CENTRO DE INVESTIGACIÓN Y ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE COMPUTACIÓN

Un Sistema Inmune Artificial para resolver el problema del Job Shop Scheduling

Tesis que presenta

Daniel Cortés Rivera

Para obtener el grado de

Maestro en Ciencias

En la especialidad de

Ingeniería Eléctrica
Opción Computación

Director de la tesis: Dr. Carlos Artemio Coello Coello

México D.F. Marzo 2004

Resumen

Los problemas de planificacíón de recursos son muy importantes en el mundo real. Se puede decir que se emplean en todos lados, de ahí la importancia de que éstos sean óptimos, de forma que se puedan ahorrar recursos que estén asociados al problema.

En esta tesis se propone un algoritmo inspirado en el sistema inmune artificial para resolver el problema denominado *Job Shop Scheduling* (JSSP). El algoritmo propuesto se basa en el principio de selección clonal y utiliza expansión clonal así como hipermutación somática. También se incorpora un mecanismo que permite explorar la vecindad de la mejor solución, con lo que se logra una mejora significativa de resultados.

El algoritmo propuesto utiliza una codificación basada en permutaciones con repeticiones. Esta es una representación muy sencilla y compacta, para la cual se diseñó un operador de mutación *ad-hoc*. El uso de esta codificación distingue a este trabajo de otros en los que tradicionalmente se han utilizado codificaciones basadas en grafos, permutaciones simples y llaves aleatorias, entre otras.

Se hace un análisis detallado de los resultados obtenidos, comparando el algoritmo propuesto en esta tesis con respecto a otras heurísticas reportadas en la literatura especializada. Se hacen comparaciones con respecto a un algoritmo génetico simple (AG), un algoritmo genético híbrido (HGA), un procedimiento de búsqueda miope aleatoria y adaptativa (GRASP) en dos versiones y una búsqueda tabú. Los resultados obtenidos indican que el algoritmo propuesto en esta tesis es una alternativa viable para resolver el JSSP, dada la calidad de las soluciones obtenidas, así como su costo computacional respectivo. Su único contendiente serio (de entre las heurísticas comparadas) es la búsqueda tabú.

Abstract

Scheduling problems are very important in the real world. It can be said that they are applicable everywhere and thus the importance of being able to produce optimal schedules, such that one can save the resources associated to the problem.

In this thesis, we propose an algorithm inspired on an artificial immune system to solve the so-called *Job Shop Scheduling Problem* (JSSP). The proposed approach is based on the clonal selection principle and it uses clonal expansion and somatic hypermutations. It also incorporates a mechanism that allows to explore the neighborhood of the best solution, with which a significant improvement of results is achieved.

The proposed algorithm uses an encoding based in permutations with repetition. This is a very simple and compact encoding for which an *ad-hoc* mutation operator was specifically designed. The use of such an encoding distinguishes this work from others in which, traditionally, researchers have adopted encodings based on graphs, simple permutations and random keys, among others.

We perform a detailed analysis of the results obtained, comparing the approach proposed in this thesis with respect to other heuristics reported in the specialized literature. We compare results with respect to a simple genetic algorithm (GA), a hybrid genetic algorithm (HGA), two versions of a blind greedy randomized adaptive search procedure (GRASP) and tabu search. The results obtained indicate that the algorithm proposed in this thesis is a viable alternative to solve the JSSP, given the quality of the solutions achieved, as well as their corresponding computational cost. The only serious contender for the algorithm proposed herein (from within the heuristics compared) is tabu search.

Agradecimientos

Agredezco a todas las personas que me estiman y aprecian, ya que sin ellos no hubiera sido posible la culminación de esta tesis. Especialmente a mi padre Eugenio, mi madre Aurora y mi hermana Dalia por todo el apoyo brindado incondicionalmente a lo largo de mi vida.

Agradezco a Erika por compartir su vida conmigo, su paciencia y apoyo brindado en la preparación de este trabajo.

Le doy gracias a Dios por dejar que comparta mi vida con todos ustedes y esperando que este trabajo pueda servirle a alguien más.

Agradezco a mi asesor el Dr. Carlos Artemio Coello Coello por su apoyo y consejos a lo largo de todo el tiempo de conocerlo.

A todos los maestros que he tenido a lo largo de mi vida no sólo académica, compañeros y todos los que han hecho algo conmigo, increíblemente soy el reflejo de todas las experiencias vividas.

Agradezco al CINVESTAV por todos los beneficios otorgados durante mi estancia en el programa de maestría.

Agradezco el apoyo obtenido a través del proyecto NSF-CONACyT titulado "Estudio y Desarrollo de Técnicas Avanzadas de Manejo de Restricciones para Algoritmos Evolutivos en el Contexto de Optimización Numérica" (Ref. 32999-A), cuyo responsable es el Dr. Carlos A. Coello Coello.

Índice general

Índice de figuras

Índice de cuadros

Índice de algoritmos

Capítulo 1

Introducción

Los problemas de planificación o programación de horarios son muy importantes en diversas aplicaciones del mundo real. Dichos problemas se utilizan en una amplia gama de aplicaciones diversas (p.ej., en las líneas de producción de una fábrica, en los hospitales para atender a los pacientes, en los aeropuertos para despachar los vuelos, en las escuelas para distribuir las actividades de los alumnos y profesores, en un taller, para decidir qué equipo es reparado primero o bien la secuencia de la reparación, etc.). Esto ha motivado un gran interés por este tipo de problemas. Dado que los problemas de planificación no son fáciles de resolver, muchos investigadores han desarrollado distintas metodologías y algoritmos que buscan eficientar su solución. Es precisamente esta complejidad del problema (el que se aborda particularmente en esta tesis es un problema NP-duro) lo que motivó el desarrollo de este trabajo de tesis, en el que se plantea el uso de una heurística inspirada en el sistema inmune para abordar el denominado *Job Shop Scheduling Problema* (JSSP) [38].

El JSSP consta básicamente de un conjunto de trabajos, donde cada uno tiene un conjunto de operaciones a ser procesadas en un conjunto de recursos, a los que denominaremos máquinas. Dichas operaciones tienen un orden y un tiempo de procesamiento en cada una de las máquinas y éste no es modificable. Por tanto, el objetivo es encontrar un plan de trabajo que cumpla con las restricciones del problema y que concluya todas las operaciones de los trabajos en el menor tiempo posible.

Se han desarrollado múltiples algoritmos para resolver el JSSP, pero debido a su complejidad en instancias grandes, no resulta posible contar con un método totalmente determinista para resolverlo en el caso general. De ahí que en los últimos años, se haya desarrollado diversas heurísticas de solución del JSSP basadas en algoritmos genéticos [33, 34], búsqueda tabú [75, 32, 4, 64, 82, 1], recocido simulado [54, 91, 47] y la colonia de hormigas [59, 60, 85, 84], entre otras.

En los últimos años ha habido un gran interés en desarrollar heurísticas para búsqueda y optimización con inspiración biológica. De entre ellas, destacan las heurísticas inspiradas en la evolución natural y las inspiradas en el funcionamiento de los organismos, fundamentalmente, aquellas basadas en nuestro sistema inmune.

Nuestro sistema inmune es un sistema cuya complejidad sólo es equiparable a la del mismo cerebro humano. Su función es defender nuestro organismo de ataques de seres extraños, como virus y bacterias.

Varias de las características de nuestro sistema inmune lo hacen muy atractivo desde el punto de vista de ciencias de la computación y de ahí el interés de emularlo en una computadora y usarlo como una herramienta para resolver problemas.

En este trabajo se utiliza una heurística inspirada en el principio de selección clonal del sistema inmune, la cual ha sido enriquecida con un algoritmo de búsqueda local que le proporciona una mayor eficacia al resolver el JSSP.

Esta tesis está organizada de la manera siguiente. En el capítulo 2, se discute el sistema inmune en general, así como las características del principio de selección clonal en particular. También se discuten los sistemas inmunes artificiales, y se proporcionan algunos conceptos básicos relacionados con éstos. Finalmente, en este capítulo se describe también un algoritmo de optimización basado en el principio de selección clonal, y en el cual se inspiró el algoritmo propuesto en esta tesis.

En el capítulo 3 se discute a mayor detalle el JSSP, proporcionando sus características, así como algunos ejemplos de codificaciones válidas para este problema.

En el capítulo 4, se describe a detalle el algoritmo propuesto en esta tesis para el JSSP, indicando cada uno de sus componentes, así como información suficiente como para permitir la replicación de nuestros resultados. Aquí se aborda tanto una versión mono-objetivo (que es la principal aportación de la tesis) como otra multi-objetivo del JSSP.

En el capítulo 5 se muestran los resultados de los problemas elegidos para validar el algoritmo propuesto, incluyéndose un análisis de los mismos, que comprende la descripción de los problemas y algunas estadísticas de las pruebas realizadas. También se comparan los resultados obtenidos por nuestro algoritmo con respecto a los producidos por otras heurísticas altamente competitivas y representativas del estado del arte en el área. Cabe destacar que esta comparación sólo se hace para la versión mono-objetivo del algoritmo, ya que en el caso multi-objetivo sólo se reportan los resultados obtenidos por el algoritmo aquí propuesto. Esto se debe a que no se pudieron

encontrar resultados contra los cuales se pudiera comparar nuestros resultados.

Finalmente se en el capítulo 6 las conclusiones de este trabajo, así como algunas rutas posibles de trabajo futuro. También se proporcionan varios apéndices en los que incluyen algunos conceptos básicos de optimización multiobjetivo, así como el código fuente de los programas utilizados para visualizar los resultados obtenidos.

Capítulo 2

El Sistema Inmune

En los últimos años, el interés por estudiar los sistemas inspirados en la biología se ha incrementado intuido en la idea de que la naturaleza es sabia y podemos tomar de ella algunos de sus modelos para resolver problemas. De los sistemas bioinspirados que se estudian actualmente tenemos los siguientes: redes neuronales [77], computación evolutiva [30] y más recientemente el sistema inmune artificial [26, 65].
El sistema inmune artificial es un modelo computacional de nuestro sistema inmune biológico que tiene la capacidad de realizar algunas tareas como el reconocimiento de patrones, aprendizaje, adquisición de memoria, generación de diversidad, tolerancia al ruido, generalización, detección distribuida y optimización; basados en los principios inmunológicos, nuevas técnicas computacionales han sido desarrolladas enfocándose no sólo a una mejor comprensión del sistema mismo, sino también a la solución de problemas de ingeniería [69].
En este capítulo se hablará del sistema inmune natural y artificial mencionando algunas de sus características principales para que se pueda comprender el modelo computacional del mismo.

2.1. Introducción

El interés en estudiar el sistema inmune se ha incrementado a través de los años. Computólogos, ingenieros, matemáticos, filósofos y otros investigadores están particularmente interesados en las capacidades de este sistema, cuya complejidad es únicamente equiparable con la del cerebro. Los sistemas inmunes artificiales son un nuevo campo de investigación que ha atraido con gran interés de investigadores de diferentes disciplinas, entre las cuales destacan [39, 25, 27, 37, 36].

Varias de las características del sistema inmune (SI) son de gran interés para científicos e ingenieros:

Singularidad Cada elemento posee su propio sistema inmune, con sus propias capacidades y puntos vulnerables.

Reconocimiento de cuerpos extraños Las moléculas (peligrosas) que no son nativas del cuerpo son reconocidas y eliminadas por el sistema inmune.

Detección de anomalías El sistema inmune puede detectar y reaccionar a los patógenos que el cuerpo no había encontrado antes.

Detección distribuida Las células del sistema inmune están distribuidas por todo el cuerpo, y lo más importante, es que ninguna está regida por ningún sistema de control centralizado.

Detección imperfecta (tolerancia al ruido) No es necesario que se haga una detección exacta de los cuerpos extraños, dado que el sistema inmune es flexible.

Reforzamiento del aprendizaje y memoria El sistema puede "aprender" las estructuras de los patógenos, de manera que las respuestas futuras a los mismos patógenos son más rápidas y fuertes.

Todas estas características que presenta el sistema inmune lo hacen una fuente de inspiración para muchos modelos computacionales. En la siguiente sección se describe brevemente el sistema inmune y después se hablará de los conceptos que sirvieron como base para la realización de este trabajo.

2.2. El Sistema Inmune

El Sistema Inmune (SI) es un complejo de células, moléculas y órganos que representan un mecanismo de identificación capaz de percibir y combatir infecciones en nuestras propias células y la acción de microorganismos infecciosos externos. La interacción entre el SI y otros órganos y sistemas permiten la regularización del cuerpo, garantizando su funcionamiento de una forma estable [42, 44].
Sin el sistema inmune, la muerte por infecciones sería inevitable. Sus células y moléculas están en constante vigilancia de los organismos infecciosos. Su reconocimiento es casi ilimitado para una gran variedad de células y sustancias externas que sean infecciosas, a las que se conoce como moléculas *impropias*, distinguiéndolas de las células nativas no infecciosas, que son conocidas como moléculas *propias* [44, 51, 50]. Cuando un agente patógeno (elemento infeccioso externo) entra en el cuerpo, éste es detectado e inmovilizado para su eliminación. El sistema es capaz de "recordar" cada infección, así que un segundo ataque del mismo agente patógeno es tratado más eficientemente; a esto se le conoce como la *respuesta secundaria*.
Hay dos formas en las que el cuerpo identifica material extraño: el denominado *sistema inmune innato* y el *sistema inmune adaptativo* [44, 45, 28, 46, 72, 20, 21, 58, 56, 57]. El sistema inmune innato es llamado así por que el cuerpo ya nace con la habilidad de reconocer ciertos microbios y la capacidad de destruirlos inmediatamente. Nuestro

sistema inmune innato puede destruir muchos agentes patógenos en su primer encuentro. Un componente importante de la respuesta del sistema inmune innato es una clase de proteína que se encuentra en la sangre, la cual es llamada *complemento* y que tiene la habilidad de asistir, o complementar la actividad de los anticuerpos. La inmunidad innata se basa en un conjunto de receptores codificados en el centro germinal que es conocido como *receptores de reconocimiento de patrones*(RRP, para reconocer patrones de moléculas asociados con patógenos microbiológicos, llamados *patrones moleculares asociados al patógeno* (PMAP. Los PMAPs son producidos únicamente por microbios y nunca por el propio organismo. Desde el reconocimiento por los RPP se producen señales que indican la presencia de agentes patógenos. De esta manera, las estructuras relacionadas al reconocimiento inmune pueden ser absolutamente distintas de nuestras propias células y moléculas para evitar el daño al tejido del cuerpo. La consecuencia de este mecanismo es que el sistema inmune innato sólo es capaz de distinguir entre lo propio y lo impropio, participando en el problema de la discriminación de lo propio e impropio y desempeñando el rol principal en la ayuda al sistema inmune adaptativo. El aspecto más importante del reconocimiento realizado por el sistema inmune innato es el hecho de que éste genera señales de co-estimulación, que puede llevar a la activación de las células T promoviendo el inicio de la respuesta del sistema inmune adaptativo. De esta manera, el reconocimiento del sistema inmune adaptativo sin el reconocimiento del sistema inmune innato puede resultar en la selección negativa (la selección negativa es el proceso del sistema inmune que autoregula la cantidad de anticuerpos que existen dentro de él)[69] de linfocitos que son los receptores involucrados en el reconocimiento adaptativo.
El sistema inmune adaptativo emplea antígenos receptores generados somáticamente que son clonados en dos tipos de linfocitos: células B y células T. Estos receptores de antígenos son generados por un proceso aleatorio, y como consecuencia, el diseño general de la respuesta inmune adaptativa se basa en la *selección clonal* de linfocitos expresando receptores con características específicas [15]. Los anticuerpos (AC) son moléculas que juegan el papel principal más importante en el sistema inmune adaptativo. Los receptores usados en la respuesta inmune adaptativa se forman juntando segmentos de genes. Cada célula está compuesta de diferentes segmentos de los disponibles para crear un receptor único, habilitando la célula para reconocer colectivamente los organismos infecciosos confrontados a lo largo de la vida del cuerpo [88]. La inmunidad adaptiva habilita el cuerpo para reconocer y responder ante el ataque de cualquier microbio, incluso si éste no había atacado el cuerpo anteriormente.

2.3. El Principio de Selección Clonal

El principio o teoría de selección clonal, plantea una explicación de cómo hace el sistema inmune para describir las características básicas de una respuesta inmune a un estímulo antigénico. Este principio establece la idea de que sólo aquellas células

que reconocen a los antígenos proliferan; de esta manera son seleccionadas aquellas que tienen la capacidad de reconocerlos. La Selección Clonal opera en las células B y las células T.

La respuesta inmune ocurre dentro de los nodos linfoides [93] y la expansión clonal de los linfocitos ocurre dentro del centro germinal (CG), en la región folicular de la pulpa blanca, que es rica en células antigénicas [83].

Cuando un animal es expuesto a un antígeno, algunas subpoblaciones de su médula ósea deriva células para producir anticuerpos. Cada célula secreta sólo un tipo de anticuerpo, que es relativamente específico para el antígeno. Por la unión de este receptor de inmunoglobulina, con una segunda señal de las células accesorio, tales como las células T auxiliares, un antígeno estimula a las células B a proliferar (dividirse) y madurar dentro del anticuerpo secretando células, llamadas células plasma. Mientras que las células plasma son los anticuerpos más activos que secretan los linfocitos B (que se dividen rápidamente), también secretan AC, aunque en una menor proporción. Las células B secretan AC, las células T no secretan anticuerpos, pero desempeñan el papel principal en la regulación de la respuesta de las células y son preeminentes en células intermediando respuestas inmunes. Los linfocitos, en adición a la proliferación o diferenciación en las células plasma, pueden diferenciarse en células B de memoria de larga vida. Las células de memoria circulan en la sangre, órganos linfoides y tejidos, y probablemente no crean anticuerpos [74], pero cuando son expuestas a un segundo estímulo antigénico comienza la diferenciación en los linfocitos capaces de producir anticuerpos de alta afinidad, preseleccionado del antígeno específico que estimuló la primera respuesta. La figura 2.3 describe el principio de selección clonal.

Las principales características de la teoría de selección clonal son[15]:

- Las nuevas células son copiadas (clonadas) de sus padres, sometidas a un mecanismo de mutación (hipermutación somática).
- Eliminación de los linfocitos recientemente diferenciados que llevan receptores auto-reactivos.
- La proliferación y diferenciación de las células maduradas con antígenos en contacto , y
- La persistencia de clones prohibidos, resistiendo la temprana eliminación por antígenos propios, como la base de enfermedades autoinmunes.

La analogía con la selección natural [38], resulta obvia, los mejores clones son los que pueden reconocer a un antígeno o, más precisamente, los que mejor se activan. Para que este algoritmo funcione, la población o repertorio de anticuerpos , tiene que ser bastante diversa para que reconozca cualquier forma extraña. El sistema inmune de

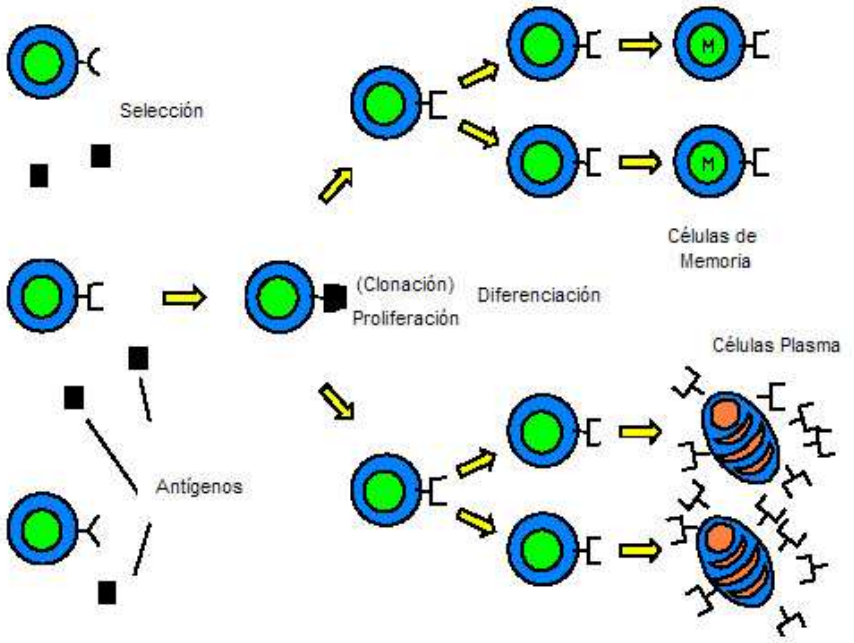


Figura 2.1: El principio de selección clonal.

un mamífero contiene un repertorio heterogéneo de aproximadamente 10^{12} linfocitos en humanos [73] y otro tanto de células B (no estimuladas) que oscila entre $10^5 - 10^7$ receptores idénticos como anticuerpos. El repertorio se dice que está completo, cuando tiene los recursos para reconocer cualquier forma.

Con lo anterior se concluye nuestra breve introducción de lo que es el sistema inmune, junto con algunas de sus características, siendo lo más interesante el haber descrito el principio de selección clonal, que es la parte que se simulará en este trabajo.

2.4. El Sistema Inmune Artificial

El principio de selección clonal es usado para explicar las características básicas de una respuesta inmune adaptativa ante un estímulo antigénico. Las células seleccionadas son sujetas a un proceso de maduración de afinidad, que mejora su afinidad con los antígenos en cuestión. En esta sección veremos algunas de las definiciones que pretenden describir el sistema inmune artificial, además de algunas de sus principales características. Al final hablaremos del funcionamiento y de las partes que componen CLONALG [70] que es un algoritmo inspirado en el principio de Selección Clonal.

Diversos autores han propuesto definiciones de lo que se entiende por un sistema inmune artificial. Algunas de ellas se presentan a continuación:

> *"Los sistemas inmunes artificiales son metodologías para la manipulación de datos, clasificación, representación y razonamiento, los cuales siguen el paradigma biológico del sistema inmune humano"* [89].
>
> *"El sistema inmune artificial es un sistema computacional basado en los principios del sistema inmune natural"* [87].
>
> *"Los sistemas inmunes artificiales son metodologías inteligentes inspiradas en el sistema inmune, enfocadas a resolver problemas del mundo real"* [26].
>
> *"Los sistemas inmunes artificiales son sistemas adaptativos, inspirados en la teoría, modelos y principios inmunológicos y las funciones inmunes observadas, que se aplican a la solución de problemas"* [66].

Se sabe que el sistema inmune tiene memoria, donde almacena experiencias pasadas, esto debido a la interacción de las células que lo forman. Debido a esto el sistema inmune puede generar respuestas a cuerpos extraños que tratan de entrar al organismo (llamados antígenos).

El sistema inmune se comporta como un sistema dinámico, capaz de ajustarse a lo que se presente. El sistema inmune es un sistema muy complejo con varios componentes funcionales. Emplea un sistema multiniveles para defenderse contra invasores, a través de mecanismos específicos (adquiridos, con el sistema inmune adaptativo) y no específicos (el sistema inmune innato). El rol principal del sistema inmune es reconocer todas las células (o moléculas) dentro del cuerpo y categorizarlas en propias e impropias. Las células impropias son categorizadas a fin de estimular un tipo apropiado de mecanismo de defensa. El sistema inmune aprende a distinguir entre cuerpos extraños llamados antígenos (bacterías, moléculas, etc.) y las células o moléculas propias del cuerpo.

De acuerdo con los inmunólogos, nuestro cuerpo mantiene un gran número de células que circulan por el cuerpo a través de la sangre. Estas células son los linfocitos, que son las células principales que participan en la respuesta inmune y que poseen atributos específicos: diversidad, memoria y adaptabilidad. Hay dos tipos principales de linfocitos que se conocen como células B y células T. Se puede encontrar información más detallada de la función que realiza cada una de estas células en [26].

Las características principales que se toman del sistema inmune natural en los sistemas inmunes artificiales son las siguientes:

Reconocimiento: El sistema inmune puede reconocer y clasificar diferentes patrones y generar determinadas respuestas.

Extracción de características: Hay células que se encargan de detectar la presencia de cuerpos extraños.

Diversidad: Esta característica ocasiona que el sistema inmune pueda reconocer una variedad casi ilimitada de antígenos, haciendo que éste sea más eficiente.

Aprendizaje: El sistema inmune es capaz de aprender de las experiencias, y con esto sabe cómo atacar a los cuerpos extraños. Este mecanismo hace que se multipliquen las células buenas y que se ataquen a las malas para poder preservar la integridad del sistema. También este tipo de mecanismo es conocido como expansión clonal.

Memoria: Es un mecanismo que mantiene un registro de lo que hizo daño al sistema, esto es, el sistema tiene la capacidad de recordar quién fué el que lo atacó y mantiene un registro de la célula que lo destruye, pudiendo después clonar esta célula para volver a atacar el cuerpo extraño.

Detección distribuida: El sistema inmune puede verse como un sistema distribuido, es decir, las células buenas circulan por todo el organismo a través de la sangre. Estas células no dependen de ningun sistema centralizado, y cuando detectan la presencia de algún cuerpo extraño, inmediatamente avisan al resto del sistema.

Auto-Regulación: Cuando el sistema produjo muchas células buenas (anticuerpos) para destruir a las células malas (antígenos), el sistema tiene que deshacerse de éstas, ya que una vez que cumplieron con su propósito no le sirven para nada al sistema. Sin embargo no se eliminan sino que se conservan unas cuantas para un posible uso posterior. Es decir, en caso de un nuevo ataque provocado por el mismo antígeno, el sistema sabe cómo defenderse.

Co-estimulación: La activación de las células B está regulada a través de la co-estimulación. La segunda señal (que viene de las células T auxiliares) ayuda a asegurar la tolerancia y juicio entre invasores peligrosos e inofensivos o falsas alarmas.

Protección dinámica: La expansión clonal y la expansión somática permiten la generación de células inmunes de alta afinidad (llamada maduración de afinidad). Este proceso realiza un equilibrio dinámico entre la exploración y la explotación en la inmunidad adaptativa. La protección dinámica incrementa el alcance que provee el sistema inmune a través del tiempo.

Detección probabilística: La reacción cruzada en la respuesta del sistema inmune es un proceso estocástico. También la detección es aproximada. Esto es, un linfocito puede emparejarse con muchos tipos (estructuralmente relacionados) de antígenos.

