Capítulo 1. Introducción

Antecedentes Generales

Los problemas de corte y empaque son problemas de optimización cuyo interés es encontrar una buena distribución de múltiples ítems (piezas) contenidos en una gran región (objetos, láminas). Este tipo de problemas se encuentra en muchas áreas de negocios y de la industria y forma parte de los problemas combinatoriales encontrados en la investigación de operaciones. El proceso de colocación de piezas está descrito por un conjunto de reglas o restricciones. El objetivo del proceso es maximizar la utilización de material y, por lo tanto, minimizar el área de pérdida. Éste es el interés particular de las industrias involucradas con producción en serie, ya que pequeñas mejoras en el "*layout*" pueden resultar en ahorros de material y considerables reducciones en los costos de producción. La complejidad del problema y el método de solución dependen de la geometría de los objetos y las restricciones impuestas.

Para la solución de pequeños problemas de empaque, se han desarrollado métodos determinísticos, tales como programación lineal. Debido a que esos métodos son exactos, ellos encuentran la solución

óptima. Para problemas de mayor complejidad, el espacio de solución es mucho mayor debido al mayor aumento en el número de combinaciones posibles. En este caso, no es posible calcular la solución óptima en un tiempo computacional razonable. De modo de poder resolver esos problemas, se han usado técnicas heurísticas para encontrar soluciones cercanas al óptimo con costos computacionales razonables. Las soluciones deben estar dentro de un rango de desviación tolerable con respecto a la solución óptima. La forma de cómo tiene lugar la colocación de piezas está descrita por un conjunto de reglas. Dado que esas reglas están hechas a la medida, para una tarea de empaque particular, los algoritmos heurísticos específicos usados para resolver el problema pueden sólo ser aplicados exitosamente a ese tipo particular de problema.

Una mayor flexibilidad es brindada por métodos heurísticos generales, también llamados meta-heurísticas (Díaz et al, 1996), los cuales describen principios de búsqueda generales más que reglas especiales. La búsqueda a través de grandes espacios de soluciones es guiada por información específica del problema. La calidad de la solución depende de la implementación de este conocimiento y de los parámetros aplicados para controlar el proceso de búsqueda. El éxito de las meta-heurísticas puede ser explicado por su gran flexibilidad en tomar en cuenta las restricciones específicas del problema y su buen compromiso entre la calidad de la solución y el esfuerzo computacional, contrario a una búsqueda exhaustiva. Dentro de las meta-heurísticas más importantes se

encuentran los Algoritmos Genéticos, Recocido Simulado, Búsqueda Tabú y Redes Neuronales Artificiales.

Muchos problemas de la investigación de operaciones e inteligencia artificial pueden ser definidos como problemas de optimización combinatoria. Entre éstos, el área de problemas de cortes de piezas ha sido extensamente estudiada en el pasado. Estos problemas tienen muchas aplicaciones en procesos de producción en la industria de corte de papel, vidrio, metal y madera.

El Problema de Corte de Piezas radica en la necesidad de minimizar la pérdida de material, cuando se desea cortar sobre una placa, una cantidad determinada de piezas. En los años 50, Kantorovich y Zalgaller (1951) y Kantorovich (1960) propusieron la primera formulación conocida del problema de corte de piezas. Sin embargo, en los años sesenta, Gilmore y Gomory (1961, 1963, 1966) realizaron el primer avance significativo en la búsqueda de una solución al problema, por medio de una técnica de generación de patrones, basada en programación lineal, para solucionar el problema unidimensional de minimización de pérdidas por corte.

Con el pasar del tiempo, el problema de corte de piezas está cada vez más presente en diferentes sectores industriales, por lo que existe gran interés en encontrar procedimientos cada vez más efectivos y que ocupen menor tiempo de procesamiento. Sweeney y Paternoster (1991) han identificado más de quinientos artículos que tratan del problema de

corte de pieza y sus aplicaciones. Sobresalen los trabajos de Wang (1983); Viswanathan y Bagchi (1988); Oliveira y Ferreira (1990); Parada, Gómes y de Diego (1995); Parada, Muñoz y Gómes (1995). Hinxman (1980) y Dyckhoff (1990) han realizado una revisión de todos los estudios registrados en la literatura y establecieron una clasificación para los problemas de corte de piezas y empaquetamiento.

Entre otras, existen tres categorías en que se pueden clasificar los problemas de corte de piezas:

- **Coordenadas de los cortes.** Los cortes se pueden realizar en una, dos o tres dimensiones.
- **Geometría de los cortes.** Dependiendo del tipo de material y del uso que tienen las piezas cortadas. Los cortes pueden ser con formas geométricas definidas (regulares) o sin un patrón definido (irregulares).
- **Forma en que se realizan los cortes.** Los cortes pueden ser realizados por distintas herramientas que restringen la forma del corte.

El problema a estudiar pertenece a un caso particular de la familia de problemas de corte de piezas, siendo éste el corte de piezas rectangulares, desde láminas también rectangulares, satisfaciendo una demanda establecida en base al número de láminas a cortar, de acuerdo a cada patrón definido. El objetivo del proceso de corte es que se efectúe de forma tal que el residuo de material cortado sea mínimo. Las siguientes condiciones tecnológicas permiten reducir todavía más el dominio del problema:

- Considerar un límite superior en el número de veces que un determinado tipo de pieza pueda ser cortado de una misma lámina.
- Realizar todos los cortes con guillotina, es decir, cortar cada vez la lámina o partes de ella de lado a lado, de forma ortogonal a alguno de los lados de la lámina.

El problema de corte de piezas es "*NP-Hard*" (Garey y Jonson, 1979). Es decir, no se conoce un algoritmo que encuentre una solución óptima en tiempo polinomial. Fowler, Paterson y Tatimoto (1981) han mostrado que los problemas de empaque rectangular, o mejor dicho, su problema de decisión asociado es NP-Completo (Garey y Jonson, 1979).

Los Algoritmos Genéticos (AGs) son métodos de búsqueda basados en principios de selección natural y genética (Goldberg, 1989a). Ellos han sido aplicados exitosamente a numerosos problemas en negocios,

ingeniería y ciencia (Goldberg, 1994). En muchas aplicaciones prácticas, los AGs encuentran buenas soluciones en tiempos razonables. Sin embargo, en algunos casos, los AGs pueden requerir cientos de miles de costosas funciones de evaluación y, dependiendo del costo de cada evaluación, “un AG puede tomar días, meses, o aún años en encontrar una solución aceptable” (Cantú-Paz, 2000).

Uno de los principales inconvenientes de su utilización es el tiempo de cómputo requerido para obtener una solución satisfactoria. Se han propuesto esquemas que permiten la implementación de estos algoritmos en múltiples procesadores, con lo que se puede obtener una reducción substancial de este tiempo. Éstos, tienen la desventaja de requerir mucho tiempo de procesador. La valorización de la función de evaluación suele tomar, en muchos casos, un tiempo considerable. Esta función debe calcularse para cada nuevo cromosoma generado y, además, por varias generaciones.

Afortunadamente los AGs son intrínsecamente paralelos. La evaluación de esta función se puede realizar de manera concurrente para diferentes cromosomas, si se dispone de múltiples procesadores y de un canal de comunicación entre ellos. A las implementaciones de los algoritmos genéticos que buscan aprovechar la disponibilidad de múltiples procesadores se les denomina Algoritmos Genéticos Paralelos (AGPs).

Finalmente, se plantea como hipótesis que es posible implementar un algoritmo genético paralelo que permita resolver el Problema de Corte de Pieza Guillotina Bidimensional Restricto (PCPGBR), mediante el cual se mejore (versus su contraparte secuencial) tanto la calidad de la solución encontrada, como el tiempo computacional invertido.

Motivación de este trabajo

Desde hace mucho tiempo que se ha estudiado el problema de corte de piezas y es razonable preguntarse por que hay todavía interés en seguir estudiando el problema. La respuesta se fundamenta en el hecho que hay nuevas técnicas que pueden aplicarse en la búsqueda de buenas soluciones de calidad para el problema.

El problema de corte de piezas ha sido estudiado rigurosamente mediante varias meta-heurísticas en su concepción secuencial (Muñoz, 1994; Muñoz, 1995; Parada, Gómes y de Diego, 1995; Parada, Muñoz y Gómes, 1995; Parada, Sepúlveda, Solar y Gómes, 1997; Oliveira y Ferreira, 1990; Rojas, 2000; Rojas, 2001; Wang, 1983). No son muchos los trabajos que resuelven el problema mediante algoritmos paralelos. Algunos de ellos son Holthöfer y Tschöke (1995); Nicklas, Atkins, Setia y Wang (1998). De hecho, es la primera vez que se utilizan Algoritmos Genéticos Paralelos para la resolución del problema de corte de piezas.

Definición de Objetivos

Objetivos Generales

Proponer un modelo paralelo basado en Algoritmos Genéticos para solucionar el Problema de Corte de Piezas Guillotinadas Bidimensional Restricto.

Comparar, en términos de desempeño computacional, el algoritmo propuesto con su contraparte secuencial.

Objetivos Específicos

Realizar una revisión bibliográfica acerca del Problema de Corte de Piezas Guillotinadas Bidimensional Restricto.

Realizar una revisión bibliográfica del método Algoritmos Genéticos Paralelos.

Proponer un algoritmo paralelo basado en Algoritmos Genéticos que resuelva el problema planteado.

Modelar y resolver el problema usando el modelo propuesto.

Realizar un experimento numérico que permita comparar los resultados con los obtenidos por medio de Algoritmos Genéticos Secuenciales.

Organización del Documento

Este documento se divide en cuatro secciones: presentación del problema de corte de piezas, descripción de los Algoritmos Genéticos Secuenciales y Paralelos, modelamiento del algoritmo propuesto y los experimentos numéricos realizados.

En el Capítulo 2 se presenta el problema de corte de piezas y una revisión bibliográfica del problema.

El Capítulo 3 entrega una descripción de los Algoritmos Genéticos Secuenciales y de los Algoritmos Genéticos Paralelos. Se presenta una clasificación de los Algoritmos Genéticos Paralelos basada en Cantú-Paz

(2000). Además, se describen algunos conceptos de Computación Paralela para una mejor compresión de las herramientas utilizadas.

En el Capítulo 4 se describe detalladamente el modelo propuesto que permite resolver el Problema de Corte de Piezas Guillotinadas Bidimensional Restricto, basándose en las capacidades de los Algoritmos Genéticos Paralelos.

En el Capítulo 5 se detallan los experimentos realizados y los resultados obtenidos al evaluar el Modelo Paralelo. Se consideran nueve problemas para realizar las pruebas. Para una instancia de prueba, se muestra con detalle la evolución que presenta cada uno de los agentes que participan en la resolución del problema propuesto. Lo anterior, se realiza con el fin de mostrar el real potencial del Algoritmo Genético Paralelo diseñado.

Finalmente, en el Capítulo 6 se presentan las conclusiones del trabajo realizado, donde se analiza la obtención de un "*Speedup*" Superlineal para el modelo paralelo propuesto. Por otro lado, se indica la superioridad, en cuanto a los tiempos de respuesta, del modelo paralelo asíncrono sobre el modelo paralelo síncrono.

Capítulo 2. Presentación del Problema

Introducción

El problema de corte de existencias (*cutting stock problem*) ha sido un área de investigación desde el año 1940, aunque la principal investigación empezó en los comienzos de 1960. Durante los últimos cincuenta años se han desarrollado muchas técnicas para producir buenos patrones de corte para una variedad de problemas.

Debido a que el problema es NP-Completo (Garey y Jonson, 1979), técnicas exactas como la programación lineal no pueden usarse para grandes instancias del problema. En los años recientes se han aplicado técnicas basadas en heurísticas para resolver el problema y es probable que ésta continúe siendo un área activa de investigación durante muchos años. Esto no sólo es debido al hecho que se han utilizado métodos basados en heurísticas para producir soluciones de buena calidad para el problema, sino que también se debe al hecho que están investigándose nuevos métodos heurísticos todo el tiempo. Si bien es cierto los Algoritmos Genéticos no son una meta-heurística nueva, es la primera vez que se

implementa un Algoritmo Genético Paralelo para resolver este tipo de problema.

¿Cuál es el Problema?

Dyckhoff (1990) da ejemplos en informática, investigación de operaciones, ingeniería, manufactura, y otras disciplinas donde el problema de corte y empaque necesita ser resuelto. Esto ha conducido a que los mismos problemas sean referenciados de muchas maneras (por ejemplo, empaque de depósitos, el problema de disminución de pérdidas y empaque unidimensional). Dyckhoff (1990) sugirió una topología para problemas de este tipo.

A continuación se mencionan algunos términos que pueden usarse para describir problemas de corte y empaque (*cutting and packing problems*).

- **Problema de distribución** (*assortment problem*): Este problema se preocupa en decidir el nivel de stock de piezas a mantener (y entonces usar) de modo que los objetos sean cortados eficientemente.
- **Problema de empaque de depósitos** (*bin packing problem*): Asignación de ítems en depósitos, minimizando la altura de los depósitos. Vea también, el problema de la mochila.
- **Problema de corte de existencias** (*cutting stock problem*): Normalmente se usa como término genérico para la clase completa de problemas de corte y empaque. Estrictamente, el

término debe usarse para referirse a un problema donde tanto el stock de láminas como las partes a ser cortadas son rectangulares.

- **Problema de diseño** (*layout problem*): Se le denomina *layout* a un arreglo de dibujos (formas) en una superficie. Normalmente, se aplica a la industria de ropa.

- **Problema de carga** (*loading problem*): Generalmente se refiere a problemas tridimensionales como carga de camiones o contenedores.
- **Problema de la mochila** (*knapsack problem*): Determina una combinación de un conjunto de objetos, donde cada uno tiene un valor asociado, tal que una función es minimizada. Este problema puede verse como un problema de empaque de depósito con algunas restricciones adicionales sobre el valor asignado a cada pieza.
- **Problema de la N-partición** (*n-partition problem*): Considera todas las posibles combinaciones de un conjunto de números enteros donde la suma debe ser N.
- **Problema de anidamiento** (*nesting problem*): Empaque de formas o figuras irregulares. A menudo usado en la industria de construcción de buques.
- **Problema de empaque ortogonal** (*orthogonal packing problem*): Los rectángulos a ser empacados deben tener sus lados ortogonales al fondo o los bordes verticales al depósito.
- **Problema de diseño de plantilla** (*template layout problem*): Un problema de corte de existencias donde no existe ninguna restricción en el número de formas a ser cortadas.
- **Problema de recorte de pérdida** (*trim loss problem*): Este

tipo de problemas tiene por finalidad minimizar el material desechado después que todas las formas han sido cortadas.

Dimensiones del problema

Un problema de corte de existencias puede categorizarse en una de *n*-dimensiones. Los valores más usuales de *n* son 1, 2 y 3.

El problema unidimensional (1D) puede verse como el corte de piezas de una longitud dada de material. Éste es a menudo referenciado como empaque de tiras (*strip packing*). En este problema sólo es necesario preocuparse de la longitud de las piezas que resultan del proceso de corte. El ancho de las piezas resultantes es fijo. Las aplicaciones típicas para el problema 1D involucran cortes longitudinales de acero desde barras de acero. Muchos problemas 2D pueden convertirse en problemas 1D ignorando el ancho de las piezas a ser cortadas, en donde sólo la altura de las formas es de preocupación.

En el problema de corte bidimensional (2D), el interés es cortar formas de dos dimensiones desde láminas también de dos dimensiones. Las formas pueden ser rectangulares o irregulares. Las aplicaciones típicas incluyen cortes en vidrio, láminas de metal, madera y cartón.

En el problema tridimensional (3D), normalmente, es de interés el empaque de formas tridimensionales dentro de un área dada. Las aplicaciones típicas para problemas 3D incluyen carga de plataformas y de vehículos de reparto.

Podría parecer imposible tener un problema n-dimensional donde n > 3, sin embargo, el problema de la corte de existencias n-dimensional

puede emplearse en aplicaciones que podrían no parecer inmediatamente obvias. Por ejemplo, en un problema de empaque 3D, tal como cargar un vehículo de reparto de mercadería, puede agregarse una dimensión temporal de acuerdo al orden en que la mercadería será descargada. Esto resulta en un intento de empacar las mercaderías de modo que sean fácilmente accesibles en cada punto del reparto.

Tipos de Cortes

Dependiendo de la tecnología usada para efectuar los cortes, el problema de corte de piezas (existencias) presenta dos variantes. La primera considera que los cortes son del tipo guillotina, es decir, se debe cortar cada vez la lámina o partes de ella de lado a lado, de forma ortogonal a alguno de los lados de la lámina. La Figura N°2.1 muestra un ejemplo de dos diseños, uno con corte guillotina y el otro no.

Corte Guillotina Corte No Guillotina

Figura N°2.1: Cortes de piezas Guillotina versus Cortes No Guillotina.

Por ejemplo, cuando se corta vidrio sólo se permiten cortes guillotina, debido a la naturaleza del material. El tipo de herramienta de

corte también puede dictar el tipo de corte que es permitido. Cortes usando una fresadora permiten cortar cualquier patrón (dependiendo del material). Esto se debe al hecho que la fresadora es, en efecto, un pedazo del taladro que puede moverse en cualquier dirección. La desventaja es que la fresadora es más ancha que la hoja de corte, entonces las formas tienen que ser aumentadas en tamaño para acomodar la pérdida producida por la fresadora.

Colocación versus Corte

El último objetivo industrial es recortar un número de formas desde una lámina dada. Sin embargo, para propósitos de investigación, el problema se reduce a menudo a poner las formas simplemente y desatender la operación de corte. Encontrar el mejor diseño es a menudo aceptable pero en algunas situaciones encontrar un buen diseño puede no ser suficientemente bueno para resolver un problema particular. Por ejemplo, una restricción adicional podría ser minimizar el tiempo de corte.

Clasificación

De modo de hacer un intercambio de información entre diferentes disciplinas Dyckhoff (1990) propuso una topología para los problemas de empaque. Los problemas de empaque se forman a partir de la intersección entre la geometría y la combinatoriedad. Los problemas pueden ser divididos entre aquellos que involucran alguna dimensión espacial (concretos) y aquellos que involucran una dimensión no espacial (abstractos). El primer grupo consiste en problemas de empaque y de carga hasta un espacio euclidiano tridimensional. El último grupo de problemas consiste en aquellos que no involucran una dimensión espacial, como por ejemplo, problemas que involucran una dimensión temporal o

una dimensión más abstracta aún como la memoria de una computadora.

Aquellos problemas que operan en el espacio euclidiano pueden ser divididos en problemas de corte y empaque. Los problemas de cortes son aquellos que involucran un gran objeto (lámina, rollo de material, etc.) que debe ser dividido en objetos más pequeños, normalmente representados como un conjunto de arreglos. Los problemas de empaquetado involucran el relleno de depósito con un número de formas dadas. De la descripción anterior puede apreciarse que los problemas de corte normalmente son de dos dimensiones y los problemas de empaque normalmente son tres dimensiones.

Las observaciones anteriores se muestran en la Figura N°2.2 (Hopper, 2000 y basado en Dyckhoff, 1990).

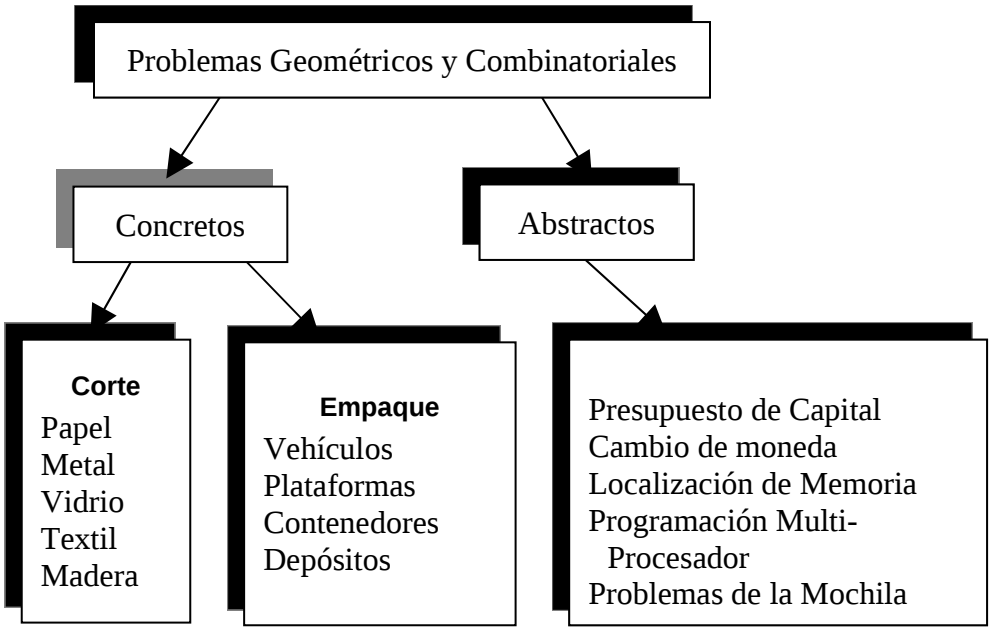


Figura N°2.2: Clasificación para problemas de corte y empaque.

Formulación del Problema

El Problema de Corte de Piezas Guillotinadas Bidimensional Restricto considera una lámina rectangular S de dimensiones (W, L), siendo W el ancho y L el largo, ambas medidas en unidades discretas. Considera, además, un conjunto R de índices que identifican los diversos tipos de piezas con dimensiones (w_i , l_i) tales que $w_i \leq W$ y $l_i \leq L$, $\forall\ i \in R$. Sea b un vector de componentes enteros, con elementos b_i, $i \in R$, que corresponde al límite impuesto sobre el número de veces que cada tipo de pieza puede ser cortado desde una misma lámina. Se considera, adicionalmente, que todos los cortes deben ser del tipo guillotina, es decir, deben ser ortogonales a alguno de los lados de la lámina rectangular.

