UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

UN ALGORITMO GENETICO COOPERATIVO PARA PROBLEMAS DE CORTE Y EMPAQUE

ALVARO FELIPE FLORES JIMENEZ

Profesor Guía: Víctor Parada Daza

Trabajo de Titulación presentado en conformidad a los requisitos para obtener el Título de Ingeniero Civil en Informática.

Santiago, Chile
2012

AGRADECIMIENTOS

Al considerar lo que esto significa no puedo dejar de pensar en los años de estudio y sacrificio, las largas jornadas en la universidad, los trasnoches y los nervios, porque este documento no tan solo corresponde al termino de un trabajo de titulo si no que a la conclusión de un proceso de estudio comenzado el año 2003. Desde entonces a la fecha muchas cosas han cambiado, muchas personas han pasado, pero sus huellas permanecen y es a ellos, todos aquellos que de una u otra forma fueron parte de este proceso a quienes agradezco.

Profesores y compañeros, de quienes aprendí muchísimo, ejemplos de trabajo, disciplina y esfuerzo, muchas gracias.

A mi profesor guía, Dr. Víctor Parada, muchas gracias por su apoyo y ayuda durante este trabajo, y mucho antes, con cuyos ramos me sentí motivado a trabajar y desarrollar una memoria relacionada con estos temas.

A mi familia, sin duda el agradecimiento más grande y profundo, por su ayuda, guía y apoyo durante todos estos años, por haberme enseñado a definir metas y objetivos y trabajar arduamente para lograrlos, por enseñarme que lo importante no es ser mejor que los demás, si no que lo mejor que yo puedo ser, por creer en mi, en mis capacidades y sueños, por estar ahí, gracias, muchas gracias.

A mi novia, quien constantemente me insto y animo a desarrollar este trabajo, por estar ahí y ayudarme a mantenerme enfocado en el objetivo final, muchas gracias.

A Dios, sin duda mi mayor apoyo y sostén.

DEDICATORIA

A mi familia, padres, hermanas y novia.

RESUMEN

Los Algoritmos Genéticos (AG) han mostrado ser bastante competitivos para resolver problemas difíciles de optimización, y por esto se han desarrollado diversos enfoques con el objetivo de mejorar su desempeño, tanto en el tiempo computacional requerido, así como en la calidad de las soluciones encontradas, uno de estos enfoques corresponde a una implementación paralela de los AG (AGP), el que a su vez presenta varias formas y modelos para llevarse acabo, en este trabajo se utilizo una implementación que considera el concepto de cooperación entre los nodos de procesamiento, es decir, el intercambio de información entre nodos con el objetivo de transmitir a todos los nodos los mejores individuos encontrados, estos se conocen como Algoritmos Genéticos Paralelos Cooperativos (AGPC).

La hipótesis planteada es que un enfoque basado en Algoritmo Genético Paralelo Cooperativo tiene mejor desempeño computacional que uno secuencia, no tan solo en términos de tiempo computacional, sino que también en la calidad de la solución, al resolver instancias típicas de prueba de los problemas de empaque y de corte de piezas guillotinable, específicamente el Problema de Empaque de Tiras (PET) y Corte de Piezas Guillotinable Bidimensional Restricto (PCPGBR).

Se obtiene que para 99.4% de las instancias del PET y el 85.4% de las instancias de PCPGBR, el AGPC obtiene un mejor promedio en las ejecuciones realizadas que el AG, por tanto se comprueba que el paralelismo y la cooperación mejoran la calidad de las soluciones obtenidas.

También se obtiene un *Speed Up* sub-lineal para ambos problemas, en específico fue de 3.450 y 2.574 para el PET y el corte de piezas, respectivamente, por lo que se comprueba que el paralelismo y la cooperación mejoran el desempeño del algoritmo en términos de tiempo computacional. Además se provee de un *software web* para dibujar los *layouts* obtenidos para ambos problemas.

Palabras claves: Algoritmo Genético, Algoritmo Genético Paralelo Cooperativo, tiempo computacional, calidad de la solución, Speed Up, layout.

TABLA DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

CAPITULO 1. INTRODUCCIÓN

1.1. ANTECEDENTES Y MOTIVACIÓN

Los problemas de optimización combinatoria están compuestos por un espacio de soluciones, conjunto de restricciones, conjunto de soluciones factibles, una función objetivo y un extremo (mínimo o máximo), y consisten en encontrar la mejor solución al problema (mínimo o máximo), según la función objetivo. Para resolverlos existen diversos enfoques, siendo el más simple probar todas las soluciones factibles y seleccionar la mejor, pero en muchos casos el tamaño del espacio de soluciones factibles es tan grande que probar todas las combinaciones, aun usando computadoras muy potentes, se torna una tarea imposible debido a los altos costos de tiempo, llegando a requerir años para evaluar todas las combinaciones posibles.

Es por esto que se han buscado otras opciones para abordar estos problemas, dentro de ellas se encuentran algunas heurísticas, determinísticas y no determinísticas, así como meta-heurísticas y algoritmos exactos, entre otros. Dentro de los modelos exactos destacan principalmente los basados en ramificación y poda y a nivel de meta-heurísticas están presentes los Algoritmos Genéticos (AG), *Simulated Annealing* (SA), *Tabu Search* (TS), cuya principal característica es que pueden ser adaptados a cualquier tipo de problema combinatorio, por otro lado, las heurísticas, por lo general, están enfocadas en un problema en particular. Ninguno de estos algoritmos asegura encontrar la mejor solución, pero en muchos casos lo hacen y si no, encuentran resultados muy cercanos al optimo.

Los AG están basados en los principios de la evolución planteados por Darwin, considerando procesos tales como la selección natural, el cruzamiento y mutación, para un grupo de individuos (soluciones), los que van evolucionado a medida que pasan las generaciones y por lo tanto buscando la supervivencia de los más fuertes o mejores soluciones. Además de esto se han propuesto una serie de modelos paralelos para mejorar

el funcionamiento de los AG, cada uno con fortalezas y debilidades, así como también se han introducidos conceptos de cooperación, que no son parte de la evolución darwiniana.

Teniendo en mente la complejidad que tienen los problemas de optimización combinatoria, ya sea por las restricciones presentes, el tamaño del espacio de soluciones, entre otros, y la importancia que estos tienen para la vida de las personas, así como las empresas e industrias, su estudio se vuelve primordial y cualquier avance en pos de obtener mejores soluciones, ya sea en cuanto a la calidad o los tiempos requeridos para obtenerla, es importante.

Los problemas de corte y empaque pertenecen a este conjunto de problemas de optimización combinatoria y además pertenecen a la clase NP-Duro, es decir, no existe un algoritmo de orden polinomial capaz de resolver el problema en un tiempo computacional razonable. El problema de corte consiste en ubicar en un área determinada una serie de elementos con el objetivo de minimizar el área sin utilizar y los problemas de empaque consideran una tira de largo infinito en la que se deben ubicar una serie de elementos con el objetivo de minimizar el largo o altura de la tira, teniendo un ancho fijo.

Anteriormente han sido resueltos usando AGs con representaciones enteras y binarias, obteniendo resultados al nivel de los mejores métodos existentes y también se ha encontrado que al sumar paralelismo y cooperación, las soluciones obtenidas por los AGs mejoran en calidad y tiempo computacional, por lo que aplicar estos conceptos a la resolución de estos problemas resulta prometedor, permitiendo evaluar la diferencia entre la calidad de las soluciones obtenidas por ambos enfoques, AGs y AGs Paralelos Cooperativos (AGPC).

1.2. DESCRIPCION DEL PROBLEMA

Los problemas de empaque y corte tienen varias similitudes, entre ellas, el que ambos consisten en la ubicación de piezas con el objetivo de maximizar el uso del espacio o minimizar la pérdida. La principal diferencia está en que para los problemas de corte existe una placa, con un alto y ancho fijo, que establece el límite de área disponible para posicionar las piezas y por tanto, un límite a la cantidad de piezas a ubicar, en cambio para los problemas de empaque de tiras solo se define el ancho, mientras que el alto crece hasta que todas las piezas sean ubicadas en la tira.

Además, existen muchas variantes para cada uno de ellos, es por esto que Dyckhoff en el año 1990 (Dyckhoff 1990) se encarga de agruparlos según sus estructuras lógicas sin considerar las aplicaciones ni disciplinas en que estos se empleen, tal como se puede ver en la Ilustración 1-1, donde se divide inicialmente en 2D y 3D, luego en regulares e irregulares, dentro de los regulares se encuentran las figuras rectangulares y otros tipos, y en los irregulares aparecen los polígonos y otros sin un patrón especifico, finalmente los rectangulares se dividen en no-guillotinables y guillotinables (cortes rectos transversales, verticales u horizontales); y los polígonos en solo convexos y convexos con cóncavos.

En esta memoria de titulo las variantes a estudiar corresponden al problema de empaque, 2D, regular, rectangular, no-guillotinable y el de corte de piezas, 2D, regular, rectangular, guillotinable, según la clasificación presentada en la Ilustración 1-1.

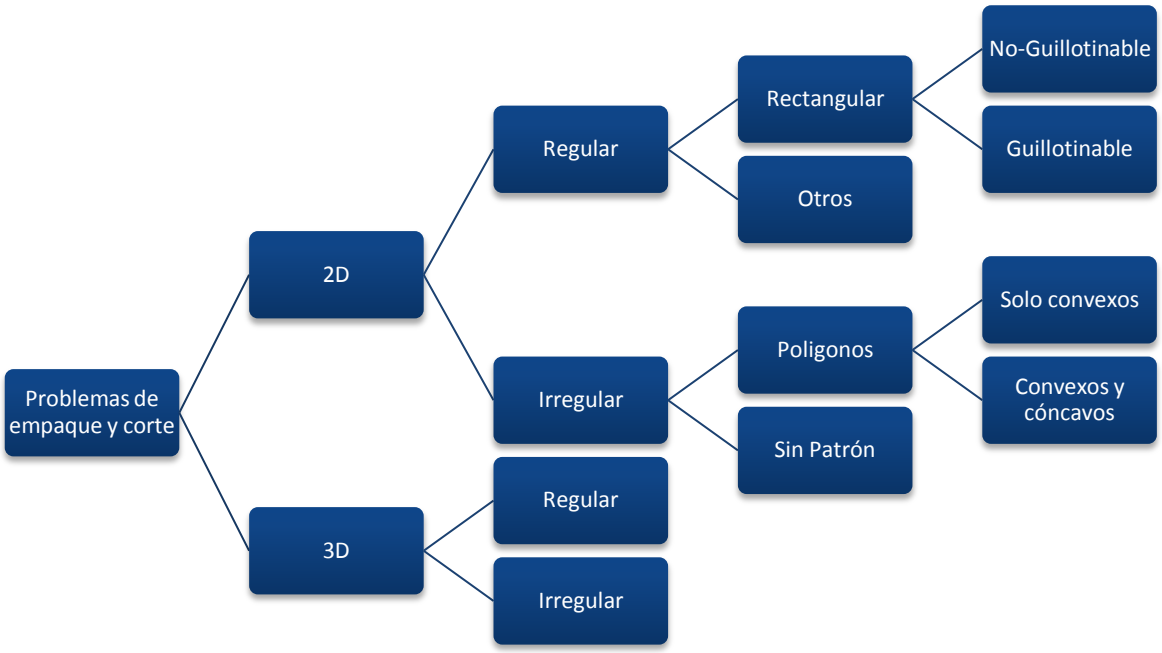


Ilustración 1-1: Clasificación de los problemas de packing y corte (Dyckhoff 1990).

1.2.1. El Problema de Empaque de Tiras (PET)

Este problema se encuentra presente en muchas partes, en industrias tales como la del vidrio, metal, papel, textil, por mencionar algunas, cuyo objetivo en común es ubicar todos los elementos en la tira minimizando su altura o largo.

El problema consiste en un tira S con ancho fijo W y alto variable L, y un conjunto R de piezas con dimensiones (w_i, l_i) tales que $0 \leq w_i \leq W \; o \; l_i \geq 0 \; \forall \; i \in R$. El objetivo, es encontrar un *layout* o esquema con todos los elementos en la tira, ortogonalmente ubicados y sin traslape, minimizando el alto de ésta. La fórmulación matemática es la siguiente.

$$Min\ H \tag{1.1}$$

$s.a.$

$$0 \leq x_i \leq W - w_i, \forall\ i \in R \tag{1.2}$$

$$0 \leq y_i \leq H, \forall\ i \in R \tag{1.3}$$

$$\forall i, j \in R, i \neq j \tag{1.4}$$

$$x_i + w_i \leq x_j \quad \text{ó} \quad x_j + w_j \leq x_i \tag{1.5}$$

$$y_i + h_i \leq y_j \quad \text{ó} \quad y_j + h_j \leq y_i \tag{1.6}$$

$$x_i, y_i \geq 0 \tag{1.7}$$

Donde w_i y h_i corresponden al ancho y alto de la pieza i, (x_i, y_i) coordenadas para la pieza i considerando como punto de origen el vértice inferior izquierdo. Además, las piezas pueden ser rotadas en 90°.

Existen algoritmos exactos capaces de resolver el problema exitosamente para muchas de las instancias comúnmente usadas en la literatura, obteniendo resultados competitivos para instancias pequeñas, sin embargo con un alto costo en tiempo de ejecución (Lesh, Marks, y otros, Exhaustive approaches to 2D rectangular perfect packings 2004). Por otro lado, existen algoritmos heurísticos, principalmente golosos que resuelven el problema obteniendo buenos resultados, pero con el costo asociado implícitamente a los algoritmos golosos. Finalmente los algoritmos meta-heurísticos son los que actualmente han presentado los resultados más competitivos, tanto en la calidad de la solución, como en el tiempo requerido para obtenerlo, por lo que se presentan como el principal camino a seguir a la hora de resolver este problema.

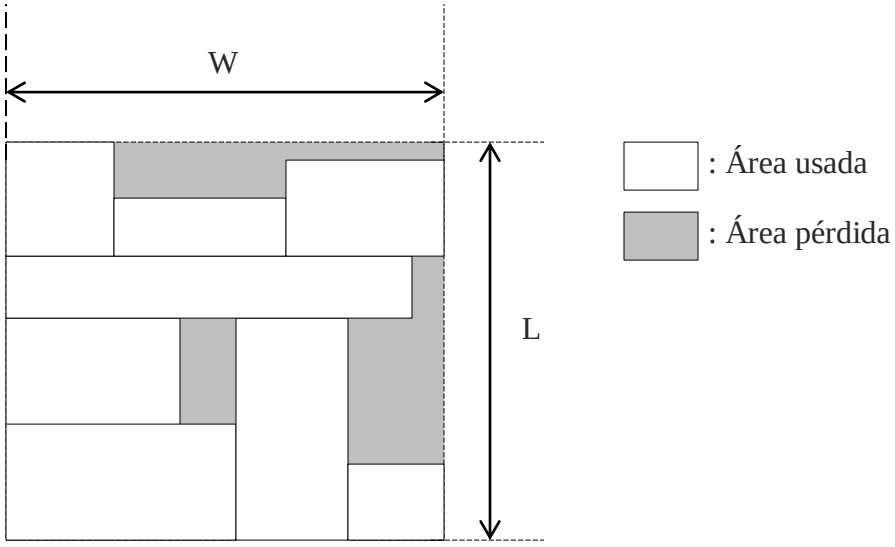


Ilustración 1-2: Layout valido para el problema del PET

1.2.2. El Problema de Corte de Piezas Guillotinable Bidimensional Restricto (PCPGBR)

Se tiene una lámina rectangular S de dimensiones (W, L), siendo W el ancho y L el largo, y un conjunto R de tipos de piezas con dimensiones (w_i, l_i) tales que $0 \leq w_i \leq W$ y $0 \leq l_i \leq L \; \forall \; i \; \in R$. Además, sea un vector b con $b_i, i \in R$, donde b_i define el máximo de veces que un elemento puede estar en la lámina. El objetivo es organizar los elementos en la lámina minimizando la pérdida o área sin usar. La formulación matemática del problema corresponde a la fórmula 2.1.

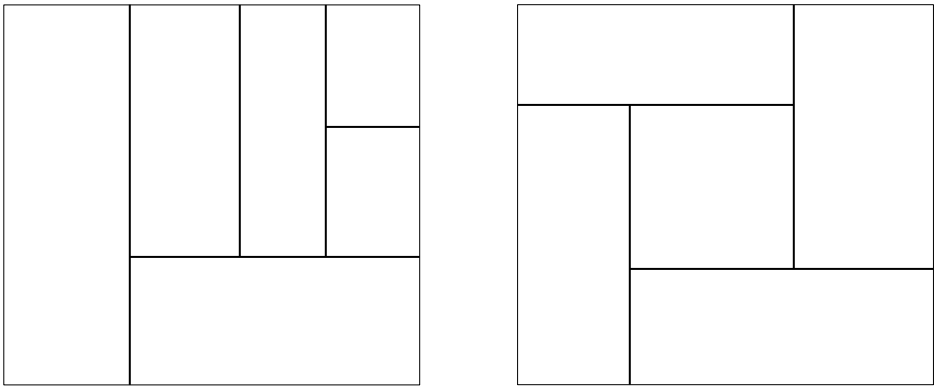
$$Min\, Z(x) = WL - \sum_{i=1}^{|R|} w_i l_i x_i \tag{2.1}$$

$s.a.$

$$0 \leq x_i \leq b_i, \forall \; i \; \in R \tag{2.2}$$

$$x_i \in \mathbb{Z}, \forall \; i \; \in R \tag{2.3}$$

Donde x_i corresponde al número de veces que la pieza tipo i aparece en el patrón de corte. Además, para este problema en particular para que los cortes sean factibles deben ser guillotinables, es decir, debe ser posible realizarlos de lado a lado sin parar, ya sea, horizontal o verticalmente, tal como se puede ver en la Ilustración 1-3.



Corte Guillotinable Corte No-Guillotinable

Ilustración 1-3: Ejemplos de cortes guillotinables y no-guillotinables

Para resolver este problema se han usado una serie de metodologías, algoritmos exactos, de flujo mínimo, constructivos, genéticos, entre otros, los que han sido capaces de resolver el problema satisfactoriamente obteniendo resultados competitivos tanto por su calidad como el tiempo requerido para obtenerla. El algoritmo genético (Tiwari y Chakraborti 2006) mencionado utiliza una representación entera y PMX como operador de

cruzamiento, y obtuvo importantes resultados, por lo que una aproximación empleando una representación binaria resulta claramente interesante.

1.3. SOLUCIÓN PROPUESTA

1.3.1. Características de la solución

La solución propuesta consiste en modelar los problemas del PET y Corte de Piezas Guillotinable Bidimensional Restricto para ser resueltos mediante un Algoritmo Genético (AG) y un Algoritmo Genético Paralelo Cooperativo (AGPC), para esto se utilizara la implementación del AG y AGPC desarrollada por Romero (Romero 2003), ya que esta maneja el concepto de genotipo y fenotipo, permitiendo abstraer la interpretación del cromosoma del núcleo de procesamiento genético. También, es parte de los esfuerzos de este trabajo el utilizar una representación binaria para modelar ambos problemas y agregar elementos para definir el orden de las piezas a la representación del cromosoma, logrando de ese modo que el algoritmo genético sea capaz de generar todas las combinaciones de piezas posibles.

El experimento computacional debe considerar y medir tanto la calidad de la solución como el tiempo computacional requerido para obtenerla, por tanto el *Speed-up* y el desvío respecto al óptimo o mejor solución (*GAP*) son las métricas seleccionadas para medir el desempeño del AG *versus* AGPC.

Para apoyar esto se presentaran tablas y gráficos que muestren los resultados de los algoritmos con cada una de las instancias de prueba.

Las instancias de prueba para el Strip Packing son obtenidas del trabajo de Zhang (Stephen, Zhang y Sim 2011), y para el Problema de Corte de Piezas Guillotinable Bidimensional Restricto, de Sepúlveda (Sepúlveda 2011).

1.3.2. Propósito de la solución

El propósito de la solución es evaluar el desempeño de los AGPC *versus* los AG y de ese modo, determinar cual es el enfoque que resulta ser más competitivo resolviendo las instancias de prueba propuestas para los problemas a resolver. Según los resultados obtenidos por M. Castrillón (Castrillón 2010), C. Toro (Toro 2011), el paralelismo y la cooperación ayudan a mejorar el desempeño de un AG, en términos de calidad, siendo capaces de encontrar mejores soluciones, y tiempo computacional, con un *Speed Up* súper-lineal.

1.4. OBJETIVOS Y ALCANCES DEL PROYECTO

1.4.1. Objetivo General

El objetivo general de esta Memoria de Título es modelar y resolver computacionalmente los problemas de Empaque de Tira y Corte de Piezas Guillotinable Bidimensional Restricto mediante un Algoritmo Genético Paralelo basado en Cooperación.

1.4.2. Objetivos Específicos

i. Modelar los problemas Empaque de Tira y Corte de Piezas Guillotinable Bidimensional Restricto.
ii. Generar una función constructora para los problemas Empaque de Tira y Corte de Piezas Guillotinable Bidimensional Restricto.
iii. Diseñar un experimento computacional para probar la versión paralela y secuencial de los AG según el modelo y función constructora implementados.
iv. Diseñar o generar instancias de prueba para ambos problemas.
v. Realizar el experimento computacional para ambos problemas en ambas versiones.
vi. Analizar los resultados.

1.4.3. Alcances

Los alcances que tiene esta Memoria de Título son modelar los problemas, desarrollar las funciones constructoras correspondientes y utilizar una versión disponible de AGPC para resolver los problemas. Para efectos del experimento computacional, se utilizan instancias conocidas o en su defecto se construyen instancias de prueba equivalentes.

Dentro de las limitaciones del proyecto se tiene el que debe ser desarrollado en C ya que el AGPC esta implementado en ese lenguaje.

1.5. METODOLOGÍAS Y HERRAMIENTAS UTILIZADAS

1.5.1. Metodología a usar

El método científico empírico analítico es la metodología seleccionada para abordar esta Memoria de Titulo, a continuación se describen las etapas para su desarrollo.

- **Estado del arte**: Es necesario recopilar información acerca del problema a resolver y el método seleccionado. El objetivo es lograr un conocimiento acabado del problema.
- **Hipótesis**: Luego de analizar los problemas y los diversos enfoques existentes en la literatura junto con los resultados obtenidos por memorias de título anteriores (Castrillón 2010), (Toro 2011), (Romero 2003), surgen una serie de preguntas, entre ellas si al incluir paralelismo y cooperación a un algoritmo genético se mejora la calidad de las soluciones al resolver los problemas involucrados en esta Memoria de Título y los datos de prueba seleccionados, o si el paralelismo y la cooperación ayudan a mejorar la convergencia del algoritmo genético o el tiempo requerido para su ejecución, entre otras. De este modo se postula la hipótesis a estudiar en esta Memoria de Título: Un enfoque basado en un Algoritmo Genético Paralelo Cooperativo tiene mejor desempeño computacional que uno secuencial, no tan solo en términos de tiempo computacional, sino que también en la calidad de la solución, al resolver

instancias típicas de prueba de los problemas de empaque y de corte de piezas guillotinable.