Dicho lo anterior, es entonces posible diseñar un algoritmo con varias de las características antes mencionadas. Así mismo, lo que podemos aprender de los sistemas biológicos complejos como el sistema inmune, y su simulación computacional puede aplicarse a la solución de problemas del mundo real.

2.5. El Algoritmo de Selección Clonal (CLONALG)

Aunque en este trabajo no se pretende describir detalladamente el principio de selección clonal, se intenta dar una idea lo suficientemente amplia de dicho principio a fin de facilitar la comprensión del algoritmo desarrollado. En esta sección se describe el algoritmo de selección clonal junto con cada una de las partes que lo componen, pero nos enfocamos únicamente a la versión del algoritmo de selección clonal para optimización, ya que no es el objetivo extenderse en este tema, sino sólo comprender sus partes básicas.

2.5.1. Aspectos computacionales

Hemos descrito el principio biológico de selección clonal en la sección 2.2, y es precisamente la idea en la que se basa el algoritmo de selección clonal la que abordaremos en esta sección. Este algoritmo se propuso inicialmente para reconocimiento de patrones y para optimización, pero sólo mencionaremos la versión para optimización pues es la relacionada con nuestro trabajo. Los principales aspectos del sistema inmune que toma este algoritmo son los siguientes: mantenimiento de un conjunto específico de memoria, clonación y selección de los más aptos, eliminación de los anticuerpos no estimulados, maduración de afinidad y reselección de genes proporcional a su afinidad antigénica, generación y mantenimiento de diversidad.

2.5.2. Aspectos teóricos

El algoritmo de selección clonal (CLONALG) [67], representa una implementación computacional del principio de selección clonal responsable de describir el comportamiento de las células B durante una respuesta inmune adaptativa. En la implementación se asume un repertorio de anticuerpos (Ab) que pueden ser estimulados por un antígeno (o el valor de una función objetivo $g(\cdot)$ a ser optimizada) y los anticuerpos con una afinidad alta pueden ser seleccionados para generar poblaciones de clones. Durante el proceso de clonación, algunos anticuerpos pueden sufrir mutaciones somáticas proporcionales a su afinidad antigénica y los clones de afinidad alta serán seleccionados para formar el conjunto de memoria. Los anticuerpos de baja afinidad son reemplazados, simulando el proceso de edición del receptor [69].

Algoritmo 1 El algoritmo de selección clonal para optimización

Require: -

Entrada: Ab, gen, n, d, L, β
Salida: Ab, f
for $t := 1$ to $t < gen$ **do**
  $f :=$*decodificación*(Ab); {Paso 2}
  $Ab_n :=$*selección*(Ab, f, n); {Paso 3}
  $C :=$*clonación*(Ab_n, β, f); {Paso 4}
  $C^* :=$*hipermutación*(C, f); {Paso 5}
  $f :=$*decodificación*(C^*); {Paso 6}
  $Ab_n :=$*selección*(C^*, f, n); {Paso 7}
  $Ab :=$*inserción*(Ab, Ab_n); {$Ab \leftarrow Ab_n$}
  $Ab_d :=$*generación*(d, L); {Genera d anticuerpos de longitud L}
  $Ab :=$*reemplazamiento*(Ab, Ab_d, f); {Paso 8}
end for
$f :=$*decodificación*(Ab);

La aplicación del principio de selección clonal a optimización se muestra en el pseudo-código del algoritmo 1. La descripción de cada uno de los pasos de este algoritmo se describen a continuación y el diagrama de flujo del mismo en la figura 2.2 [68].

1. En el paso 1 no hay una población de antígenos a reconocer explícitamente, sino una función objetivo a ser optimizada (maximizar o minimizar). De esta manera, la afinidad de un anticuerpo corresponde a la evaluación de la función objetivo, y cada anticuerpo representa un elemento en el espacio de entrada. En adición, como no existe una población específica de antígenos a ser reconocida, el conjunto de anticuerpos conforma la memoria y esto hace que no sea necesario que la memoria esté en otra estructura separada.

2. Determinar el vector f que obtiene su afinidad de todos los N anticuerpos.

3. Seleccionar los n anticuerpos con la afinidad más alta para formar el nuevo conjunto de anticuerpos.

4. Los n mejores anticuerpos pueden ser clonados, para generar una nueva población de clones de los mejores anticuerpos, es decir de clones que tienen la mejor afinidad.

5. La nueva población de clones es sometida a un proceso de mutación, proporcional a su afinidad: el clon que tiene una afinidad más alta, es clon el que tiene un porcentaje de mutación más pequeño.

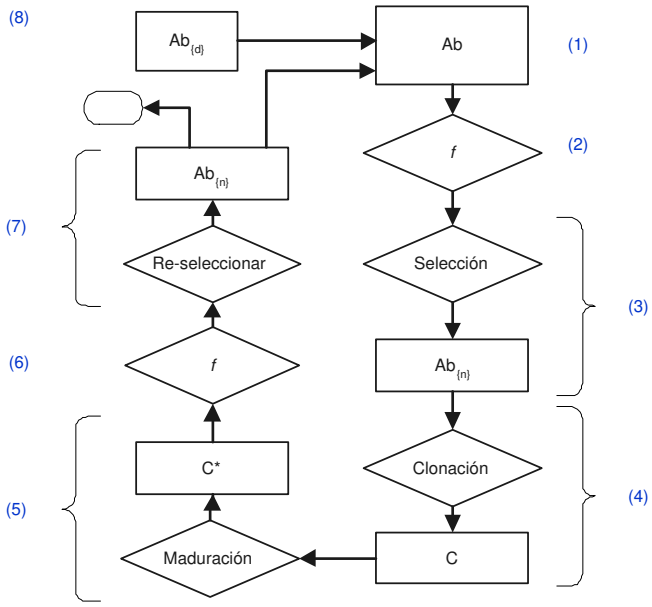


Figura 2.2: Diagrama de flujo del algoritmo de selección clonal.

6. Determinar la afinidad de la población de clones mutados.

7. Del conjunto de clones madurados, seleccionamos el anticuerpo que tenga la más alta afinidad con respecto al antígeno y éste es candidato a ser ahora el antígeno, siempre y cuando sea mejor que el actual.

8. Finalmente se reemplazan los d peores individuos de la población de anticuerpos con los mejores que se generaron después del proceso de clonación y mutación.

Después de realizar los ocho pasos, podemos decir que se completó una generación del algoritmo.
Hemos presentado el algoritmo de selección clonal. Para más información respecto de la implementación refiérase a [70], ya que aquí únicamente se muestran los aspectos mínimos necesario del algoritno para comprender este trabajo de tesis.

Capítulo 3

El problema del Job Shop Scheduling

Haciendo referencia a lo que dice Blazewics et al. [13], los problemas de planificación de tareas (scheduling) pueden ser descritos ampliamente como: "*el problema de acomodar los recursos en el tiempo para realizar un conjunto de tareas*". En la literatura de planificación de tareas se trata una gran variedad de problemas. En este capítulo se dará una definición y una introducción del problema del Job Shop Scheduling[1]. Por razones de brevedad y para hacer esta descripción concreta, se usan términos específicos de producción, pero el JSSP es importante no sólo en problemas de producción (p.e. el tránsito aéreo al despegue y aterrizaje o médicos y enfermeras cuidando a un paciente en un hospital, etc.). Este capítulo desde luego no es una descripción completa del JSSP, sino más bien una introducción necesaria para comprender los conceptos utilizados a lo largo de este trabajo.

3.1. Introducción

Dentro de la gran variedad de problemas de planificación de recursos que existen, el JSSP es uno de los que han generado un mayor número de estudios. Esto se debe, sobre todo, a que es uno de los más difíciles de resolver. Además, el JSSP se utiliza en una amplia variedad de problemas del mundo real. En este capítulo veremos las características principales del JSSP junto con los objetivos que podemos optimizar de este problema.

[1] que podría traducirse como *el problema de acomodamiento de trabajos*, aquí utilizaremos las siglas en inglés **JSSP**

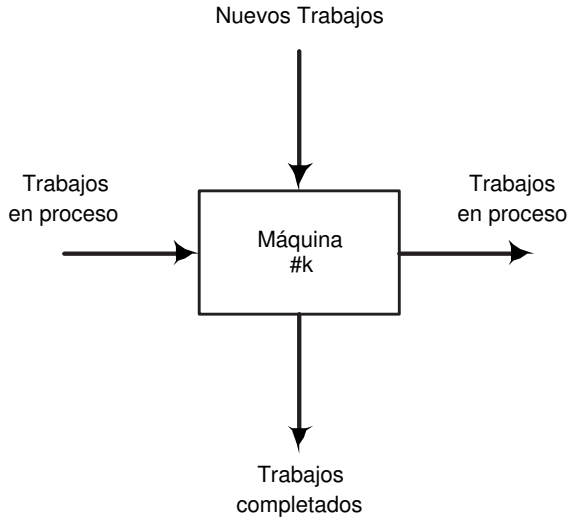


Figura 3.1: Flujo de los trabajos en el JSSP

3.2. El problema del Job Shop Scheduling Clásico

El JSSP tiene la característica de que los trabajos son unidireccionales, es decir que sólo tienen una dirección. Otra característica importante es que la máquina que inicia no lo hace precisamente con el primer trabajo, sino que cualquier máquina puede empezar o terminar, ya que el orden de procesamiento de los trabajos en las máquinas es una característica de cada uno de los problemas. En la figura 3.2 podemos ver el modelo de cada una de las máquinas en el JSSP.

Ahora describiremos el JSSP clásico: Hay m máquinas donde tienen que ser procesados n trabajos con m operaciones por trabajo. Cada operación tiene que ser procesada en una máquina. El orden de procesamiento de cada una de las operaciones de cada trabajo está dado por el problema y éste es inamovible. Una vez que una operación comienza a ser procesada no puede interrumpirse. Finalmente, todas las operaciones de todos los trabajos tienen la misma prioridad. Para poder ser más claros listaremos las restricciones del JSSP clásico:

- No está permitido que dos operaciones del mismo trabajo se procesen simultáneamente.
- Ninguna operación tiene prioridad sobre las demás.
- Ningún trabajo puede ser procesado más de una vez en la misma máquina.
- Cada trabajo es procesado hasta concluirse, aunque haya que esperar y retardarse entre las operaciones procesadas.

- Un trabajo puede iniciarse en cualquier momento siempre y cuando esté disponible la máquina y no se haya especificado un tiempo de inicio.
- Los trabajos tienen que esperar a que la siguiente máquina esté disponible para que éste sea procesado.
- Ninguna máquina puede procesar más de un trabajo a la vez.
- Los tiempos de configuración y cambio de máquina son independientes del orden de procesamiento y éste está incluido en los tiempos de procesamiento.
- Hay sólo un tipo de máquina.
- Las máquinas pueden estar ociosas en cualquier momento del plan de trabajo.
- Las máquinas están disponibles en cualquier momento.
- Las restricciones tecnológicas están bien definidas y son previamente conocidas, además de que son inamovibles.

El objetivo a ser optimizado más usado en el JSSP es el de encontrar un plan de trabajo válido (que no viole ninguna de las restricciones antes mencionadas) que complete todos los trabajos en el menor tiempo posible. Este objetivo es conocido como *makespan* y es el que se minimiza. Los planes de trabajo factibles pueden obtenerse permutando el orden de las operaciones en las máquinas, teniendo cuidado de no violar las restricciones del problema. De acuerdo con esto, lo que se hace es una minimización a un problema de optimización combinatoria, ya que la representación de las soluciones son permutaciones sobre las operaciones de cada trabajo. Más específicamente, las operaciones a ser procesadas en una máquina forman la *secuencia de operaciones* para esa máquina. El plan de trabajo está formado de n secuencias de operaciones para cada máquina.
Puesto que cada secuencia de operaciones puede ser permutada independientemente de la secuencia de operaciones de otra máquina, el número total de posibles soluciones para el JSSP es $(n!)^m$, donde n denota el número de trabajos y m el número de máquinas. Este número constituye el espacio de búsqueda del problema.
El siguiente ejemplo muestra el tamaño del problema para resolverse por métodos clásicos de búsqueda o enumeración.
Suponiendo que tenemos un problema de 24 operaciones y 1 máquina, el espacio de búsqueda para este problema es $(24!) = 6.20 \times 10^{23}$ posibles soluciones. Ahora bien, si presuponemos que la edad del universo es 20 mil millones de años tenemos que $20 \times 10^9 \times 365.25 \times 24 \times 60 \times 60 = 6.31 \times 10^{17}$ segundos, que es aproximadamente 6.31×10^{23} μseg. Si tuviéramos una computadora capaz de procesar una solución cada μseg, aún considerando toda la edad del universo apenas hubiéramos podido resolver este problema por enumeración. Esto nos da una idea de la complejidad del problema

y ésta es la razón principal por la que ha resultado de gran interés para los investigadores.
Garey y Jonhson [61], demostraron que el JSSP es un problema NP-duro, y de entre esa clase es uno de los menos tratables [5].

3.3. Definición del Problema

El JSSP que nos interesa resolver es de tamaño $m \times n$ y consiste de n trabajos $J_1, J_2, ..., J_n$ y m máquinas $M_1, M_2, ..., M_m$. Para cada trabajo J_i, se tiene una secuencia de k_i operaciones $O_i = o_{i1}, o_{i2}, ..., o_{ik_i}$ que describe el *orden de procesamiento* de las operaciones del trabajo J_i. El orden de procesamiento de los trabajos es algunas veces llamado *restricciones tecnológicas*. La operación o_{ij} es la *j-ésima* operación del trabajo i, y éste es procesado en una máquina específica $M_{o_{ij}}$ y tiene un tiempo de procesamiento $\tau_{o_{ij}}$. El conjunto de operaciones P es denotado por T. Es importante mencionar que para este problema el conjunto de k_i operaciones es de tamaño m. Además el conjunto de operaciones O_i esta representado con los trabajos J_i, donde la primera aparición de J_i corresponde a o_{i1}, la segunda aparición a o_{i2} y así sucesivamente.

Cada trabajo puede tener asociado un tiempo de liberación (*release time*) r_i antes de que otro trabajo pueda procesarse y cada máquina un tiempo de preparación (*setup time*) u_i que se requiere antes de que se pueda hacer algo en la máquina. Es común trabajar con intervalos discretos de tiempo. La idea es que los tiempos de procesamiento, los tiempos de preparación y los tiempos de liberación sean enteros. Cuando los trabajos son procesados en las máquinas, cada máquina puede procesar sólo un trabajo a la vez. Sólo una operación de cada trabajo puede ser procesada a la misma vez y ninguna operación tiene más prioridad que otra; es decir todas tienen la misma prioridad (esto es, que una vez que una operación es iniciada, no puede ser detenida hasta que ésta haya concluido).

Una instancia del JSSP se define mediante una matriz que contiene el orden de procesamiento de cada una de las operaciones (véase el cuadro 3.3, donde se muestra cada uno de los trabajos y el orden de cada una de las operaciones en las máquinas y otra matriz donde se muestra el tiempo de procesamiento de cada una de las operaciones sobre cada una de las máquinas). Para facilitar su comprensión se muestra en un solo cuadro el orden de las máquinas en las que se va a procesar cada una de las operaciones del trabajo y el tiempo de procesamiento correspondiente a cada operación en cada máquina (vease el cuadro 3.3).

Un plan de trabajo es aquel donde se tienen los trabajos acomodados en las máquinas de tal forma que cumplan todas las restricciones del problema. Un plan de trabajo es generalmente visualizado mediante un diagrama de Gantt; un ejemplo se muestra en la figura 3.3. Cada renglón en el diagrama representa una máquina y cada cuadro representa una operación. Los cuadros están marcados con el número de

trabajo	máquina		
0	0	1	2
1	0	2	1
2	1	0	2

(a)

trabajo	tiempo		
0	3	3	3
1	2	3	4
2	3	2	1

(b)

Cuadro 3.1: (a). Flujo de los trabajos en el JSSP (b). Tiempos de los trabajos en el JSSP

trabajo	máquina(tiempo)		
0	0(3)	1(3)	2(3)
1	0(2)	2(3)	1(4)
2	1(3)	0(2)	2(1)

Cuadro 3.2: Flujo y tiempos de los trabajos en el JSSP

trabajo al que corresponde, es decir se identifican por el color y el número especificado. El tiempo que tarda en ejecurse cada operación se especifica en el eje horizontal.

Además, en el JSSP clásico (que es el que estamos describiendo), usualmente se requiere que para cada trabajo J_i, la secuencia de operaciones O_i contenga exactamente una operación para ser procesada por cada una de las máquinas del problema. Los problemas tratados en esta tesis cumplen con estas restricciones.
Hay muchos objetivos a tratar en el JSSP, pero en este caso sólo hablaremos de los que se tratan en este trabajo y antes de eso es necesario conocer la notación a utilizarse:

- C_i. El tiempo de terminación (*completion time*) de la última operación del trabajo J_i.
- F_i. El tiempo de flujo (*flow time*) del trabajo J_i
- d_i. Algunos problemas especifican el tiempo comprometido d_i (*due date*), o sea el tiempo máximo para J_i, que es el tiempo en que a lo más debe ser concluido un trabajo.

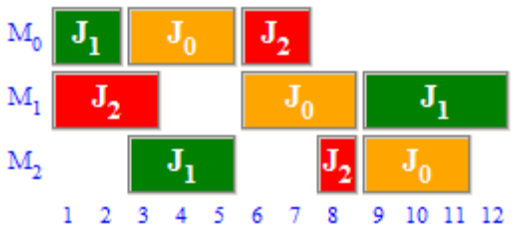


Figura 3.2: La figura muestra un plan de trabajo, esto es, una solución para el JSSP cons los datos del cuadro 3.2.

- L_i. El retraso (*lateness*) de J_i, $L_i = C_i - d_i$. La medición de retraso es el tiempo que se excede un trabajo en ser concluido sobre el tiempo comprometido. Si el trabajo termina antes, se dice que el retraso es negativo.
- T_i. La tandanza (*tardiness*)de J_i, $T_i = \text{máx}(L_i, 0)$.
- E_i. La puntualidad (*earliness*) de J_i, $E_i = \text{máx}(-L_i, 0)$.

Normalmente las medidas de desempeño para el JSSP son las siguientes:

makespan $C_{\text{máx}} = \text{máx}_{i \in \{1..n\} C_i}$. Tiempo mínimo para completar todos los trabajos.

total flow time $F_\Sigma = \Sigma_{i \in \{1..n\}} F_i$. Es el tiempo consumido por todos los trabajos.

total lateness $L_\Sigma = \Sigma_{i \in \{1..n\}} L_i$. La suma de todos los retrasos de los trabajos.

total tardiness $T_\Sigma = \Sigma_{i \in \{1..n\}} T_i$. La suma de todas las tardanzas de los trabajos.

total earliness $E_\Sigma = \Sigma_{i \in \{1..n\}} E_i$. La suma de todos los tiempos de terminación previos al tiempo comprometido.

maximum lateness $L_{\text{máx}} = \text{máx}_{i \in \{1..n\}} L_i$. El retraso del más atrasado de los trabajos.

maximum tardiness $T_{\text{máx}} = \text{máx}_{i \in \{1..n\}} T_i$. La tardanza del más tardado de los trabajos.

Todas las medidas de desempeño del JSSP se tienen que minimizar. Todas estas mediciones son importantes para muchas áreas, ya que lo más importante es reducir los costos en todas las formas, pero para el caso específico de este trabajo, sólo utilizaremos el tiempo mínimo para completar todos los trabajos (*makespan*) y el tiempo consumido por los trabajos (*flow time*). Más adelante explicaremos porque es que se toman estas medidas de desempeño para el problema.

3.3.1. Tipos de planes de trabajo

Los planes de trabajo generalmente se categorizan en las siguientes clases:

- Un plan de trabajo en que ninguna operación puede iniciarse sin cambiar el orden de procesamiento o sin que se viole alguna de las restricciones tecnológicas, es llamado *semiactivo*.
- Un plan de trabajo en que ninguna operación puede iniciarse sin que se retarde cualquier otro operación o sin que se viole alguna de las restricciones tecnológicas, es llamado *activo*.

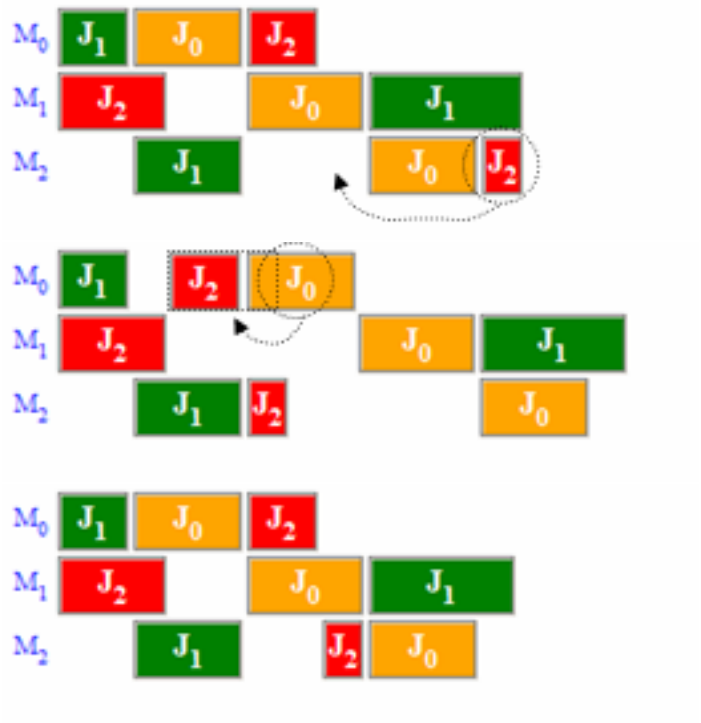


Figura 3.3: En la primer gráfica, se muestra un plan de trabajo semi-activo. Se ve que se puede cambiar la tercera operación del trabajo J_2, que si se inicia antes no afecta el tiempo de operación del plan de trabajo. En la segunda gráfica se muestra un plan de trabajo activo. En este caso si se cambia la primera operación del trabajo J_0 con la segunda operación del trabajo J_2, se hace un cambio que afecta a todo el plan de trabajo. Finalmente, en la tercera gráfica se muestra un plan de trabajo no retardado. En este caso, no existe ningún movimiento que haga que no se retrase ninguna operación.

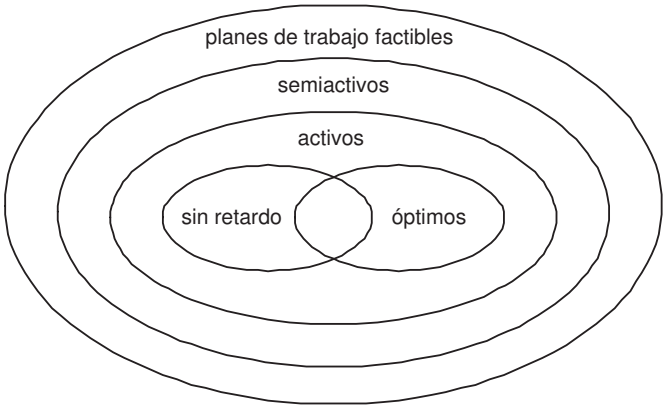


Figura 3.4: Esta figura muestra la jerarquía de las clases de planes de trabajo para las medidas de desempeño normal. El conjunto de los planes de trabajo *sin retardo* es un subconjunto de los planes de trabajo *semiactivos*. Para las medidas de desempeño normales, los planes de trabajo *óptimos* son un subconjunto de los planes de trabajo *activos*, pero éstos no son un subconjunto de los planes de trabajo *no retardados*.

- Un plan de trabajo en que ninguna máquina está nunca ociosa si una operación está lista para ser procesada en ésta, es llamado *sin retardo*.

En la figura 3.3 se muestra un ejemplo de cada una de las clases de planes de trabajo.
El conjunto de los planes de trabajo *sin retardo* es un subconjunto de los planes de trabajo *semiactivos*. Para medidas de desempeño normales de los planes de trabajo, se puede demostrar que existe un plan de trabajo óptimo [31]. Sin embargo también puede ser demostrado para algunos problemas de planes de trabajo *sin retardo* existe un óptimo. La relación entre estos planes de trabajo y los óptimos se muestra en la figura 3.4. Por esta razón cuando resolvemos problemas de planificación de tareas utilizamos medidas de desempeño normales y usualmente las tareas pertenecen al conjunto de los planes de trabajo *activos*, trayendo como consecuencia la reducción del espacio de búsqueda y garantizando que se encuentre la solución óptima.
Hemos descrito las características básicas del JSSP y ahora solamente nos queda decir que hay una amplia variedad de técnicas para resolverlo [43]. En este trabajo no se hace una revisión de éstas. Únicamente se hablará de otras técnicas relevantes para la comparación efectuado con respecto al algoritmo propuesto en esta tesis, por lo que nos limitaremos a describirlas brevemente.

Capítulo 4

Descripción de la Técnica

En este capítulo describiremos la técnica que utilizamos para optimizar el JSSP, basándonos en algunos principios del sistema inmune artificial. Concretamente, este trabajo se basó en el algoritmo de selección clonal para optimización que se describió en la sección 2.5. A dicho algoritmo se le introdujeron diversas modificaciones tales como la utilización de una estrategia para reacomodar las operaciones de los trabajos en las máquinas y eliminar huecos, esto con la finalidad de utilizar mejor los recursos. En este capítulo se describe el algoritmo, así como cada una de las partes que lo componen a fin de poder reproducir los resultados obtenidos, los cuales serán abordados en el capítulo siguiente.

Se hablará tanto de la versión del algoritmo para un solo objetivo como de la versión multiobjetivo.

4.1. Representación del problema

Como en todos los problemas que enfrentamos en el mundo real, para poder resolverlos tenemos que encontrar una forma de abstraerlos y poder representar sus posibles soluciones. A esto se le conoce como representación del problema y este caso no es la excepción, ya que éste es el primer punto a resolver antes de proponer un algoritmo de solución.

Existen varias formas de representar el JSSP para su solución. Entre las más comunes tenemos la representación con grafos disyuntivos, representación binaria y la representación de permutaciones con y sin repeticiones [96], siendo éstas de las más empleadas para representar el problema al abordarlo con heurísticas.