Con estos antecedentes, se pretende configurar un patrón de corte de dimensiones (w_p, l_p), estableciendo la combinación de piezas que, al ser cortadas desde la lámina, produce la mínima pérdida.

La formulación matemática del problema corresponde al modelo representado en las ecuaciones 2.1 a 2.4.

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

s.a.

$0 \leq x_i \leq b_i$, $\forall\ i \in R$ (2.2)

x_i entero, $\forall\ i \in R$ (2.3)

Cortes Factibles, (2.4)

donde x_i es el número de veces que la pieza tipo i se encuentra en el patrón de corte.

La función objetivo (2.1) representa el área de pérdida después de cortar una configuración de piezas x. La restricción (2.4) involucra tanto el hecho que los cortes sean tipo guillotina, como que el conjunto de piezas que aparece en el patrón puedan ser distribuidas sobre la lámina sin traslape entre las diversas piezas.

La Figura N°2.3 muestra un esquema de un patrón de corte.

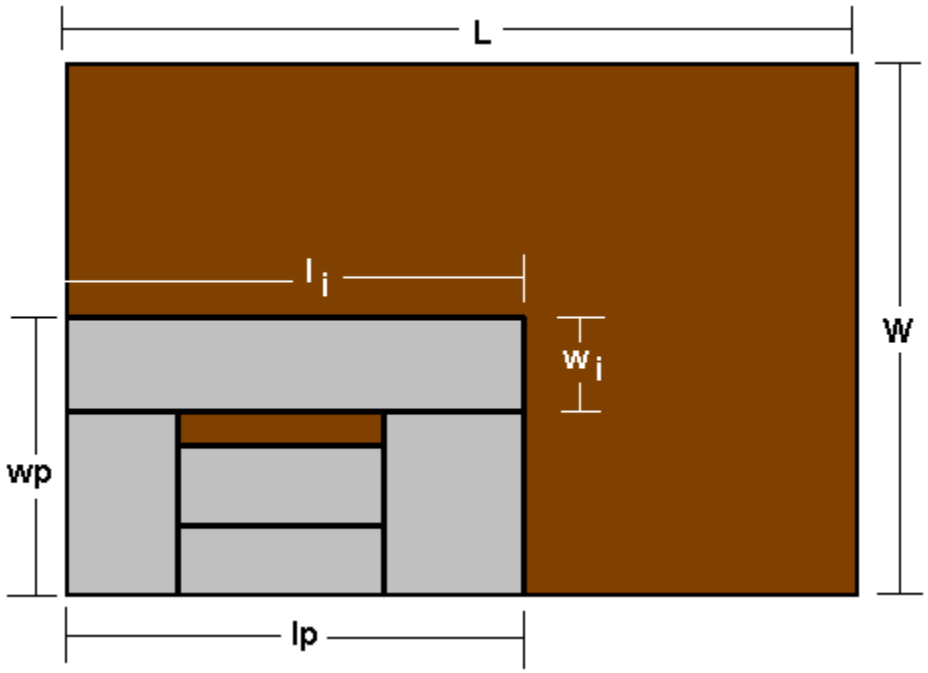


Figura N°2.3: Esquema de un patrón de corte.

Revisión Bibliográfica

Brooks, Smith, Stone, y Tutte (1940) publicaron un artículo que discute cómo dividir un rectángulo en cuadrados. Por más de diez años no hubo publicaciones sobre problemas de corte de existencias (*cutting stock problems*), hasta que Dantzig (1951); Kantorovich y Zalgaller (1951) resaltaron la relación entre el problema de corte de existencias y la teoría de la Programación Lineal (PL).

Paull (1956); Eisemann (1957); Vajda (1958) usaron técnicas de Programación Lineal para resolver una versión restringida del problema de diseño rectangular cuando se cortan rollos de papel.

La mayoría de los trabajos hechos antes de 1960 se preocupaban del papel en la industria manufacturera. Gilmore y Gomory (1961) presentaron un método de programación lineal que, por primera vez, resolvió óptimamente el problema de corte unidimensional. El trabajo de Gilmore y Gomory (1963) va más allá y se aplica al problema de corte de papel. Ellos dan un ejemplo de la complejidad del problema declarando que al enfrentarse con un rollo estándar de papel de 5,08 mts. y una demanda de cuarenta longitudes diferentes que van desde 0,508 mts. a 2,032 mts., el número de posibles patrones de cortes puede exceder los cien millones. Esto significa que ellos se enfrentaron con un problema de Programación Lineal con hasta cien millones de columnas. En su artículo ellos diseñaron un procedimiento de generación de columnas que evitó esa dificultad. El procedimiento involucró resolver el problema de la mochila en cada paso del pivote.

Muchos artículos posteriores se referirían a este trabajo y lo desarrollarían más aún. Gilmore y Gomory (1965) presentaron un método para resolver el problema de corte bidimensional de existencias. Ellos tuvieron que poner algunas restricciones, de lo contrario, el número de columnas en el modelo de Programación Lineal era demasiado grande. Sin embargo, ellos defendieron que muchas industrias tenían esas

restricciones y que su trabajo aún podría usarse para resolver problemas prácticos. Una restricción que ellos pusieron en el problema era que el corte tenía que ser hecho en etapas. En efecto, esto significó sólo permitir cortes guillotina.

Gilmore y Gomory (1966) estudiaron más de cerca el problema de la mochila aplicado al corte de existencias en una y dos dimensiones. En este trabajo, ellos usaron una técnica de programación dinámica.

Barnett y Kynch (1967) mostraron que una versión simplificada del problema de corte de existencias podría resolverse usando algoritmos más simples que aquellos presentados previamente. Su algoritmo permite cortes no guillotina que son a menudo necesarios para obtener una solución óptima. La habilidad de permitir cortes no guillotina fue a expensas de poner restricciones en el tamaño de los rectángulos.

Eilon y Christofides (1971) estudiaron el problema de la mochila como un problema de programación cero-uno y desarrollaron una heurística para resolverlo. Ellos aplicaron ambos programas a los mismos cincuenta problemas. En todos menos dos, el método heurístico encontró la solución óptima, usando significativamente menos tiempo computacional que el método cero-uno. El método heurístico trabajó ordenando ascendentemente con respecto al valor de los objetos a ser empacados y también ordenando ascendentemente con respecto al tamaño del espacio disponible en cada mochila. Entonces, el algoritmo

buscaba a través de ambas listas y asignaba objetos al mejor espacio disponible.

Haessler (1971) indicó que la disminución de pérdida no era el único factor que debía considerarse al intentar diseñar un patrón de corte. Puede darse el caso que un patrón de corte óptimo requiera que los operadores tengan que frecuentemente preparar la máquina para acomodarla a un nuevo patrón. Podría ser preferible tener un patrón de corte con una pérdida mayor pero que resulte en que los operadores no tengan que asistir a menudo a la máquina cortante. Haessler formuló los problemas como un modelo de programación matemático y usó un procedimiento heurístico para resolver el problema.

Herz (1972) usó una técnica de búsqueda recursiva para problemas de corte guillotina. Ésta fue una mejora sobre las aproximaciones exhaustivas del tipo iterativo, pero no fue eficaz para problemas medianos.

Adamowicz y Albano (1976) presentaron un método para ubicar rectángulos que usaba un método de programación dinámica. El método imitaba la manera en que un operador humano pondría los rectángulos. Es decir, poniendo los rectángulos que al menos tienen una dimensión común. Las soluciones no eran a menudo óptimas pero ellas eran generalmente buenas y los tiempos de cómputos eran razonables.

Christofides y Whitlock (1977) sugirieron el método "*Branch and Bound*" para el problema de corte guillotina restricto. Restricto significa que el número de veces que una pieza está contenida en el patrón de

corte es restringido. Ellos consideraron una enumeración de posibles cortes que pueden ser hechos desde una variedad de rectángulos.

El trabajo hecho por Gilmore y Gomory (1961); Gilmore y Gomory (1963) fue mejorado por Haessler (1980) donde se implementó un algoritmo modificado que dio los mismos valores de recorte de pérdida, pero con soluciones de mejores características. La mejora del algoritmo fue a expensas de un aumento despreciable en el tiempo de ejecución.

Albano y Osrini (1980) afirmaron que el problema bidimensional de corte de existencias, aún con restricción de corte guillotina, puede sólo resolverse óptimamente para problemas de tamaño medio. Esto es debido al tiempo y a la complejidad espacial. En su artículo ellos presentan un algoritmo heurístico de búsqueda en árbol. Éste es una mejora y una versión extendida de Adamowicz y Albano (1976). La heurística está basada en colocar los rectángulos en tiras usando una política de colocación "*bottom left*", pero además, los problemas pueden ser divididos en sub-problemas. El artículo muestra cómo se resuelve la generación de sub-problemas por medio de una de seis estrategias. Además, se muestran resultados experimentales, junto con los datos que produjeron esos resultados.

En Albano y Osrini (1980) es una de las primeras veces en que se usó el término Inteligencia Artificial (IA) en relación con el problema de corte de existencias. Albano y Osrini formularon la búsqueda de una solución óptima como una búsqueda heurística y usaron términos que son

comunes hoy en día en el dominio de IA. Por ejemplo, ellos describen que la búsqueda está en un cierto estado y se aplican operadores para moverse de un estado a otro.

Dyckhoff (1981) declaró que el modelo de Programación Lineal (PL) desarrollado por Gilmore y Gomory (1961, 1963, 1965) no trabaja en algunas situaciones. Dyckhoff presentó otro modelo PL que tenía ventajas cuando eran muchos los ítems a ser cortados o cuando había un gran número de ítems en stock. Este modelo también puede tratar con casos donde se considera el recorte de pérdida y puede usarse posteriormente en el proceso de corte.

Otten (1982) generó una estructura de datos llamada "*Slicing Tree Structure*". Ésta permitió representar una serie de cortes como un árbol, donde cada nodo representa un corte de la lámina. Aunque la idea se propuso primero con el diseño de planos en mente, la estructura se ha usado en varios problemas de corte de existencias.

Wang (1983) usó un método heurístico que, diferente al de Christofides y Whitlock (1977), no intenta encontrar todos los cortes posibles que podrían hacerse y así encontrar el patrón de corte óptimo. El método de Wang consistió en agregar iterativamente rectángulos en conjunto, de modo de formar patrones adecuados para el corte guillotina.

Beasley (1985a) presentó un algoritmo exacto que resolvió el problema bidimensional de corte de existencias, el cual no fue restringido sólo a cortes guillotina. Ésta era la primera vez que se presentaba un

algoritmo exacto para este problema. El método usa un modelo de programación entera y permite resolver en tiempos realistas problemas de tamaños moderados.

Beasley publicó otro artículo en 1985 (Beasley, 1985b) en el que indicó un error en Gilmore y Gomory (1966) y proporcionó un modelo correcto de programación dinámica. En el artículo, este modelo se aplica a grandes problemas y se presentan los resultados.

Dagli y Tatoglu (1987) usaron un método heurístico para resolver el problema de corte bidimensional. Se presentan varias restricciones que pueden originarse en este tipo de problema (por ejemplo, los defectos de la lámina) pero sólo tres son usadas en el artículo. Éstas son; los rectángulos no deben superponerse, los rectángulos deben quedar dentro del límite de la lámina y debe haber una tolerancia específica entre las formas para permitir el corte. La heurística está basada en la idea de asignar una prioridad a cada forma. Esta prioridad se usa para seleccionar qué forma se ubicará después. Se construyen once reglas de prioridad en el sistema junto con una prioridad definible por usuario. Dos de las prioridades incluyen el área máxima y mínima. Aún usando estas prioridades simples, el recorte de pérdidas fue reducido de 20% a 7,7% usando un ejemplo de la vida real.

En Dyckhoff y Finke (1992) se mencionan aproximadamente setecientos trabajos en el campo de corte de existencias y empaque. Los autores desarrollan un esquema de tipificación similar al presentado en

Dyckhoff (1990), además dan un resumen sistemático de la literatura existente sobre el tema.

Algunas ediciones especiales sobre el problema de corte de existencia y empaque pueden encontrarse en Dyckhoff y Wäscher (1990); Dyckhoff y Schster (1991); Lirow (1992); Martello (1994); Bischoff y Wäscher (1995).

Ghandforoush y Daniels (1992) aplican una nueva heurística basada en reglas que logra encontrar soluciones óptimas el 95% de las veces para el problema de corte guillotina 2D restricto.

MacLeod, Moll, Girkar y Hanifi (1993) presentan una heurística $O(n^3)$ también para el Problema de Corte Guillotina Restricto.

Los artículos de Oliveira y Ferreira (1990); Viswanathan y Bagchi (1988); Christofides y Hadjiconstantinou (1995); Parada, Gómes y de Diego (1995) resuelven el Problema de Corte Guillotina Restricto mediante un método exacto.

Parada, Muñoz, y Gómes (1995) usan un Algoritmo Genético (AG) para resolver el Problema de Corte Guillotina Bidimensional. El método usa una representación del tipo "*string*" de un árbol binario que fue manipulada por el AG. La última parte del artículo extiende la anotación de modo que considera la rotación de los rectángulos. Sus resultados demuestran que este método es superior al de Oliveira y Ferreira (1990), que era ya una mejora de Wang (1983).

Holthöfer y Tschöke (1995) presentan una paralelización del algoritmo de Branch and Bound que permite resolver instancias de mayor envergadura del Problema de Corte Guillotina Bidimensional, en forma óptima.

Shpitalni y Manevich (1996) presentan un nuevo modelo de Programación Lineal Entera (PLE) para resolver el problema de Corte de Existencias Bidimensional. La importancia de su modelo fue la habilidad de formular el problema como un problema de PLE. Esto no se había hecho antes. El enfoque de este artículo fue construir un modelo y no acelerar la solución, ya que admite que tomaría mucho tiempo de cómputo.

Dagli y Poshyanonda (1997) proponen métodos para resolver el problema de corte de existencias, los dos se basan en las Redes Neuronales Artificiales (RNA). El primer modelo usa precisamente una RNA, la cual es entrenada usando "*back-propagation*". El segundo método combina una RNA con un Algoritmo Genético (AG). Los resultados muestran que el recorte de pérdida para el método de RNA fue de 7,88%. El método combinado de RNA/AG produce recortes de pérdida entre 3% a 6%.

Nicklas, Atkins, Setia y Wang (1998) implementan una solución sobre un "*cluster*" de estaciones de trabajo interconectadas mediante una red "*Ethernet*" 10 Mbps y sobre un Multicomputador Intel Paragon, paralelizando el algoritmo de Wang (1983) para el Problema de Corte

Irregular. En ambos casos obtienen buenos resultados, siendo mejores los obtenidos por el Multicomputador.

Capítulo 3. Antecedentes Teóricos: Algoritmos Genéticos y Paralelismo

Introducción

Los Algoritmos Genéticos (AGs) son técnicas de búsqueda inspiradas en el proceso observado en la evolución natural de los seres vivos.

Los Algoritmos Genéticos han demostrado ser una herramienta poderosa en la solución de una gran variedad de problemas de optimización (Biethahn y Nissen, 1995).

Han habido múltiples esfuerzos en hacer a los AGs cada vez más rápidos, y una de las alternativas más prometedoras es usar implementaciones paralelas (Cantú-Paz, 2000). La naturaleza paralela de los algoritmos genéticos ha sido reconocida por mucho tiempo. Se ha tenido mucho éxito al usar Algoritmos Genéticos Paralelos para reducir el tiempo requerido de modo de alcanzar soluciones aceptables en problemas complejos (Cantú-Paz, 2000). AGs trabajan con una población de soluciones independientes, lo cual hace fácil distribuir la carga computacional entre varias unidades de proceso. Los Algoritmos Genéticos

Paralelos (AGPs) son complejos algoritmos no lineales controlados por muchos parámetros que afectan su eficiencia y la calidad de su búsqueda. La configuración correcta de esos parámetros es crucial para obtener buenas soluciones rápida y confiablemente.

En particular, el diseño de un AGP involucra el uso de una única o múltiples poblaciones. En ambos casos, el tamaño de la población debe ser determinado cuidadosamente, y para múltiples poblaciones se debe decidir cuántas usar. Las poblaciones pueden permanecer aisladas o ellas pueden comunicarse para intercambiar individuos o alguna otra información. El comunicarse involucra costos extras y decisiones adicionales sobre el patrón de comunicación, sobre el número de individuos a intercambiar, y sobre la frecuencia de comunicación (Cantú-Paz, 1999; Cantú-Paz, 2000).

Algoritmos Genéticos

Los Algoritmos Genéticos (AGs) fueron inicialmente propuestos por Fraser (1957); Fraser (1960); Bremermann (1958). Estos algoritmos simularon los sistemas genéticos. A pesar de la antigüedad de estos trabajos, todavía tienen relevancia hoy en día y recientemente han sido revisados por Fogel y Anderson (2000); Fogel y Fraser (2000). También se acredita a John Holland, sus estudiantes y colegas de la Universidad de Michigan en los años 1960 y 1970, con llevar a cabo muchos de los trabajos pioneros en AGs. El libro de Holland (1975), reimpreso en Holland (1992), se reconoce como uno de los trabajos fundamentales de los AGs. Los algoritmos desarrollados inicialmente por Holland eran simples, pero dieron soluciones satisfactorias a problemas que eran considerados como

“difíciles” en aquel tiempo. El campo de los AGs ha evolucionado desde entonces, principalmente debido a las innovaciones introducidas en la década de 1980. Los AGs han ido incorporando mecanismos cada vez más elaborados, bajo la motivación de la necesidad de resolución aproximada, de problemas prácticos de una amplia variedad.

Analogías con la Evolución

En la naturaleza, la evolución, en particular la de los seres vivos, tiene algunas características que motivaron a Holland a comenzar una línea de investigación en un área que eventualmente se transformó en lo que hoy se denomina Algoritmos Genéticos (AGs). La habilidad de una población de cromosomas para explorar el espacio de búsqueda “en paralelo”, y combinar lo mejor que ha sido encontrado en él mediante el mecanismo de cruzamiento (*crossover*), es algo intrínseco a la evolución natural y trata de ser explotada por los AGs.

Las características de la evolución que hay que tener en cuenta están descritas brevemente por Davis (1991) de la siguiente manera:

- La evolución es un proceso que opera en los cromosomas, en lugar de en los seres vivos que ellos codifican.

- Los procesos de selección natural provocan que aquellos cromosomas que codifican estructuras con éxito se reproduzcan más frecuentemente que aquellos que no lo

hacen.

- Las mutaciones pueden causar que los cromosomas de los hijos sean diferentes a los de los padres. Los procesos de recombinación pueden crear cromosomas bastante diferentes en los hijos, debido a la combinación de material genético de los cromosomas de ambos padres.

- La evolución biológica no tiene memoria.

La premisa de los AGs, tras la publicación del libro de Holland (1975) y de los numerosos investigadores que los utilizan como meta-heurística para optimización, es que se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de "evolución simulada", en particular como un algoritmo implementado en un computador. Debido al trabajo original de Holland, esto inicialmente ocurría en forma de algoritmos que manejan un "*string*" binario, a los que denominó "cromosomas", ya que ellos representaban un punto en el espacio de configuraciones del problema.

Como ocurre en la evolución biológica, la evolución simulada está diseñada para encontrar cada vez mejores cromosomas mediante una manipulación "ciega" de sus contenidos. El término "ciega" se refiere al hecho de que el proceso no tiene ninguna información acerca del problema que está tratando de resolver, exceptuando el valor de la función objetivo. En la concepción original de los AGs, la función objetivo es la única información por la que se evalúa el "valor" de un cromosoma. Por lo tanto, esta meta-heurística está basada en integrar e implementar eficientemente dos ideas fundamentales: la habilidad de representaciones simples (como por ejemplo, *bit strings*) para codificar configuraciones del problema de optimización, y el poder utilizar transformaciones simples para modificar y mejorar estas configuraciones. Las mejoras son guiadas mediante un mecanismo de control que permite a una población de cromosomas evolucionar como las poblaciones de seres vivientes lo hacen.

Un algoritmo genético puede ser visto como una estructura de control que organiza o dirige un conjunto de transformaciones y operaciones diseñadas para simular estos procesos de evolución.

En la evolución de los seres vivos, el problema al que cada individuo se enfrenta cotidianamente es la “supervivencia”.

Para ello, cuenta con las habilidades innatas provistas por su material genético. A nivel de los genes, el problema es buscar aquellas adaptaciones beneficiosas en un medio ambiente hostil y cambiante. Debido en parte a la selección natural, cada especie gana una cierta cantidad de “conocimiento”, el cual es codificado e incorporado en la nueva conformación de sus cromosomas. Esta conformación se ve alterada por las operaciones de reproducción. Algunas de ellas son las mutaciones aleatorias, inversión de partes del cromosoma y el cruzamiento (intercambio de material genético proveniente de dos cromosomas padres). En los AGs, las mutaciones aleatorias proveen cierta variación y ocasionalmente, introducen alteraciones beneficiosas en los cromosomas.

Evidentemente, es la existencia del cruzamiento lo que gobierna la eficiencia de los AGs y los destaca nítidamente de otro tipo de meta-heurísticas. Con él, las características de los padres pueden ser combinadas inmediatamente cuando se reproducen. Por lo tanto, la probabilidad de esta combinación se acrecienta cuando los padres tienen un nivel elevado de “*fitness*” (aptitud) debido a que se reproducen con mayor frecuencia.

Fundamentos de los Algoritmos Genéticos

Los Algoritmos Genéticos utilizan una analogía directa del fenómeno de evolución en la naturaleza. Trabajan con una población de individuos, cada uno representando una posible solución a un problema dado. A cada

individuo se le asigna una puntuación de adaptación, dependiendo de qué tan buena sea la respuesta al problema. A los más adaptados se les da la oportunidad de reproducirse mediante cruzamientos con otros individuos de la población, produciendo descendientes con características de ambos padres. Los miembros menos adaptados poseen pocas probabilidades de que sean seleccionados para la reproducción y desaparecen.

