diez veces con cada problema. Los resultados de cada ejecución pueden verse en el Anexo B.

4.7.1 Resultados para el JSSP – Tablas

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA01	666	666	666	666	0,00	-
LA02	655	655	655	655	0,00	-
LA03	597	601,8	597	617	7,44	-
LA04	590	591,9	590	597	2,33	-
LA05	593	593	593	593	0,00	-

Tabla 3: "Resultados problemas JSSP 5x10"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA06	926	926	926	926	0,00	-
LA07	890	890	890	890	0,00	-
LA08	863	863	863	863	0,00	-
LA09	951	951	951	951	0,00	-
LA10	958	958	958	958	0,00	-

Tabla 4: "Resultados problemas JSSP 5x15 "

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA11	1222	1222	1222	1222	0,00	-
LA12	1039	1039	1039	1039	0,00	-
LA13	1150	1150	1150	1150	0,00	-
LA14	1292	1292	1292	1292	0,00	-
LA15	1207	1207	1207	1207	0,00	-

Tabla 5: "Resultados problemas JSSP 5x20"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA16	945	984,8	947	999	15,42	0,21%
LA17	784	784,7	784	785	0,48	-
LA18	848	851,6	848	855	3,20	-
LA19	842	862,1	852	873	6,49	1,19%
LA20	902	917	902	941	13,15	-

Tabla 6: "Resultados problemas JSSP 10x10"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA21	1046	1084,9	1059	1115	19,48	1,24%
LA22	927	940,5	927	967	10,99	-
LA23	1032	1032	1032	1032	0,00	-
LA24	935	960,5	950	969	5,17	1,60%
LA25	977	1006	992	1030	11,49	1,54%

Tabla 7: "Resultados problemas JSSP 10x15"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA26	1218	1220,2	1218	1231	4,73	-
LA27	1235	1257,7	1246	1269	7,56	0,89%
LA28	1216	1229,8	1216	1259	15,15	-
LA29	1160	1198,4	1172	1219	16,81	1,03%
LA30	1355	1355	1355	1355	0,00	-

Tabla 8: " Resultados problemas JSSP 10x20"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA31	1784	1784	1784	1784	0,00	-
LA32	1850	1850	1850	1850	0,00	-
LA33	1719	1719	1719	1719	0,00	-
LA34	1721	1721	1721	1721	0,00	-
LA35	1888	1888	1888	1888	0,00	-

Tabla 9: "Resultados problemas JSSP 10x30"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA36	1272	1316,4	1278	1334	19,33	0,47%
LA37	1399	1433,6	1401	1477	24,94	0,14%
LA38	1196	1247,9	1209	1281	26,43	1,09%
LA39	1237	1281,8	1253	1331	25,13	1,29%
LA40	1229	1251,7	1233	1301	23,31	0,33%

Tabla 10: "Resultados problemas JSSP 15x15"

