TESIS DEFENDIDA POR

Abel García Nájera

Y aprobada por el siguiente comité:

Dr. Carlos Alberto Brizuela Rodríguez

Director del Comité

Dr. Pedro Gilberto López Mariscal

Miembro del Comité

Dr. Jesús Favela Vara

Miembro del Comité

Dr. Luis Alejandro Márquez Martínez

Miembro del Comité

Dr. Pedro Gilberto López Mariscal

Coordinador del programa en Ciencias de la Computación

Dr. Raúl Ramón Castro Escamilla

Director de Estudios de Posgrado

26 de Agosto del 2005.

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN
SUPERIOR DE ENSENADA

PROGRAMA DE POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN

Algoritmos Genéticos para Problemas de Ensamble de Tarjetas de Circuitos Impresos

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS

Presenta:

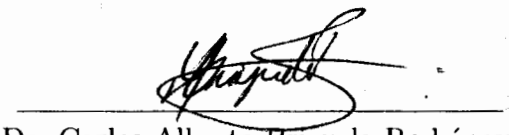
Abel García Nájera

Ensenada, Baja California, México. Agosto del 2005.

RESUMEN de la tesis que presenta **Abel García Nájera**, como requisito parcial para obtener el grado de MAESTRO EN CIENCIAS en CIENCIAS DE LA COMPUTACIÓN. Ensenada, B. C. Agosto del 2005.

Algoritmos Genéticos para Problemas de Ensamble de Tarjetas de Circuitos Impresos

Resumen aprobado por:



Dr. Carlos Alberto Brizuela Rodríguez

Director de Tesis

El proceso de manufactura de tarjetas de circuitos impresos consiste en la utilización de robots o máquinas de manufactura que se encargan de sujetar los componentes electrónicos y los colocan en sus respectivas ubicaciones en las tarjetas. En este proceso se identifican algunos problemas computacionales, los cuales pertecen a la clase de problemas para los cuales no se conoce un método eficiente de solución, es decir, pertenecen a la clase **NP-difícil**. Se han propuesto varias técnicas para encontrar soluciones aproximadas a estos problemas. Una de estas técnicas son los algoritmos genéticos.

Los algoritmos genéticos no garantizan desempeño alguno para encontrar soluciones óptimas, pero hoy en día son muy utilizados debido a los buenos resultados que se han obtenido en su uso en muchas aplicaciones.

En este trabajo se desarrolla un algoritmo genético para resolver el problema de manufactura de un tipo de tarjeta utilizando una máquina de manufactura y sus resultados se comparan con los de un método que se utiliza en la industria manufacturera, obteniendo una mejora, en promedio, del 13.93%. También se compara con otros algoritmos genéticos previamente propuestos, obteniendo mejores resultados para un caso de prueba en específico.

También se propone un algoritmo genético para solucionar el problema de manufactura de varios tipos de tarjetas utilizando una máquina de manufactura. No se conoce ningún trabajo previo que utilice un algoritmo genético para resolverlo. Los resultados de este algoritmo genético superan a los de una heurística recientemente propuesta.

La mayoría de los trabajos previos que han estudiado estos problemas, han utilizado casos de prueba que no están disponibles públicamente, esto dificulta la comparación de las diferentes técnicas propuestas. En esta tesis se propone también un método para generar casos de prueba para estos problemas y que se hace de dominio público.

Palabras clave: Algoritmos genéticos, PCB, tarjetas de circuitos impresos, ensamble, optimización combinatoria.

ABSTRACT of the thesis presented by **Abel García Nájera**, as a partial requirement to obtain the MASTER SCIENCE degree in COMPUTER SCIENCES. Ensenada, B. C. August 2005.

Genetic Algorithms for Printed Circuit Boards Assembly Problems

Abstract approved by:

Dr. Carlos Alberto Brizuela Rodríguez

Thesis director

The printed circuit board manufacturing process consists of the utilization of robots or manufacturing machines that take certain electronic components and place them in their respective locations in the board. A few problems arise in this process, which belong to the class of problems for which there is no known efficient method of solution, that is, they belong to the **NP-Hard** class of problems. There are several techniques that have been proposed to find approximated solutions to these problems. One of these techniques is based on genetic algorithms.

Genetic algorithms do not guarantee any performance to find optimal solutions, but nowadays they are widely utilized due to the good results they have provided in many applications.

In this work a genetic algorithm is developed to solve the one board type manufacturing problem using one manufacturing machine, and its results are compared with those from a method that is utilized in the manufacturing industry, improving them, on average, by 13.93%. Likewise, this genetic algorithm is compared with others previously proposed, obtaining better results.

Also, a genetic algorithm is proposed to solve the many types of boards manufacturing problem using one manufacturing machine. There is no known prior work which utilizes a genetic algorithm to solve it. The results from this genetic algorithm improves the ones from a recently proposed heuristic.

Most of the previous works related to the study of these problems used test cases that are not publicly available. This makes it difficult to compare different techniques. In this thesis a method to generate test cases for these problems is also proposed.

Keywords: Genetic algorithms, PCB, printed circuit boards, assembly, combinatorial optimization.

A mi mamá

Agradecimientos

Gracias mamá por darme todo incondicionalmente. Esto es más que un agradecimiento, es una muestra de admiración, respeto y amor.

Hay personas quienes gustan de compartir sus conocimientos y experiencias. Es grato platicar con ellas. Gracias profes Carlos Brizuela y Gilbero López por todo su apoyo que recibí durante estos dos años.

Gracias Marcel por brindarme tu confianza, tu cariño y y tu apoyo en todo momento.

Tengo la firme intención de mejorar día a día. Gracias Dres. Jesús Favela y Alejandro Márquez por ayudarme a conseguirlo al dedicar tiempo para conocer y revisar mi trabajo de tesis.

Siempre es bueno contar con apoyo moral, recibir palabras de aliento y, en ocasiones, apoyo económico. Gracias Je, Alex, Bety, Marilú, Felipe y Héctor.

Al estar lejos de casa es bueno contar con personas que nos hacen más grata la estadía, compartiendo sueños, experiencias, riñas, alegrías, entretenimiento, cerveza y fútbol. Gracias Tentori, Alhuey Hills, JBurci y MTN.

Gracias Chésar, Deivid, Leo y Petersama por los ratos de sano esparcimiento.

Gracias profe Jorge Torres por proporcionarme espacio en su laboratorio y equipo de cómputo durante la realización de mi tesis. Gracias Dr. Rafael Kelly por sus valiosos comentarios.

Gracias también Vero, Ariel, Poncho, VHugoT y DCC por ayudarme a viajar y presentar parte de mi trabajo de tesis.

Mis estudios no hubieran sido posibles sin que CICESE y CONACYT hayan aceptado mis solicitudes de admisión y beca, respectivamente. Gracias a ambas entidades.

Ensenada, México — Abel García Nájera
26 de Agosto del 2005.

Tabla de Contenido

Capítulo Página

Tabla de Contenido (Continuación)

Tabla de Contenido (Continuación)

Lista de Figuras

Lista de Figuras (Continuación)

Lista de Figuras (Continuación)

Lista de Tablas

Capítulo I

Introducción

"Comienza por el principio y sigue hasta que llegues al final, entonces detente."

Lewis Carroll (1832 - 1898), Alicia en el país de las maravillas, 1865.

Las compañias de manufactura de équipos electrónicos pretenden lograr volúmenes de producción muy altos en el menor tiempo posible. Por esta razón es que invierten una gran cantidad de recursos con el fin de que sus procesos sean automáticos, mediante la integración de sistemas electrónicos en sus plantas de manufactura, lo que involucra tener técnicas eficientes para facilitar el control de la producción.

Dentro de las industrias hay muchos procesos y máquinas automáticas de manufactura. Una de estas máquinas es la de manufactura de tarjetas de circuitos impresos.

I.1 La manufactura de tarjetas de circuitos impresos

Hoy en día, la manufactura de tarjetas de circuitos impresos (PCB por sus siglas en inglés) basa sus tareas en el uso de máquinas de manufactura. Estas máquinas se programan de acuerdo a las características de las PCB que se están ensamblando al momento.

La tarea de manufactura de PCB requiere que los componentes que se montarán en ellas se tomen en sus respectivas ubicaciones de sujeción y se coloquen en sus sitios correspondientes en las PCB. Una PCB se ilustra en la Figura 1. En esta figura se pueden observar las posiciones en donde se montarán los componentes.

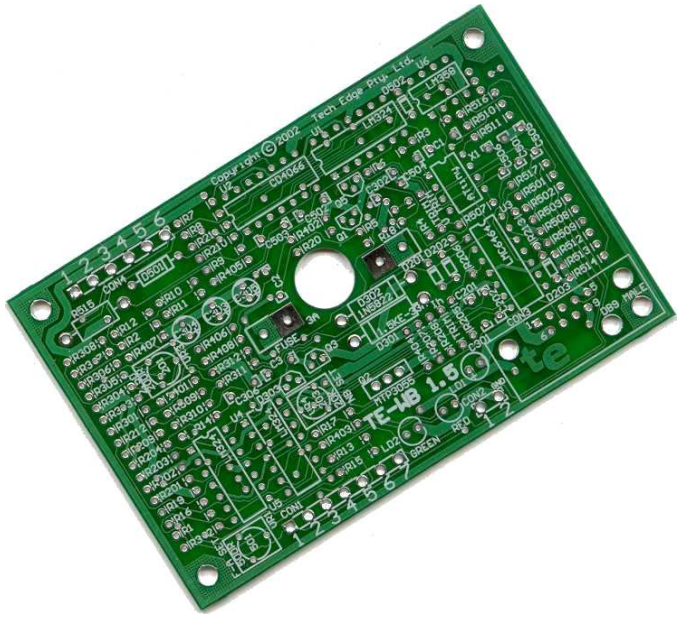


Figura 1. Una tarjeta de circuitos impresos (PCB) con las ubicaciones de los componentes (tomada de Tech Edge Pty. Ltd., http://www.techedge.com.au/.)

Para lograr los objetivos de producción, las máquinas necesitan ser configuradas y/o programadas eficientemente con el fín de, por ejemplo, reducir el consumo de energía o minimizar el tiempo de ensamble. Esto último da lugar a que surjan algunos *problemas de optimización combinatoria* que se tienen que resolver y a los cuales se tiene que dar solución.

Existen en el mercado varios tipos de máquinas de manufactura. Algunas de las copañías que fabrican éstas son: *Fuji Machine Mfg. Co.*[1], *Automated Production Systems, Inc.*[2] y *Assembléon B.V.*[3] Una máquina de una de estas compañías se muestra ilustrada en la Figura 2.

Si a la diversidad del tipo de máquinas se suma que éstas pueden ensamblar uno o varios tipos de PCB a la vez, resulta en que la dificultad varía para la solución de los problemas relacionados con este proceso.

[1]http://smt.fuji.co.jp/
[2]http://www.apsgold.com/
[3]http://www.assembleon.com/

Figura 2. Máquina de manufactura de PCB modelo L60 de Automated Production Systems, Inc.

I.2 Optimización combinatoria

Como se planteó anteriormente, los problemas de manufactura de PCB requieren la minimización del tiempo de ensamble de toda la producción. Para esto se tienen que encontrar soluciones factibles y, de ser posible, la solución que proponga el menor tiempo de manufactura de entre todas las soluciones (solución *óptima* del problema).

Los problemas de optimización se dividen naturalmente en dos categorías: aquellos con variables continuas y aquellos con variables discretas. A estos últimos se les conoce como *problemas de optimización combinatoria* (Papadimitriou y Steiglitz, 1998).

En los problemas de optimización combinatoria hay un número finito de combinaciones posibles, y en principio, se pueden enumerar todas ellas, eliminar las que no cumplen con las condiciones, y de entre las que si las cumplen seleccionar la que tenga el menor costo. La búsqueda de dicha combinación debe ser más inteligente que la enumeración completa, debido al gran número de combinaciones que pueden estar involucradas (Mitchell *et al.*, 1998).

La optimización combinatoria trata con problemas que requieren maximizar o minimizar una función de varias variables, la cual está sujeta a restricciones de igualdad, desigualdad y de integridad en algunas o en todas las variables (Nemhauser y Wolsey, 1999).

Un problema de optimización es un par (F, c), donde F es el conjunto de soluciones factibles, y c la función de costo, $c : F \longrightarrow \Re$. El problema es encontrar una $f \in F$ tal que

$$c(f) \leq c(y), \qquad \forall\ y \in F \tag{1}$$

A dicha solución f se le llama la solución *óptima global*, o, cuando no hay lugar a confusión, simplemente la solución *óptima* (Papadimitriou y Steiglitz, 1998).

En caso de que el conjunto de soluciones factibles F esté definido por una familia de igualdades y desigualdades en $\Re^n$ y todas las restricciones y la función objetivo sean lineales, se dice que el problema en (1) es un *programa lineal*. En caso de que al menos una de las restricciones o la función objetivo sea no-lineal, al problema en (1) se le llama *programa no-lineal*.

En la mayoría de los casos, los problemas de optimización combinatoria pueden ser formulados como programas enteros lineales o no-lineales y son solucionados utilizando métodos que explotan de forma eficiente la región factible (Giannessi y Tardella, 1998).

La mayoría de los problemas de optimización combinatoria del mundo real son difíciles de resolver, y para los cuales no se conocen técnicas eficientes para encontrar la mejor solución. Es por esta razón que es común utilizar heurísticas que propongan buenas soluciones, aunque probablemente no las óptimas.

I.3 Heurísticas

Con el fin de resolver problemas difíciles eficientemente, es común que se utilicen métodos que no garantizan encontrar la mejor solución, pero siempre encontrará una buena. Una heurística es una técnica que mejora la eficiencia de un proceso de búsqueda, posiblemente sacrificando algunos puntos dentro del espacio de soluciones. Usando buenas heurísticas, se pueden esperar buenas soluciones, aunque posiblemente no óptimas, a problemas difíciles en un tiempo polinomial en el tamaño del problema. Hay algunas buenas heurísticas de propósito general que se usan en una gran variedad de problemas. Además, es posible construir heurísticas de próposito especial que exploten el conocimiento que se tiene del dominio específico para resolver algunos problemas en particular (Rich y Knight, 1991).

Algunos ejemplos de heurísticas son los siguientes:

- *templado simulado*,
- *búsqueda tabú*,
- *vecinos cercanos*,
- *búsqueda local*, y
- *algoritmos genéticos*.

Estas heurísticas son de propósito general, aunque algunas funcionan mejor que otras para ciertos problemas, por ejemplo, *k-opt* (de búsqueda local) (Papadimitriou y Steiglitz, 1998), en específico 3-*opt*, obtiene los mejores resultados para el *problema del viajante* (TSP por sus siglas en inglés) sobre casos típicos (Gutin y Punnen, 2002).

I.4 Algoritmos genéticos

La idea escencial de la solución *evolutiva* de problemas es la siguiente. Un conjunto de soluciones candidatas al problema evoluciona a través de iteraciones sucesivas de variación aleatoria y selección. La variación aleatoria provee el mecanismo para descubrir nuevas soluciones, mientras que la selección determina cuáles soluciones se mantienen como base para las siguientes exploraciones. El objetivo de la exploración es localizar una solución, o un conjunto de éstas, que tengan la calidad necesaria. No sólo es suficiente encontrar estas soluciones, sino que se deben encontrar rápido. La velocidad con que pueden ser encontradas depende, en parte, de la representación que se haga del problema, de la función de evaluación de las soluciones, de los operadores específicos de variación y selección, y del tamaño e inicialización de la población (Michalewicz y Fogel, 2000).

Un *algoritmo genético* comienza con un multiconjunto de n soluciones generadas aleatoriamente, llamado población. Cada solución, o *individuo*, se representa como una cadena sobre un alfabeto finito. Cada solución es evaluada por una función, la *función de aptitud*, que devuelve valores más altos para mejores soluciones (Russell y Norvig, 2003). En la *cruza*, las soluciones se combinan por pares para producir nuevas soluciones, y con esto dar lugar a que la *mutación* actue, que es la variación de la información contenida dentro de un individuo. Finalmente, se *seleccionan* los mejores individuos que se someterán a la acción de los operadores en la siguiente iteración.

Los algoritmos genéticos, por ser una heurística, no garantizan encontrar la solución óptima, pero hoy en día, debido a los buenos resultados publicados en muchas aplicaciones, son utilizados para solucionar diversos problemas de optimización combinatoria.

I.5 Motivación

El problema bajo la presente investigación es muy importante para la industria, ya que las empresas manufactureras buscan lograr grandes volúmenes de producción en el menor tiempo posible, y si se trata de organizaciones que utilizan sistemas electrónicos de manufactura, es necesario contar con técnicas eficientes que permitan realizar la planeación de la producción.

Además, la industria manufacturera invierte una gran cantidad de dinero para que sus procesos de manufactura sean automatizados. Con ello esperan el retorno de capital y un mejor aprovechamiento de la maquinaria.

Por otro lado, en el contexto computacional, la dificultad que representa esta tarea es un aliciente para esta investigación, pues los problemas de manufactura de PCB se encuentran dentro de una clase de problemas para los que no se conoce una forma general eficiente de encontrar la solución óptima a los mismos.

Así mismo, existe escasa investigación respecto a los problemas de manufactura de varios tipos de PCB, utilizando para ello una o varias máquinas (los cuales se detallarán en el Capítulo II), por lo que se puede extender la investigación existente para lograr entender el problema a fondo y poder descubrir factores que pueden ayudar a solucionarlos.

I.6 Objetivos general y específicos

I.6.1 Objetivo general

Realizar un estudio comparativo entre los algoritmos genéticos y otras heurísticas, cuando éstos son aplicados al problema de manufactura de uno y varios tipos de PCB, utilizando una sola máquina para este propósito. Lo anterior con el fin de determinar

cuál de ellos es el mejor método para solucionar el problema.

I.6.2 Objetivos específicos

a) Entender los problemas de manufactura de uno y varios tipos de PCB utilizando una máquina.

b) Entender la aplicación del cómputo evolutivo para los problemas de manufactura de PCB.

c) Entender el funcionamiento de las heurísticas que se utilizan en la industria para resolver el problema de manufactura de un tipo de PCB con una máquina.

d) Proponer un algortimo genético para los problemas de manufactura de uno y varios tipos de PCB utilizando una máquina.

e) Diseñar casos de prueba para los problemas definidos.

f) Determinar el desempeño de los algoritmos estudiados sobre los casos de prueba generados.

g) Obtener cotas inferiores para los casos de estudio.

I.7 Organización de la tesis

Esta tesis contiene cinco capítulos y tres apéndices, y está organizada de la siguiente forma.

En el Capítulo II se describe el proceso de manufactura de tarjetas de circuitos impresos y los problemas que resultan del mismo. Se formulan y se modelan matemáticamente dos de ellos y se hace un breve repaso de los trabajos que hay respecto al estudio de éstos, en los que intervienen diversas técnicas para solucionarlos. En este

capítulo también se incluye la descripción de algunas técnicas utilizadas en la industría para resolver el problema de manufactura de un tipo de PCB con una máquina, y también se describe una heurística propuesta recientemente para resolver el problema de manufactura de varios tipos de PCB con una máquina.

El Capítulo III contiene la metodología a utilizar para la solución de los problemas de manufactura descritos en el capítulo anterior. Se incluye una explicación de los algoritmos genéticos, de la representación genética de los problemas de estudio y de los operadores genéticos que se utilizan en los algoritmos propuestos para cada problema.

Dentro del Capítulo IV se explica el diseño de los experimentos llevados a cabo, los cuales consisten básicamente en aplicar los algoritmos géneticos a un conjunto de casos de prueba generados. Los resultados de estos experimentos se presentan en este capítulo. También se aplican otros métodos sobre estos casos y sus resultados se comparan con los obtenidos con los algoritmos genéticos.

En el Capítulo V se presenta un resumen del trabajo realizado, así como las conclusiones del mismo y algunas ideas para trabajos futuros.

Un método de generación de casos de prueba para los problemas de estudio se presenta en el Apéndice A. La razón de proponer este método es que públicamente no existen, o en el mejor de los casos son pocos, los casos de prueba para estos problemas.

En el Apéndice B se presenta el detalle de algunos de los casos de prueba generados con el método descrito en el apéndice anterior. El resto de los casos de prueba están contenidos en el disco compacto que acompaña a esta tesis.

Finalmente, en el Apéndice C se modela al problema de manufactura de un tipo de PCB con una máquina como un problema de programación lineal.

Capítulo II

Los problemas de ensamble de PCB

"Veo que una gran parte de la información que tengo la he adquirido buscando algo y encontrando algo más en el camino."
Franklin Pierce Adams (1881 - 1960)

Los problemas de ensamble de PCB varían desde decidir la secuencia de colocación de componentes, hasta problemas de cómo poner en operación la planta entera para una producción eficiente. Debido a la gran variedad de problemas de planeación y control, la descomposición jerárquica es un enfoque extensamente utilizado para resolver estos problemas. Los problemas se relacionan unos con otros, de tal forma que para resolver los problemas más complejos se requieren las soluciones de los menos complejos (Johnsson y Smed, 2001).

II.1 Funcionamiento de una máquina de manufactura

El incremento de componentes a ser colocados en una sola tarjeta ha hecho que la reducción del tiempo de ensamble sea probablemente el objetivo más importante para reducir los costos de producción y aumentar la productividad (Burke *et al.*, 1999). En la actualidad existe una gran diversidad de máquinas de manufactura que permiten lograr este objetivo. Dos de éstas se muestran esquematizadas en la Figura 3.

Aun cuando se tratan de diferentes tipos de máquinas, el proceso de manufactura en ellas es similar. Por esta razón, el presente trabajo de investigación se enfoca en el funcionamiento de una máquina del tipo *pick and place*, como la que se muestra en la Figura 3(b).

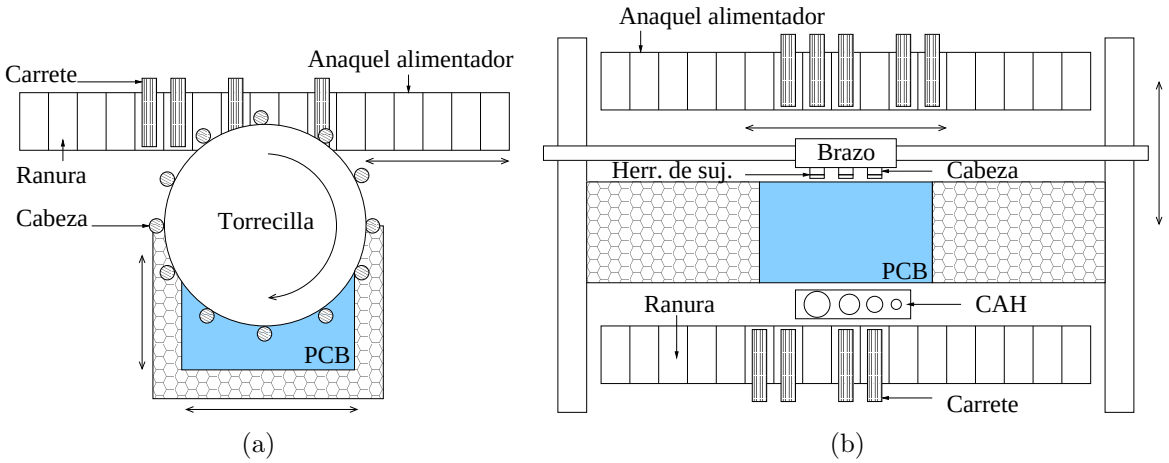


Figura 3. Dos tipos de máquinas de manufactura de PCB: (a) *chip shooter*; y (b) *pick and place*.

La funcionalidad es la siguiente (van Laarhoven y Zijm, 1993). Una PCB se transporta dentro de la máquina sobre una banda y está fija en una posición predeterminada para su manufactura. Existen uno o dos *anaqueles alimentadores* que están localizados a los lados de la banda transportadora y cada uno tiene cierto número de *ranuras* disponibles. Los *carretes* contienen los componentes que van a ser montados en la PCB y cada uno de ellos contiene sólo un tipo de componente, como resistores, capacitores, transistores, circuitos integrados, etc. Cada ranura puede alojar un carrete, mientras que un carrete puede ocupar una o más ranuras.

La máquina cuenta con un *brazo mecánico*, que puede tener una o varias *cabezas*. Las cabezas levantan los componentes de los anaqueles alimentadores y los colocan en la tarjeta. Cada cabeza puede sujetar un componente a la vez. Estas cabezas son independientes para levantar, rotar, sujetar y colocar. A las cabezas se les colocan unas *herramientas de sujeción* que sostienen un componente hasta que se coloca en la tarjeta. Cada tipo de componente puede ser sujetado por un subconjunto de estas herramientas, es decir, una cabeza con una herramienta dada sólo puede sujetar componentes de un conjunto limitado de tipos de componentes. Las herramientas se cambian

automáticamente en el *cambiador automático de herramientas* (CAH) cuando éstas no pueden sujetar al componente requerido.

El proceso de ensamble se puede dividir en dos fases. En la primera, el brazo se mueve hacia el anaquel alimentador y las cabezas sujetan secuencialmente los componentes a ser montados. En la segunda, el brazo se mueve hacia la PCB y las cabezas colocan secuencialmente los componentes en sus respectivas ubicaciones. En caso de que se requiera hacer un cambio de herramienta, se hace antes de la fase inicial.

II.2 Clasificación de los problemas de manufactura de PCB

Los problemas de manufactura de PCB se pueden clasificar, de acuerdo al número de tipos de tarjetas que se van a manufacturar y al número de máquinas de manufactura que se utilizarán para ello (como lo muestra la Figura 4), en los siguientes tipos (Johnsson y Smed, 2001):

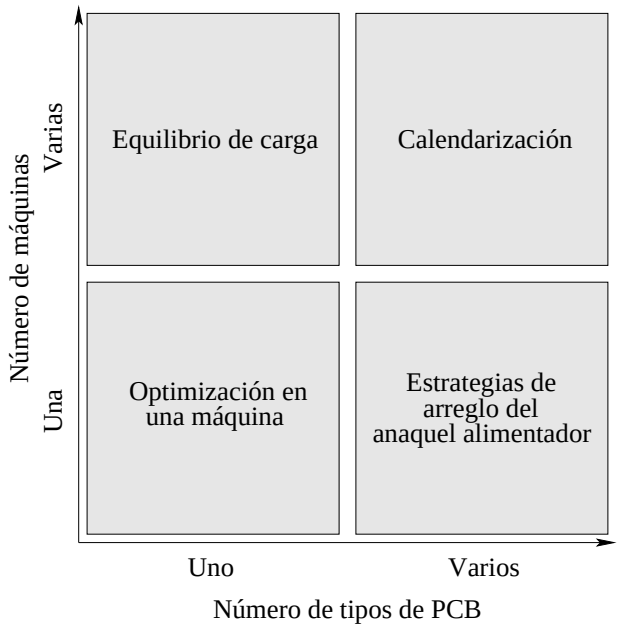


Figura 4. Clasificación de problemas de manufactura de PCB.