Una nueva población de posibles soluciones es generada mediante la selección de los mejores individuos de la generación actual, emparejándolos entre ellos para producir un nuevo conjunto de individuos. Esta nueva generación, contiene una proporción más alta de las características poseídas por los mejores miembros de la generación anterior. De esta forma, a lo largo de varias generaciones, las características buenas son difundidas en la población mezclándose con otras. Favoreciendo el emparejamiento de los individuos mejor adaptados, es posible recorrer las áreas más prometedoras del espacio de búsqueda. Si el Algoritmo Genético ha sido diseñado correctamente, la población convergerá a una solución óptima o casi óptima al problema.

Los dos procesos que más contribuyen a la evolución son el cruzamiento (*crossover*) y la adaptación basada en la selección/reproducción. La mutación es usualmente considerada como un operador secundario, ella no debe ser utilizada demasiado, ya que el Algoritmo Genético puede convertirse en una búsqueda al azar, pero su utilización asegura que ningún punto en el espacio de búsqueda tiene

probabilidad nula de ser examinado.

En la práctica, se puede implementar este modelo utilizando matrices de bits o caracteres para representar los cromosomas. Operaciones sencillas de bits permiten efectuar el cruzamiento, la mutación y otras operaciones. La mayor parte del trabajo con los Algoritmos Genéticos ha sido enfocado en cadenas de caracteres de longitud fija. Se hace énfasis en este aspecto y en la necesidad de codificar la solución como una cadena de caracteres.

Generalmente, los AGs son implementados siguiendo el siguiente ciclo:

- Generar aleatoriamente la población inicial.
- Evaluar la adaptación de todos los individuos en la población.

- Crear una nueva población efectuando operaciones como cruzamiento, reproducción proporcional a la adaptación y mutaciones en los individuos cuya adaptación acaba de ser medida.
- Eliminar la antigua población.
- Iterar utilizando la nueva población, hasta que la población converja.

Cada iteración de este ciclo es conocida como generación. La primera generación de este proceso es una población de individuos generados al azar. Desde ese punto, los operadores genéticos, en unión con la medida de adaptación, actúan para mejorar la población.

Los Algoritmos Genéticos no son la única técnica basada en una analogía de la naturaleza. Por ejemplo, las Redes Neuronales están basadas en el comportamiento de las neuronas en el cerebro. Pueden ser utilizadas en una gran variedad de tareas de clasificación, como reconocimiento de patrones o proceso de imágenes.

El poder de los Algoritmos Genéticos proviene del hecho de que la técnica es robusta, y puede manejar exitosamente un amplio rango de problemas, incluso algunos que son difíciles de resolver por otros métodos. Los Algoritmos Genéticos no garantizan que encontrarán la solución óptima al problema, pero son generalmente buenos encontrando, en corto tiempo, soluciones aceptables a problemas.

Diferencias entre los Algoritmos Genéticos y los Métodos Tradicionales

Los Algoritmos Genéticos tienen cuatro diferencias principales con los métodos más utilizados o conocidos de optimización y búsqueda:

- Trabajan con una codificación del conjunto de parámetros, no con éstos directamente.
- Buscan simultáneamente la solución en una población de puntos, no en uno sólo.
- Utilizan la función objetivo (rendimiento), no derivadas u otro conocimiento auxiliar.
- Utilizan reglas de transición probabilísticas y no determinísticas.

Principios Básicos de los Algoritmos Genéticos

Antes que un AG pueda ejecutarse, debe diseñarse una adecuada codificación (o representación) del problema. También, se requiere una función objetivo, la cual asigna un valor a cada solución. Durante la ejecución, los padres deben ser seleccionados por reproducción y volver a ser combinados para generar descendencia. Estos aspectos se describen a continuación.

Codificación

Las partes que relacionan un Algoritmo Genético con un problema

dado son la codificación y la función de evaluación.

Si un problema puede ser representado por un conjunto de parámetros (conocidos como genes), éstos pueden ser unidos para formar una cadena de valores (cromosoma), a este proceso se le denomina *codificación*. En genética, este conjunto representado por un cromosoma en particular es conocido como *genotipo*, el cual contiene la información necesaria para construir un organismo, conocido como *fenotipo*. La adaptación de cada individuo depende de su fenotipo, el cual se puede inferir de su genotipo, es decir, puede calcularse desde el cromosoma utilizando la función de evaluación.

Existen varios aspectos relacionados con la codificación de un problema a ser tomados en cuenta en el momento de su realización:

- Se debe utilizar el alfabeto más pequeño posible para representar los parámetros, normalmente se utilizan dígitos binarios.
- Las variables que representan los parámetros del problema deben ser discretizadas para poder ser representadas con cadenas de bits. Hay que utilizar suficiente resolución para asegurar que la salida tiene un nivel de precisión adecuado, se asume que la discretización es representativa de la función objetivo.

La mayor parte de los problemas tratados con Algoritmos Genéticos

son no lineales y muchas veces existen relaciones "ocultas" entre las variables que conforman la solución.

El tratamiento de los genotipos inválidos debe ser tomado en cuenta para el diseño de la codificación. Si se necesitan 1.200 valores para representar una variable, esto requiere al menos 11 bits, pero éstos codifican un total de 2.048 posibilidades, "sobrando" 848 patrones de bits no necesarios. A estos patrones se les puede dar un valor cero de adaptación o pueden ser substituidos por un valor real.

Función de evaluación

Dado un cromosoma, la función de evaluación consiste en asignarle un valor numérico de "adaptación", el cual se supone que es proporcional a la "utilidad" o "habilidad" del individuo representado. La función puede estar basada en el rendimiento y representar sólo una evaluación parcial del problema. Adicionalmente, debe ser rápida ya que hay que aplicarla para cada individuo de cada población en las sucesivas generaciones, por lo cual, gran parte del tiempo de procesamiento de un algoritmo genético se emplea en la función de evaluación.

Convergencia prematura

Un problema de los Algoritmos Genéticos dado por una mala formulación del modelo, es aquel en el cual los genes de unos pocos individuos relativamente bien adaptados, pero no óptimos, pueden

rápidamente dominar la población, causando que converja a un óptimo local. Una vez que esto ocurre, la habilidad del modelo para buscar mejores soluciones es eliminada completamente, quedando sólo la mutación como vía para buscar nuevas alternativas, y el algoritmo se convierte en una búsqueda lenta al azar. Para evitar este problema, es necesario controlar el número de oportunidades reproductivas de cada individuo, tal que no obtenga una probabilidad muy alta o muy baja. El efecto es comprimir el rango de adaptación y prevenir que un individuo "super-adaptado" tome control rápidamente.

Finalización lenta

Este es un problema contrario al anterior, luego de muchas generaciones, la población habrá convergido, pero no habrá localizado el óptimo. La adaptación promedio será alta y habrá poca diferencia entre el mejor y el individuo promedio, por consiguiente será muy baja la tendencia de la función de adaptación en llevar el algoritmo hacia el óptimo. Las mismas técnicas aplicadas en la convergencia prematura son utilizadas en este caso.

Reproducción

Durante la fase reproductiva de un Algoritmo Genético, se seleccionan individuos de la población siendo recombinados para formar descendientes que formarán la siguiente generación. Los padres son

seleccionados al azar, usando un método que favorece a los individuos mejor adaptados y asigna una probabilidad baja de ser seleccionados a los menos adaptados. Luego de ser escogidos los padres, sus cromosomas se mezclan y cambian, usando cruzamiento y mutación. Las formas básicas de estos operadores son:

Cruzamiento: toma dos individuos y corta sus cromosomas en una posición seleccionada al azar, para producir dos segmentos anteriores y dos posteriores, los posteriores se intercambian para obtener dos cromosomas nuevos (ver Figura N°3.1). Esto es conocido como "cruzamiento de un punto".

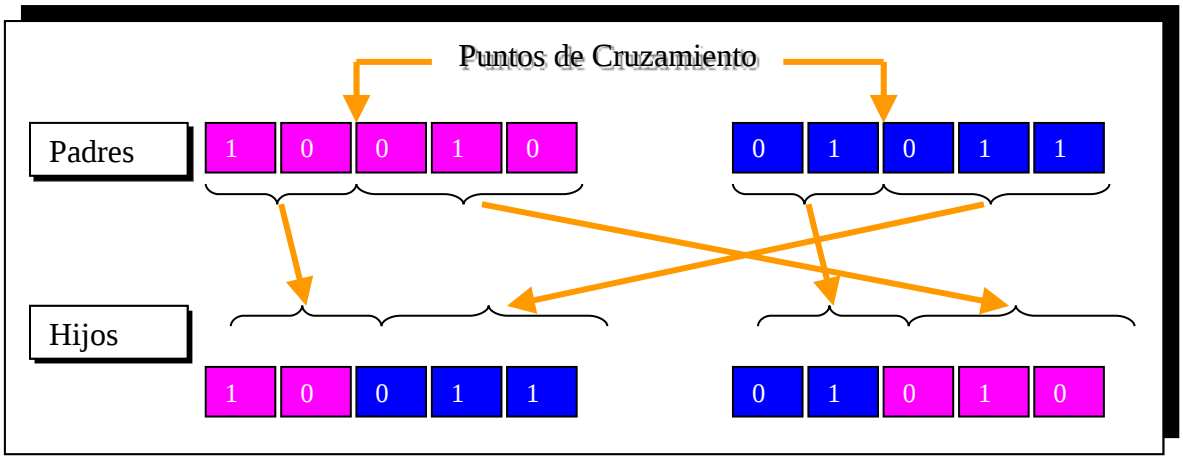


Figura N°3.1: Cruzamiento de un punto.

Mutación: es aplicada a cada descendiente individualmente luego de cada cruzamiento. Altera un gen al azar, con una probabilidad muy pe ón ser

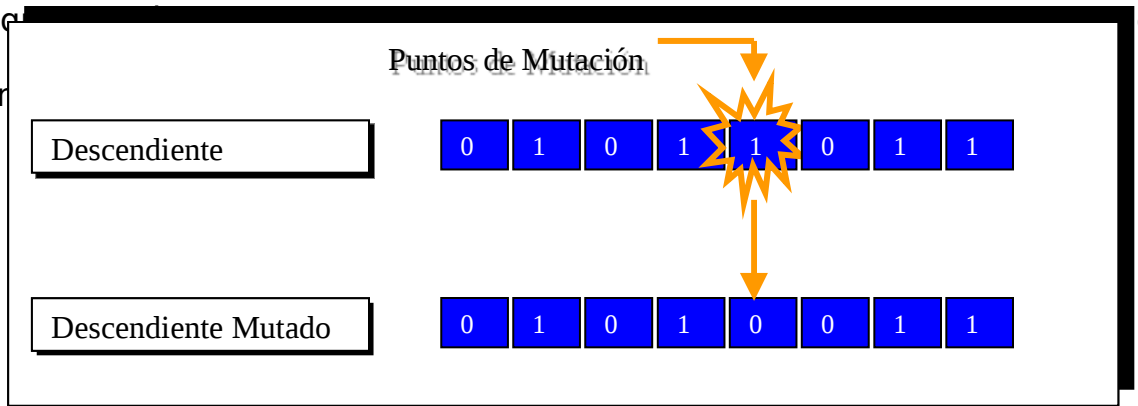


Figura N°3.2: Mutación de un punto.

Convergencia

Si el Algoritmo Genético ha sido correctamente implementado, la población evolucionará a lo largo de sucesivas generaciones de forma que la adaptación del mejor y el promedio general se incrementarán hacia el óptimo global. La convergencia es la progresión hacia la uniformidad. Un gen ha convergido cuando el 95% de la población tiene el mismo valor. La población converge cuando todos los genes de cada individuo lo hacen. Por ejemplo, la Figura N°3.3 muestra la convergencia representada por la varianza de una población a lo largo de sucesivas generaciones.

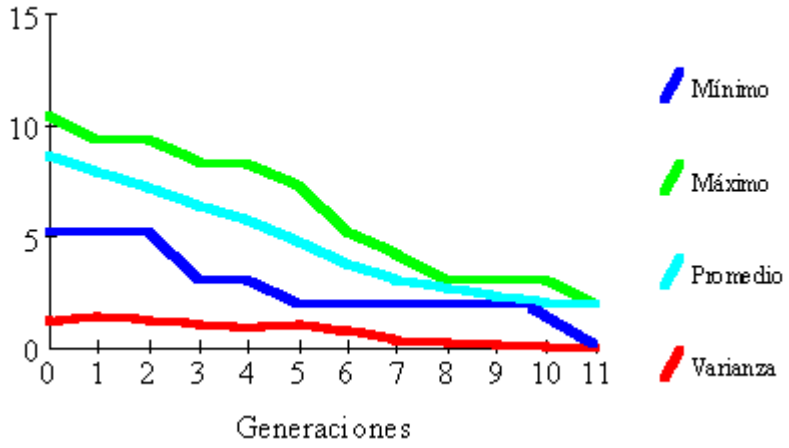


Figura N°3.3: Ejemplo de convergencia.

Mayor información de los Algoritmos Genéticos puede encontrarse en Beasley, Bull y Martin (1993); Coley (1999); Davis (1987); Davis (1991); Forrest (1993); Goldberg (1989a); Man, Tang y Kwong (1999); Michalewicz (1996); Michalewicz y Fogel (2000); Mitchell (1996); Reeves (1995).

Modelos Paralelos de los Algoritmos Genéticos

Los Algoritmos Genéticos Secuenciales han demostrado ser muy exitosos en muchas aplicaciones (Sales, 1997; Muñoz, 1995) y en diferentes dominios. Sin embargo, existen algunos problemas en su utilización (Nowostawski y Poli, 1999), los cuales pueden ser encaminados hacia una forma de Algoritmos Genéticos Paralelos.

Estos problemas son:

- La evaluación del "*fitness*" es usualmente muy consumidora de tiempo. En la literatura se han reportado tiempos de computación de más de un año de CPU al utilizar una única máquina corriendo en dominios complejos (Luke, 1998). Esto fundamenta el hecho de que la única manera práctica de proveer este poder de CPU es mediante el uso de procesamiento paralelo.
- Para algunos tipos de problemas, la población necesita ser muy grande y la memoria requerida para almacenar cada individuo puede llegar a ser considerable (ver ejemplos en Koza, 1992). En algunos casos, esta condición hace imposible correr una aplicación eficientemente usando una única máquina, entonces se hace necesario paralelizar los Algoritmos Genéticos.
- Algoritmos Genéticos Secuenciales pueden quedar atrapados en regiones sub-óptimas en el espacio de búsqueda, llegando a

ser incapaces de encontrar soluciones de mejor calidad. Los Algoritmos Genéticos Paralelos pueden buscar en diferentes sub-espacios en forma paralela, dentro del espacio de búsqueda; de esa manera se hace menos probable el hecho de quedar atrapado en sub-espacios de menor calidad.

Debido a las dos primeras razones es que se estudian los Algoritmos Genéticos Paralelos, y son usados para aplicaciones sobre máquinas masivamente paralelas (Tanese, 1987) y también sobre sistemas distribuidos (Tanese, 1989b). Sin embargo, la ventaja más importante de los Algoritmos Genéticos Paralelos es que, en muchos casos, ellos proveen mejor "*performance*" que algoritmos basados en una única población, aún cuando el paralelismo es simulado sobre máquinas convencionales (Nowostawski y Poli, 1999). La razón es que múltiples poblaciones permiten evolucionar en diferentes direcciones, esto es, hacia diferentes óptimos (Goldberg, 1989b). Por ello, los Algoritmos Genéticos Paralelos no sólo son una extensión del modelo tradicional de Algoritmos Genéticos Secuenciales, sino que ellos representan una nueva clase de algoritmos que buscan, en forma diferente, el espacio de soluciones.

Los Algoritmos Genéticos Paralelos a menudo permiten análisis teóricos, los cuales no son más complicados que los Algoritmos Genéticos Secuenciales (Cantú-Paz, 1997; Cantú-Paz, 1998b). Uno de los grandes problemas que se debe resolver cuando se utilizan Algoritmos Genéticos Paralelos es cómo determinar el modelo paralelo a usar.

Uno de los modelos paralelos de los Algoritmos Genéticos que ha mostrado buen desempeño es el de "Concurrencia Síncrona de Redes", propuesto por Solar (Solar, 1992) el cual fue probado para el entrenamiento de redes neuronales. Kri (1996) utiliza el mismo modelo paralelo para resolver el problema de planificación de actividades,

obteniendo en el 85,7% de los casos, soluciones de mejor calidad que el AG Secuencial, en tanto que en el 14,3% restante el modelo paralelo encuentra soluciones iguales al AG Secuencial.

Han existido algunos intentos para desarrollar una taxonomía unificada de los Algoritmos Genéticos Paralelos. Algunas son taxonomías generales para arquitecturas de memoria-distribuida (Bianchini y Brown, 1993), otras son taxonomías para Algoritmos Genéticos de Grano-Grueso (Lin, Punch y Goodman, 1994), o de Grano-Fino (Maruyama, Hirose y Konagaya, 1993). Otros esfuerzos relacionados con este tema se pueden ver en Grefenstette, Leuze y Pettey (1987); Belding (1995); Goldberg et al (1997). Otras taxonomías recientes sobre Algoritmos Genéticos Paralelos han sido expuestas por Cantú-Paz (1995, 1998a, 2000); Nowostawski y Poli (1999). En Crainic y Toulouse (1997) puede encontrarse una amplia revisión de los trabajos realizados en las últimas dos décadas sobre los Algoritmos Genéticos Paralelos.

Puesto que la selección natural es un proceso paralelo (Cantú-Paz, 1995), los Algoritmos Genéticos, como un modelo de selección natural, son muy adecuados para la paralelización. Cantú-Paz (2000), basado en Lin, Punch, y Goodman (1994); Adamidis (1994); Tomassini (1999); Alba y Troya (1999), proporciona una clasificación de diferentes estrategias para la paralelización de los Algoritmos Genéticos. Esta clasificación es la que se detalla en las secciones 3.3.1 a 3.3.4.

Algoritmos Genéticos Maestro-Esclavo de una Única Población

En este modelo los operadores genéticos son aplicados en paralelo a una única población. Hay diferentes formas de implementar este modelo. Una forma es paralelizar el proceso de evaluación del "*fitness*". En el caso de máquinas Multiprocesador de Memoria Compartida, cada procesador lee el individuo asignado, lo evalúa, y almacena los resultados en la memoria compartida. En este método, es necesario establecer una sincronización entre cada generación.

En un ambiente de memoria distribuida, para simplificar la implementación del algoritmo, un procesador Maestro es usado para almacenar la población y para aplicar los operadores genéticos. El Maestro es responsable de enviar un subconjunto de individuos a los procesadores Esclavos para la evaluación del "*fitness*" y recolección de resultados. El "*performance*" de esta estrategia depende fuertemente del "*overhead*" de comunicación y de la arquitectura paralela usada (Zhaksilikov y Harris, 1996). Se puede lograr "*speedup*" casi lineales cuando el tiempo de cómputo es mucho más grande que el tiempo de comunicación.

Otra forma de implementar este modelo es aplicando los operadores genéticos en paralelo, debido a que los operadores genéticos son muy simples y el tiempo gastado en la comunicación puede ser mucho mayor que el tiempo en hacer los cómputos. Es posible que este método tome más tiempo que su contraparte secuencial.

El AG Maestro Esclavo puede ser síncrono o asíncrono, dependiendo

de si el Maestro espera recibir los valores de "*fitness*" de la población completa antes de continuar con la siguiente generación. El AG Maestro Esclavo asíncrono no espera respuesta de los procesadores más lentos, y tiene el potencial de ser más eficiente que el AG Maestro Esclavo síncrono (Zeigler y Kim, 1993; Stanley y Mudge, 1995), pero su análisis es más difícil (Cantú-Paz, 2000).

El tiempo de ejecución de los AGs Maestro Esclavo tiene dos componentes básicos, el tiempo usado en los cómputos y el tiempo usado en comunicar información entre procesadores. El tiempo de cómputos es ampliamente determinado por el tamaño de la población, debido a eso puede ser atractivo reducir la población para hacer el AG más rápido. Sin embargo, el tamaño de la población no puede ser reducido arbitrariamente sin sufrir un deterioro importante en la calidad de la solución (Cantú-Paz, 2000).

Algoritmos Genéticos con Múltiples Sub-Poblaciones

En este caso la población es dividida en varias sub-poblaciones. Cada sub-población es asignada a un procesador diferente y se mantiene relativamente aislada.

Se utiliza el operador *migración* como mecanismo para intercambiar información entre las sub-poblaciones. Hay dos formas de implementar la migración: en un modelo de isla y en un modelo escalón (*stepping stone*). El modelo de isla considera una población dividida en sub-

poblaciones separadas geográficamente, con migración entre dos sub-poblaciones cualesquiera. El segundo modelo restringe la migración sólo entre sub-poblaciones vecinas.

Los AGs con múltiples sub-poblaciones también son conocidos como AGs distribuidos, porque usualmente son implementados sobre computadores de memoria distribuida (MIMD). Debido a que la tasa de cómputo versus comunicación es usualmente alta, ellos ocasionalmente son denominados como "AGs de grano grueso" (Zhaksilikov y Harris, 1996; Nowostawski y Poli, 1999).

El diseño de los Algoritmos Genéticos Paralelos con múltiples sub-poblaciones conlleva escoger algunos parámetros, los cuales se presentan a continuación:

- El tamaño y el número de sub-poblaciones a considerar.
- La topología que interconecta las sub-poblaciones.
- La tasa de migración, que controla el número de individuos a migrar.
- La frecuencia de migración, que determina qué tan a menudo la migración ocurrirá.
- La política de migración, que determina qué individuos migrar y cuales serán reemplazados en la sub-población que los reciba.