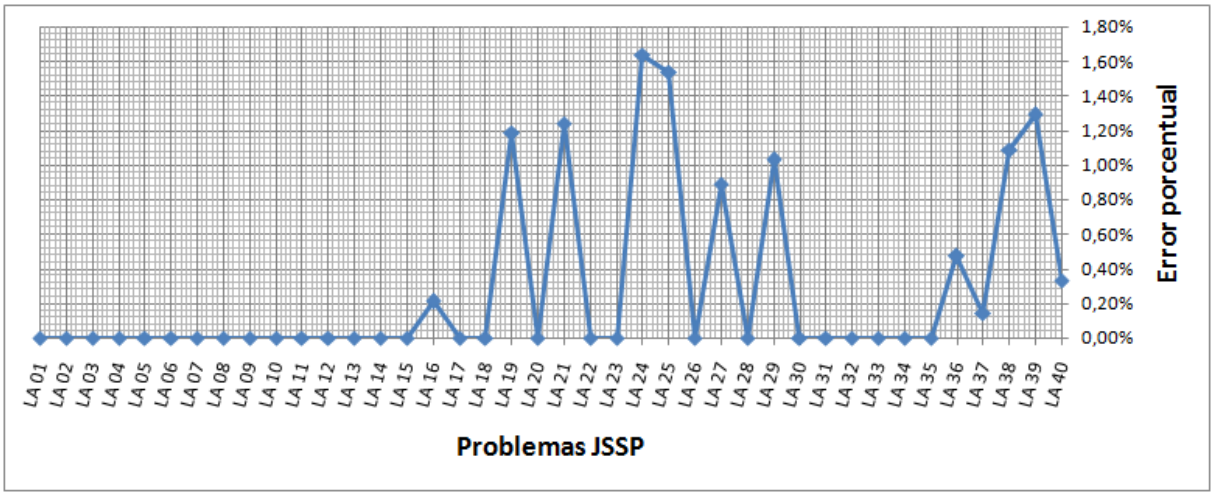


Ilustración 21: "Error porcentual problemas JSSP"

La ilustración 18 resume los resultados obtenidos para los problemas del tipo JSSP. Puede observarse que, en general, no existe ningún patrón en los valores asociado a las dimensiones del problema, y que, en ningún caso, el error supera el 2%.

Los primeros 15 problemas de la serie, tal como se observa, son resueltos perfectamente, arribando a la solución óptima en todos los casos. A partir del problema número 16 comienza a aparecer un error porcentual, siempre menor al 2%, que varía problema a problema, a excepción de los de la serie de 10x30 (LA31 a LA35), que no presentan diferencias con el valor óptimo.

Si bien resultaría lógico que el error se incremente a medida que las dimensiones del problema sean mayores (puesto que aumenta la cantidad de combinaciones que implica la resolución del problema), puede verse que esto no sucede. Un claro ejemplo es lo que ocurre, como se mencionara anteriormente, con los problemas de 10x30, cuyo error porcentual es cero.

Esto puede explicarse si se tienen en cuenta otras variables asociadas a la naturaleza del problema, ya que la complejidad del mismo no sólo depende de sus dimensiones. Los tiempos de procesamiento de cada trabajo en las diferentes máquinas, por ejemplo, que varían de un problema a otro, pueden generar que,

independientemente del tamaño del problema, el campo de soluciones óptimas o cercanas a ésta, sea más o menos amplio, facilitando o no que el programa “caiga” en una mejor solución.

De todas formas, en base a los resultados obtenidos, y teniendo en cuenta que se han evaluado problemas de variadas dimensiones y características, puede concluirse que el programa resulta satisfactorio para la resolución de este tipo de problemas.

4.7.2 Resultados para el FJSSP - Tablas

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA01	570	575	572	577	1,56	0,35%
LA02	529	532,8	531	535	1,03	0,38%
LA03	477	483	479	489	2,83	0,42%
LA04	502	503,8	502	505	0,92	-
LA05	457	462,9	459	466	2,38	0,44%

Tabla 11: "Resultados problemas FJSSP 5x10"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA06	799	802,2	800	804	1,14	0,13%
LA07	749	753,8	752	756	1,14	0,40%
LA08	765	767,1	766	768	0,57	0,13%
LA09	853	856,8	856	858	0,92	0,35%
LA10	804	805,9	805	806	0,32	0,12%

Tabla 12: "Resultados problemas FJJSP 5x15"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA11	1071	1072,9	1072	1074	0,57	0,09%
LA12	936	936,8	936	938	0,63	-
LA13	1038	1040,2	1039	1041	0,92	0,10%
LA14	1070	1071,6	1071	1072	0,52	0,09%
LA15	1089	1090,7	1090	1091	0,48	0,09%

Tabla 13: "Resultados problemas FJSSP 5x20"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA16	717	717	717	717	0	-
LA17	646	646	646	646	0	-
LA18	663	663	663	663	0	-
LA19	617	617	617	617	0	-
LA20	756	756	756	756	0	-

Tabla 14: "Resultados problemas FJSSP 10x10"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA21	806	816,2	812	820	2,62	0,74%
LA22	739	745,5	739	750	3,37	-
LA23	815	823	819	830	3,33	0,49%
LA24	777	784,7	781	790	3,23	0,51%
LA25	756	766,3	763	761	3,02	0,93%