En este trabajo optamos por la representación de permutaciones propuesta en [96, 9, 10].

Utilizamos la representación de permutaciones con repeticiones dado que encontramos muchas ventajas de esta representación sobre las otras. Cabe mencionar, sin embargo que ésta representación no está libre de problemas como se verá más adelante.

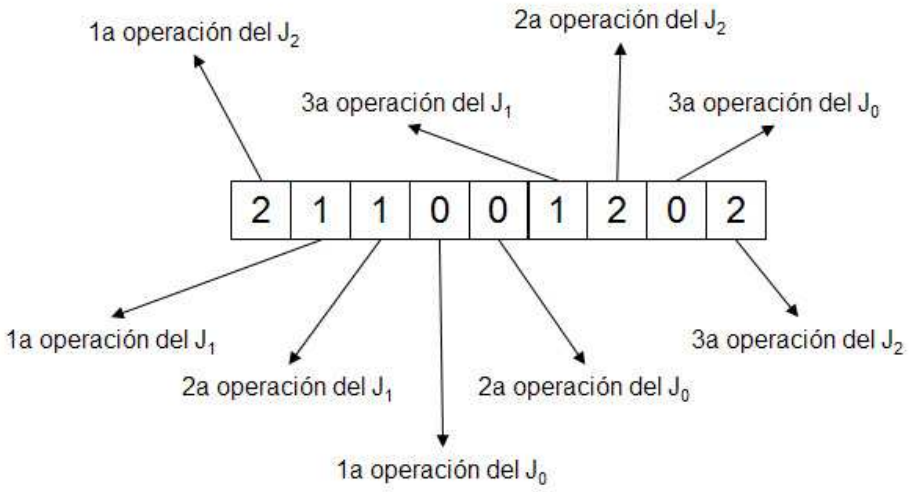


Figura 4.1: En la figura se muestra la representación del problema. Cada uno de las celdas representa una operación de los trabajos y el orden de aparición corresponde a la operación de cada trabajo.

Primero que nada vamos a definir lo que es una permutación:

Una permutación simple se define como el ordenamiento de los elementos de un conjunto finito I de tamaño N. Un ordenamiento en que los elementos pueden aparecer más de una vez es llamado permutación con repeticiones. El número de diferentes permutaciones con repeticiones que contiene el elemento $i \in I$ es exactamente l_i veces dado por:

$$\frac{(l_1+l_2+...+l_i)!}{l_1!\times l_2!\times ...\times l_i!}$$

Si $l_i = 1$ para todo $i \in I$, este número es igual a $N!$, que es sólo el número permutaciones de una permutación simple.
Vamos a ver un ejemplo del tamaño del espacio de búsqueda para un problema de 10×5, es decir aquél en el que cada uno de los 5 elementos se repite 10 veces:

$$\frac{(n_1+n_2+...+n_m)!}{n_1!\times n_2!\times ...\times n_m!} = \frac{(5+5+5+5+5+5+5+5+5+5)!}{5!\times 5!\times 5!\times 5!\times 5!\times 5!\times 5!\times 5!\times 5!\times 5!} = \frac{50!}{(5!)^{10}} = 4.91 \times 10^{43}$$

Una vez que hemos definido una permutación, ahora se explicará cuál es la estructura de datos que va a representar una solución y cómo es que funciona.
Para poder representar el problema necesitamos un arreglo de datos de tamaño $m \times n$, donde cada elemento del arreglo representa una operación de un trabajo (ver figura 4.1).

De acuerdo con esto, podemos observar que el espacio de búsqueda de esta representación es más grande que la presentada anteriormente. La diferencia es que al

usar permutaciones con repeticiones se distribuyen las operaciones a lo largo de todo el arreglo y no como en la de permutaciones sin repeticiones, en que se respeta cada espacio de una máquina. Más específicamente, se tiene que para cada trabajo se tiene un secuencia de operaciones de tamaño m y se tienen n secuencias de operaciones. En la representación de permutaciones con repeticiones lo que se tiene es que se permuta el arreglo de $m \times n$. Esto se reduce finalmente al espacio de búsqueda $(n!)^m$ cuyo tamaño se ha justificado previamente.

4.2. Algoritmo General Mono-objetivo

En esta sección se presenta la implementación del algoritmo general que utilizamos para optimizar el JSSP, además que se detalla cada una de sus partes para facilitar la comprensión de su funcionamiento. Primero, se muestra el diagrama del algoritmo donde se indican las partes de las que se compone.
En la figura 4.2, se muestra el algoritmo general con los pasos más importantes.

Ahora se describe cada una de las partes que componen al algoritmo:

- En el paso (1) se carga el problema (en el formato definido en [8]). El problema es almacenado en 2 matrices de tamaño $m \times n$. En la primera matriz se almacena la secuencia de los trabajos en las máquinas y en la segunda matriz el tiempo de procesamiento de cada operación en su respectiva máquina.

Algoritmo 2 Generación de un antígeno y anticuerpo

```
int M[m * n];
int i, j;
for i := 0 to m do
  for j := 0 to n do
    M[i * n + j] := i;
  end for
end for

for i = 0 to m * n do
  Intercambia(M[i], M[Aleatorio(i + 1, m * n)]);
end for
```

- En el paso (2) se procede a la generación del anticuerpo, el antígeno y la decodificación del antígeno. El antígeno y el anticuerpo tienen la misma estructura (tamaño y elementos). Lo único que cambia es el orden de los elementos que tiene pero en ambos casos, se usa un arreglo de enteros. La forma de generar el antígeno se muestra en el algoritmo 2 y se describe a continuación: Se

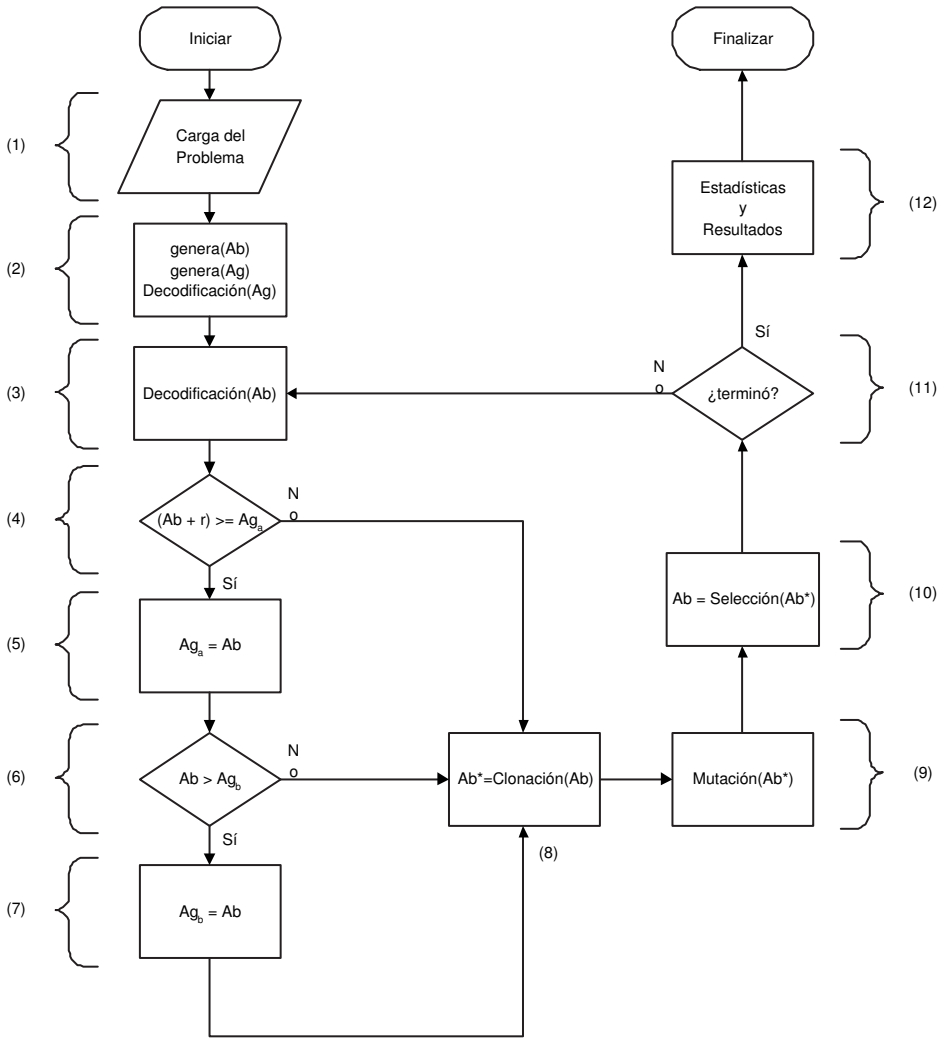


Figura 4.2: Diagrama de flujo del algoritmo del sistema inmune artificial para resolver el JSSP

tiene un arreglo de tamaño $m \times n$ donde se almacenan todas las operaciones de los trabajos. Primero se llena el arreglo con $0_1, 0_2, ..., 0_n, 1_1, 1_2, ..., 1_n, ..., (m-1)_1, (m-1)_2, ...(m-1)_n$. Posteriormente se barajan los elementos y se obtiene como resultados un arreglo que contiene todas las operaciones de todos los trabajos pero revueltos aleatoriamente. Recordemos que cada número corresponde a cada trabajo y el orden de aparición de cada número representa la operación de dicho trabajo (véase la figura 4.1). El arreglo que representa al antígeno o al anticuerpo son soluciones potenciales.
El proceso de decodificación consiste básicamente en acomodar las operaciones de los trabajos en las respectivas máquinas sin violar las restricciones (véase la subsección 4.2.1). Al decodificar el antígeno se obtiene el valor del makespan del problema con la secuencia de las operaciones que se encuentran codificadas en el antígeno.

- En el paso (3) se hace la decodificación del anticuerpo que es de la misma forma del antígeno.

- En el paso (4) se hace la comparación de la afinidad del anticuerpo con el antígeno. Esta medida de afinidad no se determina midiendo qué tanto se parece fenotípicamente el anticuerpo al antígeno sino que se hace usando el valor del makespan, que es un entero. Como se muestra en este paso, al valor del makespan del anticuerpo se le agrega un parámetro r que es un valor que se fija a 2 y que permite que el algoritmo explore la vecindad de la solución. El valor de este parámetro se determinó experimentalmente y se encontró que con el valor de 2 se obtenían los mejores resultados para los problemas de prueba adoptados en esta tesis.
  Si el valor del makespan del anticuerpo sumado del parámetro r es menor o igual al valor del makespan del antígeno, se continuá al paso (5). De lo contrario no se hace nada y se continúa hasta el paso (8).

- En el paso (5) lo que se hace es reemplazar el antígeno "a" (Ag_a) con el anticuerpo (Ab), es decir su estructura es copiada al antígeno y por lo tanto éste tiene ahora su valor del makespan.

- En la figura 4.2(6) se hace la comparación de la misma manera, pero ahora no es con Ag_a sino con Ag_b y no se utiliza el parámetro r, lo que provoca que únicamente Ag_b tenga el mejor resultado. Si Ab no es mejor que Ag_b se continúa al paso (8). De lo contrario no se continua al paso (7).

- En el paso (7) se reemplaza Ag_b con Ab y ésta es entonces la mejor solución al problema de optimización que se está resolviendo, hasta esta generación.

- En el paso (8) cualquiera que haya sido el resultado, excepto que se haya mejorado a alguno de los antígenos, se aplica una función $flip(\delta = 0.5)$ (la función

devuelve verdadero si al generar un número aleatorio en el rango 0.0 a 1.0, éste es menor que la probabilidad que se pasa como argumento; de lo contrario devuelve falso). Si el resultado es verdadero, se clona a la solución que está almacenada en Ag_a que es el Ab que probablemente se convirtió en Ag_a. De lo contrario se clona al individuo que previamente se había clonado. Es decir, el individuo que se clona posiblemente ha sufrido más de una mutación y ya no se parece mucho al Ab original. Este proceso nos da como resultado que en algunos clones se puedan hacer al menos 2 mutaciones y probablemente con esto consiga un mejor individuo. De no ocurrir esta mejora, al menos se logra una mayor variabilidad.

- En el paso (9) se tiene el proceso de mutación. En este proceso se aplica una mutación para permutaciones con repeticiones, pero ésta tiene la particularidad de haber sido diseñada para el problema que estamos atacando. La mutación se describe más adelante. La mutación se aplica con una probabilidad de $\frac{1}{m \times n}$. Con este valor se garantiza que sólo se aplicará la mutación a una posición de la estructura del Ab.

- En el paso (10), de los clones mutados, se selecciona el mejor de todos ellos. Para este caso, el que tenga un valor de makespan menor será elegido para pasar a la siguiente etapa. Este paso es el llamado maduración de afinidad.

- En esta parte únicamente se verifica que se cumpla la condición de que no se hayan completado todas las generaciones. Si la condición es falsa se regresa al paso (3). De lo contrario se continúa con el paso (12).

- En el último paso (12) se presenta el mejor resultado que es el que se encuentra en Ag_b. Además, se calculan las estadísticas del problema, que son: promedio de las mejores soluciones encontradas por el algoritmo y desviación típica de las mejores soluciones.

Se mencionaron entonces todos los pasos que componen el algoritmo mono-objetivo propuesto. Unicamente resta detallar las partes correspondientes a la decodificación y la mutación.
Para completar esta parte, todos los pasos utilizados en este algoritmo se muestran en pseudo código en el algoritmo 3.

4.2.1. Decodificación

La decodificación juega un papel importante en el algoritmo, ya que es el proceso que se encarga de interpretar la cadena del anticuerpo y representar la solución del JSSP. Entonces, si tenemos la cadena del anticuerpo y queremos ponerla en términos comprensibles debe decodificarse. Esto es precisamente el proceso que describiremos

Algoritmo 3 Sistema Inmune Artificial para resolver el JSSP mono-objetivo

Require: Ab, Ag_a, Ag_b, C, Ab^*, r, δ
Salida: f

$cargaProblema()$;
$Ag_a := Ag_b := generaAntigeno()$;
$Ab := generaAnticuerpo()$;.
repeat
  if $(Ab - r) > Ag_a$ **then**
    $Ag_a := Ab$;
    if $Ab > Ag_b$ **then**
      $Ag_b := Ab$;
    end if
  else
    if $flip(\delta)$ **then**
      $Ab := Ag_a$;
    end if
  end if
  $C := Clonar(Ab)$;
  $Ab^* := Mutar(C)$;
  $Ab := Selecciona(Ab^*)$;
until $i > p$
$f := decodifica(Ag_b)$;

a continuación.
Para llevar a cabo el proceso de decodificación, no solamente se necesita la cadena del anticuerpo, sino también la matriz que codifica al problema (podemos recordar en el capítulo anterior cómo se codifica el problema).
Ahora presentamos el pseudo código del algoritmo de decodificación en el algoritmo 4.

Algoritmo 4 Decodificación

```
estructura operacion{
int trabajo, inicio, final,
operacion * siguiente, *anterior
};
operacion plan[m][n];
int tmaquina[m], ttrabajo[n];
int cuerpo[m * n];
for i = 0 to m * n do
  ps := dameInformacionOperacion(cuerpo[i]);
  buscaPosicion(plan, ps, ttmaquina, ttrabajo);
  actualizaPlan(plan);
end for
ordenaOperaciones();
makespan = max(tmaquina);
```

El proceso de decodificación de la solución es la operación con mayor costo computacional. Para poder decodificar la solución se requiere de varias estructuras de datos. La principal estructura de datos que se requiere es una estructura que contiene la información de cada una de las operaciones *plan*. Dicha estructura como puede observarse en el algoritmo 4, consta de: trabajo al que corresponde la operación, tiempo de inicio, tiempo de finalización y ligas hacia la anterior y siguiente operación. Como se puede observar se está haciendo uso de una lista doblemente ligada dinámica, la cual va a almacenar el plan de trabajo, o sea la solución decodificada.
Una lista que contenga los tiempos que cada máquina emplea en procesar los trabajos *tmaquina*.
Una lista que contenga los tiempos que requiere cada uno de los trabajos a ser procesados por las máquinas *trabajo*.
Ahora podemos proceder a describir el proceso de decodificación. Primero se obtiene la información de cada una de las posiciones que están contenidas en el cuerpo a decodificar, que para el caso de nuestro algoritmo es un antígeno o un anticuerpo. Dicha información consta del trabajo que será procesado, qué operación de dicho trabajo se realizará, la máquina en que será procesada la operación y el tiempo de procesamiento de la operación. Esta información es almacenada en *ps*.

trabajo	máquina(tiempo)		
0	0(3)	1(3)	2(3)
1	0(2)	2(3)	1(4)
2	1(3)	0(2)	2(1)

2	1	2	0	0	0	1	2	1

Figura 4.3: Problema de ejemplo para decodificación

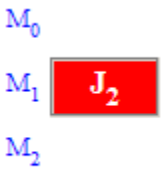
Una vez que se conoce la información de la operación, se verifica que tiene que cumplir con la condición de orden de procesamiento que le corresponde (este orden está previamente definido en el problema; véase la definición del problema). Una vez que se ha verificado el orden, se busca que haya espacio en el plan de trabajo para colocarse sin poder mover alguna otra operación que esté previamente posicionada. Es importante mencionar que para poder posicionar una operación en el plan de trabajo no se debe violar ninguna de las restricciones del problema.
Ya localizada la posición, se procede a colocar la operación y actualizar el plan de trabajo. La actualización del plan de trabajo consiste en actualizar la lista que contiene las operaciones y las listas que contienen los tiempos de las máquinas y de los trabajos.
En la decodificación se hace una búsqueda de espacios entre operaciones para poder utilizar mejor los recursos. Si existe un espacio que se pueda ocupar, es colocada ahí la operación teniendo cuidado de no violar ninguna de las restricciones del problema.

Una vez que se terminan los pasos anteriores para cada una de las operaciones del problema, se hace un ordenamiento del cuerpo, es decir ahora se ordenan las operaciones de cada trabajo, según su orden de procesamiento, con el fin de que la siguiente vez que se vuelva a decodificar con un cambio mínimo provocado por la mutación, la búsqueda de huecos sea menor.

Para comprender mejor todo lo anterior, se muestra el proceso de decodificación gráficamente. Nos ayudamos de un gráfico de Gantt, para su visualización.

En la figura 4.3 se muestra el problema y la cadena que codifica la solución. Lo que se necesita es interpretar la información contenida en la cadena mediante el proceso de decodificación. A continuación se muestran los pasos que sigue el proceso de decodificación.

Ahora explicaremos la versión gráfica de la decodificación, refiriéndonos a los puntos que se muestran en las figuras 4.4, 4.5 y 4.6.

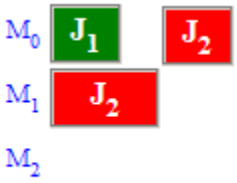


(1)

2	-	-	-	-	-	-	-	-

(2)

2	1	-	-	-	-	-	-	-



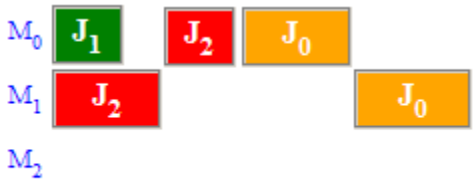
(3)

2	1	2	-	-	-	-	-	-

(4)

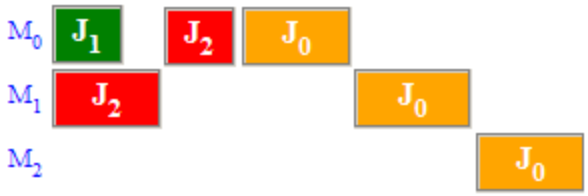
2	1	2	0	-	-	-	-	-

Figura 4.4: Proceso de decodificación



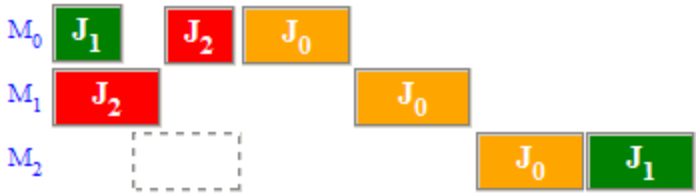
(5)

2	1	2	0	0	-	-	-	-



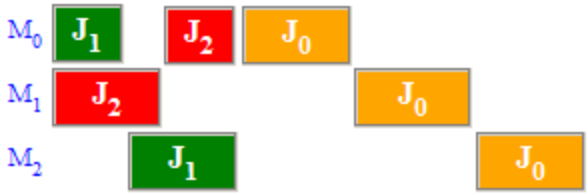
(6)

2	1	2	0	0	0	-	-	-



(7)

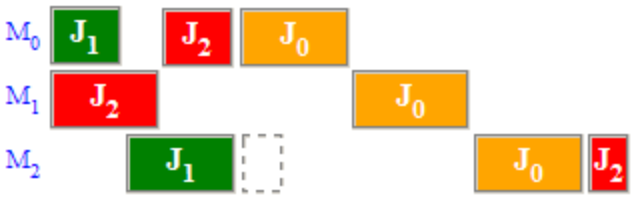
2	1	2	0	0	0	1	-	-



(8)

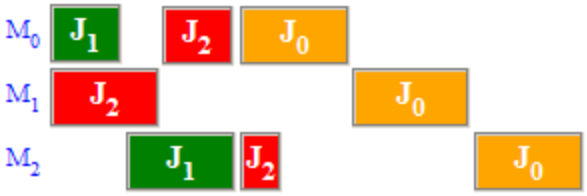
2	1	2	0	0	0	1	-	-

Figura 4.5: Proceso de decodificación (continuación)



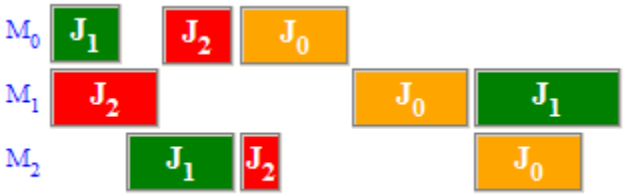
(9)

2	1	2	0	0	0	1	2	-



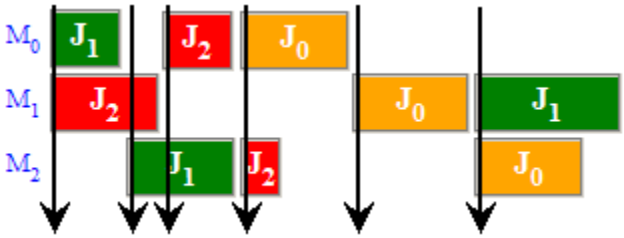
(10)

2	1	2	0	0	0	1	2	-



(11)

2	1	2	0	0	0	1	2	1



(12)

1	2	1	2	0	2	0	1	0

Figura 4.6: Proceso de decodificación (continuación)

(1) La primera posición de la cadena 2, corresponde a la 1^a operación de J_2 de acuerdo con lo visto en la figura 4.1.

(2) La segunda posición de la cadena 1, corresponde a la 1^a operación de J_1.

(3) La tercera posición de la cadena 2, corresponde a la 2^a operación de J_2.

(4) La cuarta posición de la cadena 0, corresponde a la 1^a operación de J_0.

(5) La quinta posición de la cadena 0, corresponde a la 2^a operación de J_0.

(6) La sexta posición de la cadena 0, corresponde a la 3^a operación de J_0.

(7) La séptima posición de la cadena 1, corresponde a la 2^a operación de J_1. Esta operación se coloca normalmente hasta después de la última operación en su máquina correspondiente y verificando que no se viole ninguna de las restricciones del problema. Observamos que hay huecos de tiempo en la máquina antes de la última operación y vemos que se puede colocar la operación que estamos posicionando, sin violar ninguna de las restricciones del problema.

(8) La operación anterior se mueve al hueco que existía y entonces se está utilizando de una mejor manera el tiempo de las máquinas, para este caso M_2.

(9) La octava posición de la cadena 2, corresponde a la 3^a operación de J_2. También podemos observar que existe un hueco antes de la última operación y que en ese hueco se puede colocar la operación, sin violar ninguna de las restricciones del problema.

(10) La operación anterior es movida al hueco que existía y de nuevo se utiliza de una mejor manera el tiempo de las máquinas.

(11) La novena operación de la cadena 1, corresponde a la 3^a operación de J_1. Esta es colocada normalmente, ya que no existe algún hueco en el cual se pueda colocar sin afectar a las otras operaciones previamente situadas.

(12) Finalmente se ordena la cadena con el orden en el que se sitúan las operaciones de los trabajos, con la finalidad de que en la siguiente decodificación ésta no tenga que buscar huecos, ya que el hecho de ordenar la cadena las reduce.

Como se puede observar, el proceso de decodificación utiliza una estrategia para optimizar el tiempo de las máquinas, evitando dejar huecos y reduciendo el makespan del plan de trabajo final.

Este es un buen aditamento al algoritmo, ya que esta estrategia está dentro del proceso de decodificación y su costo es muy bajo, y además de tener una gran funcionalidad en el proceso de optimización del JSSP.

Lo que se obtiene como resultado de ordenar la cadena al final, es que varias cadenas con diferente orden de sus elementos pueden ser decodificadas en la misma solución, esto debido a la estrategia que aplicamos para decodificar buscando huecos en el plan de trabajo.

4.2.2. Clonación

Esta parte del algoritmo se encarga de hacer copias del anticuerpo, es decir se hacen copias identicas en estructura.

4.2.3. Mutación

El proceso de mutación es el que nos va a dar la diversidad de soluciones en el algoritmo. La mutación en este caso se considera como un operador primario. Es decir que su uso es necesario para poder generar nuevos individuos a partir de un anticuerpo dado.
Se utilizan dos operadores de mutación, los cuales están diseñados para permutaciones con repeticiones y generalmente son utilizados para problemas de optimización combinatoria. Otra de sus particularidades es que están especialmente diseñados para el JSSP. Se dice que son especialmente diseñados para el JSSP, ya que hacen los cambios en función del orden de las operaciones y por cada máquina.
El impacto que tiene la mutación en un anticuerpo para generar un nuevo individuo es mínima, ya que se aplica de tal forma que sólo se realiza un cambio en la cadena. Se utiliza un porcentaje de mutación de $\frac{1}{m \times n}$.