Usualmente, esos parámetros son sintonizados para obtener un mejor "*performance*". Por ejemplo, Hines, Torpe, Winiecki y Harris (1995)

reportaron que si la tasa de migración es demasiado alta, la población converge demasiado rápido y raramente se producen resultados satisfactorios. Pero la ausencia de migración también conduce a escasos resultados. Cantú-Paz (2000) indica que el "*speedup*" no es muy significativo cuando las sub-poblaciones están aisladas, pero existe un sustancial mejoramiento en el "*performance*" cuando las sub-poblaciones se comunican.

La topología, usualmente, está relacionada con la conectividad de una red. Diferentes publicaciones reportan buenos resultados usando topologías tanto baja como altamente interconectadas (Bianchini y Brown, 1993; Cantú-Paz y Majia-Olvera, 1994). En caso de alta conectividad, buenas soluciones pueden dispersarse por la red. En el caso de topologías escasamente conectadas, resultan soluciones más aisladas que pueden ser recombinadas más tarde para formar resultados potencialmente mejores. El número de migrantes podría ser suficiente para proveer el intercambio de información, pero tasas elevadas de migración pueden ser, probablemente, un desaprovechamiento de los recursos de comunicación.

Bossert (1967) fue probablemente el primero que propuso un algoritmo evolutivo con múltiples poblaciones para mejorar la calidad de las soluciones en un problema de optimización.

Grefenstette, Leuze y Pettey (1987) describen un AGP con múltiples poblaciones. En este algoritmo, los mejores individuos de cada sub-población son difundidos en cada generación a todas las demás sub-

poblaciones. Ellos reconocen la complejidad de los Algoritmos Genéticos Paralelos de grano grueso y plantean muchas interrogantes acerca de la frecuencia de migración, el destino de los migrantes, y el efecto de la migración para evitar convergencias prematuras hacia soluciones sub-óptimas.

Grosso (1985) es el primero en observar que la tasa de mejoramiento del "*fitness*" es mayor en una población dividida en sub-poblaciones que en una única población. Pero si las sub-poblaciones permanecen aisladas durante todo el experimento, la calidad de la solución final es menor que la alcanzada por una única población de tamaño equivalente. Grosso, también, examina un esquema de retardo de migración, en el cual las comunicaciones ocurren sólo después que las sub-poblaciones están cerca de la convergencia. Entonces, las sub-poblaciones intercambian individuos con una elevada tasa de migración. El esquema de retardo reduce los costos de comunicación y obtiene la misma calidad que la que obtiene con migraciones frecuentes.

Tanese (1987, 1989a, 1989b); realiza un estudio sistemático de la migración y sus efectos sobre la eficiencia y calidad de los Algoritmos Genéticos Paralelos. Su método consiste en dividir un número fijo de individuos en sub-poblaciones del mismo tamaño y variar tanto la tasa como el intervalo de migración. Sus experimentos muestran que para encontrar soluciones con la misma calidad que la encontrada por un Algoritmo Genético Secuencial, se requieren tasas medianas de migración.

Tasas de migración demasiado pequeñas o muy elevadas resultan en soluciones inferiores. Como Grosso (1985), Tanese examina las sub-poblaciones completamente aisladas y encuentra que las soluciones son generalmente inferiores a aquellas encontradas por el AG Secuencial o por el AGP con migración. En términos de eficiencia del algoritmo paralelo, se puede obtener "*speedups*" cercanos a valores lineales cuando la migración es poco frecuente y la tasa de migración es baja. Además, Tanese estudia cómo los parámetros de la migración afectan la pérdida de diversidad genética. Para prevenir la pérdida prematura de buenas soluciones, ella propone que se configuren distintos parámetros AG en cada población, causando que algunas poblaciones puedan converger más lentamente que otras.

Por ejemplo, las poblaciones pueden usar distintos tipos o probabilidades de cruzamiento y diferentes métodos de selección. La idea es que la diversidad se conserve por mucho tiempo en las poblaciones que converjan lentamente, y la migración podría reintroducir diversidad en las poblaciones que converjan rápidamente.

Otros (Lin, Punch y Goodman, 1994; Adamidis y Petridis, 1996; Herrera y Lozano, 2000) han adoptado exitosamente la idea de Tanese de usar diferentes parámetros en cada población para tratar de balancear la exploración y explotación. De este enfoque heterogéneo se han obtenido buenos resultados (Goldberg, Deb y Thierens, 1993). Lin, Punch y Goodman (1994) implementaron con éxito un algoritmo heterogéneo que usa

diferentes representaciones en cada sub-población.

Una pregunta importante es determinar bajo qué condiciones un AGP puede encontrar una mejor solución que un AG Secuencial. Starkweather, Whitley y Mathias (1991) observaron que mediante el uso de sub-poblaciones relativamente aisladas se consigue evolucionar en diferentes direcciones.

A continuación, se presenta una categorización de los Algoritmos Genéticos con Múltiples Sub-Poblaciones en tres dimensiones: los métodos de migración, sus esquemas de conexión y la homogeneidad de sus nodos (elementos de procesos).

Métodos de Migración de los AGs con Múltiples Sub-Poblaciones

Los Métodos de Migración determinan la frecuencia y bajo qué restricciones, los individuos son intercambiados entre poblaciones.

AGs con Múltiples Sub-Poblaciones Aisladas

No existe migración entre sub-poblaciones. Éste es el modelo más simple de los Algoritmos Genéticos con Múltiples Sub-poblaciones.

AGs con Múltiples Sub-Poblaciones Síncronas

La Migración entre dos sub-poblaciones es sincronizada. Debido a la migración síncrona, las poblaciones evolucionan, antes de que los intercambios ocurran, en aproximadamente la misma proporción (tasa).

Esta medida (número de generaciones, porcentaje de convergencia, etc.) determina la sincronización de las sub-poblaciones. Un equipamiento paralelo dedicado puede directamente soportar tal sincronización, pero dicha sincronización en un ambiente distribuido de estaciones de trabajo puede causar cargas irregulares de trabajo. Diferentes velocidades de las máquinas y diferentes cargas de trabajo pueden causar que algunos nodos tengan que “esperar”, retardando la velocidad de evolución a la del nodo más lento.

AGs con Múltiples Sub-Poblaciones Asíncronas

El modelo asíncrono permite migraciones basadas en un único evento que no relaciona el estado de la evolución de las sub-poblaciones de todo el sistema. Tal comportamiento asíncrono es el tipo de migración que típicamente puede encontrarse en la naturaleza, ya que diferentes entornos son responsables de las diferencias en la velocidad de evolución.

En este tipo de configuración, cada proceso AG compite por recursos computacionales con cualquier usuario/proceso sobre la misma máquina. Las diferentes cargas y las diversas arquitecturas de las máquinas causan diferentes velocidades de evolución sobre cada nodo, haciendo la migración asíncrona más apropiada. Esto es especialmente cierto cuando el número de nodos que procesan es del orden de los centenares y las arquitecturas de máquinas son de diferentes tipos.

Cuando un individuo con “*fitness*” relativamente alto, proveniente de

un nodo con rápida evolución, es insertado en una población de bajo "*fitness*", sobre un nodo de lenta evolución, surgen las siguientes interrogantes:

- ¿Es esta sub-población de bajo "*fitness*" dominada rápidamente por el individuo de alto "*fitness*"?.
- ¿Esta inserción ayuda a la sub-población de bajo "*fitness*"?.

Esquemas de Conexión de los Algoritmos Genéticos con Múltiples Sub-Poblaciones

La conectividad de los nodos de procesamiento, en términos de grado de conectividad y de topología de conexión, afecta el "*performance*" de los AGs con múltiples sub-poblaciones (Lin, Punch, y Goodman, 1994). Existen dos esquemas de conexión: estáticos y dinámicos.

Esquemas de Conexión Estáticos de los Algoritmos Genéticos con Múltiples Sub-Poblaciones

Las conexiones entre los nodos son establecidas al comienzo y no son modificadas, manteniendo la topología estática durante la ejecución. Las topologías pueden ser de varios tipos: líneas, anillos, mallas, n-cubos, etc., pero la topología determina qué vecinos pueden intercambiar información y qué topología es estática.

Esquemas de Conexión Dinámicos de los Algoritmos Genéticos con Múltiples Sub-Poblaciones

En este caso, la topología de conexión es alterada durante el tiempo

de ejecución. Hay dos razones básicas para permitir modificación en la topología durante el tiempo de ejecución. Primero, tales modificaciones hacen que el paralelismo distribuido en estaciones de trabajo sea práctico en ambientes del mundo real. Si un “proceso AG” de algún nodo es detenido, entonces ocurre una reconfiguración de la red de modo de continuar con el procesamiento. Los AGs de grano grueso pueden resistir un número de aquellos eventos antes de que se produzca la pérdida del proceso evolutivo. Segundo, la reconfiguración podría ocurrir debido a cambios en la evolución de las sub-poblaciones. Uno de los inconvenientes de los AG de grano grueso es que la inserción de un nuevo individuo desde otra sub-población puede no ser efectiva. El nuevo individuo puede ser demasiado incompatible con la sub-población, y por tanto ser ignorado o controlado por la sub-población. Para evitar esto, las sub-poblaciones pueden partir procesando sin ningún vecino, y cuando ocurra la migración, los nodos vecinos pueden ser determinados por similitud (o disimilitud) con otras sub-poblaciones. Por ejemplo, los mejores individuos de cada población pueden ser comparados, y aquellos con distancia “*Hamming*” más semejante pueden ser establecidos como vecinos. Esto establece una topología en base a una distancia que varía en el tiempo y mantiene el intercambio entre sub-poblaciones similares (o distintas).

Homogeneidad de los Nodos en los Algoritmos Genéticos con Múltiples Sub-Poblaciones

La *Homogeneidad* de los nodos es una medida que indica la similaridad de los procesos AGs en diferentes nodos de procesamiento.

En el caso que los nodos sean homogéneos, el AG sobre cada nodo usa los mismos parámetros (tamaño de población, taza de cruzamiento, tasa de mutación, intervalo de migración, etc.), los mismos operadores genéticos, funciones objetivos, y métodos de codificación. La mayoría de las investigaciones en el campo de los Algoritmos Genéticos Paralelos (AGPs) ha sido basada en AGPs homogéneos. Una ventaja es que son relativamente fáciles de implementar.

Si los nodos son heterogéneos, el AG sobre cada nodo usa parámetros, operadores genéticos, funciones objetivos y métodos de codificación diferentes. En general, es difícil encontrar el mejor parámetro inicial para un AG. Esto es especialmente cierto en el caso de problemas de optimización con múltiples funciones objetivos, donde varias configuraciones de parámetros pueden causar que un AG focalice la búsqueda en diferentes porciones del espacio de búsqueda.

Muchos problemas pueden ser codificados en un AG usando diferentes métodos, cada uno con ventajas y desventajas particulares. Tales variaciones en los parámetros de configuración y en los métodos de codificación presentan problemas para el intercambio entre poblaciones.

Algoritmos Genéticos de Grano Fino

En el caso de una paralelización de grano fino, la población es dividida

en un gran número de pequeñas sub-poblaciones. El caso ideal es tener un individuo por cada elemento de proceso disponible. La migración se puede establecer mediante la superposición de las sub-poblaciones. El mecanismo de selección y el operador de cruzamiento son aplicados considerando los cromosomas vecinos. Por ejemplo, cada cromosoma selecciona el mejor vecino para la recombinación, luego el cromosoma es reemplazado por el individuo resultante. La literatura muestra que el "*performance*" de los AGPs de grano fino depende del tamaño de la sub-población y de la topología de interconexión. Spiessen y Manderick (1990, 1991) muestran que el "*performance*" usualmente se degrada al incrementarse el tamaño de la sub-población. Anderson y Ferris (1990) y Schwehm (1992) muestran que existen topologías que proveen buenos resultados.

Estos tipos de AGs son, también, conocidos como Algoritmos Genéticos Celulares, los cuales son usualmente implementados en computadores masivamente paralelos. Algunos ejemplos pueden encontrarse en Mühlenbein (1989); Gorg-Schleuter, (1990); Spiessen y Manderick (1991); Collins (1992) y Baluja (1992). Ellos reportan mejores soluciones utilizando AGs celulares sobre problemas difíciles que al utilizar un AG estándar.

Algoritmos Genéticos Híbridos

Las estrategias híbridas combinan características de diferentes métodos de paralelización. Por ejemplo, la población puede ser dividida

como en el caso de la paralelización de grano grueso, y el operador de migración puede ser usado para intercambiar individuos, pero la evaluación de los individuos se efectúa en paralelo. Este enfoque no introduce nuevos problemas analíticos y puede ser exitosamente implementado cuando se trabaja con funciones objetivos de aplicaciones complejas que necesitan una considerable cantidad de tiempo de cómputo.

Conceptos de Computación Paralela

A continuación se describen algunos conceptos de computación paralela necesarios para una mejor compresión de las herramientas utilizadas para resolver el PCPGBR mediante Algoritmos Genéticos Paralelos.

Tipos de Computadores Paralelos

Un modelo de computador convencional, tal como el que se muestra en la Figura N°3.4, consiste de un procesador ejecutando un programa almacenado en memoria.

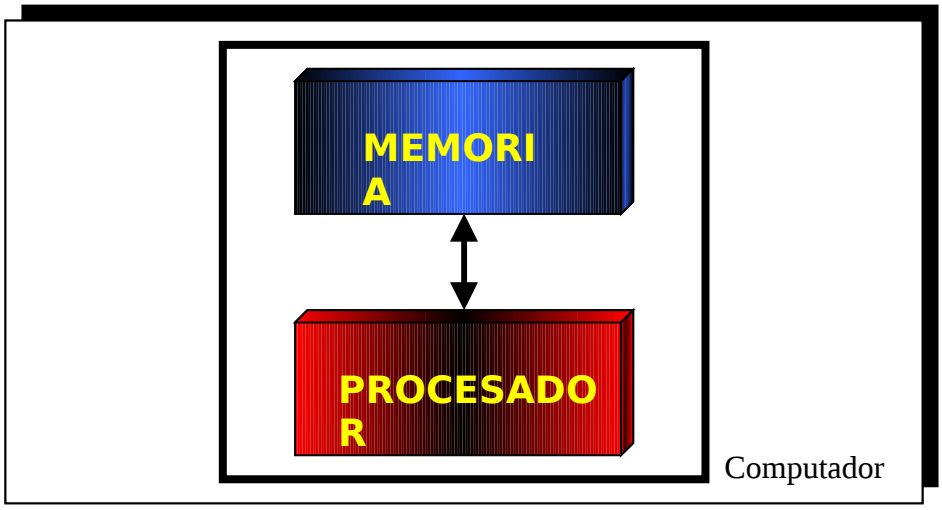


Figura N°3.4: Modelo de Computador convencional con procesador y memoria únicos.

El modelo anterior puede extenderse tanto a un Multiprocesador como

a un Multicomputador. La Figura N°3.5 muestra el Modelo Multiprocesador que representa a un Computador Paralelo con múltiples procesadores, conectados a múltiples módulos de memoria. Cada procesador puede acceder a cada módulo de memoria, por lo que se le conoce también como Multiprocesador de Memoria Compartida (Wilkinson y Allen, 1999).

Un programa se almacena en la memoria compartida para que cada procesador pueda ejecutarlo. Los datos también son almacenados en la memoria compartida, de modo que cada programa pueda acceder a todos los datos si se requiere. En esta configuración es substancialmente difícil implementar el hardware necesario para lograr rápidos accesos a toda la memoria compartida por parte de todos los procesadores.

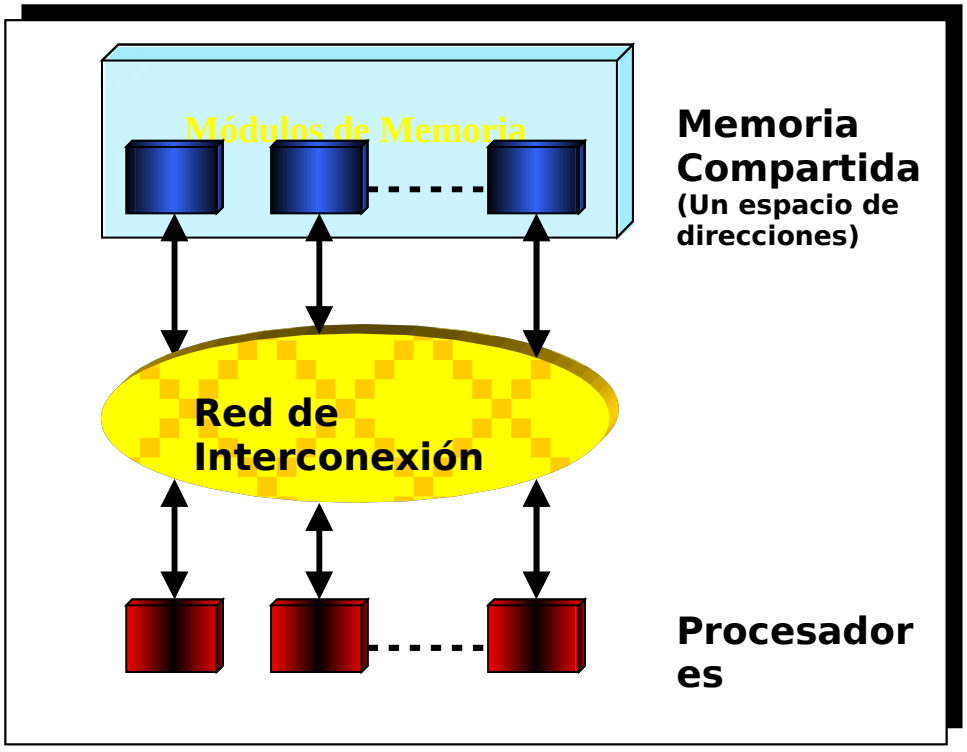


Figura N°3.5: Modelo Multiprocesador de Memoria Compartida.

La Figura N°3.6 muestra el Modelo Multicomputador, en el cual cada computador consiste en un procesador y memoria local que no es accesible desde otros procesadores. En esta configuración, la memoria está distribuida a través de los computadores y cada computador tiene su propio espacio de direcciones. Mediante la red de interconexión, los procesadores pueden enviar mensajes a otros procesadores. Estos mensajes pueden incluir datos que otros procesadores pueden requerir para realizar sus propios cálculos.

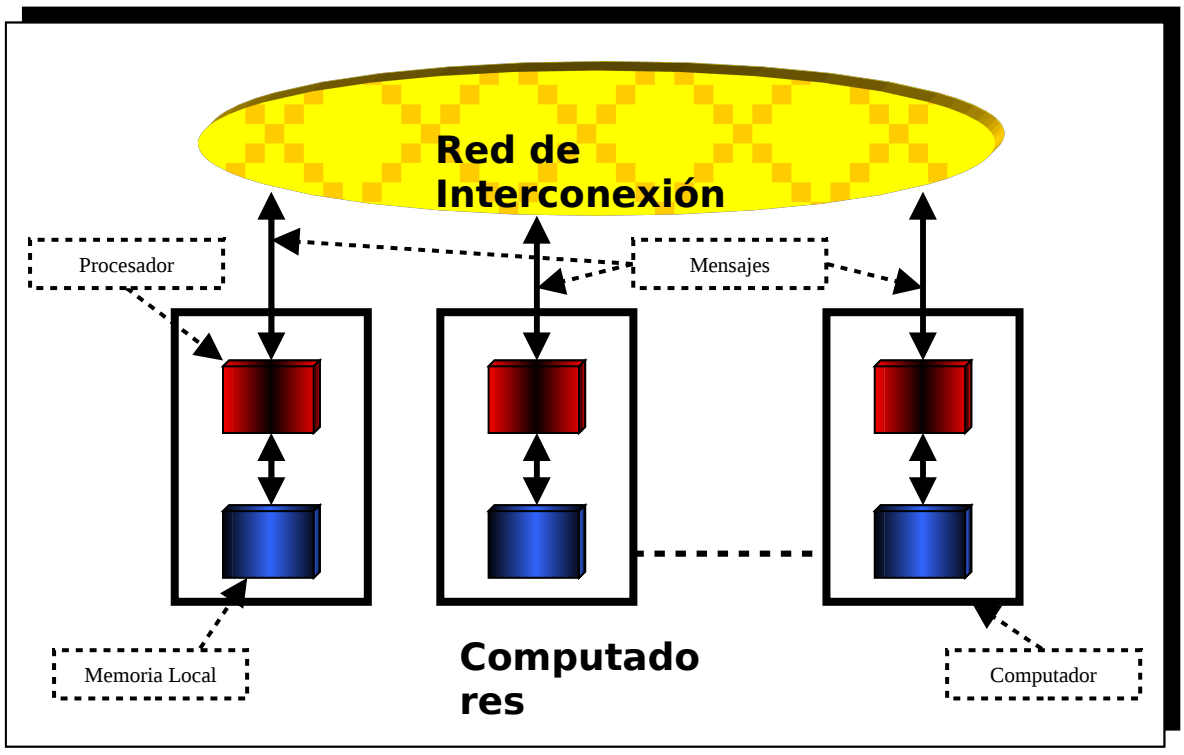


Figura N°3.6: Modelo Multicomputador.

La programación en un Multicomputador involucra dividir el problema

en partes, que son ejecutadas simultáneamente para resolver el problema. Se pueden utilizar lenguajes paralelos, pero es más común usar rutinas de librerías de paso de mensajes unidas a programas secuenciales convencionales, de modo de realizar el intercambio de información, vía mensajes.