Tabla 15: "Resultados problemas FJSSP 10x15"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA26	1054	1057	1055	1059	1,41	0,09%
LA27	1085	1087,8	1086	1090	1,62	0,09%
LA28	1070	1073	1071	1075	1,70	0,09%
LA29	994	997,7	996	1000	1,34	0,20%
LA30	1069	1074,3	1072	1079	2,36	0,28%

Tabla 16: "Resultados problemas FJSSP 10x20"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA31	1520	1521,7	5121	5123	0,67	0,07%
LA32	1658	1658,8	1658	1661	0,92	-
LA33	1497	1498,9	1498	1499	0,32	0,07%
LA34	1535	1536,1	1535	1537	0,57	-
LA35	1549	1550,5	1550	1552	0,71	0,06%

Tabla 17: Resultados problemas FJSSP 10x30"

	Cmáx	Promedio	Mín	Máx	DesvEst	% Error
LA36	948	948	948	948	0	-
LA37	986	986	986	986	0	-
LA38	943	943	943	943	0	-
LA39	922	922	922	922	0	-
LA40	955	955	955	955	0	-

Tabla 18: "Resultados problemas FJSSP 15x15"

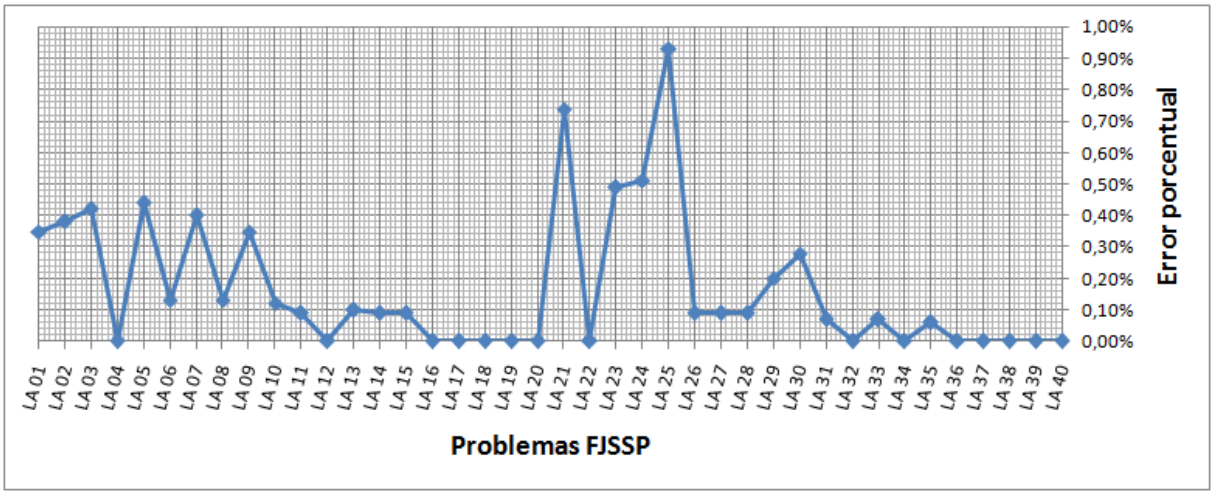


Ilustración 22: "Error porcentual problemas FJSSP"

En la ilustración 17, se resumen los resultados obtenidos para los problemas tipo FJSSP. Se puede observar que, al igual que en lo ocurrido para los JSSP, no existe un patrón de error asociado directamente a las dimensiones del problema. Así mismo, se observa también que el error continúa siendo menor al 2%.

En este tipo de problemas entra en juego una nueva variable que influye en la complejidad de los mismos, que es el número de alternativas de ejecución para cada tarea, es decir, la posibilidad de que una misma tarea pueda ser efectuada en diferentes máquinas. Este factor aumenta exponencialmente el número de combinaciones existentes para la resolución del problema, lo cual, en principio, tendería a dificultar el acceso del programa a la solución óptima. Un claro ejemplo de ello es lo que ocurre con los problemas de 5x10 (LA01 a LA05) o de 5x15 (LA06 a LA10), en donde puede observarse que, mientras en el JSSP el error porcentual era cero, para el FJSSP comienzan a haber dificultades para hallar el valor óptimo.