Ahora se muestran en el algoritmo 5 y algoritmo 6 las dos versiones de las mutaciones.

En general las dos mutaciones son muy similares en cuanto a su funcionamiento, salvo la *mutación A*, que hace una reparación del orden de las operaciones en las máquinas.

Los pasos a seguir son los siguientes:

- Se aplica una función $flip(pm)$ para cada una de las posiciones del Ab. El valor de pm es el porcentaje de mutación. Para todas las pruebas que se realizaron se tomó el valor de $\frac{1}{m \times n}$.

Algoritmo 5 Mutación A

```
for i = 0 to m * n do
  if flip(pm) then
    ps := dameInformacionOperacion(Ab[i]);
    np := otraPosicionaAleatoria(ps);
    intercambiaPosiciones(ps, ns);
    reparaOrden(Ab, ps, ns);
  end if
end for
```

Algoritmo 6 Mutación B

```
for i = 0 to m * n do
  if flip(pm) then
    ps := dameInformacionOperacion(Ab[i]);
    np := otraPosicionaAleatoria(ps);
    intercambiaPosiciones(ps, ns);
  end if
end for
```

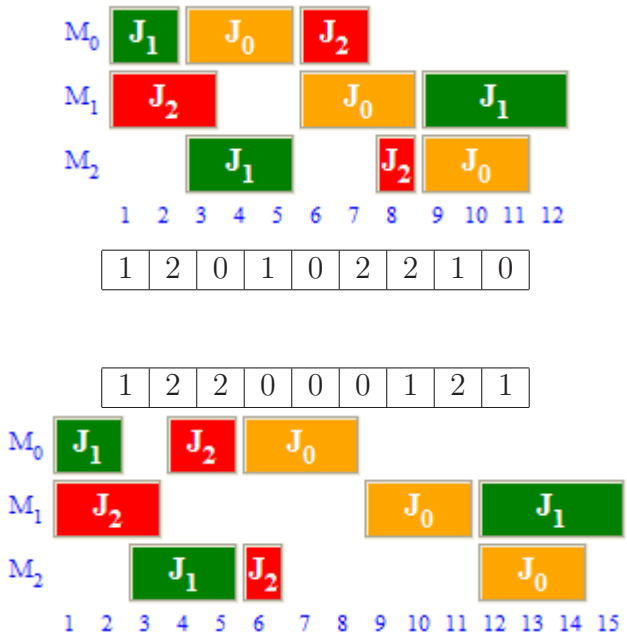


Figura 4.7: Proceso de mutación A

- Una vez que se seleccionó una posición, se obtiene su información que es posición y máquina. Esta información es almacenada en *ps*.

- Se selecciona al azar otra operación que esté en la misma máquina, esto es que la operación será de otro trabajo, pero se moverá dentro de la misma máquina.

- Se intercambian las operaciones de posición. Para este caso únicamente cambian los valores que se seleccionaron y no hay más operaciones. Termina el proceso de mutación. Se puede ver un ejemplo de la *mutación B* en la figura 4.8.

- Este proceso de reparación de orden, se aplica exclusivamente al proceso de *mutación A*. El proceso de ordenamiento consiste en que una vez que se cambiaron las posiciones de las operaciones se tiene que tomar en cuenta, que como toda la cadena representa todas las operaciones en todas las máquinas y éstas dependen de su orden de aparición y posición en el plan de trabajo, se puede afectar a la tarea de esa máquina y cambiar su orden de aparición y provocar que se convierta en otra operación del mismo trabajo. Lo que se hace es mover realmente todas las operaciones de ese trabajo de tal forma que se aplique a la operación que se había seleccionado previamente. Esto hace que se recorran las otras operaciones de éste y los demás trabajos. Un ejemplo de operación de la *mutación A* se puede ver en la figura 4.7.

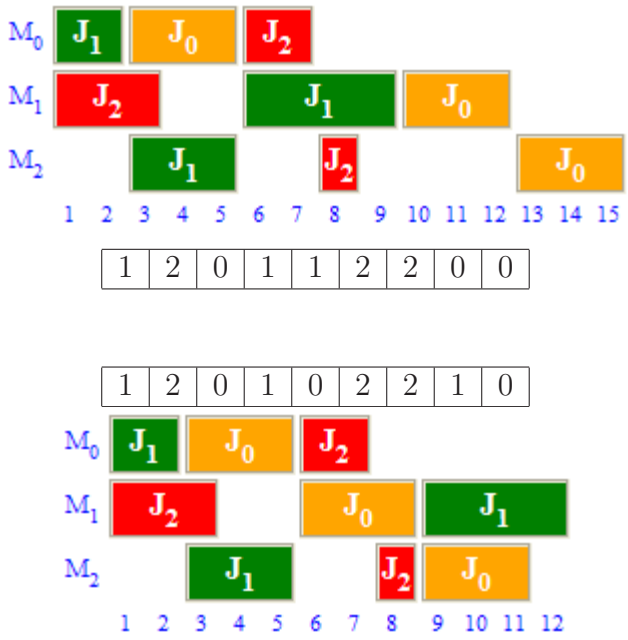


Figura 4.8: Proceso de mutación B

4.2.4. Selección

El proceso de selección también tiene un papel importante en el algoritmo, ya que éste se encarga de seleccionar el mejor de los clones; aquí solamente se selecciona al que tiene la mejor aptitud. La aptitud para nuestro trabajo es el valor del *makespan* y se está minimizando, lo que implica que el individuo que tenga el valor de *makespan* más pequeño será el seleccionado.

Se han descrito las partes que componen el algoritmo que se presenta en este trabajo. La finalidad de haber descrito a detalle las partes que lo componen es para que alguna persona interesada pueda reproducir los resultados que presentamos en el capítulo siguiente.

4.3. Algoritmo Multiobjetivo

Ahora toca el turno de describir el algoritmo que utilizamos para optimizar el JSSP multiobjetivo. El algoritmo está basado en el algoritmo mono-objetivo, aunque obviamente tiene algunas variaciones, dado que en la versión multiobjetivo no se busca una solución única sino que se tiene un conjunto de soluciones igualemente buenas[1] (cada

[1] A este conjunto de soluciones igualmente buenas se les conoce como no dominadas. Si no existe ninguna solución que las puede dominar, entonces se dice que forman parte del conjunto de óptimos

una de ellas puede ser considerada como una solución al problema). Esta característica es llevada a cabo gracias a que ninguna de ellas domina a las demás soluciones en todos los objetivos a optimizar (de ahí que se llamen no dominadas).

Como se mencionó en el capítulo del JSSP, el problema tiene múltiples objetivos que pueden ser optimizados por separado o bien considerarse juntos para hacer el problema multiobjetivo. Para fines de esta parte del trabajo solamente nos concentraremos a describir las diferencias del algoritmo multiobjetivo y el mono-objetivo.

Ahora veremos el algoritmo en pseudo código del algoritmo en la versión multiobjetivo y describiremos las partes en las que se diferencía de la versión mono-objetivo.

Algoritmo 7 Sistema Inmune Artificial para resolver el JSSP multiobjetivo

Require: $\overline{Ag}$, Ab, C, Ab^*, δ
Salida: f

$cargaProblema()$;
$Ab := generaCuerpo()$;
repeat
  if $Ab \prec \overline{Ag}$ **then**
    $Agrega(\overline{Ag}, Ab)$;
  else
    if $flip(\delta)$ **then**
      $Ab := Elemento(\overline{Ag}, rand())$;
    end if
  end if
  $C := Clonar(Ab)$;
  $Ab^* := Mutar(C)$;
  $Ab := Selecciona(Ab^*)$;
until $i > p$
$\overline{f} := decodifica(\overline{Ag})$;

Los objetivos a optimizar para fines de este trabajo son el *makespan* y el *mean flow time*. La modificación se hace en la decodificación, ya que en lugar de solamente calcular el *makespan*, ahora también calcula el *mean flow time*.
Ahora describiremos el algoritmo 7, que corresponde a la versión multiobjetivo. Como se puede observar, las principales diferencias con respecto a la versión mono-objetivo son mínimas. Una de las principales diferencias es que en esta versión no se cuenta con el mecanismo de ir guardando la mejor solución en un punto, esto es debido a que se

de Pareto.

tienen varias soluciones y todas éstas se guardan dentro en un conjunto de soluciones factibles que forman el frente de Pareto.
Funciona de la misma manera que el anterior, excepto porque en este caso no se compara el anticuerpo contra un solo antígeno, sino que ahora se verifica que el anticuerpo tenga una solución no dominada con respecto al conjunto de soluciones que son los antígenos. Al obtener los valores de las funciones objetivo correspondientes a este conjunto de soluciones no dominadas ($\overline{Ag}$) se obtiene el frente de pareto del problema.

Para tener una referencia de optimización multiobjetivo favor de referirse al Apéndice A.

Los operadores de clonación, mutación y selección son idénticos en las dos versiones del algoritmo. El tamaño del conjunto de soluciones no dominadas no es fijo tampoco se limita su crecimiento. Varios de los detalles de la implementación se discuten en el capítulo 5.

4.4. Salida del programa

Al ser ejecutado el algoritmo, se le puede especificar el formato la salida, es decir, podemos ver los resultados de la mejor solución conocida. Esta solución que es la salida al problema que hemos estados usando en este capítulo (véase la figura 4.2.1), está formada de la siguiente manera:

```
3 3
1 2 0 1 0 2 2 1 0
12
1 1 2
0 3 5
2 6 7
2 1 3
0 6 8
1 9 12
1 3 5
2 8 8
0 9 11
#
```

En el primer renglón se muestra el tamaño del problema; recordemos que es un problema de 3×3. En el segundo renglón se muestra la cadena que codifica la solución. En el tercer renglón se muestra para la versión mono-objetivo el valor del *makespan* y para la versión multiobjetivo el valor del *makespan* y el *mean flow time*.

Los siguientes renglones, están agrupados por máquina, es decir cada m renglones corresponde a cada una de las máquinas. El primer grupo a la máquina 1, el segundo grupo a la máquina 2 y así sucesivamente. Cada uno de los renglones tiene 3 valores. El primer valor corresponde al trabajo. El segundo valor corresponde al tiempo de inicio para la operación en la máquina. Y el tercer valor corresponde al tiempo de finalización para la operación en la máquina.
Con esta información es fácil ver la solución, pero aún así ver todos estos números puede ser engorroso. Por tal motivo se escribió un código en lenguaje *perl* que convierte esta salida en un documento HTML que contiene una gráfica de Gantt. Con la gráfica de Gantt es mucho más sencillo ver el resultado y la forma en como están acomodadas las operaciones de los trabajos en las máquinas.
Para tener acceso a este código refiérase al Apéndice B.

4.4.1. Otras utilerías

Para múltiples pruebas que se realizaron con los resultados obtenidos, se encontró un software llamado *Lekin* que es dirigido por el profesor Michael Pinedo, el profesor Xiuli Chao y el profesor Joseph Leung. Este trabajo esta desarrollado con fines educativos y se puede encontrar en: http://www.stern.nyu.edu/om/software/lekin/.
Es una herramienta muy completa para análisis de problemas de planificación de recursos. Es por eso que desarrollamos código en *perl* para poder pasar los problemas y los resultados que utilizamos en nuestro trabajo y visualizarlos en *Lekin*. Refierase al Apéndice C para más detalles.

Capítulo 5

Problemas de Prueba y Análisis de Resultados

En este capítulo se presentan los resultados que se obtuvieron con los problemas seleccionados de la literatura especializada. Se presentan los problemas y las estadísticas de la soluciones encontradas. Así mismo se hace una comparación contra varios trabajos encontrados en la literatura, los cuales son de los más recientes y tienen una calidad de resultados bastante buena. Al final del capítulo podremos concluir que nuestro algoritmo es una opción competitiva en la optimización del JSSP. La comparación de resultados de nuestro algoritmo se hace contra GRASP, Algoritmos Genéticos, Algoritmos Genéticos Híbridos y Búsqueda Tabú.

5.1. Métricas y Contendientes

Para poder medir la eficiencia de nuestro algoritmo nos basamos en el número de evaluaciones a la función objetivo, que en este caso es la decodificación del anticuerpo en una solución al JSSP. También nos basamos en la calidad de las soluciones, es decir, el valor de la solución obtenido con respecto al mejor resultado conocido. Así mismo se muestran estadísiticas como el valor promedio y la desviación estándar de la mejor solución obtenida de todas las ejecuciones del algoritmo que se hicieron para cada uno de los problemas. Estos resultados únicamente se presentan para los problemas realizados con nuestro algoritmo, y para las otras técnicas contra las cuales se hace la comparación, únicamente se muestran los resultados que los autores presentan así como algunos datos relevantes.

5.2. Descripción de los problemas

El conjunto de funciones de prueba que se utilizó para medir la eficiencia de nuestro algoritmo, consta de 162 problemas divididos en 7 clases. Estos se describen a continuación:

FT 3 problemas de 3 diferentes tamaños propuestos Fisher y Thompson [29]:6 $\times$ 6, 10 $\times$ 10 y 20 $\times$ 5. El nombre *FT* fue dada por Applegate y Cook [3]. Los tiempos de procesamiento fueron generados en el intervalo $[1, 10]$ para *ft06* y en el intervalo $[1, 99]$ para *ft10* y *ft20*. Para simular un problema típico en *ft10* y *ft20*, las máquinas con los números más pequeños le son asignadas las primeras operaciones y a las máquinas con los números más grandes se le asignan las últimas operaciones.

LA 40 problemas de 8 diferentes tamaños propuestos por Lawrence [49]: 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10 y 15×15. Lawrence [49] llamó *Fl-5*, *Gl-5*, *Hl-5*, *Al-5*, *Bl-5*, *Cl-5*, *Dl-5* y *Il-5* a las instancias respectivamente. Sin embargo el nombre *LA* fué dado por Applegate y Cook [3] y es uno de los más comunmente utilizados. Los tiempos de procesamiento fueron generados en el intervalo $[5, 99]$.

ABZ 5 problemas de 2 diferentes tamaños propuestos por Adams, Balas y Zawack [2]: 10×10 y 20×15. El nombre ABZ fué dado por Applegate y Cook [3]. Los tiempos de procesamiento de ABZ5, ABZ6 y ABZ(7-9) fueron generados en los intervalos $[50, 100]$, $[25, 100]$ y $[11, 40]$ respectivamente.

ORB 10 problemas usados por Applegate y Cook [3]: 10×10. Fueron formulados en Bonn en 1986 y son caracterizados como problemas "difíciles". El nombre *ORB* fué dado por Applegate y Cook [3].

SWV 20 problemas de 4 diferentes tamaños propuestos por Storer, Wu y Vaccari [79]: 20×10, 20×15, 50×10 y 50×10. El nombre SWV ha sido dado por Vaessens [76]. Los tiempos de procesamiento son generados en el intervalo $[1, 100]$. Estas instancias son consideradas difícil, difícil, difícil y fácil. En estos problemas el conjunto de máquinas es divido en k $(1 \leq k \leq m)$ subconjuntos del mismo tamaño. Una secuencia de precedencia es generada pasando el trabajo a través de toda la máquina de un sistema usando asignaciones uniformemente distribuidas antes de que llegue a una máquina del siguiente conjunto. Para el conjunto fácil $k = 1$, mientras que para las instancias difíciles $k = 2$.

YN 4 problemas propuestos por Yamada y Nakano [95]: 20×20. El nombre *YN* fue dado por Vaessens [76]. Los tiempos de procesamiento fueron generados en el intervalo $[10, 50]$.

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
ft06	6	6	55	55	55	0.00	0.00	1969	[7]
ft10	10	10	930	930	936	0.00	0.65	1989/1984	[17] / [48]
ft20	20	5	1165	1165	1165	0.00	0.00	1989/1975	[17] / [55]

Cuadro 5.1: Problemas pertenecientes a la clase **FT**

TA 80 problemas de 8 diferentes tamaños propuestos por Taillard [81]: 15×15, 20×15, 20×20, 30×15, 30×20, 50×15, 50×20 y 100×20. El nombre *TA* fue usado por Taillard. Los tiempos de procesamiento fueron generados en el intervalo $[1, 99]$.

5.2.1. Detalles de los problemas

Ahora se presenta a detalle cada una de las instancias de los problemas que se tomaron de [8], junto con los mejores resultados encontrados en múltiples trabajos realizados para optimizar el JSSP. Se muestran también los resultados obtenidos con nuestro algoritmo, así como la desviación de la mejor solución encontrada contra la mejor solución teórica, tanto de los otros trabajos como de nuestro trabajo. Las principales características de cada uno de los problemas: el nombre del problema (Problema), tamaño (n, m), límite inferior (LB[1]) y límite superior (UB[2]), la desviación que existe entre el límite inferior y el valor mínimo del makespan calculado teóricamente, la fecha en que se publicó y quién lo reporta.

Por simplicidad y manejo de espacio, los problemas se muestran agrupados por clase y tamaño.

Es importante mencionar que esta comparación es únicamente de valores, ya que sería un trabajo muy extenso comparar cada uno de los resultados, ya que cada trabajo corresponde a técnicas diferentes.

5.3. Resultados

En esta sección se muestran los detalles de los resultados obtenidos con los problemas previamente descritos. La prueba realizada con cada uno de los problemas consta de 20 ejecuciones del algoritmo. Se resume al final los resultados obtenidos, para este caso el makespan, la evaluación en la que se encontró, y las estadísticas de todas las ejecuciones del algoritmo.

[1]Por sus siglas en inglés: Lower Bound
[2]Por sus siglas en inglés: Upper Bound

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
la01	10	5	666	666	666	0.00	0.00	1988	[2]
la02	10	5	655	655	655	0.00	0.00	1988/1988	[2] / [54]
la03	10	5	597	597	597	0.00	0.00	1991/1988	[3] / [54]
la04	10	5	590	590	590	0.00	0.00	1991/1988	[3] / [54]
la05	10	5	593	593	593	0.00	0.00	1988	[2]
la06	15	5	926	926	926	0.00	0.00	1988	[2]
la07	15	5	890	890	890	0.00	0.00	1988	[2]
la08	15	5	863	863	863	0.00	0.00	1988	[2]
la09	15	5	951	951	951	0.00	0.00	1988	[2]
la10	15	5	958	958	958	0.00	0.00	1988/1992	[2] / [91]
la11	20	5	1222	1222	1222	0.00	0.00	1988	[2]
la12	20	5	1039	1039	1039	0.00	0.00	1988	[2]
la13	20	5	1150	1150	1150	0.00	0.00	1988	[2]
la14	20	5	1292	1292	1292	0.00	0.00	1988	[2]
la15	20	5	1207	1207	1207	0.00	0.00	1988	[2]
la16	10	10	945	945	945	0.00	0.00	1990	[18]
la17	10	10	784	784	784	0.00	0.00	1990/1988	[18] / [54]
la18	10	10	848	848	848	0.00	0.00	1991/1988	[3] / [54]
la19	10	10	842	842	842	0.00	0.00	1991/1988	[3] / [54]
la20	10	10	902	902	907	0.00	0.55	1991/1992	[3] / [91]
la21	15	10	1046	1046	1046	0.00	0.00	1996	[76]
la22	15	10	927	927	927	0.00	0.00	1991/1988	[3] / [54]
la23	15	10	1032	1032	1032	0.00	0.00	1988	[2]
la24	15	10	935	935	935	0.00	0.00	1991	[3]
la25	15	10	977	977	979	0.00	0.20	1991	[3]
la26	20	10	1218	1218	1218	0.00	0.00	1988/1988	[2] / [54]
la27	20	10	1235	1235	1240	0.00	0.40	1988/1994	[2] / [19]
la28	20	10	1216	1216	1216	0.00	0.00	1988/1988	[2] / [54]
la29	20	10	1152	1152	1170	0.00	1.56	1996	[52]
la30	20	10	1355	1355	1355	0.00	0.00	1988	[2]
la31	30	10	1784	1784	1784	0.00	0.00	1988	[2]
la32	30	10	1850	1850	1850	0.00	0.00	1988	[2]
la33	30	10	1719	1719	1719	0.00	0.00	1988	[2]
la34	30	10	1721	1721	1721	0.00	0.00	1988	[2]
la35	30	10	1888	1888	1888	0.00	0.00	1988	[2]
la36	15	15	1268	1268	1281	0.00	1.03	1990	[18]
la37	15	15	1397	1397	1408	0.00	0.79	1990/1991	[18] / [3]
la38	15	15	1196	1196	1204	0.00	0.67	1995/1996	[76] / [64]
la39	15	15	1233	1233	1249	0.00	1.30	1991	[3]
la40	15	15	1222	1222	1228	0.00	0.49	1991	[3]

Cuadro 5.2: Problemas pertenecientes a la clase **LA**

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
abz5	10	10	1234	1234	1234	0.00	0.00	1991	[3]
abz6	10	10	943	943	943	0.00	0.00	1991/1988	[3] / [2]
abz7	20	15	656	656	674	0.00	2.74	1996/1996	[53] / [53]
abz8	20	15	646	665	680	2.94	5.26	1999/1996	[14] / [52]
abz9	20	15	662	679	692	2.57	4.53	1999/1995	[14] / [6]

Cuadro 5.3: Problemas pertenecientes a la clase **ABZ**

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
orb01	10	10	1059	1059	1059	0.00	0.00	1991	[3]
orb02	10	10	888	888	889	0.00	0.11	1991	[3]
orb03	10	10	1005	1005	1005	0.00	0.00	1991	[3]
orb04	10	10	1005	1005	1011	0.00	0.60	1991	[3]
orb05	10	10	887	887	889	0.00	0.23	1991	[3]
orb06	10	10	1010	1010	1013	0.00	0.30	1991	[3]
orb07	10	10	397	397	397	0.00	0.00	1991	[3]
orb08	10	10	899	899	899	0.00	0.00	1991	[3]
orb09	10	10	934	934	934	0.00	0.00	1991	[3]
orb10	10	10	944	944	944	0.00	0.00	1991	[3]

Cuadro 5.4: Problemas pertenecientes a la clase **ORB**

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
swv01	20	10	1407	1407	1439	0.00	2.27	1996	[52]
swv02	20	10	1475	1475	1515	0.00	2.71	1995/1996	[90] / [90]
swv03	20	10	1398	1398	1463	0.00	4.65	1999/1996	[14] / [90]
swv04	20	10	1450	1474	1519	1.66	4.76	1996/2002	[90] / [64]
swv05	20	10	1424	1424	1465	0.00	2.88	1996	[52]
swv06	20	15	1591	1673	1749	5.15	9.93	1996/2003	[90] / [35]
swv07	20	15	1447	1600	1638	10.57	13.20	1999/2002	[14] / [64]
swv08	20	15	1641	1759	1863	7.19	13.53	1999/2002	[14] / [35]
swv09	20	15	1605	1661	1776	3.49	10.65	1999/2002	[14] / [64]
swv10	20	15	1632	1761	1859	7.90	13.91	1999/2002	[14] / [35]
swv11	50	10	2983	2983	3154	0.00	5.73	1996/2002	[90] / [64]
swv12	50	10	2972	3003	3156	1.04	6.19	1996/1997	[90] / [86]
swv13	50	10	3104	3104	3286	0.00	5.86	1996/1997	[90] / [86]
swv14	50	10	2968	2968	3041	0.00	2.46	1996/1995	[90] / [6]
swv15	50	10	2885	2903	3121	0.62	8.18	1996/2002	[90] / [35]
swv16	50	10	2924	2924	2924	0.00	0.00	1996/1992	[90] / [80]
swv17	50	10	2794	2794	2794	0.00	0.00	1996/1992	[90] / [80]
swv18	50	10	2852	2852	2852	0.00	0.00	1996/1992	[90] / [80]
swv19	50	10	2843	2843	2843	0.00	0.00	1996/1992	[90] / [80]
swv20	50	10	2823	2823	2823	0.00	0.00	1996/1992	[90] / [80]

Cuadro 5.5: Problemas pertenecientes a la clase **SWV**

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
yn1	20	20	846	885	894	4.61	5.67	1999/2002	[14] / [64]
yn2	20	20	870	909	922	4.48	5.98	1999/1996	[14] / [90]
yn3	20	20	840	892	901	6.19	7.26	1999/2002	[14] / [64]
yn4	20	20	920	968	987	5.22	7.28	1999/1997	[14] / [86]