Un proceso es un conjunto de instrucciones ejecutables (programa) las cuales corren sobre un procesador. Uno o más procesos se pueden ejecutar sobre el mismo procesador. En un sistema de paso de mensajes, la única forma de intercambiar datos o resultados entre procesos es mediante el envío de mensajes, aún si ellos corren sobre el mismo procesador. Por razones de eficiencia, sin embargo, generalmente en sistemas de paso de mensajes se asocia sólo un proceso por procesador. Si existen más procesos que computadores, entonces, uno o más computadores correrán más de un proceso simultáneamente.

ralelo

con ... es de

com

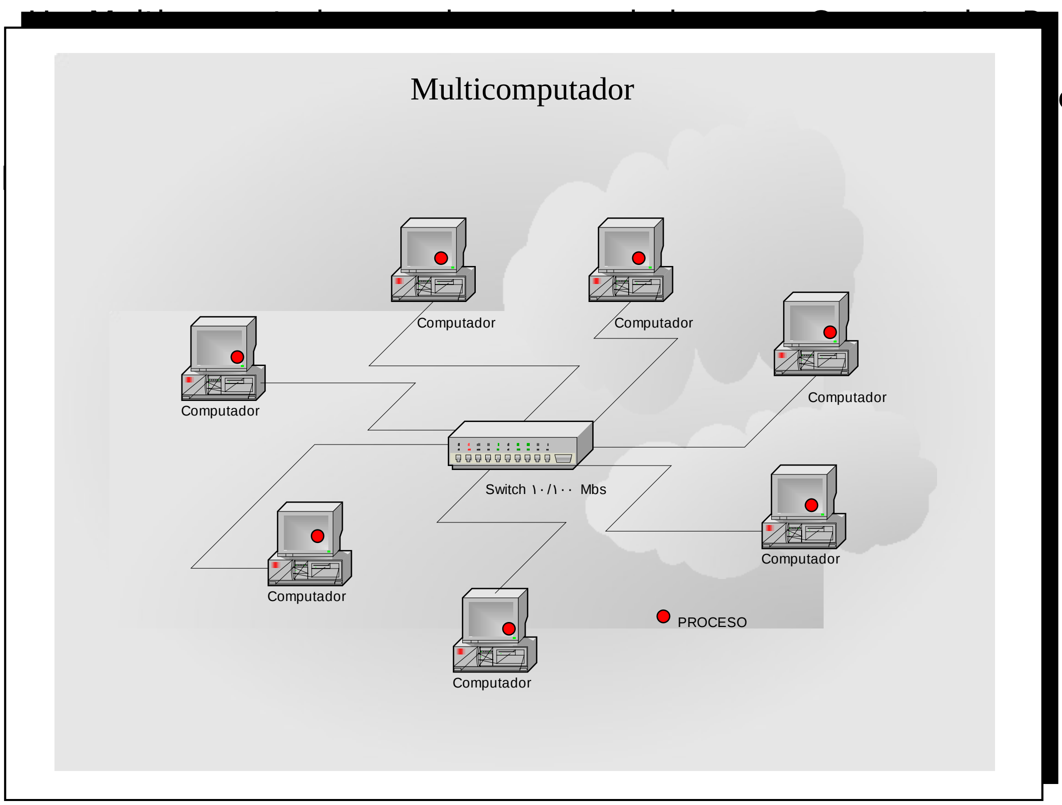


Figura N°3.7: Multicomputador conformado por varios Computadores o estaciones de trabajo.

El paradigma del paso de mensajes tiene la ventaja que no se necesitan mecanismos especiales, para controlar simultáneamente el acceso a los datos. Esos mecanismos pueden incrementar significativamente el tiempo de ejecución de un programa paralelo. Quizás, la razón más significativa para usar el paradigma de paso de mensajes es la aplicabilidad directa de éste en computadores que están conectados en una red, tal como se muestra en la Figura N°3.7.

Muchos de los Sistemas Multiprocesador tienen, cada vez más, una vida útil más corta, debido a la rápida evolución que han tenido los computadores personales.

Es obvio preferir usar un único procesador nuevo en lugar de un multiprocesador viejo, si el nuevo opera *k* veces más rápido que cada uno de los *k* procesadores viejos del multiprocesador. Lo anterior es cierto cuando el nuevo procesador es muchos más barato que el multiprocesador. El uso de computadores interconectados permite que nuevos computadores puedan ser incorporados fácilmente en el sistema.

"*Message Passing Interface*" – MPI

MPI (MPI Forum, 1994; Gropp, Lusk y Skjellum, 1995) –Interfaz de Paso de Mensajes– es una especificación de una interfaz estándar de paso de mensajes, cuyos principales objetivos son la funcionalidad, la eficiencia y la portabilidad. MPI incluye paso de mensajes punto a punto y operaciones colectivas, todas ellas dentro del ámbito de un grupo de procesos especificado por el usuario. En MPI todos los procesos pertenecen a grupos. Si un grupo contiene *n* procesos, éstos se identifican mediante enteros dentro del rango *0*, *n-1*.

El objetivo principal de MPI es lograr la portabilidad a través de diferentes máquinas, tratando de obtener un grado de portabilidad comparable al de un lenguaje de programación que permita ejecutar de manera transparente, aplicaciones sobre sistemas heterogéneos.

A continuación se describen algunas de las principales características que ofrece MPI:

- ***Modos de comunicación.*** MPI soporta operaciones

"bloqueantes", "no bloqueantes". Una operación de envío "bloqueante" detiene al proceso que la ejecuta sólo hasta que el buffer pueda ser reutilizado de nuevo. Una operación "no bloqueante" permite solapar el cálculo con las comunicaciones. En MPI es posible esperar por la finalización de varias operaciones "no bloqueantes". Un envío síncrono bloquea la operación hasta que la correspondiente recepción tiene lugar. En este sentido, una operación "bloqueante" no tiene por que ser síncrona.

- ***Operaciones colectivas*.** Una operación colectiva es una operación ejecutada por todos los procesos que intervienen en un cálculo o comunicación. En MPI existen dos tipos de operaciones colectivas: operaciones de movimiento de datos y operaciones de cálculo colectivo. Las primeras se utilizan para intercambiar y reordenar datos entre un conjunto de procesos. Un ejemplo típico de operación colectiva es la difusión (*broadcast*) de un mensaje entre varios procesos. Las segundas permiten realizar cálculos colectivos como mínimo, máximo, suma, OR lógico, etc., así como operaciones definidas por el usuario.
- ***Tipos de datos*.** MPI ofrece un amplio conjunto de tipos de datos predefinidos (caracteres, enteros, números en coma flotante, etc.) y ofrece la posibilidad de definir tipos de datos derivados. Un tipo de datos derivado es un objeto que se construye a partir de tipos de datos ya existentes. En general, un tipo de datos derivado se

especifica como una secuencia de tipos de datos ya existentes y desplazamientos (en *bytes*) de cada uno de estos tipos de datos. Estos desplazamientos son relativos al buffer que describe el tipo de datos derivado.

- ***Modos de programación*.** Con MPI se puede desarrollar aplicaciones paralelas que sigan el modelo SPMD (*Single Program Multiple Data*) o MPP (*Massively Parallel Processor*). En el caso SPMD, al arrancar una aplicación se lanzan en paralelo *N* copias del mismo programa (procesos). Estos procesos no avanzan sincronizados instrucción a instrucción, sino que la sincronización, cuando sea necesaria, tiene que ser explícita. Los procesos tienen un espacio de memoria completamente separado. El intercambio de información, así como la sincronización, se hace mediante paso de mensajes. Además, la interfaz MPI tiene una semántica "*multithread*", que le hace adecuado para ser utilizado en programas MPI y entornos "*multithread*".
- ***Topologías virtuales*.** Ofrecen un mecanismo de alto nivel para manejar grupos de procesos sin tratar con ellos directamente. MPI soporta grafos y redes de procesos.
- ***Soporte para redes heterogéneas*.** Los programas MPI están pensados para poder ejecutarse sobre redes de máquinas heterogéneas con formatos y tamaños de los tipos de datos elementales totalmente diferentes.

En 1997 se añadieron algunas extensiones al estándar inicial, apareciendo MPI-2 (MPI Forum, 1997). Este nuevo estándar incluye, entre otras características, funcionalidad para la gestión y creación de procesos, nuevas operaciones colectivas y operaciones de E/S (MPI-IO).

A continuación se describen algunas de las implementaciones de MPI existentes en la actualidad:

- LAM (Burns, Daoud y Vaigl, 1994) –*Local Area Multicomputer*– es una implementación disponible en el Centro de Supercomputación de Ohio, que se ejecuta sobre computadores heterogéneos Sun, SGI, DEC, IBM y HP, así como sobre una red de computadores homogéneos o heterogéneos. Con LAM, un cluster o una red de computadores puede comportarse como un computador paralelo resolviendo un problema.
- CHIMP-MPI (Alasdair, Bruce, Mills y Smith, 1994) es una implementación desarrollada en el Centro de Computación Paralela de Edimburgo basada en CHIMP (Clarke, Fletcher, Trewin, Bruce, Smith y Chapple, 1994). Esta implementación se ejecuta sobre estaciones de trabajo Sun, SGI, DEC, IBM y HP, y máquinas Meiko y Fujitsu AP-1000.

- MPICH (Gropp, Lusk, Doss y Skjellum, 1996) es una implementación desarrollada de forma conjunta por el Laboratorio Argonne y la Universidad de Mississippi. Esta implementación ejecuta sobre diferentes plataformas, redes de estaciones de trabajo Sun, SGI, RS6000, HP, DEC, Alpha, y multicomputadores como el IBM SP2, Meiko CS-2 y Ncube.
- Unify (Vaughan, Skjellum, Reese y Chen Cheng, 1995) disponible en la Universidad del Estado de Mississippi, es una implementación que permite el uso de llamadas MPI y PVM de forma simultánea dentro del mismo programa.

Speedup

Una definición convencional del "*Speedup*" de un algoritmo es la razón entre el tiempo de ejecución del mejor algoritmo secuencial, T_s, y el tiempo de ejecución del programa paralelo, T_p (Almasi y Gottlieb, 1994). En este caso, se considera que el mejor algoritmo secuencial es un Algoritmo Genético con una población lo suficientemente grande como para alcanzar la calidad deseada y que use los mismos operadores que el Algoritmo Genético Paralelo.

Capítulo 4. Solución al Problema PCPGBR mediante un Algoritmo Genético Paralelo

Introducción

A continuación se presenta la metodología utilizada para resolver el Problema de Corte de Piezas Guillotina Bidimensional Restricto (PCPGBR) mediante Algoritmos Genéticos Paralelos.

Para resolver este problema, se definen adecuadamente la representación cromosomática de los individuos de la población y la forma en la que se realizarán las operaciones de selección, cruzamiento, mutación y migración.

Además, se presentan tanto la topología a utilizar en el Algoritmo Genético Paralelo, como las políticas de intercambio de individuos.

Representación Cromosomática y Operadores Genéticos

En esta sección se muestran todos los aspectos relacionados directamente con los Algoritmos Genéticos.

Representación Cromosomática

Rojas (2000) representa una solución al problema de corte de pieza mediante un arreglo de largo *n*, donde *n* indica el número de piezas. En cada posición del arreglo se almacena el número, ancho y largo de la pieza. Si una pieza puede ser incluida *k* veces dentro del corte, también se incluye *k* veces en el arreglo.

Por ejemplo, en la Figura N°4.1, para una demanda de cinco tipos diferentes de piezas (ocho piezas en total), se genera el arreglo que se encuentra en la parte inferior de la figura. Las piezas se colocan en el arreglo sin realizar ningún ordenamiento predefinido.

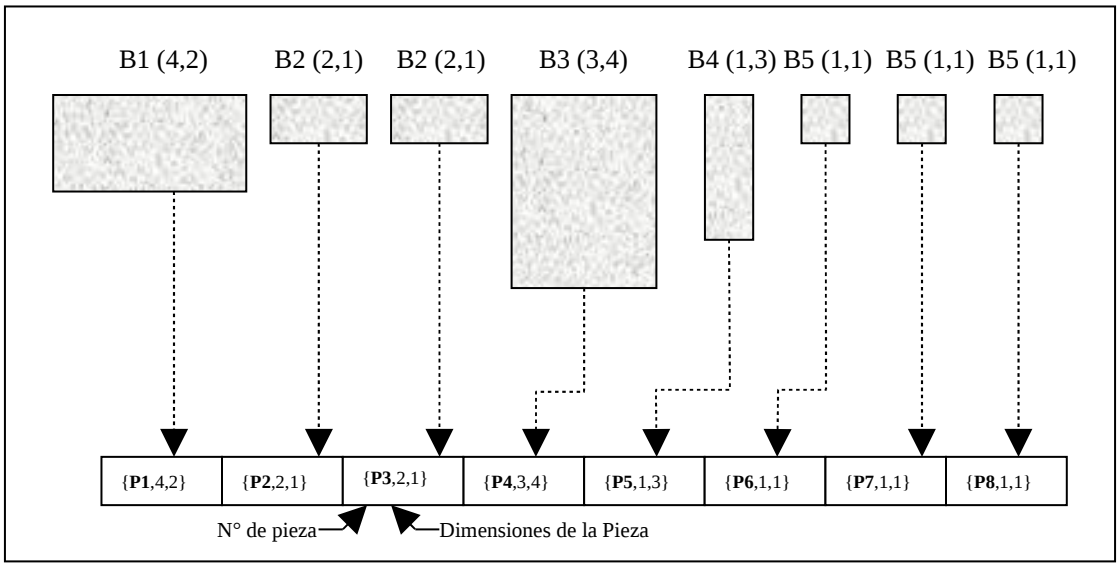


Figura N°4.1: Representación inicial del conjunto de piezas.

Del arreglo de la Figura N°4.1, se obtienen dos cromosomas de 1’s y

0’s tal como se ilustra en la Figura N°4.2.

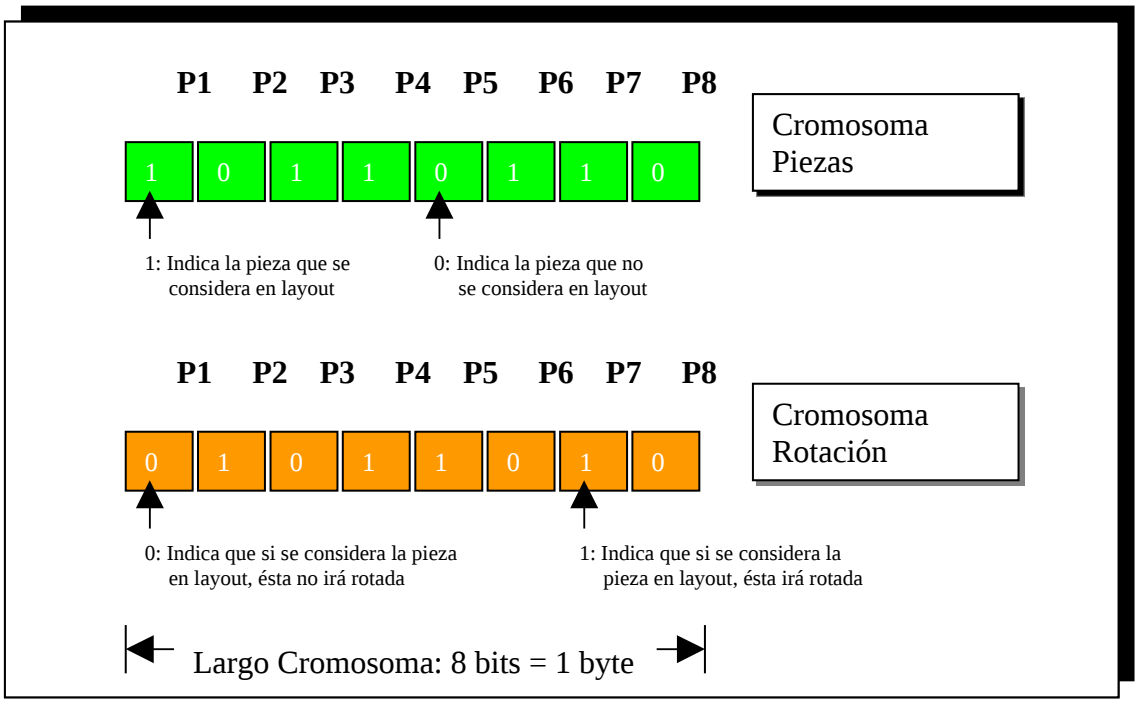


Figura N°4.2: Representación Cromosomática del Problema de Corte de Pieza Guillotina Bidimensional Restricto.

En la Figura N°4.2 se muestran dos cromosomas que representan cromosomáticamente una instancia del PCPGBR:

- **Cromosoma Piezas:** Almacena cada pieza del problema indicando si va o no en el layout del problema. Cada posición (gen binario) del cromosoma representa una pieza del problema. Si $P_x = 1$ (donde $x = 1,\ldots, n$) indica que dicha pieza será considerada, si es que cabe, en el layout del problema. Si $P_x = 0$ (donde $x = 1,\ldots, n$) indica que dicha pieza no será considerada en el layout del problema.

- **Cromosoma Rotación:** Almacena cada pieza del problema indicando si va rotada o no en el layout del problema. Cada posición (gen binario) del cromosoma representa una pieza del problema. Si P_x = 1 (donde x = 1,..., n) indica que si la pieza es considerada en el layout del problema, ésta será rotada. Si P_x = 0 (donde x = 1,..., n) indica que si la pieza es considerada en el layout del problema, ésta no será rotada.

Dado que estos cromosomas involucran a todas las piezas del problema (número ocho en el ejemplo de la Figura N°4.1) y a cada pieza se le asigna un "*bit*", el tamaño de cada cromosoma será n bits. El número de bytes que ocupará cada cromosoma será L:

$$L\,(bytes) = \frac{n}{8} + \begin{cases} 1 \text{ si Módulo}(n,8) \neq 0 \\ 0 \text{ si Módulo}(n,8) = 0 \end{cases} \qquad (4.1)$$

Por ejemplo, si el problema involucra a 45 piezas, L = 5 + 1 = 6 bytes.

El orden de las piezas en los cromosomas de la Figura N°4.2 tiene relación según como éstas fueron colocadas en el arreglo obtenido en la Figura N°4.1. Sin embargo, también se ha considerado agregarle más información a los cromosomas. Esto es, ordenar inicialmente el arreglo de piezas según el área que ocupa cada pieza, según su lado vertical o por su lado horizontal.

La Figura N°4.3 muestra la representación inicial del conjunto de piezas ordenadas por área (las piezas de mayor área se colocan primero de izquierda a derecha en el arreglo, y así sucesivamente) y su

correspondiente representación cromosomática.

La Figura N°4.4 muestra la representación inicial del conjunto de piezas ordenadas por lado vertical de cada pieza (las piezas de mayor lado vertical se colocan primero de izquierda a derecha en el arreglo, y así sucesivamente) y su correspondiente representación cromosomática.

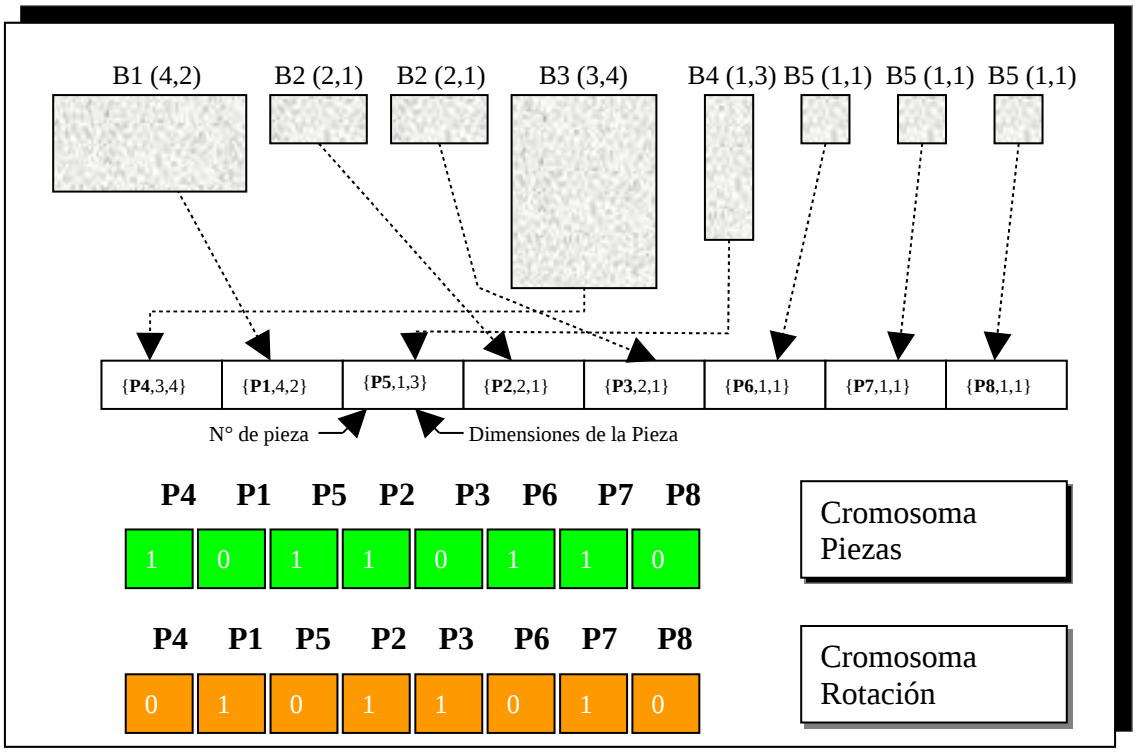


Figura N°4.3: Representación inicial del conjunto de piezas ordenadas por área y su correspondiente representación cromosomática.