Así mismo, no deja de estar presente el factor asociado a los tiempos de procesamiento de cada tarea, que al igual que en el caso anterior pueden influir en la complejidad del problema. Esto podría explicar que problemas como los de dimensiones 10x10 o 15x15 (LA16 a LA25), tengan en principio un campo de soluciones más amplio (por aumentar el número de combinaciones en FJSSP), pero que, dada la combinación arbitraria de tiempos de los distintos procesos, este resulte reducirse, siendo más simple para el programa acercarse a la solución óptima.

La Ilustración 23, refleja las diferencias comentadas entre los resultados obtenidos para los problemas JSSP y FJSSP:

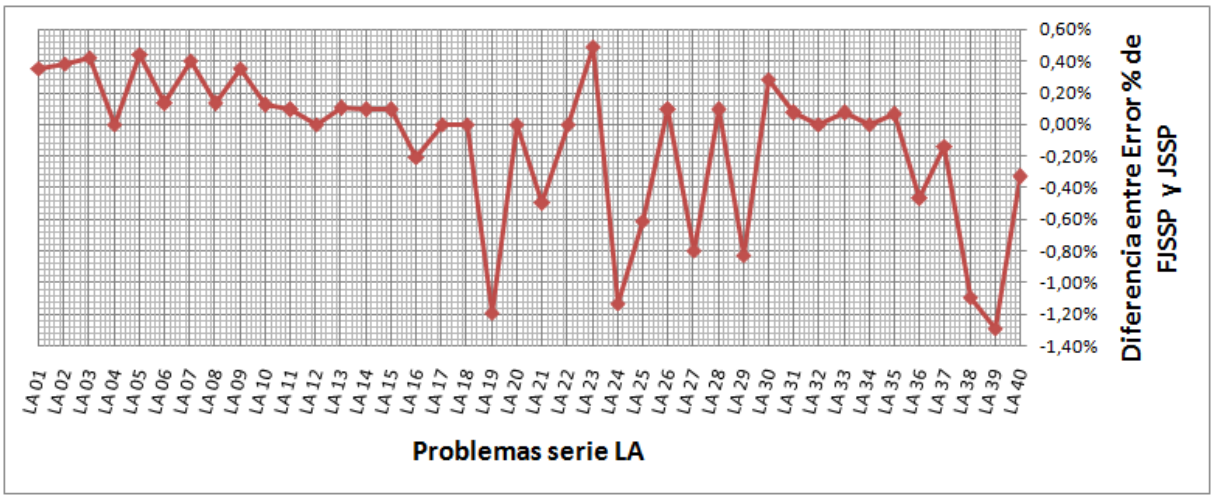


Ilustración 23: "Diferencia de errores porcentuales en problemas serie LA"

Teniendo en cuenta lo anteriormente desarrollado se puede afirmar que la capacidad del programa para encontrar una solución válida para secuenciar la producción se ve condicionada por las características propias del problema. Por tal motivo no se puede garantizar el hallazgo de una solución óptima para todo problema productivo afrontado.

Sin embargo, dado que se evaluaron problemas de diversos tamaños y niveles de complejidad, y en ningún caso el error porcentual superó los dos puntos, puede decirse que el programa es capaz de resolver este tipo de problemas con un nivel elevado de eficacia, constituyendo una herramienta confiable para tal finalidad.

CONCLUSIONES

Antes de establecer las conclusiones, resulta importante recordar los objetivos planteados en el principio de este trabajo. Los mismos consistían, por un lado, en desarrollar un programa capaz de resolver el problema de secuenciación, que devolviese resultados aceptables a los problemas enunciados, y por otro, lograr que este programa le sirviese a la empresa como herramienta de apoyo para la toma de decisiones.

No sería correcto afirmar que la secuencia productiva que el programa genera debe ser tomada como el programa final productivo. Esto es así dado la cantidad de imponderables que cualquier empresa productiva genera, entre ellos el ausentismo por enfermedad, la rotura o desperfectos de maquinaria, los reprocesos, entre otros. De todas formas el cronograma resulta más que útil para prever los caminos críticos y destinar los recursos necesarios para que estos eventos se minimicen lo más posible.