- **Diseño**: En esta etapa se modela el problema, definiendo el diseño y representación, de modo tal, que pueda ser resuelto usando algoritmo genético.
- **Desarrollo**: Implementación de la función constructora y todo lo necesario para ejecutar el algoritmo genético paralelo basado en cooperación.
- **Experimento Computacional**:
  - Construcción de archivos de prueba.
  - Pruebas del algoritmo que permitan medir y comparar el rendimiento del mismo.
- **Análisis de los resultados obtenidos**: en esta etapa se analizan los resultados obtenidos para contrastarlos con el objetivo e hipótesis del trabajo.

1.5.2. Herramientas de desarrollo

- **Las herramientas de software**:
  - Sistema operativo Ubuntu Server 10.04.4
  - Sistema operativo Windows 7
  - IDE NetBeans 7.1
  - Lenguaje de programación C
  - Office 2010 (Beta gratuita)
  - Algoritmo Genético Paralelo basado en cooperación
  - Condor
  - OpenLDAP
- **Las herramientas de hardware**:
  - Procesador: Intel Core2Duo 2.4 GHz.
  - Memoria RAM: DD3 4 GB.
  - Disco duro: 250 GB.
  - Clúster ubicado en el Departamento de Ingeniería Informática (DIINF), compuesto por 6 servidores SUN FIRE X 2200, con las siguientes características

- Procesador: Dual-Core AMD Opteron Processor 2210 de 1.8 Ghz
- Memoria RAM: 4 GB

1.5.3. Ambiente de desarrollo

Con respecto al ambiente de desarrollo, se necesita un computador, escritorio, Internet y acceso al clúster ubicado en el DIINF junto con un ambiente propicio para la concentración y trabajo, por lo tanto el trabajo podrá ser realizado tanto en las dependencias del DIINF como en la casa del alumno usando una cuenta de acceso VPN para acceder a la red de la universidad.

1.6. ORGANIZACIÓN DEL DOCUMENTO

El documento contempla 5 capítulos además de éste. En el primero se presenta el estado del arte, donde se analizan los últimos avances científicos relacionados con los problemas a estudiar, luego de eso, se presentan los materiales y métodos a emplear indicando el modelamiento de los problemas, los datos de prueba seleccionados, la calibración de parámetros para el AG y AGPC, el diseño experimental y *hardware* utilizado. Luego se presentan los resultados de la experimentación, en el Capítulo 4, y en el Capítulo 5 son analizados. Finalmente en el Capítulo 6 se presentan las conclusiones.

1.7. GLOSARIO

- **Algoritmo Evolutivo (AE)**: Es un método de optimización y búsqueda cuyos fundamentos se encuentran en la evolución de la especies. Considera un conjunto de individuos, donde cada uno representa una solución, se reproducen y compiten entre ellos haciendo que los más aptos prevalezcan y con el paso de las generaciones encontrar al mejor o uno muy cercano (Holland 1975), (Ashlock 2005).

- **Algoritmo Genético (AG)**: Es parte de los AE y se basa en los principios de selección natural. Consiste en una población compuesta por un grupo de individuos, quienes codifican una solución, los que son cruzados y mutados probabilísticamente, buscando que los mejores permanezcan y evolucionen a progresivamente. El objetivo es que mediante la evolución se encuentre la solución óptima o una satisfactoria.
- **Algoritmo Genético Paralelo (AGP)**: Corresponde al enfoque paralelo de un AG. Existen diversas formas de implementarlos, variando su topología, cantidad de procesadores, entre otros. Sus principales ventajas son el hecho de que los AG son naturalmente propensos al paralelismo y que la distribución de la población de individuos en varios subconjuntos ayuda a mejorar la calidad de las soluciones (E. Alba 2005).
- **Algoritmo Genético Paralelo con Cooperación (AGPC)**: es un AGP que introduce el concepto de cooperación entre los diferentes AG. Consiste en compartir individuos con el resto de los AG según las políticas y el porcentaje de emigración y a su vez los AG los incorporan en su población según las políticas y el porcentaje de inmigración, todo esto según la tasa de migración, que indica cada cuantas generaciones se realiza el intercambio. La cooperación puede realizarse de forma síncrona como asíncrona.
- **Heurística Bottom-Left**: Es una heurística que consiste en ubicar las piezas lo más abajo y al izquierda que se pueda, en un rectángulo, es de orden $O(N^2)$, con N igual al número de piezas a ubicar, por lo tanto es una buena opción dada su simplicidad y orden. Su principal desventaja son los espacios que deja a medida que se van ubicando las piezas (Jakobs 1996). El algoritmo es el siguiente:
    i. Ubicar la primera pieza en la esquina inferior izquierda.
    ii. Para las demás piezas comenzar en la esquina superior derecha y alternadamente moverla hacia abajo todo lo que sea posible, luego hacia la izquierda todo lo que sea posible, hasta que ya no pueda realizar más movimientos.
- **Fitness**: Corresponde al valor asociado a un individuo en un AG, es decir, al valor de una solución, éste depende de la interpretación que se realice del individuo, por tanto esta asociado al problema particular que se esté resolviendo.
- **Población de un AG**: Corresponde a un conjunto de individuos que representan soluciones tentativas para un problema en particular. Cada individuo de la población

tiene asociado un fitness. Para pasar de una generación a otra se aplica una serie de operadores, primero el de selección, luego cruzamiento y finalmente mutación, con esto se obtiene una nueva población de individuos.

- **Operador de Selección**: Es un operador de un AG y corresponde al proceso de seleccionar los individuos a cruzar, existen diversas formas tales como torneo, ruleta, ruleta ponderada, aleatoria, entre otros. El objetivo es seleccionar los individuos más competitivos.
- **Operador de Cruzamiento**: Es un operador de un AG que trabaja sobre 2 o más individuos de una población y cruzándolos según la probabilidad de cruzamiento, con el fin de crear nuevos e idealmente mejores individuos. Para esto existen diversas técnicas, entre ellas el cruzamiento de k-puntos y el uniforme (Goldberg 1989).
- **Operador de Mutación**: Es un operador de un AG que modifica aleatoriamente a un individuo, según la probabilidad de mutación. Para esto, se recorre toda la población y cada individuo en detalle, evaluando cada vez si éste debe ser mutado o no, la forma en que se realice la mutación depende de la representación o codificación del cromosoma, en el caso binario, consiste en cambiar un 0 por un 1 o viceversa (Goldberg 1989).
- **Operador de Selección Torneo**: Es un operador de AG que consiste en comparar 2 o más individuos y seleccionar al mejor. En caso de que todos sean igual de buenos, se selecciona el primero.
- **Operador de Cruzamiento de k-puntos**: Es un operador de AG y consiste en dividir el cromosoma de los individuos k veces y luego combinarlos. Los más usados son los de 1 y 2 puntos, en este trabajo se usara el de 1 punto, el cual consiste en dividir los cromosomas en 2 y luego combinar la sección que queda a la izquierda del corte en el primer cromosoma con la que queda a la derecha del corte en el segundo y viceversa.
- **Speed Up**: Es una métrica para medir el desempeño de un algoritmo paralelo. Corresponde a la razón entre el tiempo empleado para resolver el algoritmo usando un procesador y m procesadores, es decir, el *Speed Up* para m procesadores (s_m) es definido por la fórmula 3.1, donde T_1 corresponde al tiempo empleado por un procesador para ejecutar el algoritmo y T_m al tiempo empleado por m procesadores.

$$s_m = \frac{T_1}{T_m} \tag{3.1}$$

En caso de que los algoritmos sean no-determinísticos, se debe usar el promedio de las ejecuciones, por tanto la fórmula para éstos casos es 3.2, donde $E[T_1]$ y $E[T_m]$ corresponden al tiempo de ejecución promedio para uno y m procesadores, respectivamente.

$$s_m = \frac{E[T_1]}{E[T_m]} \tag{3.2}$$

El Speed Up puede ser sub-lineal, lineal y súper-lineal, dependiendo del valor de s_m y m (E. Alba 2005).

i. Si $s_m < m$, entonces es sub-lineal
ii. Si $s_m = m$, entonces es lineal
iii. Si $s_m > m$, entonces es súper-lineal

CAPITULO 2. ESTADO DEL ARTE

A continuación se presentan los principales avances en el estudio de los problemas a abordar en esta Memoria de Título.

2.1. EL PROBLEMA DE EMPAQUE DE TIRAS (PET)

El estado del arte de este problema es bastante variado, existen enfoques exactos, heurísticos y meta-heurísticos, entre otros.

Dentro de los métodos exactos, se tiene a Martello et al. (Martello, Monaci y Vigo 2003), quienes el año 2003 propusieron un algoritmo para resolver el PET 2D sin rotaciones encontrando resultados competitivos para instancias de Hopper (Hopper y Turton 2001) con menos de 30 rectángulos, la clave de este trabajo está en la relajación de las restricciones respecto al área del objeto, dividiéndolo en un conjunto de segmentos, que luego debían ser unidos para formar la solución completa. El principal aporte de este trabajo está en la forma de calcular el límite inferior para el proceso de ramificación y poda. Además, obtuvo alentadores resultados con instancias más grandes, de hasta 200 rectángulos, por lo que con esto se abre un camino para desarrollar algoritmos capaces de resolver instancias aun más grandes.

Lesh et al. (2004) (Lesh, Marks, y otros, Exhaustive approaches to 2D rectangular perfect packings 2004), propusieron otro algoritmo exacto basado en ramificación y poda, la principal idea de éste era buscar la forma de ir agregando elementos sin generar espacios vacíos, en caso de que fuese imposible, se podaba esa rama, por lo que rápidamente se podía ir descartando ramas y seguir las que progresaban en la búsqueda de la solución. Este algoritmo fue capaz de encontrar la solución óptima a varias de las instancias propuestas por Hopper y Turton (Hopper y Turton 2001) con menos de 30 elementos., sin embargo, el tiempo requerido para obtenerlas, en el peor caso fue de hasta 10 minutos para instancias

con 29 elementos, por lo que se pueden mejorar los tiempos requeridos para obtener la solución.

Respecto a los métodos heurísticos, se tienen los basados en *Bottom Left* (BL) y en *Best Fit* (BF). En el año 1980, Baker et al. (Baker, Coffman y Rivest 1980), presentaron la heurística BL que consiste en comenzar desde lo más arriba e ir empujando los elementos hasta lo más abajo y a la izquierda que se pueda. Luego, este método fue mejorado por Chazelle (Chazelle 1983) en el año 1983 y lo llamo *Bottom Left Fit* (BLF), donde cada objeto es ubicado en la posición más baja e izquierda posible. En el año 2001 Hopper y Turton presentaron BLD (Hopper y Turton 2001), basado en BL y que agrega una mejora estratégica respecto al ordenamiento de los elementos usando varios criterios (alto, ancho, perímetro y área) y el algoritmo selecciona el mejor entre éstos. En el año 2005 y 2006, Lesh et al. (Lesh, Marks, y otros, New heuristic and interactive approaches to 2D rectangular strip packing 2005) y (Lesh y Mitzenmacher, Bubble search: a simple heuristic for improving priority-based greedy algorithms 2006), respectivamente, se enfocaron en mejorar el BLD, llamándolo BLD*, para esto crearon listas de objetos de acuerdo a su alto, ancho u otro criterio, también agregaron la posibilidad de que los elementos fuesen rotados, según las siguientes reglas:

i. El algoritmo evalúa ambas orientaciones y selecciona la que permite la posición más abajo a la izquierda.
ii. El algoritmo evalúa ambas orientaciones y selecciona aquella en la que el centro del objeto tiene la posición inferior.
iii. El algoritmo evalúa ambas orientaciones y selecciona aquella que en que la esquina superior derecha queda más abajo.

Según los resultados obtenidos, se concluyo que el criterio que considera la esquina superior derecha es el mejor para decidir si rotar o no y que la mejor forma de ordenar las piezas es según el ancho, de menor a mayor. De los resultados obtenidos se concluyo el método BLD* es el mejor para resolver las instancias más usadas, incluyendo las de Hopper (Hopper y Turton 2001), pero dada su naturaleza golosa, el algoritmo puede consumir grandes recursos y tiempo para encontrar las soluciones, por lo que abre una brecha de desarrollo para los algoritmos meta heurísticos, donde una de sus principales

características es que visitan una ínfima porción del espacio de soluciones para encontrar la mejor o una buena solución.

Respecto a la heurística BF, el año 2004, Burke et al. (Burke, Kendall y Whitwell, A new placement heuristic for the orthogonal stock-cutting problem 2004), presentaron un ordenamiento dinámico para los rectángulos a ser ubicados, éste consiste en una búsqueda desde la posición más abajo a la izquierda disponible y va seleccionado la pieza que mejor queda, si existe. Este enfoque es usado ampliamente en meta-heurísticas hibridas.

Finalmente se tienen los enfoques meta-heurísticos, los que principalmente usan *Tabu Search* (TS), *Simulated Annealing* (SA) y Algoritmos Genéticos (AG) en conjunto con alguna heurística.

En el año 2006, Burke et al. (Burke, Kendall y Whitwell, Metaheuristic enhancements of the best-fit heuristic for the orthe orthogonal stock-cutting problem 2006), proponen un AG hibrido con SA y TS combinados con BF y BLF. Para esto las soluciones se generan en 2 pasos.

i. Asignar una cantidad inicial de piezas usando BF, estas quedan establecidas de forma inalterable.
ii. Asignar el resto de las piezas usando BLF en combinación con AG, SA y TS respectivamente, es decir, se asignan primero con BLF y luego se mejora esa distribución usando las 3 heurísticas por separado.

De este trabajo se concluyo que SA obtuvo resultados más competitivos al compararlo con AG y TS, para la mayoría de las instancias probadas.

En el año 2009, Burke et al. (Burke, Kendall y Whitwell, A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock-Cutting Problem 2009), presentan una mejora al trabajo del año 2006 (Burke, Kendall y Whitwell, Metaheuristic enhancements of the best-fit heuristic for the orthe orthogonal stock-cutting problem 2006), donde la solución se genera en las mismas 2 etapas, pero solo considera SA en la segunda.

En el año 2006, Soke y Bingul (Soke y Bingul 2006), presentan un trabajo que usa AG y SA en conjunto con BLF, es decir, AG con BLF y SA con BLF, ambos, tienen como objetivo encontrar el mejor orden para los elementos a ubicar en la tira, sin rotaciones. El trabajo se basa en comparar ambas alternativas, en el caso de AG el parámetro a analizar fue el operador de cruzamiento y en SA, el de enfriamiento. Sin embargo, este trabajo solo

consideró las instancias con menos de 30 rectángulos, propuestas por Hopper (Hopper y Turton 2001), por lo que su desempeño con instancias más grandes es desconocido.

El año 2006, Bortfeldt (Bortfeldt 2006), introduce un algoritmo genético llamado *Strip Packing Genetic Algorithm Layer* (SPGAL), usa la heurística *Best Fit Decreasing Height* (BFDH) (Mumford-Valenzuela, Vick y Wang 2003) con algunas mejoras para generar la población inicial. Considera una estructura de capas, tomando en cuenta la restricción de corte guillotina. Luego el AG realiza la búsqueda directamente en esta estructura de capas, por otro lado, cuando los problemas no requieren la restricción de corte guillotina, se realiza una post-optimización para romper la estructura de capas. Este enfoque encontró mejores soluciones que todos los demás enfoques existentes en esa época y que consideraran la rotación de las piezas.

En el año 2008 se publico un articulo en el que Alvarez-Valdés et al. (Alvarez-Valdés, Parreño y Tamarit 2008), proponen un *Greedy Randomized Adaptive Search Procedure* (GRASP), que se divide en 2 etapas, la primera ubica las piezas usando un procedimiento aleatorio BF, en la segunda la solución es pasada a una búsqueda de vecindades variable, cuya solución solo se conserva si es mejor que la anterior.

En el año 2011, Burke et al. (Burke, Kendall y Hyde, A squeaky wheel optimisation methodology for two-dimensional strip packing 2011), presentan una metodología de *packing* iterativa basada en optimización por *squeaky wheel*. Esta consiste en que luego de cada *packing* completo (iteración) se penalizan aquellas piezas que degradaron directamente la calidad de la solución, de ese modo, en la próxima iteración, las piezas penalizadas son puestas antes. Este método obtuvo mejores resultados que el propuesto el año 2009 (Burke, Kendall y Whitwell, A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock-Cutting Problem 2009) y en algunas instancias fue capas de encontrar el óptimo, además mostro ser competitivo con GRASP (Alvarez-Valdés, Parreño y Tamarit 2008) y mucho más simple de implementar.

El mismo año Defu Zhang et al. (Stephen, Zhang y Sim 2011), presentan un enfoque basado en 2 etapas, en la primera un algoritmo heurístico basado en una simple regla de puntaje que se encarga de seleccionar un rectángulo dentro de todos los disponibles según un determinado espacio. En la segunda etapa búsqueda local y SA son combinados para mejorar la solución al problema, búsqueda local se encarga de encontrar una buena solución

y SA aumenta su capacidad de búsqueda. Luego de los experimentos computacionales el algoritmo encontró mejores soluciones que las existentes hasta ese entonces en la literatura, superando los resultados de GRASP (Alvarez-Valdés, Parreño y Tamarit 2008) y se desempeño mejor con problemas grandes.

Luego de revisar los avances realizados usando algoritmos meta-heurísticos se puede ver que obtienen resultados mucho más competitivos, aun con instancias grandes, por lo que es un buen camino a seguir para resolver este problema.

2.2. EL PROBLEMA DE CORTE DE PIEZAS GUILLOTINABLE BIDIMENSIONAL RESTRICTO (PCPGBR)

El estado del arte de este problema es similar al del PET, ya que tienen muchos aspectos en común, por tanto sus estudios han seguido las mismas líneas.

En el año 2006 fue publicado un artículo por Mhand Hifi y Rym M'Hallah (Hifi y M'Hallah 2006), que resuelve el problema en varias etapas, comenzando con una adaptación del problema de la mochila para generar tiras con elementos, luego combinan las tiras generadas usando un proceso de llenado, encontrando de ese modo mejores combinaciones de elementos en la tira, finalmente mejoran la solución usando la estrategia *hill-climbing*, que se basa en una búsqueda local, haciendo que el algoritmo se adapte mejor a problemas grandes. Luego de los experimentos computacionales concluyeron que el algoritmo obtiene buenas soluciones en tiempos computacionales razonables.

También en el año 2006, S. Tiwari y N. Chakraborti (Tiwari y Chakraborti 2006) utilizan un algoritmo genético para enfrentar el problema, con 2 objetivos, minimizar la pérdida y la cantidad de cortes a realizar para obtener las piezas, además se considero una versión con cortes guillotinables y otra no-guillotinables. La representación del gen se puede ver como un árbol - de forma única - leído desde las hojas, de izquierda a derecha y con notación postfija, donde las piezas se encuentran en las hojas, identificadas con un número, y los niveles superiores representan el alineamiento que tienen los nodos hijos, H (horizontal) o V (vertical), y una sección binaria que indica si la pieza esta girada o no. Para

la mutación, se seleccionan 2 posiciones en el gen y luego se intercambian, para así obtener un gen valido, para la sección binaria, simplemente se alterna su valor. El cruzamiento se realiza usando PMX en la parte numérica del gen. El algoritmo obtuvo resultados competitivos en las instancias evaluadas, por lo que resulta interesante evaluar el desempeño de un algoritmo genético con una representación binaria.

El año 2010, Cláudio Alves et al (Macedo, Alves y Valério de Carvalho 2010), proponen un modelo exacto usando *Arc-Flow*, programación lineal entera, formulándolo como un problema de flujo mínimo. Este trabajo se presenta como una extensión del realizado por Valério (Valério de Carvalho 1999), en el que propone un modelo de *Arc-Flow* para el problema de corte unidimensional, donde cada camino del grafo dirigido acíclico corresponde a un patrón de corte. Para extender el modelo de una dimensión a 2 dimensiones, lo que hicieron fue manejar el problema bidimensional como 2 de una dimensión. El modelo originalmente considera una pila donde se van ubicando los elementos, para el caso bidimensional cada pila corresponde a una tira horizontal, en la primera etapa se define el alto de la tira y en la segunda se ubican los elementos en ella, para seleccionar los elementos que pueden ser parte de cada tira se eliminan aquellos cuyo alto sea mayor al definido en la tira, luego la unión de las tiras produce la solución final. Luego de probar el algoritmo con varias instancias se obtuvieron resultados competitivos en comparación con otros enfoques presentes en la literatura y con mejores tiempos de ejecución.

Otro trabajo publicado el mismo año (2010), de Charalambous y Fleszar (Charalambous y Fleszar 2011), presenta una nueva heurística constructiva, para esto utilizan un algoritmo recursivo que va llenando sub-rectángulos usando el modo *first fit* hasta que no se puedan agregar más elementos, esto constituye un patrón, para cada rectángulo se elaboran muchos patrones, seleccionando el mejor usando un criterio de suficiencia del área promedio. Para elaborar los patrones, los elementos son ordenados de forma no creciente, usando la siguiente fórmula $\lambda h_i + (1-\lambda)a_i$, donde h_i y a_i corresponden a la altura y área del elemento i respectivamente y λ es un parámetro que se mueve entre 0 y 1, entonces cuando es 0, se ordenan por área y cuando es 1, por altura, así se obtiene un patrón por cada ordenamiento, luego de esto se retorna el mejor patrón de

todos los evaluados para el rectángulo, para evaluar se considera el layout completo y los patrones correspondientes al rectángulo en cuestión.

El criterio del área promedio considera el área promedio de la lista de elementos ($\bar{a}(I)$) a ubicar y el área promedio del patrón a evaluar ($\bar{a}(P)$), el principio se satisface si $\bar{a}(P) \geq \bar{a}(I)$, en caso de evaluar 2 patrones, si ambos satisfacen el criterio, entonces se opta por el que tiene la mayor suma de área de los elementos contenidos, si solo uno satisface el criterio, entonces se opta por ese, y en caso de que ninguno lo haga, se opta por el que este menos lejos del área promedio de la lista de elementos, con este criterio buscan posicionar los elementos grandes primero, en relación a los existentes en la lista.

Finalmente, de este trabajo se concluyo que las principales fortalezas son el desarrollar muchos patrones simples y con la forma en que se selecciona el mejor, el criterio del área promedio, se evita el sobreuso de las piezas fáciles de posicionar en el comienzo de la construcción de la solución. Además se considero un sesgo con el objetivo de relajar o restringir el criterio de suficiencia para algunas instancias, obteniendo mejores resultados y una post-optimización mejoro aún más las soluciones. Finalmente la combinación de la heurística constructiva con el sesgo y la post-optimización obtuvo los mejores resultados a la fecha de su publicación para las instancias utilizadas, respecto a la calidad de la solución como el tiempo computacional empleado, según lo planteado por sus autores.