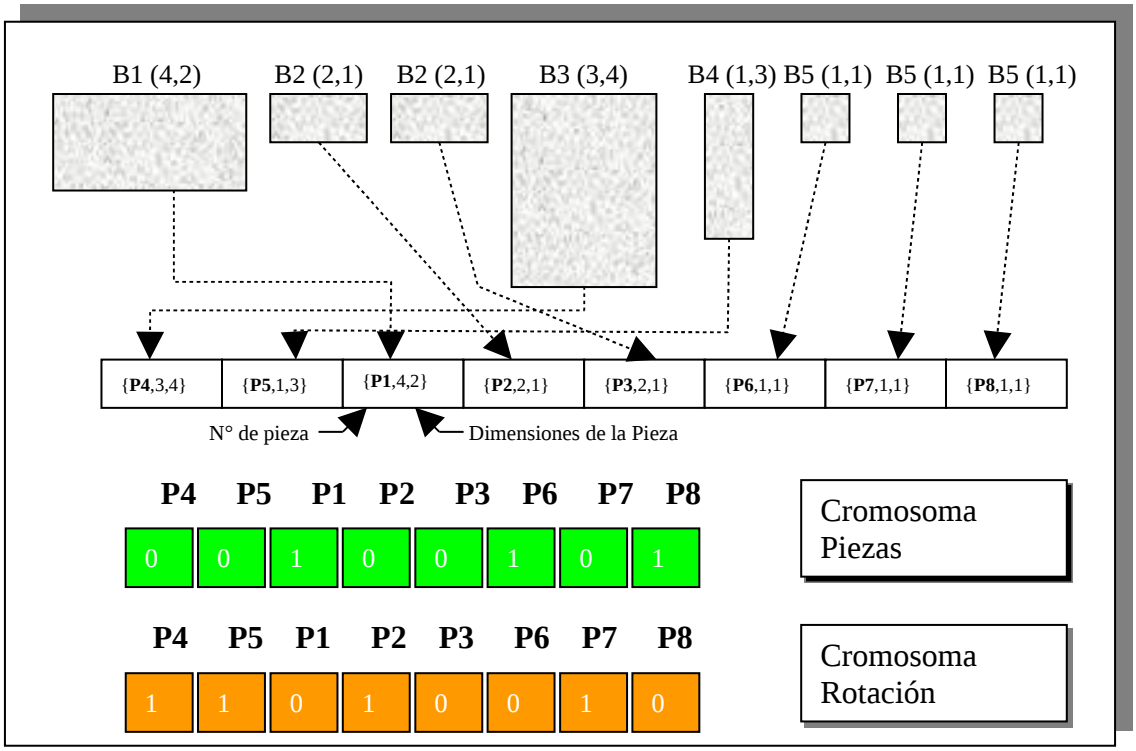


Figura N°4.4: Representación inicial del conjunto de piezas ordenadas por lado vertical y su correspondiente representación cromosomática.

La Figura N°4.5 muestra la representación inicial del conjunto de piezas ordenadas por lado horizontal de cada pieza (las piezas de mayor lado horizontal se colocan primero de izquierda a derecha en el arreglo, y así sucesivamente) y su correspondiente representación cromosomática.

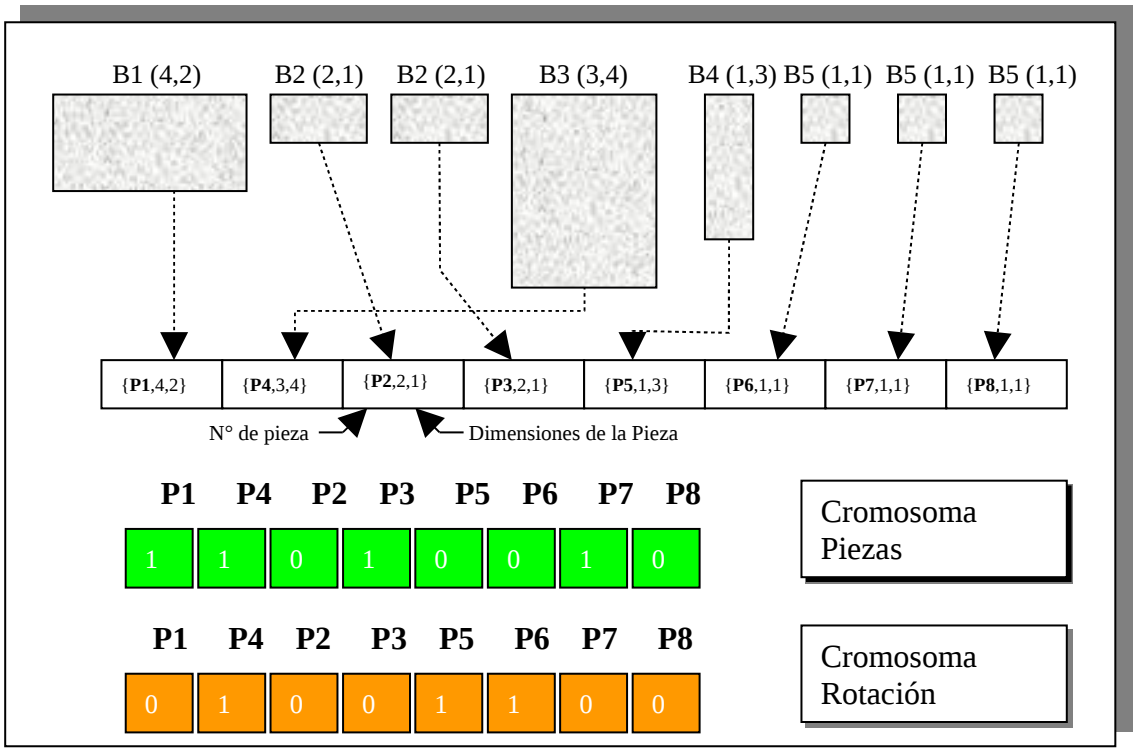


Figura N°4.5: Representación inicial del conjunto de piezas ordenadas por lado horizontal y su correspondiente representación cromosomática.

En el Capítulo 5 se muestran los resultados e implicancias de elegir una u otra configuración inicial.

Operadores Genéticos

A continuación se presentan los operadores genéticos de selección, cruzamiento y mutación implementados para resolver el problema.

La utilización de los operadores genéticos tradicionales es conveniente ya que el problema puede representarse mediante una cadena de unos y ceros. En estas condiciones existe la garantía de convergencia del algoritmo (Goldberg, 1989a).

Selección

El propósito de la selección de padres es incrementar la probabilidad de reproducir miembros de la población que tengan "buenos valores" de la función objetivo. En este caso se utiliza uno de los métodos más comunes, éste es el método de la ruleta. Este método representa a una rueda que gira en la que cada individuo tiene una sección circular que es directamente (o inversamente) proporcional al "*fitness*" del individuo en el caso de maximización (o minimización). Un paso de selección es entonces análogo a un giro de esa ruleta, seleccionando el individuo correspondiente según el arco de círculo donde se detiene.

Dado que el problema a resolver es de minimización[1], se puede obtener el área que ocupará cada individuo en la ruleta mediante la ecuación (4.2).

$$Área(\chi^k) = \frac{\frac{1}{f(\chi^k)}}{\sum_{j=1}^{Q} \frac{1}{f(\chi^j)}} \qquad (4.2)$$

donde,

χ^k : Solución *k*-ésima para la cual se evalúa el área.

[1] En el caso de maximización se puede reemplazar f(x) por 1/f(x).

Q : Tamaño de la Población de soluciones.

$f(\chi^k)$: Valor de la Función Objetivo o "*fitness*" de la solución *k*-ésima.

$\sum_{j=1}^{Q} \frac{1}{f(\chi^j)}$: Costo total de todas las soluciones.

El rango de valores para el $Área(\chi^k)$ es]0, 1]. En la medida que el "*fitness*" de la solución *k*-ésima disminuya, el área será mayor y existirá mayor probabilidad de seleccionar esta solución.

Cruzamiento

El cruzamiento se realiza escogiendo al azar un punto de cruce en cada uno de los padres. Los hijos mantienen los bits a la izquierda del punto de cruce, mientras que los bits a la derecha del punto de cruce se intercambian.

Se realiza el mismo tipo de cruzamiento (cruzamiento de un punto) ta

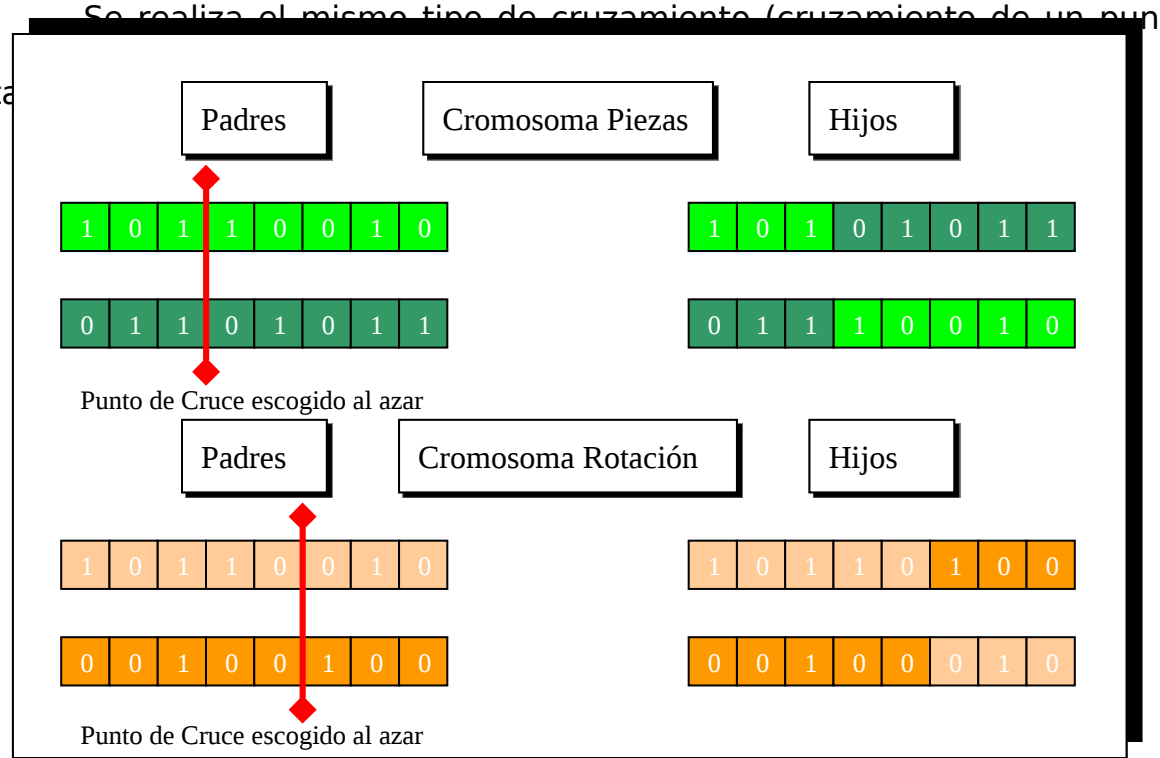


Figura N°4.6: Cruzamiento de Cromosoma Piezas y Cromosoma Rotación.

Mutación

Se han implementado dos maneras de realizar la mutación, éstas son:

- **Mutación de 1 *bit*:** En cada cromosoma se escoge al azar un *bit* a reemplazar. Si el valor del *bit* seleccionado es cero, se cambia a uno y viceversa. La Figura N°4.7 muestra la mutación de un *bit* para ambos cromosomas.
- **Mutación mediante una máscara de *bits*:** En cada cromosoma se genera una máscara de *bits* del mismo tamaño del cromosoma que se está realizando la mutación. Cada *bit* de esta máscara es determinado aleatoriamente. Finalmente, se realiza la operación OR-Exclusivo, *bit* a *bit*, entre cada uno de los elementos del cromosoma y la máscara. La Figura N°4.8 muestra la mutación mediante una Máscara de *bits* para ambos cromosomas.

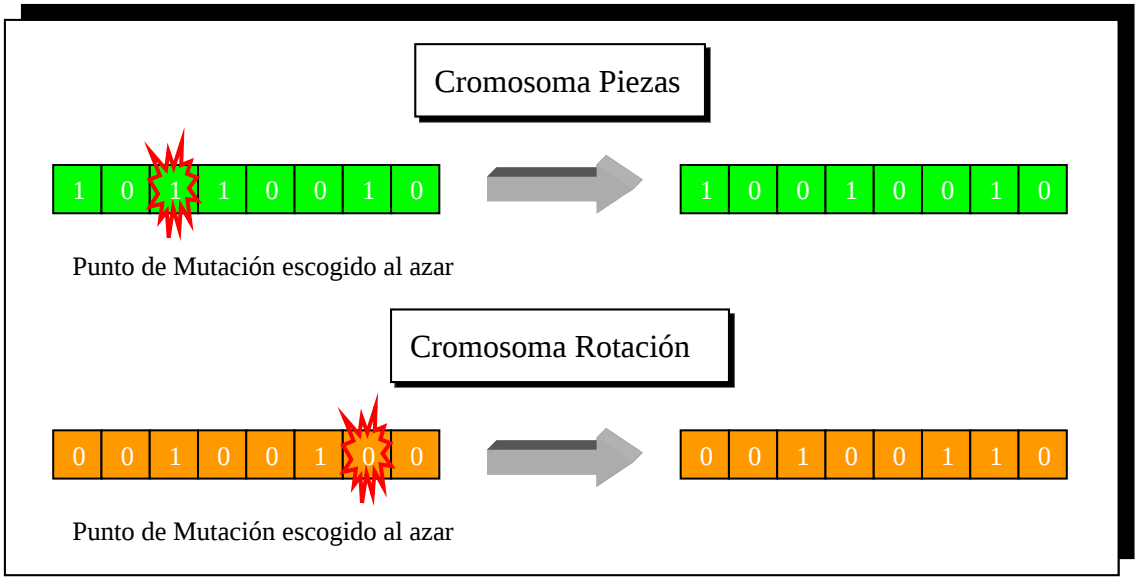


Figura N°4.7: Mutación de un *bit* en Cromosoma Piezas y Cromosoma Rotación.

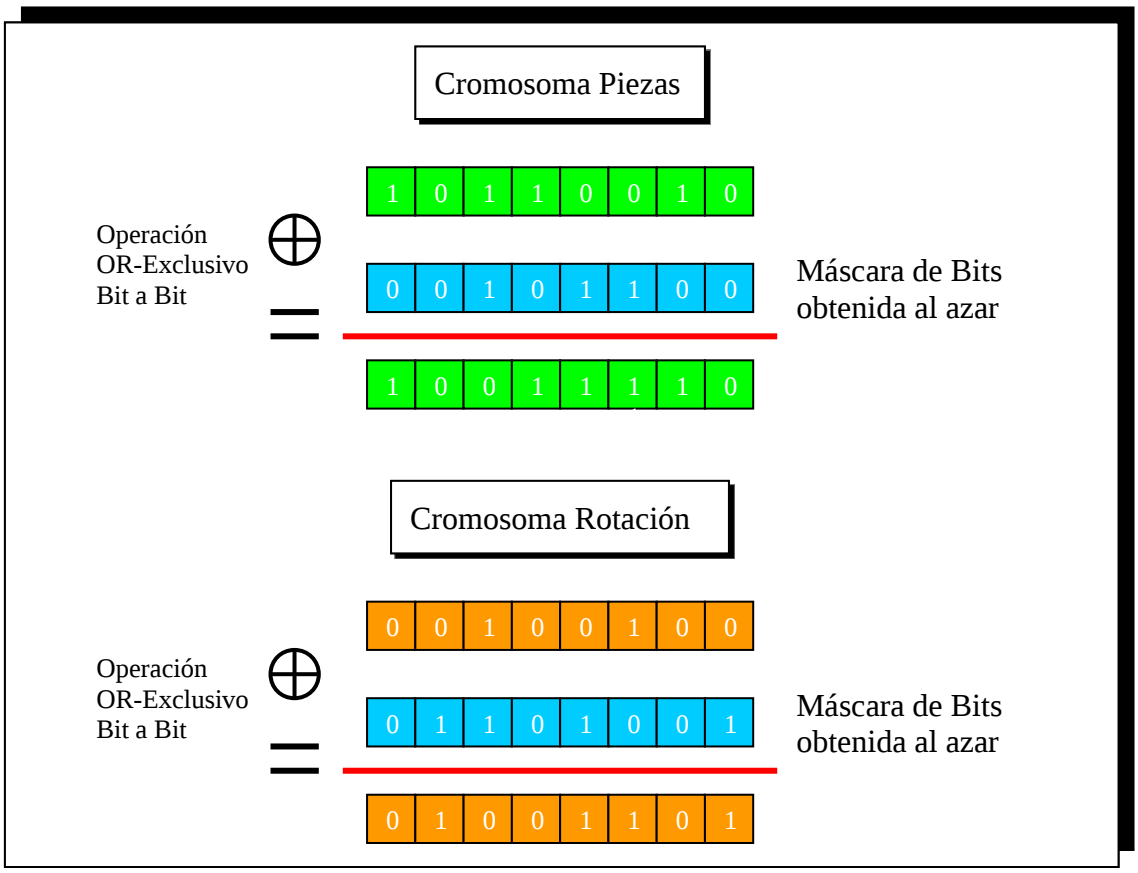


Figura N°4.8: Mutación usando Máscara de *bits* para Cromosoma Piezas y Cromosoma Rotación.

Función de Evaluación

Antes de poder evaluar el fitness de cada individuo, se debe determinar el arreglo de piezas que será valorado por la Función de Evaluación. Este arreglo, denominado “Arreglo Temporal”, se obtiene

modificando la representación inicial del conjunto de piezas de acuerdo con los valores entregados por el Cromosoma Piezas y el Cromosoma Rotación. Esto es, para cada una de las piezas P_x (donde x = 1,..., n) con dimensiones ($ancho_x$, $alto_x$) => (w_x, l_x) del arreglo inicial del conjunto de piezas, se analiza el valor binario correspondiente en el Cromosoma Piezas.

- Si el valor en el Cromosoma Piezas es 1, entonces la pieza será considerada en la función de evaluación. Se analiza, luego, el valor binario correspondiente en el Cromosoma Rotación:
    - Si el valor en el Cromosoma Rotación es 1, entonces se rota la pieza P_x, quedando con dimensiones (l_x, w_x) en el arreglo inicial del conjunto de piezas.
    - Si el valor en el Cromosoma Rotación es 0, entonces la pieza P_x queda con las mismas dimensiones (w_x, l_x) en el arreglo inicial del conjunto de piezas.
- Si el valor en el Cromosoma Piezas es 0, entonces la pieza no será considerada en la función de evaluación, por lo que se le asigna dimensiones (0, 0). En este caso, no se analiza el valor binario de la pieza P_x en el Cromosoma Rotación.

Por ejemplo, si se considera la representación inicial del conjunto de piezas de la Figura N°4.1 y los cromosomas Piezas y Rotación de la Figura N°4.2, el "Arreglo Temporal" a valorizar con la Función de Evaluación es el que se muestra en la Figura N°4.9. Nótese en el ejemplo, que tanto la pieza P_4 como P_7 aparecen rotadas en el "Arreglo Temporal".

El "*fitness*" de cada individuo se calcula mediante una heurística particular, que denominaremos *Combinación Heurística VH* [2], aplicada al "Arreglo Temporal". Esta heurística se detalla a continuación.

[2] VH: Vertical - Horizontal

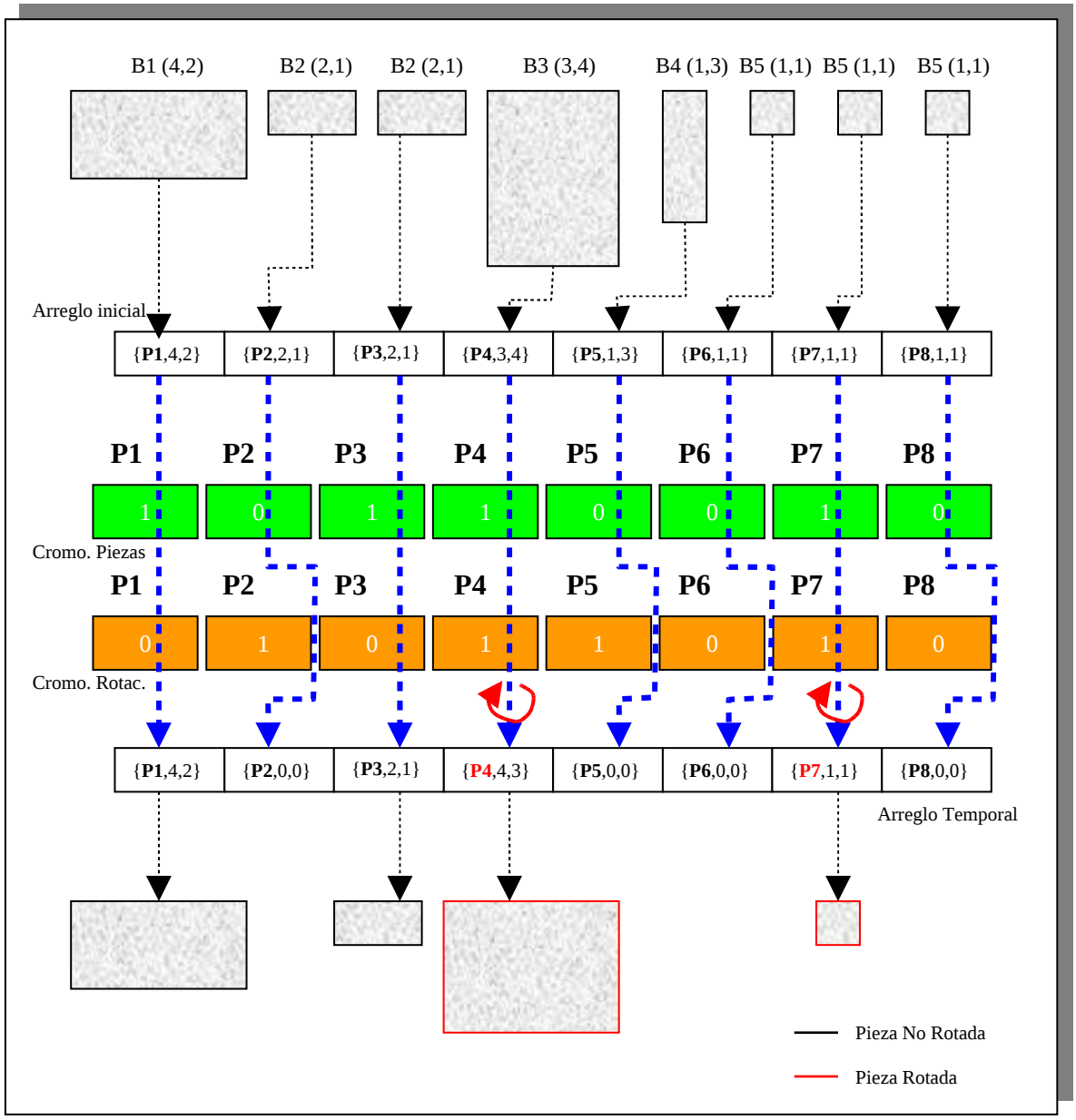


Figura N°4.9: Obtención del "Arreglo Temporal" de piezas a valorizar por la Función de Evaluación.