1. Un tipo de PCB y una máquina (problema 1-1). Esta clase comprende problemas de optimización en una sola máquina, que involucran la ubicación de los componentes dentro del anaquel alimentador y la secuencia de colocación de los mismos.

2. Un tipo de PCB y varias máquinas (problema 1-V). Esta clase se concentra en la asignación de componentes a máquinas secuenciales de manufactura, donde el objetivo es equilibrar la carga de trabajo en las máquinas.

3. Varios tipos de PCB y una máquina (problema V-1). En esta clase de problemas se determina la estrategia de arreglo del anaquel alimentador (estrategia de colocación de carretes en las ranuras) para una sola máquina, esto con el fin de minimizar el número de cambios a realizar en el anaquel alimentador.

4. Varios tipos de PCB y varias máquinas (problema V-V). Ésta es una clase de problemas de calendarización que principalmente se concentra en la asignación de tareas a líneas de producción y el secuenciamiento de las líneas de producción.

II.3 El problema 1-1

Como se describió en la sección anterior, en este tipo de problemas se manufactura un tipo de PCB con una sola máquina. La reducción del tiempo de manufactura bajo este esquema da pie a los siguientes subproblemas:

a) Ubicación de los tipos de componentes dentro del anaquel alimentador, es decir, la asignación de las ranuras a los tipos de componentes, al que llamaremos *problema de asignación de ranuras* (PAR),

b) Secuencia de colocación de los componentes, al que llamaremos *problema de secuencia de colocación* (PSC), y

c) Decidir de cuál ubicación, dentro del anaquel alimentador, sujetar un componente de cierto tipo, cuando éste se asigna a más de una ranura, al que llamaremos *plan de recuperación* (PR).

El problema hasta aquí descrito pertenece a la clase **NP-difícil** (van Laarhoven y Zijm, 1993).

Por simplicidad, se considera que los carretes ocupan sólo una ranura y, debido a que es deseable reducir el número de carretes en las máquinas de manufactura (Crama *et al.*, 1997), un carrete puede ser asignado sólamente a una ranura. Con estas consideraciones, el plan de recuperación queda resuelto.

Otra consideración importante es que el tiempo que tardan las cabezas exclusivamente para sujetar y colocar un componente, por estar implícito en el proceso de manufactura, no interviene en la optimización del tiempo total.

Así mismo, la velocidad de desplazamiento del brazo mecánico se considera constante, aun cuando en la realidad se tengan aceleraciones y desaceleraciones.

II.3.1 Trabajo previo

El problema 1-1 se ha venido estudiando desde hace al menos dos décadas. En esta sección se describen algunos de los trabajos que han analizado a éste, los cuales hacen uso de técnicas, para la solución del problema, que van desde heurísticas sencillas, pasando por algoritmos genéticos y templado simulado, hasta metodologías exactas como la programación dinámica y la programación lineal entera.

Ball y Magazine, 1988. En este trabajo se considera que el problema de asignación de ranuras está resuelto y se modela al PSC como un caso del *problema del cartero rural*. Se presenta una formulación matemática de programación lineal entera para éste y se propone la heurística *balancear y conectar* (*balance and connect*) para resolverlo.

Leipälä y Nevalainen, 1989. Se considera al PAR como un *problema de asignación cuadrática* y al PSC como un *problema del viajante asimétrico tridimensional*. El problema se formula matemáticamente y se resuelve mediante heurísticas.

Leu *et al.*, 1993. En este trabajo los autores aplican un algoritmo genético para resolver ambos problemas en conjunto, esto para tres tipos de máquinas de manufactura de PCB diferentes. Se proporciona un caso detallado de una PCB sobre el cual se aplica el algoritmo propuesto.

van Laarhoven y Zijm, 1993. En este caso los autores hacen un planteamiento matemático de ambos problemas, PAR y PSC, y los resuelven mediante heurísticas basadas en templado simulado.

Kumar y Li, 1995. En éste trabajo se trata al PAR como un caso del *problema de apareamiento de peso mínimo* (MWMP por sus siglas en inglés) y al PSC como uno del *problema del viajante*, para obtener una cota superior. Por otro lado, el problema en general se define como un caso de *programación lineal entera* para obtener una cota inferior. Para obtener soluciones del MWMP y del problema de programación lineal entera se utiliza software comercial y para el TSP se utilizan heurísticas bien conocidas como vecinos cercanos, 2-*opt* y 3-*opt*, entre otras. Los resultados en este trabajo reflejan una mejora, en promedio, del 24% sobre las técnicas utilizadas por la industria.

Dikos *et al.*, 1997. Se usa un algoritmo genético para resolver el PAR cuando la secuencia de colocación está predeterminada para el tipo de PCB que se va a manufacturar. Este trabajo toma en cuenta que los carretes pueden ocupar más de una ranura.

Lee *et al.*, 1997. Se formula al PAR como un *problema de programación entera cuadrática* y al PSC como un caso del *problema del viajante*. Para encontrar soluciones al primero se propone una heurística que asigna un carrete a la ranura más cercana a las ubicaciones de los componentes en dicho carrete. Para el segundo se utiliza la heurística de vecinos cercanos. Los resultados obtenidos en este caso se comparan con los de una secuencia de colocación aleatoria, dando una mejora del 31%.

Lee *et al.*, 1998. En éste, se hace un preprocesamiento del PAR utilizando heurísticas sencillas con el fin de formularlo como un *problema de programación entera* y se utiliza programación dinámica para dar soluciones al mismo. En el caso del PSC se utiliza la misma heurística que en (Lee *et al.*, 1997), dando como resultado una mejora del 3% respecto a los resultados obtenidos en este último trabajo.

Ong y Khoo, 1999. Utilizan un algoritmo genético para encontrar soluciones a los problemas PAR y PSC en conjunto. Se aplica este algoritmo sobre el caso propuesto por Leu *et al.* (1993) y lo mejora en 7%.

Wang *et al.*, 1999. Se utiliza un algoritmo genético para encontrar soluciones a los problemas PAR y PSC. Se proponen varias técnicas de selección y de cruzamiento, comparando la calidad de sus soluciones con algunas otras técnicas en general. Los resultados obtuvieron una mejora del 7.2% respecto a los propuestos por el software propietario de una compañia de fabricación de máquinas de manufactura de PCB.

Lee *et al.*, 2000. Se utiliza un algoritmo genético para resolver ambos problemas, PAR y PSC, simultaneamente. Se presenta primero un procedimiento para el caso de una máquina con una cabeza, para posteriormente aplicar éste al problema con tres cabezas. Se obtenien resultados con una mejora del 16% sobre los obtenidos utilizando

búsqueda voraz para la asignación de ranuras y *vecinos cercanos* para la secuencia de colocación.

Ellis *et al.*, 2001. Se desarrolla un algoritmo para encontrar soluciones para el PAR y el PSC, que consiste en un procedimiento que utiliza un conjunto de reglas para generar una solucion inicial de la asignación de ranuras y de la secuencia de colocación. Después se utiliza un procedimiento, basado en la heurística 2-*opt* (Papadimitriou y Steiglitz, 1998), para mejorar estas soluciones.

Jeevan *et al.*, 2002. Se propone un algoritmo genético para la solución del PSC, considerando una máquina que puede utilizar desde una hasta cuatro cabezas para la sujeción y colocación de componentes. Muestran resultados comparando experimentos propios, haciendo uso de diferente cantidad de cabezas en ellos.

Ho y Ji, 2003. En este trabajo los autores utilizan un algoritmo genético *híbrido* (AGH) para dar soluciones a los problemas PAR y PSC simultaneamente. En este trabajo se obtiene una mejora de casi 50% sobre los resultados publicados por Leu *et al.* (1993).

Kumar y Luo, 2003. Se utiliza programación no lineal entera para formular el problema de manufactura de PCB y modelarlo como un caso del *problema del viajante generalizado*. Se hacen consideraciones realistas para convertirlo en un *problema de programación lineal entera* y modelarlo como un caso del *problema del viajante*. Posteriormente se aplican heuríticas para dar soluciones a este problema. Los resultados reportados mejoran en 28% a los obtenidos por los métodos utilizados en la industria.

Ho y Ji, 2005. Se propone un AGH para encontrar soluciones al problema 1-1 y aplicarlo al caso descrito por Leu *et al.* (1993). El resultado refleja un 0.23% de mejoría

respecto al publicado por Ong y Khoo (1999), y es el mejor resultado reportado hasta el momento para dicho caso.

En resumen, el trabajo previo respecto al problema 1-1 se puede dividir en dos enfoques. El primero es el modelado matemático del problema y uso de métodos exactos (i.e. programación lineal entera y programación dinámica), así como su resolución con heurísticas. La desventaja de estos métodos es que están limitados a trabajar sobre casos de hasta cierto tamaño.

El segundo enfoque es el uso del cómputo evolutivo, en específico, los algoritmos genéticos, para encontrar soluciones al problema. En este caso, las variantes de los algoritmos genéticos propuestos se introducen con los diversos operadores genéticos que se utilizan en ellos.

Por otro lado, la mayoría de los trabajos han ocupado casos de prueba diferentes, lo que hace difícil la comparación de los métodos propuestos.

II.3.2 Métodos que se utilizan en la industria para resolver el problema 1-1

En general, en la industria, para solucionar el problema de asignación de ranuras, se utiliza un algoritmo de *búsqueda voraz* (BV) (Kumar y Li, 1995), es decir, se prueba un carrete en cada ranura y se registra el tiempo de colocación de todos los componentes de ese carrete. El carrete se asigna a la ranura en donde se obtenga el menor tiempo. La prioridad de asignación de ranuras a carretes se basa en la cantidad de componentes de cada carrete.

En el caso del problema de secuencia de colocación de componentes, se utilizan los dos métodos que se describen a continuación (Kumar y Li, 1995), los cuales consideran a la PCB como un rectángulo en el plano euclidiano, cuyos límites son paralelos a los ejes coordenados x y y:

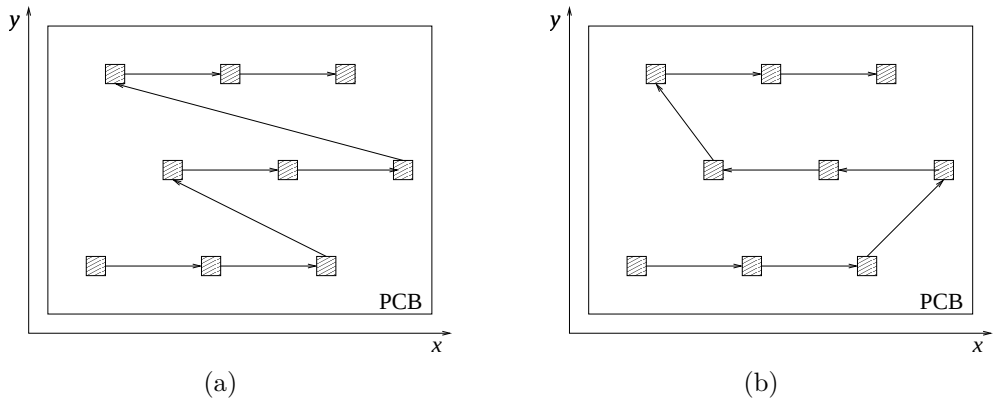


Figura 5. Métodos que se utilizan en la industria para resolver el PSC: (a) método *type-writer*; y (b) método *s-shape*.

i. *Type-writer* (TW). Los componentes se ordenan de acuerdo a la coordenada y de su ubicación en la PCB en forma ascendente. En caso de que dos componentes tengan la misma coordenada y, se ordenan de acuerdo a su coordenada x en forma ascendente, como se muestra en la Figura 5(a).

ii. *S-shape* (SS). Los componentes se ordenan de acuerdo a la coordenada y de su ubicación en la PCB en forma ascendente. En caso de que dos componentes tengan la misma coordenada y, se ordenan de acuerdo a su coordenada x en forma ascendente y descendente, de manera alternada, como se muestra en la Figura 5(b).

II.3.3 Modelo matemático del problema 1-1

Para presentar el modelo matemático se introduce la siguiente notación:

N = número total de componentes,

M = número total de tipos de componentes,

R = número de ranuras disponibles,

$\tau(i)$ = función que devuelve el tipo del componente i,

$\tau : \{1, ..., N\} \longrightarrow \{1, ..., M\}$,

h_{ik} = 1 si los componentes i y k pueden ser sujetados con la misma herramienta,
0 en otro caso,

t_{ij} = tiempo de desplazamiento entre la ubicación del componente i y la ranura j,

t_{ij}^{CAH} = tiempo de desplazamiento entre la ubicación del componente i y la ranura j, pasando por el CAH para hacer un cambio de herramienta.

Sea x_{ik} una variable de decisión igual a 1 si el componente i es montado justo antes que el componente k y 0 si no.

Sea y_{jl} una variable de decisión igual a 1 si la ranura j contiene a los componentes de tipo l y 0 si no.

Supongamos que el componente i se coloca justo antes que el componente k ($x_{ik} = 1$), que ambos componentes se pueden sujetar con las mismas pinzas ($h_{ik} = 1$) y que el componente k se encuentra en la ranura j ($y_{j\tau(k)} = 1$), entonces el tiempo que le toma al robot trasladarse desde la ubicación del componente i a la ranura j, sujetar al componente k y colocar a éste en su ubicación, está dado por $t_{ij}+t_{kj}$. Si los componentes no pueden ser sujetados con las mismas pinzas ($h_{ik} = 0$) este tiempo es $t_{ij}^{CAH} + t_{kj}$.

De esta forma, el problema P1-1 de manufactura de un tipo de PCB con una máquina puede ser formulado de la manera siguiente:

$$\underset{x_{ik},\, y_{jl} \in \{0,1\}}{\text{minimizar}}\ S = \sum_{i \le N} \sum_{j \le R} \sum_{k \le N} \Big[(t_{ij} h_{ik} + t_{ij}^{CAH}(1 - h_{ik})) + t_{kj} \Big] x_{ik} y_{j\tau(k)} \tag{2}$$

sujeto a las siguientes restricciones:

$$\sum_{k \le N} x_{ik} = 1, \quad \forall\, i \in \{1, ..., N\} \tag{3}$$

$$\sum_{i \le N} x_{ik} = 1, \quad \forall\, k \in \{1, ..., N\} \tag{4}$$

$$\sum_{i \in \mathbf{Q}} \sum_{k \in \mathbf{Q}} x_{ik} \le |\mathbf{Q}| - 1, \quad \forall\, \mathbf{Q} \subset \{1, ..., N\} \tag{5}$$

$$\sum_{j \leq R} y_{jl} = 1, \quad \forall\, l \in \{1, ..., M\} \tag{6}$$

$$\sum_{l \leq M} y_{jl} \leq 1, \quad \forall\, j \in \{1, ..., R\} \tag{7}$$

$$x_{ik} \in \{0, 1\}, \quad \forall\, i, k \in \{1, ..., N\} \tag{8}$$

$$y_{jl} \in \{0, 1\}, \quad \forall\, j \in \{1, ..., R\},\ l \in \{1, ..., M\} \tag{9}$$

$$h_{ik} \in \{0, 1\}, \quad \forall\, i, k \in \{1, ..., N\} \tag{10}$$

La restricción (3) requiere que haya exactamente un movimiento saliendo de la ubicación de cada componente, mientras que (4) lo requiere entrando a la ubicación de ellos.

La restricción (5) requiere que para cada subconjunto $\mathbf{Q} \subset \{1, ..., N\}$ de ubicaciones de componentes, la secuencia de movimientos no forme un ciclo.

La restricción (6) requiere que cada tipo de componente sea asignado a una sola ranura, mientras que (7) requiere que cada ranura contenga a lo más un tipo de componente.

Por las restricciones planteadas, la secuencia de colocación de componentes es una secuencia cíclica.

El problema P1-1 planteado se representa gráficamente en la Figura 6. En esta figura el componente i se coloca justo antes que el componente k y la ranura j contiene a los componentes del tipo $\tau(k)$. Si los componentes pueden ser sujetados con las mismas herramientas, como se muestra en la Figura 6(a), existe un tiempo asociado t_{ij} correspondiente al desplazamiento desde la ubicación del componente i hasta la ranura j y otro tiempo t_{kj} que corresponde al desplazamiento desde la ranura j hasta la ubicación del componente k.

En el otro caso, el de la Figura 6(b), los componentes i y k no pueden ser sujetados con las mismas herramientas, entonces hay un tiempo asociado t_{ij}^{CAH} correspondiente

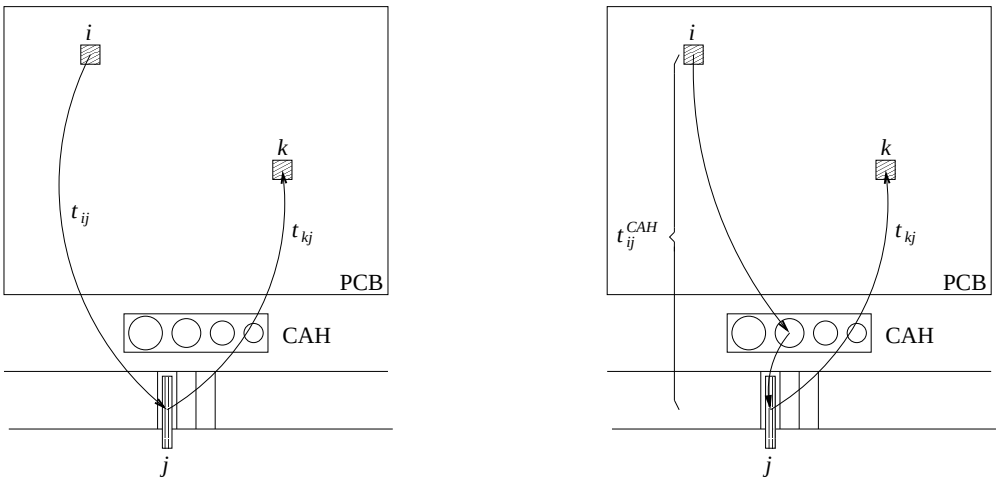


Figura 6. Representación gráfica del problema P1-1: (a) desplazamiento cuando los componentes i y k se pueden sujetar con las mismas herramientas; y (b) desplazamiento cuando no se pueden sujetar con las mismas herramientas.

al desplazamiento desde la ubicación del componente i a la ubicación del CAH y de éste a la ranura j y otro tiempo t_{kj} como el descrito en el caso anterior.

Si consideramos el caso en que todos los componentes se pueden sujetar con las mismas herramientas (i.e. $h_{ik} = 1, \forall\ i, k \leq N$), el término t_{ij}^{CAH} se elimina. Entonces tenemos que (2) queda como:

$$\underset{x_{ik},\, y_{jl} \in \{0,1\}}{\text{minimizar}}\ S' = \sum_{i \leq N} \sum_{j \leq R} \sum_{k \leq N} (t_{ij} + t_{kj}) x_{ik} y_{j\tau(k)} \tag{11}$$

sujeto a las restricciones (3)-(9).

El problema en (11) fue formulado por Kumar y Li (1995), en donde el problema fue separado en otros dos: un *problema de apareamiento de peso mínimo* (MWMP) y un *problema del viajante* (TSP). Se sabe que el TSP es **NP-difícil**, por lo que el problema (11) es al menos tan complejo como el TSP. Por lo tanto, el problema P1-1 es al menos tan difícil de resolver como (11), ya que la función que define a éste último es sólo un caso particular de (2).

II.4 El problema V-1

Este tipo de problemas se presenta cuando se van a manufacturar varios tipos de PCB utilizando una máquina. Como se describió en la Sección II.2, en esta clase de problemas se determina la estrategia de arreglo del anaquel alimentador para una sola máquina. Hay dos opciones para reducir el tiempo que toma esta actividad: reducir el tiempo al instalar un carrete nuevo, y reducir el número de carretes nuevos a instalar.

Dentro de la segunda opción existen varias alternativas para lograr el objetivo (Ammons *et al.*, 1997):

a) Estrategia de arreglo sencillo: se trata de configurar una máquina, o un grupo de ellas, para producir una familia de tipos de tarjetas, usando un arreglo sencillo. Hay dos estrategias:

   i. Estrategia de arreglo único. Se utiliza cuando la estrategia de arreglo sencillo se aplica a una familia que contiene un solo tipo de PCB. Se resuelve el problema 1-1 para este tipo de PCB.

   ii. Estrategia de arreglo por familia. Se aplica cuando hay varios tipos de PCB en una familia. Este arreglo elimina la necesidad de modificar el anaquel alimentador entre los tipos de tarjeta, puesto que en éste se encuentran todos los tipos de componentes necesarios para todos los tipos de PCB.

b) Estrategia de varios arreglos: debido a que la capacidad de los anaqueles alimentadores es limitada, en ocasiones no se puede considerar la estrategia de arreglo sencillo, sino que se requieren hacer arreglos adicionales para una misma familia. Existen dos posibles estrategias:

   i. Descomponer y secuenciar. Se forman conjuntos más pequeños de PCB en la familia, luego se define una secuencia de estos conjuntos con el fin de minimizar

los cambios en el anaquel alimentador cuando se termina de manufacturar un conjunto y se comienza con otro.

ii. Dividir y repetir. Se divide el conjunto de todos los tipos de componentes requeridos por todos los tipos de PCB, de tal forma que la o las máquinas tengan capacidad suficiente para cada subconjunto. Después se manufacturan las PCB parcial o totalmente, dependiendo de los tipos de componentes existentes en el anaquel alimentador, configurando las máquinas con cada subconjunto de tipos de componentes.

De estas estrategias, cuando los tipos de PCB se manufacturan en lotes pequeños, las eficientes son la de descomponer y secuenciar (Ammons *et al.*, 1997; Leon y Peters, 1998; Smed *et al.*, 1998; Salonen *et al.*, 2003; Narayanaswami e Iyengar, 2004) y dividir y repetir (Balakrishnan y Vanderbeck, 1999; Crama *et al.*, 2002).

La estrategia descomponer y secuenciar engloba a dos problemas: el agrupamiento de los diferentes tipos de PCB y la secuencia de manufactura de dichos grupos.

Al manufacturar un tipo de PCB, el anaquel alimentador debe contener los tipos de componentes necesarios para esta tarea. Por lo tanto, al manufacturar un grupo de PCB, los tipos de componentes necesarios para la manufactura de todos los tipos de PCB dentro del grupo deben estar contenidos en el anaquel alimentador.

Al dejar de producir las PCB de un grupo, se debe hacer un cambio en el anaquel alimentador, procurando cambiar el menor número de carretes. Así mismo, el número de grupos formados debe ser minimizado, todo esto con el fin de tener un menor tiempo de manufactura de toda la producción (Smed *et al.*, 1998).

II.4.1 Trabajo previo

Para el problema V-1 se han propuesto soluciones basadas esencialmente en heurísticas, debido a la dificultad del mismo. A continuación se mencionan algunos de estos trabajos.

Ammons *et al.*, 1997. Los autores plantean un modelo de *programación lineal entera* para el problema de manufactura de varios tipos de PCB y varias máquinas, en específico, se considera que ya existen los grupos formados y se intenta asignar los tipos de componentes a las diferentes máquinas de manufactura. Se presentan dos alternativas de solución: una heurística basada en *procesamiento de listas*, que intenta minimizar la diferencia entre la máxima y mínima de carga de trabajo de las máquinas, y una método de *ramificación y acotamiento* basado en *programación lineal*. Los resultados muestran una mejora en promedio, de acuerdo a la carga de la máquina con mayor carga de trabajo, del 11.85% respecto a la solución dada por un experto, esto con la primera heurística, y del 9% para la técnica de *ramificación y acotamiento*.

Crama *et al.*, 1997. Se propone una solución basada en una descomposición jerárquica del problema V-V, esto es, se divide al problema en cinco subproblemas independientes. Se considera que ya existen familias de PCB y la secuencia de manufactura de los tipos de PCB para cada familia, por lo que se contempla la estrategia de arreglo por familia. Para resolver el problema se utilizan heurísticas y métodos de *búsqueda local*.

Leon y Peters, 1998. Se estudian seis estrategias de arreglo del anaquel alimentador para la solución del problema de manufactura de varios tipos de PCB utilizando una máquina. En particular, el estudio se enfoca hacia las decisiones asociadas con el cambio de carretes y las secuencias de manufactura de tipos de PCB y de colocación

de componentes. Se comparan los resultados obtenidos entre los diferentes métodos, resultando en una diferencia, en promedio, de 14.42% entre el mejor y el peor resultado.

Smed *et al.*, 1998. En este trabajo se proponen algunas heurísticas para formar grupos de PCB y secuenciarlos para su manufactura, basadas en la estrategia descomponer y secuenciar, y se prueban sobre casos aleatorios. Se presentan los resultados de estas heurísticas y la que obtiene mejores resultados se compara, aplicada a un caso real, con el sistema de una empresa manufacturera, obteniendo una mejora del 25.7%.

Balakrishnan y Vanderbeck, 1999. En su trabajo, los autores plantean un modelo matemático para el problema de manufactura de varios tipos de PCB con varias máquinas y proponen dos métodos denominados *generación de columnas*, uno para obtener resultados de forma heurística y otro para la obtención de cotas inferiores.

Salonen *et al.*, 2000. Se proponen varias heurísticas para el agrupamiento de PCB y el secuenciamiento de estos grupos, basadas en las estrategias de arreglo por familia y descomponer y secuenciar. Estas heurísticas se aplican a tres casos documentados. Se propone una función de costo de la solución, la cual toma en cuenta el número de grupos formados y la cantidad de cambios que se realizan en el anaquel alimentador.

Crama *et al.*, 2002. Este es un trabajo puramente teórico, en donde se hace una revisión de todos los problemas que intervienen en la manufactura de varios tipos de PCB y varias máquinas y se presenta un modelo matemático para cada uno de ellos.