Cuadro 5.6: Problemas pertenecientes a la clase **YN**

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
ta01	15	15	1231	1231	1249	0.00	1.46	1994	[82]
ta02	15	15	1244	1244	1259	0.00	1.21	1995/1993	[90] / [64]
ta03	15	15	1218	1218	1231	0.00	1.07	1999/1995	[14] / [6]
ta04	15	15	1175	1175	1199	0.00	2.04	1999/1995	[14] / [94]
ta05	15	15	1224	1224	1237	0.00	1.06	1999/1999	[14] / [14]
ta06	15	15	1238	1238	1251	0.00	1.05	1999/1999	[14] / [14]
ta07	15	15	1227	1227	1247	0.00	1.63	1999/1999	[14] / [14]
ta08	15	15	1217	1217	1224	0.00	0.58	1999/1995	[14] / [6]
ta09	15	15	1274	1274	1301	0.00	2.12	1999/1995	[14] / [6]
ta10	15	15	1241	1241	1257	0.00	1.29	1995/1995	[90] / [6]
ta11	20	15	1323	1358	1402	2.64	5.97	2000/2003	[78] / [35]
ta12	20	15	1351	1367	1413	1.18	4.59	2000/1995	[78] / [6]
ta13	20	15	1282	1342	1382	4.68	7.80	2000/2002	[78] / [35]
ta14	20	15	1345	1345	1365	0.00	1.49	1995/1996	[90] / [64]
ta15	20	15	1304	1340	1373	2.76	5.29	2000/2002	[78] / [35]
ta16	20	15	1302	1360	1394	4.45	7.07	2000/2000	[78] / [35]
ta17	20	15	1462	1462	1498	0.00	2.46	2000/2002	[78] / [64]
ta18	20	15	1369	1396	1457	1.97	6.43	1995/1996	[90] / [6]
ta19	20	15	1297	1335	1407	2.93	8.48	2000/2001	[78] / [64]
ta20	20	15	1318	1351	1391	2.50	5.54	2000/2001	[78] / [64]
ta21	20	20	1539	1644	1695	6.82	10.14	1995/2002	[90] / [64]
ta22	20	20	1511	1600	1664	5.89	10.13	1995/2001	[90] / [64]
ta23	20	20	1472	1557	1615	5.77	9.71	1995/2001	[90] / [64]
ta24	20	20	1602	1647	1688	2.80	5.37	1995/2002	[90] / [64]
ta25	20	20	1504	1595	1657	6.05	10.17	1995/2002	[90] / [64]
ta26	20	20	1539	1645	1703	6.88	10.66	1995/2002	[90] / [64]
ta27	20	20	1616	1680	1747	3.96	8.11	1995/2001	[90] / [64]
ta28	20	20	1591	1614	1649	1.44	3.65	1995/2002	[90] / [64]
ta29	20	20	1514	1625	1667	7.33	10.11	1995/1996	[90] / [82]
ta30	20	20	1472	1584	1637	7.61	11.21	1995/2001	[90] / [64]
ta31	30	15	1764	1764	1815	0.00	2.89	1994/1999	[82] / [1]
ta32	30	15	1774	1796	1872	1.24	5.52	1994/2002	[82] / [35]
ta33	30	15	1778	1793	1880	0.84	5.74	1995/2002	[90] / [64]
ta34	30	15	1828	1829	1921	0.05	5.09	1994/2001	[82] / [64]
ta35	30	15	2007	2007	2007	0.00	0.00	1995/1994	[90] / [82]
ta36	30	15	1819	1819	1881	0.00	3.41	1995/1999	[90] / [1]
ta37	30	15	1771	1778	1843	0.39	4.07	1994/2002	[82] / [64]
ta38	30	15	1673	1673	1765	0.00	5.50	1994/2000	[82] / [35]
ta39	30	15	1795	1795	1839	0.00	2.45	1995/1999	[90] / [1]
ta40	30	15	1631	1674	1737	2.64	6.50	1995/2001	[90] / [64]

Cuadro 5.7: Problemas pertenecientes a la clase **TA**

Problema	n	m	LB	UB	AIS	Desv	dAIS	Fecha (LB/UB)	Referencia (LB/UB)
ta41	30	20	1859	2014	2163	8.34	16.35	1995/2003	[90] / [35]
ta42	30	20	1867	1956	2032	4.77	8.84	1995/2001	[90] / [64]
ta43	30	20	1809	1859	1974	2.76	9.12	1995/2002	[90] / [64]
ta44	30	20	1927	1984	2087	2.95	8.30	1995/2002	[90] / [64]
ta45	30	20	1997	2000	2074	0.15	3.86	1995/2001	[90] / [64]
ta46	30	20	1940	2016	2125	3.92	9.54	1994/2003	[82] / [35]
ta47	30	20	1789	1903	2016	6.37	12.69	1995/2002	[90] / [64]
ta48	30	20	1912	1952	2059	2.09	7.69	1995/2002	[90] / [64]
ta49	30	20	1915	1968	2057	2.77	7.42	1995/2002	[90] / [35]
ta50	30	20	1807	1926	2047	6.58	13.28	1995/2003	[90] / [16]
ta51	50	15	2760	2760	2780	0.00	0.72	1994/1994	[82] / [82]
ta52	50	15	2756	2756	2756	0.00	0.00	1994/1994	[82] / [82]
ta53	50	15	2717	2717	2719	0.00	0.07	1994/1994	[82] / [82]
ta54	50	15	2839	2839	2839	0.00	0.00	1994/1994	[82] / [82]
ta55	50	15	2679	2679	2731	0.00	1.94	1994/1993	[82] / [64]
ta56	50	15	2781	2781	2781	0.00	0.00	1994/1994	[82] / [82]
ta57	50	15	2943	2943	2943	0.00	0.00	1994/1994	[82] / [82]
ta58	50	15	2885	2885	2885	0.00	0.00	1994/1994	[82] / [82]
ta59	50	15	2655	2655	2679	0.00	0.90	1994/1994	[82] / [82]
ta60	50	15	2723	2723	2761	0.00	1.40	1994/1994	[82] / [82]
ta61	50	20	2868	2868	2955	0.00	3.03	1994/1993	[82] / [64]
ta62	50	20	2869	2869	3045	0.00	6.13	1995/2003	[90] / [16]
ta63	50	20	2755	2755	2926	0.00	6.21	1994/1993	[82] / [64]
ta64	50	20	2702	2702	2797	0.00	3.52	1995/1993	[6] / [64]
ta65	50	20	2725	2725	2787	0.00	2.28	1994/1993	[82] / [64]
ta66	50	20	2845	2845	2968	0.00	4.32	1994/1993	[82] / [64]
ta67	50	20	2825	2825	2887	0.00	2.19	1995/1999	[90] / [1]
ta68	50	20	2784	2784	2884	0.00	3.59	1995/1993	[6] / [64]
ta69	50	20	3071	3071	3071	0.00	0.00	1994/1993	[82] / [64]
ta70	50	20	2995	2995	3123	0.00	4.27	1994/1993	[82] / [64]
ta71	100	20	5464	5464	5464	0.00	0.00	1994/1994	[82] / [82]
ta72	100	20	5181	5181	5181	0.00	0.00	1994/1994	[82] / [82]
ta73	100	20	5568	5568	5568	0.00	0.00	1994/1994	[82] / [82]
ta74	100	20	5339	5339	5339	0.00	0.00	1994/1994	[82] / [82]
ta75	100	20	5392	5392	5392	0.00	0.00	1994/1994	[82] / [82]
ta76	100	20	5342	5342	5342	0.00	0.00	1994/1994	[82] / [82]
ta77	100	20	5436	5436	5436	0.00	0.00	1994/1994	[82] / [82]
ta78	100	20	5394	5394	5394	0.00	0.00	1994/1994	[82] / [82]
ta79	100	20	5358	5358	5358	0.00	0.00	1994/1994	[82] / [82]
ta80	100	20	5183	5183	5183	0.00	0.00	1994/1993	[82] / [64]

Cuadro 5.8: Problemas pertenecientes a la clase **TA**(continuación)

Los resultados de los problemas se muestran en el orden que se describieron previamente.
Las medidas estadísticas que se reportan son el promedio $\overline{x}$, ls varianza S^2 y la desviación estándar S. Todas las medidas son realizadas con al mejor resultado de cada una de las ejecuciones del algoritmo.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$S^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \overline{x})^2$$

$$S = \sqrt{S^2}$$

5.3.1. Resultados a los problemas de la clase FT

Los resultados de la clase **FT** se muestran en el cuadro 5.9. Estos problemas son de los más populares en cuanto al JSSP se refiere. Son 3 pero solamente en **FT10** y **FT20** encontramos dificultad para resolverlos, ya que **FT06** es muy sencillo de resolver, dado su tamaño y características.

5.3.2. Resultados a los problemas de la clase LA

Los problemas de la clase **LA** son de los más famosos en la literatura, ya que son muy variados en cuanto a tamaño y complejidad. Los resultados de éstos se muestran en los cuadros 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16 y 5.17.

5.3.3. Resultados a los problemas de la clase ORB

Los problemas de la clase **ORB**, algunos se pueden considerar difíciles. Los resultados se pueden observar en los cuadros 5.18 y 5.19.

5.3.4. Resultados a los problemas de la clase SWV

De entre los problemas de la clase **SWV**, los 3 primeros grupos de 5 problemas son difíciles, pero los últimos 5 son verdaderamente sencillos de resolver.
Los resultados de esta clase **SWV** se muestran en los cuadro 5.20, 5.21, 5.22 y 5.23.

5.3.5. Resultados a los problemas de la clase YN

Los problemas de la clase **YN** son difíciles, como se puede apreciar en nuestros resultados que quedan muy por debajo de lo deseable.
Los resultados se muestran en el cuadro 5.24.

	$FT06$		$FT10$		$FT20$	
#	E	C_{max}	E	C_{max}	E	C_{max}
0	453	58	234265	949	493679	1173
1	498	56	33710	1016	137159	1180
2	433	58	3162	967	98019	1178
3	64	55	3421	1004	75550	1175
4	93	55	190041	1001	511329	1173
5	204	55	369978	973	36997	1177
6	21	55	110490	967	59393	1173
7	483	57	60419	1000	55620	1178
8	51	55	107572	983	99912	1165
9	217	55	21912	1003	56794	1178
10	402	58	177302	967	547883	1178
11	245	55	55095	992	213191	1178
12	269	55	10294	953	82676	1178
13	135	55	90008	967	442816	1165
14	395	55	1422	1000	122880	1180
15	183	57	8142	970	46412	1175
16	269	56	62507	967	573509	1178
17	280	56	208374	993	14916	1214
18	5	58	65345	943	74650	1173
19	477	55	195213	936	135856	1180

Problema	$\overline{x}$	S^2	S
ft06	55.95	1.44	1.20
ft10	977.60	487.54	22.08
ft20	1177.45	87.94	9.37

Cuadro 5.9: Resultados a los problemas de la clase **FT**

	$LA01$		$LA02$		$LA03$		$LA04$		$LA05$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	1074	666	170950	655	647230	597	1139	598	28	593
1	1568	666	10571	660	967085	597	5205	598	68	593
2	2259	666	29547	655	973603	597	7942	598	502	593
3	783	666	181171	660	3863	621	5259	595	23	593
4	3046	666	143001	655	253537	597	7130	590	33	593
5	2316	666	51386	655	749253	603	1314	598	85	593
6	369	666	191285	655	61082	621	5073	598	75	593
7	8012	666	3676	655	1940305	597	4791	590	62	593
8	5254	666	283144	655	324026	597	1420	595	0	593
9	386	666	3499	667	37830	620	3688	590	110	593
10	431	666	957427	662	354349	597	9830	593	25	593
11	265	666	644389	655	1444	615	2302	598	186	593
12	725	666	35025	660	1722961	597	3268	596	17	593
13	3603	666	106497	655	545791	597	6334	595	227	593
14	6627	666	18290	655	163034	597	8572	590	32	593
15	3162	666	72570	655	64218	597	8369	611	79	593
16	820	666	270247	669	1490361	597	3607	602	14	593
17	4463	666	83482	655	66628	597	7812	597	71	593
18	154	666	2053	655	479845	597	2188	598	37	593
19	1577	666	163052	662	71060	603	1276	595	6	593

Problema	$\overline{x}$	S^2	S
LA01	666.00	0.00	0.00
LA02	657.75	18.09	4.25
LA03	602.05	78.35	8.85
LA04	596.25	22.29	4.72
LA05	593.00	0.00	0.00

Cuadro 5.10: Resultados a los problemas de la clase **LA**

	$LA06$		$LA07$		$LA08$		$LA09$		$LA10$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	26	926	2588	890	809	870	366	951	65	958
1	160	926	641	890	2622	863	103	951	146	958
2	147	926	2106	890	936	863	555	951	706	958
3	79	926	803	890	4582	863	443	951	222	958
4	0	926	918	890	1250	863	975	951	29	958
5	52	926	1226	890	2607	863	615	951	6	958
6	76	926	2604	890	492	863	242	951	56	958
7	176	926	3220	890	456	863	2232	951	270	958
8	47	926	470	890	1986	863	457	951	176	958
9	430	926	932	890	891	863	506	951	144	958
10	216	926	864	890	370	863	387	951	28	958
11	142	926	1012	890	1826	863	191	951	10	958
12	133	926	2251	890	2916	863	282	951	18	958
13	0	926	926	890	1192	863	221	951	51	958
14	201	926	1691	890	982	863	831	951	36	958
15	13	926	691	890	543	863	697	951	122	958
16	15	926	715	890	360	863	277	951	111	958
17	367	926	1315	890	753	863	68	951	8	958
18	85	926	5142	890	416	863	226	951	175	958
19	319	926	3164	890	3368	863	1274	951	245	958

Problema	$\overline{x}$	S^2	S
LA06	926.00	0.00	0.00
LA07	890.00	0.00	0.00
LA08	863.35	2.33	1.53
LA09	951.00	0.00	0.00
LA10	958.00	0.00	0.00

Cuadro 5.11: Resultados a los problemas de la clase **LA** (continuación)

	$LA11$		$LA12$		$LA13$		$LA14$		$LA15$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	568	1222	245	1039	431	1150	4	1292	4984	1207
1	0	1222	894	1039	337	1150	10	1292	5824	1207
2	1	1222	398	1039	770	1150	13	1292	9663	1207
3	134	1222	367	1039	2328	1150	50	1292	5132	1207
4	74	1222	266	1039	542	1150	50	1292	4381	1207
5	100	1222	152	1039	242	1150	0	1292	26260	1207
6	396	1222	2122	1039	206	1150	10	1292	4732	1207
7	80	1222	493	1039	823	1150	110	1292	9205	1207
8	201	1222	293	1039	3904	1150	0	1292	940	1207
9	17	1222	356	1039	639	1150	157	1292	4324	1207
10	85	1222	979	1039	395	1150	31	1292	1211	1207
11	143	1222	741	1039	784	1150	0	1292	8387	1207
12	339	1222	91	1039	467	1150	5	1292	9771	1207
13	36	1222	921	1039	347	1150	40	1292	27983	1207
14	130	1222	152	1039	1009	1150	0	1292	7422	1207
15	88	1222	249	1039	702	1150	4	1292	6378	1207
16	318	1222	163	1039	658	1150	0	1292	10706	1207
17	129	1222	865	1039	1776	1150	47	1292	1890	1207
18	136	1222	680	1039	1022	1150	3	1292	3431	1207
19	158	1222	94	1039	204	1150	8	1292	4137	1207

Problema	$\overline{x}$	S^2	S
LA11	1222.00	0.00	0.00
LA12	1039.00	0.00	0.00
LA13	1150.00	0.00	0.00
LA14	1292.00	0.00	0.00
LA15	1207.00	0.00	0.00

Cuadro 5.12: Resultados a los problemas de la clase **LA** (continuación)

	$LA16$		$LA17$		$LA18$		$LA19$		$LA20$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	2598	960	16348	793	2646	889	2042	854	7830	955
1	25296	970	3260	797	68417	852	24775	846	4264	911
2	52346	946	32086	787	26318	861	45794	848	8987	934
3	30612	946	77736	787	21579	861	51345	851	56022	912
4	3273	982	83918	787	11297	857	43254	868	2971	911
5	85161	959	96382	790	3512	907	28615	860	61358	912
6	4004	979	24575	785	36988	901	51640	854	98482	963
7	29971	946	30468	787	9657	861	15100	883	20197	911
8	71117	947	815	797	4870	855	34557	863	12298	911
9	6362	946	31030	787	7339	861	1327	859	25514	914
10	60453	960	98380	787	39518	852	78387	848	14136	912
11	968	982	53705	784	3085	861	57243	848	47499	912
12	3995	982	18436	787	35906	861	31635	868	91903	926
13	23820	979	20158	793	64601	882	8196	868	6183	930
14	10789	980	46016	784	5047	861	5897	882	9450	912
15	1750	982	2747	797	17507	861	8656	860	4895	911
16	41352	982	21762	784	3377	861	45913	887	4589	911
17	21159	982	20210	797	2836	861	20730	865	38605	912
18	16801	946	57558	787	4425	861	56320	864	65251	914
19	37181	982	89731	790	3794	857	59193	842	2988	911

Problema	$\overline{x}$	S^2	S
LA16	966.90	240.39	15.50
LA17	789.35	20.53	4.53
LA18	866.15	232.03	15.23
LA19	860.90	153.69	12.40
LA20	919.25	220.89	14.86

Cuadro 5.13: Resultados a los problemas de la clase **LA** (continuación)

	$LA21$		$LA22$		$LA23$		$LA24$		$LA25$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	51612	1115	823438	927	18556	1032	656987	951	772858	1005
1	278507	1111	402228	982	33269	1032	720803	973	610335	997
2	170514	1104	194449	945	53681	1032	998365	953	199688	1008
3	589912	1060	386609	949	76398	1032	979188	954	694755	993
4	381780	1082	157082	941	24109	1032	60218	956	433434	996
5	95666	1093	727821	949	80119	1032	123507	971	996205	992
6	520234	1084	74469	947	14480	1032	321168	950	533357	998
7	125940	1082	679093	927	56180	1032	991864	963	265951	984
8	402174	1050	31848	952	96555	1032	443418	958	735385	1000
9	160589	1117	716511	935	50280	1032	87192	956	222224	1003
10	234359	1059	775521	935	13628	1032	279490	956	103336	1000
11	60375	1068	212029	943	34257	1032	221042	979	278329	993
12	998795	1082	444831	974	38762	1032	155306	971	306162	984
13	727858	1089	284618	934	23710	1032	576736	999	138354	993
14	385927	1062	853008	943	15700	1032	237986	960	33674	1006
15	78318	1082	314922	945	120087	1032	912475	969	107700	990
16	799673	1066	241308	936	8062	1032	214035	956	385099	1012
17	133095	1053	867610	968	45018	1032	446654	1003	475576	990
18	135602	1104	215876	967	3831	1032	69867	973	55670	1002
19	237171	1072	775664	943	39018	1032	978328	948	800478	1000

Problema	$\overline{x}$	S^2	S
LA21	1081.75	406.29	20.16
LA22	947.10	214.89	14.66
LA23	1032.00	0.00	0.00
LA24	964.95	219.45	14.81
LA25	997.30	54.41	7.38

Cuadro 5.14: Resultados a los problemas de la clase **LA** (continuación)

	$LA26$		$LA27$		$LA28$		$LA29$		$LA30$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	189865	1223	185706	1273	746240	1234	576593	1235	61202	1355
1	710828	1238	829558	1279	305356	1218	897946	1201	57026	1355
2	737816	1239	517852	1266	180279	1225	257638	1215	65976	1355
3	869484	1218	908579	1304	396922	1218	239868	1242	79689	1355
4	86972	1232	239140	1265	761673	1225	428762	1224	478437	1355
5	153576	1218	579817	1270	941562	1216	413960	1234	119037	1355
6	623058	1218	393253	1284	353174	1217	205537	1260	85480	1355
7	734990	1239	581490	1273	988291	1236	890710	1210	33488	1355
8	687293	1218	520784	1264	501478	1225	686799	1187	209674	1355
9	572900	1231	119999	1265	875522	1232	709428	1220	211100	1355
10	330899	1222	430325	1269	466770	1223	542888	1241	72330	1355
11	497892	1223	652364	1256	367809	1234	501334	1238	148091	1355
12	153709	1218	133545	1271	223476	1223	984251	1245	94482	1355
13	564232	1218	761344	1269	409131	1263	799167	1231	65704	1355
14	220780	1223	836041	1257	453022	1235	415883	1216	14985	1355
15	295745	1218	482181	1260	233300	1237	864736	1229	243093	1355
16	987391	1220	95349	1269	250178	1223	517290	1201	86439	1355
17	187518	1219	254153	1269	193232	1241	751190	1222	33473	1355
18	579508	1218	265564	1257	364097	1244	688405	1220	40350	1355
19	375884	1223	738345	1272	682501	1223	634185	1198	78595	1355

Problema	$\overline{x}$	S^2	S
LA26	1223.80	54.76	7.40
LA27	1269.60	109.64	10.47
LA28	1229.60	123.64	11.12
LA29	1223.45	315.75	17.77
LA30	1355.00	0.00	0.00

Cuadro 5.15: Resultados a los problemas de la clase **LA** (continuación)

	$LA31$		$LA32$		$LA33$		$LA34$		$LA35$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	9994	1786	9701	1850	4312	1719	23798	1735	9191	1888
1	8887	1809	8838	1850	2929	1725	21919	1736	9749	1909
2	7891	1784	7877	1879	4633	1719	23414	1727	9116	1901
3	8786	1789	9634	1884	3459	1724	22622	1742	9612	1913
4	4814	1784	8570	1859	9344	1749	20619	1739	8172	1898
5	3926	1784	8678	1858	8850	1725	19436	1733	9619	1923
6	5537	1784	9362	1882	6766	1722	24273	1755	8645	1895
7	6126	1798	9933	1906	5393	1721	23678	1728	6660	1898
8	5706	1784	8732	1866	6414	1719	22909	1781	9866	1916
9	9881	1820	8348	1869	5788	1722	24015	1729	8913	1897
10	5089	1784	7291	1850	6475	1721	23103	1727	6454	1901
11	9413	1799	8504	1850	2865	1725	23485	1721	9181	1900
12	5743	1784	6756	1851	8844	1721	14291	1728	7281	1938
13	7179	1784	7372	1866	5836	1722	10072	1721	7951	1904
14	4563	1792	9242	1871	1772	1719	19796	1744	2684	1888
15	7701	1784	9761	1866	5537	1725	23612	1747	9199	1912
16	3202	1822	5721	1871	8260	1719	20179	1721	6952	1898
17	7794	1784	9393	1884	7991	1719	24746	1738	2141	1898
18	5688	1784	7967	1870	5938	1719	23685	1732	9384	1915
19	9669	1790	6334	1864	9368	1727	22404	1728	6200	1898

Problema	$\overline{x}$	S^2	S
LA31	1791.45	140.25	11.84
LA32	1867.30	204.01	14.28
LA33	1723.10	41.79	6.46
LA34	1735.60	186.04	13.64
LA35	1904.50	140.15	11.84

Cuadro 5.16: Resultados a los problemas de la clase **LA** (continuación)

	$LA36$		$LA37$		$LA38$		$LA39$		$LA40$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	186993	1301	596503	1447	690376	1274	425119	1257	774117	1257
1	28808	1316	770692	1445	725508	1262	873113	1256	984727	1246
2	290006	1301	724700	1426	930314	1260	240252	1263	624448	1268
3	809444	1308	630907	1459	85585	1231	251736	1270	323194	1249
4	24920	1304	994447	1446	747879	1276	856132	1258	967550	1240
5	65813	1314	159960	1467	55835	1263	90104	1331	765560	1253
6	28467	1315	102429	1425	779554	1266	465503	1281	74741	1259
7	344620	1301	46049	1446	889350	1276	56277	1259	397817	1266
8	313627	1298	61051	1483	584691	1283	282406	1250	872967	1246
9	107057	1301	537196	1475	338553	1235	990949	1285	415360	1245
10	594141	1292	998209	1483	526145	1256	38860	1305	164611	1275
11	212827	1340	922886	1453	875370	1248	366092	1275	592979	1253
12	342008	1302	474295	1455	59468	1265	650665	1305	677438	1245
13	198171	1340	527965	1429	262202	1243	114481	1280	116960	1269
14	348306	1292	393978	1430	974402	1279	767567	1255	465345	1234
15	27755	1303	293267	1440	494068	1243	949857	1273	140178	1251
16	147135	1291	854036	1481	375248	1273	28233	1314	296488	1248
17	568678	1303	411125	1453	25047	1264	159771	1263	103693	1248
18	645320	1291	155860	1442	71820	1291	933323	1252	544176	1261
19	170627	1316	297673	1447	631602	1262	670633	1251	197222	1245

Problema	$\overline{x}$	S^2	S
LA36	1306.45	185.05	13.60
LA37	1451.60	320.34	17.90
LA38	1262.50	245.05	15.65
LA39	1274.15	514.03	22.67
LA40	1252.90	106.99	10.34

Cuadro 5.17: Resultados a los problemas de la clase **LA** (continuación)

	$ORB01$		$ORB02$		$ORB03$		$ORB04$		$ORB05$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	62939	1077	999722	917	165133	1023	275550	1032	55325	891
1	816572	1113	142575	929	93500	1032	634694	1055	110812	919
2	146067	1122	342816	924	641206	1042	364479	1034	4186	923
3	381082	1095	7740	889	185236	1038	42448	1011	204217	936
4	112757	1142	364502	890	611512	1058	3622	1069	261089	918
5	639417	1059	160992	889	938641	1038	432927	1032	159850	901
6	724628	1095	33726	896	962324	1085	7418	1052	27423	899
7	14147	1060	30459	889	9108	1062	788160	1047	649258	925
8	8040	1114	152969	917	254762	1061	339002	1038	936569	894
9	101096	1070	17454	897	292295	1043	1112	1044	152195	921
10	62939	1077	199722	917	655133	1023	275550	1032	55325	891
11	816572	1113	142575	929	93500	1032	634694	1055	110812	919
12	146067	1122	342816	924	641206	1042	364479	1034	4186	923
13	381082	1095	7740	889	856236	1038	42448	1011	204217	936
14	112757	1142	364502	890	611512	1058	3622	1069	261089	918
15	639417	1059	160992	889	938641	1038	432927	1032	159850	901
16	724628	1095	33726	896	962324	1085	7418	1052	27423	899
17	14147	1060	30459	889	9108	1062	788160	1047	649258	925
18	8040	1114	552969	917	254762	1061	339002	1038	936569	894
19	101096	1070	17454	897	292295	1043	1112	1044	512195	921

Problema	$\overline{x}$	S^2	S
orb01	1094.70	709.21	26.63
orb02	903.70	234.61	15.32
orb03	1048.20	297.56	17.25
orb04	1041.40	226.44	15.05
orb05	912.70	208.21	14.43

Cuadro 5.18: Resultados a los problemas de la clase **ORB**

	$ORB06$		$ORB07$		$ORB08$		$ORB09$		$ORB10$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	49292	1026	290363	413	542059	944	2747	970	1019237	947
1	1816247	1019	32735	404	465104	908	236435	943	48821	983
2	249573	1031	240949	397	2371	944	1819817	962	10078	995
3	44531	1099	793582	407	131881	912	135875	954	39192	982
4	744649	1084	28765	403	565176	955	85906	943	854778	952
5	390082	1054	77194	407	28297	936	115327	952	108338	967
6	58255	1105	87314	408	113247	940	1511082	952	593163	972
7	252236	1026	1741084	408	992855	908	35197	952	15083	1001
8	13762	1031	186166	403	1215626	899	20245	952	105704	1030
9	1857694	1030	12780	403	320751	933	16679	957	19917	972
10	49292	1026	290363	413	542059	944	2747	970	1019237	947
11	1816247	1019	32735	404	465104	908	236435	943	48821	983
12	249573	1031	240949	397	2371	944	1819817	962	10078	995
13	44531	1099	793582	407	131881	912	135875	954	39192	982
14	744649	1084	28765	403	565176	955	85906	943	854778	952
15	390082	1054	77194	407	28297	936	115327	952	108338	967
16	58255	1105	87314	408	113247	940	1511082	952	593163	972
17	252236	1026	1741084	408	992855	908	35197	952	15083	1001
18	13762	1031	186166	403	1215626	899	20245	952	105704	1030
19	1857694	1030	12780	403	320751	933	16679	957	19917	972