CAPITULO 3. MATERIALES Y MÉTODOS

Para verificar nuestra hipótesis y determinar si un enfoque paralelo cooperativo permite obtener resultados más competitivos que su versión secuencial, es necesario diseñar un experimento computacional con el objetivo de evaluar su desempeño y comparar los resultados obtenidos por ambos, para esto, se seleccionaron instancias de prueba ampliamente utilizadas en la literatura. El valor a comparar corresponde al *fitness* de la función, que para el caso del PET corresponde al alto de la tira y para el PCPGBR, corresponde a la pérdida o área sin usar. El objetivo de ambos problemas es minimizar el *fitness*, por lo tanto, mientras menor sea, mejor es la solución.

3.1. ALGORITMO GENÉTICO PARALELO

Los AG paralelos nacen como una forma de mejorar el desempeño de los AG secuenciales, además de presentarse como solución a muchos problemas identificados al usar AGS (Nowostawski y Poli 1999).

Los AG son altamente paralelizables debido a su funcionamiento y parámetros requeridos. Además, presentan mejoras en los tiempos y esfuerzos requeridos para ejecutar el AG, aun cuando el paralelismo sea simulado en una maquina mono-procesador (Alba y J. 2000).

Dentro de las principales ventajas de los AGP se encuentran las siguientes.

- **Búsqueda descentralizada**, al dividirse la carga en varias sub-algoritmos estos recorren el espacio de soluciones de forma independiente, permitiendo que se sigan varias direcciones de búsqueda.
- **Diversidad**, al tener varias direcciones de búsqueda, aumenta la diversidad de las soluciones ya que corresponden a diferentes espacios del universo de soluciones.

- **Intensa exploración**, tal como se mencionó en los puntos anteriores, al tener varios sub-algoritmos recorriendo el espacio de soluciones, la exploración realizada es mucho más intensa ya que se actúa en varios sectores y al mismo tiempo.
- **Evolución independiente**, cada población evoluciona dentro del sub-algoritmo, por tanto la exploración que este pueda hacer del algoritmo es independiente a los demás.

Existen diversas formas de paralelizar un AG (E. Alba 2005), sin embargo el modelo que ha sido seleccionado para esta Memoria de Título corresponde al propuesto por Romero (Romero 2003), que considera *n* nodos AG con subpoblaciones independientes y un coordinador, e introduce el conceptos de cooperación, dado que según, una tasa de migración, los nodos envían sus mejores individuos al coordinador, quien se encarga de seleccionar los mejores y enviarlos a los nodos, según un porcentaje de migración, tal como muestra la Ilustración 3-1.

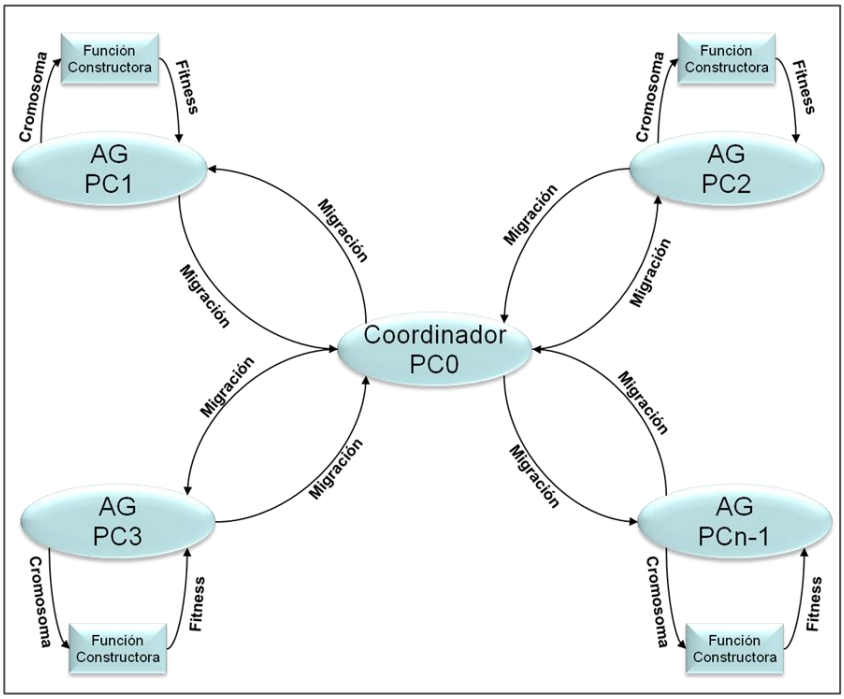


Ilustración 3-1: Modelo de AGPC propuesto por Romero (Romero 2003)

3.2. MODELAMIENTO DEL PROBLEMA

3.2.1. Modelamiento para el PET

Para resolver el problema se utiliza un *string* binario como cromosoma en el AGPC, con un largo igual a $n + 5$, donde n es la cantidad de piezas a ubicar en el tira y 5 bits reservados que sirven para interpretar el *string* en la función objetivo y calcular su *fitness*. En la Ilustración 3-2 se puede ver como se distribuyen las posiciones dentro del *string* y a qué corresponden.

Además, existe un cromosoma encargado de indicar si la pieza va rotada o no, este es el cromosoma de rotación, cuyo largo es n y cada posición representa a una pieza, si el valor es 1 entonces la pieza conserva su orientación original, si no, entonces se rota en 90 grados.

Operadores genéticos

Los operadores genéticos y el tipo de funcionamiento seleccionado para la implementación del AGPC son:

- **Selección**: Torneo con 2 individuos seleccionados al azar.
- **Cruzamiento**: Binario con un punto de corte.
- **Mutación**: binaria, indica si una pieza va rotada o no.

Función objetivo

La función objetivo se divide en 2 partes, la primera, consiste en obtener el orden en que se deben agregar las piezas en la tira y la segunda, en insertar las piezas en la tira usando heurística *Bottom-Left*.

Para obtener el orden de las piezas a ingresar en la tira, el *string* debe ser decodificado, para esto se define el valor, dirección y salto, tal como se muestra en la Ilustración 3-2, a continuación se detalla el significado de cada uno.

- **Valor**: es binario y corresponde al valor que debe tener la posición del string para que la pieza, en cuestión, sea agregada al listado ordenado. Se invierte en cada pasada al *string*, ya sea de izquierda a derecha o derecha a izquierda.

- **Dirección**: es binario e indica en que dirección se comienza a recorrer la sección del *string* correspondiente al orden de las piezas. Si es uno se recorre de izquierda a derecha y luego de derecha a izquierda, si es 0 de derecha a izquierda y luego de izquierda a derecha. Se invierte luego de cada vuelta al *string*, es decir, de izquierda a derecha y derecha a izquierda y, viceversa.
- **Salto**: es un número binario codificado en 3 bits, por tanto, puede obtener valores entre 0 y 7 y, define cuántas posiciones se saltará para recorrer el *string* que contiene el orden de las piezas, a este valor se le suma 1 para que comience en 1 y termine en 8.

Para obtener el orden en que las piezas deben ser agregadas en la tira se aplica el siguiente procedimiento.

OrdenaPiezas(Cromosoma)

1. **Mientras** (existen piezas sin ordenar) **hacer** {
2. **Si** (Dirección == 0) **entonces** {
3. SaltoAux ← -Salto
4. i ← n
5. } **Si no** {
6. SaltoAux ← Salto
7. i ← 1
8. }
9. **Mientras** (i <=n y i>0) **hacer** {
10. **Si** (Cromosoma[i] pertenece al listado ordenado) **entonces** {
11. i se desplaza una posición según la dirección en que se esté recorriendo el *string*
12. } **Si no** {
13. **Si** (Cromosoma[i] == Inicio) **entonces** {
14. Se agrega i al listado ordenado
15. i ← i + SaltoAux
16. }
17. }

18. }
19. Se invierte el valor de Dirección
20. **Si** (se completo un ciclo) **entonces** se invierte el valor de Valor
21.}
22 **Retorna** listado ordenado

Una vez obtenido el orden de las piezas, se aplica la heurística *Bottom-Left* para insertarlas en la tira, junto con el cromosoma de rotación. Finalmente el valor del *fitness* corresponde a la altura máxima de la tira, luego de insertar todas las piezas.

En la Ilustración 3-3 se muestra las 2 partes de la función objetivo, en la primera, se obtiene el orden en que deben ser ubicadas las piezas y en la segunda, se obtiene el *layout* luego de ubicarlas usando la heurística *Bottom-Left*, asumiendo que el cromosoma de rotación indica que ninguna pieza debe ser rotada.

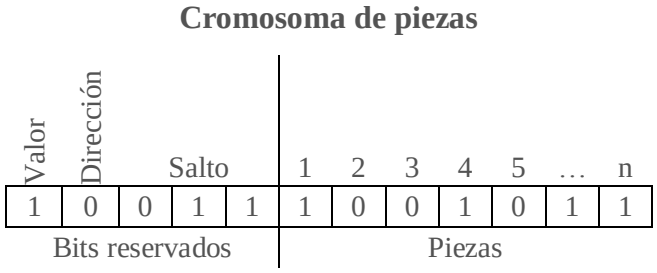


Ilustración 3-2: Cromosoma binario para representar una solución al problema del PET

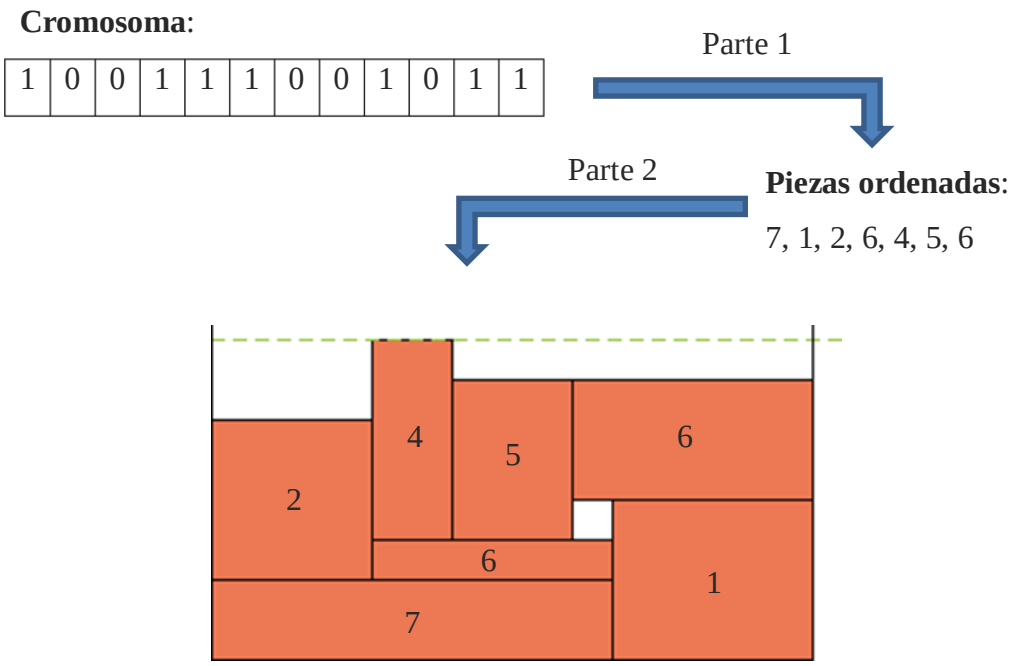


Ilustración 3-3: Diagrama y resultados al aplicar la función objetivo a cromosoma de la Ilustración 3-2

3.2.2. Modelamiento para el PCPGBR

Para resolver el problema se utiliza la solución propuesta por Romero (Romero 2003), con la diferencia que se agrega una etapa al principio para obtener el orden en que deben ser ingresadas las piezas en la placa, esto es, 2 cromosomas, ambos *strings*, el primero corresponde al cromosoma de las piezas y el segundo al cromosoma de rotación.

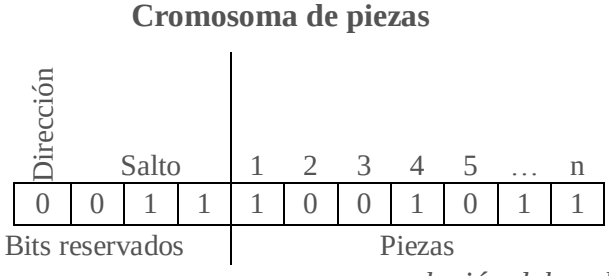
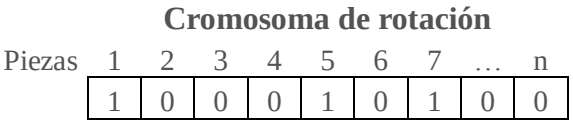


Ilustración 3-4: Cromosoma que representa una solución del problema de corte



1: la pieza va rotada en 90°
0: la pieza no va rotada

Ilustración 3-5: Cromosomas binarios para representar la solución al problema de corte de piezas guillotinable bidimensional restricto.

Como se puede ver en la Ilustración 3-4 el largo de los cromosomas es $n + 4$, donde n es igual al número de piezas a ubicar en la plancha, y cada posición en el string representa una pieza, como el problema corresponde al corte de piezas guillotinable restricto, una pieza se puede repetir una cierta cantidad de veces, por lo que la cantidad de piezas corresponde a la suma de la cantidad de veces que cada pieza puede estar en el *layout*. Para el caso del cromosoma de piezas un "1" indica que la pieza debe ir en el *layout*, por el contrario, el "0" indica que la pieza no es parte de ese *layout*. Para el caso del cromosoma de rotación, el "1" indica que la pieza va rotada en 90°, en cambio, el "0" indica que la pieza conserva su orientación original. Al igual que para el PET se agregan algunos bits (4) para obtener el orden en que deben ser ingresadas las piezas en la plancha, el funcionamiento es el mismo, con la diferencia que para el corte de piezas no se considera en el cromosoma el valor con que se debe comenzar la búsqueda, si no que se define de forma constante como 1 y no varia al leer el cromosoma en sentido contrario, para de ese modo obtener todas las piezas cuyo valor de la posición en el cromosoma sea 1, es decir, indique que debe ser parte del *layout*.

Operadores genéticos

Los operadores genéticos y el tipo de funcionamiento seleccionado para la implementación del AGPC, son los siguientes:

- **Selección**: Torneo con 2 individuos seleccionados al azar.
- **Cruzamiento**: Binario de dos puntos.
- **Mutación**: binaria, modifica un valor del cromosoma de piezas, haciendo que la pieza en cuestión se agregue o elimine del *layout*.

Función objetivo

La función objetivo se divide en 2 partes, la primera es, obtener las piezas y el orden en que se deben ingresar en el *layout* y la segunda es, crear el layout usando la heurística Combinación Heurística VH (Vertical-Horizontal) (Romero 2003) y así obtener el *fitness*.

Las piezas a incluir en el *layout* se almacenan en un arreglo denominado “Arreglo Temporal”. Para crear este arreglo se utiliza el cromosoma de piezas y rotación, es decir para cada pieza se analiza el valor binario correspondiente en los cromosomas.

- Se recorre el cromosoma según como lo indiquen los bits reservados (dirección y salto), si el valor en la posición es 1, entonces la pieza se agrega al “Arreglo Temporal”.
    - Si el valor del cromosoma de rotación para la misma pieza es 1, entonces la pieza se rota en 90°.
    - Si el valor de cromosoma de rotación para la misma pieza es 0, entonces la pieza permanece igual.
- Si el valor en el cromosoma de piezas es 0, entonces, se agrega la pieza con dimensiones 0x0 al “Arreglo Temporal” ya que no debe ser considerada en el layout.

Luego para calcular el *fitness* se utiliza la heurística denominada Combinación Heurística VH (Romero 2003), cuyo funcionamiento se basa en combinaciones verticales y horizontales, a continuación se explican ambas.

- **Combinación Vertical**: consiste en ubicar una pieza debajo de un patrón existente, creando un nuevo patrón de corte, tal como muestra la Ilustración 3-6.
- **Combinación Horizontal**: consiste en ubicar una pieza a la derecha de un patrón existente, creando un nuevo patrón de corte, tal como muestra la Ilustración 3-6.

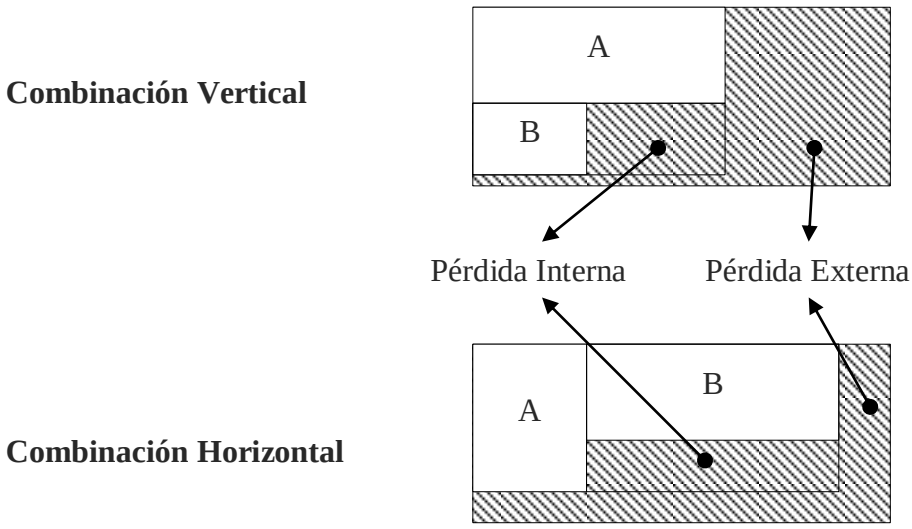


Ilustración 3-6: Combinación Vertical y Horizontal de las piezas A y B.

Los pasos que describen el algoritmo de posicionamiento Combinación Heurística VH son los siguientes:

- Recorrer el "Arreglo Temporal" de izquierda a derecha y solo considerar las piezas con área mayor a 0.
- Ubicar la primera pieza factible en la esquina superior izquierda de la lámina.
- Con la siguiente pieza factible, intentar realizar una Combinación Vertical con ambas piezas, si el nuevo patrón de corte no cabe en la lámina, entonces, intentar una Combinación Horizontal, si nuevamente el patrón no cabe en la lamina, entonces, la pieza se descarta y se pasa a la siguiente hasta introducir el nuevo patrón de corte.
- Para el resto de las piezas se realiza lo siguiente:
    - Si la pieza cabe dentro de alguna región[1] correspondiente una pérdida producida en una etapa anterior se reemplaza aquella región por un nuevo patrón de corte.

[1] Esta región se conoce como Pérdida Interna. Esta pérdida es generada por el Patrón de Corte, a diferencia de la Pérdida Externa que corresponde a la superficie de la Lámina que queda fuera de cualquier Patrón de Corte.

- Si no, se intenta unir el patrón anteriormente generado con la nueva pieza, utilizando Combinación Vertical u Horizontal dependiendo de las dimensiones de la placa, creándose un nuevo patrón.

- Los criterios de término son:
  - Si se han considerado todas las piezas del "Arreglo Temporal".
  - Si al incluir más piezas, no se pueden generar más patrones, remplazando las pérdidas existentes o agregando nuevos.

En la Ilustración 3-7, se puede ver el funcionamiento de la heurística empleada, donde en el primer paso se ubica la primera pieza en la esquina superior izquierda, en el segundo, una Combinación Vertical entre ambas piezas, en el tercero, una Combinación Horizontal y en el cuarto, se inserta dentro de una perdida interna, con esto se obtiene el *layout* y el *fitness*, cuyo valor corresponde a la suma de todas las perdidas internas y externas.

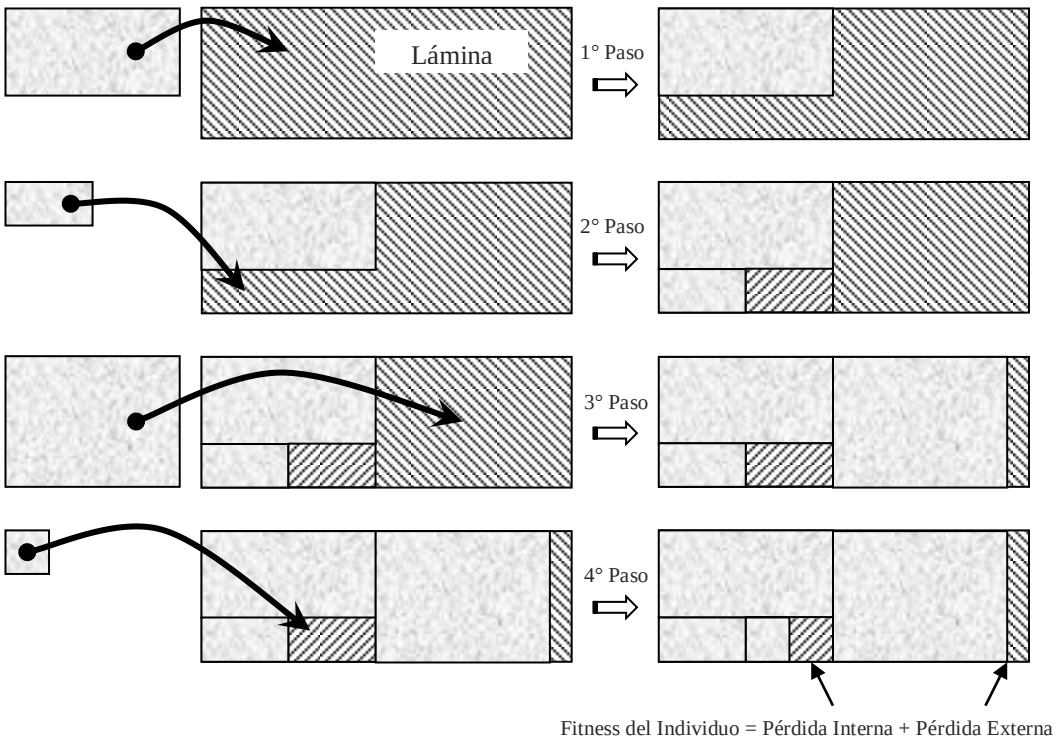


Ilustración 3-7: Aplicación del algoritmo Combinación Heurística VH (Romero 2003)

3.3. DATOS DE PRUEBA

A continuación se indican las instancias que se utilizan para probar el desempeño del algoritmo, en ambos casos se realizan 10 pruebas con cada instancia y se selecciona el mejor resultado (*Best*) y el promedio de las 10 ejecuciones (*Mean*).

3.3.1. Datos de prueba para PET

Las instancias de prueba a utilizar en este trabajo fueron obtenidas del trabajo publicado por Zhang et al. (Stephen, Zhang y Sim 2011), las que han sido usadas ampliamente en el mundo científico, para poder comparar los resultados con los algoritmos más competitivos en la actualidad. Los grupos de instancias seleccionados son los siguientes:

- **C**: 21 instancias de problema generadas por Hopper (Hopper y Turton 2001)
- **N**: 14 instancias de problema generadas por Burke (Burke, Kendall y Whitwell, A new placement heuristic for the orthogonal stock-cutting problem 2004)
- **BWMV**: 500 instancias de problema, de C01 a C06 fueron generadas por Berkey (Berkey y Wang 1987) y de C7 a C10 por Martello (Lodi, Martello y Vigo 1999).