Combinación Heurística VH

- Considerar del "Arreglo Temporal" las piezas de izquierda a derecha, sólo aquellas con área mayor a cero.

- Colocar la primera pieza factible en el borde izquierdo superior de la lámina.
- Con la siguiente pieza de área mayor a cero, realizar una Combinación Vertical entre ambas piezas siempre que el nuevo patrón de corte quepa dentro de la lámina. Caso contrario, utilizar Combinación Horizontal. Si nuevamente no cabe el patrón de corte en la lámina, se descarta la pieza y se continúa con la siguiente hasta lograr introducir el nuevo patrón de corte en la lámina.
- Luego, existen dos opciones:
    - Si la pieza siguiente cabe dentro de alguna región[3] que representa una pérdida producida en una etapa anterior se reemplaza aquella región por un nuevo patrón de corte.
    - Si no, se intenta unir el patrón anteriormente generado con la nueva pieza utilizando combinación vertical u horizontal dependiendo de las restricciones en el tamaño de la placa, con lo cual se crea un nuevo patrón.
- Los criterios de términos son:
    - Si se han considerado todas las piezas del “Arreglo Temporal”.
    - Si al incluir más piezas, no se pueden generar más patrones, reemplazando las pérdidas existentes o

[3] 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.

agregando nuevos patrones.

Esta heurística fue desarrollada e implementada con éxito por Rojas (2000).

Este proceso es determinístico ya que siempre que se evalúe un mismo “Arreglo Temporal” dará el mismo resultado. Esto no hace que el Algoritmo Genético utilizado sea menos aleatorio, ya que la aleatoriedad del algoritmo se encuentra en el proceso evolutivo.

Como ejemplo, considérese el “Arreglo Temporal” de la Figura N°4.9. Del primer punto de la “Combinación Heurística VH”, se deduce que la primera pieza a insertar es P_1, y que las piezas P_2, P_5, P_6 y P_8 no serán consideradas.

De los siguientes puntos de la “Combinación Heurística VH”, se van formando los patrones de cortes guillotinables, tal como se muestra en la Figura N°4.10.

De la Figura N°4.10 se observa que en el 2° Paso se realiza una Combinación Vertical, en el 3° paso se realiza una Combinación Horizontal, mientras que en el 4° paso se inserta una pieza dentro de una “Pérdida Interna”. Finalmente, se obtiene un patrón de corte guillotinable con “Pérdida Interna” y “Pérdida Externa”, por lo tanto, el “*fitness*” del individuo conformado por los Cromosomas Pieza y Cromosoma Rotación de la Figura N°4.9 será la suma de las “Pérdidas Internas” más la “Pérdida Externa”.

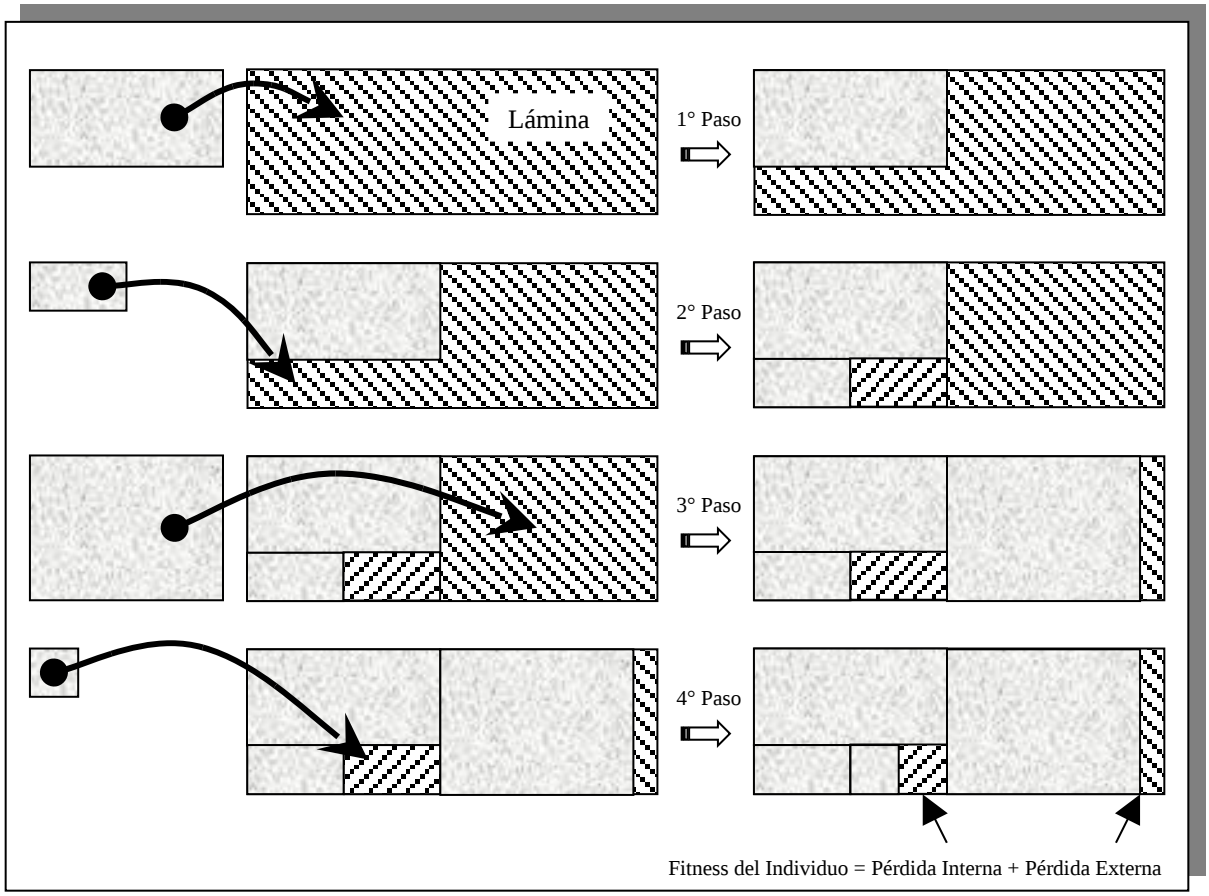


Figura N°4.10: Obtención del *Fitness* de un Individuo mediante Combinación Heurística VH aplicada al Arreglo Temporal de la Figura N°4.9.

Entonces, la función de evaluación del *fitness* queda conformada por la ecuación (4.3):

$$F(x) = P_E + P_I = \underbrace{(W\ L - w_p l_p)}_{} + \underbrace{\left(\sum_{i=1}^{k} w_i l_i\right)}_{}$$

(4.3)

P_E P_I

donde,

$F(x)$: Función de Evaluación del *fitness* de un individuo,

P_E : Pérdida Externa,

P_I : Pérdida Interna total del Patrón de Corte,

WL : Área total ocupada por la lámina (W : Ancho, L : Largo),

$w_p l_p$: Área total ocupada por el Patrón de Corte (w_p : Ancho, l_p : Largo),

$w_i l_i$: Área ocupada por i-ésima Pérdida Interna (w_i : Ancho, l_i : Largo),

k : Número total de Pérdidas Internas.

La función de Evaluación se calcula una vez que se ha determinado la superficie de cada una de las “Pérdidas Internas” y la superficie del Patrón de Corte final.

Tamaño del Espacio de Soluciones y fracción del espacio recorrido

El tamaño del espacio de soluciones que se obtiene con esta representación cromosomática es 3^n, donde *n* corresponde al número total de piezas del problema.

Por lo tanto, el porcentaje del espacio recorrido por el algoritmo está dado en la ecuación 4.4:

$$\varphi = 100 \times \left(\frac{N°Generaciones \times N°Individuos\ de\ la\ Población}{3^n} \right) \qquad (4.4)$$

Esto quiere decir, por ejemplo, si se desea resolver un Problema de Corte de Piezas Guillotina Bidimensional Restricto de 100 piezas (*n* = 100) con Algoritmos Genéticos que utilicen la Representación Cromosomática de la Figura N°4.2, considerando 1.000 Generaciones y una Población Total de 4.000 Individuos, el porcentaje del espacio recorrido será $\varphi = 7{,}76e-40\ \%$.

No se ha encontrado referencia en la literatura respecto a la relación entre el tamaño del espacio de soluciones y la cantidad de soluciones evaluadas durante el proceso de búsqueda.

Modelo Paralelo Propuesto

Para resolver el PCPGBR mediante AGP, se considera un Modelo Paralelo que utiliza Algoritmos Genéticos con Múltiples Sub-Poblaciones (detalles en el Capítulo 3). En esta configuración, existen *N* procesadores, donde uno realiza la función de "Coordinador", mientras que (*N – 1*) procesadores corren cada uno un Algoritmo Genético Secuencial con las mismas características mencionadas en el punto 4.2.

La Figura N°4.11 muestra el AGP con Múltiples Sub-Poblaciones propuesto.

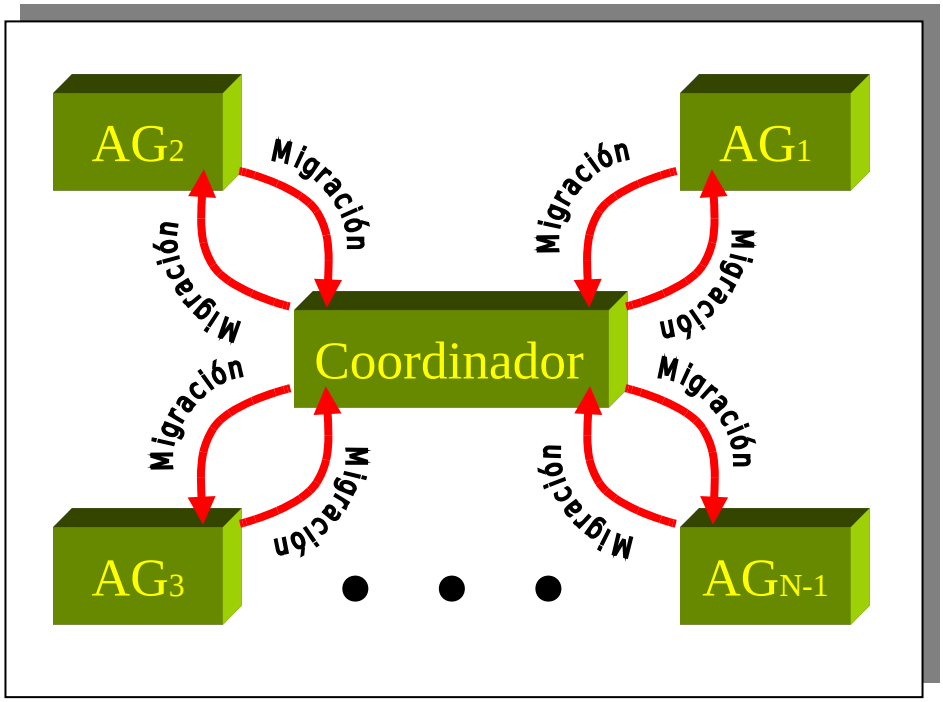


Figura N°4.11: Modelo Paralelo de AG con Múltiples Sub-Poblaciones.

En la Figura N°4.11, se aprecia que el modelo paralelo contempla la ejecución de múltiples (*N-1*) Algoritmos Genéticos Secuenciales que

intercambian información, mediante migración (esta migración puede ser síncrona como asíncrona), con un “Coordinador”. La población global se divide en los (*N-1*) AG Secuenciales, conformando (*N-1*) Sub-Poblaciones.

El Coordinador tiene la misión de regular el proceso de intercambio de individuos entre los AGs; debe conocer los parámetros del sistema y del problema a resolver de modo de entregar a cada AG la información necesaria para que éstos puedan resolver, mediante un proceso evolutivo, el problema planteado. Una vez finalizado el proceso evolutivo debe proveer el layout final al usuario.

Cada AG debe resolver, mediante los operadores genéticos, el problema planteado y debe entregar la información necesaria al Coordinador para que éste pueda discriminar y evaluar el avance de cada AG.

Los pasos más importantes a seguir por el Algoritmo Genético Paralelo se presentan en las Figuras N°4.12 y N°4.13. En estas figuras se muestran elementos nítidos y difuminados. Cuando un elemento se presenta en forma nítida, significa que es relevante en la etapa del proceso que se está mostrando, sin embargo, cuando se presenta difuminado, indica que éste no es relevante en dicha etapa. Por ejemplo, en la etapa a) de la Figura N°4.12, los AG’s no intervienen, por lo que están difuminados y debido a que el Usuario envía información al Coordinador, ambos elementos se dibujan nítidos.

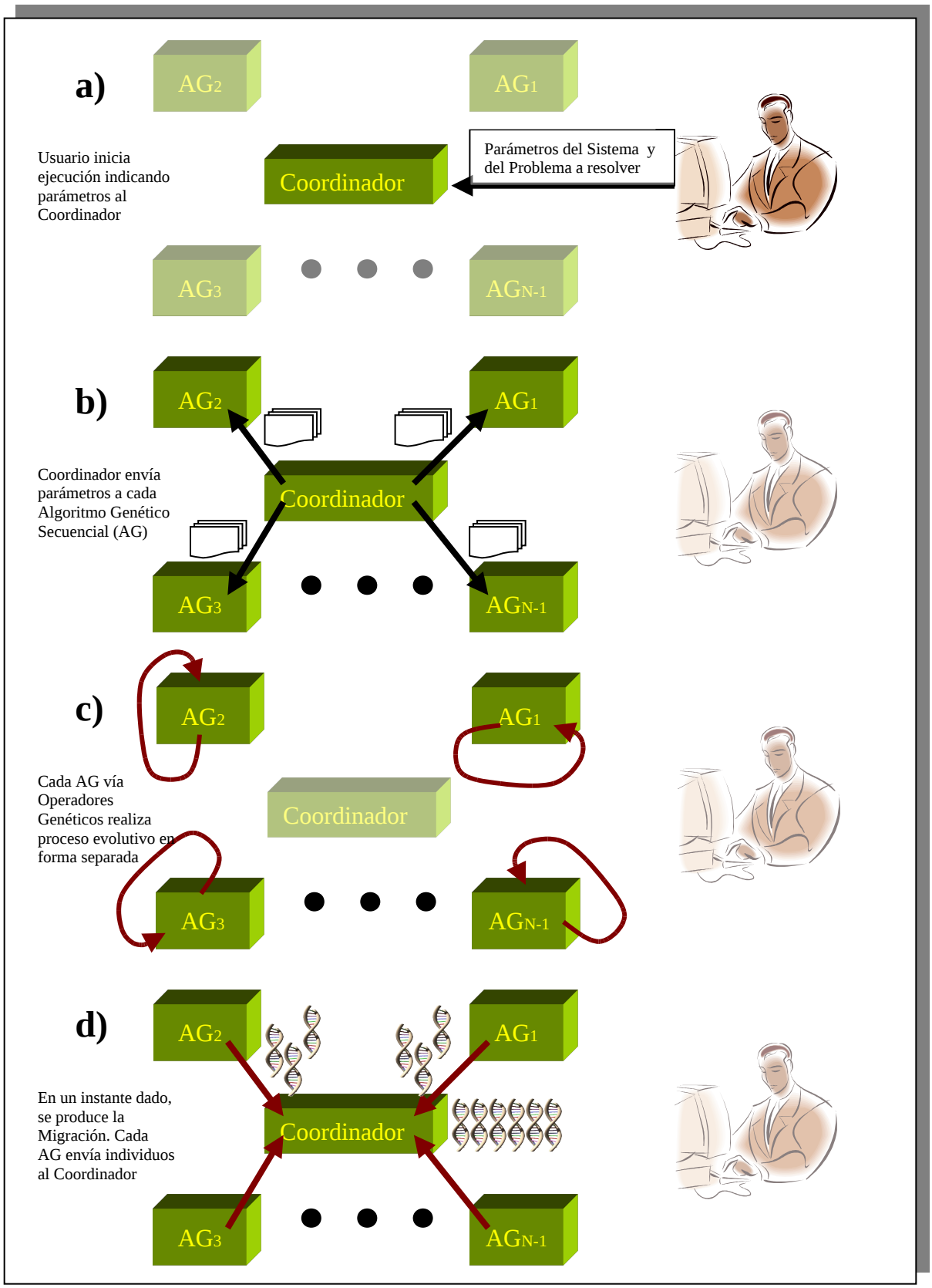

a)
AG2
AG1
Usuario inicia ejecución indicando parámetros al Coordinador
Coordinador
Parámetros del Sistema y del Problema a resolver
AG3
AGN-1
b)
AG2
AG1
Coordinador envía parámetros a cada Algoritmo Genético Secuencial (AG)
Coordinador
AG3
AGN-1
c)
AG2
AG1
Cada AG vía Operadores Genéticos realiza proceso evolutivo en forma separada
Coordinador
AG3
AGN-1
d)
AG2
AG1
En un instante dado, se produce la Migración. Cada AG envía individuos al Coordinador
Coordinador
AG3
AGN-1

Figura N°4.12: Pasos a seguir por Algoritmo Genético Paralelo.

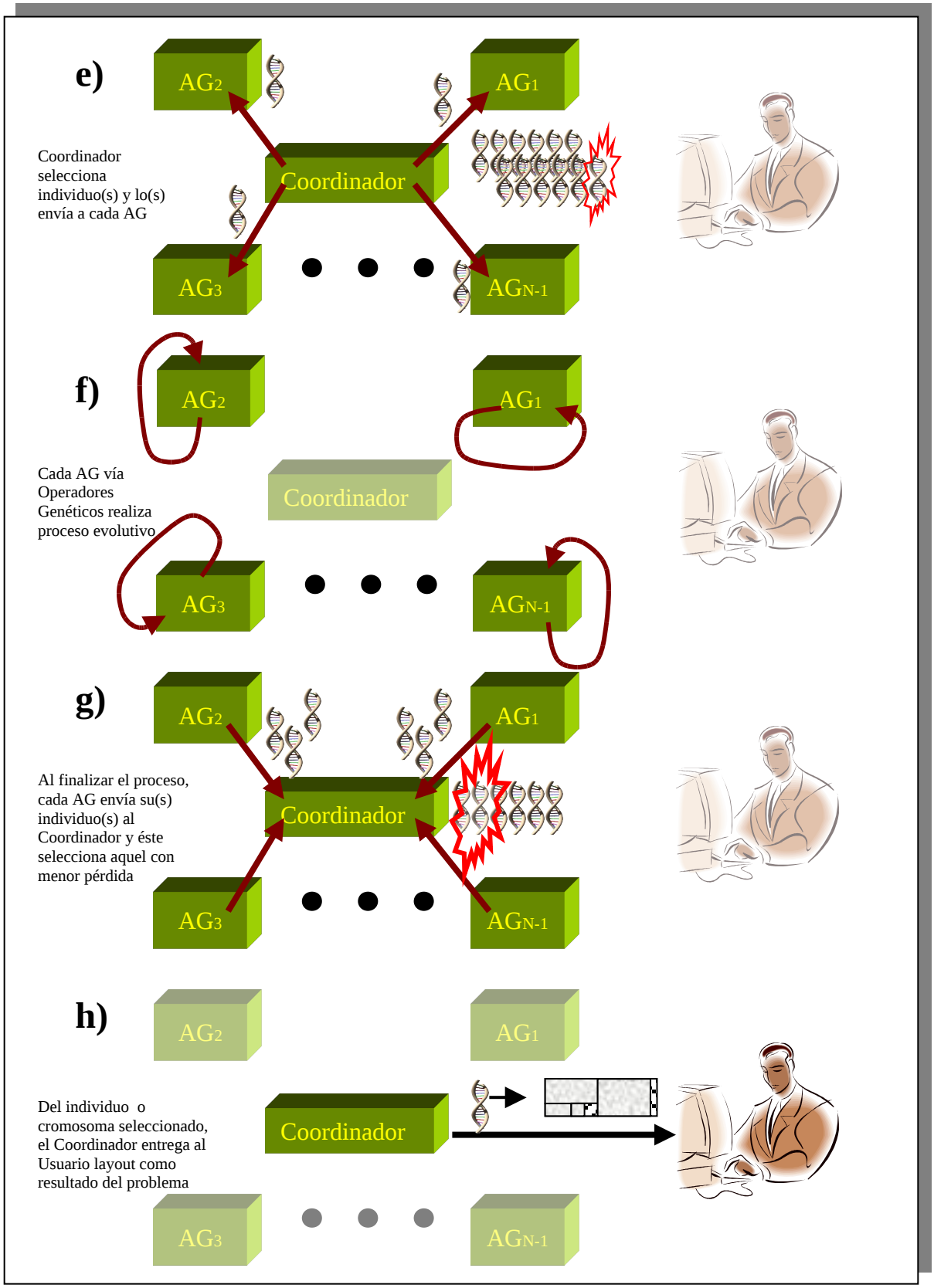


Figura N°4.13: Continuación de los pasos a seguir por Algoritmo Genético Paralelo.