Problema	$\overline{x}$	S^2	S
orb06	1050.50	983.05	31.35
orb07	405.30	16.61	4.08
orb08	927.90	337.09	18.36
orb09	953.70	58.61	7.66
orb10	980.10	534.89	23.13

Cuadro 5.19: Resultados a los problemas de la clase **ORB** (continuación)

	$SWV01$		$SWV02$		$SWV03$		$SWV04$		$SWV05$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	331843	1508	591774	1550	255895	1515	404143	1534	131500	1575
1	946065	1472	324418	1547	137028	1463	250436	1604	202864	1593
2	225369	1475	362747	1586	389302	1503	486251	1541	158322	1545
3	439888	1480	196781	1525	884256	1508	438820	1547	181111	1553
4	342176	1486	287826	1536	408875	1472	67245	1606	244348	1535
5	181601	1470	642371	1561	577826	1510	833884	1519	234156	1501
6	851195	1497	324239	1519	720440	1492	475549	1573	141136	1548
7	489918	1499	545046	1518	234988	1549	392194	1535	135993	1547
8	53705	1538	793529	1557	304934	1524	786386	1527	199781	1564
9	120091	1479	723629	1532	961871	1490	952978	1589	249046	1589
10	393545	1485	526701	1560	398351	1485	888604	1572	205237	1566
11	234811	1496	422538	1562	33916	1488	286687	1549	220409	1596
12	160794	1528	883265	1586	906001	1509	482379	1573	242212	1546
13	271472	1458	868953	1515	450212	1510	159084	1578	216951	1586
14	075037	1503	216420	1554	424175	1556	160974	1593	212200	1565
15	454137	1474	016445	1553	617932	1513	920264	1543	230702	1587
16	406823	1491	216299	1562	324988	1515	138498	1635	182259	1586
17	936940	1459	467908	1540	564806	1481	497326	1606	116187	1547
18	318291	1499	280888	1550	664128	1541	508310	1563	175198	1585
19	292865	1479	643285	1543	296035	1509	380867	1594	232846	1541

Problema	$\overline{x}$	S^2	S
swv01	1488.80	401.66	20.04
swv02	1547.80	379.56	19.48
swv03	1506.65	550.53	23.46
swv04	1569.05	953.35	30.88
swv05	1562.75	575.09	23.98

Cuadro 5.20: Resultados a los problemas de la clase **SWV**

	$SWV06$		$SWV07$		$SWV08$		$SWV09$		$SWV10$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	238348	1796	241754	1785	181691	1908	243803	1883	114072	1924
1	229147	1819	201244	1805	181059	1958	186654	1856	206455	1942
2	234750	1847	202539	1765	228775	1905	240715	1832	238282	1952
3	210952	1852	227347	1798	235808	1957	199432	1874	236196	1928
4	225509	1825	242928	1769	247130	1927	165284	1827	246328	1892
5	212647	1837	196682	1746	166158	1992	150659	1906	146268	1924
6	169482	1899	149484	1756	219578	1957	155943	1857	222534	1906
7	236962	1772	165213	1714	235053	1984	220008	1907	233804	1936
8	242587	1852	242868	1738	219021	1926	184321	1823	238887	1933
9	231890	1891	155558	1759	217340	1959	205400	1885	99278	1893
10	233503	1853	249161	1742	104159	1913	178050	1776	201576	1872
11	131000	1841	238239	1774	233307	1922	143484	1819	170847	1951
12	189588	1879	171042	1702	207690	1925	177224	1831	131389	1933
13	247822	1809	155528	1758	189965	1926	243646	1920	242034	1935
14	130953	1853	207600	1743	245944	1920	238147	1849	173466	1944
15	189903	1839	242166	1767	58298	1981	154057	1850	246503	1886
16	175414	1896	167873	1695	206079	1963	225899	1882	231212	1890
17	232367	1876	141414	1704	242039	1955	83117	1817	196134	1932
18	241823	1828	245093	1734	213411	1932	249340	1896	241042	1952
19	245679	1847	233751	1722	246867	1931	185754	1840	233221	1900

Problema	$\overline{x}$	S^2	S
swv06	1845.55	1024.25	32.00
swv07	1748.80	910.56	30.18
swv08	1942.05	644.35	25.38
swv09	1856.50	1306.25	36.14
swv10	1921.25	578.29	24.05

Cuadro 5.21: Resultados a los problemas de la clase **SWV** (continuación)

	$SWV11$		$SWV12$		$SWV13$		$SWV14$		$SWV15$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	185775	3384	248859	3409	248973	3363	199985	3214	243225	3254
1	248613	3312	248802	3331	249022	3478	235377	3185	245994	3229
2	231976	3369	248082	3398	239442	3423	241924	3349	246818	3277
3	248590	3288	245709	3336	241241	3386	241010	3187	177824	3287
4	233603	3319	220318	3348	245540	3367	190627	3206	225778	3266
5	249075	3365	240238	3352	163765	3424	248245	3248	190377	3280
6	241866	3327	215758	3378	190610	3395	246103	3195	246961	3230
7	234441	3358	247430	3389	224182	3460	241250	3212	248515	3236
8	243636	3267	233270	3376	195496	3449	194679	3193	229487	3255
9	244674	3296	247336	3338	230965	3316	236509	3267	242608	3222
10	235560	3300	236422	3334	245553	3391	244767	3190	198449	3269
11	233112	3306	248097	3356	245836	3377	229948	3262	248670	3236
12	240896	3345	246197	3369	236887	3460	249031	3190	248615	3265
13	191156	3340	191567	3377	249663	3435	242314	3214	248485	3241
14	243780	3251	241979	3333	238353	3393	210336	3228	248593	3281
15	216442	3305	216900	3366	234383	3383	232995	3218	216417	3212
16	247367	3339	237011	3347	181780	3436	248105	3179	227041	3258
17	249886	3274	248301	3316	240232	3419	215847	3216	249872	3254
18	242951	3358	235795	3396	227884	3371	244064	3150	249118	3213
19	232290	3239	242909	3336	247769	3401	248007	3213	242906	3267

Problema	$\overline{x}$	S^2	S
swv11	3317.10	1570.49	39.63
swv12	3359.25	658.39	25.66
swv13	3406.35	1519.53	38.98
swv14	3215.80	1677.96	40.96
swv15	3251.60	503.54	22.44

Cuadro 5.22: Resultados a los problemas de la clase **SWV** (continuación)

	$SWV16$		$SWV17$		$SWV18$		$SWV19$		$SWV20$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	214	2924	123	2794	2368	2852	3882	2872	3336	2823
1	222	2924	1594	2795	245	2852	4952	2922	2407	2823
2	941	2924	1211	2794	1147	2855	4742	2843	4040	2823
3	0	2924	611	2794	812	2863	4525	2864	3720	2823
4	806	2924	4880	2818	941	2852	4916	2922	1021	2823
5	7	2924	2476	2794	3935	2854	4520	2885	4572	2823
6	805	2924	2517	2794	168	2852	4861	2866	3232	2824
7	997	2925	4971	2806	4799	2852	4307	2901	2734	2823
8	1167	2924	3489	2794	4225	2852	2445	2889	1100	2823
9	54	2924	4065	2807	4360	2852	3386	2884	1168	2823
10	550	2924	1093	2794	101	2852	4639	2879	1190	2823
11	600	2924	3079	2794	262	2852	2996	2944	1723	2823
12	2228	2924	1799	2794	403	2852	3888	2843	4472	2823
13	559	2924	1253	2794	643	2852	4734	2892	1480	2823
14	1169	2924	4591	2809	417	2852	4551	2855	2469	2823
15	101	2924	3590	2800	289	2852	4998	2872	712	2823
16	299	2924	598	2794	1762	2852	4950	2892	1999	2823
17	107	2924	1077	2794	885	2852	3584	2917	2026	2825
18	7	2924	1045	2794	384	2852	4108	2917	4699	2823
19	62	2924	908	2794	708	2852	4502	2879	959	2823

Problema	$\overline{x}$	S^2	S
swv16	2924.05	0.05	0.22
swv17	2797.55	44.95	6.70
swv18	2852.80	6.06	2.46
swv19	2886.90	710.29	26.65
swv20	2823.15	0.23	0.48

Cuadro 5.23: Resultados a los problemas de la clase **SWV** (continuación)

	$YN1$		$YN2$		$YN3$		$YN4$	
#	E	C_{max}	E	C_{max}	E	C_{max}	E	C_{max}
0	831741	924	725665	936	993809	943	871959	1003
1	480789	931	503149	937	748671	922	890990	1004
2	933086	917	320986	922	718085	933	165004	1025
3	822877	906	716165	965	601929	961	784793	1012
4	467225	913	888610	943	571313	926	298326	1010
5	592363	941	722949	944	890474	920	594132	1003
6	610871	924	198271	945	852155	950	882586	987
7	611572	916	206188	961	842685	943	608714	1009
8	733438	908	964175	950	839613	925	302453	994
9	300123	905	898795	944	942198	918	912026	1009
10	592541	908	504323	944	744831	916	971429	1024
11	447150	927	613203	935	780092	936	263012	1007
12	697312	936	472151	939	559367	928	930286	1027
13	879145	909	871609	939	851777	918	750723	1003
14	950848	923	833238	948	760688	935	857559	1015
15	862846	917	952134	956	991025	953	452148	999
16	525086	931	959103	938	306683	935	984464	1018
17	949441	923	563947	961	300164	955	751522	1025
18	635452	921	887990	941	566986	932	989408	1028
19	919290	917	575542	960	993815	931	684691	1016

Problema	$\overline{x}$	S^2	S
yn1	919.85	98.03	9.90
yn2	945.40	110.34	10.50
yn3	934.00	166.30	12.90
yn4	1010.90	123.59	11.12

Cuadro 5.24: Resultados a los problemas de la clase **YN**

5.3.6. Parámetros

La forma en cómo se seleccionaron los parámetros fue por medio de ensayo y error, ya que no tenemos una forma exacta para poder determinar el número de evaluaciones que se requiere hacer a la función objetivo para obtener resultados razonablemente buenos. Esto hizo que se comenzara con un valor de 100000 y posteriormente se aumentara o disminuyera el valor de éste de acuerdo con los resultados obtenidos y comparando éstos con la mejor solución conocida.
El valor para el pm(porcentaje de mutación), se estableció a $\frac{1}{m \times n}$, ya que con esto se garantiza que únicamente haya un movimiento en la estructura del anticuerpo y que tenga un efecto muy disruptivo en la siguiente solución. El valor del parámetro δ también fue determinado a base de ensayo y error, fijándose en 0.5.

5.4. Contendientes

En esta sección se describen brevemente los algoritmos contra los cuales se hace una comparación más extensa. La comparación se hace en términos de: número de evaluaciones y del mejor resultado obtenidos. La comparación con tiempo no se pudo realizar, ya que los tiempos que se reportan en cada trabajo están ligados con la arquitectura y plataforma de la computadora donde se ejecutaron las pruebas.

Los contendientes contra los cuales se compararó nuestro Sistema Inmune Artificial (SIA) son los siguientes:

GA Algoritmo Genético [33]. Es un algoritmo genético simple que utiliza representación de llaves aleatorias(*random keys*) [63].

HGA Algoritmo Genético Híbrido [34]. Utiliza representación de llaves aleatorias, además de una búsqueda local para mejorar las soluciones de cada generación.

GRASP Procedimiento de Búsqueda Miope Aleatoria y Adaptativa [12]. Utiliza una representación de grafos y agrega algunos mecanismos que se describen en [12].

GRASP + PR Procedimiento paralelo de Búsqueda Miope Aleatoria y Adaptativa con reenlazado de rutas [11]. Utiliza representación de permutaciones y adopta varios esquemas de paralelización.

TS Búsqueda Tabú [64]. Desafortunadamente no tuvimos acceso a esta referencia, y solamente presentamos los resultados que presentan otros autores que comparan contra este mismo algoritmo [34].

Lo que se menciona acerca de los contendientes es básico. Si se quiere tener una referencia completa deben consultarse [33, 34, 12, 11, 64]. Los resultados aquí mostrados fueron tomados de sus respectivas publicaciones.

La comparación de resultados contra los algoritmos mencionados anteriormente se realizó con 43 problemas que se tomaron de [8] y que además han sido previamente descritos. Los resultados de la comparación se muestran en los cuadros 5.25 y 5.26. Más adelante se hace una comparación en cuanto a la calidad de resultados.

En los cuadros 5.25 y 5.26 se presenta la siguiente información:

Problema: Se refiere al nombre del problema tomado de [8]

BKS: La mejor solución conocida; por sus siglas en inglés Best Know Solution.

HGA: Valor del *makespan* correspondiente al **HGA** [34].

SIA: Valor del *makespan* correspondiente a nuestro algoritmo.

 # eval: Número de evaluaciones efectuadas a la función objetivo por SIA.

GRASP: Valor del *makespan* correspondiente a **GRASP** [12].

GRASP + RT: Valor del *makespan* correspondiente a **GRASP+RT** [11].

TS: Valor del *makespan* correspondiente a **TS** [64].

GA: Valor del *makespan* correspondiente a **GA** [33].

5.4.1. Entorno Computacional

Los experimentos y ejecuciones del programa fueron ejecutados en una PC equipada con procesador Intel Pentium 4 a 2.66GHz con 512 MB de RAM. El código fue escrito en C++ y se compiló para la plataforma Linux con el compilador GNU C++ sobre Red Hat 9.0 y para Windows con Visual C++ .NET 2003.
No es necesario mencionar los demás medios físicos con los que se cuenta dado que el código es muy compacto y se optimizó para dicho procesador, lo cual hace que todo el programa pueda residir en memoria y en el peor caso para la ejecución con el problema el uso de memoria no es mayor a 1MB de memoria.

Problema	BKS	HGA	SIA	# eval.	GRASP	GRASP+	TS	GA
ft06	55	55	55	100	55	55	55	55
ft10	930	930	936	250000	938	930	930	936
ft20	1165	1165	1165	500000	1169	1165	1165	1177
la01	666	666	666	1000	666	666	666	666
la02	655	655	655	10000	655	655	655	666
la03	597	597	597	10000	604	597	597	597
la04	590	590	590	1000	590	590	590	590
la05	593	593	593	1000	593	593	593	593
la06	926	926	926	1000	926	926	926	926
la07	890	890	890	1000	890	890	890	890
la08	863	863	863	1000	863	863	863	863
la09	951	951	951	1000	951	951	951	951
la10	958	958	958	1000	958	958	958	958
la11	1222	1222	1222	1000	1222	1222	1222	1222
la12	1039	1039	1039	1000	1039	1039	1039	1039
la13	1150	1150	1150	1000	1150	1150	1150	1150
la14	1292	1292	1292	1000	1292	1292	1292	1292
la15	1207	1207	1207	1000	1207	1207	1207	1207
la16	945	945	945	10000	946	945	945	977
la17	784	784	784	10000	784	784	784	787
la18	848	848	848	10000	848	848	848	848
la19	842	842	842	10000	842	842	842	857
la20	902	907	907	250000	907	902	902	910

Cuadro 5.25: Comparación de resultados

Problema	BKS	HGA	SIA	# eval.	GRASP	GRASP+	TS	GA
la21	1046	1046	1046	250000	1091	1057	1047	1047
la22	927	935	927	250000	960	927	927	936
la23	1032	1032	1032	250000	1032	1032	1032	1032
la24	935	953	935	250000	978	954	939	955
la25	977	986	979	250000	1028	984	977	1004
la26	1218	1218	1218	200000	1271	1218	1218	1218
la27	1235	1256	1240	500000	1320	1269	1236	1260
la28	1216	1232	1216	1000000	1293	1225	1216	1241
la29	1157	1196	1170	5000000	1293	1203	1160	1190
la30	1355	1355	1355	100000	1368	1355	1355	1356
la31	1784	1784	1784	5000	1784	1784	1784	1784
la32	1850	1850	1850	25000	1850	1850	1850	1850
la33	1719	1719	1719	25000	1719	1719	1719	1719
la34	1721	1721	1721	10000	1753	1721	1721	1730
la35	1888	1888	1888	50000	1888	1888	1888	1888
la36	1268	1279	1281	250000	1334	1287	1268	1305
la37	1397	1408	1408	250000	1457	1410	1407	1441
la38	1196	1219	1204	250000	1267	1218	1196	1248
la39	1233	1246	1249	250000	1290	1248	1233	1264
la40	1222	1241	1228	2500000	1259	1244	1229	1252

Cuadro 5.26: Comparación de resultados (continuación)

	SIA		
	G	E	P
HGA	3	32	8
GRASP	0	24	19
GRASP + PR	3	30	10
TS	9	31	3
GA	0	23	20

Cuadro 5.27: Resultados generales de la comparación

	OA	SIA	Mejora
HGA	0.42 %	0.18 %	0.23 %
GRASP	1.90 %	0.18 %	1.72 %
GRASP + PR	0.47 %	0.18 %	0.28 %
TS	0.06 %	0.18 %	-0.13 %
GA	0.93 %	0.18 %	0.74 %

Cuadro 5.28: Mejora del SIA sobre los otros algoritmos (OA)

5.5. Análisis de resultados

En la sección previa se presentó un cuadro en el cual se muestra el desempeño del SIA frente a los otros algoritmos que seleccionamos para prueba. Podemos decir que nuestro algoritmo tiene un desempeño mucho muy bueno en comparación con los demás algoritmos, principalmente en aquellos problemas de tamaño considerable, sin tener en cuenta que TS le ganó en varios de los problemas (principalmente en los de 15×15). En el cuadro 5.27, se muestra el resumen de lo que se acaba de mencionar, indicándose qué tanto mejoró la calidad de solución nuestro SIA.

Para poder leer el cuadro 5.27, se observa que las 3 columnas de la derecha corresponden al número de instancias en el que el SIA propuesto obtuvo esos resultados. Las letras significan **G**:Ganó, **E**:Empató y **P**:Perdió con respecto a los algoritmos que se muestran en la parte izquierda del cuadro.

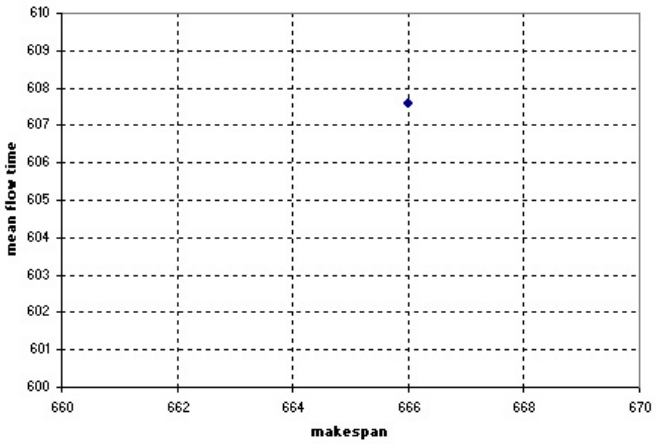
En el cuadro 5.28, en la segunda columna se muestra la desviación de los resultados obtenidos por el algoritmo de la primer columna con respecto a la mejor solución conocida. En la tercera columna se muestra la desviación que obtuvo el SIA con respecto a la mejor solución conocida. En la última columna se muestra el que tanto mejora la solución el SIA con respecto a los algoritmos de cada fila.
Se puede observar que el SIA mejora por considerablemente las soluciones de todos los algoritmos, excepto a la TS que aún es mejor que el SIA. Por ejemplo GRASP, queda muy por abajo de ser una buena opción, ya que si se revisa [12], aparte de quedar muy por debajo de los resultados obtenidos por el SIA, los autores presentan

que éste requiere de un alto costo computacional. La versión de GRASP + PR, es una versión paralela que emplea varios mecanismos para optimizar el JSSP. A pesar de eso el SIA muestra una mejor desempeño que este algoritmo. El HGA y GA son algoritmos genéticos, ambos basados en la representación de llaves aleatorias y la única diferencia que presenta HGA con respecto a GA es un mecanismo de búsqueda local que está dentro del proceso de evaluación del cromosoma, y que solo lo deja avanzar hasta que mejora su aptitud. Los autores únicamente publican el número de generaciones, pero no dicen cuántos llamados a la función objetivo hacen, esto para fines de comparación con el SIA. Despúes de haber mencionado algunas de sus características podemos observar que el SIA obtiene mejores resultados.
Podemos decir, para concluir, que el SIA es una buena opción para la optimización del JSSP mono-objetivo, ya que los resultados que presentamos son competitivos respecto a los publicados recientemente y con un bajo costo computacional.

5.6. Resultados Multiobjetivo

En esta sección nos enfocaremos a hablar de los resultados obtenidos con la versión multiobjetivo del JSSP.
En primera instancia mencionaremos que el JSSP es un problema que tiene múltiples objetivos, que aparentemente están en conflicto, pero la realidad es que cuando se optimiza el problema con varios objetivos, basta con optimizar uno de ellos, para proceder después a optimizar los otros. Esto hace que en el frente de Pareto haya pocos puntos y que en la mayoría de los casos que probamos sólo hay uno. Otros autores[5], presentan la solución a varias instancias del JSSP con múltiples objetivos reportando varios puntos en el frente de Pareto. Sin embargo, al usar dichos problemas, encontramos que se llega también en esos casos a un solo punto en el frente de Pareto. Lo cual nos deja concluir que estaban reportantdo un falso frente de Pareto. Esto es debido a que el algoritmo que se usó para optimizar el JSSP no es muy bueno. Es difícil que podamos conocer el verdadero frente de Pareto de alguna de las instancias del JSSP, debido a que el espacio de búsqueda es muy grande y sería prácticamente imposible generarlo por enumeración[3].
Se realizó un intento de generar el frente de Pareto real de una de las instancias del JSSP: ft06, que es un problema de 6×6 y que tiene un espacio de búsqueda de 1.39×10^{39}. Este proceso costó al rededor de 9 días de procesamiento intensivo en la computadora donde se realizaron las pruebas. El resultado encontrado después de haber procesado todas las soluciones, nos llevó a un solo punto.

[3]Por tanto, resulta difícil en general poder saber si el frente hallado por un algoritmo determinado es el verdadero o no.



C_{max}	$\overline{F_\Sigma}$
666	607.6

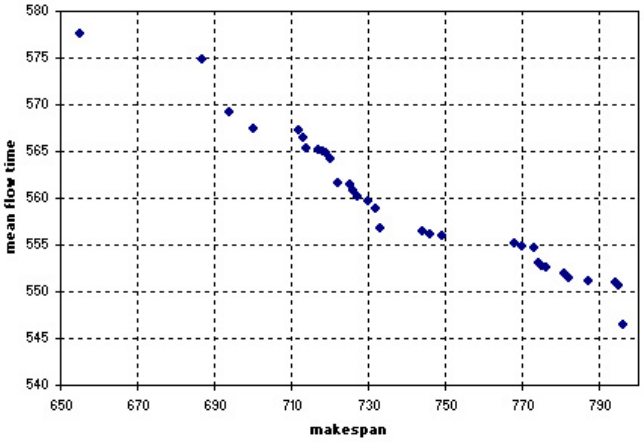
Figura 5.1: Frente de Pareto correspondiente al problema la01

5.6.1. Instancias resueltas del JSSP multiobjetivo

Se probó únicamente con 10 instancias del JSSP, esto debido a que como se pudo observar al optimizar un solo objetivo, se podía luego trabajar sobre otro, para obtener un mejor punto o algunos cuantos puntos de tal forma que las instancias multiobjetivo podieran ser realmente tratadas como mono-objetivo. Recordemos que los objetivos a minimizar para el JSSP multiobjetivo son el *makespan* y el *mean flow time*, descritos en el capítulo referente al JSSP.
Ahora mostraremos los resultados obtenidos de las instancias resueltas. Se presenta la gráfica que compone el frente de Pareto y el cuadro con los puntos que forman el conjunto de óptimos de Pareto.

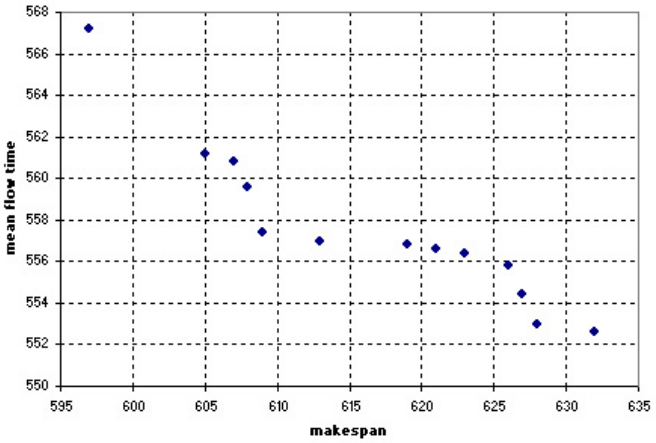
Es posible que a pesar de que el JSSP sea un problema multiobjetivo no se hayan hecho más trabajos al respecto, al parecer por lo dicho anteriormente de que con optimizar un objetivo se puede obtener un buen resultado en los demás objetivos[4].
Con esto concluimos que la versión multiobjetivo del algoritmo funcionó, aunque el planteamiento multiobjetivo del JSSP que utilizamos no fue el adecuado.
Los resultados que se presentan son de los problemas $la01 - la10$ y se muestran de la figura 5.1 a la figura 5.10. Se muestra la gráfica que contiene el frente de Pareto y el cuadro con los puntos que lo forman.
Los parámetros que se utilizaron son los mismos que los empleados en la versión mono-objetivo, aunque en este caso se utilizó siempre el número de evaluaciones mayor a $1,000,000$. En la mayoría de los casos ese número de evaluaciones bastaba para encontrar los puntos que se muestran. Se probó con un número mayor de evaluaciones pero los resultados no mejoraron. Tal es el caso de $la02$, en el que no se pudieron

[4]Esto implica que no hay conflicto entre los objetivos, lo cual restaría interés al problema desde la perspectiva de optimización multiobjetivo.