Las instancias C y N son instancias sin pérdida y con óptimos conocidos. La instancia BWMV tiene pérdida y las soluciones óptimas, por lo tanto, tienen áreas de pérdida, algunos óptimos son conocidos, en el caso de los desconocidos se tiene la cota inferior (LB), en la Tabla 1 y Tabla 2 se presenta un resumen de cada grupo de instancias, indicando el número de piezas (n), ancho de la tira (W) y mejor resultado conocido u optimo (LB) para cada una de las instancias. Además, los resultados obtenidos para cada una de las instancias serán comparados con los obtenidos por BSA (Burke, Kendall y Whitwell, A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock-Cutting Problem 2009), SVC (Belov, Scheithauer y Mukhacheva 2008), GRASP (Alvarez-Valdés, Parreño y Tamarit 2008) e ISA (Stephen, Zhang y Sim 2011), para las mismas instancias.

Tabla 1: Resumen del grupo de instancias C y N

Instancia	***n***	***W***	***LB***	**Instancia**	***n***	***W***	***LB***
C11	16	20	20	**N1**	10	40	40
C12	17	20	20	**N2**	20	30	50
C13	16	20	20	**N3**	30	30	50

C21	25	40	15	**N4**	40	80	80
C22	25	40	15	**N5**	50	100	100
C23	25	40	15	**N6**	60	50	100
C31	28	60	30	**N7**	70	80	100
C32	29	60	30	**N8**	80	100	80
C33	28	60	30	**N9**	100	50	150
C41	49	60	60	**N10**	200	70	150
C42	49	60	60	**N11**	300	70	150
C43	49	60	60	**N12**	500	100	300
C51	73	60	90	**N13**	3152	640	960
C52	73	60	90				
C53	73	60	90				
C61	97	80	120				
C62	97	80	120				
C63	97	80	120				
C71	196	160	240				
C72	197	160	240				
C73	196	160	240				

Tabla 2: Resumen de grupo de instancias BWMV

Instancia	***n***	***W***	***LB***	**Instancia**	***n***	***W***	***LB***
C01	20	10	60,3	**C06**	20	300	159,9
	40	10	121,6		40	300	323,5
	60	10	187,4		60	300	505,1
	80	10	262,2		80	300	699,7
	100	10	304,4		100	300	843,8
C02	20	30	19,7	**C07**	20	100	490,4
	40	30	39,1		40	100	1049,7
	60	30	60,1		60	100	1515,9
	80	30	83,2		80	100	2206,1
	100	30	100,5		100	100	2627
C03	20	40	157,4	**C08**	20	100	434,6
	40	40	328,8		40	100	922
	60	40	500		60	100	1360,9
	80	40	701,7		80	100	1909,3
	100	40	832,7		100	100	2362,8
C04	20	100	61,4	**C09**	20	100	1106,8
	40	100	123,9		40	100	2189,2
	60	100	193		60	100	3410,4
	80	100	267,2		80	100	4578,6
	100	100	322		100	100	5430,5
C05	20	100	512,2	**C10**	20	100	337,8
	40	100	1053,8		40	100	642,8
	60	100	1614		60	100	911,1
	80	100	2268,4		80	100	1177,6
	100	100	2617,4		100	100	1476,5

3.3.2. Datos de prueba para PCPGBR

La instancias de prueba a utilizar con este problema son las mismas utilizadas por Sepúlveda (Sepúlveda 2011) que a su vez corresponden a las de Álvarez-Valdés (Álvarez-Valdés, Parajón y Tamarit 2002), solo se consideran aquellas con óptimos conocidos y que se aplican al problema de corte guillotina restricto, y no ponderadas. Los grupos de instancias son los siguientes.

- **GT1**: 13 instancias, algunas extraídas de la publicación de Fayard (1998) (Fayard, Hifi y Zissimopoulos 1998) y otras generadas aleatoriamente.
- **GT2**: 25 instancias, algunas de la publicación de Cung (2000) (Cung, Hifi y Cun 2000) y otras generadas aleatoriamente.
- **GT3**: 10 instancias generadas aleatoriamente.

En la Tabla 3, Tabla 4 y Tabla 5, se presenta un resumen de cada grupo de instancias de prueba donde se indica el ancho (W) y largo (L) de la plancha, junto con el número de tipos de piezas (NTP), número de piezas totales (NP), ancho máximo de las piezas (MW), alto máximo de las piezas (ML), el promedio de repetición de las piezas (AR), porcentaje de piezas con un área mayor a un octavo de la plancha (PG), optimo y pérdida.

Tabla 3: Resumen del grupo de instancias GT1

Instancia	W	L	NTP	NP	MW	ML	AR	PG	Óptimo	Pérdida
OF1	70	40	10	23	36	22	2,3	56,50	2737	63
W	70	40	20	62	33	23	3,1	80,60	2721	79
CU1	100	125	25	82	75	38	3,3	31,70	12330	170
CU2	150	175	35	90	102	80	2,6	48,90	26100	150
CU3	134	125	45	158	71	66	3,5	41,10	16723	27
CU4	285	354	45	113	206	120	2,5	46,00	99495	1395
CU5	456	385	50	120	236	292	2,4	62,50	173364	2196
CU6	356	447	45	124	269	189	2,8	38,70	158572	560
CU7	563	458	25	56	217	267	2,2	62,50	247150	10704
CU8	587	756	35	78	466	344	2,2	69,20	433331	10441
CU9	856	785	25	76	319	434	3	67,10	657055	14905
CU10	794	985	40	129	364	429	3,2	30,20	773772	8318
CU11	977	953	50	134	364	429	2,7	20,10	924696	6385

Tabla 4: Resumen del grupo de instancias GT3, el guion (-) indica que la instancia no tiene el valor correspondiente

Instancia	W	L	NTP	NP	MW	ML	AR	PG	Óptimo	Pérdida
APT30	927	152	38	192	50	328	5,1	2,10	140904	0
APT31	856	964	51	258	280	265	5,1	1,20	822393	2791
APT32	307	124	56	249	24	120	4,4	-	38068	0
APT33	241	983	44	224	357	82	5,1	1,80	236611	292
APT34	795	456	27	130	150	308	4,8	2,30	360084	2436
APT35	960	649	29	153	138	216	5,3	-	620797	2243
APT36	537	244	28	153	79	175	5,5	2,60	130744	284
APT37	440	881	43	222	119	175	5,2	-	387276	364
APT38	731	358	40	202	118	287	5,1	2,00	261154	544
APT39	538	501	33	163	142	154	4,9	-	268750	788

Tabla 5: Resumen del grupo de instancias GT2, el guion (-) indica que la instancia no tiene el valor correspondiente

Instancia	W	L	NTP	NP	MW	ML	AR	PG	Óptimo	Pérdida
P2s	40	70	10	23	8	13	2,3	8,70	2778	22
P3s	40	70	20	62	25	31	3,1	80,60	2721	79
A1s	50	60	20	62	25	31	3,1	75,80	2950	50
A2s	60	60	20	53	29	33	2,7	64,20	3535	65
STS2s	55	85	30	78	43	31	2,6	50,00	4653	22
STS4s	99	99	20	50	34	44	2,5	12,00	9770	31
CHL1s	132	100	30	63	41	33	2,1	7,90	13099	101
CHL2s	62	55	10	19	16	23	1,9	31,60	3279	131
CHL3s	157	121	15	35	24	29	2,3	-	7402	11595
CHL4s	207	231	15	27	24	31	1,8	-	13932	33885
A3	70	80	20	46	35	31	2,3	52,20	5451	149
A4	90	70	20	35	25	31	1,8	37,10	6179	121
A5	132	100	20	45	53	49	2,3	11,10	12985	215
CHL5	20	20	10	18	7	7	1,8	55,60	390	10
CHL6	130	130	30	65	41	33	2,2	12,30	16869	31
CHL7	130	130	35	75	34	42	2,1	-	16881	19
Hchl1	130	130	30	65	41	33	2,2	12,30	11303	5597
Hchl2	130	130	35	75	34	42	2,1	-	9954	6946
Hchl3s	127	98	10	51	23	22	5,1	-	12215	231
Hchl4s	127	98	10	32	23	22	3,2	-	11994	452
Hchl5s	205	223	25	60	56	62	2,4	-	45361	354
Hchl6s	253	244	22	60	86	103	2,7	5,00	61040	692
Hchl7s	263	241	40	90	56	108	2,3	3,30	63112	271
Hchl8s	49	20	10	18	7	7	1,8	-	911	69
Hchl9	65	76	35	76	21	31	2,2	39,50	5240	-300

3.4. CALIBRACIÓN DE PARAMETROS

Para calibrar los parámetros de los AG y AGPC se utilizo el software de parametrización ParamILS en su versión 2.3 (Hutter, y otros 2009). Los parámetros a calibrar para el AG son la probabilidad de cruzamiento y mutación y el tamaño de la población, para el AGPC son la probabilidad de cruzamiento y mutación, tamaño de la población, tasa de migración, tipo de migración (síncrona o asíncrona) y los porcentajes de envío y recepción de individuos en cada migración. Estos parámetros fueron calculados una vez y empleados en todas las instancias de prueba.

ParamILS es una herramienta para la optimización de parámetros, para esto el usuario debe proveer 3 elementos, un algoritmo parametrizable, todos los parámetros con sus valores posibles y una instancia de prueba para ser resuelta por el algoritmo. Lo que hace es ejecutar el algoritmo con diferentes configuraciones de parámetros y resolver la instancia de prueba buscando la configuración con mejor rendimiento global y así obtener la mejor combinación de parámetros (Hutter, y otros 2009).

En primera instancia, se calibra la versión secuencial del algoritmo, es decir, el AG, obteniéndose la probabilidad de cruzamiento y mutación y, el tamaño de la población, con estos parámetros definidos se calibra el AGPC y de ese modo obtener los valores para el resto de los parámetros, para el AGPC el tamaño de la población es igual al numero de nodos AG multiplicado por el tamaño de la población encontrado para el AG.

Para ambos problemas los valores posibles de los parámetros a calibrar son los mismos y se encuentran en la Tabla 6.

Tabla 6: Valores posibles para cada parámetro a ser calibrado por ParamILS, el guion (-) indica que esos parámetros no aplican al problema o que fueron parametrizados en una etapa anterior

Parámetro	AG	AGPC
Probabilidad de cruzamiento	0,6; 0,7; 0,8; 0,9 y 1,0	-
Probabilidad de mutación	0,1; 0,2; 0,3; 0,4 y 0,5	-
Tamaño de la población	10, 20, 50, 100, 150, 200, 250, 300, 400, 500	-
Tasa de migración	-	10, 20, 30, 40, 50, 100
Tipo de migración	-	Síncrona, Asíncrona
Porcentaje de envío	-	10%, 12%, 15%, 20%
Porcentaje de recepción	-	10%, 12%, 15%, 20%

Los parámetros obtenidos para el AG de ambos problemas son los presentados en la Tabla 7 y para el AGPC, en la Tabla 8.

Tabla 7: Valores para los parámetros del AG para PET y PCPGBR, obtenidos por ParamILS

Parámetro	PET	PCPGBR
Probabilidad de cruzamiento	0,9	0,9
Probabilidad de mutación	0,1	0,1
Tamaño de la población	250	250

Tabla 8: Valores para los parámetros del AGPC para PET y PCPGBR, obtenidos por ParamILS

Parámetro	PET	PCPGBR
Tasa de migración	10	10
Tipo de migración	Asíncrona	Asíncrona
Porcentaje de envío	10%	10%
Porcentaje de recepción	10%	10%

3.5. DISEÑO EXPERIMENTAL

Para verificar la hipótesis es necesario analizar la calidad de las soluciones obtenidas por ambos algoritmos para PET y PCPGBR, es decir, obtener el fitness para cada una de las instancias y, para analizar los tiempos de ejecución se debe tomar el tiempo que requieren ambos algoritmos para resolver las instancias de prueba.

Como los algoritmos son no-determinísticos, cada algoritmo es ejecutado 10 veces para cada instancia con semillas aleatorias y cada resultado se registra. Las métricas seleccionadas para evaluar el desempeño de los algoritmos son *Best*, *Mean*, GAP y *Speed Up*, *Best* corresponde a la mejor solución (Mejor *fitness*) encontrada por el algoritmo en alguna de las 10 ejecuciones, *Mean* corresponde al promedio de las soluciones obtenidas en las 10 ejecuciones, GAP corresponde a la desviación del *Mean* respecto del mejor valor conocido para la instancia y el *Speed Up* corresponde a la razón entre el promedio de tiempo de las ejecuciones secuenciales y paralelas.

La primera etapa del experimento, luego de la calibración de parámetros, consiste en ejecutar el AG y AGPC para el PET con los parámetros encontrados por ParamILS (Hutter, y otros 2009), lo mismo para el PCPGBR. Para cada ejecución, el algoritmo entrega la

mejor solución encontrada y el tiempo computacional requerido para encontrarla, ambos valores se registran y tabulan para calcular las métricas mencionadas, además el algoritmo indica el cromosoma y *layout* correspondiente a la solución entregada, junto con esto se registra la mejor solución de cada generación en el AG y para el AGPC, la mejor solución de cada generación en cada nodo AG.

La segunda etapa, corresponde al cálculo de las métricas usando los valores obtenidos en la etapa anterior, con esto se obtiene el *Best*, *Mean* y GAP para cada instancia. Respecto al *Speed Up*, solo se consideran los resultados obtenidos por AG y AGPC para los grupos de instancias C y GT1, del PET y PCPGBR respectivamente, los valores de T_1 y T_m corresponde al promedio de tiempo requerido por el AG y AGPC para resolver cada grupo de instancias, siendo ejecutado 10 veces para cada instancia. En el caso de AGPC el tiempo corresponde a la ejecución completa del algoritmo, incluyendo los tiempos de comunicación entre los nodos AG y el coordinador.

3.6. HARDWARE Y SOFTWARE UTILIZADO

El hardware y software empleado para la ejecución de los AG y AGPC corresponde al *cluster* ubicado en la Universidad de Santiago de Chile, en el departamento de Ingeniería Informática.

El *cluster* está compuesto por 6 SUN FIRE X2200, con las siguientes características cada uno:

- Hardware
  - Procesador: Dual-Core AMD Opteron Processor 2210
  - Arquitectura del procesador: x86_64.
  - Velocidad del Procesador: 1.8 Ghz
  - Cantidad de núcleos: 2.
  - Memoria (RAM): 4 GB.
- Software
  - Sistema Operativo: Ubuntu server 10.04.4
  - Condor

- OpenLDAP

Por lo tanto se cuenta con un total de 12 núcleos y 24 GB de RAM distribuidos en 6 maquinas.

CAPITULO 4. RESULTADOS

A continuación se presentan los resultados obtenidos por cada algoritmo para el PET y PCPGBR.

Los AG y AGPC fueron ejecutados 10 por cada instancia y se presenta el promedio de las ejecuciones (*Mean*) y la mejor ejecución (*Best*).

4.1. RESULTADOS PARA EL PET

Los parámetros empleados para resolver este problema corresponden a los encontrados mediante la calibración de ParamILS (Hutter, y otros 2009), y pueden ser vistos en la Tabla 9.

Tabla 9: Parámetros para el AG y AGPC, para el PET, calibrados por ParamILS

Parámetro	AG	AGPC
Probabilidad de cruzamiento	0,9	0,9
Probabilidad de mutación	0,1	0,1
Tamaño de la población	250	250 x 4 = 1000
Tasa de migración	-	10
Tipo de migración	-	Asíncrona
Porcentaje de envío	-	10%
Porcentaje de recepción	-	10%

Para el AGPC se emplearon 4 nodos para ejecutar los AG y un nodo coordinador, por esto la población total es de 1000 individuos ya que se divide de forma uniforme en los 4 nodos.

Las tablas 10, 11 y 12, muestran los resultados obtenidos por AGPC y AG para las instancias C, N y BWMV respectivamente, junto con los resultados obtenidos por BSA (Burke, Kendall y Whitwell, A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock-Cutting Problem 2009), SVC (Belov, Scheithauer y Mukhacheva

2008), GRASP (Alvarez-Valdés, Parreño y Tamarit 2008) e ISA (Stephen, Zhang y Sim 2011), y el óptimo o mejor resultado conocido (LB) con el fin de comparar el desempeño, además de eso se muestra el margen de error (%GAP).

Para el AGPC y AG se presentan 2 resultados, el mejor (*Best*, abreviado con una B) y el promedio de las 10 ejecuciones (*Mean*, abreviado con una M). Para el caso del grupo de instancias BWMV, se tiene que se subdividen en 10 grupos, los que a su vez están subdivididos en 5 subconjuntos, según la cantidad de piezas que contienen y cada una de estas subdivisiones contempla 10 instancias distintas, por ejemplo, en el subgrupo C01 existen 50 instancias, 10 por cada subgrupo según el número de piezas, por lo tanto los valores presentados en la tabla corresponden al promedio de los resultados obtenidos para las cada una de las 10 instancias (*Best*, abreviado con una B) y el promedio de 100 ejecuciones realizadas en total (*Mean*, abreviado con una M), y esto por cada línea de la Tabla 12.

Para cada tabla la primera columna corresponde a la instancia, con la cantidad de piezas que tiene (n), ancho de la tira (W), mejor valor conocido u óptimo (LB), luego cada columna corresponde a un algoritmo para los que se indica el *Best* (B) y *Mean* (M), en algunos casos solo el *Best* y otros solo *Mean*. La última columna corresponde al margen de error, la que se subdivide en cada algoritmo indicando el error porcentual para cada instancia (%GAP).

Tabla 10: Resultados para el grupo de instancias C, n, W y LB corresponden a la cantidad de piezas, ancho y óptimo de cada instancia, respectivamente. B y M, corresponden al Best y Mean obtenido por cada algoritmo

Instancia				**BSA**	**SVC**	**GRASP**		**ISA**		**AGPC**		**AG**		**%GAP**					
	n	W	LB	B	M	B	M	B	M	B	M	B	M	BSA	SVC	GRASP	ISA	AGPC	AG
C11	16	20	20	**20**	**20**	**20**	**20**	**20**	**20,0**	21	21	21	21,3	**0,0**	**0,0**	**0,0**	**0,0**	5,0	6,5
C12	17	20	20	**20**	21	**20**	**20**	**20**	**20,0**	22	22	21	21,9	**0,0**	5,0	**0,0**	**0,0**	10,0	9,5
C13	16	20	20	**20**	**20**	**20**	**20**	**20**	**20,0**	21	21	20	21,1	**0,0**	**0,0**	**0,0**	**0,0**	5,0	5,5
C21	25	40	15	16	**15**	**15**	**15**	**15**	**15,0**	16	16	16	16,3	6,7	**0,0**	**0,0**	**0,0**	6,7	8,7
C22	25	40	15	16	**15**	**15**	**15**	**15**	**15,0**	16	16	16	16,9	6,7	**0,0**	**0,0**	**0,0**	6,7	12,7
C23	25	40	15	16	**15**	**15**	**15**	**15**	**15,0**	16	16	16	16,6	6,7	**0,0**	**0,0**	**0,0**	6,7	10,7
C31	28	60	30	31	**30**	**30**	**30**	**30**	**30,0**	32	32	33	33,0	3,3	**0,0**	**0,0**	**0,0**	6,7	10,0
C32	29	60	30	**31**	**31**	**31**	**31**	**31**	**31,0**	33	33	33	33,4	**3,3**	**3,3**	**3,3**	**3,3**	10,0	11,3
C33	28	60	30	31	**30**	**30**	**30**	**30**	**30,0**	33	33	32	33,0	3,3	**0,0**	**0,0**	**0,0**	10,0	10,0
C41	49	60	60	**61**	**61**	**61**	**61**	**61**	**61,0**	66	66	65	67,0	**1,7**	**1,7**	**1,7**	**1,7**	10,0	11,7
C42	49	60	60	**61**	**61**	**61**	**61**	**61**	**61,0**	66	66	66	67,0	**1,7**	**1,7**	**1,7**	**1,7**	10,0	11,7
C43	49	60	60	61	61	61	61	**60**	**60,9**	66	66	65	66,0	1,7	1,7	1,7	**1,5**	10,0	10,0
C51	73	60	90	**91**	**91**	**91**	**91**	**91**	**91,0**	100	100	98	100,2	1,1	**1,1**	**1,1**	**1,1**	11,1	11,3
C52	73	60	90	91	91	91	91	**90**	**90,8**	99	99	101	102,4	1,1	1,1	1,1	**0,9**	10,0	13,8
C53	73	60	90	92	**91**	**91**	**91**	**91**	**91,0**	100	100	100	101,7	2,2	**1,1**	**1,1**	**1,1**	11,1	13,0
C61	97	80	120	122	**121**	122	122	**121**	**121,0**	135	135	137	139,6	1,7	**0,8**	1,7	**0,8**	12,5	16,3
C62	97	80	120	**121**	**121**	**121**	**121**	**121**	**121,0**	135	135	134	136,6	**0,8**	**0,8**	**0,8**	**0,8**	12,5	13,8
C63	97	80	120	122	**121**	122	122	**121**	**121,0**	134	134	136	138,0	1,7	**0,8**	1,7	**0,8**	11,7	15,0
C71	196	160	240	244	**242**	244	244	**242**	**242,0**	286	286	290	293,7	1,7	**0,8**	1,7	**0,8**	19,2	22,4
C72	197	160	240	244	242	243	243	**241**	**241,0**	283	283	285	292,7	1,7	**0,8**	1,3	**0,4**	17,9	22,0
C73	196	160	240	245	**242**	243	243	**242**	**242,0**	289	289	290	296,3	2,1	**0,8**	1,3	**0,8**	20,4	23,5
Promedio				**83,62**	**82,95**	**83,19**	**83,19**	82,76	82,84	**93,76**	**93,76**	**94,05**	**95,94**	**2,33**	**1,03**	**0,95**	0,76	**10,62**	**12,82**

Tabla 11: Resultados para el grupo de instancias N, n, W y LB corresponden a la cantidad de piezas, ancho y óptimo de cada instancia, respectivamente. B y M, corresponden al Best y Mean obtenido por cada algoritmo