Narayanaswami e Iyengar, 2004. Se hace la descripción de dos heurísticas, una para el agrupamiento de PCB, basada en la similitud de cada par de éstas, y otra para el secuenciamiento, basada en la similitud de los grupos formados. Se compara

esta heurística con otras publicadas anteriormente, usando casos de prueba aleatorios, obteniendo mejores resultados en todos ellos.

Al igual que en los trabajos que tratan sobre el problema 1-1, en éstos se utilizan casos de prueba que no están disponibles públicamente, por ende no es posible hacer una comparación de los métodos propuestos.

En la mayoría de los estudios aquí mencionados, el problema V-1 se separa completamente de los otros problemas involucrados en la manufactura de un tipo de PCB en una máquina (PAR y PSC), debido al incremento en la dificultad (Leon y Peters, 1998).

Por otro lado, todos los trabajos realizados hasta el momento no han considerado al cómputo evolutivo como una opción para la solución del problema V-1, sino en métodos basados en heurísticas que comúnmente se aplican para solucionar problemas conocidos.

II.4.2 Heurística propuesta por Narayanaswami e Iyengar (2004) para resolver el problema V-1

Esta heurística se basa principalmente en el coeficiente de similitud de Jaccard (Salonen *et al.*, 2000; Smed *et al.*, 2000; Salonen *et al.*, 2003; Narayanaswami e Iyengar, 2004), el cual, dados dos conjuntos $\mathbf{K}_1$ y $\mathbf{K}_2$, da la similitud J_{12} de ambos conjuntos, es decir, qué tanto se parecen los conjuntos considerando los elementos que contienen. Si $J_{12} = 1$, entonces los conjuntos son iguales. Este coeficiente se define de la siguiente manera:

$$J_{12} = \frac{|\mathbf{K}_1 \cap \mathbf{K}_2|}{|\mathbf{K}_1 \cup \mathbf{K}_2|} \tag{12}$$

En el caso del problema V-1, se considera a un conjunto $\mathbf{K}$ como un tipo de PCB, en donde los elementos del conjunto son los tipos de componentes. De esta forma, el coeficiente de similitud de Jaccard entre los tipos de PCB PCB_1 y PCB_2 para el

problema V-1 tiene la siguiente forma:

$$J_{12} = \frac{|\text{PCB}_1 \cap \text{PCB}_2|}{|\text{PCB}_1 \cup \text{PCB}_2|} \tag{13}$$

En (13), el numerador es la cantidad de tipos de componentes que tienen en común los tipos de PCB PCB_1 y PCB_2, y el denominador es la cantidad de tipos de componentes que se utilizan para manufacturar a ambos.

La heurística propuesta por Narayanaswami e Iyengar (2004), para agrupar los tipos de PCB, sigue los siguientes pasos. Se determina el primer tipo de PCB para formar el primer grupo, que es el que tenga la similitud global máxima, de acuerdo con el coeficiente de Jaccard. Se asigna este tipo de PCB al grupo y los tipos de componentes que se utilizan en este tipo de PCB se consideran para ser cargados en el anaquel alimentador. Se calcula la similitud entre cada uno del resto de los tipos de PCB y el anaquel alimentador. Se selecciona el tipo de PCB con mayor similitud y si es suficiente la capacidad restante en el anaquel alimentador, se agrega este tipo de PCB al mismo grupo y se vuelve a calcular la similitud entre el resto de los tipos de PCB y el anaquel alimentador. Si no es suficiente la capacidad restante, se agrega a un nuevo grupo. Se continua de esta forma hasta agregar el resto de los tipos de PCB a algún grupo.

Una vez formados los grupos, se procede a secuenciarlos para su manufactura de la siguiente manera. Se calcula la similitud de Jaccard entre los grupos de PCB. Para cualquier grupo i, el grupo j con el que tenga la mayor similitud forma el primer grupo en la secuencia de manufactura que inicia con el grupo i. Para cada grupo i, la secuencia relacionada con éste debe contener el resto de los grupos. De esta forma se genera una secuencia de manufactura para cada grupo. Si en alguna secuencia se forma un ciclo, éste se rompe seleccionando un grupo con otra prioridad. La secuencia que contemple el menor número de cambios es la que se elige.

II.4.3 Modelo matemático del problema V-1

El modelo matemático del problema V-1 se da en dos partes: la construcción de grupos y la secuencia de los mismos para su manufactura. En las siguientes secciones se modelan por separado estos subproblemas.

II.4.3.1 Formación de grupos

Dados N tipos de PCB $(1, ..., N)$, M tipos de componentes $(1, ..., M)$ utilizados por los N tipos de PCB, y los respectivos M_i tipos de componentes que la PCB tipo i contiene $(\forall\ i \in \{1, ..., N\}, M_i \leq M)$, agrupar los tipos de PCB, de tal forma que el número de grupos resultantes $|\mathbf{G}|$ sea mínimo, sin exceder el número de tipos de componentes R que cada máquina puede alojar (número de ranuras), considerando que cada tipo de componente se aloja en una ranura.

Cabe notar que M en este problema representa a los tipos de componentes que se utilizan en todos los tipos de PCB, a diferencia del problema 1-1, en donde representa a los tipos de componentes que se usan en una sola PCB.

De los datos de entrada podemos definir $a_{ki} = 1$ si el tipo de componente k se utiliza en la PCB tipo i y 0 en otro caso.

Sea x_{ij} una variable de decisión igual a 1 si la PCB tipo i se asigna al grupo j y 0 en otro caso.

Sea y_j una variable de decisión igual a 1 si el grupo j se utiliza y 0 en otro caso.

Sea la variable

$$z_{kj} = \left\lceil \frac{1}{M} \sum_{i=1}^{N} x_{ij} a_{ki} \right\rceil$$

es decir, $z_{kj} = 1$ si el componente tipo k se utiliza en al menos una tarjeta del grupo j y 0 en otro caso.

Podemos ahora modelar el problema de formación de grupos PV-1G de la siguiente forma:

$$\underset{x_{ij},y_j,z_{kj}\in\{0,1\}}{\text{minimizar}} |\mathbf{G}| = \sum_{j=1}^{N} y_j \tag{14}$$

sujeto a

$$\sum_{k=1}^{M} z_{kj} \leq R, \qquad \forall\, j \in \{1, ..., N\} \tag{15}$$

$$\sum_{j=1}^{N} x_{ij} = 1, \qquad \forall\, i \in \{1, ..., N\} \tag{16}$$

$$x_{ij} \in \{0, 1\} \tag{17}$$

$$y_j \in \{0, 1\} \tag{18}$$

$$z_{kj} \in \{0, 1\} \tag{19}$$

donde $\mathbf{G}$ es el conjunto de los grupos de PCB creados.

La restricción (15) indica que el número de tipos de componentes utilizados en cada grupo no debe exceder el número de ranuras de la máquina. La restricción (16) obliga a asignar cada PCB a un solo grupo.

De este modelo podemos observar que el problema de la formación de grupos puede ser visto como un problema de *bin packing*, el cual pertenece a la clase de problemas **NP-difícil**.

II.4.3.2 Secuencia de manufactura de grupos

Dado el conjunto $\mathbf{G}$ de los grupos de PCB creados y los M tipos de componentes que se utilizan en todas las PCB a manufacturar, secuenciar los grupos en $\mathbf{G}$ para su manufactura, de forma que el número de cambios C en las ranuras del anaquel alimentador, al terminar de manufacturar un grupo de PCB y comenzar con otro, sea mínimo.

Un cambio se refiere a cuando se tiene que retirar un carrete que contiene cierto tipo de componente que está siendo utilizado para manufacturar el grupo de PCB actual y

el siguiente grupo de PCB no lo utiliza, y viceversa, cuando el grupo de PCB actual no lo está usando y se tiene que colocar para el siguiente grupo.

Cabe aclarar que, mientras se está manufacturando un grupo de PCB, no se hacen cambios en el anaquel alimentador, sino que los cambios sólamente se hacen, como se mencionó en el párrafo anterior, cuando un grupo de PCB se termina de manufacturar y la manufactura de otro comienza.

Sea la variable de decisión $w_{jl} = 1$ si el grupo l se manufactura inmediatamente después que el grupo j y 0 si no.

El problema de secuencia de grupos de PCB PV-1C se puede modelar como:

$$\underset{w_{jl} \in \{0,1\}}{\text{minimizar}} \; C = \sum_{j \in \mathbf{G}} \sum_{l \in \mathbf{G}} \sum_{k=1}^{M} (z_{kj} - z_{kl})^2 w_{jl} \tag{20}$$

sujeto a

$$\sum_{l \in \mathbf{G}} w_{jl} = 1, \qquad \forall \; j \in \mathbf{G} \tag{21}$$

$$\sum_{j \in \mathbf{G}} w_{jl} = 1, \qquad \forall \; l \in \mathbf{G} \tag{22}$$

$$\sum_{j \in \mathbf{G}} \sum_{j \in \mathbf{G}} w_{jl} \leq |\mathbf{Q}| - 1, \qquad \forall \; \mathbf{Q} \subset \mathbf{G} \tag{23}$$

$$w_{jl} \in \{0, 1\} \tag{24}$$

El término $\sum_{k=1}^{M} (z_{kj} - z_{kl})^2 w_{jl}$ en (20), indica el número cambios que se hacen en el anaquel alimentador al terminar de manufacturar el grupo j y comenzar con el grupo l. Por ejemplo, si en el grupo j se utiliza el tipo de componente k y en el grupo l no, o viceversa, si en el grupo j no se utiliza y en el l si, el término entre paréntesis es 1. Por otro lado, si en los dos grupos se utiliza el tipo de componente k o si no se utiliza en ninguno de los dos, el término es cero.

La restricción (21) indica que sólo un grupo se manufactura inmediatamente después

de cualquier otro, mientras que (22) requiere que sólo un grupo se manufacture inmediatamente antes.

La restricción (23) requiere que para cada subconjunto $\mathbf{Q} \subset \mathbf{G}$ de grupos de PCB, la secuencia de manufactura no forme un ciclo.

Por las restricciones planteadas, la secuencia de manufactura de grupos es una secuencia cíclica.

Es claro, por el modelo presentado, que éste, el problema de secuenciamiento de grupos de PCB, es un *problema del viajante*, el cual pertenece a la clase **NP-difícil**.

II.5 Resumen

En este capítulo se hizo una breve descripción del funcionamiento de una máquina de manufactura de PCB, así como de la clasificación de los problemas de manufactura de PCB. Se formularon matemáticamente y se revisó el trabajo previo de dos de los problemas dentro de la clasificación mencionada, que son los de manufactura de uno y de varios tipos de PCB utilizando una máquina para este propósito.

Del modelo matemático de estos problemas ha quedado fundamentado que ambos problemas son difíciles de resolver (pertenecen a la clase **NP-difícil**), por lo que en todos los trabajos citados se emplean diversas heurísticas para resolverlos.

Los métodos más comunes para la solución de ambos problemas han sido heurísticas diseñadas para problemas conocidos, por ejemplo para los problemas *de programación lineal entera*, *del viajante*, *de apareamiento de peso mínimo*, o bien, heurísticas que han sido propuestas por los autores de los diferentes trabajos de investigación.

La minoría de los trabajos desarrollados resuelven los problemas mediante un enfoque basado en el cómputo evolutivo, i.e. algoritmos genéticos. Los resultados reportados por éstos muestran su superioridad sobre otros, aunque no se puede hacer una comparación de estos algoritmos debido a que no se utilizan los mismos casos de prueba,

excepto en los trabajos de Leu *et al.* (1993), Ong y Khoo (1999), y Ho y Ji (2005).

Capítulo III

Algoritmos genéticos para los problemas de ensamble de PCB

"En todos los ámbitos es sano colocar de vez en cuando un signo de interrogación a las cosas que se han dado por sentado."
Bertrand Russell (1872 – 1970)

Los algoritmos genéticos, al igual que las heurísticas en general, no garantizan soluciones óptimas en tiempo finito, ni desempeño alguno. Sin embargo, debido a los buenos resultados obtenidos en la práctica, el uso de éstos se ha hecho muy común en muchas aplicaciones hoy en día.

III.1 Algoritmos genéticos

Los algoritmos genéticos (AG) son algoritmos de búsqueda que se basan en la teoría de la evolución *Neo-Darwiniana*. Estos combinan la ley de supervivencia del más fuerte dentro de una estructura de datos, con el intercambio aleatorio de información. En cada generación se crea un nuevo conjunto de individuos que usan la información de sus antecesores y, en ocasiones también, se prueba una parte innovadora (Goldberg, 1989).

Los AG reunen dos características importantes: la explotación eficiente de la información histórica de las soluciones y la exploración de nuevos puntos de búsqueda dentro del espacio de soluciones. Un balance adecuado de ambas características asegura un buen desempeño de los AG.

En la terminología de los AG se utilizan varios sinónimos para referenciar a los componentes que se utilizan en la representación o codificación de un problema. Una

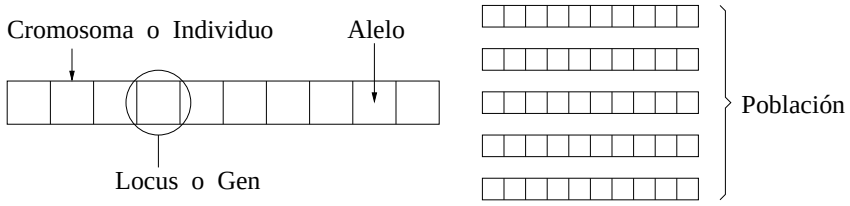


Figura 7. Representación esquemática de la terminología utilizada en los algoritmos genéticos.

solución para algún problema es conocida como *cromosoma* o *individuo*, el cual es un arreglo de caracteres. A un conjunto de individuos se le denomina *población*. Dentro de un individuo, en cada posición del arreglo existe un *gen* (o *gene*) o *locus* (*loci* en plural). A la información contenida en los genes se le llama *alelo* (Eiben y Smith, 2003). La representación esquemática de esta terminología se muestra en la Figura 7.

El algoritmo comienza con un multiconjunto de soluciones iniciales, llamado *población inicial*. Los individuos de una población se usan para crear una nueva población (siguiente generación). Cada individuo tiene asignado una *aptitud*, la cual está directamente relacionada con la calidad de la solución que éste propone, es decir, aquel individuo que represente a una mejor solución, tendrá una mejor aptitud.

La forma en que se utiliza una población para crear una nueva generación, está basada en los operadores genéticos: *selección*, *cruzamiento* y *mutación* (Coello, 2001). Estos operadores funcionan de manera probabilística.

III.1.1 Selección

Una parte fundamental del algoritmo genético es el proceso de selección, el cual consiste en seleccionar de entre los individuos de la generación actual, aquellos que serán tomados en cuenta para la siguiente.

Existen varias técnicas para esta operación, en las que los individuos con mayor aptitud tienen mayor probabilidad de ser seleccionados. En algunas técnicas, aun cuando

un individuo tenga la peor aptitud, gracias a la naturaleza probabilística del operador, éste puede sobrevivir para la siguiente generación. La probabilidad p_{s_i} de que un individuo i sea seleccionado está directamente relacionada con la aptitud f_i del mismo.

Este operador puede o no incluir *elitismo*, que consiste en buscar en cada generación al mejor individuo de la población y duplicarlo para que exista una referencia de calidad de solución entre las generaciones, sin que la información contenida en los genes de éste sea modificada.

III.1.2 Cruzamiento

El cruzamiento es un proceso que ocurre en los sistemas biológicos, en donde dos cromosomas se alinean para fragmentarse e intercambiar estos fragmentos entre si. En los algoritmos genéticos se trata de simular este proceso. Es decir, se eligen dos individuos de la población y de acuerdo a la probabilidad de cruzamiento p_c se decide si se cruzan o no. En caso afirmativo se seleccionan uno o varios segmentos de genes en ellos para ser intercambiados entre si. El efecto de dos diferentes técnicas de este operador se puede ver esquemáticamente en la Figura 8.

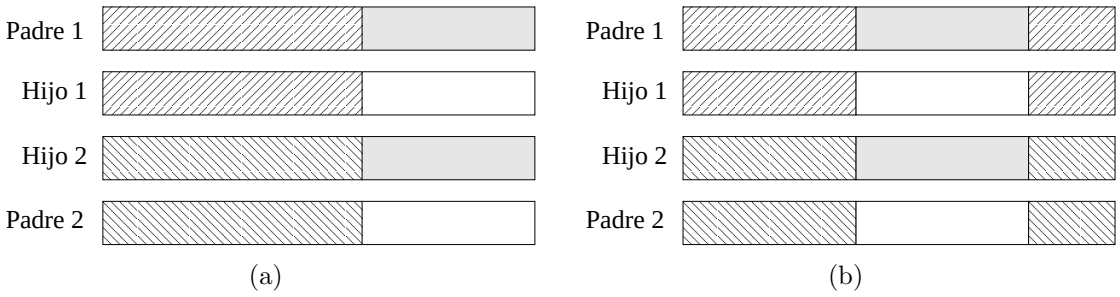


Figura 8. Operador de cruzamiento aplicado a un par de cromosomas, seleccionando: (a) un punto de cruza; y (b) dos puntos de cruza.

El resultado de esta operación es regularmente la creación de dos nuevos individuos, aunque en algunos casos sólo se elige a uno de ellos. Con este operador se pretende hacer un salto en el espacio de soluciones para explorar otras áreas.

III.1.3 Mutación

La mutación, como su nombre lo indica, consiste en alterar o modificar a un individuo ya existente dentro de la población. Esto se lleva a cabo cuando se selecciona, utilizando generalmente una probabilidad de mutación p_m muy baja, a un cromosoma dentro de la población para ser mutado y cambiando la información contenida en sus genes (alelos). La mutación de un cromosoma de ejemplo se muestra en la Figura 9.

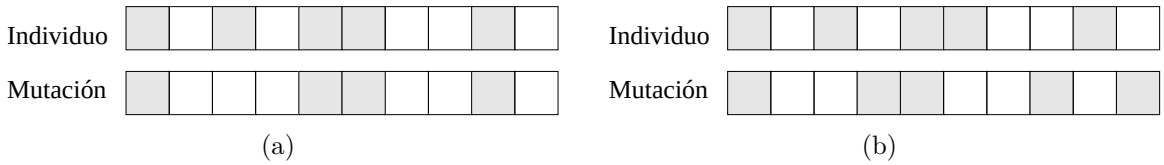


Figura 9. Operador de mutación aplicado a un cromosoma mediante dos técnicas: (a) cambio en un gen; y (b) recorriendo un segmento de genes.

El resultado de este proceso es un nuevo individuo. La mutación previene que las soluciones queden estancadas en un óptimo local e introduce diversidad genética en la población.

III.1.4 Algoritmo genético canónico

El AG canónico consiste de una n-tupla de cadenas binarias b_i de longitud l, donde los bits de cada cadena se consideran los genes del cromosoma y donde la n-tupla de cromosomas son la población. Cada individuo b_i representa una solución factible de algún problema (Holland, 1975) y su valor de la función objetivo $f(b_i)$ es su aptitud, la cual trata de ser maximizada (Günter, 1994). El Algoritmo 1 describe al algoritmo genético canónico.

III.2 Algoritmo genético para el problema 1-1

Como se vio en la Sección II.3, en este problema se presentan tres subproblemas: la asignación de las ranuras del anaquel alimentador a los tipos de componentes (PAR),

Algoritmo 1 Algoritmo genético canónico

```
1: elegir una población inicial
2: determinar la aptitud de cada individuo
3: seleccionar
4: repetir
5:     aplicar cruzamiento
6:     aplicar mutación
7:     determinar la aptitud de cada individuo
8:     seleccionar
9: hasta algún criterio de paro
```

la secuencia de colocación de componentes en la PCB (PSC), y el plan de recuperación (PR). Éste último, por las consideraciones ya planteadas, está resuelto.

Estos problemas se han estudiado de manera extensa mediante diversas técnicas, las cuales van desde heurísticas muy sencillas, pasando por templado simulado y algoritmos genéticos, hasta métodos de programación lineal y programación dinámica.

III.2.1 Representación genética del problema 1-1

La representación genética del problema consiste en un cromosoma con dos partes: una para la asignación de ranuras y otra para la secuencia de colocación, como se muestra en la Figura 10(a).

La cantidad de genes que corresponden al problema de asignación de ranuras es el número de ranuras disponibles en la máquina. En la Figura 10(b) se puede observar que el locus indica el tipo de componente, mientras que el alelo indica el número de la ranura que le corresponde al tipo de componente.

En el caso del problema de la secuencia de colocación, como se muestra en la Figura 10(c), el locus está relacionado con la secuencia de colocación, mientras que el alelo indica el número que identifica al componente.

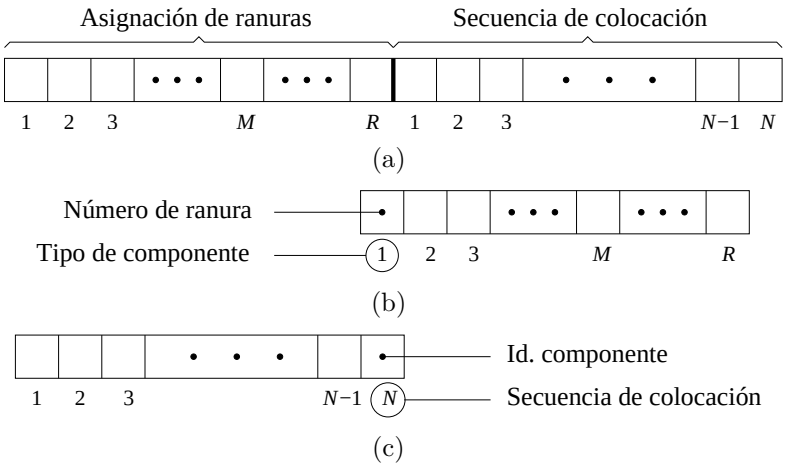


Figura 10. Representación genética del problema 1-1: (a) el cromosoma para la codificación del problema; (b) los genes correspondientes al PAR; y (c) los genes correspondientes al PSC.

III.2.2 Función de aptitud

Como se requiere que la manufactura de la PCB sea lo más rápida posible, se toma al tiempo que se tarda en manufacturar completamente la PCB como indicador de la calidad de las soluciones. Antes de presentar la función de aptitud, es conveniente mostrar la notación que se utiliza en la misma.

$\mathbf{H}$ = conjunto de las herramientas a utilizar,

$h(i)$ = herramienta que sujeta al tipo de componente i,

$h : \{1, ..., M\} \longrightarrow \{1, ..., |\mathbf{H}|\}$,

$\sigma(i)$ = ranura a la cual está asignado el tipo de componente i,

$\sigma : \{1, ..., M\} \longrightarrow \{1, ..., R\}$,

$s(i)$ = identificador del componente a ser montado en el i-ésimo orden,

$u(i)$ = posición del componente i en la PCB,

$T(i, j)$ = tiempo de traslado entre las ubicaciones i y j,

T_{CAH} = tiempo para cambiar las herramientas de sujeción,

u_{CAH} = posición del cambiador automático de herramientas.

Definimos entonces la función de aptitud de la forma siguiente:

$$f = \sum_{k=1}^{N} \Big[T(\sigma(\tau(s(k))), u(s(k))) + g(k) \Big], \tag{25}$$

donde:

$$g(k) = \begin{cases} 0 & \text{si } k = N, \\ T(u(s(k)), \sigma(\tau(s(k+1)))) & \text{si } \pi, \\ T(u(s(k)), u_{CAH}) + T_{CAH} + T(u_{CAH}, \sigma(\tau(s(k+1)))) & \text{en otro caso.} \end{cases} \tag{26}$$

donde la condición π se refiere a $h(\tau(s(k))) = h(\tau(s(k+1)))$, es decir, si la misma herramienta se utiliza para sujetar a los componentes secuenciados en la posición k y $k+1$.

El primer sumando en (25) representa el tiempo que tarda el robot en trasladarse desde la ranura en donde se encuentra el tipo del componente en el k-ésimo orden, a su posición de montaje dentro de la PCB, mientras que el segundo sumando puede representar tres opciones: si el componente que se está montando es el último, ya no se toma en cuenta el regreso; si el siguiente componente puede ser sujetado por la herramienta actual, se toma en cuenta el tiempo de traslado desde la misma posición en la PCB a la ranura donde se encuentra el $(k+1)$-ésimo componente a ser montado; y por último, si el siguiente componente no puede ser sujetado por la herramienta actual, se suma el tiempo de traslado desde la ubicación en la PCB del k-ésimo componente en la secuencia hasta el cambiador automático de herramientas y de ésta posición a la ranura en donde se encuentra el siguiente componente a ser montado.

Es claro que la función de aptitud es $O(N)$, debido a que se tiene que calcular la suma de N términos.

III.2.3 Población inicial

Los individuos pertenecientes a la población inicial se generan de la siguiente manera. Para la parte de la asignación de ranuras, aleatoriamente se forma una permutación con los números $1, 2, ..., R$. Para la parte de la secuencia de colocación, primero se agrupan los componentes que pueden ser sujetados con la misma herramienta, y ya que se tienen estos grupos, dentro de cada grupo se forma una secuencia aleatoria con los identificadores de los componentes. La forma de generar la secuencia de colocación en los individuos iniciales se hace con el fin de tener el menor número de cambios de herramienta, pues se supone que más cambios empeorarían el resultado.

La Figura 11 muestra dos ejemplos de individuos que podrían formar parte de una población inicial. Se considera un caso con diez componentes, los cuales son de cuatro tipos diferentes. La máquina cuenta con cinco ranuras para asignar a los tipos de componentes. Se puede observar que la parte correspondiente a la asignación de ranuras, en ambos individuos, contiene una permutación de los números 1, 2, 3, 4 y 5, mientras que la parte correspondiente a la secuencia de colocación contiene dos secuencias: una con los números 1, 2, 3 y 7, que son los identificadores de los componentes que pueden ser sujetados con la herramienta número 12, y otra con los números 4, 5, 6, 8, 9 y 10, que son los componentes que pueden ser sujetados con la herramienta identificada con el número 38.