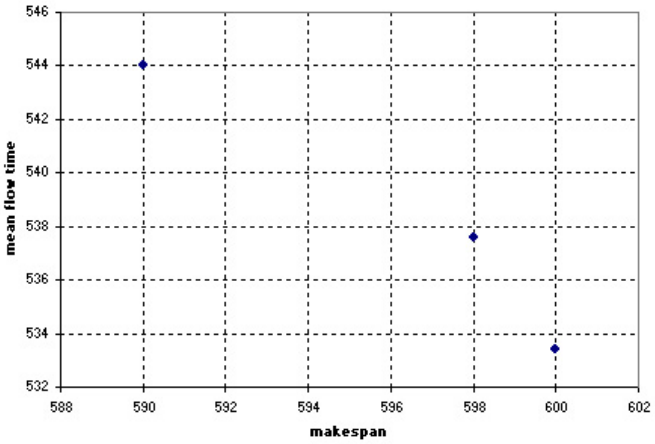
C_{max}	$\overline{F_\Sigma}$	C_{max}	$\overline{F_\Sigma}$
722	561.6	770	554.8
714	565.4	700	567.4
712	567.2	694	569.2
713	566.4	717	565.2
768	555.2	725	561.4
720	564.2	718	565
719	564.8	796	546.4
774	553	744	556.4
726	560.8	727	560.2
733	556.8	749	556
746	556.2	775	552.8
776	552.6	730	559.6
732	558.8	687	574.8
782	551.4	773	554.6
781	552	787	551.2
794	551	795	550.6
655	577.6		

Figura 5.2: Frente de Pareto correspondiente al problema la02



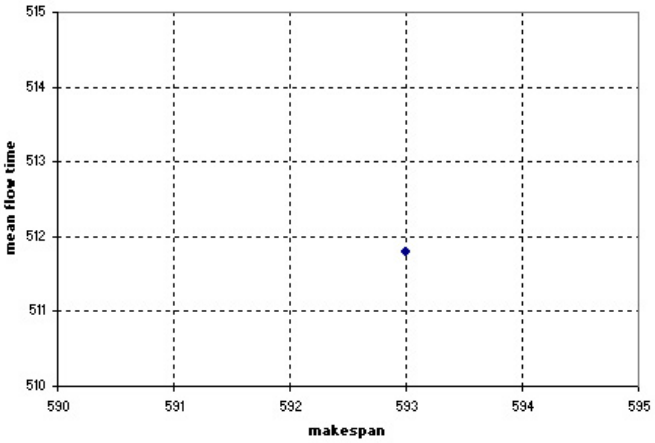
C_{max}	$\overline{F_\Sigma}$
608	559.6
609	557.4
607	560.8
605	561.2
621	556.6
613	557
619	556.8
628	553
627	554.4
626	555.8
623	556.4
632	552.6
597	567.2

Figura 5.3: Frente de Pareto correspondiente al problema la03



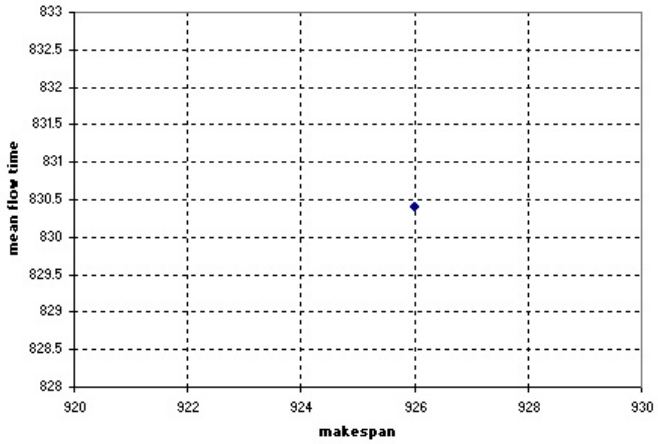
C_{max}	$\overline{F_\Sigma}$
600	533.4
598	537.6
590	544

Figura 5.4: Frente de Pareto correspondiente al problema la04



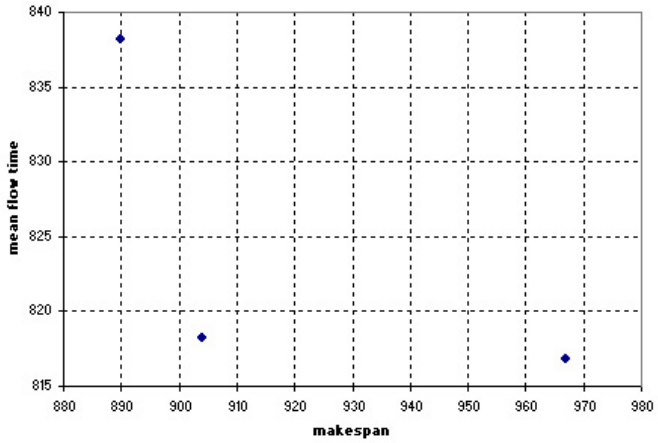
C_{max}	$\overline{F_\Sigma}$
593	511.8

Figura 5.5: Frente de Pareto correspondiente al problema la05



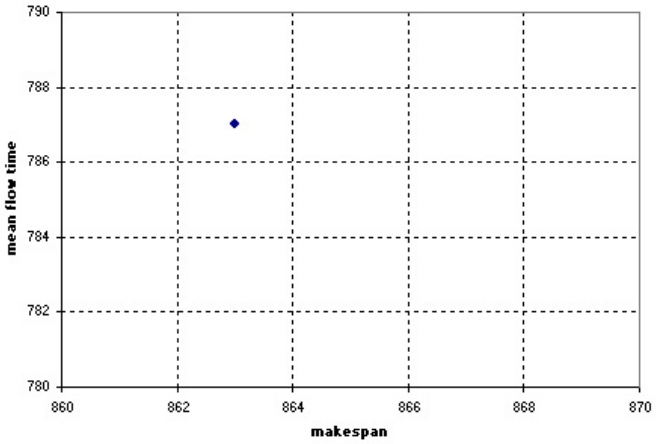
C_{max}	$\overline{F_\Sigma}$
926	830.4

Figura 5.6: Frente de Pareto correspondiente al problema la06



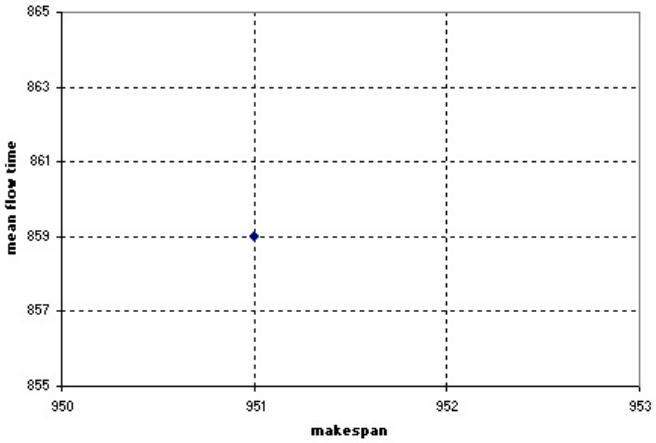
C_{max}	$\overline{F_\Sigma}$
904	818.2
967	816.8
890	838.2

Figura 5.7: Frente de Pareto correspondiente al problema la07



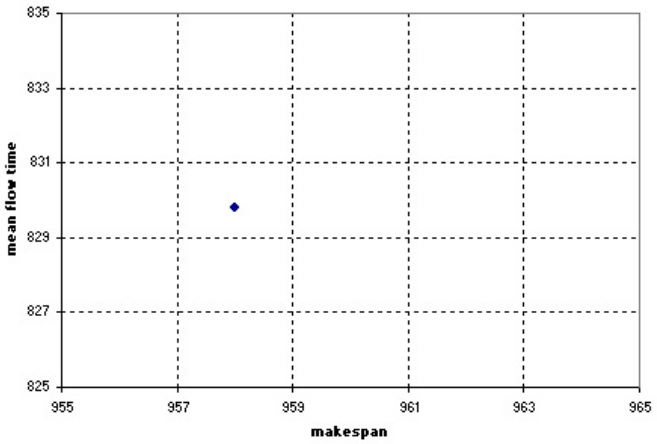
C_{max}	$\overline{F_\Sigma}$
863	787

Figura 5.8: Frente de Pareto correspondiente al problema la08



C_{max}	$\overline{F_\Sigma}$
951	859

Figura 5.9: Frente de Pareto correspondiente al problema la09



C_{max}	$\overline{F_\Sigma}$
958	829.8

Figura 5.10: Frente de Pareto correspondiente al problema la10

obtener menos puntos en el frente; es posible que esto se deba a las características del problema.

Capítulo 6

Conclusiones

El objetivo de esta tesis fue proponer un algoritmo inspirado en el sistema inmune artificial, en particular en el modelo de selección clonal, para resolver el problema del Job Shop Scheduling. El algoritmo propuesto es el resultado de muchas pruebas realizadas con diferentes mecanismos basados en el principio de selección clonal. En general el algoritmo tiene un buen desempeño en comparación con otros algoritmos que tratan de resolver el JSSP; esto lo podemos observar en el capítulo 5. Una de las ventajas del algoritmo propuesto es que es muy compacto en cuanto a recursos, ya que gracias a la representación y los mecanismos utilizados para la optimización de éste, se usa poca memoria y las operaciones requeridas para decodificar una solución también son pocas en comparación con otros algoritmos que utilizan una representación diferente para optimizar el JSSP.
El algoritmo se compara contra GRASP, GA, HGA y TS, pudiendo observar que contra los primeros 3 se tiene un mejor resultado y contra TS se tiene un serio adversario bastante difícil de vencer. Los algoritmos que comparamos tienen muchas características que hacen más costosa (computacionalmente hablando) la optimización del problema y, en algunos casos, se usa incluso paralelismo.
Se puede observar que en general para la mayoría de los problemas atacados nuestro algoritmo tiene un muy buen desempeño, que lo hace ser competitivo respecto a las opciones presentadas en esta tesis, siendo con esto una buena opción para resolver el JSSP.

Podemos mencionar algunas de las experiencias que se tuvieron en la realización de la presente tesis:

- Originalmente la tesis estaba enfocada a resolver el JSSP multiobjetivo. Sin embargo, los experimentos y los resultados obtenidos hicieron ver que no es necesario tratar el JSSP como tal, ya que con optimizar un solo objetivo es fácil optimizar los restantes. Esto originó que se tomara la decisión de hacer un mejor algoritmo pero únicamente para el caso mono-objetivo.

- Se realizaron 3 versiones del algoritmo inspirado en el principio de selección clonal. La primera versión utilizaba bibliotecas de anticuerpos para construirlos y mutación por intercambio, un operador general para permutaciones con repetición. Esta versión del programa mostró buenos resultados y fué publicada en [23]. La segunda versión del algoritmo siguió utilizando las bibliotecas de anticuerpos y se cambió el operador de mutación por uno más avanzado que corresponde a la mutación A (véase la sección 4.2.3); ésta versión se publicó en [22]. La última versión que se desarrolló y es la que se presenta en este trabajo. Se eliminaron las bibliotecas para construir los anticuerpos, se agregó un operador más de mutación que corresponde a la mutación B (véase la sección 4.2.3). Además se agregó un mecanismo de memoria, el cual permite que el algoritmo explore la vecindad del mejor individuo para generar una mejor solución. Esta última versión se publica en [24].

- Todos las versiones que se desarrollaron fueron publicadas y fuerón presentadas cada una en un evento distinto.

Este fué el recorrido que seguimos para la culminación de esta tesis.

6.1. Trabajo Futuro

Como parte del trabajo futuro posible, se podrían estudiar planteamientos multi-objetivo distintos del JSSP, buscando lograr un conflicto entre las funciones objetivo. Para la versión mono-objetivo del algoritmo sería interesante experimentar con un híbrido que incorporara mecanismos de la búsqueda tabú (p.ej. las listas tabú [64, 82]), a fin de hacer más competitivo. La paralelización del algoritmo podría ser también una ruta promisoria de investigación, junto con el uso de otros mecanismos más elaborados de búsqueda local, tales como los de los algoritmos meméticos [62].
Otra de las ideas que se podrían utilizar es por ejemplo la búsqueda entre los vecinos que utiliza [64]. Esto agregandose en el proceso de mutación, es decir, que sea una mutación con memoria para no aplicarla más de una vez de la misma forma o que se obtenga un mismo resultado.
La aplicación de algunas estrategias en la construcción de la solución inicial. Para esto se podría utilizar algún algoritmo de los existentes [92, 2, 41, 40, 3, 29].
Sin lugar a duda queda mucho trabajo a futuro que se puede realizar agregar para tener un algoritmo más robusto.

Apéndice A

Optimización Multiobjetivo

La toma de decisiones es algo que no podemos eludir, ocurre diariamente y a cada instante. En la vida cotidiana estos problemas son resueltos mediante el uso del sentido común o intuición, pero en ingeniería se deben resolver eficientemente y en un tiempo razonable. La toma de decisiones puede verse como un problema de optimización en el cual generalmente están involucrados dos o más objetivos (y en la mayoría de las ocasiones éstos están expresados en unidades diferentes y se encuentran en conflicto entre sí). Este problema es conocido como optimización multiobjetivo o multicriterio. Tiene la particular característica de que no existe una solución única, sino un conjunto de soluciones válidas [1] (cada una de ellas puede ser considerada como una solución del problema). Esta característica es llevada a cabo gracias a que ninguna de ellas domina (o es mejor) a las demás soluciones en todas las funciones objetivo (de ahí que se les llame no dominadas).

A.1. Conceptos básicos y terminología

El problema de optimización multiobjetivo (MOP[2]) es definido por Osyczka como [71]: "*El problema de encontrar un vector de variables de decisión que satisfaga las restricciones y optimice una función vectorial cuyos elementos representen las funciones objetivo. Estas funciones forman una descripción matemática de criterios de desempeño que están usualmente en conflicto entre sí. Por lo tanto, el término optimizar significa encontrar aquella solución que dará un valor aceptable al diseñador en todas las funciones objetivo.* "

La definición verbal anterior será formalizada posteriormente, pero antes es necesario introducir ciertos conceptos:

[1] A este conjungo se le conoce como conjunto de óptimos de Pareto

[2] Por sus siglas en inglés

Variables de decisión : Las variables de decisión son un conjunto de n parámetros cuyos valores dan una solución (puede o no ser válida) a un problema de optimización. Estos parámetros serán denotados como $x_j, j = 1, 2, ..., n$. Estas variables de decisión serán representadas en este trabajo de la siguiente manera:

$$\bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Dicho vector se manejará por conveniencia como:

$$\bar{f}(\bar{x}) = [f_1(\bar{x}), f_n(\bar{x}), ..., f_n(\bar{x})]^T$$

Función objetivo : Las funciones objetivo forman el criterio de evaluación para saber qué tan buena es una solución; al igual que las restricciones, son funciones de las variables de decisión. En la optimización multiobjetivo existen dos o más funciones objetivo $(f_1(\bar{x}), f_2(\bar{x}), ..., f_n(\bar{x}))$ en cada problema.

A continuación se darán tanto una definición matemática para el problema de la optimización multiobjetivo así como algunas definiciones necesarias para comprender de una mejor manera esta problemática:

El problema de optimización multiobjetivo : Un MOP incluye, un conjunto de n variables de decisión y k funciones objetivo. En estos problemas el objetivo de la optimización es encontrar el vector variables de decisión $\bar{x}$ tal que

$$minimicef(\bar{x}) = [f_1(\bar{x}), f_n(\bar{x}), ..., f_n(\bar{x})]^T$$

En palabras esto significa encontrar los parámetros necesarios que optimicen la función vectorial objetivo y que satisfagan las restricciones del problema.

Conjunto factible : El conjunto factible S está definido como el conjunto de vectores de decisión x que cumple con las restricciones del problema.

$$S = \{\bar{x} \in \bar{X}\}$$

La imagen de S en el espacio de las funciones objetivo está definido por $Z = f(S)$.

Dominancia de Pareto : Para dos vectores de decisión $\overline{x^*}, \overline{y^*} \in X$,

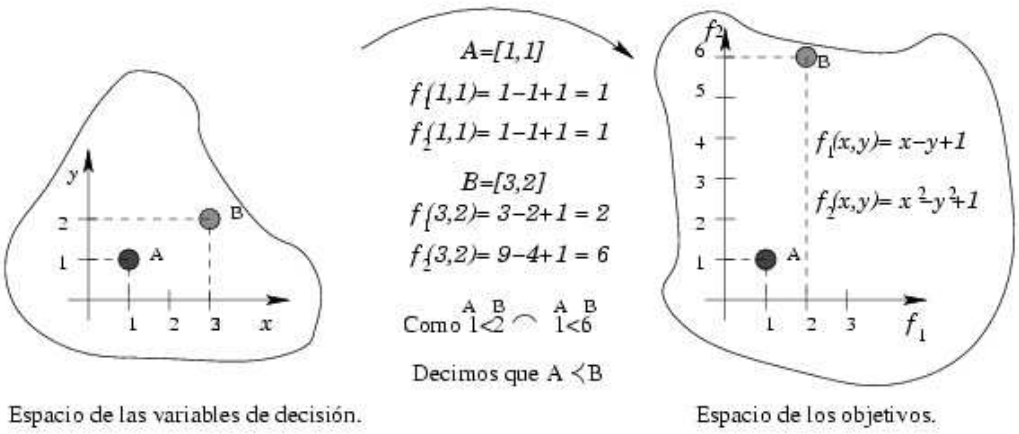


Figura A.1: Se muestra el espacio de las variables de decisión y el espacio de los objetivos.

$$\overline{x^*} \prec \overline{y^*} \quad \text{ssi } f_i(\overline{x^*}) < f_i(\overline{y^*})$$
$$\overline{x^*} \text{ domina a } \overline{y^*} \quad \text{para toda } i = 1, .., k$$

$$\overline{x^*} \preceq \overline{y^*} \quad \text{ssi } f_i(\overline{x^*}) \leq f_i(\overline{y^*})$$
$$\overline{x^*} \text{ domina débilmente a } \overline{y^*} \quad \text{para toda } i = 1, .., k$$

$$\overline{x^*} \sim \overline{y^*} \quad \text{ssi } f_i(\overline{x^*}) \nleq f_i(\overline{y^*}) \bigwedge f_i(\overline{y^*}) \nleq f_i(\overline{x^*})$$
$$\overline{x^*} \text{ es indiferente a } \overline{y^*} \quad \text{para toda } i = 1, .., k$$

Las definiciones son análogas para problemas de maximización ($\succ, \succeq, \sim$).

Es importante notar que aunque la dominancia se da en el espacio de las variables de decisión, la comparación se da en el resultado de la evaluación de las funciones objetivo (espacio de las funciones objetivo). En la figura A.1 se puede observar la diferencia entre el espacio de las variables de decisión y el espacio de las funciones objetivo.

Optimalidad de Pareto : Un vector de decisión $\overline{x^*}$ es óptimo de Pareto si y sólo si:

$$\overline{x^*} \in S | \neg \exists \overline{y^*} \in S | \bar{f}(\overline{x^*}) \preceq \bar{f}(\overline{y^*})$$

En palabras podemos decir que un vector de decisión que es miembro del conjunto factible S es óptimo de Pareto si no existe otro vector de variables de decisión $\overline{y^*}$ que pertenezca a S y que lo domine.

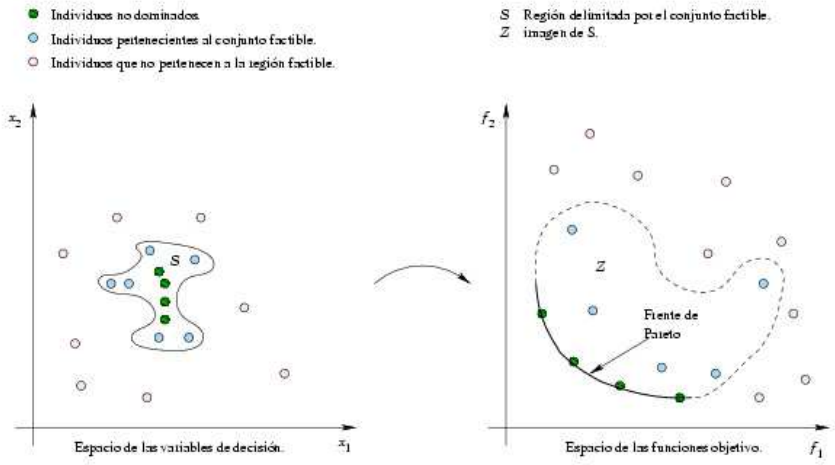


Figura A.2: Conjunto factible S y el conjunto óptimo de Pareto en el espacio de las variables de decisión(izquierda), y de sus imagenes en el espacio de las funciones objetivo (derecha).

Conjunto de Optimos de Pareto : El conjunto de óptimos de Pareto (P^*) esta definido como:

$$P^* = \{\overline{x^*} \in S | \neg \exists \overline{y^*} \in S | \bar{f}(\overline{x^*}) \preceq \bar{f}(\overline{y^*})\}$$

El conjunto de óptimos de Pareto se encuentra en el espacio de las variables de decisión, y está representado gráficamente en la figura A.2.

Cabe mencionar que algunos problemas presentan diferentes frentes de Pareto ficticios que atraen la mayoría de las soluciones, estos frentes son conocidos como falsos frentes de Pareto. En la figura A.3 se ilustra gráficamente la diferencia entre un falso frente de Pareto y uno verdadero.

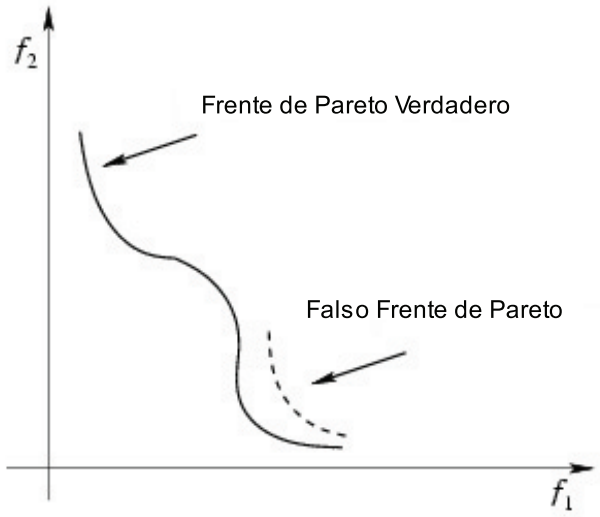


Figura A.3: Diferencias entre el frente de Pareto falso y verdadero.