Instancia				**BSA**		**SVC**		**GRASP**		**ISA**		**AGPC**		**%GAP**					
	n	W	LB	B	M	B	M	B	M	B	M	B	M	BSA	SVC	GRASP	ISA	AGPC	AG
N1	10	40	40	**40**	**40**	**40**	**40**	**40**	**40,0**	**40**	**40**	40	41,3	**0,0**	**0,0**	**0,0**	**0,0**	**0,0**	3,2
N2	20	30	50	**50**	**50**	**50**	**50**	**50**	**50,0**	52	52	53	53,9	**0,0**	**0,0**	**0,0**	**0,0**	4,0	7,8
N3	30	30	50	51	51	51	51	**50**	**50,1**	53	53,9	53	53,9	2,0	2,0	2,0	**0,2**	7,8	7,8
N4	40	80	80	82	82	81	82	**80**	**80,0**	86	86	85	86,2	2,5	2,5	2,5	**0,0**	7,5	7,8
N5	50	100	100	103	**101**	102	102	**101**	**101,0**	111	111	110	112,8	3,0	1,0	2,0	**1,0**	11,0	12,8
N6	60	50	100	102	101	101	101	**100**	**100,9**	108	108	108	110,5	2,0	1,0	1,0	**0,9**	8,0	10,5
N7	70	80	100	104	101	101	101	**100**	**100,0**	109	109	111	113,5	4,0	1,0	1,0	**0,0**	9,0	13,5
N8	80	100	80	82	**81**	**81**	**81**	**81**	**81,0**	88	88	88	92	2,5	**1,3**	**1,3**	**1,3**	10,0	15,0
N9	100	50	150	152	151	151	151	**150**	**150,9**	169	169	168	172,9	1,3	0,7	0,7	**0,6**	12,7	15,3
N10	200	70	150	152	151	151	151	**150**	**150,8**	167	167	168	171,9	1,3	0,7	0,7	**0,5**	11,3	14,6
N11	300	70	150	153	151	151	151	**150**	**150,7**	182	182	185	192,6	2,0	0,7	0,7	**0,5**	21,3	28,4
N12	500	100	300	306	301	304	304	**301**	**301,0**	379	379	378	392,6	2,0	0,3	1,3	**0,3**	26,3	30,9
N13	3152	640	960	964	963	965	965	**960**	**960,0**	1739	1739	1726	1747	0,4	0,3	0,5	**0,0**	81,1	82,0
Promedio				**180,08**	**178,77**	**179,15**	**179,23**	177,92	178,18	**252,54**	**252,61**	**251,77**	**257,01**	**1,78**	**0,88**	**1,05**	0,41	**16,16**	**19,19**

Tabla 12: Resultados para el grupo de instancias BWMV, n, W y LB corresponden a la cantidad de piezas, ancho y mejor solución conocida para cada instancia, respectivamente. B y M, corresponden al Best y Mean obtenido por cada algoritmo

Instancia				SVC	GRASP		ISA		AGPC		AG		%GAP				
	n	W	LB	M	B	M	B	M	B	M	B	M	SVC	GRASP	ISA	AGPC	AG
C01	20	10	60,3	61,4	61,4	61,4	61,3	61,3	**60,3**	**60,3**	59,7	60,6	1,8	1,8	1,7	**0,0**	0,5
	40	10	121,6	122	121,9	121,9	121,8	121,8	**121,7**	**121,8**	121,5	123,6	0,3	0,2	0,2	**0,2**	1,6
	60	10	187,4	**188,6**	**188,6**	**188,6**	**188,6**	**188,6**	189,5	189,7	190,4	193,6	**0,6**	**0,6**	**0,6**	1,2	3,3
	80	10	262,2	**262,6**	**262,6**	**262,6**	**262,6**	**262,6**	263,8	264,0	265,1	270,1	**0,2**	**0,2**	**0,2**	0,7	3,0
	100	10	304,4	**304,9**	305	305	**304,9**	**304,9**	321,5	321,6	322,6	329,3	**0,2**	0,2	**0,2**	5,6	8,2
C02	20	30	19,7	**19,8**	**19,8**	**19,8**	19,8	19,9	20,6	20,6	20,5	20,8	**0,5**	**0,5**	1,0	4,6	5,5
	40	30	39,1	**39,1**	**39,1**	**39,1**	**39,1**	**39,1**	41,3	41,4	41,3	42,0	**0,0**	**0,0**	**0,0**	5,9	7,5
	60	30	60,1	**60,1**	60,3	60,3	**60,1**	**60,1**	64,7	64,7	64,5	65,5	**0,0**	0,3	**0,0**	7,7	9,0
	80	30	83,2	**83,2**	83,3	83,3	**83,2**	**83,2**	89,9	90,1	90,0	91,2	**0,0**	0,1	**0,0**	8,3	9,6
	100	30	100,5	**100,5**	100,6	100,6	**100,5**	**100,5**	109,6	109,8	109,4	111,2	**0,0**	0,1	**0,0**	9,2	10,6
C03	20	40	157,4	164,6	163,5	163,5	163,7	164	**163,4**	**163,5**	163,0	166,7	4,6	3,9	4,2	**3,9**	5,9
	40	40	328,8	333,9	334,2	334,2	333,4	333,8	**332,6**	**332,7**	337,7	344,4	1,6	1,6	1,5	**1,2**	4,7
	60	40	500	506,9	506,6	506,6	**505,4**	**505,8**	527,8	529,2	534,4	544,9	1,4	1,3	**1,2**	5,8	9,0
	80	40	701,7	710,1	709,7	709,7	**708,8**	**709,2**	740,9	741,6	750,5	767,6	1,2	1,1	**1,1**	5,7	9,4
	100	40	832,7	839,9	840,2	840,2	**837,4**	**837,8**	895,9	895,9	910,2	929,7	0,9	0,9	**0,6**	7,6	11,6
C04	20	100	61,4	63,8	**63,3**	**63,3**	63,6	63,9	65,5	65,5	65,6	67,0	3,9	**3,1**	4,1	6,7	9,1
	40	100	123,9	126,2	126,2	126,2	**125,5**	**126,1**	134,8	135,2	135,0	137,2	1,9	1,9	**1,8**	9,1	10,7
	60	100	193	195,6	196,6	196,6	**195,1**	**195,5**	212,4	212,4	212,1	216,1	1,3	1,9	**1,3**	10,1	12,0
	80	100	267,2	270,5	272	272	**269,5**	**269,8**	298,5	298,7	297,8	301,9	1,2	1,8	**1,0**	11,8	13,0
	100	100	322	325,3	327,3	327,3	**324,1**	**324,6**	362,7	362,8	361,7	366,6	1,0	1,6	**0,8**	12,7	13,9
C05	20	100	512,2	537,9	533,9	533,9	534,2	534,6	**512,6**	**513,1**	512,3	520,8	5,0	4,2	4,4	**0,2**	1,7
	40	100	1053,8	1076,4	1074,4	1074,4	1073,3	1073,6	**1050,1**	**1050,1**	1052,2	1073,3	2,1	2,0	1,9	**-0,4**	1,9
	60	100	1614	1647,6	1645,5	1645,5	**1642,7**	**1643,4**	1666,4	1667,8	1682,3	1712,2	2,1	2,0	**1,8**	3,3	6,1
	80	100	2268,4	**2288,9**	2290,5	2290,5	2288,7	2289	2318,0	2318,0	2339,5	2383,4	**0,9**	1,0	0,9	2,2	5,1
	100	100	2617,4	2653,5	2651,1	2651,1	**2642,5**	**2644,4**	2844,1	2844,1	2855,7	2905,8	1,4	1,3	**1,0**	8,7	11,0

Instancia				SVC	GRASP		ISA		AGPC		AG		%GAP				
	n	W	LB	M	B	M	B	M	B	M	B	M	SVC	GRASP	ISA	AGPC	AG
C06	20	300	159,9	169,6	**167,2**	**167,2**	168,7	169,6	173,1	173,4	173,5	177,0	6,1	**4,6**	6,1	8,4	10,7
	40	300	323,5	**332,6**	333,4	333,4	332,2	334	358,8	358,8	358,4	363,5	**2,8**	3,1	3,2	10,9	12,4
	60	300	505,1	**517,2**	519,8	519,9	517,8	519	564,0	564,0	561,8	571,5	**2,4**	2,9	2,8	11,7	13,1
	80	300	699,7	**714,7**	718,3	718,4	713,8	715,5	794,0	794,0	787,0	798,9	**2,1**	2,7	2,3	13,5	14,2
	100	300	843,8	**860,6**	865,1	865,1	859,8	861,1	958,4	958,4	958,7	970,7	**2,0**	2,5	2,1	13,6	15,0
C07	20	100	490,4	501,9	501,9	501,9	501,9	501,9	**454,8**	**455,0**	455,7	462,9	2,3	2,3	2,3	**-7,2**	-5,6
	40	100	1049,7	1059,9	1059	1059	1059	1059	**1001,2**	**1001,3**	1001,7	1023,0	1,0	0,9	0,9	**-4,6**	-2,5
	60	100	1515,9	1530	1529,6	1529,6	1529,6	1529,6	**1477,7**	**1478,3**	1482,9	1514,1	0,9	0,9	0,9	**-2,5**	-0,1
	80	100	2206,1	2222,1	2222,2	2222,2	2222,1	2222,1	**2154,0**	**2154,0**	2177,1	2219,4	0,7	0,7	0,7	**-2,4**	0,6
	100	100	2627	2644	2644	2644	2644,7	2645,4	**2623,2**	**2623,2**	2635,6	2696,7	0,6	0,6	0,7	**-0,1**	2,7
C08	20	100	434,6	461,2	**458,3**	**458,3**	457,3	458,6	464,0	464,1	465,0	475,2	6,1	**5,5**	5,5	6,8	9,3
	40	100	922	956,5	954,3	954,3	**950,1**	**951,9**	997,1	997,4	1007,5	1027,3	3,7	3,5	**3,2**	8,2	11,4
	60	100	1360,9	1403,5	1405	1405	**1395,6**	**1399,4**	1506,7	1509,2	1508,6	1537,83	3,1	3,2	**2,8**	10,9	13,0
	80	100	1909,3	1965	1971,5	1971,5	**1950,1**	**1954,7**	2129,1	2129,1	2140,4	2180,3	2,9	3,3	**2,4**	11,5	14,2
	100	100	2362,8	2425	2436,8	2436,8	**2406,5**	**2410,8**	2637,9	2637,9	2650,9	2710,7	2,6	3,1	**2,0**	11,6	14,7
C09	20	100	1106,8	1106,8	1106,8	1106,8	1106,8	1106,8	**996,0**	**996,0**	994,3	1004,3	0,0	0,0	0,0	**-10,0**	-9,3
	40	100	2189,2	2190,6	2190,6	2190,6	2190,6	2190,6	**1983,2**	**1984,4**	1987,9	2009,8	0,1	0,1	0,1	**-9,4**	-8,2
	60	100	3410,4	3410,4	3410,4	3410,4	3410,4	3410,4	**3087,5**	**3087,5**	3093,7	3125,6	0,0	0,0	0,0	**-9,5**	-8,4
	80	100	4578,6	4588,1	4588,1	4588,1	4588,1	4588,1	**4131,3**	**4131,3**	4169,2	4204,5	0,2	0,2	0,2	**-9,8**	-8,2
	100	100	5430,5	5434,9	5434,9	5434,9	5434,9	5434,9	**5056,9**	**5056,9**	5095,0	5149,7	0,1	0,1	0,1	**-6,9**	-5,2
C10	20	100	337,8	351,5	350,5	350,5	**350,2**	**350,4**	350,4	350,5	350,5	355,7	4,1	3,8	**3,7**	3,8	5,3
	40	100	642,8	667	664,4	664,4	**663,1**	**664**	686,3	686,5	686,7	699,2	3,8	3,4	**3,3**	6,8	8,8
	60	100	911,1	936,6	934,6	934,7	**931,3**	**933,1**	985,4	985,4	990,0	1004,6	2,8	2,6	**2,4**	8,2	10,3
	80	100	1177,6	1212,4	1209,9	1209,9	**1201,6**	**1204,1**	1290,4	1290,4	1300,5	1324,6	3,0	2,7	**2,3**	9,6	12,5
	100	100	1476,5	1514	1512,3	1512,3	**1501,1**	**1504,2**	1642,2	1642,2	1653,0	1683,0	2,5	2,4	**1,9**	11,2	14,0
Promedio				**1043,19**	**1043,33**	**1043,34**	1040,74	1041,53	**1038,24**	**1038,47**	**1043,61**	**1060,70**	**1,80**	**1,77**	1,66	**4,32**	**6,37**

A continuación se presentan algunos gráficos con las curvas de convergencia de los 4 nodos de AGPC y la versión secuencial (AG). En el eje y se muestra la mejor altura y en el eje x, las generaciones. Solo se seleccionaron algunas ejecuciones para ser graficadas, el objetivo es mostrar las curvas de convergencia de los AGPC y AG para las distintas instancias.

Las curvas muestran el mejor individuo de cada generación y corresponden a la mejor ejecución del algoritmo.

La Ilustración 4-1 corresponde al grupo de instancias C, la Ilustración 4-2 al grupo de instancias N y la Ilustración 4-3 al grupo de instancias BWMV, además, los layouts correspondientes a esas instancias pueden ser vistos en la Ilustración 4-4, Ilustración 4-5 y Ilustración 4-6, respectivamente.

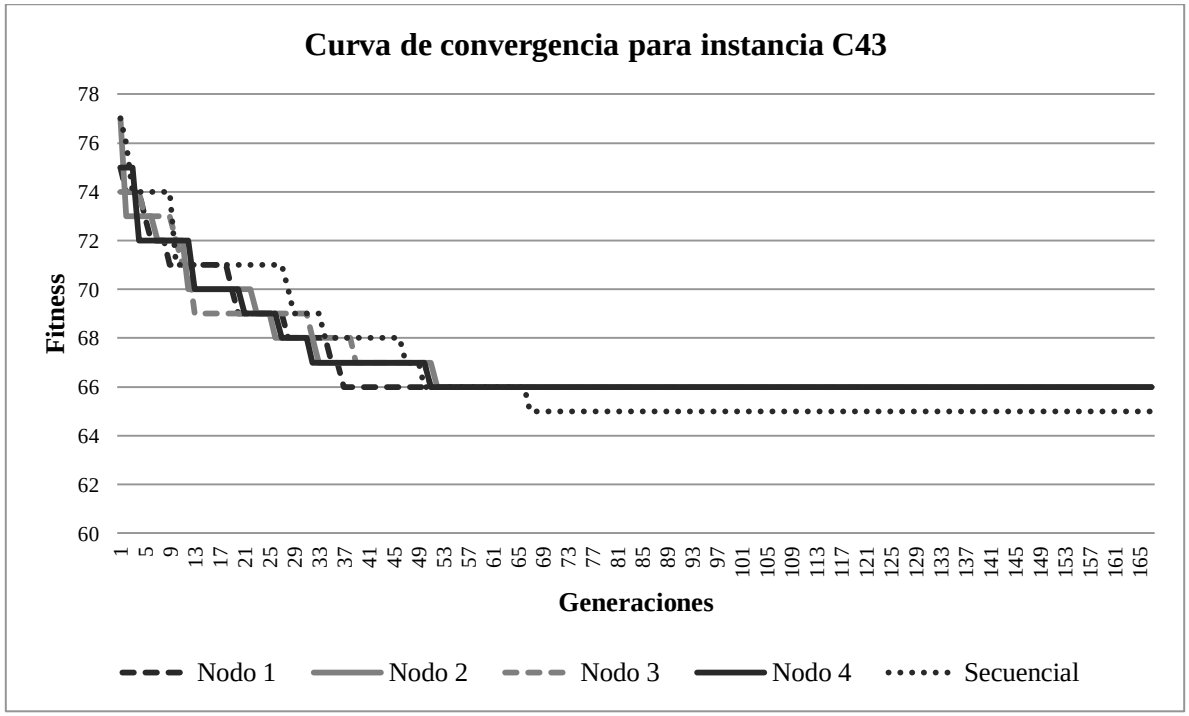


Ilustración 4-1: Gráfico de convergencia para la instancia C43 del grupo de instancias C, con n=49, W=60, Óptimo = 60 y mejor solución del AG = 65 y del AGPC=66

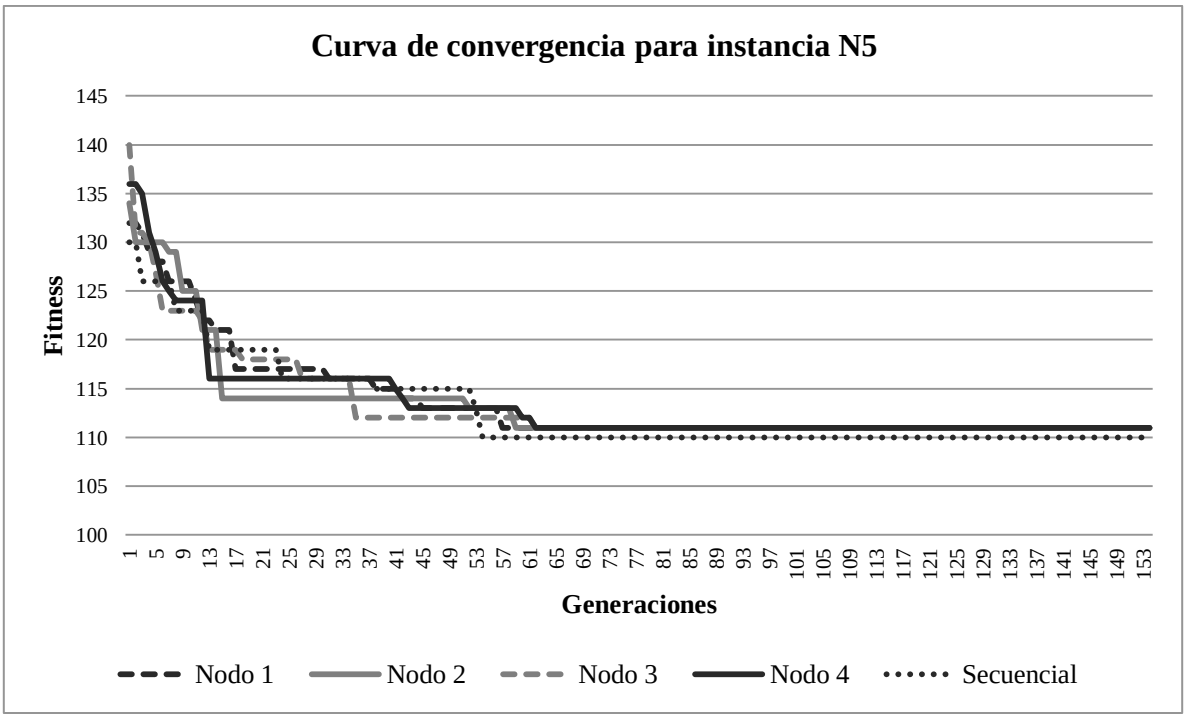


Ilustración 4-2: Gráfico de convergencia para la instancia N5 del grupo de instancias N, con n=50, W=100, Óptimo=100 y mejor solución del AG = 110 y del AGPC=111

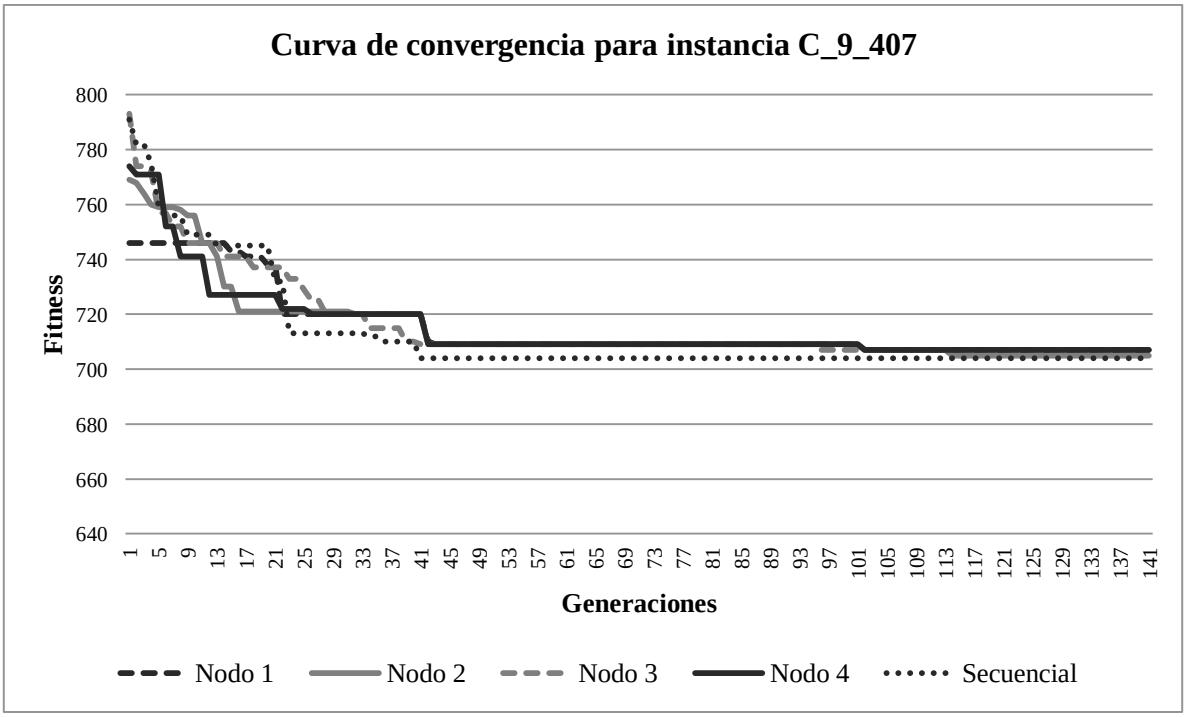


Ilustración 4-3: Gráfico de convergencia para la instancia C_9_407, del subconjunto C01 del grupo de instancias BWMV, n=20, W=100, mejor solución del AG=704 y del AGPC=705