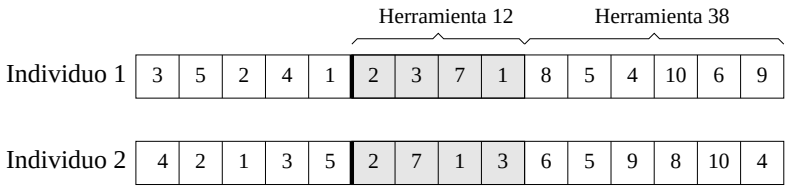


Figura 11. Dos individuos que forman parte de una población inicial para un caso con diez componentes de cuatro tipos diferentes y cinco ranuras en la máquina de manufactura.

III.2.4 Selección

En el AG para solucionar el problema 1-1, se probaron dos técnicas de selección, las cuales se detallan a continuación.

III.2.4.1 Muestreo determinístico

Esta técnica fue propuesta por De Jong (1975). En ésta, la probabilidad de que un individuo i sea seleccionado se calcula de acuerdo a la función:

$$p_{s_i} = \frac{f_i}{\sum_{k \leq N} f_k} \tag{27}$$

Después, el número esperado e_i de copias de cada individuo para la siguiente generación se calcula con la función:

$$e_i = p_{s_i} \times N \tag{28}$$

Cada individuo se copia, en la siguiente generación, un número de veces igual a la parte entera de su valor e_i. Se ordenan los individuos, de forma descendente, de acuerdo al valor fraccionario de su valor e_i y se toman los primeros para completar la población (Goldberg, 1989). Este procedimiento se muestra en el Algoritmo 2.

Algoritmo 2 Selección por muestreo determinístico

1: calcular el número esperado de copias e_i de cada individuo
2: copiar la parte entera de e_i veces cada individuo
3: ordenar los individuos de forma descendente de acuerdo a la parte fraccionaria del número esperado de copias e_i
4: copiar los primeros individuos, en el orden que aparecen, para completar la población

Si consideramos que en la población hay n individuos, esta técnica de selección, por utilizar un ordenamiento, es $O(n \log n)$, a pesar de que los valores e_i se calculan en $O(n)$.

III.2.4.2 Sobrante estocástico con reemplazo

Esta técnica de selección fue propuesta por Brindle (1981). Comienza de forma idéntica que el muestreo determinístico. Se calculan las copias esperadas e_i de cada individuo y se copian un número de veces igual a la parte entera de éste.

Para completar la población, se utiliza algún método de selección proporcional, como el de *la ruleta* (que se explica más adelante), tomando en cuenta la parte fraccionaria de los valores e_i de cada individuo.

El Algoritmo 3 se refiere a esta técnica.

Algoritmo 3 Selección por sobrante estocástico con reemplazo

1: calcular el número esperado de copias e_i de cada individuo
2: copiar la parte entera de e_i veces cada individuo
3: aplicar algún método de selección proporcinal, tomando en cuenta la parte fraccionaria del número esperado de copias e_i, para completar la población

La complejidad de esta técnica depende del método de selección proporcional que se utilice. Hasta antes de utilizar este método, el algoritmo es $O(n)$, donde n es la cantidad de individuos en la población.

III.2.5 Cruzamiento

Existen varios operadores de cruzamiento, pero sólo se eligieron dos de ellos para ser incluidos en el algoritmo genético. Ambos operadores se detallan a continuación. El primero de ellos no ha sido considerado en trabajos previos, mientras que el segundo si ha sido utilizado.

Estos operadores generan dos nuevos individuos, por lo cual se pueden considerar ambos o sólo uno de ellos para la siguiente generación. Esta última opción trae consigo que para cada nuevo individuo se tiene que calcular la aptitud del mismo.

III.2.5.1 Position-based crossover

Este operador, *position based crossover* (Syswerda, 1991), se detalla a continuación.

Se eligen aleatoriamente de la población dos cromosomas que tendrán el rol de padres y se verifica que se cumpla la probabilidad de cruzamiento p_c. En seguida, se elige aleatoriamente un número de genes en los padres, cuyos alelos se copiarán a los hijos en el mismo gen, del primer padre al primer hijo y del segundo padre al segundo hijo. Finalmente, los alelos del segundo padre que no existan en el primer hijo, se copian en el orden que aparecen en el segundo padre. Se hace lo mismo con el primer padre y el segundo hijo.

Este operador se aplica tanto para la parte relacionada con la asignación de tipos de componentes, como para la relacionada con la secuencia de colocación, como se muestra en la Figura 12.

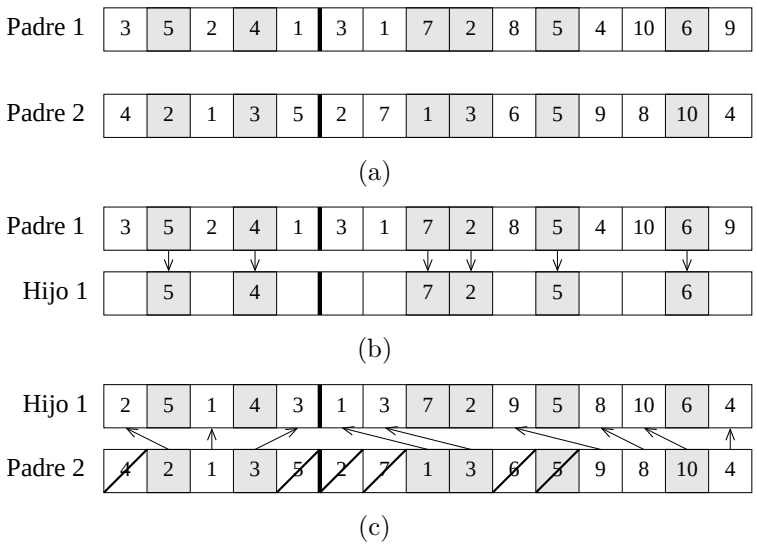


Figura 12. Operador *position-based crossover*: (a) los genes sombreados son los seleccionados aleatoriamente para copiarse de los padres a los hijos; (b) copia de los alelos en los genes seleccionados del primer padre al primer hijo; y (c) copia del resto de los alelo del segundo padre al primer hijo.

Debido a que el algoritmo para este operador debe hacer N y R búsquedas de un

elemento en arreglos no ordenados de tamaño N y R, respectivamente, tiene un tiempo de ejecución $O(R^2 + N^2)$.

III.2.5.2 Order crossover

Esta técnica, *order crossover*, fue propuesta por Davis (1985), la cual consiste en lo siguiente.

Se eligen aleatoriamente de la población dos cromosomas que tendrán el rol de padres y se verifica que se cumpla la probabilidad de cruzamiento p_c. En seguida, se elige aleatoriamente un segmento dentro de los padres (los mismos genes consecutivos en ambos padres) y se copia a los hijos: del primer padre al primer hijo y del segundo padre al segundo hijo. Finalmente, los alelos del segundo padre que no existan en el primer hijo, se copian en el orden que aparecen en el segundo padre. Se hace lo mismo con el primer padre y el segundo hijo.

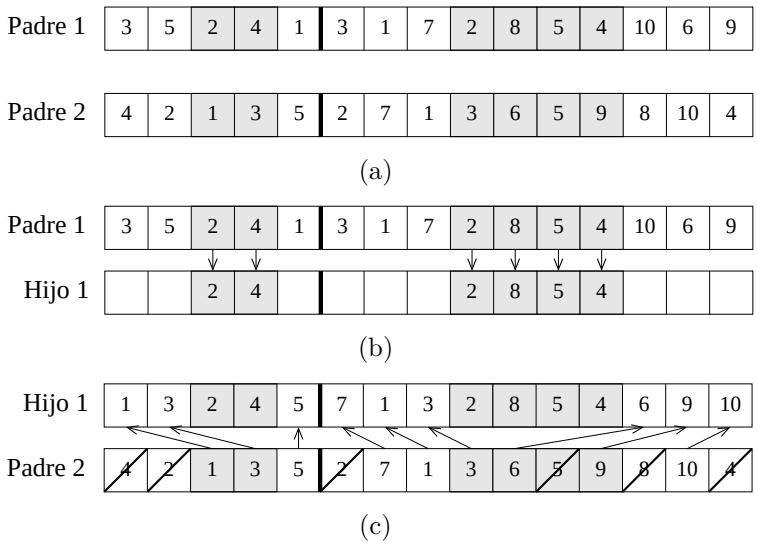


Figura 13. Operador *order crossover*: (a) los genes sombreados son los seleccionados aleatoriamente para copiarse de los padres a los hijos; (b) copia de los alelos en los genes seleccionados del primer padre al primer hijo; y (c) copia del resto de los alelo del segundo padre al primer hijo.

Al igual que el anterior, este operador se aplica tanto para la parte relacionada con

la asignación de tipos de componentes, como para la relacionada con la secuencia de colocación, como se muestra en la Figura 13.

Similar a *position-based crossover*, debido a que el algoritmo para este operador debe hacer N y R búsquedas de un elemento en arreglos no ordenados de tamaño N y R, respectivamente, tiene un tiempo de ejecución $O(R^2 + N^2)$.

III.2.6 Mutación

Se probaron dos técnicas de mutación para ser utilizadas en el algoritmo genético, las cuales se describen a continuación.

III.2.6.1 Mutación heurística

La técnica de *mutación heurística* (Gen y Cheng, 2000) se describe a continuación.

Se eligen k genes al azar del individuo a mutar y se crean $k! - 1$ individuos con las $k! - 1$ posibles permutaciones de los genes seleccionados. Enseguida se calcula la aptitud de estos nuevos individuos y se elige al mejor de los $k!$ individuos.

Al igual que en la cruza, este operador se aplica a ambas partes del cromosoma, sólo que primero se aplica a la parte correspondiente a la asignación, se genera un nuevo individuo y después se aplica a la parte correspondiente a la secuencia de colocación. Un ejemplo de este procedimiento con $k = 3$ se muestra esquemáticamente en la Figura 14. Primero se crean cinco mutaciones del individuo original con las cinco posibles permutaciones de los tres genes en la parte correspondiente al PAR, y se calcula la aptitud de ellos. Se elige la mejor mutación, en este caso la Mutación 3, y se generan las cinco mutaciones posibles con los tres genes de la parte correspondiente al PSC. Se vuelven a calcular las aptitudes de las mutaciones y se elige a la mejor de ellas para remplazar al individuo original. Cabe señalar que el individuo original también compite para formar parte de la población en la siguiente generación.

Como ya se mencionó, este operador genera $k! - 1$ nuevos individuos y para cada

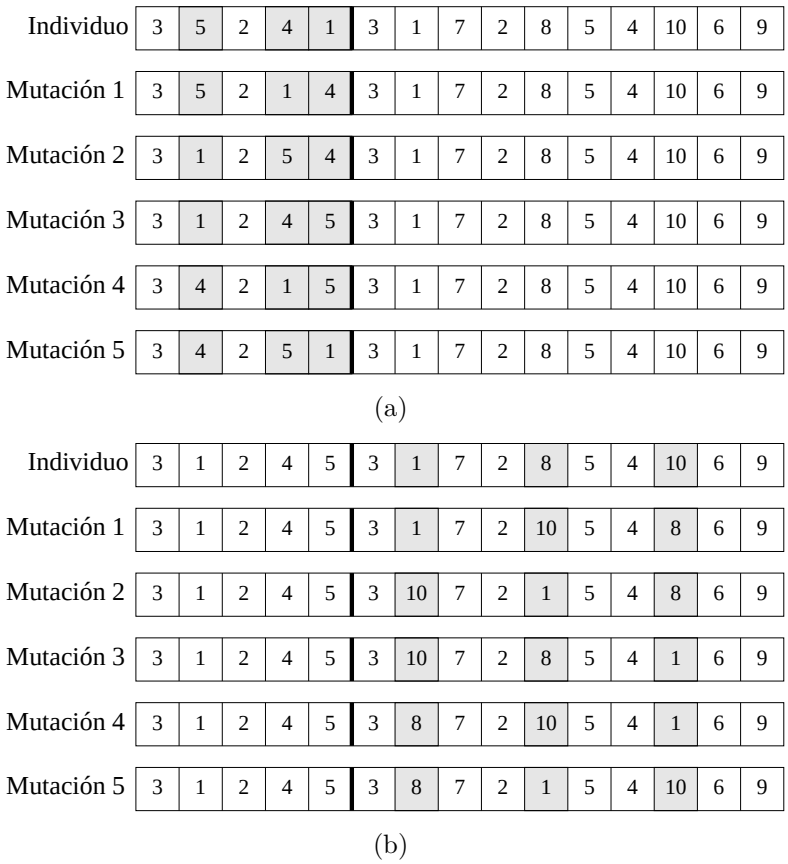


Figura 14. Mutación heurística con $k = 3$: selección aleatoria y permutaciones posibles de los genes en la parte correspondiente a: (a) la asignación de ranuras; y, suponiendo que la Mutación 3 es la que propone una mejor solución, (b) la secuencia de colocación.

uno de ellos se calcula su aptitud, entonces, el tiempo de ejecución de este operador es $O(k!N)$.

III.2.6.2 Mutación por inversión

En esta técnica se elige aleatoriamente un segmento de genes continuos en el individuo a mutar. Se copian de forma inversa los alelos dentro del segmento seleccionado al individuo mutado, en los mismos genes del segmento seleccionado en el individuo original. Cabe señalar que el segmento seleccionado al azar, contiene únicamente componentes que se pueden sujetar con una misma herramienta.

Esta técnica se usa, en conjunto con la mutación heurística, vista previamente,

para mutar la parte correspondiente al PSC y la segunda sólo para mutar la parte correspondiente al PAR.

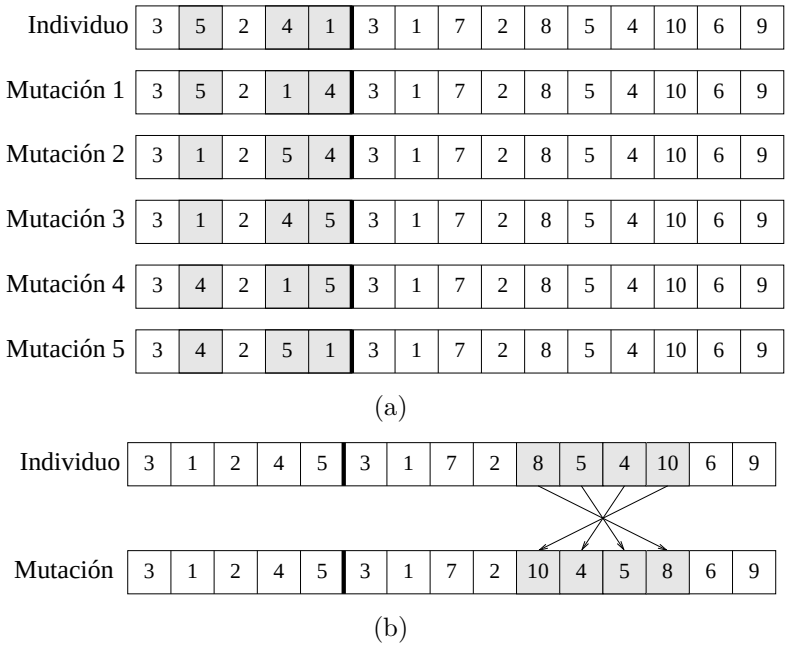


Figura 15. Mutación por inversión con $k = 3$: (a) selección aleatoria y permutaciones posibles de los genes en la parte correspondiente a la asignación de ranuras; y, suponiendo que la Mutación 3 es la que propone una mejor solución, (b) la inversión de un segmento seleccionado aleatoriamente en la parte de la secuencia de colocación.

La Figura 15 muestra un ejemplo de esta técnica. La mutación heurística actúa de forma idéntica a la descrita en la sección anterior. Ya que se elige la mejor mutación, en este ejemplo la Mutación 3, se selecciona aleatoriamente un segmento de este nuevo individuo y se copian de manera invertida a la mutación.

En el peor de los casos, en este operador se tendrían que invertir todo los genes de la parte del cromosoma corresponciente al PSC, lo que implica que este operador tiene una complejidad $O(N)$ exclusivamente para el operador de inversión.

III.3 Algoritmo genético para el problema V-1

En la Sección II.4 se describieron los dos subproblemas que surgen cuando se analiza el problema V-1, que básicamente consisten en el agrupamiento de los tipos de PCB a manufacturar y en establacer una secuencia de manufactura de estos grupos.

III.3.1 Representación genética del problema V-1

Falkenauer (1996), describe una representación basada en grupos, la cual nos ofrece la flexibilidad necesaria para la formación de grupos de objetos, en nuestro caso, grupos de tipos de PCB.

En esta representación, cada cromosoma consta de dos partes: la primera hace uso de la representación de un gen por cada objeto, es decir, en el caso del problema V-1, nos da la información de a qué grupo pertenece cada tipo de PCB, y la segunda parte contiene los grupos formados. Esta representación se puede ver esquematizada en la Figura 16(a).

En la parte correspondiente a la asignación de objetos a los grupos, el locus indica el tipo de PCB y el alelo el grupo al cual fue asignado éste, como se muestra en la Figura 16(b).

Por las características que tiene esta representación, también se puede utilizar ésta para definir la secuencia en la que los grupos de PCB van a ser manufacturados. De lo anterior, en la parte correspondiente a los grupos, el alelo representa al grupo formado y el locus representa la secuencia de manufactura del grupo.

Es importante destacar que, en problemas de agrupamiento, la información que nos interesa saber es cuáles objetos forman a cada grupo y no la etiqueta o identificador del grupo.

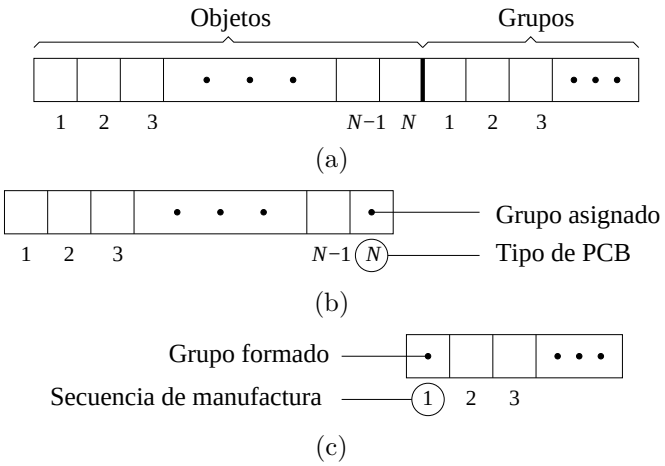


Figura 16. Representación genética del problema V-1: (a) el cromosoma para la codificación del problema; (b) los genes correspondientes a los tipos de PCB; y (c) los genes correspondientes a los grupos formados y su secuencia de manufactura.

III.3.2 Función de aptitud

Como se mencionó anteriormente, lo que interesa en la manufactura de varios tipos de PCB es la minimización del número de grupos formados y la cantidad de cambios que se realizan en el alimentador al terminar de producir un grupo de PCB y comenzar con otro. De esta forma la función de aptitud que nos da la calidad de la solución propuesta por cada individuo es la siguiente:

$$f = \frac{\alpha}{G} + \frac{1}{C} \tag{29}$$

donde

G = número de grupos formados

C = cantidad de cambios en el alimentador durante la manufactura de todos los tipos de PCB

El coeficiente α en (29) se introduce para igualar el orden de magnitud de ambos sumandos.

Como lo que se requiere es minimizar tanto el número de grupos, como la cantidad

de cambios, en el algoritmo utilizado lo que se trata es de maximizar la función de aptitud. Queda claro de (29) que el problema es de naturaleza multi-objetivo.

III.3.3 Población inicial

La idea detrás de la población inicial es que, originalmente, cada tipo de PCB forma un grupo unitario. De esta forma, en un principio se consideran N grupos. Para formar un individuo de la población inicial, cada tipo de PCB se asigna aleatoriamente a uno de estos N grupos.

La Figura 17 muestra dos ejemplos de individuos que forman parte de una población inicial para un caso con diez tipos de PCB. Se puede observar que en el primer individuo los diez tipos de PCB se agruparon en nueve grupos, mientras que en el segundo solamente en ocho.

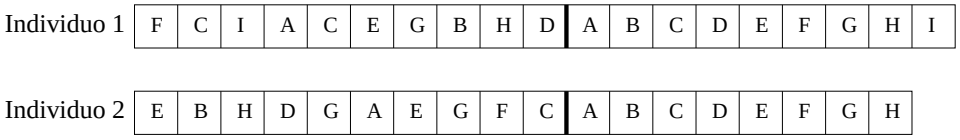


Figura 17. Dos individuos que forman parte de una población inicial para un caso de manufactura de diez tipos de PCB.

III.3.4 Selección

La técnica de selección utilizada es la que se conoce como la ruleta (De Jong, 1975). Esta técnica funciona de la siguiente manera. Imaginemos una ruleta en donde se colocan todos los cromosomas de la población. El tamaño de la sección de ruleta para cada cromosoma es proporcional al valor de su aptitud: mientras más alta sea la aptitud, más grande será la sección de ruleta que le corresponda.

Supongamos ahora que se arroja una canica y de acuerdo a la sección en donde ésta se detenga, el cromosoma correspondiente será seleccionado. Claramente, los cromosomas

con mayor aptitud serán seleccionados más veces[1].

El algoritmo de la ruleta de De Jong (1975) se muestra en el Algoritmo 4.

Algoritmo 4 Selección por ruleta

1: calcular la suma S de los valores esperados e_i de cada individuo
2: **repetir**
3: generar un número aleatorio r en el intervalo $[0, S]$
4: ciclar a través de los individuos sumando sus valores esperados hasta que la suma sea mayor o igual a r
5: el individuo que haga que esta suma cumpla con el límite es el seleccionado
6: **hasta** completar la población con los individuos seleccionados

Si consideramos una población de n individuos, por el ciclo dentro del algoritmo, es claro que se pueden hacer hasta n iteraciones, por lo que esta técnica de selección tiene un tiempo de ejecución $O(n^2)$.

III.3.5 Cruzamiento

Se utiliza el operador de cruzamiento propuesto por Falkenauer (1996), el cual trabaja sobre la parte relacionada a los grupos formados y, por consiguiente, con cromosomas de diferente tamaño. A continuación se presenta este operador.

Se seleccionan aleatoriamente dos puntos de cruza en cada padre, que delimitan la sección de cruza, para posteriormente copiar el contenido de la sección de cruza del segundo padre al inicio del primer hijo, es decir, copiar del segundo padre los grupos seleccionados para la cruza y los objetos que contienen al primer hijo. Se copian los grupos del primer padre al primer hijo, siempre y cuando no haya conflicto con los objetos que contienen, es decir, un objeto sólo puede pertenecer a un grupo. Si este conflicto existe, el grupo no se copia. Finalmente, si después de copiar los grupos existen objetos que no fueron asignados, se aplica la heurística *first fit decreasing* (Gen

[1]Introduction to Genetic Algorithms, http://cs.felk.cvut.cz/∽xobitko/ga/. Czech Technical University, Prague.

y Cheng, 2000), teniendo en cuenta las restricciones del problema. Si es necesario, se crea un nuevo grupo. Estos pasos se aplican con los roles de los padres invertidos para formar el segundo hijo.

Un ejemplo de cómo se aplica este operador se muestra en la Figura 18. Los genes sombreados representan la sección de cruza que fue seleccionada aleatoriamente (Figura 18(a): para el caso del Padre 1, los genes conteniendo los alelos C, D, E y F y para el Padre 2 los genes que contienen los alelos M y N. Ensenguida, del Padre 2 se copian al Hijo 1 los genes que contienen los alelos M y N (sección de cruza), como se muestra en la Figura 18(b). Del Padre 1 se copian al Hijo 1 todos los alelos en los respectivos genes, excepto aquellos que ya contienen información del paso anterior (Figura 18(c)), es decir, los genes que contienen los alelos B, C, D, E, G e I. En este ejemplo, al finalizar estos pasos, todos los tipos de PCB fueron asignados a algún grupo, pero no siempre se da este caso. Cuando existen PCB sin ser asignadas, lo que se hace es asignarlas mediante la heurística *first fit decreasing* (Gen y Cheng, 2000) a alguno de los grupos ya existentes. En caso de que no pueda ser asignada debido a las restricciones del problema, se crea un nuevo grupo.

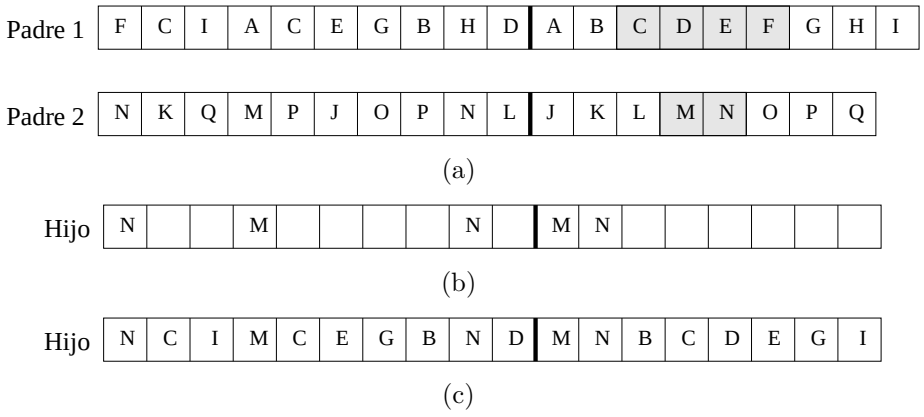


Figura 18. Operador de cruzamiento: (a) selección aleatoria de la sección de cruza en cada padre; (b) copia de la sección de cruza del segundo padre al primer hijo; y (c) copia de los grupos del primer padre al primer hijo.

En este operador, en el peor caso se copiaran todos lo grupos del Padre 2 al Hijo 1, y se buscarán todos los grupos del Padre 1 en el mismo hijo. De esta forma, este operador tiene un tiempo de ejecución $O(N^2)$, puesto que a lo más pueden haber N grupos formados.

III.3.6 Mutación

El operador de mutación, al igual que el de cruza, trabaja sobre la parte relacionada con los grupos formados. Existen tres estrategias en general: crear un nuevo grupo, eliminar uno o intercambiar objetos entre los grupos (Falkenauer, 1996). A continuación se describe el operador de mutación propuesto.

Se selecciona aleatoriamente un grupo del individuo a mutar, es decir, un gen de la parte del cromosoma correspondiente a los grupos. Se copia el contenido de los genes del individuo original al individuo mutado, exceptuando los correspondientes a las PCB asignadas al grupo seleccionado anteriormente. Se aplica *first fit decreasing* (Gen y Cheng, 2000) para asignar a los grupos existentes las PCB que quedaron sin asignación, teniendo en cuenta las restricciones del problema. Si es necesario, se crea un nuevo grupo.