Para la medición del primer objetivo, se estableció como criterio de éxito un error no mayor al 2% en el resultado obtenido con el programa respecto a los valores óptimos de cada problema. Para ello, se resolvieron mediante el programa los cuarenta problemas de Lawrence para JSSP y FJSSP, obteniéndose los resultados mostrados en el apartado anterior. Los mismos demuestran que los valores óptimos de makespan hallados por el programa no difieren en ningún caso en más de un 2% respecto a los valores conocidos. Dado el amplio espectro que estos problemas abarcan en cuanto a dimensiones y complejidad, y teniendo en cuenta los resultados obtenidos, puede esperarse que el programa resuelva este tipo de problemas en general de manera aceptable.

Hay que aclarar, que si bien no estaba contemplado dentro de los objetivos el análisis de los tiempos de ejecución del programa (por tratarse de un tema de complejidad y análisis computacional más profundo, que escapa al alcance de este trabajo), los tiempos de resolución del programa han resultado satisfactorios. Con

ello lo que se quiere expresar es que el programa ha ejecutado cada problema en un tiempo no mayor a 10 minutos, lo cual resulta razonable para los propósitos planteados.

En cuanto al segundo objetivo, se trató de elaborar el programa de forma tal que resultara aplicable al caso de la empresa estudiada y que fuese fácil de implementar por la misma. Se buscó que las salidas del programa fueran simples de interpretar y útiles a la hora de planificar. El resultado obtenido cumple satisfactoriamente con todos estos puntos. Puede verse que, por un lado, el programa ofrece dos alternativas de ingreso de datos, ambas muy sencillas, y que requieren solamente del conocimiento de las máquinas y procesos que intervienen en el esquema productivo, como así también de sus respectivos tiempos de ejecución. Por otra parte, la salida del programa resulta también muy simple; los valores obtenidos pueden utilizarse para confeccionar un diagrama de Gantt en donde se visualicen fácilmente las secuencias de producción de cada lote y del proceso productivo en general. Esta secuenciación, la cual debe ser considerada como una guía al momento de planificar, permite a la empresa tomar decisiones acerca de cómo organizar sus recursos, qué políticas de producción aplicar, prever cuáles serán sus tiempos de producción totales en función de sus objetivos y los recursos disponibles, etc.

5.1 Pasos a seguir

El trabajo aquí desarrollado constituye tan solo una parte de un proyecto mayor, que por ser de mayores dimensiones y escapar al propósito de un proyecto final, no ha sido desarrollado. Con la intención de dar continuidad a este trabajo podría plantearse como objetivo a futuro realizar la implementación del programa, lo cual implica:

- en primer lugar, realizar un relevamiento de los procesos involucrados en el sistema productivo, así como la toma de tiempos y el estudio de los mismos,

para determinar los valores representativos de cada taller en las distintas actividades y conformar una base de datos;

- acoplar el programa desarrollado al software utilizado actualmente por la empresa;
- ejecutar el programa utilizando toda la información recabada, analizar los resultados y tomar las decisiones de programación correspondientes en función ello;
- por último, transcurrido el período de producción programado, estudiar el impacto de la implementación del programa y utilizar el feedback como herramienta para mejorar el sistema.