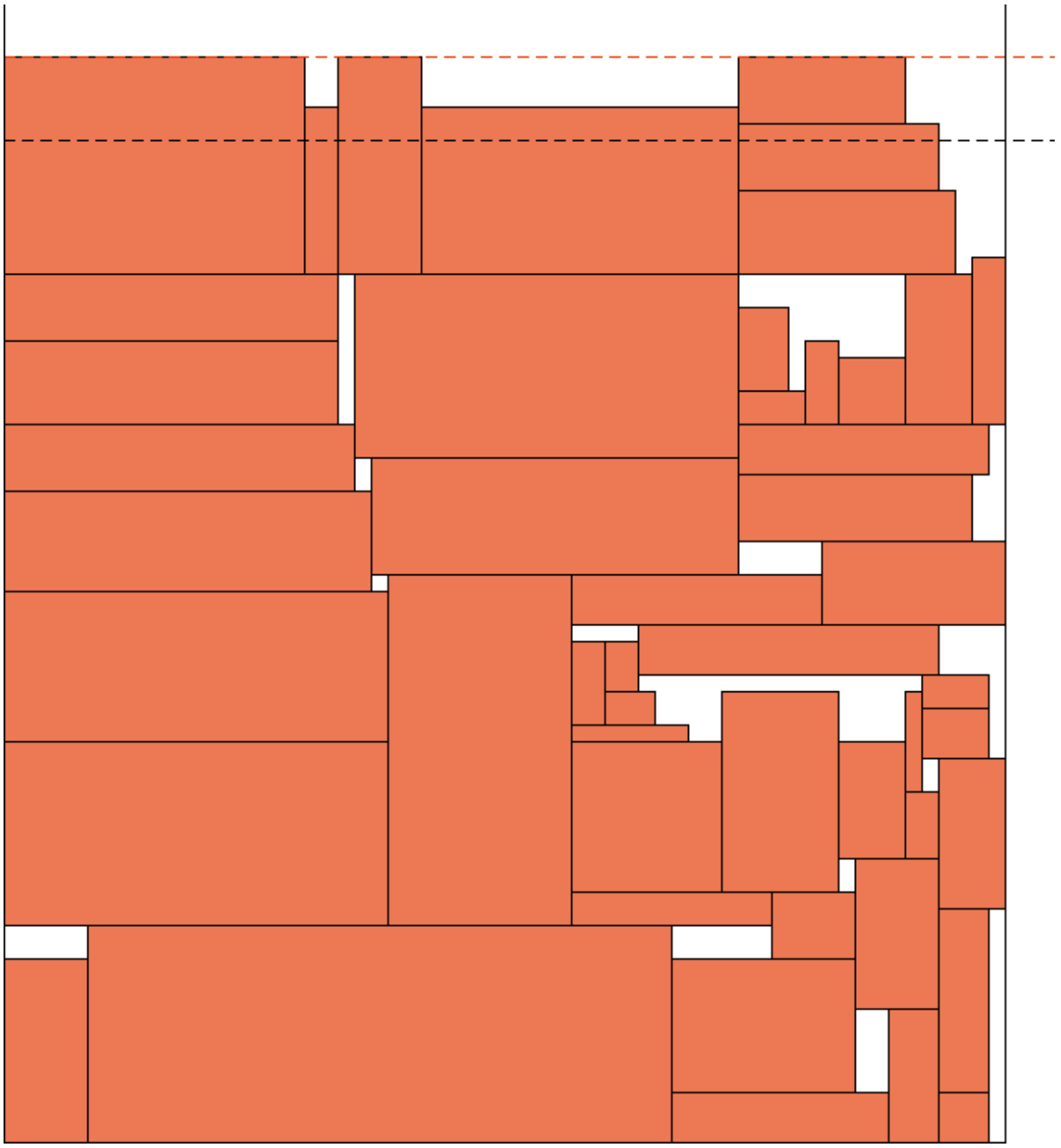


Ilustración 4-4: Layout obtenido por el AG como mejor solución para la instancia C43, con n=49, W=60, y altura de 65 (Línea segmentada clara), mientras que la optima es de 60 (Línea segmentada oscura)

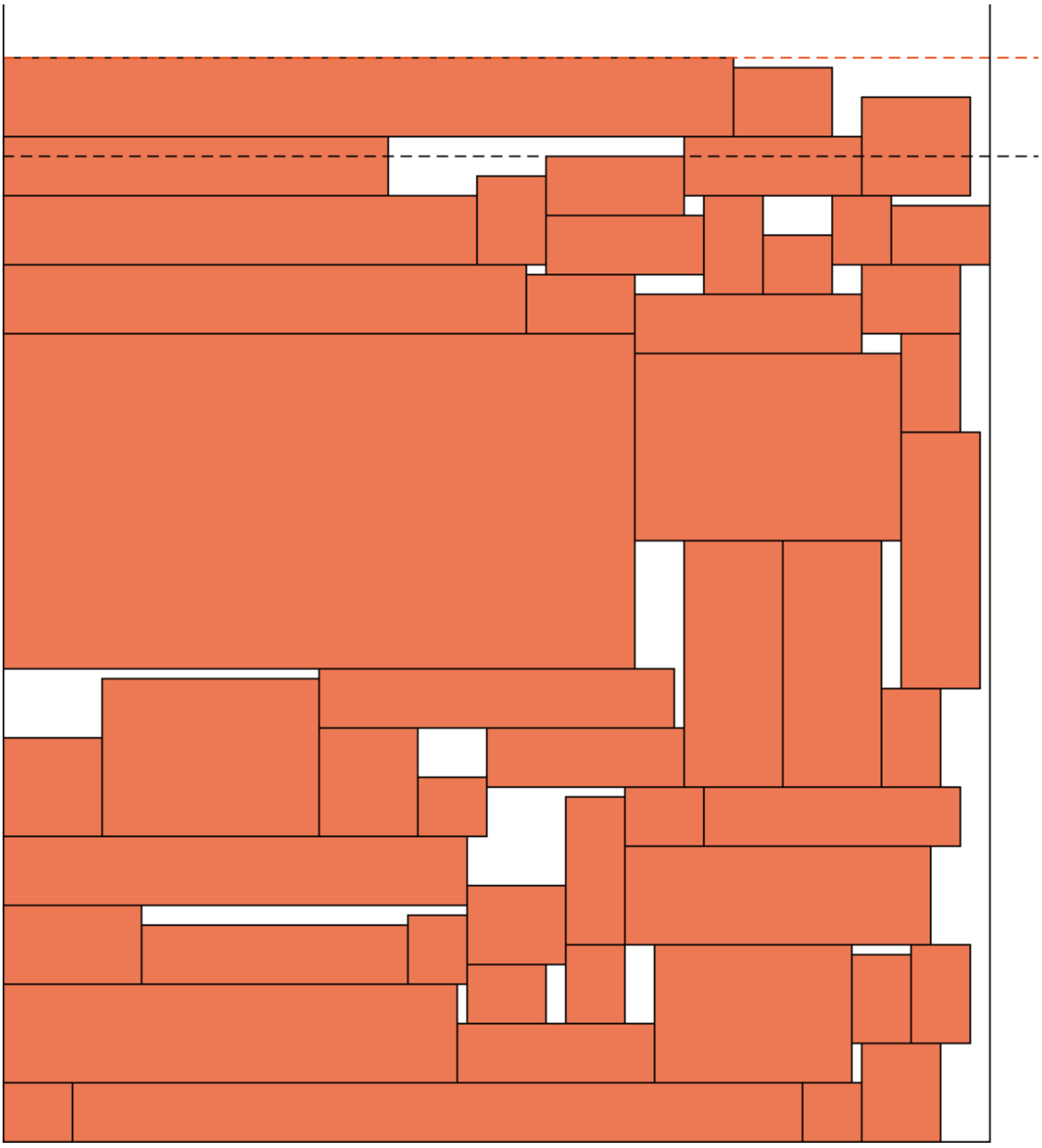


Ilustración 4-5 : Layout obtenido por el AG como mejor solución para la instancia N5, con n=50, W=100 y altura de 110 (Línea segmentada clara), mientras que la optima es de 100 (Línea segmentada oscura)

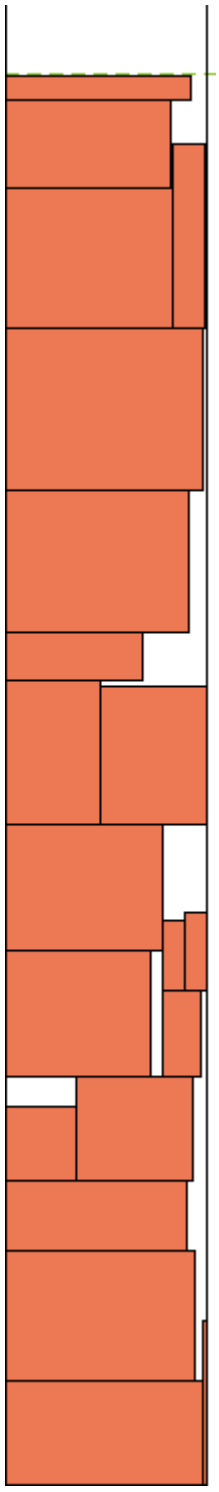


Ilustración 4-6: Layout obtenido por el AG como mejor solución para la instancia C_9_407, con n=20, W=100 y altura de 704 (Línea segmentada clara)

4.2. RESULTADOS PARA EL PCPGBR

Los parámetros empleados para resolver este problema corresponden a los encontrados mediante la calibración de ParamILS (Hutter, y otros 2009), y pueden ser vistos en la Tabla 13.

Tabla 13: Parámetros para el AG y AGPC, para el PCPGBR, calibrados por ParamILS

Parámetro	AG	AGPC
Probabilidad de cruzamiento	0,9	0,9
Probabilidad de mutación	0,1	0,1
Tamaño de la población	250	250 x 4 = 1000
Tasa de migración	-	10
Tipo de migración	-	Asíncrona
Porcentaje de envío	-	10%
Porcentaje de recepción	-	10%

Para el AGPC se emplearon 4 nodos para ejecutar los AG y un nodo coordinador, por esto la población total es de 1000 ya que se divide de forma uniforme en los 4 nodos.

Las tablas 14, 15 y 16 muestran los resultados obtenidos por AGPC y AG para los grupos de instancias GT1, GT2 y GT3 respectivamente, junto con los resultados obtenidos por GRASP (Alvarez-Valdés, Parreño y Tamarit 2008) y el óptimo o mejor resultado conocido con el fin de comparar el desempeño, además de eso se muestra el margen de error (%GAP).

Para el AGPC y AG se presentan 2 resultados, el mejor (*Best*, abreviado con una B) y el promedio de las 10 ejecuciones (*Mean*, abreviado con una M). Los resultados que se muestran corresponden al área utilizada, por lo tanto, mientras mayor mejor, pero el problema esta diseñado para optimizar la pérdida, por tanto, como resultado para AGPC y AG se muestra el área de la plancha menos la pérdida.

Para cada tabla la primera columna corresponde a la instancia, con el ancho (W) y alto (L) de la lamina, y el numero de piezas distintas (NP), mejor valor conocido u óptimo (LB), luego cada columna corresponde a un algoritmo para los que se indica el *Best* (B) y *Mean* (M), en algunos casos solo el *Best* y otros solo *Mean*. La última columna corresponde al margen de error, la que se subdivide en cada algoritmo indicando el error porcentual para cada instancia (%GAP).

Tabla 14: Resultados para el grupo de instancias GT1, W, L, NP y LB corresponden al ancho, alto de la lamina, el numero de piezas y la mejor solución conocida para cada instancia, respectivamente. B y M, corresponden al Best y Mean obtenido por cada algoritmo

	Instancia				**GRASP**	**AGPC**		**AG**		**%GAP**		
	W	L	NP	LB		B	M	B	M	GRASP	AGPC	AG
OF1	70	40	23	2737	**2737**	2627	2627	2640	2614,5	**0,00**	0,04	0,04
W	70	40	62	2721	**2721**	2718	2718	2718	2656,1	**0,00**	0,00	0,02
CU1	100	125	82	12330	**12312**	12101	12101	12120	11867,6	**0,00**	0,02	0,04
CU2	150	175	90	26100	**26100**	24777	24691,5	24788	24463,6	**0,00**	0,05	0,06
CU3	134	125	158	16723	**16652**	16524	16524	16210	16055,6	**0,00**	0,01	0,04
CU4	285	354	113	99495	**99264**	97029	97029	97194	96705,4	**0,00**	0,02	0,03
CU5	456	385	120	173364	**173364**	168768	168768	169538	167294,7	**0,00**	0,03	0,04
CU6	356	447	124	158572	**158572**	156293	156293	153420	150585,3	**0,00**	0,01	0,05
CU7	563	458	56	247150	**247150**	245424	245424	245424	242122,3	**0,00**	0,01	0,02
CU8	587	756	78	433331	**432714**	428679	428679	427132	423658,4	**0,00**	0,01	0,02
CU9	856	785	76	657055	**651597**	640122	640122	641983	635064,3	**0,01**	0,03	0,03
CU10	794	985	129	773772	**767580**	753513	753513	754773	743377,2	**0,01**	0,03	0,04
CU11	977	953	134	924696	**909898**	886300	886300	894808	876489,6	**0,02**	0,04	0,05
Promedio				**271388,15**	**269281,62**	**264221,15**	**264214,58**	**264826,77**	**260996,51**	**0,003**	**0,023**	**0,038**

Tabla 15: Resultados para el grupo de instancias GT2, W, L, NP y LB corresponden al ancho, alto de la lamina, el numero de piezas y la mejor solución conocida para cada instancia, respectivamente. B y M, corresponden al Best y Mean obtenido por cada algoritmo

	Instancia				GRASP	AGPC		AG		%GAP		
	W	L	NP	LB		B	M	B	M	GRASP	AGPC	AG
P2s	40	70	23	2778	**2740**	2618,0	2618,0	2598,0	2580,3	**0,01**	0,06	0,07
P3s	40	70	62	2721	**2721**	2680,0	2680,0	2680,0	2680	**0,00**	0,02	0,02
A1s	50	60	62	2950	**2950**	2889,0	2889,0	2890,0	2866	**0,00**	0,02	0,03
A2s	60	60	53	3535	**3535**	3480,0	3480,0	3480,0	3480	**0,00**	0,02	0,02
STS2s	55	85	78	4653	4653	**4675,0**	**4675,0**	4675,0	4656,7	0,00	**0,00**	0,00
STS4s	99	99	50	9770	**9583**	9549,0	9548,1	9540,0	9460,3	**0,02**	0,02	0,03
CHL1s	132	100	63	13099	**13014**	12448,0	12448,0	12888,0	12608,6	**0,01**	0,05	0,04
CHL2s	62	55	19	3279	3231	**3278,0**	**3278,0**	3278,0	3218,2	0,01	**0,00**	0,02
CHL3s	157	121	35	7402	**7402**	**7402,0**	**7402,0**	**7402,0**	**7402**	**0,00**	**0,00**	**0,00**
CHL4s	207	231	27	13932	**13932**	**13932,0**	**13932,0**	**13932,0**	**13932**	**0,00**	**0,00**	**0,00**
A3	70	80	46	5451	**5410**	5405,0	5405,0	5405,0	5405	**0,01**	0,01	0,01
A4	90	70	35	6179	6052	**6092,0**	**6092,0**	**6092,0**	**6092**	0,02	**0,01**	**0,01**
A5	132	100	45	12985	12832	**12888,0**	**12888,0**	12831,0	12647,8	0,01	**0,01**	0,03
CHL5	20	20	18	390	390	**400,0**	**400,0**	400,0	394	0,00	**-0,03**	-0,01
CHL6	130	130	65	16869	**16643**	16049,0	16049,0	16282,0	15845,9	**0,01**	0,05	0,06
CHL7	130	130	75	16881	**16583**	16323,0	16323,0	16138,0	15733,8	**0,02**	0,03	0,07
Hchl1	130	130	65	11303	10829	**16843,0**	**16843,0**	16830,0	16796,4	0,04	**-0,49**	-0,49
Hchl2	130	130	75	9954	9691	**16872,0**	**16872,0**	16822,0	16795,7	0,03	**-0,69**	-0,69
Hchl3s	127	98	51	12215	**12159**	11650,0	11650,0	11793,0	11384,1	**0,00**	0,05	0,07
Hchl4s	127	98	32	11994	**11621**	11617,0	11617,0	11596,0	11301,9	**0,03**	0,03	0,06
Hchl5s	205	223	60	45361	**44346**	42712,0	42712,0	43999,0	42617,7	**0,02**	0,06	0,06
Hchl6s	253	244	60	61040	**60403**	57671,0	57671,0	58129,0	56885	**0,01**	0,06	0,07
Hchl7s	263	241	90	63112	**62547**	60857,0	60857,0	61942,0	60133,1	**0,01**	0,04	0,05
Hchl8s	49	20	18	911	904	**924,0**	**924,0**	924,0	896,6	0,01	**-0,01**	0,02
Hchl9	65	76	76	5240	**5240**	4937,0	4937,0	4937,0	4934,8	**0,00**	0,06	0,06
Promedio				**13760,16**	**13576,44**	**13767,64**	**13767,60**	**13899,32**	**13629,92**	**0,01**	**-0,03**	**-0,02**

Tabla 16: Resultados para el grupo de instancias GT3, W, L, NP y LB corresponden al ancho, alto de la lamina, el numero de piezas y la mejor solución conocida para cada instancia, respectivamente. B y M, corresponden al Best y Mean obtenido por cada algoritmo

Instancia					**AGPC**			**AG**		**%GAP**		
	W	L	NP	LB	GRASP	B	M	B	M	GRASP	AGPC	AG
APT30	927	152	192	140904	**138863**	136265	136265	135673	134020,9	**0,01**	0,03	0,05
APT31	856	964	258	822393	**801767**	800057	799398,4	801239	785620,3	**0,03**	0,03	0,04
APT32	307	124	249	38068	**37786**	36638	36638	37246	36359,4	**0,01**	0,04	0,04
APT33	241	983	224	236611	**231178**	227928	227928	227404	225169,2	**0,02**	0,04	0,05
APT34	795	456	130	360084	**353822**	346359	346359	349396	344860,6	**0,02**	0,04	0,04
APT35	960	649	153	620797	**607864**	598548	598548	602385	587943,3	**0,02**	0,04	0,05
APT36	537	244	153	130744	**129634**	124009	124009	125253	123416,1	**0,01**	0,05	0,06
APT37	440	881	222	387276	**379329**	372770	372770	372480	367719,6	**0,02**	0,04	0,05
APT38	731	358	202	261154	**252605**	250870	250870	248706	247068,4	**0,03**	0,04	0,05
APT39	538	501	163	268750	**263076**	258585	258585	261231	258042,6	**0,02**	0,04	0,04
Promedio				**326678,10**	**319592,40**	**315202,90**	**315137,04**	**316101,30**	**311022,04**	**0,02**	**0,04**	**0,05**

A continuación se presentan algunos gráficos con las curvas de convergencia de los 4 nodos de AGPC y la versión secuencial (AG). En el eje y se muestra la menor pérdida y en el eje x, las generaciones. Solo se seleccionaron algunas ejecuciones para ser graficadas, el objetivo es mostrar las curvas de convergencia de los AGPC y AG para las distintas instancias.

Las curvas muestran el mejor individuo de cada generación. Las ilustraciones desde 4-27 hasta 4-31 corresponden al conjunto de instancias GT1, de 4-32 a 4-37 al conjunto GT2 y de 4-38 a 4-42 al conjunto GT3.

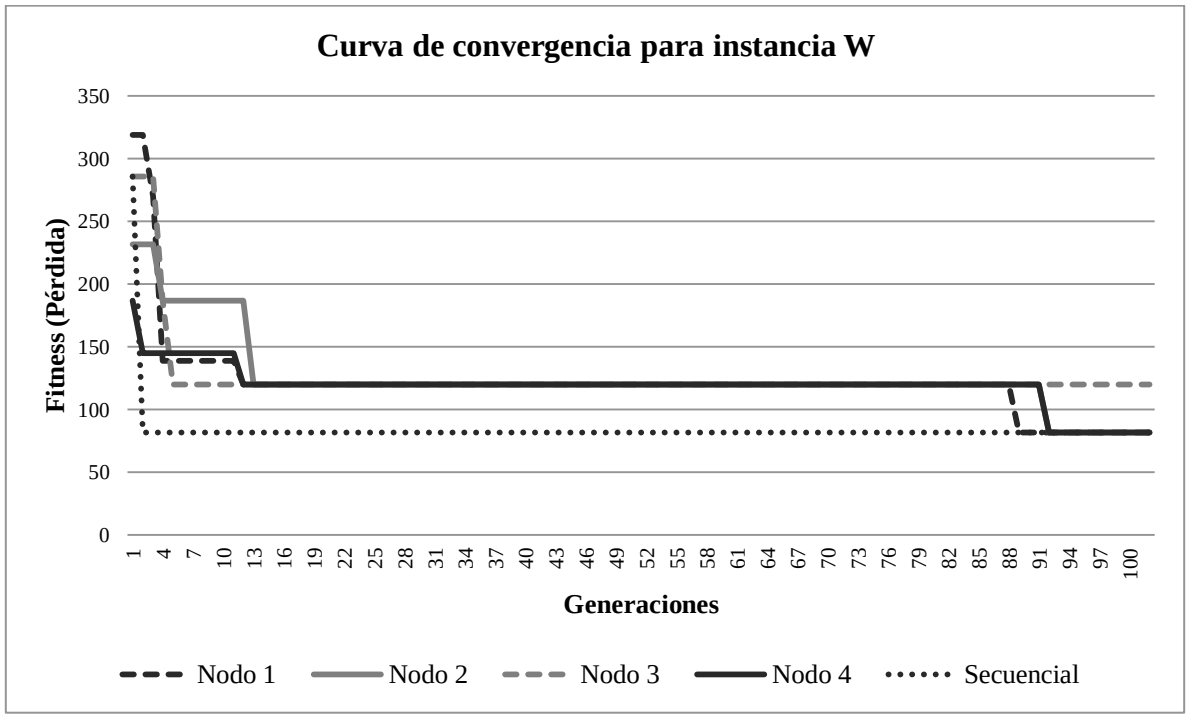


Ilustración 4-7: Gráfico de convergencia de la mejor ejecución para la instancia W del grupo de instancias GT1, con W=70, L=40, LB=2721, Pérdida=79, pérdida encontrada por el AG=82 y por el AGPC=82

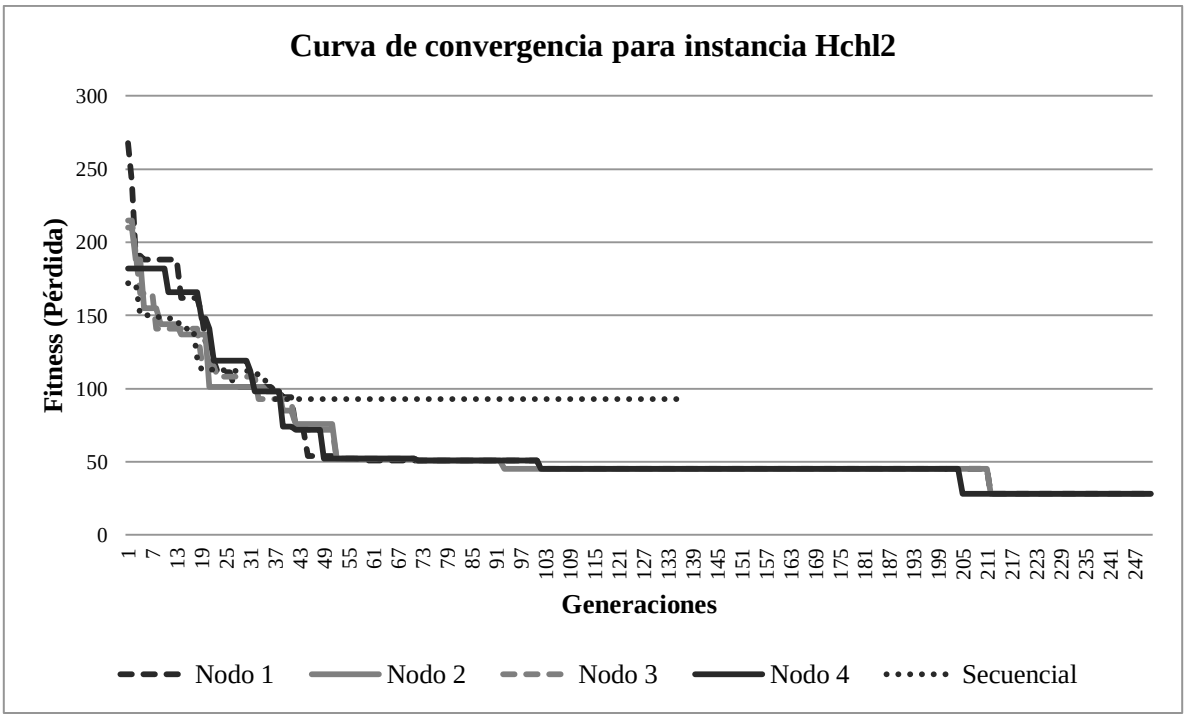


Ilustración 4-8: Gráfico de convergencia de la mejor ejecución para la instancia Hchl2 del grupo de instancias GT2, con W=130, L=130, LB=9954, Pérdida=6946, pérdida encontrada por el AG=78 y por el AGPC=28