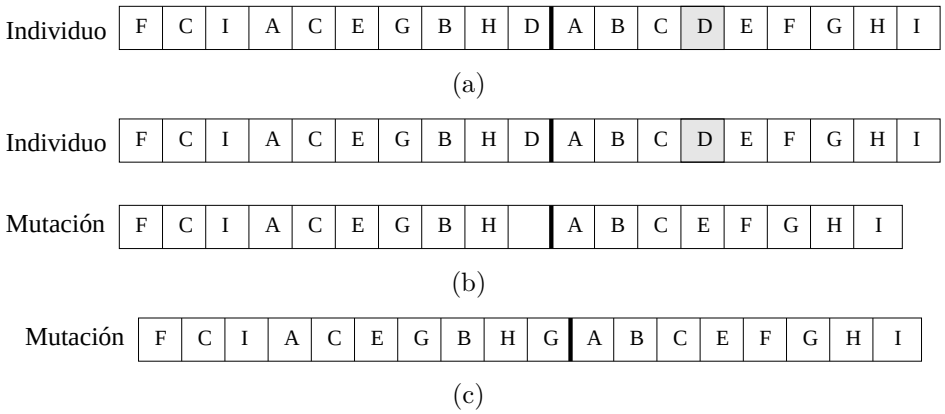


Figura 19. Operador de mutación: (a) el grupo seleccionado aleatoriamente; (b) copia de los genes del individuo original, excepto del grupo seleccionado; y (c) la PCB sin asignación se asigna al grupo G.

La Figura 19 muestra un ejemplo de este operador sobre un caso con 10 tipos de PCB. Se comienza con la selección aleatoria del grupo D en la parte del cromosoma correspondiente a los grupos (Figura 19(a)), que se muestra sombreado. Se copian todos los genes del invidiuo a la mutación, excepto aquellos que estén relacionados con el grupo D, como se muestra en la Figura 19(b). Al final sólo queda una PCB sin asignar (PCB tipo 10). Se verifica si esta PCB puede ser asignada a algún grupo existente. En este caso, como se puede observar en la Figura 19(c), la PCB tipo 10 fue asignada al grupo G.

En caso de que el tipo de PCB no pueda ser asignada a un grupo ya existente debido a las restricciones del problema, se crea un nuevo grupo y el tipo de PCB se asigna a éste.

Al igual que el operador de cruzamiento, este algoritmo tiene un tiempo de ejecución $O(N^2)$, debido a las búsquedas que se tienen que realizar en los arreglos no ordenados.

III.4 Resumen

En este capítulo se presentó la metodología que se utilizó en el desarrollo de esta tesis, se hizo una descripción de los algoritmos genéticos y de los operadores que intervienen en ellos. Se describieron las representaciones genéticas utilizadas para los problemas de manufactura de uno y de varios tipos de PCB utilizando una máquina de ensamble para este propósito, y también se hizo una descripción de los operadores utilizados en los algoritmos gen éticos, los cuales son conocidos.

El propósito de estos algoritmos es el cumplir con el objetivo de los problemas planteados en el Capítulo II, que es el minimizar el tiempo de manufactura de la producción de PCB. Estos AGs buscan soluciones para los problemas de acuerdo a funciones de aptitud basadas en el planteamiento matemático de los mismos.

Los operadores genéticos utilizados fueron seleccionados debido a que mejoran los

resultados de otras propuestas, tal como se verá en el capítulo siguiente.

Capítulo IV

Diseño de experimentos y análisis de resultados

"No puedes adquirir experiencia haciendo experimentos. No puedes crear experiencia. Debes experimentarlo."
Albert Camus (1913 - 1960)

En este capítulo se presenta el desarrollo de los experimentos llevados a cabo para los problemas 1-1 y V-1, los cuales consistieron en aplicar los algoritmos genéticos descritos en el capítulo anterior para solucionar ambos problemas.

IV.1 Casos de prueba

Debido a que los casos de prueba para los problemas estudiados en esta tesis son muy escasos, se diseñó un procedimiento para la generación de tipos de PCB, el cual se encuentra detallado en el Apéndice A. Este procedimiento, aun cuando genera PCB aleatorias, toma en cuenta algunas recomendaciones de diseñadores de éstas.

Utilizando este procedimiento se generaron 26 tipos de PCB, los cuales contienen desde 112 hasta 492 componentes, perteneciendo éstos desde 16 hasta 74 tipos diferentes. Estos datos se encuentran en la Tabla I. En esta tabla se muestran las cantidades de componentes, de tipos de componentes y de herramientas que sujetan a estos tipos de componentes para cada caso, respectivamente.

El detalle de cada caso generado, es decir, los tipos de componentes que intervienen, las herramientas que son utilizadas para sujetar éstos y la ubicación de cada componente para cada tipo de PCB, se encuentran en el Apéndice B.

Tabla I. Datos de los casos de prueba generados.

Caso	Comps.	Tipos	Herr.
1	112	16	4
2	124	18	4
3	126	20	4
4	134	21	5
5	160	24	5
6	176	26	6
7	194	28	6
8	200	30	5
9	211	31	6
10	242	35	6
11	255	37	6
12	268	41	7
13	276	42	7

Caso	Comps.	Tipos	Herr.
14	298	44	7
15	316	46	7
16	328	48	8
17	335	50	7
18	340	51	9
19	350	53	8
20	370	55	8
21	383	57	7
22	413	62	9
23	432	64	8
24	441	66	9
25	472	71	8
26	492	74	10

IV.2 Problema 1-1

Los experimentos realizados para solucionar el problema 1-1 consistieron en la combinación de los operadores de selección, cruzamiento y mutación descritos en el capítulo anterior para formar un AG y en la aplicación de este algoritmo a los casos generados. Esto es, de los dos operadores de selección, muestreo determinístico (M) y sobrante estocástico con rempleazo (S), de los dos operadores de cruzamiento, *order crossover* (O) y *position-based crossover* (P), ambos con uno y dos hijos (O1, O2, P1, P2), y de los dos operadores de mutación, mutación heurística a ambas partes del cromosoma (H) y mutación heurística a la parte del cromosoma correspondiente al PAR e inversión a la parte correspondiente al PSC (I), se formaron las 16 posibles combinaciones para el algoritmo genético.

A lo largo de las secciones siguientes, para representar una combinación de operadores para un algoritmo, se utiliza la notación del párrafo anterior. Por ejemplo, para representar el AG que usa sobrante estocástico con remplazo, *order crossover* con dos hijos y mutación heurística, se usa la notación SO1H.

Los algoritmos genéticos se aplicaron 30 veces a cada caso, y en cada repetición

existieron 600 generaciones de la población, la cual constó de 70 individuos. Las probabilidades de cruzamiento y mutación fueron 1.0 y 0.05, respectivamente.

IV.2.1 AG con order crossover - dos hijos

La Tabla II y la Figura 20 muestran los resultados obtenidos con los cuatro AG que hacen uso del operador de cruzamiento *order crossover*, tomando a los dos hijos de cada cruza.

Tabla II. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *order crossover*, considerando a ambos hijos que resultan de la cruza.

No.	No.	SO2H		SO2I		MO2H		MO2I		O2	
Caso.	Comps.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.
1	112	110.82	0.93	109.28	1.06	110.99	1.16	110.80	1.07	110.47	0.63
2	124	121.41	0.94	120.06	1.28	121.55	1.05	121.48	1.39	121.13	0.51
3	126	123.57	1.39	123.13	1.21	124.95	1.58	123.90	1.00	123.89	0.54
4	134	132.35	1.20	131.28	1.40	132.99	1.07	132.89	1.52	132.38	0.51
5	160	159.98	1.04	158.70	1.58	161.62	1.23	160.37	1.37	160.17	0.65
6	176	182.83	1.21	182.16	1.23	183.97	1.10	183.74	1.39	183.18	0.40
7	194	196.51	1.38	195.07	1.69	198.13	1.66	196.67	1.34	196.60	0.55
8	200	206.89	1.54	206.20	1.51	208.57	1.59	207.76	1.50	207.36	0.43
9	211	217.49	1.37	215.86	1.32	219.68	1.25	218.33	1.39	217.84	0.64
10	242	252.04	1.30	252.17	1.63	255.03	1.52	255.18	1.42	253.61	0.59
11	255	260.90	1.29	259.96	1.40	263.61	1.16	263.75	1.31	262.06	0.63
12	268	284.95	1.10	284.38	1.34	287.54	1.37	285.47	1.27	285.59	0.42
13	276	285.43	1.28	285.44	1.62	287.76	1.21	288.06	1.36	286.67	0.43
14	298	314.16	1.25	315.26	1.46	317.77	1.34	315.38	1.14	315.64	0.42
15	316	346.58	1.03	344.70	0.97	347.56	1.57	347.19	1.25	346.51	0.32
16	328	357.99	1.06	358.36	1.24	360.44	1.20	360.96	1.20	359.44	0.36
17	335	364.80	1.27	365.03	1.08	367.38	1.31	367.19	1.18	366.10	0.32
18	340	368.42	1.24	367.71	1.43	372.22	0.97	367.27	1.13	368.91	0.53
19	350	384.32	1.19	383.81	0.96	383.65	0.98	384.73	1.05	384.13	0.11
20	370	415.58	1.15	414.45	1.15	415.70	1.05	417.14	1.27	415.72	0.23
21	383	418.88	1.28	416.01	1.12	418.59	1.00	419.44	1.14	418.23	0.32
22	413	469.39	0.77	468.26	0.75	470.31	0.92	470.15	0.91	469.53	0.17
23	432	500.17	1.03	500.56	1.23	503.15	0.82	501.72	0.83	501.40	0.23
24	441	510.34	0.87	509.23	1.00	510.97	0.65	509.55	0.96	510.02	0.13
25	472	550.91	0.70	551.60	0.72	552.20	0.65	553.88	0.62	552.15	0.20
26	492	581.18	0.70	581.15	0.65	581.20	0.81	581.89	0.66	581.36	0.05

La primera y segunda columnas de la Tabla II identifican al caso que se está resolviendo. Las siguientes cuatro pares de columnas representan los tiempos promedio

de manufactura (en segundos) y su correspondiente desviación estándar (en porcentaje con respecto al promedio) de los cuatro AG que usan *order crossover* y consideran a ambos hijos que resultan de la cruza. Las últimas dos columnas son el promedio y la desviación estándar de los tiempos de manufactura de estos mismos algoritmos.

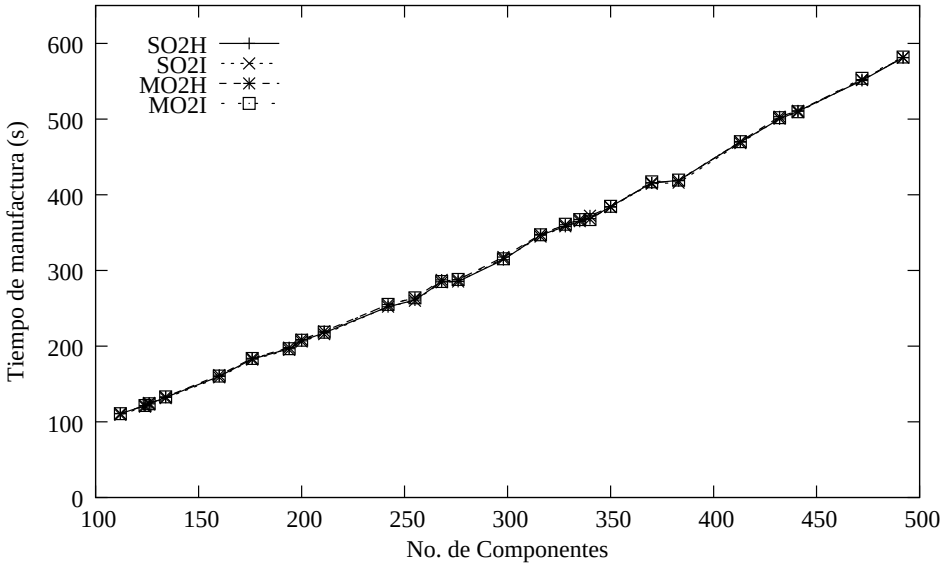


Figura 20. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *order crossover*, considerando a ambos hijos que resultan de la cruza.

De la Figura 20 se puede observar que prácticamente los cuatro algoritmos convergen al mismo resultado y esto se confirma con los datos mostrados en la Tabla II, puesto que la desviación estándar (última columna) de los resultados de los algoritmos para cada caso es muy pequeña.

La Figura 21 muestra el tiempo promedio de utilización de CPU de cada uno de los algoritmos SO1H, SO1I, MO1H y MO1I. En esta figura se puede observar que los algoritmos que usan mutación heurística en la parte del cromosoma correspondiente al PAR e inversión en la parte correspondiente al PSC, tienen una reducción de 9.01% en el tiempo de utilización de CPU, respecto a los que utilizan mutación heurística en ambas partes. Se puede conjeturar que ésto es debido a que los segundos tienen que calcular las aptitudes de cinco individuos más. Este mismo fenómeno, como se

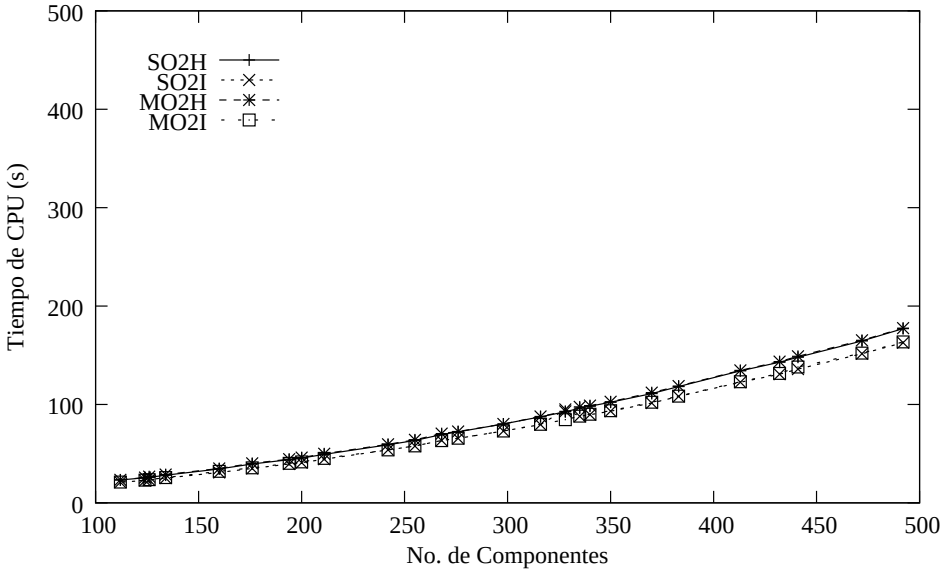


Figura 21. Tiempo promedio de utilización de CPU de los AG con *order crossover*, considerando a ambos hijos que resultan de la cruza.

observará, se repite en los experimentos siguientes.

IV.2.2 AG con order crossover - un hijo

La Tabla III y la Figura 22 muestran los resultados obtenidos con los cuatro AG que hacen uso del operador de cruzamiento *order crossover*, eligiendo al mejor hijo de cada cruza.

La primera y segunda columnas de la Tabla III identifican al caso que se está resolviendo. Las siguientes columnas representan los tiempos promedio de manufactura (en segundos) y su correspondiente desviación estándar (en porcentaje con respecto al promedio) para los diferentes AG.

Analizando la Tabla III y la Figura 22 podemos observar que el algoritmo que emplea sobrante estocástico con remplazo y mutación heurística e inversión (SO1I) tiene una leve mejora respecto a los otros tres algoritmos. De hecho, este algoritmo supera, en promedio, en 2.48% al algoritmo SO1H, en 2.61% al MO1H y en 0.60% al MO1I. Este mismo algoritmo (SO1I) supera, en promedio, al promedio de los algoritmos que

Tabla III. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *order crossover*, considerando al mejor hijo que resulta de la cruza.

No.	No.	SO1H		SO1I		MO1H		MO1I	
Caso.	Comps.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.
1	112	105.42	0.59	103.67	0.50	105.55	0.57	103.66	0.39
2	124	114.91	0.64	112.23	0.52	114.65	0.66	112.47	0.51
3	126	115.87	0.50	113.11	0.52	115.69	0.66	113.26	0.47
4	134	124.06	0.76	121.19	0.47	124.10	0.67	120.86	0.55
5	160	149.12	0.54	145.96	0.56	148.99	0.56	146.00	0.50
6	176	170.57	0.57	166.49	0.42	170.35	0.67	166.61	0.42
7	194	180.24	0.61	175.55	0.52	180.39	0.58	175.65	0.56
8	200	189.76	0.51	185.14	0.37	188.97	0.56	185.69	0.96
9	211	199.09	0.57	194.55	0.44	198.00	0.61	194.51	0.44
10	242	229.60	0.57	223.73	0.33	228.49	0.75	223.99	0.43
11	255	236.21	0.73	230.70	0.43	234.65	0.57	230.50	0.55
12	268	257.74	0.59	250.63	0.42	257.08	0.83	250.52	0.56
13	276	257.04	0.71	249.84	0.51	255.83	0.70	249.24	0.55
14	298	281.64	0.57	274.05	0.57	280.59	0.76	274.36	0.62
15	316	311.81	0.69	302.39	0.64	310.87	0.91	302.17	0.63
16	328	321.74	0.52	311.24	0.45	320.55	0.70	310.95	0.62
17	335	328.73	0.61	318.44	0.47	328.01	0.78	319.33	0.96
18	340	330.45	0.79	320.81	0.58	330.78	0.84	322.67	1.74
19	350	344.54	0.66	333.99	0.50	344.94	0.65	334.77	0.63
20	370	375.23	0.68	363.70	0.71	376.81	1.47	364.49	1.48
21	383	378.15	0.80	364.77	0.73	377.07	0.78	367.95	1.90
22	413	432.30	1.88	416.39	1.05	438.50	2.26	432.40	3.48
23	432	461.37	0.68	449.82	1.97	467.05	1.80	462.12	3.39
24	441	476.84	1.97	474.15	3.72	487.84	1.65	486.18	1.98
25	472	523.18	2.25	515.62	3.45	534.34	1.50	532.66	1.01
26	492	564.27	1.41	560.36	2.22	566.91	0.63	567.33	0.80

consideran a los dos hijos de la cruza *order crossover* en 10.47%.

La Figura 23 muestra el tiempo promedio de utilización de CPU de cada uno de los algoritmos anteriores. En esta figura se puede observar también que, los algoritmos que usan mutación heurística en la parte del cromosoma correspondiente al PAR e inversión en la parte correspondiente al PSC, tienen una pequeña reducción del 4.35% en el tiempo de utilización de CPU, respecto a los que usan mutación heurística en ambas partes del cromosoma.

Los algoritmos que consideran al mejor hijo de la cruza *order crossover* tienen una utilización, en promedio, de 108.35% más respecto a los algoritmos que consideran a

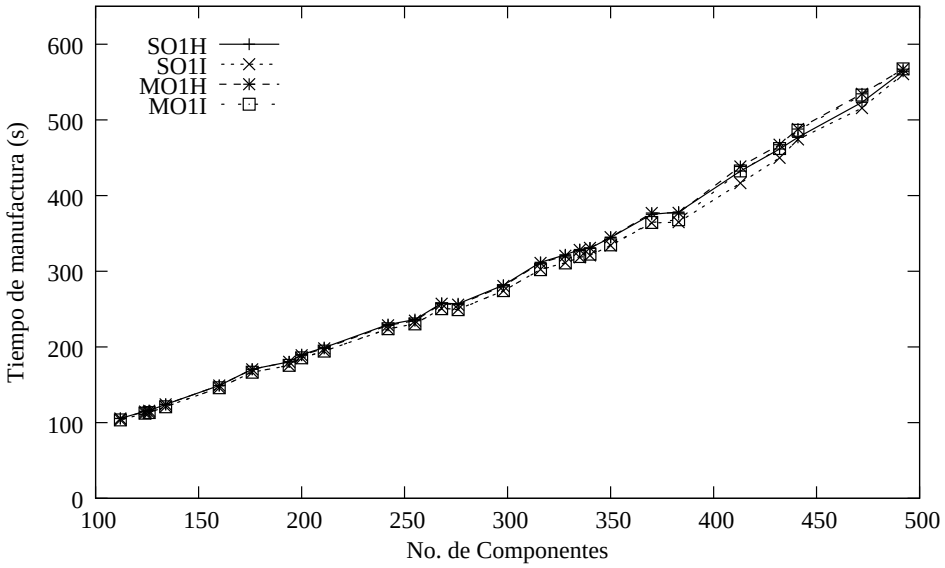


Figura 22. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *order crossover*, considerando al mejor hijo que resulta de la cruza.

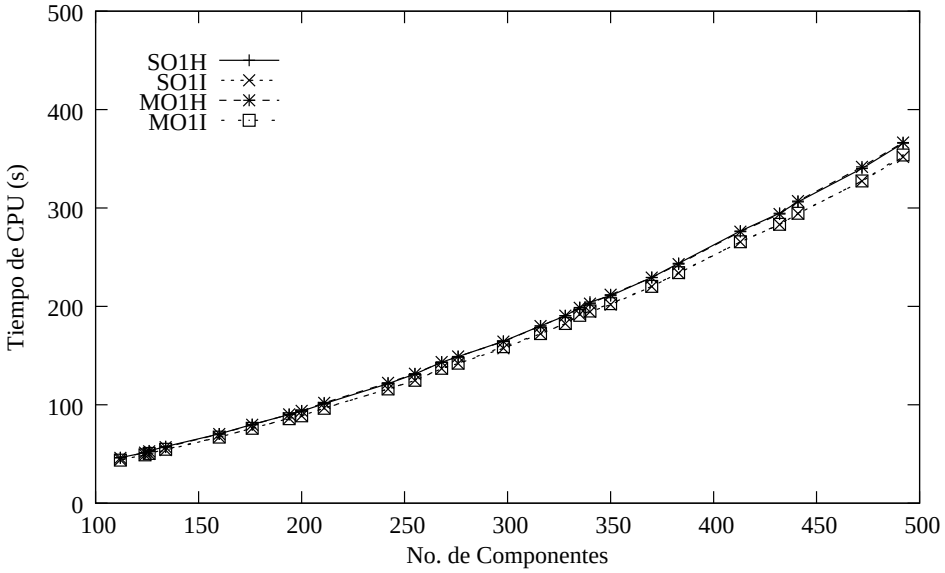


Figura 23. Tiempo promedio de utilización de CPU de los AG con *order crossover*, considerando al mejor hijo que resulta de la cruza.

ambos hijos.

IV.2.3 AG con position-based crossover - dos hijos

La Tabla IV y la Figura 24 muestran los resultados obtenidos con los cuatro AG que hacen uso del operador de cruzamiento *position-based crossover*, considerando a ambos hijos que resultan de cada cruza.

Tabla IV. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *position-based crossover*, considerando a ambos hijos que resultan de la cruza.

No.	No.	SP2H		SP2I		MP2H		MP2I		P2	
Caso.	Comps.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.
1	112	108.00	0.58	108.80	0.65	108.44	0.71	108.75	0.53	108.50	0.25
2	124	118.52	0.66	118.67	0.92	118.65	0.92	119.28	1.00	118.59	0.70
3	126	120.43	0.84	121.23	1.12	120.69	0.93	121.38	0.83	120.73	0.65
4	134	128.94	0.90	129.66	0.98	129.98	1.39	130.18	1.03	129.38	0.80
5	160	156.59	1.37	157.30	1.06	156.93	1.08	157.49	1.20	156.69	0.80
6	176	178.15	1.02	180.00	1.26	179.07	0.86	180.14	1.50	178.94	1.20
7	194	189.97	1.08	192.34	1.38	191.10	1.46	192.43	1.56	190.69	1.74
8	200	201.50	1.18	203.24	1.48	201.69	1.00	202.99	1.32	201.38	1.61
9	211	210.27	1.25	212.88	1.64	211.64	0.80	211.87	1.37	210.78	1.09
10	242	246.64	1.03	247.17	1.39	247.55	1.66	248.39	1.55	246.84	1.55
11	255	255.22	1.15	255.34	1.35	254.55	1.45	255.69	1.51	254.94	0.75
12	268	278.79	1.25	280.85	1.46	279.77	1.41	279.74	1.32	278.92	0.82
13	276	279.09	1.43	280.04	1.49	278.68	1.32	281.21	1.42	279.27	1.94
14	298	306.26	1.27	309.67	1.32	307.37	1.49	309.14	1.41	306.94	2.20
15	316	337.39	1.17	340.38	1.34	339.48	1.28	339.00	1.06	337.98	1.02
16	328	349.91	1.62	354.08	1.59	350.13	1.38	351.91	1.76	350.33	1.59
17	335	356.75	1.28	359.28	1.34	356.41	1.33	359.22	1.56	356.91	2.31
18	340	361.64	1.35	361.27	1.53	359.70	1.24	361.20	1.30	360.27	0.93
19	350	373.87	1.18	377.47	1.22	375.44	1.17	375.28	1.16	374.39	0.89
20	370	406.13	1.23	409.96	1.28	408.17	1.04	408.05	1.07	406.68	1.37
21	383	409.06	1.18	411.57	1.18	409.52	0.97	410.60	0.90	408.98	1.62
22	413	460.52	0.98	462.74	0.90	458.57	1.02	461.40	1.11	459.83	1.57
23	432	491.17	0.92	495.64	0.81	493.39	1.02	494.87	0.81	493.24	1.63
24	441	500.79	0.81	504.01	0.93	500.96	1.04	503.88	1.05	501.35	2.53
25	472	543.67	0.83	546.91	0.73	545.33	0.74	546.70	0.72	544.64	2.06
26	492	574.32	0.79	576.14	0.86	575.33	0.66	576.66	0.56	574.85	1.81

La primera y segunda columnas de la Tabla IV identifican al caso que se está resolviendo. Las siguientes cuatro pares de columnas representan los tiempos promedio de manufactura (en segundos) y su correspondiente desviación estándar (en porcentaje con respecto al promedio) de los cuatro AG que usan *position-based crossover* y consideran

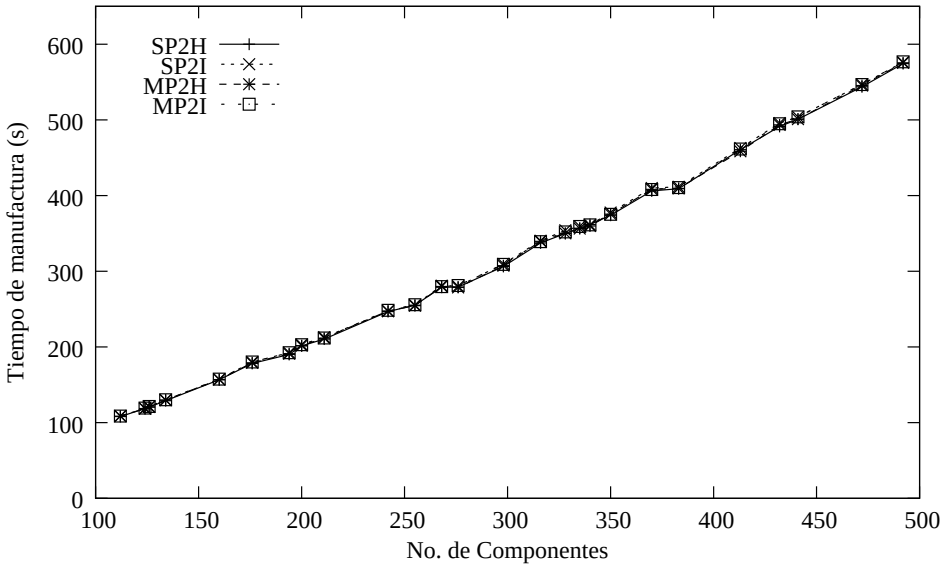


Figura 24. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *position-based crossover*, considerando ambos hijos que resultan de la cruza.