BIBLIOGRAFÍA

- Beasley, J.E. (1990). *OR-Library: distributing test problems by electronic mail*. Journal of the Operational Research Society, 41(11):1069-1072.
- Cortés Rivera, D. (2004). *Un Sistema Inmune Artifical para resolver el problema del Job Shop Scheduling*. Tesis para obtener el grado de Maestro en Ciencias. CINVESTAV. México.
- Diaz, A. et al. (1996). *Optimización heurística y Redes Neuronales*. Editorial Paraninfo.
- Domínguez Machuca, J. A. et al. (1997). *Dirección de operaciones – Aspectos tácticos y operativos en la producción y los servicios*. España: McGraw-Hill.
- Duarte Muñoz, A. et al. (2007). *Metaheurísticas*. España: Los Autores.
- Free Pascal Lazarus Project. 2010. *About Lazarus*. Disponible en: http*://www.lazarus.freepascal.org/index.php?PHPSESSID=0d899ab9c69fafa0621852a1fe0cd809&page=7*. 25 Abril 2010
- Jin, Y. (2005). *Knowledge incorporation in evolutionary computation*. Springer.
- Kuik, R. y Salomon, M. (1990*). Multi-item Lot-sizing Problem: Evaluation of a Simulated Annealing Heuristic*. European Journal of Operational Research, Vol 45, pp 25-37.
- Lawrence, S. (1984). *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement).* Graduate School of Industrial Administration, Carnegie-Mellon University. Pittsburgh, Pennsylvania, Estados Unidos.

- López de Haro, S. et al. (2004). *Secuenciación de tareas mediante metaheurísticos*. VIII Congreso de Ingeniería de Organización. Léganes, España.
- Nahmias, S. (2007). *Análisis de la producción y las operaciones*. MacGraw-Hill.
- Peña, V. y Zulmelzu, L (2006). *Job Shop Scheduling Problem: Minimización de Tardanza Máxima con RFL*. Departamento de Informática - Universidad Técnica Federico Santa María. Chile.
- Pinedo, M. L., (2008). *Scheduling: Theory, algorithms and systems*. Springer.
- Salazar Gonzalez, J.J. (2001). *Programación matemática*. Diaz de Santos.
- Valenzuela, M. (2004). *Recocido Simulado: Algoritmo Básico*. Apuntes de cátedra, Departamento de Mecatrónica y Automatización, Tecnológico de Monterrey. Moterrey, México.
- Vicentini, F. y Puddu, S. (2003). *Algoritmos heurísticos y el problema de job shop scheduling.* Facultad de Ciencias Exactas y Natutales - UBA. Buenos Aires, Argentina.

ANEXO A: Optimización Combinatoria

Algoritmos Genéticos (AG)

Este modelo, basado en la teoría de la selección natural, es un método heurístico en donde los individuos (soluciones) tratan de optimizar su existencia mediante la recombinación y mutación de genes. Tienen la gran ventaja de que pueden emplearse para buscar soluciones en ámbitos tan diversos como el análisis de imágenes, el procesamiento de señales, el diseño de robots autónomos y la programación automática, siendo también posible su uso en la resolución de problemas combinatorios como el JSSP.

Otra de las ventajas de los AG es la robustez, al ser éstos capaces de obtener soluciones subóptimas aún en los casos en los que la solución óptima es muy difícil de encontrar.

Aunque desde su primera aparición, en el libro *Adaptation in Natural and Artificial Systems* (Holland, 1975), se han creado muchas variantes del modelo, todas ellas respetan un mismo principio: un conjunto de individuos o población que tienen dos representaciones denominadas *fenotipo* y *genotipo*, siendo la primera una solución potencial al problema que se desea resolver y la segunda una codificación de dicha solución en forma de cromosoma. Un *cromosoma* está formado por una secuencia de *genes* que representan características hereditarias.

Los cromosomas de una población *evolucionan* en sucesivas iteraciones, denominadas *generaciones*. Los operadores *cruce* y *mutación* son mecanismos de recombinación genética capaces de generar nuevos cromosomas a partir de otros. Un operador de cruce sencillo podría intercambiar las secuencias que se formarían al cortar ambos cromosomas por un *punto de corte*.

Este método suele funcionar bien bajo una representación binaria, aunque para otras no suele ser el más indicado. La eficacia del algoritmo de cruce está directamente relacionada con su capacidad de recoger las características de los progenitores que los hacen *buenos* y recombinarlas para crear individuos *mejores*. El operador mutación es el encargado de modificar esporádicamente individuos

aleatorios para evitar que en el transcurso de las iteraciones el operador de cruce genere individuos cada vez más homogéneos y la población se concentre alrededor de un mínimo local.