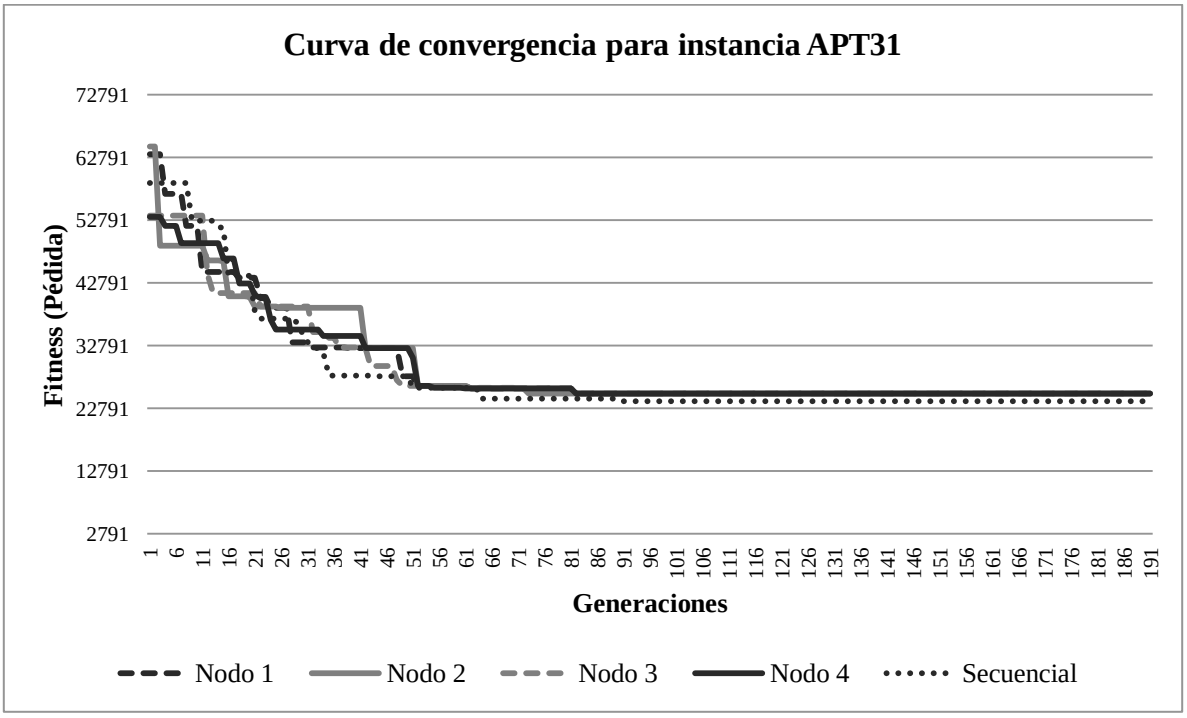


Ilustración 4-9: Gráfico de convergencia de la mejor ejecución para la instancia APT31 del grupo de instancias GT3, con W=856, L=964, LB=822393, Pérdida=2791, pérdida encontrada por el AG=23945 y por el AGPC=25127

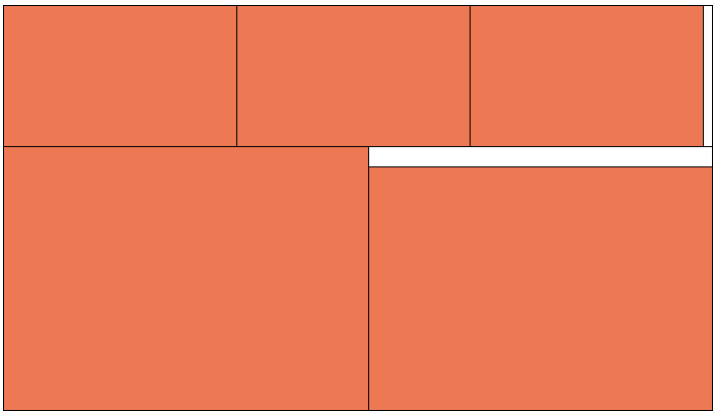


Ilustración 4-10: Layout obtenido por el AGPC como mejor solución para la instancia W, con W=70, L=40, área usada igual a 2718 (pérdida = 82), la mejor conocida es de 2721 (pérdida = 79).

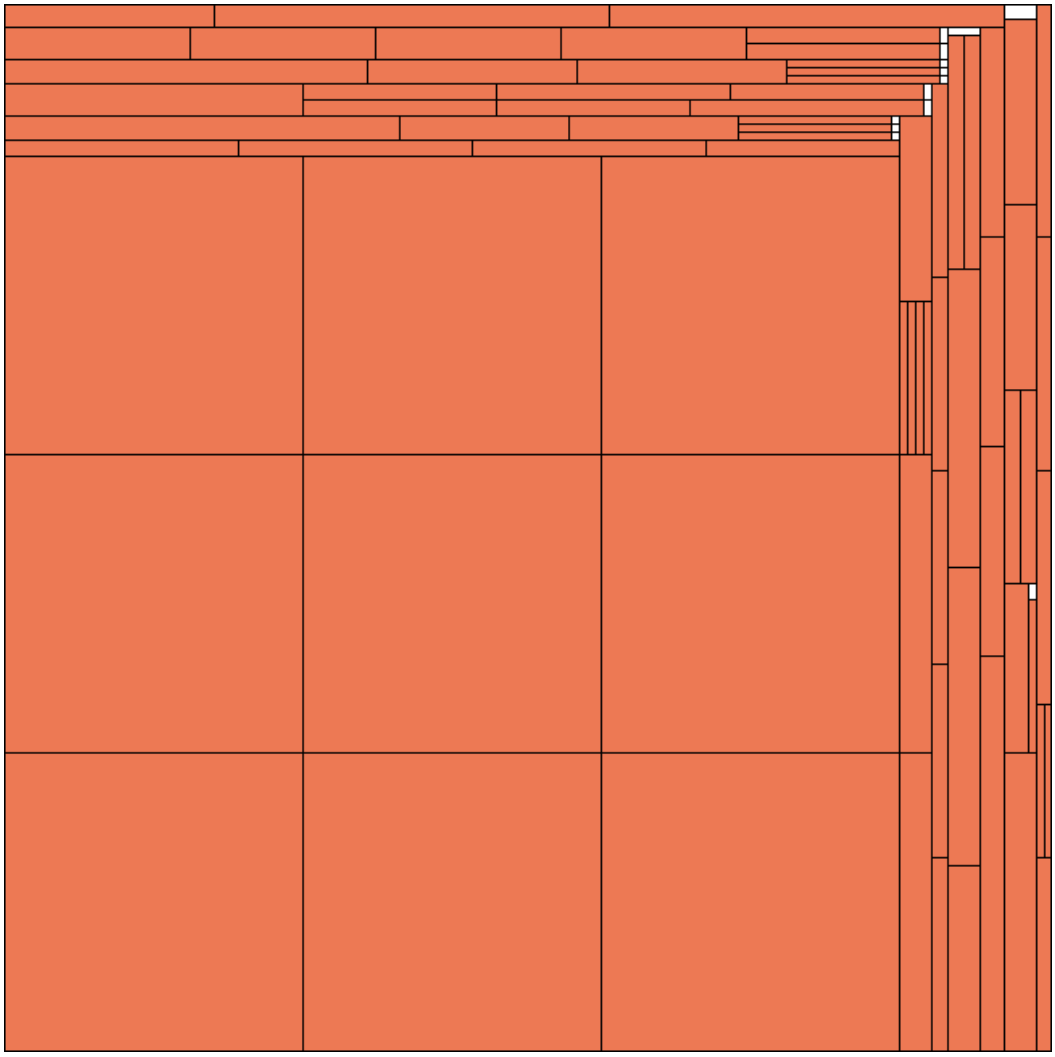


Ilustración 4-11: Layout obtenido por el AGPC como mejor solución para la instancia Hchl2, con W=130, L=130, área usada igual a 16872 (pérdida = 28), la mejor conocida es de 9954 (pérdida = 6946).

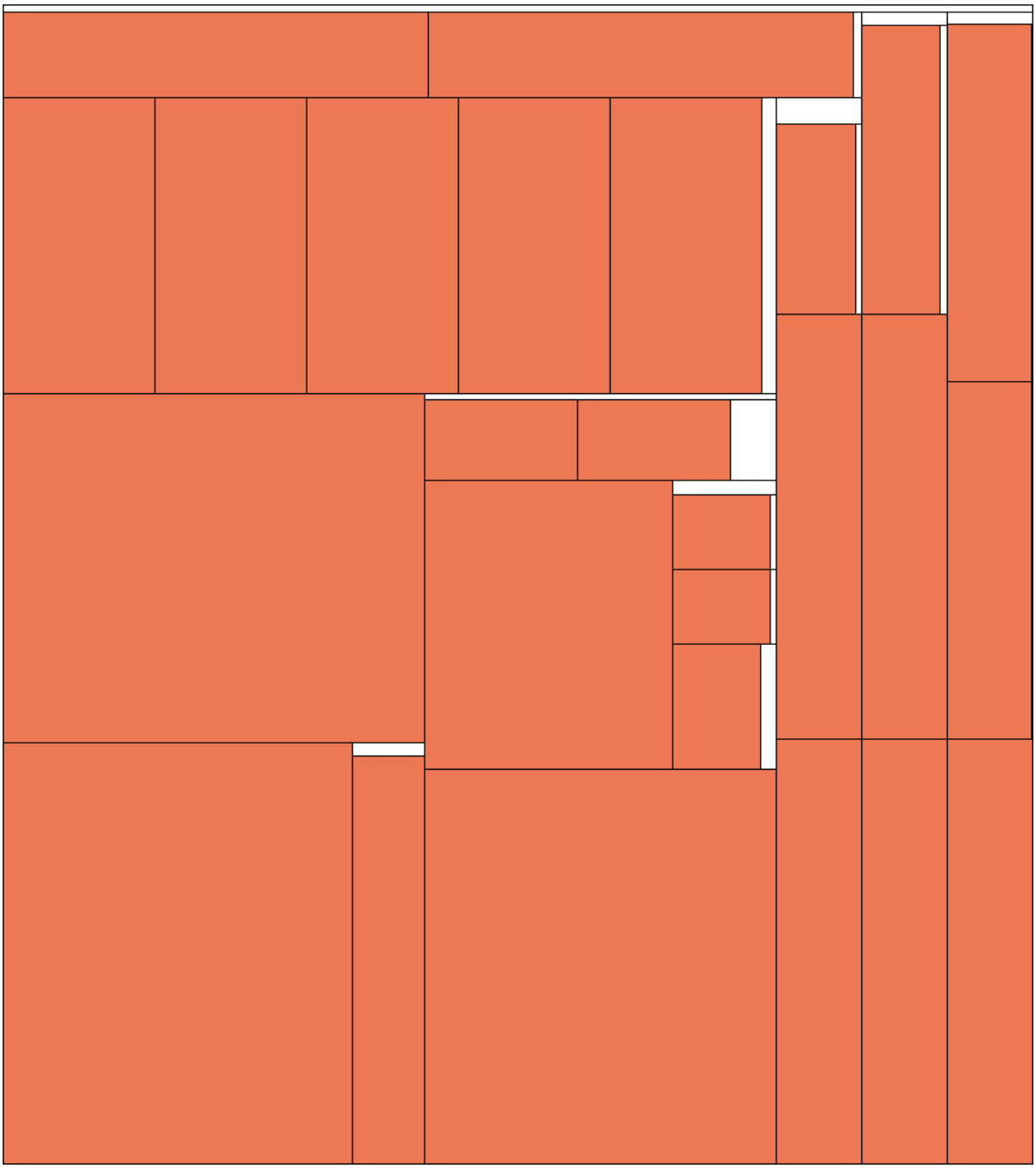


Ilustración 4-12: Layout obtenido por el AG como mejor solución para la instancia APT31, con W=856, L=964, área usada igual a 801239 (pérdida = 23945), la mejor conocida es de 822393 (pérdida = 2791).

4.3. SPEED UP

Para calcular el *Speed Up* se utiliza el grupo de instancias C y GT1 y, se comparan los tiempos requeridos por el AGPC y el AG, los parámetros con los que se ejecutan los algoritmos corresponden a los encontrados por ParamILS en la calibración y pueden ser vistos en la Tabla 17.

Tabla 17: Parámetros obtenidos por ParamILS y usados para obtener el Speed Up

Parámetro	AG	AGPC
Probabilidad de cruzamiento	0,9	0,9
Probabilidad de mutación	0,1	0,1
Tamaño de la población	250	250
Tasa de migración	-	10
Tipo de migración	-	Asíncrona
Porcentaje de envío	-	10%
Porcentaje de recepción	-	10%

Ambos con un tope de 250 generaciones, para obtener los resultados, ambos algoritmos fueron ejecutados 10 veces con cada instancia y se registro el tiempo comprendido entre el comienzo de la ejecución del algoritmo hasta que finaliza su ejecución entregando la mejor solución encontrada. El objetivo de esta medición es verificar la hipótesis que plantea que el AGPC mejora los tiempos de ejecución del AG.

Para calcular el *Speed Up* se utilizara lo planteado por Alba en su libro *Parallels Metaheuristics, A New Class of Algorithms* (E. Alba 2005), quien para algoritmos no-determinísticos lo define de la siguiente forma, donde s_m corresponde al *Speed Up* de m, $E[T_1]$ al promedio de tiempo empleado por un procesador para ejecutar el algoritmo y $E[T_m]$ al promedio de tiempo empleado por m procesadores, incluyendo tiempos de comunicación.

$$s_m = \frac{E[T_1]}{E[T_m]}$$

En la Tabla 18 se presentan los tiempos promedio para cada instancia del grupo de instancias C del PET, para el AGPC y el AG, junto con el *Speed Up* asociado a cada instancia, al fondo de ésta se encuentran los valores promedio.

Tabla 18: Tiempos promedio de AGPC, AG y Speed Up para C

	Tiempo promedio AGPC (T_4)	Tiempo promedio AG (T_1)	Speed Up
C11	0,284	0,828	2,912
C12	0,300	0,844	2,816
C13	0,309	0,792	2,560
C21	0,508	1,669	3,289
C22	0,516	1,617	3,135
C23	0,497	1,582	3,182
C31	0,683	2,400	3,515
C32	0,722	2,455	3,399
C33	0,679	2,367	3,484
C41	1,142	4,058	3,554
C42	1,163	3,971	3,415
C43	1,111	3,924	3,533
C51	1,608	5,857	3,642
C52	1,582	5,811	3,673
C53	1,634	5,963	3,648
C61	2,537	9,460	3,729
C62	2,469	9,047	3,665
C63	2,573	9,347	3,633
C71	8,252	31,321	3,795
C72	7,800	30,661	3,931
C73	8,000	31,512	3,939
Promedio	2,113	7,880	3,450

En la Tabla 19 se presentan los tiempos promedio para cada instancia del grupo de instancias GT1 del PCPGBR, para el AGPC y el AG, junto con el *Speed Up* asociado a cada instancia, al fondo de ésta se encuentran los valores promedio.

Tabla 19: Tiempos promedio de AGPC, AG y Speed Up para GT1

	Tiempo promedio AGPC (T_4)	Tiempo promedio AG (T_1)	Speed Up
OF1	0,206	0,315	1,532
W	0,295	0,612	2,072
CU1	0,371	0,891	2,400
CU2	0,375	0,971	2,587
CU3	0,538	1,582	2,942
CU4	0,432	1,178	2,728
CU5	0,465	1,154	2,483
CU6	0,425	1,286	3,028
CU7	0,246	0,623	2,529
CU8	0,337	0,839	2,493
CU9	0,305	0,808	2,652
CU10	0,478	1,385	2,897
CU11	0,514	1,605	3,121
Promedio	0,384	1,019	2,574

CAPITULO 5. ANALISIS DE RESULTADOS

En este capítulo se analizan los resultados presentados en el capítulo anterior, primero se analizan los resultados del PET y luego los del problema de Corte de Piezas Guillotinable Bidimensional Restricto.

Para los resultados obtenidos para cada grupo de instancias se analizan los siguientes puntos:

1. Promedio de las soluciones para cada instancia (*Mean*)
2. Mejores soluciones obtenidas para cada instancia (*Best*)
3. Soluciones óptimas encontradas
4. Resultados respecto a los demás algoritmos

5.1. ANALISIS DE RESULTADOS PARA EL PET

Para probar los algoritmos AGPC y AG se emplearon 3 grupos de instancias, C (Hopper y Turton (Hopper y Turton 2001)), N (Burke (Burke, Kendall y Whitwell, A new placement heuristic for the orthogonal stock-cutting problem 2004)) y BWMV (Berkey y Wang (Berkey y Wang 1987) y, Martello y Vigo (Lodi, Martello y Vigo 1999)) con 21, 13 y 500 instancias respectivamente. Además los resultados serán comparados con los obtenidos por BSA (Burke et al. (Burke, Kendall y Whitwell, A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock-Cutting Problem 2009)), SVC (Belov et al. (Belov, Scheithauer y Mukhacheva 2008)), GRASP (Alvarez-Valdés et al. (Alvarez-Valdés, Parreño y Tamarit 2008)), y finalmente con ISA (Defu Zhang (Stephen, Zhang y Sim 2011)).

5.1.1. Grupo de instancias C

El Algoritmo Genético Paralelo Cooperativo encuentra, en promedio, mejor calidad de soluciones que el Algoritmo Genetico en la mayoría de las instancias del grupo C, tal como se puede ver en la Tabla 10 el AGPC obtiene un mejor *Mean* que el AG en 18 de las 21 instancias pertenecientes al grupo C, en otras 2 quedan igual y en una AG encuentra un mejor *Mean* que AGPC, con un error (GAP) promedio de 10.62% para el AGPC y de un 12.82% para el AG, por tanto el AGPC muestra ser más competitivo en este grupo de instancias.

Las mejores soluciones obtenidas por AGPC y AG son igual calidad para un tercio de las instancias, para otro tercio lo hace AGPC y el tercio restante es para AG.

El AGPC no es capaz de encontrar la solución óptima para las instancias del grupo C, mientras que el AG si pudo hacerlo para la instancia C13.

La calidad de las soluciones obtenidas por el AGPC y AG quedan por debajo de las obtenidas por los demás algoritmos presentes en la Tabla 10, específicamente el mejor obtiene un 0.76% de error promedio y corresponde al algoritmo ISA, seguido por GRASP con un GAP de 0.95%, SVC con 1.03%, BSA con 2.33%, AGPC con 10.62% y AG con 12.82%.

5.1.2. Grupo de instancias N

El AGPC encuentra, en promedio, mejor calidad de soluciones que AG para todas las instancias del grupo N, tal como se puede ver en la Tabla 11, el AGPC obtiene un mejor *Mean* en cada una de las 13 instancias de este grupo, finaliza con un GAP promedio de 16.16% versus un 19.19% del AG.

Las mejores soluciones encontradas por el AGPC y AG son de igual calidad para 5 de las 13 instancias, para 4 instancias el AGPC encuentra mejores soluciones que AG y para las 4 restantes AG es capaz de encontrar mejores soluciones que el AGPC.

El AGPC y AG son capaces de encontrar la solución óptima para una instancia del grupo de instancias N, específicamente, para la instancia N1, mientras que para las demás, no son capaces.

La calidad de las soluciones obtenidas por el AGPC y AG es inferior a las encontradas por los demás algoritmos, tal como se puede ver en la Tabla 11, donde

nuevamente ISA obtiene el mejor error promedio, 0.41%, seguido de SVC 0.88%, GRASP 1.05, BSA 1.78%, AGPC 16.16% y AG 19.19%.

5.1.3. Grupo de instancias BWMV

El AGPC encuentra, en promedio, mejor calidad de soluciones que AG en todas las instancias del grupo BWMV, específicamente, al revistar los resultados obtenidos para las instancias del grupo BWMV, presentados en la Tabla 12, se tiene que el AGPC obtiene un mejor *Mean* en cada una de las 500 instancias al compararlo con el AG, termina con un promedio de error de un 4.32% *versus* un 6.37% obtenido por el AG.

Las mejores soluciones obtenidas por el AGPC son mejores que las del AG en la mayoría de las instancias, específicamente, se tiene que el AGPC obtiene un mejor promedio de mejores soluciones que el AG en 35 de los 50 subgrupos de instancias, mientras que el AG lo hace en 14 subgrupos y empatan en 1, cada subgrupo contiene 10 instancias.

El AGPC y AG son capaces de encontrar soluciones de mejor o igual calidad que la mejor conocida hasta ahora para un 12% de las instancias del grupo BWMV. Es importante recordar que este grupo de instancias es con pérdida, por lo tanto los mejores resultados que se presentan corresponden a las mejores soluciones encontradas y no necesariamente al óptimo. El AGPC y AG encuentran resultados iguales o mejores a la mejor solución conocida para 12 de los 100 subgrupos, en los grupos C01, C05, C07 y C09.

La calidad de las soluciones obtenidas por el AGPC y AG es inferior a las encontradas por los demás algoritmos, tal como se puede ver en la Tabla 12, nuevamente el algoritmo más competitivo resulta ser ISA con 1.66%, seguido de GRASP con 1.77%, SVC con 1.8%, AGPC con 4.31% y AG con 6.37%.

El AGPC encuentra mejor calidad de soluciones que AG para la mayoría de las instancias probadas en este problema, específicamente, el AGPC obtiene un mejor *Mean* que AG en 531 de las 534 instancias probadas, lo que corresponde al 99.4%.

Las mejores soluciones encontradas por el AGPC, son mejores que las del AG para la mayoría de las instancias, específicamente, el AGPC obtiene un mejor, mejor resultado que AG para 361 de las 534 instancias, lo que equivale al 67.6%, mientras que el AG lo hace en 152, correspondiente al 28.5% y obtienen el mismo mejor resultado para 21

instancias equivalente al 3.9%, al ver la distribución de los mejores *Mean* y *Best* se puede determinar que AGPC es más estable en sus resultados dado que el promedio de las ejecuciones fue ampliamente mejor que el de AG mientras que al considerar las mejores soluciones encontradas, ambos están parejos, es decir, comparten la capacidad de encontrar buenas soluciones, pero el AGPC lo hace con mayor frecuencia y constancia, mientras que AG itera entre buenos y no tan buenos resultados, empeorando su *Mean*.