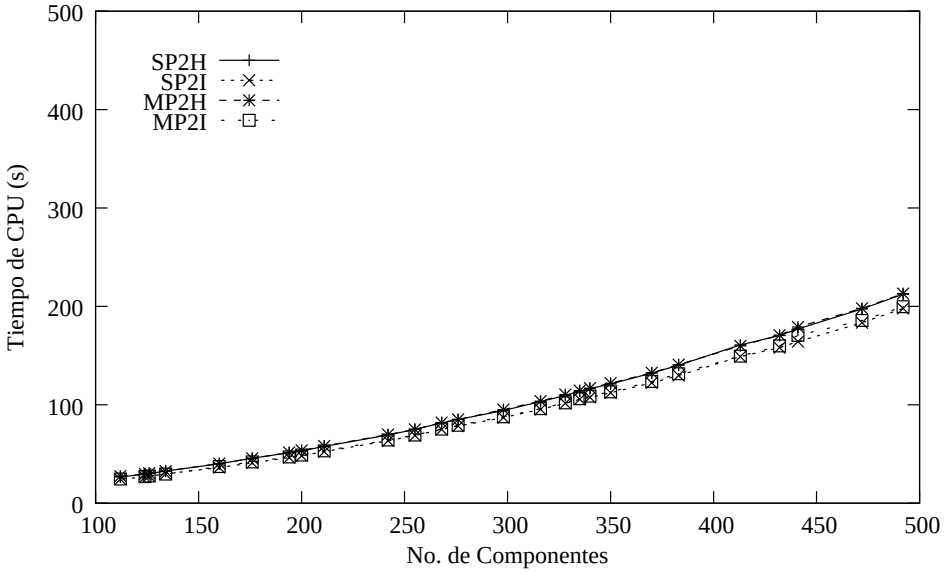


Figura 25. Tiempo promedio de utilización de CPU de los AG con *position-based crossover*, considerando ambos hijos que resultan de la cruza.

a ambos hijos que resultan de la cruza. Las últimas dos columnas son el promedio y la desviación estándar de los tiempos de manufactura de estos mismos algoritmos.

De la Figura 24 se puede observar que prácticamente los cuatro algoritmos convergen al mismo resultado, y analizando los datos de la Tabla IV se confirma, ya que la

desviación estándar de los resultados de los algoritmos es muy pequeña.

En promedio, los resultados de éstos algoritmos superan a los de los algoritmos que usan *order crossover* (dos hijos) en 2.31%. Por el contrario, se ve superado en 7.78% por el algoritmo SO1I.

La Figura 25 muestra el tiempo promedio de utilización de CPU de cada uno de los algoritmos anteriores. En esta figura se puede observar que, al igual que en los algoritmos que usan *order crossover*, los algoritmos que usan mutación heurística en la parte del cromosoma correspondiente al PAR e inversión en la parte correspondiente al PSC, tienen una reducción en el tiempo de utilización de CPU. Esta reducción es de 7.85% respecto a los algoritmos que usan mutación heurística en ambas partes del cromosoma.

IV.2.4 AG con position-based crossover - un hijo

La Tabla V y la Figura 26 muestran los resultados obtenidos con los cuatro AG que hacen uso del operador de cruzamiento *position-based crossover*, eligiendo al mejor hijo de cada cruza.

La primera y segunda columnas de la Tabla V identifican al caso que se está resolviendo. Las siguientes columnas representan los tiempos promedio de manufactura y su correspondiente desviación estándar para los diferentes AG

De la Tabla V y de la Figura 26 podemos observar que el algoritmo que emplea muestreo determinístico y mutación heurística (MP1H) tiene una mejora considerable respecto a los otros tres algoritmos. Este algoritmo supera, en promedio, en 8.38% al algoritmo SP1H, en 8.65% al SP2I, en 8.55% al MP1I, en 9.05% al promedio de los algoritmos que consideran ambos hijos de la cruza *position-based crossover*, en 11.15% al promedio de los algoritmos que consideran ambos hijos de la cruza *order crossover* y en 0.74% al algoritmo SO1I.

Tabla V. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *position-based crossover*, considerando al mejor hijo que resulta de la cruza.

No.	No.	SP1H		SP1I		MP1H		MP1I	
Caso.	Comps.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.	Prom.	D.E.
1	112	108.03	0.69	108.10	0.77	104.07	0.43	108.27	0.56
2	124	117.64	0.68	118.16	0.84	112.60	0.36	117.97	0.57
3	126	120.12	0.73	120.35	0.75	113.43	0.44	120.47	1.01
4	134	128.35	0.79	129.09	0.91	121.29	0.49	128.74	0.99
5	160	155.48	0.84	156.01	0.95	146.07	0.39	155.82	0.71
6	176	177.50	0.99	177.86	1.06	166.87	0.36	178.33	1.00
7	194	188.94	1.09	189.91	0.87	175.23	0.35	189.72	0.98
8	200	200.11	1.16	199.90	0.95	185.72	0.36	199.54	1.00
9	211	208.54	1.05	209.22	0.84	194.28	0.49	209.10	1.42
10	242	244.38	1.06	243.63	1.18	224.17	0.58	243.99	1.28
11	255	253.25	1.17	253.10	1.24	230.89	0.58	252.32	1.04
12	268	276.45	1.09	276.94	1.42	250.15	0.70	277.11	1.45
13	276	275.93	1.04	277.34	1.20	248.83	0.57	277.32	1.29
14	298	305.10	1.28	306.74	1.24	273.65	0.76	305.03	0.99
15	316	335.17	1.32	335.74	1.41	302.75	1.00	334.97	1.28
16	328	346.39	1.01	350.36	1.48	311.25	0.50	347.56	1.00
17	335	352.92	1.04	355.58	1.33	318.34	0.73	355.00	1.17
18	340	356.49	1.03	356.46	1.42	319.30	0.69	356.39	1.00
19	350	371.46	1.13	372.50	1.00	332.99	0.67	373.28	1.07
20	370	404.08	1.07	405.12	0.93	362.62	0.74	404.53	1.15
21	383	406.21	0.87	408.82	1.17	363.71	0.75	407.42	1.16
22	413	459.41	1.05	458.01	0.78	412.03	0.79	458.01	1.05
23	432	490.68	1.13	492.99	0.83	440.46	0.87	491.93	0.87
24	441	499.05	0.72	500.67	0.80	453.99	1.03	500.22	0.89
25	472	542.69	0.75	544.79	0.71	491.66	0.83	544.09	0.79
26	492	572.31	0.92	573.02	0.84	522.41	1.13	573.64	0.74

La Figura 27 muestra el tiempo promedio de utilización de CPU de cada uno de los algoritmos anteriores. Al igual que en los casos anteriores, los algoritmos que utilizan mutación heurística en ambas partes del cromosoma, tienen una mayor utilización del CPU (3.66% más).

Los algoritmos que consideran al mejor hijo de la cruza *position-based crossover* tienen una utilización, en promedio, de 108.35% más respecto a los algoritmos que consideran a ambos hijos.

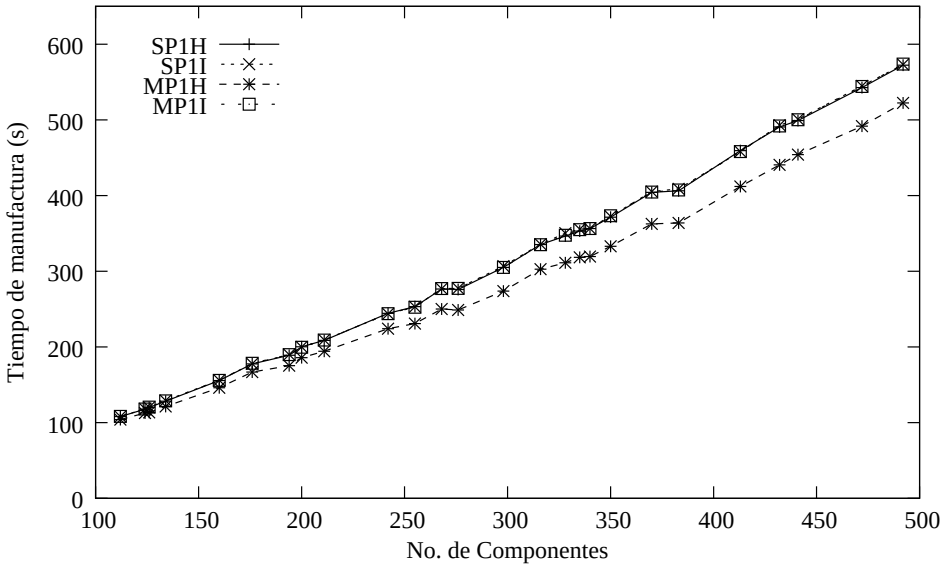


Figura 26. Tiempo promedio de manufactura de los casos de prueba generados, usando los AG con *position-based crossover*, considerando al mejor hijo que resulta de la cruza.

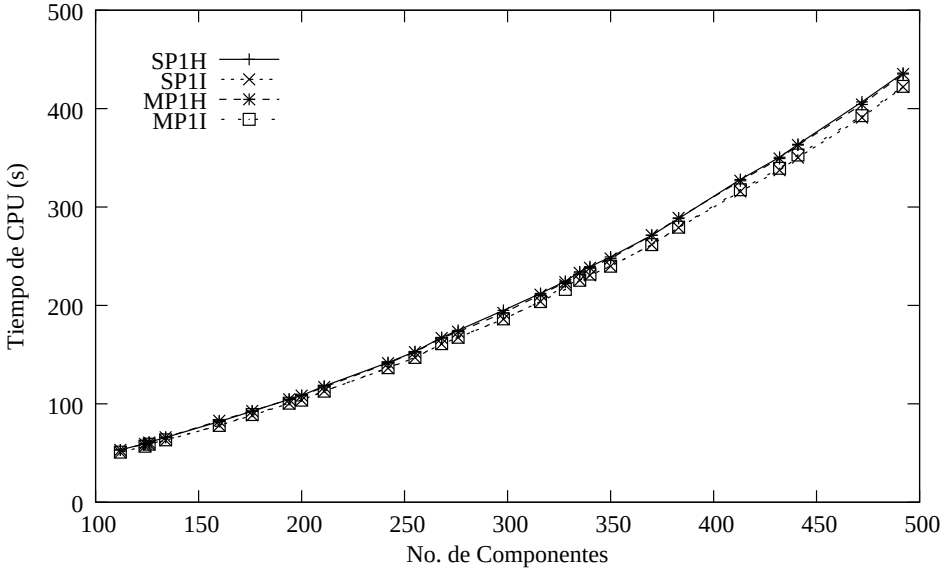


Figura 27. Tiempo promedio de utilización de CPU de los AG con *position-based crossover*, considerando al mejor hijo que resulta de la cruza.

IV.2.5 BV+TW y order crossover vs. position-based crossover

Haciendo un análisis de los resultados de todos los AG anteriores, se puede decir que los que obtienen mejores resultados son el que usa sobrante estocástico con reemplazo, *order*

crossover (un hijo) y mutaciones heurística e inversión (SO1I), y el que usa muestreo determinístico, *position-based crossover* (un hijo) y mutación heurística (MP1H).

A continuación se presenta la Tabla VI y la Figura 28, que muestran los resultados de los algoritmos mencionados en el párrafo anterior, además de los resultados obtenidos con los métodos de búsqueda voraz y *type-writer* (BV+TW) para los casos de prueba.

Tabla VI. Resultados del método BV+TW y de los algoritmos SO1I y MP1H aplicados a los casos de prueba.

			SO1I		MP1H		MP1H vs.	
Caso	Comps.	BV+TW	Prom.	D. E.	Prom.	D. E.	BV+TW	SO1I
1	112	115.52	103.67	0.50	104.07	0.43	9.91	-0.39
2	124	127.36	112.23	0.52	112.60	0.36	11.59	-0.33
3	126	131.18	113.11	0.52	113.43	0.44	13.53	-0.28
4	134	138.93	121.19	0.47	121.29	0.49	12.70	-0.08
5	160	166.33	145.96	0.56	146.07	0.39	12.18	-0.08
6	176	188.16	166.49	0.42	166.87	0.36	11.31	-0.23
7	194	201.65	175.55	0.52	175.23	0.35	13.10	0.18
8	200	212.83	185.14	0.37	185.72	0.36	12.74	-0.31
9	211	221.32	194.55	0.44	194.28	0.49	12.22	0.14
10	242	260.63	223.73	0.33	224.17	0.58	13.99	-0.20
11	255	271.12	230.70	0.43	230.89	0.58	14.84	-0.08
12	268	293.85	250.63	0.42	250.15	0.70	14.87	0.19
13	276	292.77	249.84	0.51	248.83	0.57	15.01	0.40
14	298	316.70	274.05	0.57	273.65	0.76	13.59	0.15
15	316	353.72	302.39	0.64	302.75	1.00	14.41	-0.12
16	328	364.85	311.24	0.45	311.25	0.50	14.69	0.00
17	335	373.08	318.44	0.47	318.34	0.73	14.67	0.03
18	340	377.05	320.81	0.58	319.30	0.69	15.32	0.47
19	350	392.92	333.99	0.50	332.99	0.67	15.25	0.30
20	370	423.36	363.70	0.71	362.62	0.74	14.35	0.30
21	383	430.72	364.77	0.73	363.71	0.75	15.56	0.29
22	413	489.93	416.39	1.05	412.03	0.79	15.90	1.05
23	432	520.79	449.82	1.97	440.46	0.87	15.42	2.08
24	441	530.67	474.15	3.72	453.99	1.03	14.45	4.25
25	472	579.26	515.62	3.45	491.66	0.83	15.12	4.65
26	492	616.96	560.36	2.22	522.41	1.13	15.33	6.77
						Promedio	**13.93**	**0.74**

La primera y segunda columnas de la Tabla VI identifica al caso que se está resolviendo. La tercera columna corresponde al tiempo de manufactura mediante el algoritmo de búsqueda voraz y la heurística *type-writer* (en segundos). La cuarta y quinta

columnas corresponden al tiempo de manufactura promedio mediante el algoritmo SO1I (en segundos) y su desviación estándar (en porcentaje respecto al promedio), respectivamente. La sexta y séptima columnas corresponden al tiempo de manufactura promedio mediante el algoritmo MP1H (en segundos) y su desviación estándar (en porcentaje respecto al promedio), respectivamente. La penúltima y última columnas corresponden al procentaje de mejora del algoritmo MP1H respecto al método BV+TW y al algoritmo SO1I, respectivamente.

De la Figura 28 se puede observar que Los resultados de los AG superan al método BV+TW en todos los casos. Otro punto a notar es que estos AG obtienen resultados similares, hasta cierto caso de entrada. Analizando la Tabla VI, vemos que este caso es el 22, con 413 componentes, que es para el cual el AG MP1H comienza a superar al SO1I. También se observa una similitud en el comportamiento de los tres métodos. Podemos conjeturar que esto sucede debido a que la mutación heurística y la selección del mejor hijo son procedimientos voraces.

La diferencia entre estos dos algoritmos radica principalmente en el operador de cruzamiento, ya que *order crossover* selecciona un segmento del cromosoma que se conserva en el hijo, lo cual puede no dar lugar a que se exploren adecuadamente otros puntos dentro del espacio de soluciones, pues se está conservando una misma asignación de ranuras y una misma secuencia de colocación. En cambio, *position-based crossover* selecciona genes que no necesariamente son contiguos y por ende no se conservan ni la asignación de ranuras, ni la secuencia de colocación que proponen los individuos.

Adicionalmente, la Figura 29 muestra el tiempo de utilización de CPU de los algoritmos genéticos y del método BV+TW. Éste último siempre se mantuvo por debajo de los dos segundos para todos los casos.

Se puede observar de la Figura 29 que para el caso de prueba más pequeño (Caso 1, 112 componentes), al AG MP1H le toma aproximadamente 50 segundos en dar una

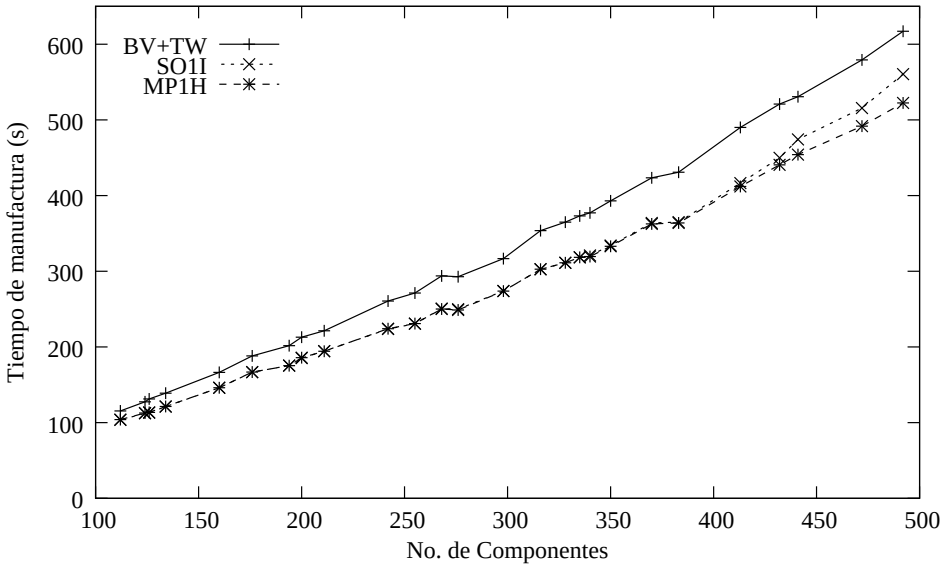


Figura 28. Resultados del método BV+TW y de los algoritmos SO1I y MP1H aplicados a los casos de prueba.

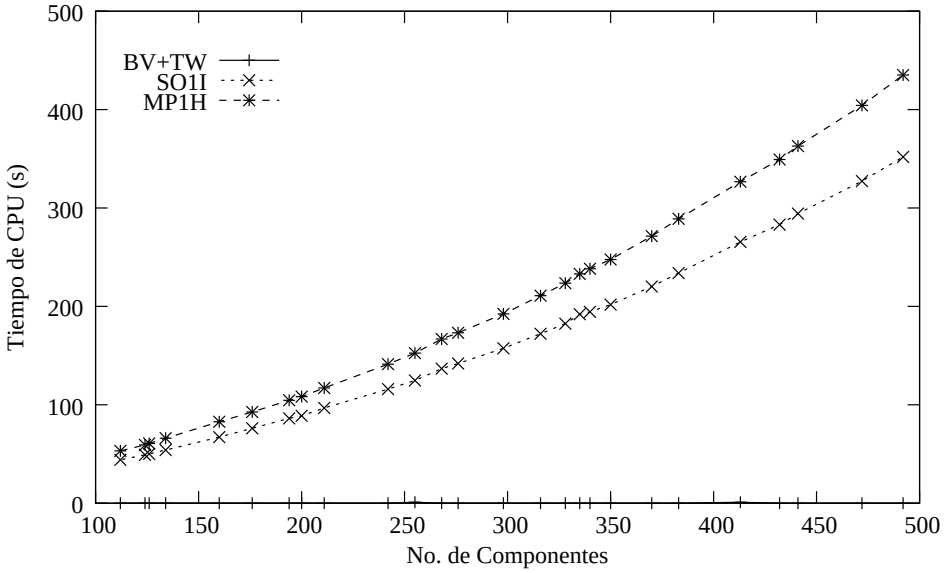


Figura 29. Tiempo de utilización de CPU del método BV+TW y de los algoritmos genéticos SO1I y MP1H.

solución al problema 1-1 que tarda 104 segundos (analizando la Tabla VI) para manufacturar una tarjeta de este tipo. Si se tuviera que manufacturar, por ejemplo, 500 tarjetas como la de éste caso de prueba, al algoritmo genético le tomaría alrededor de 1,500 segundos calcular una solución promedio (50 segundos por cada una de las

30 repeticiones) y la máquina de manufactura requeriría 52,000 segundos aproximadamente para la manufactura de todas las tarjetas (104 segundos por cada PCB), dando un tiempo total de 53,500 segundos. Por otro lado, el método BV+TW tarda aproximadamente 57,500 segundos en manufacturar todas las PCB (115 segundos por cada una). Estos datos muestran que, si se utiliza el AG para solucionar el problema, hay un ahorro de 4,000 segundos (un poco más de una hora).

Si se hace el mismo ejercicio con el caso más grande (Caso 26, 492 componentes), se tiene un ahorro de 33,500 segundos (un poco más de nueve horas). Con estos datos se puede decir que el AG es adecuado para solucionar el problema 1-1.

IV.2.6 AG MP1H aplicado al caso de prueba de Leu *et al.* (1993)

Leu *et al.* (1993), describen un caso a detalle, formado por 200 componentes de 10 tipos diferentes. De acuerdo con los autores, todos los componentes pueden ser sujetados con la misma herramienta, por lo que no es necesario hacer cambios de éstas y se contempla a la distancia recorrida por el brazo como índice de calidad de la solución. Ellos presentan sus resultados, los cuales son superados por Ong y Khoo (1999), y éstos a su vez por Ho y Ji (2005). El algoritmo genético MP1H también se aplicó al caso en cuestión. Los resultados se muestran en la Tabla VII. En este caso, los parámetros se modificaron con fines de comparación. La probabilidad de cruza y de mutación fueron 1.0 y 0.2, respectivamente.

Tabla VII. Comparación de resultados para el caso descrito en (Leu *et al.*, 1993).

	Leu *et al.* (1993)	Ong y Khoo (1999)	Ho y Ji (2005)	AG MP1H
Tamaño pob.	100	100	25	25
No. generaciones	6150	6150	3000	3000
Mejor soln. (cm)	6129 aprox.	5673.7	5660.5	5618.1
Prom. sols. (cm)	-	-	-	5641.3
Desv. std. (%)	-	-	-	0.27

Las parámetros utilizados y resultados obtenidos por Leu *et al.* (1993), Ong y Khoo (1999), Ho y Ji (2005) y el AG propuesto, se muestran en la segunda, tercera, cuarta y quinta columnas, respectivamente, de la Tabla VII. El segundo renglón de la misma representa el tamaño de la población y el tercero el número de iteraciones o generaciones. En el cuarto están las mejores soluciones obtenidas. El quinto y el sexto muestran el promedio y la desviación estándar, respectivamente, de las soluciones.

Desafortunadamente los trabajos anteriores (Leu *et al.*, 1993; Ong y Khoo, 1999; Ho y Ji, 2005) no publican resultados promedio, por lo que sólamente se pueden hacer comparaciones de acuerdo a los mejores resultados obtenidos. En este caso, la mejor solución obtenida por el AG MP1H supera en 0.75% a la mejor solución conocida (Ho y Ji, 2005) para dicho caso de prueba. Inclusive, si se toma en cuenta el resultado promedio más la desviación estándar se sigue superando al resultado de Ho y Ji (2005) (0.07%). Este resultado se presenta en (García-Nájera y Brizuela, 2005).

IV.3 Problema V-1

Los experimentos realizados para el problema V-1 consistieron de dos fases. La primera es aplicar el algoritmo genético descrito en el capítulo anterior al caso de prueba detallado por Narayanaswami e Iyengar (2004), y comparar el resultado con el publicado por ellos.

La segunda es aplicar el mismo algoritmo genético a los casos de prueba para este problema. Estos casos de prueba se formaron con subconjuntos de los tipos de PCB que se generaron para el problema 1-1 y los cuales se presentan en el Apéndice B.

El AG propuesto, para cada caso, se ejecutó 30 veces, con 300 iteraciones y 25 individuos en la población. Las probabilidades de cruzamiento y de mutación fueron 1.0 y 0.05, respectivamente.

IV.3.1 AG Propuesto aplicado al caso de prueba de Narayanaswami e Iyengar (2004)

Narayanaswami e Iyengar (2004), describen un caso a detalle, formado por 12 tipos de PCB, cuyas cantidades de tipos de componentes varian entre 10 y 17, de un universo de 30.

El AG propuesto se aplicó a este caso y los resultados se comparan con los publicados en (Narayanaswami e Iyengar, 2004) y se muestran en la Tabla VIII.

Tabla VIII. Comparación de resultados para el caso descrito por Narayanaswami e Iyengar (2004).

No.	Narayanaswami e Iyengar (2004)		AG Propuesto	
PCB	No. Grupos	No. Cambios	No. Grupos	No. Cambios
12	6	62	5	45

Cabe resaltar que, al final de su trabajo, Narayanaswami e Iyengar (2004), utilizan la política *keep tools needed soonest* (KTNS), que trata de conservar en el anaquel alimentador algunos tipos de componentes que se utilizarán en grupos futuros, siempre y cuando haya espacio suficiente. Esta política sólo intenta reducir el número de cambios en el anaquel alimentador cuando se termina de manufacturar un grupo de PCB y se comienza con otro, lo que trae consigo una mejora en el tiempo de manufactura de todos los grupos de PCB. En los resultados mostrados en la Tabla VIII y en los siguientes, no se considera esta política, ni en la heurística ni en el AG.