Apéndice B

Diagrama de Gantt HTML en perl

Aquí se muestra el listado del código *perl* para convertir la salida del algoritmo en un documento HTML.
El programa se ejecuta de la siguiente manera: perl gantt.pl la01.666.txt > la01.666.html, donde el primer argumento es el archivo de entrada que contiene la solución que se obtiene del programa y el segundo argumento el archivo donde se tiene el documento HTML de salida.

```
die "Args: {fileName}\n" if $#ARGV != 0;

$counter = 0;
$machine = 0;
$job = 0;


#paleta de colores HTML
@color = (
"aliceblue", "antiquewhite", "aqua", "aquamarine", "azure",
"beige", "bisque", "black", "blanchedalmond", "blue",
"blueviolet",  "brown", "burlywood", "cadetblue", "chartreuse",
"chocolate", "coral", "cornflowerblue", "cornsilk", "crimson",
"cyan", "darkblue", "darkcyan", "darkgoldenrod", "darkgray",
"darkgreen", "darkkhaki", "darkmagenta", "darkolivegreen",
"darkorange", "darkorchid", "darkred", "darksalmon",
"darkseagreen", "darkslateblue", "darkslategray", "darkturquoise",
"darkviolet", "deeppink", "deepskyblue", "dimgray",
"dodgerblue", "firebrick", "floralwhite", "forestgreen",
"fuchsia", "gainsboro", "ghostwhite", "gold", "goldenrod",
"gray", "green", "greenyellow", "honeydew", "hotpink",
```

```
"indianred", "indigo", "ivory", "khaki", "lavender",
"lavenderblush", "lawngreen", "lemonchiffon", "lightblue",
"lightcoral", "lightcyan", "lightgoldenrodyellow", "lightgreen",
"lightgray", "lightpink", "lightsalmon", "lightseagreen",
"lightskyblue", "lightslategray", "lightsteelblue", "lightyellow",
"lime", "limegreen", "linen", "magenta", "maroon", "mediumaquamarine",
"mediumblue", "mediumorchid", "mediumpurple", "mediumseagreen",
"mediumslateblue", "mediumspringgreen", "mediumturquoise",
"mediumvioletred", "midnightblue", "mintcream", "mistyrose",
"moccasin", "navajowhite", "navy", "oldlace", "olive", "olivedrab",
"orange", "orangered", "orchid", "palegoldenrod", "palegreen",
"paleturquoise", "palevioletred", "papayawhip", "peachpuff", "peru",
"pink", "plum", "powderblue", "purple", "red", "rosybrown",
"royalblue", "saddlebrown", "salmon", "sandybrown", "seagreen",
"seashell", "sienna", "silver", "skyblue", "slateblue", "slategray",
"snow", "springgreen", "steelblue", "Tan", "teal", "thistle",
"tomato", "turquoise", "violet", "wheat", "white", "whitesmoke",
"yellow", "yellowgreen" );

while ( <> )
{
 if ( ! /^#/ )
 {
  if ( $counter == 0 )
  {
   @values = split;
   $n = $values[0];
   $m = $values[1];
   $counter ++;
  }
  elsif ( $counter == 1 )
  {
   @antibody = split;
   $counter ++;
  }
  elsif ( $counter == 2 )
  {
   @values = split;
   $fitness = $values[0];
   $counter ++;
  }
```

```
  elsif ( $counter == 3 )
  {
   @values = split;
   $schedule[0][ $machine ][ $job ] = $values[ 0 ];
   $schedule[1][ $machine ][ $job ] = $values[ 1 ];
   $schedule[2][ $machine ][ $job ] = $values[ 2 ];
   $job ++;
   if ( $machine == $m - 1 && $job == $n )
   {
    $counter ++;
   }

   if ( $job == $n )
   {
    $job = 0;
    $machine ++;
   }
  }
 }
}


$i = 0;
$j = 0;


$TSTYLE = ".free\n{\n\tbackground-color:white;";
$TSTYLE .= "color:black;font-size:6pt;\n}\n";
$TSTYLE .= ".axe\n{\n\twidth:15px;height:15px;";
$TSTYLE .= "font-size:8pt;border:none;\n}\n";
$TSTYLE .= ".machine\n{\n\twidth:20;height:30px;";
$TSTYLE .= "border:none;font-size:9pt;\n}\n";
$TSTYLE .= "TABLE\n{\n\ttext-align:center;border:thin groove skyblue;";
$TSTYLE .= "background-color:ghostwhite;color:blue;\n}\n";

for ( $i = 0; $i < $n; $i ++ )
{
 $TSTYLE .= "\n.j".$i."\n{\n\tborder:thin groove;background-color:".
 $color[int(rand() * scalar(@color))].";color:white;font-weight:bold;\n}\n"
}
```

```
print "<HTML>\n";
print "<HEAD>\n\t"
print "<TITLE>Job Shop Scheduling Problem - Gantt Chart</TITLE>\n
print "</HEAD>\n";
print "<STYLE>$TSTYLE</STYLE>\n";
print "</HEAD>\n";
print "<BODY>\n";
print "<TABLE border=\"1\">\n";


for ( $i = 0; $i < $m; $i++ )
{
 print "\t<TR>\n";

 for ( $j = 0; $j < $n; $j++ )
 {
  if ( $j == 0 )
  {
   print "\t\t<TD class=\"machine\">M<SUB>".$i."</SUB></TD>\n";

   if ( $schedule[1][$i][$j] > 1 )
   {
    print "\t\t<TD class=\"free\" colspan=\"",$schedule[1][$i][$j] -
          1," \"></TD>\n";
   }

   print "\t\t<TD class=\"j", $schedule[0][$i][$j], "\" colspan=\"";
   print $schedule[2][$i][$j] - $schedule[1][$i][$j] + 1, "\">";
   print "J<SUB>", $schedule[0][$i][$j], "</SUB></TD>\n";
  }
  elsif ( $j == $n - 1 )
  {
   if ( $schedule[1][$i][$j] - $schedule[2][$i][$j - 1] > 1 )
   {
    print "\t\t<TD class=\"free\" colspan=\"", $schedule[1][$i][$j] -
          $schedule[2][$i][$j - 1] - 1, "\"></TD>\n";
   }

      print "\t\t<TD class=\"j", $schedule[0][$i][$j], "\" colspan=\"";
      print $schedule[2][$i][$j] - $schedule[1][$i][$j] + 1, "\">";
```

```
                print "J<SUB>", $schedule[0][$i][$j], "</SUB></TD>\n";

   if ( $schedule[2][$i][$j] < $fitness )
   {
    print "\t\t<TD class=\"free\" colspan=\"", $fitness -
          $schedule[2][$i][$j], "\"></TD>\n";
   }
  }
  else
  {
   if ($schedule[1][$i][$j] - $schedule[2][$i][$j - 1] > 1 )
   {
    print "\t\t<TD class=\"free\" colspan=\"", $schedule[1][$i][$j] -
          $schedule[2][$i][$j - 1] - 1, "\"></TD>\n";
   }

   print "\t\t<TD class=\"j", $schedule[0][$i][$j],"\" colspan=\"";
   print $schedule[2][$i][$j] - $schedule[1][$i][$j] + 1, "\">J<SUB>",
         $schedule[0][$i][$j],"</SUB></TD>\n";
  }
 }
 print "\t</TR>\n";
}


print "\t<TR>\n\t\t<TD></TD>\n";
for ( $i = 1; $i <= $fitness; $i ++ )
{
 print "\t\t<TD class=\"axe\">$i</TD>\n";
}
print "\t</TR>\n";
print "</TABLE>\n";
print "</HTML>\n";
```

Apéndice C

Utilerías para Lekin

Lekin es un sistema para planificación de recursos desarrollado en la Stern School of Business, de la Universidad de Nueva York. Muchas de sus partes fueron codificadas por estudiantes de la Universidad de Columbia. Lekin fue creado con fines educativos, con la finalidad de introducir a los estudiantes en la teoría de planificación de recursos y sus aplicaciones. Aunado a esto, se ha usado para el desarrollo de algoritmos para resolver problemas de planificación. El proyecto es dirigido por el profesor Michael Pinedo, el profesor Xiuli Chao y el profesor Joseph Leung.

Lekin se puede descargar desde: http://www.stern.nyu.edu/om/software/lekin/. La versión de Lekin, tiene un módulo para resolver el JSSP. Esa es la parte que nosotros utilizamos para comprobar nuestros resultados y poder imprimir gráficas. Su uso fue solo como material de apoyo.

Se contruyuron 3 pequeños programas que convierten el problema en nuestro formato al formato de Lekin, esto con la finalidad de hacer pruebas.

Se muestra el código de los programas y qué es lo que hace cada uno de ellos.

El primer programa *job.pl*, se usa para convertir el archivo que contiene el problema, tal y como lo encontramos en [8], a un archivo $\{nombre\}.job$, donde se especifican los trabajos con las operaciones.

```
die "Args: (filename)\n" if $#ARGV != 0;

$counter = 1;
$machine = 0;
$job = 0;

print "Shop:                Job\n" ;
```

```
while ( <> )
{
 if ( ! /^#/ )
 {
  if ( $counter == 1 )
  {
   @values = split;
   $n = $values[0];
   $m = $values[1];
   $counter ++;
  }
  elsif ( $counter == 2 )
  {
   @values = split;

   printf ( "Job:                J%02d\n", $job + 1 );
   printf ( "  RGB:                %d;%d;%d\n",
            int(255 * rand() ),
            int(255 * rand() ),
            int(255 * rand() ) );
   printf ( "  Release:            0\n" );
   printf ( "  Due:                0\n" );
   printf ( "  Weight:             1\n" );

   for ( $i = 0; $i < $m * 2; $i+=2 )
   {
    printf "  Oper:               M%02d;%d;A\n",
           $values[ $i ] + 1, $values[ $i + 1 ] ;
   }
   printf ( "\n" );

   $job++;
  }
 }
}
```

El segundo programa *mch.pl*, se usa para convertir el archivo que contiene el problema, tal y como lo encontramos en [8], a un archivo {*nombre*}.*mch*, donde se especifican las máquinas.

```
die "Args: (filename)\n" if $#ARGV != 0;

$counter = 1;
$machine = 0;
$job = 0;

print "Ordinary:\n" ;

while ( <> )
{
 if ( ! /^#/ )
 {
  if ( $counter == 1 )
  {
   @values = split;
   $n = $values[0];
   $m = $values[1];
   $counter ++;

   for ( $i = 0; $i < $m; $i++ )
   {
    printf "Workcenter:         M%02d\n", $i + 1 ;
    printf "  RGB:                %d;%d;%d\n",
            int(255 * rand()),
            int(255 * rand()),
            int(255 * rand()) ;
    print "  Release:            0\n" ;
    print "  Status:             A\n\n" ;
   }
  }
 }
}
```

En el último programa se convierte la salida de nuestro algoritmo a un archivo que contiene la secuencia de operaciones acomodadas en las máquinas y que cumplen con las condiciones del problema especificado. El archivo debe ser $\{nombre\}.seq$.

```
die "Args: (filename)\n" if $#ARGV != 0;

$counter = 0;
$machine = 0;
```

```
$job = 0;

$i, $j;

print "Schedule:           Local Search / Cmax\n";
print "  RGB:                223;181;201\n";
print "  Time:               0\n";

while ( <> )
{
 if ( ! /^#/ )
 {
  if ( $counter == 0 )
  {
   @values = split;
   $n = $values[0];
   $m = $values[1];
   $counter ++;

   #print $m, $n;
  }
  elsif ( $counter == 1 )
  {
   @antibody = split;
   $counter ++;
  }
  elsif ( $counter == 2 )
  {
   @values = split;
   $fitness = $values[0];
   $counter ++;

   #print $fitness;
  }
  elsif ( $counter == 3 )
  {
   @values = split;
   $schedule[0][ $machine ][ $job ] = $values[ 0 ];
   $schedule[1][ $machine ][ $job ] = $values[ 1 ];
   $schedule[2][ $machine ][ $job ] = $values[ 2 ];
```

```
   $job ++;

   if ( $machine == $m - 1 && $job == $n )
   {
    $counter ++;
   }

   if ( $job == $n )
   {
    $job = 0;
    $machine ++;
   }
  }
 }
}

for ( $i = 0; $i < $m; $i++ )
{
 printf "  Machine:            M%02d\n", $i + 1 ;

 for ( $j = 0; $j < $n; $j++ )
 {
  printf "    Oper:               J%02d\n", $schedule[0][$i][$j] + 1;
 }
}
```

Mostraremos un ejemplo para utilizar los programas.

Tenemos:

```
perl job.pl la01.txt > la01.job
perl mch.pl la01.txt > la01.mch
perl seq.pl la01.666.txt > la01.seq
```

En el ejemplo, $la01.txt$ es el archivo que contiene la definición del problema, en el formato de [8]. El archivo $la01.666.txt$ contiene la solución encontrada con nuestro algoritmo.
Una vez que se tienen los 3 archivos se puede abrir el proyecto en Lekin y utilizar esta aplicación.

Apéndice D

Trabajos publicados

Durante la realización de esta tesis se publicaron los siguientes trabajos y fueron presentados en sus respectivos eventos.

- *Use of an Artificial Immune System for Job Shop Scheduling* [23]
  Presentado en el segundo congreso *ICARIS 2003* (International Conference on Artificial Immune System) Septiembre 1 - 3 de 2003. Edimburgo, Escocia.

- *Uso de un Sistema Inmune Artificial para Problemas de Calendarización* [22]
  Presentado en el tercer congreso español *MAEB 04* (Metahurísticas, Algoritmos Evolutivos y Bioinspirados) Febrero 4 - 6 de 2004. Facultad de Ciencias del Trabajo, Universidad de Córdoba, España.

- *Job Shop Scheduling using the Clonal Selection Principle* [24]
  Presentado en el sexto congreso *ACDM 2004* (Adaptive Computing in Design and Manufacture) Abril 20 - 22 de 2004. Engineers House, Clifton, Bristol, Reino Unido.

Apéndice E

Código Fuente y Resultados

Para tener acceso al código fuente de la implementación presentada en esta tesis, escribir al autor:
dcortes@computacion.cs.cinvestav.mx.
Para ver los resultados detallados en su forma codificada y decodificada, se puede consultar el CD que acompaña este trabajo o bien al recurso en internet:
http://computacion.cs.cinvestav.mx/∼dcortes.

Bibliografía

[1] E. Aarts, Huub ten Eikelder, J. K. Lenstra, and R. Schilham. An adaptive memory programme embedding a taboo search algorithm with ns neighbourhood and constant time ta. Personnal communication, June 1999. Personnal communication to E. Taillard.

[2] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401, 1988.

[3] D. Applegate and W. Cook. *A computational study of the job-shop scheduling instance*. ORSA Journal on Computing 3, 149-156., 1991.

[4] V. A. Armentano and C. R. Scrich. Tabu search for minimizing total tardiness in a job shop. *Int. J. Production Economics*, 63:131–140, 2000.

[5] Tapan P. Bagchi. *MultiObjective Scheduling by Genetic Algorithms*. Kluwer Academic Publishers, New York, September 1999. ISBN 0-7923-8561-6.

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

[7] Egon Balas. Machine sequencing via disjunctive graph: An implicit enumeration algorithm. *Operation Research*, 17:941 – 957, 1969.

[8] J. E. Beasley. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operations Research Society*, 41(11):1069–1072, 1990.

[9] C. Bierwirth. A generalized permutation approach to job shop scheduling with genetic algorithms, 1995.

[10] Christian Bierwirth, Dirk C. Mattfeld, and Herbert Kopfer. On permutation representations for scheduling problems. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 310–318, Berlin, 1996. Springer.

[11] S. Binato, R. M. Aiex, and M. G. C. Resende. Parallel grasp with path relinking for job shop scheduling. 2002.

[12] S. Binato, W. J. Hery, D. M. Loewenstern, and M. G. C. Resende. A GRASP for Job Shop Scheduling. In Celso C. Ribeiro and Pierre Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 59–80. Kluwer Academic Publishers, Boston, 2001.

[13] J. Blazewicz, K. H. Ecker, G. Schmidt, and J. Weglarz. *Scheduling in Computer and Manufacturing System*. Springer, 1994.

[14] Wolfgang BrinkkötterandPeterBrucker. Solving open benchmark problems for the job shop problem.

[15] F. M. Burnet. *Clonal Selection and After*. In Theoretical Immunology, (Eds.) G. I. Bell, A., S. Perelson and G. H. Pimbley jr. and Marcel Dekker inc. edition, 1978.

[16] João Paulo Caldeira. hybrid evolutionary-tabu algorithm. personal communication to E. Taillard, 2003.

[17] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164 – 176, 1989.

[18] J. Carlier and E. Pinson. A practical use of jackson's preemptive schedule for solving the job-shop problem. *Annal of Operation Research*, 26:269–287, 1990.

[19] J. Carlier and E. Pinson. Adjustments of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78:146 – 161, 1994.

[20] M. C. Carol and A. P. Prodeus. Linkages of innate and adaptive immunity. *Current Opinion in Immunology*, (10):36 – 40, 1998.

[21] C. Colaco. Acquired wisdom in innate immunity. *Today Immunology*, 19(1):50, 1998.

[22] Daniel Cortés Rivera and Carlos A. Coello Coello. Uso de un Sistema Inmune Artificial para Problemas de Calendarización. In F. J. Martínez D. Ortiz y S. Ventura (editores) C. Hervás, N. García, editor, *Actas del III Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 04)*, pages 507 – 514, Córdoba, España, Febrero 2004. Universidad de Córdoba. ISBN 84-688-4224-9.

[23] Daniel Cortés Rivera, Carlos A. Coello Coello, and Nareli Cruz Cortés. Use of an Artificial Immune System for Job Shop Scheduling. In Jon Timmis, Peter Bentley, and Emma Hart, editors, *Second International Conference on Artificial Immune Systems (ICARIS'2003)*, pages 1–10, Edinburgh, Scotland, September 2003. Springer-Verlag. Lecture Notes in Computer Science Vol. 2787.

[24] Daniel Cortés Rivera, Carlos A. Coello Coello, and Nareli Cruz Cortés. Job shop scheduling using the clonal selection principle. In *ACDM'2004 (aceptado)*, UK, April 2004. Springer.

[25] Dipankar Dasgupta. Artificial neural network and artificial immune system: Similarities and differences. In *Proceedings of the IEEE SMC*, number 1, pages 873 – 878, 1997.

[26] Dipankar Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, 1999.

[27] MacCoy D. F. and Devaralan V. Artificial immune system and aerial image segmentation. In *Proceeding of the SMC'97*, pages 867 – 872, 1997.

[28] D. T. Fearon and R. M. Locksley. The instructive role of innate immunity in the acquired immune response. *Science*, 1(272):50 – 53, 1996.

[29] H. Fisher and G. L. Thompson. *Probabilistic learning combinations of local job-shop scheduling rules*. Prentice Hall, Englewood Cliffs, New Jersey, 225-251., industrial scheduling edition, 1963.

[30] David B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. The Institute of Electrical and Electronic Engineers, New York, 1995.

[31] S. French. *Sequencing and Scheduling. Mathematics and its applications.* Ellis Horwood Limited, 1982.

[32] F. Glover, J. P. Kelly, and M. Laguna. Genetic algorithms and tabu search: hybrids for optimization. *Computers Ops Res.*, 22:111–134, 1995.

[33] J. F. Gonçalves and N. C. Beirão. Um algoritmo genético baseado em chaves aleatórias para sequenciamento de operações. *Revista Associação Portuguesa de Desenvolvimento e Investigação Operacional*, 19:123 – 137, 1999.

[34] José Fernando Gonçalves, Jorge José Mendes, and Mauricio G. C. Resende. A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem. Technical Report TD-5EAL6J, AT&T Labs Research, 180 Park Avenue, Florham Park, NJ 07932 USA, September 2002.

[35] A. Henning. *Praktische Job-Shop Scheduling-Probleme*. PhD thesis, Department of Mathematics and Computer Science, Friedrich-Schiller-University Jena, 2002.

[36] Steven A. Hofmeyr. An interpretative introduction to the immune system. In I. Cohen & L. A. Segel, editor, *Design Principles for the Immune System and Other Distributed Autonomous Systems*. Oxford University Press, 2000.

[37] Steven A. Hofmeyr and S. Forrest. Immunity by desing: An artificial immune system. In *Proceedings of GECCO 99*, pages 1289 – 1296, 1999.

[38] J. H. Holland. *Adaptation in Natural and Artificial Systems, 4th Ed.* MIT Press., 1995.

[39] J. E. Hunt and D. E. Cook. Learning using an artificial immune system. *Journal of Network and Computer Applications*, 19:189 – 212, 1996.

[40] E. Balas J. Adams and D. Zawack. *The shifting bottleneck procedure for job shop scheduling.* Management Science 34, 391-401., 1988.

[41] A. S. Jain, B. Rangaswamy, and S. Meeran. New and "'stronger"' job shop neighbourhoods: A focus on the method of nowicki and smutnicki (1996). *J. Heuristics*, 6(4):457–480, 2000.

[42] N. K. Jarne. The immune system. *Scientific American*, 229(1):52 – 60, 1973.

[43] Albert Jones and Luis C. Rabelo. Survey of Job Shop Scheduling Techniques. NISTIR, National Institute of Standards and Technology, 1998.

[44] C. A. Janeway Jr. The immune system evolved to discriminate infectious nonself from noninfectious self. *Imm Today*, 13(1):11 – 16, 1992.

[45] C. A. Janeway Jr. How the immune system recognizes invaders. *Scientific American*, 1993.

[46] C. A. Janeway Jr and P. Travers. Immunobiology the immune system in health and disease. *Artes Médicas (in Portuguese), 2nd Ed.*, 1997.

[47] M. Kolonko. Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research*, 113(1):123–136, 1999.

[48] B. J. Lageweg. Personal communication with authors of jobshop1.txt, or-library. OR-library, 1984. http://mscmga.ms.ic.ac.uk/info.html.

[49] S. Lawrence. *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement),.* Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[50] M. D. Mannie. Immunological self/nonself discrimination. *Immunologic Research*, 19(1):65 – 87, 1999.

[51] P. Marrack and J. W. Kappler. How the immune system recognize the body. *Scientific American*, 269(3):49 – 55, 1993.

[52] P. D. Martin. *A Time-Oriented Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem*. PhD thesis, School of Operations Research & Industrial Engineering, Cornell University, Ithaca, New York, August 1996.

[53] Paul Martin and David B. Shmoys. A New Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Proceedings of the 5th International Conference on Integer Programming and Combinatorial Optimization, IPCO'96*, pages 389–403, Vancouver, British Columbia, Canada, 1996.

[54] H. Matsuo, C. J. Suh, and R. S. Sullivan. A controlled search simulated annealing method for the general jobshop scheduling problem. In *Working Paper*. Graduate School of Business, University of Texas, Austin, 1988.

[55] G. B. McMahon and M. Florian. On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research*, 23:475 – 482, 1975.

[56] R. Medzhitov and C. A. Janeway Jr. Innate immunity: Impact on the adaptive immune response. *Current Opinion in Immunology*, (9):4 – 9, 1997.

[57] R. Medzhitov and C. A. Janeway Jr. Innate immunity: The virtues of a nonclonal system of recognition. *Cell*, (91):295 – 298, 1997.

[58] R. Medzhitov and C. A. Janeway Jr. Innate immune recognition and control of adaptive immune responses. *Seminars in Immunoly*, (10):351 – 353., 1998.

[59] Daniel Merkle and Martin Middendorf. A new approach to solve permutation scheduling problems with ant colony optimization. In Egbert J. W. Boers, Stefano Cagnoni, Jens Gottlieb, Emma Hart, Pier Luca Lanzi, Gunther Raidl, Robert E. Smith, and Harald Tijink, editors, *Applications of Evolutionary Computing. EvoWorkshops2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM. Proceedings*, volume 2037, pages 484–494, Como, Italy, 18-19 2001. Springer-Verlag.

[60] Daniel Merkle and Martin Middendorf. Ant colony optimization with the relative pheromone evaluation method. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and Gunther Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279, pages 321–329, Kinsale, Ireland, 3-4 2002. Springer-Verlag.

[61] David S. Johnson Michael R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences).* W H Freeman & Co., June 1979. ISBN 0-7167-1045-5.

[62] Pablo Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts. Towards Memetic Algorithms. Technical Report 158–79, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, California, September 1989.

[63] B. A. Norman and J. C. Bean. A genetic algorithm methodology for complex scheduling problems. *Nav. Res. Logist.*, 46(2):199–211, 1999.

[64] Eugeniusz Nowicki and Czeslaw Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

[65] Leandro Nunes de Castro and Jon Timmis. *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm.* Springer-Verlag, 2002.

[66] Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immnue System: A New Computational Intelligence Approach.* Springer Verlang, Great Britain, September 2002. ISBN 1-8523-594-7.

[67] Leandro Nunes de Castro and Fernando José Von Zuben. Artificial immune systems: Part ii - a survey of applications. Technical Report DCA-RT 02/00, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, SP, Brazil, February 2000.

[68] Leandro Nunes de Castro and Fernando José Von Zuben. Learning and Optimization Using the Clonal Selection Principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, 2002.

[69] Leandro Nunes de Castro and Fernando José Von Zuben. Artificial Immune Systems: Part I - Basic Theory and Applications. Technical Report TR-DCA 01/99, FEEC/UNICAMP, Brazil, December 1999.

[70] Leandro Nunes de Castro and Fernando Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, Special Issue on Artificial Immune Systems, vol. 6, n. 3(3):239 – 251, 2002.

[71] A. Osyczka. *Multicriterion Optimization in engineering with FORTRAN programs.* Ellis Horwood Limited, 1984.

[72] C. R. Parish and E. R. O'Neill. Dependence of the adaptive immune response on innate immunity: Some questions answered but new paradoxes emerge. *Immunity and Cell Biology*, (75):523 – 527, 1997.

[73] A. S. Perelson, M. Mirmirani, and G. F. Oster. Optimal strategies in immunology i. b memory cell production. *J. Math. Biology*, (3):325 – 367, 1976.

[74] A. S. Perelson, M. Mirmirani, and G. F. Oster. Optimal strategies in immunology ii. b memory cell production. *J. Math. Biology*, (5):213 – 256, 1978.

[75] F. Pezzella and E. Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem, 2000.

[76] E. H. L. Aarts R. J. M. Vaessens and J. K. Lenstra. Job shop scheduling by local search. *INFORMS J. Comput.*, 8(3):302–317, 1996.

[77] Raúl Rojas. *Neural Networks. A Systematic Introduction.* Springer, Berlin, 1996.

[78] R. Schilham. Personal communication (to e. taillard)., 2000.

[79] R. H. Storer, S. D. Wu, and R. Vaccari. *New search spaces for sequencing instances with application to job shop scheduling,.* Management Science 38, 1495-1509., 1992.

[80] R. H. Storer, S. D. Wu, and R. Vaccari. Problem and heuristic space search strategies for job shop scheduling. *ORSA J. Comput.*, 7(4):453–467, 1995.

[81] E. D. Taillard. *Benchmarks for basic scheduling problems.* European Journal of Operational Research 64, Pages 278-285., 1993.

[82] E. D. Taillard. Parallel taboo search techniques for the job shop scheduling problem. *RSA J. Comput.*, 6(2):108–117, 1994.

[83] D. Tarlinton. Germinal centers: Form and function. *Current Operations in Immunology*, (10):245 – 251, 1998.

[84] T. Teich, M. Fischer, A. Vogel, and J. Fischer. Solving the job shop scheduling problem using ant algorithms. In *Proceedings of the 17th International Conference on CAD/CAM, Robotics and Factories of the Future*, pages 604–611, Durban, South Africa, 2001. Springer-Verlag.

[85] Tobias Teich, Marco Fischer, Andre Vogel, and Jens Fischer. A new ant colony algorithm for the job shop scheduling problem. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, page 803, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[86] S. Thomsen. Metaheuristics combined with branch & bound. Technical report, Copenhagen Business School, Copenhagen, Denmark, 1997.

[87] J. Timmis and M. Neal. Invetigatting the evolution and stability of a resource limited artificial immune system. In *Proc. of the Genetic and Evolutionary Computation Conference, Workshop on Artificial Immune System and Their Applications*, pages 40–41, 2000.

[88] S. Tonegawa. Somatic generation of antibody diversity. *Nature*, (302):575 – 581, 1983.

[89] user.pandora.be/richard.wheeler1/ais/inn.html. Technical report.

[90] R. J. M. Vaessens. Operation research library of problems. Management School, Imperial College london, 1996.

[91] P. J. M. van Laarhovenñad E. H. L. Aarts and J. K. Lenstra. Job shop scheduling by simulated annealing. *Oper. Res.*, 40(1):113–125, 1992.

[92] Ramiro Varela, Camino R. Vela, Jorge Puente, and Alberto Gomez. A knowledge-based evolutionary estrategy for scheduling problems with bottlenecks. *European Journal of Operational Research*, (145):57 – 71, 2003.

[93] I. L. Weissman and M. D. Cooper. How the immune system develops. *Scientific American*, 269(3):33 – 40, 1993.

[94] M. Wennink. Personal communitation to e. taillard. http://www.idsia.ch/∼eric/problemes.dir/ordonnancement.dir /job-shop.dir/best_lb_up.txt.

[95] Takeshi Yamada and Ryohei Nakano. *A genetic algorithm applicable to large-scale job-shop instances.* R. Manner, B. Manderick (eds.), Parallel instance solving from nature 2, North-Holland, Amsterdam, 281-290., 1992.

[96] Takeshi Yamada and Ryohei Nakano. Job-shop scheduling. In A. M. S. Zalzala and P. J. Fleming, editors, *Genetic Algorithms in Engineering Systems*, IEE control engineering series, chapter 7, pages 134–160. The Institution of Electrical Engineers, 1997.