Colonia de Hormigas (CH)

El método conocido como Colonia de Hormigas se encuentra basado en el comportamiento que posee una comunidad de hormigas para encontrar y aprovechar globalmente la comida encontrada por uno de sus miembros.

La idea es que las hormigas salen de forma dispersa del hormiguero, durante su recorrido dejan un ligero rastro mediante la segregación de una hormona llamadas feromona. Cuando una hormiga localiza la comida vuelve sobre sus pasos intensificando su rastro. Todas las hormigas que crucen este camino podrán encontrar la comida siguiendo el camino señalado. Con el paso del tiempo los aromas dejados en los caminos recorridos por las hormigas se van disipando.

En los problemas de optimización combinatoria se pretende generar soluciones aleatorias a partir de los caminos o decisiones que llevan a otras soluciones dadas, en este proceso resulta vital marcar con mayor intensidad aquellos caminos que llevan a las mejores soluciones.

Las variables que resultan más relevantes para hallar una buena solución al problema son la definición de la feromona y de los movimientos de las hormigas.

Búsqueda Tabú

La búsqueda tabú es un procedimiento metaheurístico diseñado para encontrar soluciones subóptimas en problemas de optimización combinatoria. Fue diseñado inicialmente por Glover (1990) y ha sido aplicado a varios problemas de optimización combinatoria como programación de horarios y rutas, y secuenciación de tareas.

Se trata de un algoritmo de búsqueda local, para cualquier solución se genera un conjunto de desplazamientos que definen su vecindario, y se selecciona el mejor de ellos para la siguiente iteración. Para evitar que el procedimiento iterativo entre en un bucle y que se quede estancado en óptimos locales, se mantiene en memoria

el histórico de la búsqueda. Se suele distinguir dos tipos de memoria: a *corto plazo* para los movimientos más recientes y a *largo plazo* para la parte de la búsqueda más distante.

La memoria a corto plazo está formada por una *lista tabú* que contiene información sobre las últimas soluciones visitadas, de modo que desplazamientos hacia soluciones vecinas que se correspondan con la lista tabú se prohíban. Típicamente, la lista tabú *T* consiste en una lista FILO (First In First Out) de longitud definida que contiene los últimos desplazamientos unitarios prohibidos, de modo que en cada iteración se introduce el desplazamiento inverso al realizado al principio de la lista *T* y se elimina el desplazamiento que se encuentra al final.

ANEXO B: Resultados del Programa

Tablas Completas del JSSP

	1	2	3	4	5	6	7	8	9	10
LA01	666	666	666	666	666	666	666	666	666	666
LA02	655	655	655	655	655	655	655	655	655	655
LA03	597	597	603	597	597	597	597	617	603	613
LA04	590	597	590	593	590	593	593	590	590	593
LA05	593	593	593	593	593	593	593	593	593	593
LA06	926	926	926	926	926	926	926	926	926	926
LA07	890	890	890	890	890	890	890	890	890	890
LA08	863	863	863	863	863	863	863	863	863	863
LA09	951	951	951	951	951	951	951	951	951	951
LA10	958	958	958	958	958	958	958	958	958	958
LA11	1222	1222	1222	1222	1222	1222	1222	1222	1222	1222
LA12	1039	1039	1039	1039	1039	1039	1039	1039	1039	1039
LA13	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150
LA14	1292	1292	1292	1292	1292	1292	1292	1292	1292	1292
LA15	1207	1207	1207	1207	1207	1207	1207	1207	1207	1207
LA16	980	994	991	995	979	980	947	999	984	999
LA17	785	784	785	785	785	784	785	785	785	784
LA18	855	853	848	848	853	853	855	848	855	848
LA19	870	861	861	861	852	867	861	854	861	873
LA20	922	922	902	941	926	922	902	907	902	924
LA21	1087	1109	1115	1088	1106	1070	1059	1073	1076	1066
LA22	935	935	967	927	937	939	949	944	937	935
LA23	1032	1032	1032	1032	1032	1032	1032	1032	1032	1032
LA24	950	962	959	959	969	967	959	962	959	959
LA25	994	1004	1019	1012	1003	992	1004	1001	1001	1030
LA26	1218	1218	1218	1218	1218	1218	1218	1227	1231	1218
LA27	1266	1256	1267	1250	1246	1256	1259	1254	1254	1269
LA28	1222	1223	1254	1216	1234	1222	1222	1216	1259	1230
LA29	1219	1194	1210	1192	1172	1218	1188	1187	1185	1219
LA30	1355	1355	1355	1355	1355	1355	1355	1355	1355	1355
LA31	1784	1784	1784	1784	1784	1784	1784	1784	1784	1784
LA32	1850	1850	1850	1850	1850	1850	1850	1850	1850	1850
LA33	1719	1719	1719	1719	1719	1719	1719	1719	1719	1719
LA34	1721	1721	1721	1721	1721	1721	1721	1721	1721	1721
LA35	1888	1888	1888	1888	1888	1888	1888	1888	1888	1888
LA36	1309	1331	1334	1331	1278	1319	1308	1331	1292	1331
LA37	1434	1447	1401	1401	1432	1427	1468	1477	1422	1427
LA38	1224	1244	1249	1281	1274	1211	1262	1248	1277	1209
LA39	1261	1266	1298	1254	1253	1331	1270	1289	1298	1298
LA40	1234	1265	1267	1240	1233	1272	1236	1235	1301	1234