Además el AGPC y AG muestran un desempeño mucho más competitivo con el grupo de instancias BWMV que con C y N, esto se puede deber a la heurística empleada para insertar las piezas en el *layout* (*Bottom-Left*), ya que una de sus desventajas conocidas es que en la medida que las piezas son agregadas va dejando espacios sin usar o pérdida, en los cuales no vuelve a insertar piezas, por lo tanto, permanecen hasta el final sin posibilidad de ser eliminados, y BWMV considera pérdida en sus mejores layout, mientras que C y N fueron diseñados para que no existiese pérdida en la solución óptima, por lo tanto basta con que BL deje un pequeño espacio para que el óptimo no pueda ser alcanzado, además se puede ver como en la medida que aumenta la cantidad de piezas, la calidad de la solución disminuye, aumentando el porcentaje de desviación de la mejor solución conocida (%GAP).

5.2. ANALISIS DE RESULTADOS PARA EL PCPGBR

Para probar los algoritmos AGPC y AG se emplean 3 grupos de instancias GT1 (Fayard (Fayard, Hifi y Zissimopoulos 1998) y otras generadas aleatoriamente), GT2 (Cung (Cung, Hifi y Cun 2000) y otras generadas aleatoriamente) y GT3 (Generadas aleatoriamente (Sepúlveda 2011)). Además los resultados son comparados con los obtenidos por GRASP (Alvarez-Valdés et al. (Alvarez-Valdés, Parreño y Tamarit 2008)).

5.2.1. Grupo de instancias GT1

El AGPC encuentra, en promedio, mejor calidad de soluciones que AG para todas las instancias del grupo GT1, tal como se muestra en la Tabla 14.

Las mejores soluciones obtenidas por el AG son mejores que las obtenidas por el AGPC para la mayoría de las instancias, específicamente, AG obtiene 8 mejores valores, AGPC 3 y 2 empates.

El AGPC es capaz de encontrar una solución de igual calidad a las mejores conocidas para una instancia, mientras que no puede hacerlo para las demás, el AG no es capaz de encontrar la mejor solución para las instancias del grupo GT1.

AGPC es capaz de encontrar soluciones de mejor calidad que GRASP para una de las instancias, mientras que AG estuvo por debajo de GRASP en todas ellas, el promedio de GAP para todas las instancias fue de 0.003% para GRASP, seguido de AGPC con 0.02% y AG con 0.04%.

5.2.2. Grupo de instancias GT2

El AGPC encuentra, en promedio, mejor calidad de soluciones que AG para la mayoría de las instancias del grupo GT2, tal como se muestra en la Tabla 15, al comparar los resultados *Mean* de AGPC y AG, el primero obtiene un mejor valor para 18 de las 25 instancias, mientras que AG lo hace para una instancia y en las 6 restantes obtienen el mismo promedio.

Las mejores soluciones obtenidas por AGPC y AG son de igual calidad para la mayoría de las instancias, específicamente, AG es mejor en 7 de las instancias, AGPC en 7 y empatan en las 11 restantes.

El AGPC y AG son capaces de encontrar soluciones de mejor o igual calidad a la mejor solución conocida para 7 de las instancias del grupo GT2, específicamente para STS2s, CHL3s, CHL4s, CHL5, Hchl1, Hchl2 y Hchl8s.

AGPC obtiene soluciones de mejor calidad para las instancias de este grupo que GRASP y AG, específicamente, AGPC es mejor que GRAPS para 10 de las 25 instancias, y AG lo hace para 7 de las 25, por esto AGPC obtiene el mejor promedio de GAP para todas las instancias de este grupo con un -0.03%, seguido de AG con -0.02% y GRASP con 0.01%, el que el porcentaje de GAP sea negativo para AGPC y AG significa que mejoran la calidad de las mejores soluciones conocidas.

5.2.3. Grupo de instancias GT3

El AGPC encuentra, en promedio, mejor calidad de soluciones que AG para todas las instancias del grupo GT3, tal como se muestra en la Tabla 16, donde se presentan los resultados obtenidos para este grupo de instancias y se puede ver que al comparar los resultados *Mean* de AGPC y AG, el primero obtuvo un mejor promedio para las 10 instancias.

Las mejores soluciones obtenidas por el AG son mejores que las del AGPC para la mayoría de las instancias, específicamente, AG logra un mejor valor para 6 de las 10 instancias y AGPC para las 4 restantes.

El AGPC y AG no son capaces de encontrar soluciones de calidad igual o superior a las conocidas.

El AGPC y AG no son capaces de mejorar o igualar la calidad de las soluciones obtenidas por GRASP para ninguna de las instancias del grupo GT3, por lo que éste fue el que obtuvo el mejor promedio de GAP con un 0.02% seguido de AGPC con un 0.04% y AG con un 0.05%.

El AGPC encuentra, en promedio, mejores soluciones para la mayoría de las instancias seleccionadas para el PCPGBR, específicamente AGPC obtiene un mejor *Mean* que AG para 41 de las 48 instancias, es decir, para el 85.4%, el AG es mejor en solo una de las instancias, equivalente a un 2.1% y empatan en 6, es decir, un 12.5%.

Las mejores soluciones de AG son mejores que las de AGPC para la mayoría de las instancias seleccionadas, específicamente, AGPC obtiene un mejor resultado que AG para 17 de las 48 instancias, o sea, un 35.4%, AG para 19 instancias, es decir, 39.6% y empatan en 12 instancias, equivalentes a un 25%. Nuevamente se ve esta diferencia, al igual que en el PET, ya que el AGPC muestra ser más estable en los resultados que obtiene, logrando de ese modo un mejor promedio que el AG, mientras que este ultimo logra mejores resultados, pero de forma esporádica.

El AGPC obtiene un mejor porcentaje de GAP que AG y GRASP si se consideran todas las instancias, independientemente del grupo al que pertenecen, específicamente con un GAP promedio de -0.0001% para AGPC, seguido GRASP con 0.0106% y AG con 0.0116%, es decir, AGPC resulto ser el algoritmo más competitivo considerando las 48

instancias evaluadas, logrando mejorar los resultados conocidos para 4 e igualando el mejor para 3 de las instancias de GT2 y mejorando uno en GT1.

La calidad de las soluciones obtenidas por AGPC es más estable que la calidad de AG, esto se puede deber a la cooperación existente entre los AG locales, dado que basta con que uno de los AG encuentre un buen valor para que este se comparta a los demás, al momento de migrar, mejorando la calidad de la soluciones transversalmente, aumentando la probabilidad de converger a mejores soluciones, mientras que sin cooperación, el algoritmo es capaz de obtener muy buenos resultados, pero en otros casos, la solución encontrada esta lejos de la mejor obtenida. Debido a esto, se puede ver que la cooperación ayuda a mejorar la calidad de la solución, aumentando la estabilidad y probabilidad de obtener resultados competitivos.

No se ve una influencia de la cooperación en la convergencia de los algoritmos, dado que, al analizar las curvas de convergencia de ambos problemas, se puede ver como en algunos casos el AGPC converge antes a la mejor solución y en otros lo hace antes el AG, sin mostrar una tendencia clara.

La cooperación ayuda a que aumente la probabilidad de converger a buenas soluciones, se puede ver como la cooperación ayuda a que las mejores soluciones de cada nodo AG se normalicen en cada migración, mejorando la mejor solución global y de ese modo ayudando a que las poblaciones sean capaces de evolucionar a mejores soluciones, tal como se puede ver en las tablas presentadas, logrando de ese modo ser más estable en las soluciones encontradas.

5.3. SPEED UP

El *Speed Up* encontrado se sub-lineal, pero mayor a 1, por lo tanto si se produce una mejora en los tiempos de ejecución gracias al paralelismo y cooperación, tal como se puede ver en la Tabla 18, el *Speed Up* promedio para el grupo de instancias C fue 3.45, y según lo planteado por Alba (E. Alba 2005), corresponde a un *Speed Up* sub-lineal, dado que 3.45 es menor que 4, al igual que el grupo de instancias GT1 (Tabla 19) cuyo *Speed Up* promedio fue de 2.574, también menor que 4, pero ambos valores son mayor a 1.

CAPITULO 6. CONCLUSIONES

El objetivo principal de este trabajo es modelar y resolver computacionalmente problemas de corte y empaque, específicamente los problemas de Corte de Piezas Guillotinable Bidimensional Restricto y de Empaque de Tiras, mediante un algoritmo genético en 2 versiones, una secuencial y otra paralela basada en cooperación, con la hipótesis de que la cooperación y el paralelismo ayudan a mejorar el desempeño del algoritmo, en términos de tiempo computacional y calidad de la solución al resolver instancias típicas de prueba. Para esto se propuso una representación binaria para ambos problemas, en la que se introducen algunos bits para decodificar el orden en que deben ser puestas las piezas, esto constituye un avance ya que en trabajos anteriores se había utilizado una representación binaria para el problema de corte de piezas, pero que no era capaz de recorrer todo el espacio de soluciones ya que no consideraba el orden en la decodificación del cromosoma, quedando sujeto a un ordenamiento arbitrario y que no era parte de la evolución, y para el problema del PET se había usado representaciones numéricas (Enteros), por tanto, en la representación binaria realizada se encuentra un aporte al trabajo con algoritmos genéticos.

Para estudiar si se comprueba o no la hipótesis planteada se utilizó 534 instancias del PET y 48 para el corte de piezas, por lo tanto se realizaron 5340 pruebas por cada algoritmo (AG y AGPC) para las instancias del PET y 480 para el corte de piezas, es decir, en total los algoritmos fueron ejecutados 11640 veces.

La calidad de las soluciones obtenidas se ve afectada por el algoritmo de encaje empleado, al considerar el desempeño del AGPC y AG con las instancias de prueba para *el* PET, se puede ver como ambos algoritmos tienen dificultades con las instancias sin pérdida, es decir, que el layout óptimo no considera espacios sin piezas, y por el contrario, muestran un buen desempeño con aquellas que si consideran pérdida en sus mejores soluciones, mejorando incluso algunas de las mejores soluciones hasta ahora conocidas, teniendo esto en cuenta y que la heurística de encaje empleada para este problema fue

Bottom-Left, se comprueba, en primer lugar, que este tiene dificultades para posicionar piezas sin pérdida, dado que por su algoritmo tiende a dejar espacios en la medida que va construyendo el layout y, en segundo lugar, que la calidad de las soluciones obtenidas esta asociada al algoritmo de encaje, quedando abierta la puerta para que se estudie si cambiando el algoritmo de encaje por uno que maneje mejor los espacios sin usar o que posicione las piezas de forma más eficiente, se mejoren los resultados obtenidos para las instancias consideradas, sin necesidad de cambiar la representación del cromosoma. Además en la medida que aumenta la cantidad de piezas a ubicar en la tira, aumenta el GAP, es decir, baja la calidad de la solución.

Se comprueba la hipótesis respecto a la calidad de la solución, ya que en el 99.4% de las instancias del PET y el 85.4% de las instancias de corte de piezas, el AGPC obtiene un mejor promedio en las ejecuciones realizadas (*Mean*) para obtener las soluciones a las instancias de prueba, muestra, de ese modo ser más estable y competitivo que la versión secuencial, considerando que estos algoritmos son no-determinísticos se puede dar que en una ejecución encuentre un resultado óptimo, pero que en las siguientes no sea capaz de encontrarlo e incluso obtener malas soluciones, por tanto, el valor que mejor describe el desempeño de un algoritmo de este tipo es el promedio (*Mean*) y no tan solo el mejor resultado obtenido (*Best*).

No es posible determinar una tendencia clara en las curvas de convergencia, por tanto, no se puede concluir que la cooperación ayuda a acelerar la convergencia del algoritmo.

La cooperación, a través de las migraciones, ayuda a propagar las mejores soluciones haciendo que la calidad de las poblaciones en cada nodo del AGPC mejore, aumentando de ese modo la estabilidad en cuanto a la calidad de la solución obtenida por el algoritmo, tal como se comprueba con los resultados presentados en las Tabla 10 a la Tabla 16, es decir, la cooperación ayuda a la estabilidad de las soluciones del algoritmo haciendo que este converja a buenas soluciones más frecuentemente que sin ella, considerando que el *Mean* de AGPC fue mejor que el del AG en el 99.4% y el 85.4%, para las instancias del PET y PCPGBR, respectivamente.

El *Speed Up* fue sub-lineal, es decir, la razón entre el tiempo empleado por un procesador y el empleado por 4 procesadores fue menor a 4 y mayor a 1, por lo tanto se

logra una mejora en cuanto al tiempo computacional requerido para ejecutar los algoritmos, comprobándose la hipótesis relacionada con el mejor desempeño en términos de tiempo computacional del AGPC *versus* el AG, ya que si bien el *Speed Up* fue menor al número de procesadores, este sigue siendo mayor a 1, por tanto existe una mejora en cuanto al tiempo requerido para ejecutar el algoritmo.

Para el problema del PET, ambas versiones del algoritmo resultan ser las menos competitivas para las instancias evaluadas, mientras que para el problema de corte de piezas el AGPC obtiene el mejor promedio de GAP, considerando los resultados obtenidos para todas las instancias evaluadas, es decir, en promedio obtuvo las mejores soluciones, seguido de GRASP y finalmente AG.

Además se provee de un *software web* para dibujar los *layouts* obtenidos por el AG y AGPC para los problemas estudiados, este *software* es almacenado y publicado haciendo uso de los servidores del Departamento de Ingeniería Informática de la Universidad de Santiago de Chile.

Como trabajos futuros se puede desarrollar una demostración de que la representación binaria propuesta es capaz de obtener todas las combinaciones posibles, también mejorar la implementación de esta representación para hacer un mejor uso de los recursos computacionales y como consecuencia una mejora en tiempo computacional.

Además se puede implementar un mejor algoritmo de encaje para el PET, remplazando *Bottom-Left* por uno que ubique las piezas de forma más eficiente, sin empeorar la complejidad del algoritmo.

Aplicar la representación binaria propuesta a otros problemas de optimización, tales como, el vendedor viajero.

REFERENCIAS BIBLIOGRAFICAS

Alba, E., y Troya J. «Influence of the Migration Policy in Parallel Distributed GAs with Structured and Panmictic Populations.» *Applied Intelligence*, nº 12 (2000): 163-181.

Alba, Enrique. *Parallels Metaheuristics, A New Class of Algorithms.* New Jersey, EE.UU.: Wiley-Interscience, 2005.

Álvarez-Valdés, R., A. Parajón, y J.M. Tamarit. «A tabu search algorithm for large-scale guillotine (un) constrained two-dimensional cutting problems.» *Computers & Operations Research*, nº 29 (2002): 925–947.

Alvarez-Valdés, R., F. Parreño, y J.M. Tamarit. «Reactive GRASP for the strip-packing problem.» *Computers & Operations Research*, nº 35 (2008): 1065-1083.

Ashlock, Daniel. *Evolutionary Computation for Modeling and Optimization.* Ontario, Canada: Springer, 2005.

Baker, B.S., E.G. Coffman, y R.L. Rivest. «Orthogonal packings in two dimensions.» *SIAM Journal on Computing*, nº 9 (1980): 846–855.

Belov, G., G. Scheithauer, y E.A. Mukhacheva. «One-dimensional heuristics adapted for two-dimensional rectangular strip packing. .» *Journal of the Operational Research Society*, nº 59 (2008): 823–832.

Berkey, J.O., y P.Y. Wang. «Two-dimensional finite bin packing algorithms.» *Journal of the Operational Research Society*, nº 38 (1987): 423–429.

Bortfeldt, A. «A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces.» *European Journal of Operational Research*, nº 172 (2006): 814-837.

Burke, E., G. Kendall, y G. Whitwell. «A new placement heuristic for the orthogonal stock-cutting problem.» *Operations Research*, nº 52 (2004): 697-707.

Burke, E., G. Kendall, y G. Whitwell. «A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock-Cutting Problem.» *Informs Journal on Computing*, nº 21 (2009): 505-516.

Burke, E., G. Kendall, y G. Whitwell. «Metaheuristic enhancements of the best-fit heuristic for the orthe orthogonal stock-cutting problem.» *Technical Report*, 2006.

Burke, E., G. Kendall, y M. R. Hyde. «A squeaky wheel optimisation methodology for two-dimensional strip packing.» *Computers & Operations Research*, nº 38 (2011): 1035-1044.

Burke, E., G. Kendall, y M. R. Hyde. «Evolving bin packing heuristics with genetic programming.» *Parallel Problem Solving from Nature-PPSN* IX (2006): 860–869.

Castrillón, M. *Efectos de la Cooperación en Algoritmos Genéticos Paralelos.* Santiago: Memoria de Ingenieria Civil Informatica, Departamento de Ingenieria Informatica, Facultad de Ingenieria, Universidad de Santiago de Chile, 2010.

Charalambous, C., y K. Fleszar. «A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts.» *Computers & Operations Research*, nº 38 (2011): 1443–1451.

Charalambous, Christoforos, y Krzysztof Fleszar. «A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts.» *Computers & Operations Research*, nº 38 (2011): 1443-1451.

Chazelle, B. «The bottom left bin packing heuristic: an efficient implementation.» *IEEE Transactions on Computers*, nº 32 (1983): 697–707.

Cung, V., M. Hifi, y B. Cun. «Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm.» *International Transactions in Operational Research*, nº 7 (2000): 185–210.

Darwin, C. *On the origin of the species by natural selection (Murray.).* Londres, Reino Unido, 1859.

Dyckhoff, H. «A typology of cutting and packing problems.» *European Journal of Operational Research* 44, nº 2 (1990): 145-159.

Fayard, D., M. Hifi, y V. Zissimopoulos. «An efficient approach for large-scale two-dimensional guillotine cutting stock problems.» *Journal of the Operational Research Society*, nº 49 (1998): 1270–1277.

Goldberg, David Edward. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, 1989.

Hifi, Mhand, y Rym M'Hallah. «Strip generation algorithms for constrained two-dimensional two-staged cutting problems.» *European Journal of Operational Research*, nº 172 (2006): 515-527.

Holland, John Henry. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 1975.

Hopper, E., y B. C. H. Turton. «An empirical investigation on metaheuristic and heuristic algorithms for a 2d packing problem.» *European Journalof Operational Research*, nº 128 (2001): 34-57.

Hutter, Frank, Holger Hoos, Kevin Leyton-Brown, y Thomas Stützle. «ParamILS: An Automatic Algorithm Configuration Framework.» *Journal of Artificial Intelligence Research*, nº 36 (2009): 267-306.

Imahori, S, M Yagiura, y T Ibaraki. «Improved local search algorithms for the rectangle packing problem with general spatial costs.» *European Journal of Operational Research*, nº 167 (2005): 48–67.

Jakobs, S. «On genetic algorithms for the packing of polygons.» *European Journal of Operational Research*, nº 88 (1996): 165–181.

Jara, J. L. *Estudio de Modelos Paralelos para Algoritmos Genéticos con Memoria Compartida, Tesis de maestría no publicada.* Santiago, Chile: Universidad de Santiago de Chile, 1996.

Koza, J. R. «Genetic programming: on the programming of computers by means of natural selection.» *The MIT press*, 1992.

Lesh, N., J. Marks, A. Mc Mahon, y M. Mitzenmache. «Exhaustive approaches to 2D rectangular perfect packings.» *Information Processing Letters*, nº 90 (2004): 7-14.

Lesh, N., J. Marks, A. Mc Mahon, y M. Mitzenmache. «New heuristic and interactive approaches to 2D rectangular strip packing.» *ACM Journal of Experimental Algorithmics*, nº 10 (2005): 1-18.

Lesh, N., y M. Mitzenmacher. «Bubble search: a simple heuristic for improving priority-based greedy algorithms.» *Information Processing Letters*, nº 97 (2006): 161-169.

Lodi, A., S. Martello, y D. Vigo. «Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems.» *INFORMS Journal on Computing*, nº 11 (1999): 345–357.

Luke, S., y et al. «Genetic programming produced competitive soccer softbot teams for robocup97.» *Genetic Programming*, 1998: 214–222.

Macedo, Rita, Cláudio Alves, y J.M. Valério de Carvalho. «Arc-flow model for the two-dimensional guillotine cutting stock problem.» *Computers & Operations Research*, nº 37 (2010): 991-1001.

Martello, S., M. Monaci, y D. Vigo. «An exact approach to the strip-packing problem.» *INFORMS Journal of Computing*, nº 15 (2003): 310–319.

Mumford-Valenzuela, C., J. Vick, y P.Y. Wang. «Heuristics for Large Strip Packing Problems with Guillotine Patterns: An Empirical Study.» *Kluwer Academic Publishers*, 2003: 501-522.

Nowostawski, M., y R Poli. «Parallel genetic algorithm taxonomy.» *Proceedings of the Third International*, 1999: 88–92.

Pinto, E., y J.F. Oliveira. «Algorithm based on graphs for the non-guillotinable two-dimensional packing problem.» *Second ESICUP Meeting.* Southampton, 2005.

Riff, M. C., X. Bonnaire, y B. Neveu. «A revision of recent approaches for two-dimensional strip-packing problems.» *Engineering Applications of Artificial Intelligence*, nº 22 (2009): 823-827.

Romero, F. *Un Algoritmo Genético Paralelo para Resolver el Problema de Corte de Piezas Guillotina Bidimensional Restricto.* Santiago, Chile: Tesis de maestría no publicada. Universidad de Santiago de Chile., 2003.

Sastry, K., D. E. Goldberg, y G. Kendall. «Genetic Algorithms: A tutorial.» En *Introductory Tutorials in Optimization*, de E. Burke y G. (Eds.) Kendall, 97-125. Berlin, Alemania: Sprinter, 2005.

Sepúlveda, M. *Generación automatica de algoritmos para problemas de optimización combinatoria.* Santiago: Tesis de Doctorado en Ciencias de la Ingenieria con Mención en Automática, Departamento de Ingenieria Informatica, Facultad de Ingenieria, Universidad de Santiago de Chile, 2011.

Soke, A., y Z. Bingul. «Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems.» *Engineering Applications of Artificial Intelligence*, nº 19 (2006): 557-567.

Stephen, C.H., Defu Zhang, y Kwang Mong Sim. «A two-stage intelligent search algorithm for the two-dimensional strip packing problem.» *European Journal of Operational Research*, nº 215 (2011): 57-69.

Tiwari, S., y N. Chakraborti. «Multi-objective optimization of a two-dimensional cutting problem using genetic algorithms.» *Journal of Materials Processing Technology*, nº 173 (2006): 384-393.

Toro, Carolina. *Resolución del problema de Strip-Packing mediante una máquina evolutiva paralela basada en Algoritmos Genéticos.* Santiago: Memoria de Ingenieria Civil Informatica, Departamento de Ingenieria Informatica, Facultad de Ingenieria, Universidad de Santiago de Chile, 2011.

Valério de Carvalho, J.M. «Exact solution of bin-packing problems using column generation and branch-and-bound.» *Annals of Operations Research*, nº 86 (1999): 629-59.