IV.3.2 AG propuesto aplicado a los casos de prueba generados

De los 26 tipos de PCB generados, se formaron 20 casos que contienen entre 12 y 20 tipos de PCB. Se realizaron experimentos tanto con la heurística propuesta por Narayanaswami e Iyengar (2004), como con el algoritmo genético propuesto. Los resultados para los grupos formados se muestran en la Tabla IX y en la Figura 30, y para la

cantidad de cambios en la Tabla X y en la Figura 31.

Tabla IX. Comparación de números de grupos formados para los casos de prueba generados.

No. Caso	No. PCB	Heurística	AG Propuesto Mejor	AG Propuesto Prom.	AG Propuesto D. E.	% Mejora AG Propuesto
1	12	6	5	5.00	0.00	16.67
2	12	7	5	5.03	3.57	28.14
3	12	7	5	5.07	4.92	27.57
4	12	5	4	4.00	0.00	20.00
5	14	9	7	7.20	5.56	20.00
6	14	9	7	7.17	5.20	20.33
7	14	10	6	6.43	7.70	35.70
8	14	8	6	6.10	4.92	23.75
9	16	9	8	8.00	0.00	11.11
10	16	5	3	3.13	10.85	37.40
11	16	10	7	7.00	0.00	30.00
12	16	8	4	4.07	6.13	49.13
13	18	10	5	5.03	3.57	49.70
14	18	9	7	7.00	0.00	22.22
15	18	9	6	6.67	7.07	25.89
16	18	11	8	8.00	0.00	27.27
17	20	10	8	8.00	0.00	20.00
18	20	9	6	6.00	0.00	33.33
19	20	11	7	7.13	4.77	35.18
20	20	11	8	8.00	0.00	27.27
				Promedio		28.03

La primera y segunda columnas de la Tabla IX identifican el caso que se está resolviendo. La tercera columna muestra el resultado de la cantidad de grupos formados mediante la heurística propuesta por Narayanaswami e Iyengar (2004). La cuarta, quinta y sexta columnas muestran el mejor resultado, el promedio y la desviación estandar (en porcentaje), respectivamente, de las soluciones encontradas mediante el AG propuesto. La última columna muestra el procentaje de mejora del AG respecto a la heurística.

La primera y segunda columnas de la Tabla X identifican el caso que se está resolviendo. La tercera columna muestra el resultado de la cantidad de cambios realizados mediante la heurística propuesta por Narayanaswami e Iyengar (2004). La cuarta,

Tabla X. Comparación de números de cambios para los casos de prueba generados.

No. Caso	No. PCB	Heurística	AG Propuesto Mejor	AG Propuesto Prom.	AG Propuesto D. E.	% Mejora AG Propuesto
1	12	415	384	392.03	1.71	5.53
2	12	549	485	488.37	1.33	11.04
3	12	510	486	492.13	1.22	3.50
4	12	379	370	371.20	0.26	2.06
5	14	754	726	734.10	1.08	2.64
6	14	794	732	739.63	1.42	6.85
7	14	699	616	627.83	2.20	10.18
8	14	638	596	600.50	1.02	5.88
9	16	771	736	749.80	1.13	2.75
10	16	310	259	260.90	2.09	15.84
11	16	672	648	648.60	0.14	3.48
12	16	435	369	374.93	2.12	13.81
13	18	541	484	489.07	0.90	9.60
14	18	730	696	705.70	0.74	3.33
15	18	679	622	633.03	1.49	6.77
16	18	825	772	789.70	0.97	4.28
17	20	822	742	762.37	1.03	7.25
18	20	615	537	555.97	1.66	9.60
19	20	837	726	733.00	1.53	12.43
20	20	828	784	802.13	0.95	3.12
				Promedio		7.00

quinta y sexta columnas muestran el mejor resultado, el promedio y la desviación estandar, respectivamente, de las soluciones encontradas mediante el AG propuesto. La última columna muestra el procentaje de mejora del AG respecto a la heurística.

De los resultados obtenidos, se observa que en todos los casos hay una reducción tanto en el número de grupos formados como en el número de cambios necesarios para la manufactura de todos los grupos de PCB. Se puede conjeturar que esto se debe a que la heurística de Narayanaswami e Iyengar (2004), al tratar de incluir un tipo de PCB en algún grupo, sólo se contempla el que actualmente se está formando y no a los que ya existen. De hecho, esta heurística puede mejorar si se considera lo anterior, a un costo mayor en el tiempo de cómputo.

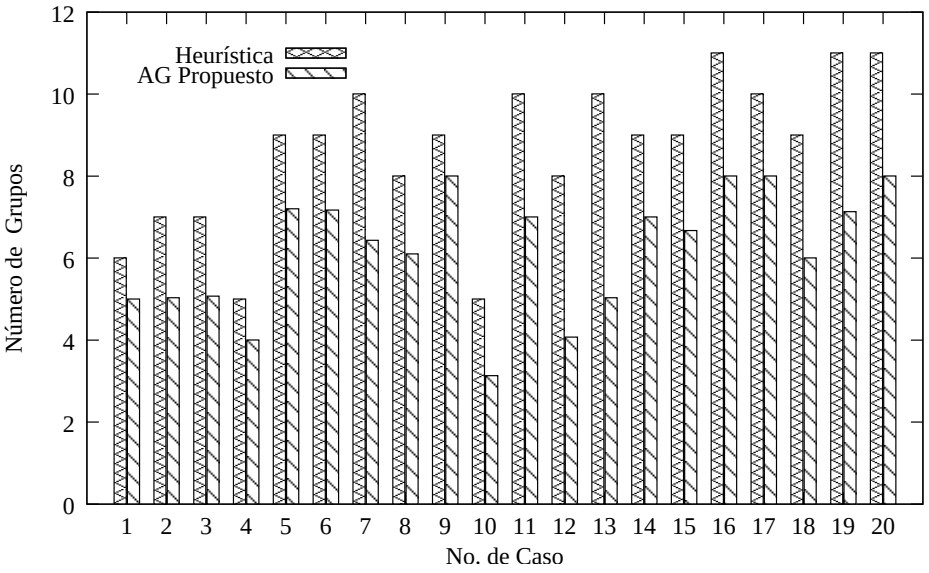


Figura 30. Resultados de los grupos formados mediante la heurística propuesta por Narayanaswami e Iyengar (2004) y el AG propuesto.

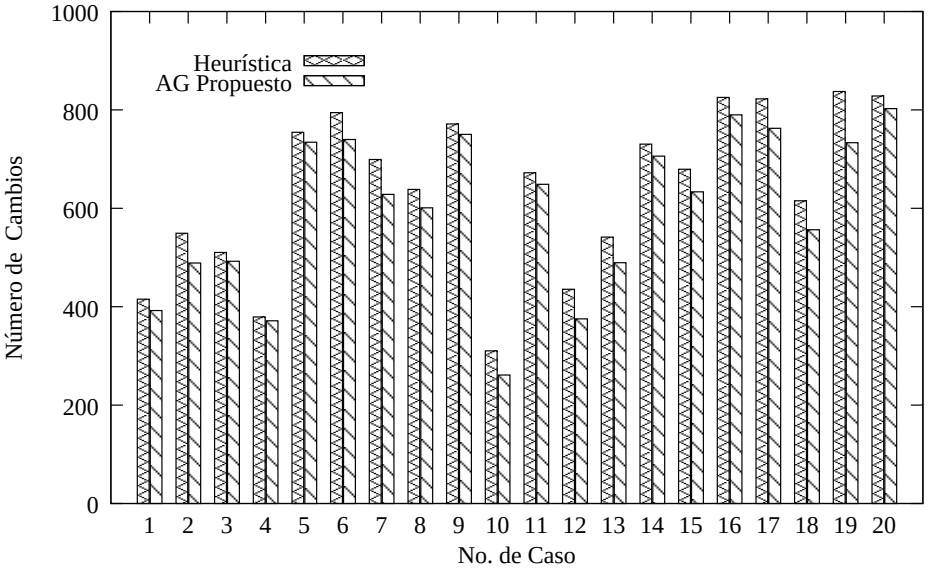


Figura 31. Resultados de los cambios realizados mediante la heurística propuesta por Narayanaswami e Iyengar (2004) y el AG propuesto.

IV.4 Cotas inferiores

Una forma de evaluar la calidad absoluta de las soluciones generadas por los algoritmos es encontrar cotas inferiores para cada caso de prueba. Esto se puede llevar a cabo

resolviendo los problemas P1-1 y PV-1G de programación lineal (el problema 1-1 modelado en (2) con sus restricciones (3)-(10), tiene una versión de programación lineal, la cual se presenta en el Apéndice C) y el problema PV-1C de programación cuadrática. Existen algunas herramientas computacionales para resolver estos problemas, algunas de ellas son LINDO (Schrage, 1984), el resolvedor de Microsoft Excel[1] y MATLAB Optimization Toolbox (The Math Works, 2001).

Cuando se quiere solucionar, por ejemplo, un caso del problema 1-1 con tres componentes de tres tipos diferentes y una máquina con tres ranuras, se requiere resolver un sistema con 18 variables binarias y 12 restricciones. Si el número de componentes se incrementa a cinco, las variables binarias se incrementan a 60 y las restricciones a 44. En general, este número de restricciones crece como $O(2^n)$, donde n es el número de componentes. En específico, la restricción (5) es la que tiene este orden de crecimiento. Es decir que para los casos de prueba utilizados en este trabajo, el número de restricciones es muy grande y las herramientas computacionales con las que se cuenta no podrían utilizarse.

Se pueden hacer algunas consideraciones (Kumar y Li, 1995) con el fin de que el número de restricciones sea menor y se puedan utilizar las herramientas computacionales para encontrar la solución a estos casos de prueba, pero esto se propone como un trabajo futuro.

IV.5 Resumen

En este capítulo se presentaron los resultados de los experimentos realizados para los problemas 1-1 y V-1. Para el primero de ellos, los resultados se mostraron de acuerdo al operador de cruzamiento utilizado por los AG. De esta forma fue evidente comprobar que los algoritmos que seleccionan al mejor de los hijos de cada cruza tienen un mejor

[1] http://www.solver.com/

desempeño y logran encontrar mejores soluciones que los algoritmos que contemplan a ambos hijos. Esto obviamente tiene un costo adicional en la utilización de CPU, puesto que por cada cruza se tienen que calcular las aptitudes de los nuevos individuos.

También se pudo verificar que, para este problema en específico, el operador de cruza *position-based crossover* influye para que los algoritmos genéticos obtengan mejores resultados, al comparar éstos con los que hacen uso de *order crossover*.

Para el segundo problema, los experimentos consistieron en aplicar el AG propuesto a un conjunto de casos de prueba generados y en comparar los resultados con aquellos que resultan de aplicar una heurística basada en el coeficiente de similitud de Jaccard (Narayanaswami e Iyengar, 2004). En este caso, el algoritmo que se propone en esta tesis muestra superioridad respecto al otro método, pues las soluciones que encuentra resultan en menor número de grupos de PCB formados y en menor número de cambios en el anaquel alimentador. De hecho, esta es la primera vez que se utiliza un algoritmo genético para solucionar el problema V-1.

En general, se puede afirmar que el cómputo evolutivo, en específico, los algoritmos genéticos, son un método eficiente para solucionar los problemas de estudio en esta tesis.

Capítulo V

Conclusiones y trabajo futuro

"Cree en aquellos quienes están en busca de la verdad. Duda de aquellos quienes la encuentran."
André Gide (1869 – 1951)

En esté capítulo se presenta un resumen del trabajo realizado, se hacen algunas conclusiones y conjeturas, y se proponen algunas ideas como trabajo futuro de investigación.

V.1 Resumen

Este trabajo trató sobre un estudio comparativo de los resultados de algunas técnicas utilizadas para resolver dos problemas en la manufactura de tarjetas de circuitos impresos. El primero de ellos es el de manufactura de un tipo de PCB con una máquina (problema 1-1), el cual contempla dos subproblemas: el problema de asignación de ranuras (PAR) y el problema de secuencia de colocación de componentes (PSC). El segundo de ellos es el problema de manufactura de varios tipos de PCB utilizando una máquina (problema V-1), que a su vez contiene dos subproblemas: el problema de agrupamiento de tipos de PCB y el problema de secuenciamiento de grupos de tipos de PCB. Ambos problemas pertenecen a la clase **NP-difícil**, lo que quiere decir que no se conoce un método eficiente (polinomial) de solución que asegure la solución óptima para todos los casos de dichos problemas.

El primero de estos problemas (problema 1-1), ha sido estudiado desde la decada de los ochentas. Desde entonces se han propuesto métodos y heurísticas que se aplicaron sobre algunos casos de prueba. Pocos de estos casos han sido publicados, los cuales han servido para comparar algunas de las técnicas de solución, mejorando éstas con

el paso del tiempo. En esta tesis se desarrollaron varios algoritmos genéticos que se aplicaron sobre casos de prueba generados mediante un método que se documenta en el Apéndice A. Estos algoritmos genéticos combinan varios operadores que se describieron en el Capítulo III. De los resultados obtenidos en los experimentos sobre los casos de prueba, el algoritmo que hace uso del muestreo determinístico como técnica de selección, de *position-based crossover* como operador de cruzamiento (seleccionando al mejor hijo de cada cruzamiento), y de mutación heurística, obtiene los mejores resultados. Es este algoritmo el que se aplicó posteriormente a un caso de prueba disponible públicamente, mejorando los resultados de otros AG que se han propuesto para este problema.

El segundo (problema V-1) ha sido menos estudiado y ninguno de los trabajos relacionados con él contempla al cómputo evolutivo como una alternativa para solucionarlo. El algoritmo genético propuesto en este documento es el primero conocido que se utiliza para resolver este problema. Este algoritmo se aplicó sobre un caso de prueba que se encuentra disponible publícamente y sobre algunos casos generados haciendo uso de los tipos de PCB utilizados en los casos de prueba para el problema 1-1. Los resultados se compararon con aquellos que produjo una heurística que hace uso del coeficiente de similitud de Jaccard, recientemente propuesta (2004). En base a los resultados obtenidos, fue el AG el método que propone mejores soluciones respecto a la heurística. No obstante los resultados, este trabajo tiene una desventaja, al igual que varios otros estudios que se han realizado sobre el problema V-1, y es que resuelve este problema separando el claro vínculo que existe entre éste y el problema 1-1. No es evidente que la solución óptima para ambos problemas sea la solución óptima para el problema completo.

Debido a la naturaleza no-determinística de los AG, fue necesario hacer un análisis de los resultados obtenidos, basado en la obtención de valores promedio y desviación estándar de los tiempos de manufactura para el problema 1-1, y del número de grupos

formados y cantidad de cambios necesarios para el problema V-1.

V.2 Conclusiones

En el Capítulo IV se presentaron los resultados de los AG desarrollados para el problema 1-1. De todos estos algoritmos fue el MP1H el que presentó los mejores. Se puede conjeturar que esto es debido al operador de cruzamiento que utiliza, ya que, como se describió en el Capítulo III, este operador intenta hacer búsquedas en otros puntos del espacio de soluciones y no quedarse estancado en óptimos locales. A diferencia del operador *order crossover*, *position-based crossover* no necesariamente conserva la asignación de ranuras y secuencia de colocación que traen consigo los padres, debido a que las ubicaciones de los genes que son seleccionados para conservarse de los padres a los hijos, son independientes unas de otras.

Para este problema se pudo comprobar que las técnicas de selección no fueron parte fundamental para la obtención de mejores resultados, no así la selección del mejor hijo producto de la cruza, en lugar de ambos hijos. Debido a esto último, existe un incremento en la utilización de CPU, ya que para cada hijo se tiene que calcular su aptitud. Este mismo fenómeno sucede cuando se utiliza la mutación heurística, pues tiene que calcular la aptitud de más individuos.

Por otro lado, la diferencia entre los resultados del algoritmo genético propuesto para resolver el problema V-1 y aquellos que fueron arrojados por la heurística con la cual se comparó, es debida a que ésta última no busca incorporar los tipos de PCB a grupos que ya están formados, sino únicamente al grupo que se está construyendo en ese momento. Por el contrario, el AG hace esta búsqueda gracias a los operadores de cruzamiento y mutación. También podemos concluir que, una vez formados los grupos, la heurística hace una búsqueda voraz (enumera todas las posibles soluciones y selecciona la mejor), de aquí que si no se agrupan adecuadamente los tipos de PCB,

los resultados podrían ser superados por otros métodos.

De acuerdo con el análisis de los resultados obtenidos se puede concluir que el cómputo evolutivo, en específico, los algoritmos genéticos, son una herramienta eficaz y eficiente para su aplicación en la resolución de los problemas planteados en esta tesis.

V.3 Trabajo futuro

Como resultado de este trabajo de tesis, las siguientes ideas se presentan como propuestas para trabajo futuro sobre la misma línea de investigación.

- Considerar que más de una ranura puede contener componentes de un mismo tipo, es decir, contemplar el plan de recuperación (PR).
- Extender el problema 1-1 a otras arquitecturas de máquinas de manufactura de PCB.
- Considerar la política *keep tools needed soonest* en el problema V-1.
- Hacer uso de alguna herramienta computacional para encontrar cotas inferiores para los casos de prueba estudiados y comparar con los resultados obtenidos por los AG, tanto para el problema 1-1, como para el problema V-1.
- Considerar el vínculo que existe entre los problemas 1-1 y V-1 para resolver este último.
- Comenzar con el estudio de los problemas 1-V y V-V.

Bibliografía

Ammons, J. C., M. Carlyle, L. L. Crammer, G. DePuy, K. Ellis, L. F. McGinnis, C. A. Tovey, y H. Xu 1997. "Component allocation to balance workload in printed circuit card assembly systems". IIE Transcations, 26:265-275 p.

Balakrishnan, A. y F. Vanderbeck 1999. "A tactical planning model for mixed-model electronics assembly operations". Operations Research, 47(3):395-409 p.

Ball, M. O. y M. J. Magazine 1988. "Sequencing of insertions in printed circuit board assembly". Operations Research, 36(2):192-201 p.

Brindle, A. 1981. "Genetic algorithms for function optimization". Tesis de Doctorado, University of Alberta. 234 pp.

Burke, E. K., P. I. Cowling, y R. Keuthen 1999. "New models and heuristics for component placement in printed circuit board assembly". En: "Proceedings of the 1999 IEEE International Conference on Information, Intelligence and Systems", Washington, DC, EU, 31 octubre-3 noviembre. IEEE Press. 133-139 p.

Coello, C. A. 2001. "Introducción a la programación evolutiva (notas de curso)". CINVESTAV. No publicado.

Crama, Y., O. E. Flippo, J. van de Klundert, y F. C. R. Spieksma 1997. "The assembly of printed circuit boards: A case with multiple machines and multiple board types". European Journal of Operational Research, 98(3):457-472 p.

Crama, Y., J. van de Klundert, y F. C. R. Spieksma 2002. "Production planning problems in printed circuit boards assembly". Discrete Applied Mathematics, 123(1-3):339-361 p.

Davis, L. 1985. "Job shop scheduling with genetic algorithms". En: Grefenstette, J. J., editor, "Proceedings of the First International Conference on Genetic Algorithms", Illinois, EU, 24-26 julio. Lawrence Erlbaum Associates. 136-140 p.

De Jong, K. A. 1975. "An analysis of the behavior of a class of genetic adaptive systems". Tesis de Doctorado, University of Michigan. 266 pp.

Dikos, A., P. C. Nelson, T. M. Tirpak, y W. Wang 1997. "Optimization of high-mix printed circuit card assembly using genetic algorithms". Annals of Operations Research, 75(1):303-324 p.

Eiben, A. E. y J. E. Smith 2003. "Introduction to evolutionary computing". Springer. 299 pp.

Ellis, K. P., F. J. Vittes, y J. E. Kobza 2001. "Optimizing the performance of a surface mount placement machine". IEEE Transactions on Electronics Packaging Manufacturing, 24(3):160-170 p.

Falkenauer, E. 1996. "A hybrid grouping genetic algorithm for bin packing". Journal of Heuristics, 2:5-30 p.

García-Nájera, A. y C. A. Brizuela 2005. "PCB assembly: An efficient genetic algorithm for slot assignment and component pick and place sequence problems". En: "Proceedings of the 2005 IEEE Congress on Evolutionary Computation", Edimburgo, Escocia, RU, 2-5 septiembre, volumen 2. IEEE Press. 1485-1491 p.

Gen, M. y R. Cheng 2000. "Genetic algorithms and engineering optimization". John Wiley & Sons, Inc. 512 pp.

Giannessi, F. y F. Tardella 1998. "Connections between nonlinear programming and

discrete optimization". En: Du, D.-Z. y P. M. Pardalos, editores, "Handbook of Combinatorial Optimization", volumen 1. Kluwer Academic Publishers, 149-188 p.

Goldberg, D. E. 1989. "Genetic algorithms in search, optimization, and machine learning". Addison-Wesley. 432 pp.

Günter, R. 1994. "Convergence analysis of canonical genetic algorithms". IEEE Transactions on Neural Networks, 5:98-101 p.

Gutin, G. y A. Punnen 2002. "Traveling salesman problem and its variations". Kluwer Academic Publishers. 848 pp.

Ho, W. y P. Ji 2003. "Component scheduling for chip shooter machines: a hybrid genetic algorithm approach". Computers & Operations Research, 30(14):2175-2189 p.

Ho, W. y P. Ji 2005. "A genetic algorithm to optimise the component placement process in PCB assembly". International Journal of Advanced Manufacturing Technologies. Enero, formato electrónico.

Holland, J. H. 1975. "Adaptation in natural and artificial systems". The University of Michigan Press. 211 pp.

Jeevan, K., A. Parthiban, K. N. Seetharamu, I. A. Azid, y G. A. Quadir 2002. "Optimization of PCB component placement using genetic algorithms". Journal of Electronics Manufacturing, 11(1):69-79 p.

Johnsson, M. y J. Smed 2001. "Observations on PCB assembly optimization". Electronic Packaging and Production, 41(5):38-42 p.

Kumar, R. y H. Li 1995. "Integer programming approach to printed circuit board assembly time optimization". IEEE Transactions on Components, Packaging and Manufacturing Technology - Part B, 18(4):720-727 p.

Kumar, R. y Z. Luo 2003. "Optimizing the operation sequence of a chip placement machine using TSP model". IEEE Transactions on Electronics Packaging Manufacturing, 26(1):14-21 p.

Lee, S., T. Park, B. Lee, W. Kwon, y W. Kwon 1998. "A dynamic programming approach to a reel assignment problem of a surface mounting machine in printed circuit board assembly". En: "Proceedings of the 1998 IEEE International Conference on Robotics and Automation", volumen 1. 227-232 p.

Lee, S. H., J. M. Hong, D. W. Kim, y B. H. Lee 1997. "An effective algorithm for a surface mounting machine in printed circuit board assembly". En: "Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems", volumen 2. 932-937 p.

Lee, W., S. Lee, B. Lee, y Y. Lee 2000. "A genetic optimization approach to operation of a multi-head surface mounting machine". IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E83-A(9):1748-1756 p.

Leipälä, T. y O. Nevalainen 1989. "Optimization of the movements of a component placement machine". European Journal of Operational Research, 38:167-177 p.

Leon, V. J. y B. A. Peters 1998. "A comparison of setup strategies for printed circuit board assembly". Computers and Industrial Engineering, 34(1):219-234 p.

Leu, M. C., H. Wong, y Z. Ji 1993. "Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm". Transactions of the ASME, 115:424-432 p.

Michalewicz, Z. y D. B. Fogel 2000. "How to solve it: Modern heuristics". Springer-Verlag. 467 pp.

Mitchell, J. E., P. M. Pardalos, y M. G. C. Resende 1998. "Interior points methods for combinatorial optimization". En: Du, D.-Z. y P. M. Pardalos, editores, "Handbook of Combinatorial Optimization", volumen 1. Kluwer Academic Publishers, 189-297 p.

Narayanaswami, R. y V. Iyengar 2004. "Setup reduction in printed circuit board assembly by efficient sequencing". International Journal of Advanced Manufacturing Technology, 26(3):276-284 p.

Nemhauser, G. L. y L. A. Wolsey 1999. "Integer and combinatorial optimization". Wiley-Interscience. 763 pp.

Ong, N. y L. P. Khoo 1999. "Genetic algorithm approach in PCB assembly". Integrated Manufacturing Systems, 10(5):256-265 p.

Papadimitriou, C. H. y K. Steiglitz 1998. "Combinatorial optimization: Algorithms and complexity". Dover Publications, Inc. 496 pp.

Rich, E. y K. Knight 1991. "Artificial intelligence". McGraw-Hill, Inc., 2da edición. 640 pp.

Russell, S. J. y P. Norvig 2003. "Artificial intelligence: A modern approach". Pearson Education, Inc., 2da edición. 1132 pp.

Salonen, K., M. Johnsson, J. Smed, T. Johtela, y O. Nevalainen 2000. "A comparison of group and minimum setup strategies in PCB assembly". En: "Proceedings of Group Technology/Cellular Manufacturing World Symposium", San Juan, Puerto Rico, 27-29 marzo, volumen 1. 95-100 p.

Salonen, K., J. Smed, M. Johnsson, y O. Nevalainen 2003. "Job grouping with minimum setup in PCB assembly". En: "Proceedings of Group Technology/Cellular

Manufacturing World Symposium", Columbus, Ohio, EU, 28-31 julio, volumen 1. 221-225 p.

Schrage, L. E. 1984. "Linear, integer and quadratic programming with LINDO: user's manual". Scientific Press. 274 pp.

Smed, J., M. Johnsson, M. Puranen, T. Leipälä, y O. Nevalainen 1998. "Job grouping in surface mounted component printing". Reporte Técnico 196, Turku Centre for Computer Science. 23 pp.

Smed, J., K. Salonen, M. Johnsson, T. Johtela, y O. Nevalainen 2000. "A comparison of group and minimum setup strategies in PCB assembly". Reporte Técnico 327, Turku Centre for Computer Science. 19 pp.