Tablas Completas del FJSSP

	1	2	3	4	5	6	7	8	9	10
LA01	577	572	574	576	574	574	576	575	575	577
LA02	533	533	532	532	533	531	533	533	533	535
LA03	483	483	480	481	489	484	482	479	485	484
LA04	503	503	504	504	504	505	504	502	504	505
LA05	466	466	462	461	463	462	466	459	462	462
LA06	801	802	802	800	803	803	803	802	804	802
LA07	754	756	755	754	752	753	754	754	753	753
LA08	767	767	767	768	767	768	767	767	766	767
LA09	856	858	856	858	856	858	857	856	856	857
LA10	806	806	806	806	806	806	805	806	806	806
LA11	1072	1074	1073	1073	1073	1073	1073	1072	1073	1073
LA12	936	937	936	937	937	938	937	937	937	936
LA13	1041	1039	1041	1039	1040	1041	1041	1039	1040	1041
LA14	1071	1072	1071	1071	1071	1072	1072	1072	1072	1072
LA15	1091	1091	1090	1090	1091	1091	1090	1091	1091	1091
LA16	717	717	717	717	717	717	717	717	717	717
LA17	646	646	646	646	646	646	646	646	646	646
LA18	663	663	663	663	663	663	663	663	663	663
LA19	617	617	617	617	617	617	617	617	617	617
LA20	756	756	756	756	756	756	756	756	756	756
LA21	816	820	816	816	818	819	812	813	818	814
LA22	743	750	748	739	746	748	745	744	749	743
LA23	822	825	830	819	819	825	821	824	821	824
LA24	787	782	781	789	783	787	790	783	782	783
LA25	763	766	764	763	771	770	763	769	767	767
LA26	1059	1056	1056	1059	1056	1058	1057	1056	1058	1055
LA27	1088	1089	1089	1086	1090	1087	1086	1086	1087	1090
LA28	1071	1073	1075	1074	1072	1075	1073	1071	1071	1075
LA29	998	999	996	1000	997	998	997	997	996	999
LA30	1072	1077	1074	1076	1073	1074	1074	1072	1072	1079
LA31	1522	1523	1521	1522	1521	1521	1521	1522	1522	1522
LA32	1659	1659	1659	1659	1658	1658	1659	1661	1658	1658
LA33	1499	1499	1499	1498	1499	1499	1499	1499	1499	1499
LA34	1536	1536	1536	1537	1536	1536	1537	1535	1536	1536
LA35	1551	1551	1550	1550	1550	1550	1551	1550	1550	1552
LA36	948	948	948	948	948	948	948	948	948	948
LA37	986	986	986	986	986	986	986	986	986	986
LA38	943	943	943	943	943	943	943	943	943	943
LA39	922	922	922	922	922	922	922	922	922	922
LA40	955	955	955	955	955	955	955	955	955	955