Syswerda, G. 1991. "Schedule optimization using genetic algorithms". En: Davis, L., editor, "Handbook of Genetic Algorithms". Van Nostrand Reinhold, capítulo 21, 332-349 p.

The Math Works 2001. "Optimization toolbox for use with MATLAB: User's guide". Versión 3. 386 pp.

van Laarhoven, P. J. M. y W. H. M. Zijm 1993. "Production preparation and numerical control in PCB assembly". International Journal of Flexible Manufacturing Systems, 5(3):187-207 p.

Wang, W., P. C. Nelson, y T. M. Tirpak 1999. "Optimization of high-speed multi-station SMT placement machines using evolutionary algorithms". IEEE Transactions on Electronics Packaging Manufacturing, 22(2):137-146 p.

Apéndice A

Generación de casos de prueba para el problema 1-1

"Se puede resistir a una invasión de ejércitos, pero no a una idea cuyo tiempo ha llegado."

Victor Hugo (1802 - 1885), Historia de un crimen, 1852.

Los trabajos que se han realizado respecto a la solución de los problemas de manufactura de PCB generalmente usan datos que no están disponibles públicamente, lo que hace difícil la comparación de las nuevas propuestas. Por esta razón, se necesita contar con un conjunto de casos de prueba que estén disponibles para que los algoritmos propuestos se puedan comparar. Con el fin de cumplir con lo anterior, se propone la construcción de estos datos de la manera siguiente.

A.1 Universos de tipos de componente y pinzas

Primero se necesita establecer el universo Ω de $|\Omega|$ tipos de componentes y enumerarlos ($\Omega = \{1, 2, ..., |\Omega|\}$) para crear el conjunto de casos (se pueden crear más de una PCB). Después se debe especificar el universo Θ de $|\Theta|$ herramientas de sujeción y enumerarlas ($\Theta = \{1, 2, ..., |\Theta|\}$). Posteriormente, definir el conjunto de tipos de componentes que cada herramienta puede manipular. Para los casos con los que se trabajó durante el período de tesis se consideraron $|\Omega| = 500$ y $|\Theta| = 100$.

Ya que tenemos los parámetros mencionados en el párrafo anterior, podemos diseñar una PCB, estableciendo el número de componentes que contendrá, así como los tipos de componentes a los que éstos pertenecen.

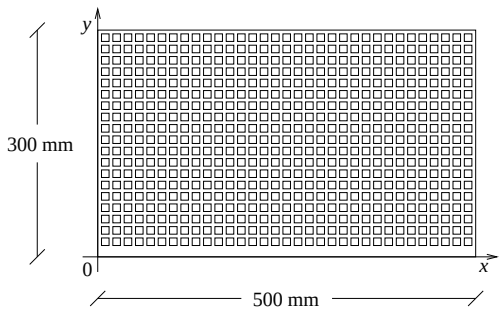


Figura 32. Ejemplo de una PCB con una capacidad máxima de 627 componentes. La PCB puede ser vista como una matriz de 19 renglones y 33 columnas.

A.2 Diseño de una PCB

De acuerdo con las especificaciones de una máquina de un fabricante en específico[1], las dimensiones de una PCB pueden llegar hasta 343×813 mm^2. En los casos generados se utilizaron PCB de 300×500 mm^2. Podemos considerar a una PCB como un rectángulo en el plano euclidiano, con el origen centrado en la esquina inferior izquierda, teniendo las coordenadas máximas x y y en 500 y 300 mm, respectivamente, como se muestra en la Figura 32.

Los productores de PCB dicen que ellos diseñan para manufacturar (Ball y Magazine, 1988), lo que quiere decir:

i. No colocar los componentes más cerca de la distancia mínima específicada entre ellos,

ii. Alinear los componentes a lo largo de una dirección pre-especificada, y

iii. Colocar los componentes a no menos de la distancia especificada de la orilla de la PCB.

Tomando en cuenta las recomendaciones anteriores, en los casos generados se consideraron 1 y 6 mm de separación entre cada par de componentes en las direcciones x y

[1] http://www.apsgold.com/pdf_public/Lseries.pdf.

y, respectivamente, y 5 mm como la distancia mínima especificada desde la orilla de la PCB al primer componente en las direcciones izquierda, derecha, arriba y abajo.

A pesar de los diferentes tipos de componentes, por simplicidad, se consideró un tamaño único para de los componentes, que fue 10×10 mm^2.

El siguiente paso es definir el número de componentes N que contendrá la PCB, el número de tipos de componentes $M \leq |\Theta|$ y el número de componentes de cada tipo $N_i, \forall\ i \in \{1, ..., M\}$. Cada tipo de componente tendrá asignado al menos un componente. Una vez hecha la asignación de un componente a cada tipo, el resto de los componentes se asignan aleatoriamente, con cierta probabilidad, a algún tipo.

Las ubicaciones de los componentes en la PCB pueden ser vistas como elementos (números enteros) de una matriz, que corresponden a los números que identifican a los componentes. Por ejemplo, la PCB en la Figura 32 puede ser representada como una matriz de 19 filas y 33 columnas. Cada elemento de esta matriz corresponde a ciertas coordenadas x y y conocidas, y el componente asignado a un elemento se centra en éstas.

Podemos ahora asignar coordenadas en la PCB a los componentes. Para esto, se enumeran los componentes $(1, 2, ..., N)$ y cada uno de ellos se asigna aleatoriamente a un elemento de la matriz. Se debe tener en cuenta que cada elemento de la matriz puede contener uno o ningún componente asignado y que cada componente se puede asignar solamente a un elemento.

Una vez que tenemos toda la información anterior, podemos seleccionar un conjunto de herramientas $\mathbf{H} \subset \Theta$ que puedan sujetar a todos los tipos de componentes utilizados en la PCB. Debido a que puede haber más de una opción del conjunto de pinzas, mediante un algoritmo voraz, se selecciona el que tenga la cardinalidad mínima (van Laarhoven y Zijm, 1993).

Un caso de ejemplo se muestra en la Tabla XI. Esta PCB contiene 10 componentes

de cuatro tipos diferentes. En este caso, se necesitan sólo dos pinzas para colocar todos los componentes.

Tabla XI. Caso de prueba ejemplo: 10 componentes de cuatro tipos diferentes.

Identificador Componente	Tipo Componente	Identificador Herramienta	Coordenada x	Coordenada y
1	201	12	354.00	251.00
2	201	12	200.00	171.00
3	201	12	365.00	203.00
4	293	38	453.00	107.00
5	293	38	167.00	75.00
6	293	38	57.00	107.00
7	352	12	398.00	251.00
8	427	38	101.00	267.00
9	427	38	343.00	171.00
10	427	38	233.00	235.00

Apéndice B

Datos de los casos de prueba para los problemas 1-1 y V-1

"Podemos tener hechos sin pensar, pero no podemos tener pensamientos sin hechos."

John Dewey (1859 - 1952).

A continuación se muestran los datos de dos casos de prueba para el Problema 1-1. El resto de los casos de prueba está contenido en el disco compacto que viene con este trabajo de tesis. Los casos de prueba para el Problema V-1 se presentan también en este apéndice.

B.1 Casos de prueba para el problema 1-1

La primera columna de las Tablas XII y XIII es el identificador del componente. La segunda es el tipo de componente al cual pertenece. La tercera es el identificador de la herramienta que sujeta al componente. La cuarta y la quinta columnas son las coordenadas x y y de la ubicación del componente en la PCB.

Tabla XII: Datos del caso de prueba (Id. PCB) 1.

Comp.		Id.	Coord.		Comp.		Id.	Coord.		Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y	Id.	Tipo	Herr.	x	y	Id.	Tipo	Herr.	x	y
1	6	14	288	27	39	174	80	321	171	76	309	9	244	267
2	6	14	376	187	40	174	80	299	75	77	309	9	90	91
3	6	14	200	107	41	174	80	156	187	78	309	9	343	43
4	6	14	453	251	42	174	80	277	107	79	309	9	167	267
5	12	3	145	59	43	178	80	475	91	80	361	80	101	91
6	12	3	189	59	44	178	80	420	219	81	361	80	255	91
7	12	3	200	59	45	178	80	266	219	82	361	80	222	107
8	12	3	167	107	46	178	80	321	267	83	361	80	123	283
9	12	3	277	187	47	178	80	222	43	84	361	80	387	251
10	12	3	145	267	48	178	80	35	43	85	361	80	222	267
11	76	80	244	219	49	178	80	354	43	86	361	80	112	251
12	76	80	134	123	50	282	9	409	91	87	439	80	288	123

continua ...

continuación ...

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
13	76	80	46	27
14	76	80	464	91
15	76	80	57	59
16	76	80	387	75
17	76	80	266	59
18	76	80	255	59
19	76	80	101	267
20	96	14	24	155
21	96	14	464	219
22	96	14	288	139
23	96	14	156	267
24	96	14	46	283
25	96	14	211	219
26	96	14	365	43
27	96	14	145	43
28	96	14	178	75
29	105	8	365	219
30	105	8	167	235
31	105	8	266	155
32	105	8	13	251
33	105	8	156	235
34	105	8	420	155
35	105	8	387	107
36	174	80	189	107
37	174	80	178	139
38	174	80	90	107

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
51	282	9	332	75
52	282	9	453	235
53	282	9	486	187
54	282	9	156	251
55	282	9	420	251
56	285	9	387	283
57	285	9	343	203
58	285	9	57	235
59	285	9	178	267
60	285	9	145	27
61	285	9	200	139
62	285	9	332	107
63	285	9	211	91
64	308	80	178	187
65	308	80	277	123
66	308	80	233	27
67	308	80	277	27
68	308	80	167	251
69	308	80	90	27
70	309	9	321	251
71	309	9	365	155
72	309	9	57	155
73	309	9	299	251
74	309	9	101	235
75	309	9	211	203

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
88	439	80	134	187
89	439	80	266	139
90	439	80	90	251
91	439	80	255	251
92	439	80	46	251
93	441	14	464	267
94	441	14	453	11
95	441	14	112	107
96	441	14	46	123
97	441	14	376	155
98	441	14	156	171
99	445	14	376	59
100	445	14	277	11
101	445	14	178	43
102	445	14	233	43
103	445	14	299	235
104	445	14	409	59
105	445	14	277	203
106	445	14	299	91
107	445	14	266	251
108	445	14	123	91
109	461	80	486	251
110	461	80	112	235
111	461	80	123	235
112	461	80	167	171

Tabla XIII: Datos del caso de prueba (Id. PCB) 2.

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
1	26	15	398	251
2	26	15	35	155
3	26	15	464	219
4	26	15	211	107
5	26	15	189	235
6	38	15	178	123
7	38	15	233	203
8	38	15	398	219
9	38	15	464	123
10	38	15	475	27
11	38	15	343	27
12	38	15	398	91
13	38	15	387	75
14	38	15	79	171
15	38	15	343	123
16	64	20	24	155
17	64	20	101	187
18	64	20	277	139
19	64	20	156	283
20	90	51	255	219
21	90	51	167	219
22	90	51	101	155
23	90	51	123	43
24	90	51	79	91
25	119	6	145	187
26	119	6	178	187
27	119	6	442	219
28	119	6	189	107
29	119	6	299	43
30	119	6	376	123

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
43	126	3	112	283
44	191	6	310	123
45	191	6	354	235
46	191	6	475	155
47	191	6	112	155
48	191	6	90	155
49	191	6	255	43
50	203	51	90	171
51	203	51	255	283
52	203	51	79	219
53	203	51	299	155
54	205	19	145	171
55	205	19	464	203
56	205	19	442	123
57	205	19	365	251
58	205	19	398	267
59	205	19	57	251
60	205	19	420	43
61	205	19	486	91
62	205	19	233	59
63	205	19	200	219
64	266	51	266	267
65	266	51	299	107
66	266	51	398	203
67	266	51	266	187
68	266	51	68	203
69	266	51	222	203
70	266	51	178	43
71	266	51	90	27
72	267	6	90	219

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
84	329	6	68	235
85	329	6	112	139
86	329	6	222	75
87	329	6	277	267
88	329	6	431	219
89	329	6	233	91
90	329	6	288	123
91	354	6	189	139
92	354	6	222	251
93	354	6	442	75
94	354	6	244	203
95	354	6	420	187
96	354	6	46	235
97	427	9	68	59
98	427	9	255	27
99	427	9	233	251
100	427	9	409	75
101	427	9	354	27
102	427	9	299	123
103	427	9	35	107
104	427	9	343	283
105	427	9	200	107
106	427	9	409	155
107	431	6	178	59
108	431	6	189	267
109	431	6	365	267
110	431	6	200	91
111	452	20	167	43
112	452	20	376	75
113	452	20	464	171

continua ...

continuación ...

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
31	119	6	420	155
32	119	6	222	43
33	119	6	101	11
34	126	3	233	139
35	126	3	57	43
36	126	3	420	75
37	126	3	134	59
38	126	3	24	187
39	126	3	156	267
40	126	3	486	171
41	126	3	332	219
42	126	3	145	235

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
73	267	6	431	171
74	267	6	123	267
75	267	6	310	75
76	267	6	222	91
77	267	6	112	267
78	267	6	24	219
79	267	6	46	155
80	267	6	409	235
81	267	6	354	91
82	329	6	211	171
83	329	6	145	203

Comp.		Id.	Coord.	
Id.	Tipo	Herr.	x	y
114	452	20	266	91
115	461	3	79	27
116	461	3	200	139
117	461	3	299	171
118	461	3	442	171
119	461	3	57	219
120	461	3	35	203
121	461	3	299	27
122	477	3	475	43
123	477	3	321	139
124	477	3	90	59

La Tabla XIV muestra los datos del caso de Leu *et al.* (1993). La primera columna es el identificador del componente. La segunda es el tipo de componente al cual pertenece. La tercera y la cuarta columnas son las coordenadas x y y de la ubicación del componente en la PCB. En este caso, todos los componentes pueden ser sujetados con una misma herramienta.

Tabla XIV: Datos del caso de prueba de Leu *et al.* (1993).

Comp.		Coord.	
Id.	Tipo	x	y
1	1	100	90
2	1	100	140
3	1	100	190
4	1	120	70
5	1	140	170
6	1	180	50
7	1	180	90
8	1	200	110
9	1	200	140
10	1	200	150
11	1	220	180
12	1	240	50
13	1	240	90
14	1	240	110
15	1	240	220
16	1	100	370
17	1	130	340
18	1	160	300
19	2	100	110
20	2	100	220
21	2	120	80
22	2	120	180
23	2	120	190
24	2	140	210
25	2	160	60
26	2	160	110
27	2	160	140
28	2	180	100
29	2	180	150
30	2	200	50
31	2	220	130
32	2	220	190

Comp.		Coord.	
Id.	Tipo	x	y
68	4	140	230
69	4	160	100
70	4	160	130
71	4	180	110
72	4	180	120
73	4	180	170
74	4	220	80
75	4	240	70
76	4	240	190
77	4	240	230
78	4	130	390
79	4	190	310
80	4	190	320
81	4	190	370
82	4	190	390
83	5	100	70
84	5	100	80
85	5	100	100
86	5	100	230
87	5	120	50
88	5	140	140
89	5	140	200
90	5	160	160
91	5	160	170
92	5	160	180
93	5	160	230
94	5	180	70
95	5	180	180
96	5	180	200
97	5	200	220
98	5	200	230
99	5	220	50

Comp.		Coord.	
Id.	Tipo	x	y
135	7	140	190
136	7	160	90
137	7	180	190
138	7	200	160
139	7	220	70
140	7	240	200
141	7	130	300
142	7	130	360
143	7	190	340
144	8	100	180
145	8	100	210
146	8	120	40
147	8	120	90
148	8	120	110
149	8	120	210
150	8	140	130
151	8	140	180
152	8	160	40
153	8	160	70
154	8	160	80
155	8	180	80
156	8	180	130
157	8	200	40
158	8	200	120
159	8	200	180
160	8	200	190
161	8	220	100
162	8	220	210
163	8	220	230
164	8	240	40
165	8	240	140
166	8	100	330

continua ...

continuación ...

Comp.		Coord.		Comp.		Coord.		Comp.		Coord.	
Id.	Tipo	x	y	Id.	Tipo	x	y	Id.	Tipo	x	y
33	2	220	220	100	5	220	140	167	8	100	350
34	2	240	170	101	5	220	150	168	8	160	380
35	2	100	300	102	5	220	170	169	8	190	350
36	2	100	380	103	5	240	80	170	8	190	360
37	2	100	390	104	5	240	130	171	8	190	380
38	2	130	320	105	6	100	160	172	9	140	110
39	2	130	350	106	6	120	150	173	9	140	220
40	2	160	360	107	6	120	220	174	9	160	50
41	2	160	370	108	6	120	230	175	9	160	150
42	3	100	200	109	6	140	40	176	9	160	210
43	3	120	60	110	6	140	60	177	9	200	90
44	3	120	120	111	6	140	70	178	9	200	170
45	3	120	140	112	6	140	100	179	9	200	200
46	3	140	50	113	6	140	150	180	9	200	210
47	3	140	160	114	6	160	190	181	9	220	200
48	3	180	40	115	6	160	220	182	9	240	180
49	3	180	140	116	6	180	60	183	9	240	210
50	3	180	220	117	6	180	230	184	9	100	360
51	3	200	60	118	6	200	70	185	9	160	310
52	3	240	100	119	6	200	100	186	10	100	40
53	3	240	160	120	6	200	130	187	10	100	150
54	3	100	310	121	6	220	40	188	10	120	100
55	3	100	320	122	6	220	60	189	10	120	160
56	3	130	310	123	6	220	160	190	10	160	120
57	3	130	380	124	6	240	60	191	10	160	200
58	3	160	330	125	6	240	150	192	10	180	160
59	3	160	350	126	6	130	330	193	10	180	210
60	3	160	390	127	6	130	370	194	10	200	80
61	3	190	300	128	6	160	320	195	10	220	90
62	3	190	330	129	7	100	50	196	10	220	110
63	4	100	120	130	7	100	60	197	10	220	120
64	4	100	130	131	7	120	130	198	10	240	120
65	4	100	170	132	7	120	170	199	10	100	340
66	4	140	80	133	7	120	200	200	10	160	340
67	4	140	120	134	7	140	90				

B.2 Casos de prueba para el problema V-1

La Tabla XV muestra los datos del caso de Narayanaswami e Iyengar (2004). La primera columna muestra los tipos de componentes utilizados en los diferentes tipos de PCB. Las columnas segunda (Tipo PCB 1) a la decimotercera (Tipo PCB 12) corresponden a los tipos de PCB. Si en algún renglón de estos tipos de PCB se encuentra el símbolo ✓, quiere decir que el tipo de componente del renglón está contenido en el tipo de PCB. El último renglón de la tabla indica la cantidad de tipos de componentes que contiene el tipo de PCB.

Tabla XV: Datos del caso de prueba de Narayanaswami e Iyengar (2004).

Tipo Comp.	Tipo PCB 1	2	3	4	5	6	7	8	9	10	11	12
1	✓		✓		✓	✓	✓		✓			
2						✓				✓		
3	✓	✓	✓	✓			✓	✓				
4			✓	✓	✓		✓		✓	✓		
5	✓			✓	✓		✓		✓		✓	✓
6	✓									✓	✓	✓
7			✓			✓		✓		✓	✓	
8	✓	✓	✓	✓	✓	✓				✓		✓
9	✓	✓			✓		✓	✓				
10	✓							✓	✓			✓
11	✓							✓		✓	✓	
12		✓			✓		✓	✓		✓		✓
13		✓				✓	✓				✓	✓
14			✓	✓					✓			
15	✓			✓	✓					✓	✓	✓
16	✓				✓		✓	✓		✓		
17			✓	✓	✓		✓					✓
18							✓					
19						✓						
20				✓		✓	✓		✓			✓
21		✓		✓						✓		
22	✓	✓				✓	✓				✓	✓
23		✓						✓		✓	✓	
24							✓	✓	✓		✓	✓
25	✓		✓		✓		✓	✓	✓		✓	✓
26		✓		✓		✓	✓		✓			✓
27			✓			✓						
28		✓	✓	✓			✓		✓	✓		
29	✓	✓	✓	✓		✓		✓	✓			
30		✓	✓	✓			✓	✓		✓		
Tipos Comps.	13	12	12	13	10	11	17	12	11	13	10	13

La Tabla XVI contiene los casos de prueba para el Problema V-1. La primera columna es el identificador del tipo de PCB. Las columnas segunda a la vigésimoprimera corresponden a los casos de prueba. Si en algún renglón de estos casos se encuentra el símbolo ✓, quiere decir que el tipo de PCB del renglón está contenido en el caso de prueba. El último renglón de la tabla indica la cantidad de tipos de PCB que contiene el caso de prueba.

Tabla XVI: Datos de los casos de prueba para el Problema V-1.

Id. PCB	Casos de Prueba para el Problema V-1 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1					✓		✓	✓	✓	✓		✓	✓	✓	✓		✓		✓	✓
2			✓			✓	✓	✓	✓	✓		✓	✓	✓	✓		✓		✓	✓
3			✓	✓	✓			✓		✓		✓	✓		✓	✓	✓		✓	
4		✓		✓		✓			✓	✓		✓	✓		✓	✓	✓	✓	✓	✓
5	✓	✓			✓		✓		✓	✓				✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✓			✓	✓			✓				✓		✓		✓	✓	✓
7	✓		✓	✓	✓			✓		✓		✓	✓	✓		✓		✓	✓	✓
8	✓			✓		✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
9					✓		✓	✓	✓	✓		✓	✓		✓		✓	✓	✓	

continua ...

continuación ...

Id.	Casos de Prueba para el Problema V-1																			
PCB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
10		✓	✓			✓	✓			✓		✓	✓		✓	✓	✓	✓	✓	
11		✓	✓	✓	✓					✓	✓			✓		✓	✓	✓		✓
12		✓		✓		✓			✓	✓	✓			✓	✓		✓	✓		✓
13	✓				✓		✓	✓	✓	✓	✓	✓	✓	✓	✓			✓		✓
14	✓		✓			✓	✓	✓		✓	✓	✓	✓	✓		✓		✓		✓
15	✓		✓	✓	✓			✓		✓	✓	✓	✓			✓	✓	✓		✓
16	✓	✓		✓		✓			✓	✓	✓	✓	✓		✓		✓	✓		✓
17		✓			✓		✓		✓		✓			✓	✓	✓	✓	✓	✓	
18		✓	✓			✓	✓		✓		✓			✓	✓	✓	✓	✓	✓	
19			✓	✓	✓			✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
20				✓		✓		✓	✓		✓	✓	✓	✓				✓	✓	✓
21	✓				✓		✓	✓	✓		✓	✓	✓		✓	✓		✓	✓	✓
22	✓	✓	✓			✓	✓				✓	✓	✓		✓	✓	✓	✓	✓	✓
23	✓	✓	✓	✓	✓						✓		✓	✓	✓		✓	✓	✓	✓
24	✓	✓		✓		✓			✓		✓		✓	✓		✓	✓		✓	
25					✓	✓	✓	✓	✓		✓			✓	✓	✓	✓		✓	✓
26					✓	✓	✓	✓	✓		✓			✓		✓	✓		✓	✓
PCB	12	12	12	12	14	14	14	14	16	16	16	16	18	18	18	18	20	20	20	20

Apéndice C

Problema 1-1 como un caso de programación lineal

"No te preocupes por tus dificultades en las matemáticas. Te puedo asegurar que las mías son aun más grandes."

Albert Einstein (1879 – 1955)

El siguiente modelo del problema 1-1 es una extensión del presentado por Kumar y Li (1995).

Considerese la notación utilizada en el Capítulo II para el problema 1-1.

Sea π_{ijk} una variable de decisión igual a 1 si y sólo si después de colocar el componente i, el brazo mecánico toma el componente k de la ranura j y lo coloca en su posición en la PCB. Entonces, el problema 1-1 puede ser modelado también como:

$$\underset{\pi_{ijk} \in \{0,1\}}{\text{minimizar}} \; S = \sum_{i \leq N} \sum_{j \leq R} \sum_{k \leq N} \left[(t_{ij} h_{ik} + t_{ij}^{CAH}(1 - h_{ik})) + t_{kj} \right] \pi_{ijk} \tag{30}$$

Sujeto a las siguientes restricciones:

$$\sum_{k \leq N} x_{ik} = 1, \quad \forall\, i \in \{1, ..., N\} \tag{31}$$

$$\sum_{i \leq N} x_{ik} = 1, \quad \forall\, k \in \{1, ..., N\} \tag{32}$$

$$\sum_{i \in \mathbf{Q}} \sum_{k \in \mathbf{Q}} x_{ik} \leq |\mathbf{Q}| - 1, \quad \forall\, \mathbf{Q} \subset \{1, ..., N\} \tag{33}$$

$$y_{jk'} = y_{jk}, \quad \forall\, k' \in \tau^{-1}(\tau(k)), \forall\, k \in \{1, ..., N\}, \forall\, j \in \{1, ..., M\} \tag{34}$$

$$\sum_{l\leq M}\left[\frac{1}{|\tau^{-1}(l)|}\sum_{k\in\tau^{-1}(l)} y_{jk}\right]\leq 1, \quad \forall\, j\in\{1,...,M\} \tag{35}$$

$$\pi_{ijk}\in\{0,1\}, \quad \forall\, i,k\in\{1,...,N\}, \forall\, j\in\{1,...,M\} \tag{36}$$

$$h_{ik}\in\{0,1\}, \quad \forall\, i,k\in\{1,...,N\} \tag{37}$$

donde

$$x_{ik} := \sum_{j\leq M}\pi_{ijk}; \qquad y_{jk} := \sum_{i\leq n}\pi_{ijk}$$

Debido a que π_{ijk} representa el número de movimientos, ya sea cero o uno, desde la ubicación del componente i a la ubicación del componente k, pasando por la ranura j, x_{ik} representa el número de movimientos desde la unicación del componente i a la ubicación del componente k, pasando por cualquier ranura. Por lo tanto, $x_{ik}=1$ si y sólo si el componente k es colocado inmediatamente después de colocar al componente i, por lo que deberá satisfacer las restricciones (31)-(33) (que son las mismas restricciones (3)-(5) para el modelo presentado en el Capítulo II).

Las restricciones (34) y (35) son necesarias para garantizar que los componentes del mismo tipo se sujeten de la misma ranura y que cada ranura contenga a lo más un tipo de componente. y_{jk} representa el número total de movimientos desde la ranura j hasta la ubicación del componente k.