En la etapa a) de la Figura N°4.12, el usuario alimenta al Coordinador con los parámetros del sistema y datos del problema a resolver. Posteriormente, en b) el Coordinador alimenta a cada AG con los parámetros necesarios para que cada uno pueda correr un Algoritmo Genético Secuencial. En la etapa c) se realiza un proceso evolutivo en cada AG, simultáneamente. En un instante dado, en d) comienza el proceso de migración, donde cada AG entrega el(los) individuo(s) al Coordinador, seleccionado(s) bajo algún criterio. Luego, en la etapa e) de la Figura N°4.13, el Coordinador procede a seleccionar el(los) individuo(s), también mediante algún criterio, y lo(s) envía(n) a todas las Sub-Poblaciones, de modo que éstas introduzcan el(los) individuo(s) mediante alguna política de reemplazo. En esta etapa es donde se produce el intercambio de información, entre los distintos elementos del sistema que colaboran en la búsqueda de mejores soluciones para el problema que se está resolviendo. Con los nuevos individuos en la Sub-Población de cada AG (etapa f), se continúa el proceso evolutivo hasta que se origine un nuevo proceso de migración. Al producirse la migración (etapa g), cada AG envía el(los) individuo(s) al Coordinador, y si se cumple con algún criterio de parada del algoritmo (se encuentra el óptimo o se cumple el número máximo de generaciones), el Coordinador selecciona aquel individuo con menor

pérdida y genera el layout en un archivo de resultados (etapa h).

A continuación se resumen las funciones de cada uno de los agentes que interviene en el sistema.

Coordinador:

- Carga, distribuye información y coordina migración de individuos con cada AG.
- Lee parámetros del sistema como: problema a resolver, tamaño población, probabilidad de cruzamiento, probabilidad de mutación, frecuencia de migración, cantidad de individuos a migrar, modo en que se realiza la migración (asíncrona o síncrona), etc.
- Envía, mediante mensajes, los parámetros del sistema a cada AG.
- Lee el problema a resolver como: dimensiones de la lámina, cantidad de piezas del problema, dimensiones y restricciones de cada pieza del problema.
- Envía a cada AG, mediante mensajes, los datos del problema a resolver.
- Envía un mensaje a cada AG, dando la orden para comenzar la evolución.
- Si la migración se efectúa en forma asíncrona, el Coordinador debe esperar hasta que algún AG le indique que ha completado el número de generaciones necesario para realizar la migración. En

ese instante, el Coordinador envía un mensaje a todos los AGs indicándoles que deben enviar su(s) individuo(s), dándose por iniciado el proceso de migración.

- Si la migración se efectúa en forma sincrónica, se establece una barrera de sincronización que se satisface cuando todos los AGs han completado el número de generaciones predeterminado para realizar el proceso de migración. En ese instante, el Coordinador envía un mensaje a todos los AG indicándoles que deben enviar su(s) individuo(s), dándose por iniciado el proceso de migración.
- Una vez recibidos todos los individuos, debe seleccionar mediante alguna política, un grupo de individuos los cuales debe enviar a cada AG. El Coordinador no evalúa la pérdida en cada individuo recibido ya que este valor viene implícito dentro de la información de cada individuo.
- El Coordinador evalúa la condición de término. Si en los individuos recibidos se encuentra el óptimo global, el Coordinador envía mensaje a cada AG para que dé término a su proceso evolutivo. En caso contrario, continúa el proceso evolutivo hasta que se complete el número de generaciones preestablecido.

- Cuando se satisface la condición de término, el Coordinador selecciona el mejor individuo entre todos los individuos recibidos durante los procesos migratorios y con éste genera un archivo de resultados, en donde se especifica el layout que resuelve el problema.
- Finalmente, el Coordinador continúa resolviendo otros problemas o, cuando se agotan los problemas suministrados por el usuario, termina por completo el procesamiento.

Cada Algoritmo Genético Secuencial:

- Debe recibir los parámetros del sistema entregados por el Coordinador.
- Debe recibir datos del problema a resolver, suministrados por el Coordinador.
- Con los datos del problema, debe generar al azar su propia Población Inicial (Sub-Población de tamaño equivalente a la Población global dividida por el número de AGs involucrados en el proceso evolutivo).
- Queda a la espera de recibir un mensaje de parte del Coordinador, para comenzar la evolución.
- Realiza la selección, cruzamiento y mutación sobre la sub-población local.

- Puede dar inicio al Proceso de Migración ya sea porque recibe una orden desde el Coordinador o porque se ha cumplido con alguno de los siguientes criterios para migrar:
    - Se ha alcanzado el número de generaciones para migrar (frecuencia de migración).
    - Se ha encontrado el óptimo.
    - Se ha alcanzado el número máximo preestablecido de generaciones.
- Al dar inicio al Proceso de Migración, cada AG seleccionan de acuerdo a alguna política, individuos para ser enviados al Coordinador. Adicionalmente, se reciben individuos desde el Coordinador y se insertan, mediante alguna política de reemplazo, en la Sub-Población local.
- Una vez que el AG ha finalizado su proceso evolutivo, envía un mensaje al Coordinador indicando que ha concluido el procesamiento, y queda a la espera de la confirmación del Coordinador para finalizar ejecución.

Las funciones de cada AG mencionadas anteriormente, pueden apreciarse con mayor claridad en el diagrama de flujo de la Figura N°4.15.

El diagrama de flujo del Coordinador se muestra en la Figura N°4.16.

En las Figuras N°4.15 y N°4.16, se han utilizado distintas formas para

representar el paso de mensajes entre cada AG y el Coordinador. La Figura N°4.14.A representa un mensaje que se envía desde un AG hacia el Coordinador. Mientras que la Figura N°4.14.B representa un mensaje que debe enviarse, simultáneamente, desde el Coordinador a cada AG.

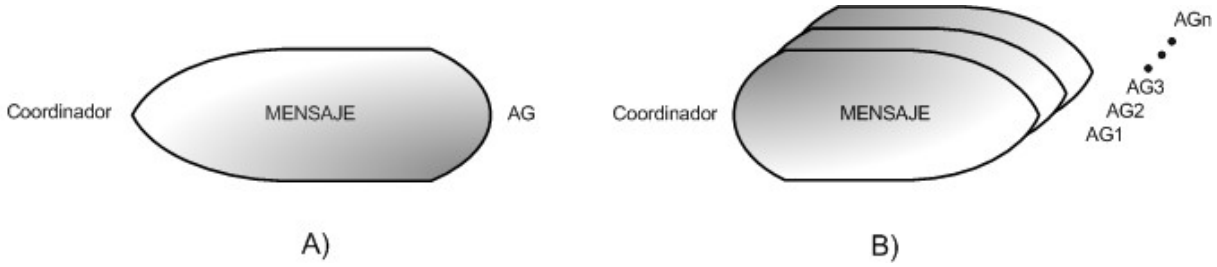


Figura N°4.14: Formas que representan los tipos de mensajes enviados desde un AG al Coordinador y viceversa.

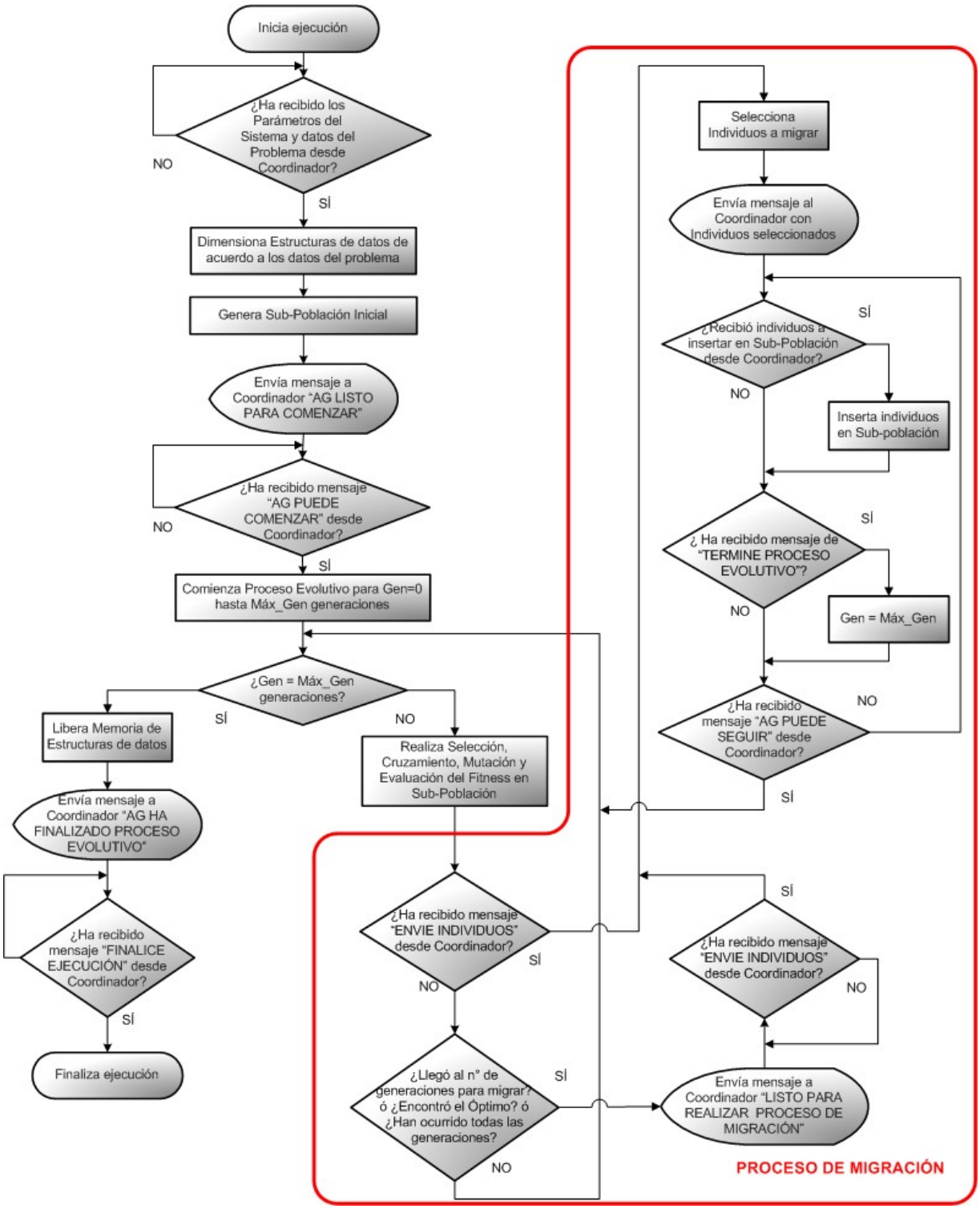


Figura N°4.15: Diagrama de Flujo de los Procesos que debe realizar cada

AG Secuencial.

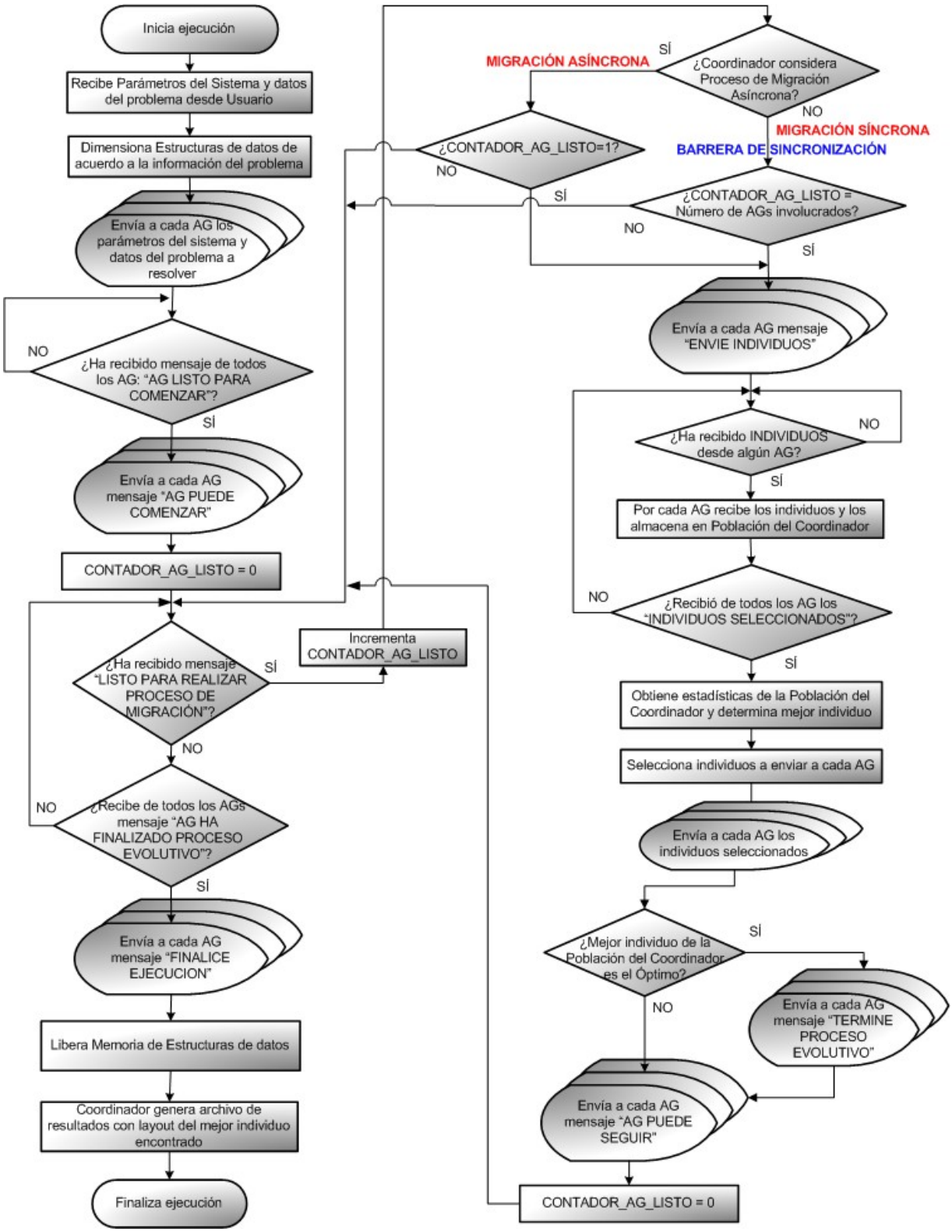


Figura N°4.16: Diagrama de Flujo de los Procesos que debe realizar el

Coordinador.

En la Figura N°4.16 la diferencia entre Migración Asíncrona y Síncrona corresponde a que en la Migración Asíncrona el Coordinador puede iniciar el proceso de migración en cualquier instante de tiempo a solicitud de algún AG. Mientras que en el caso Síncrono, se inicia el proceso de migración sólo una vez que todos los AGs han alcanzado el mismo número de generación necesario para migrar (se establece una barrera de sincronización). Es decir, si la migración ocurre cada 100 generaciones, en el caso Asíncrono, el Coordinador deberá esperar sólo hasta que el AG más rápido llegue a las 100 generaciones. Mientras que, en el caso Síncrono, deberá esperar que todos los AG lleguen a la generación 100 para iniciar el proceso de migración.

Nótese que el procesamiento de cada AG no depende del tipo de Migración Asíncrona o Síncrona, sólo el Coordinador se ve afectado por el tipo de migración.

Los mensajes a enviar desde cada AG al Coordinador pueden ser de tamaño fijo o variable. A continuación se mencionan cada uno de ellos:

- **“AG LISTO PARA COMENZAR”:** Mensaje de tamaño fijo que se envía al Coordinador una vez que el AG ha generado su Población inicial y está listo para comenzar el proceso evolutivo.
- **“LISTO PARA REALIZAR PROCESO DE MIGRACIÓN”:** Mensaje de tamaño fijo que se envía al Coordinador una vez que el AG ha alcanzado el número de generaciones para

migrar, ha encontrado el óptimo o una vez que se han realizado todas las generaciones preestablecidas.

- **“INDIVIDUOS SELECCIONADOS”:** Mensaje de tamaño variable que se envía al Coordinador desde cada AG. Este mensaje contiene una fracción de individuos de la Sub-Población según se especifique en el archivo de entrada del Apéndice A. Este mensaje es de tamaño variable ya que el tamaño de cada individuo depende de la cantidad de piezas del problema a resolver (ver Cromosoma de Piezas y Cromosoma Rotación en la Figura N°4.2) y cada problema puede tener distinto número de piezas.
- **“AG HA FINALIZADO PROCESO EVOLUTIVO”:** Mensaje de tamaño fijo que se envía al Coordinador una vez que el AG ha finalizado su proceso evolutivo y está listo para finalizar su ejecución.

Los mensajes a enviar desde el Coordinador a cada AG también pueden ser de tamaño fijo o variable. Además, los mensajes del Coordinador se envían simultáneamente a cada AG. A continuación se menciona cada uno de ellos:

- **“PARÁMETROS DEL SISTEMA Y DATOS DEL PROBLEMA A RESOLVER”:** Mensaje de tamaño variable que se envía desde el Coordinador a cada AG, indicando los parámetros genéticos

obtenidos del archivo de entrada de la Figura N°A.1 del Apéndice A y los datos del problema a resolver obtenidos del archivo de entrada de la Figura N°A.2 del Apéndice A. El tamaño variable se debe a que diversos problemas pueden tener distintas cantidades de piezas que considerar.

- **“AG PUEDE COMENZAR”:** Mensaje de tamaño fijo que se envía desde Coordinador a todos los AGs, dando la partida al proceso evolutivo. Este mensaje es la respuesta al mensaje “AG LISTO PARA COMENZAR”, enviado por cada AG.
- **“ENVÍE INDIVIDUOS”:** Mensaje de tamaño fijo que se envía desde el Coordinador a todos los AGs, indicándoles que deben comenzar con el proceso de migración.

- **“INDIVIDUOS SELECCIONADOS”:** Mensaje de tamaño variable que se envía desde el Coordinador a cada AG. Este mensaje contiene una fracción de individuos de los que recibe el Coordinador, según se especifique en el archivo de entrada del Apéndice A. Este mensaje es de tamaño variable ya que el tamaño de cada individuo depende de la cantidad de piezas del problema a resolver (ver Cromosoma de Piezas y Cromosoma Rotación en la Figura N°4.2) y cada problema puede tener distintas cantidades de piezas.
- **“TERMINE PROCESO EVOLUTIVO”:** Mensaje de tamaño fijo que se envía desde el Coordinador a todos los AGs, indicándoles que deben finalizar el proceso de migración, ya que el Coordinador ha encontrado un individuo que representa una solución óptima (pérdida cero).
- **“AG PUEDE SEGUIR”:** Mensaje de tamaño fijo que se envía desde el Coordinador a todos los AGs, indicándoles que pueden continuar con su proceso evolutivo hacia una siguiente generación o que pueden finalizar su ejecución.
- **“FINALICE EJECUCIÓN”:** Mensaje de tamaño fijo que se envía desde el Coordinador a todos los AGs, indicándoles que pueden finalizar su ejecución. Este mensaje es la respuesta al mensaje “AG HA FINALIZADO PROCESO EVOLUTIVO”, enviado por cada

AG.

En el proceso de migración, así como cada AG debe seleccionar uno o un grupo de individuos a enviar, el Coordinador también debe seleccionar uno o un grupo de individuos a enviar a cada AG. La cantidad de individuos a enviar y recibir por cada AG, es parámetro del sistema, expresado en porcentaje del tamaño de población local de cada AG.

Las ecuaciones 4.5 a 4.8 muestran cómo calcular el tamaño de la sub-población de cada AG, la cantidad de individuos a enviar y recibir por cada AG, y el tamaño de la población del Coordinador, respectivamente.

$$T_{AG} = \frac{T_{PG}}{N_{AG}} \tag{4.5}$$

$$C_{ie} = P_{ie} \bullet T_{AG} \tag{4.6}$$

$$C_{ir} = P_{ir} \bullet T_{AG} \tag{4.7}$$

$$T_{C} = C_{ie} \bullet N_{AG} \tag{4.8}$$

donde,

T_{PG} : Tamaño de la Población Global de individuos,

N_{AG} : Cantidad de Algoritmos Genéticos Secuenciales involucrados,

T_{AG} : Tamaño de la Sub-Población de cada AG,

P_{ie} : Porcentaje de individuos a enviar desde cada AG al Coordinador,

C_{ie} : Cantidad de individuos a enviar desde cada AG al Coordinador,

P_{ir} : Porcentaje de individuos a enviar desde el Coordinador a cada AG,

C_{ir} : Cantidad de individuos a enviar desde el Coordinador a cada AG,

T_C : Tamaño de la Población del Coordinador.

Por ejemplo, si el tamaño total de la población es 2.000 individuos y se consideran 5 AG, según la ecuación (4.5) cada AG tendrá una Sub-población de 400 individuos. Si el porcentaje de individuos a enviar y recibir es 10% y 20%, respectivamente, entonces, según las ecuaciones (4.6) y (4.7) cada AG enviará 40 individuos y recibirá 80 individuos. Finalmente, de la ecuación (4.8) la población del Coordinador será de 200 individuos.

El proceso de migración considera tres etapas de selección de individuos. La Figura N°4.17 muestra las etapas de selección y reemplazo de individuos involucradas en el proceso de migración del Algoritmo Genético Paralelo.

A continuación, se menciona cada etapa de la Figura N°4.17.

- **Primera etapa:** Permite seleccionar una fracción de individuos de la Sub-Población de cada AG, los que son enviados al Coordinador. La fracción de individuos se

escoge de acuerdo a alguno de los siguientes criterios: al azar o los mejores.

- **Segunda etapa:** El Coordinador selecciona una fracción de individuos de todos los recibidos en la etapa anterior. Los individuos son enviados a cada AG para su reemplazo. La fracción de individuos se escoge de acuerdo a alguno de los siguientes criterios: al azar o los mejores.
- **Tercera etapa:** En la Sub-Población de cada AG, se selecciona una fracción de individuos que son reemplazados por los individuos enviados desde el Coordinador en la etapa anterior. La fracción de individuos se escoge de acuerdo a alguno de los siguientes criterios: al azar o los peores

Nótese en la Figura N°4.17, la diversidad genética que se logra después que los individuos son reemplazados en cada AG, al final de la Tercera Etapa.

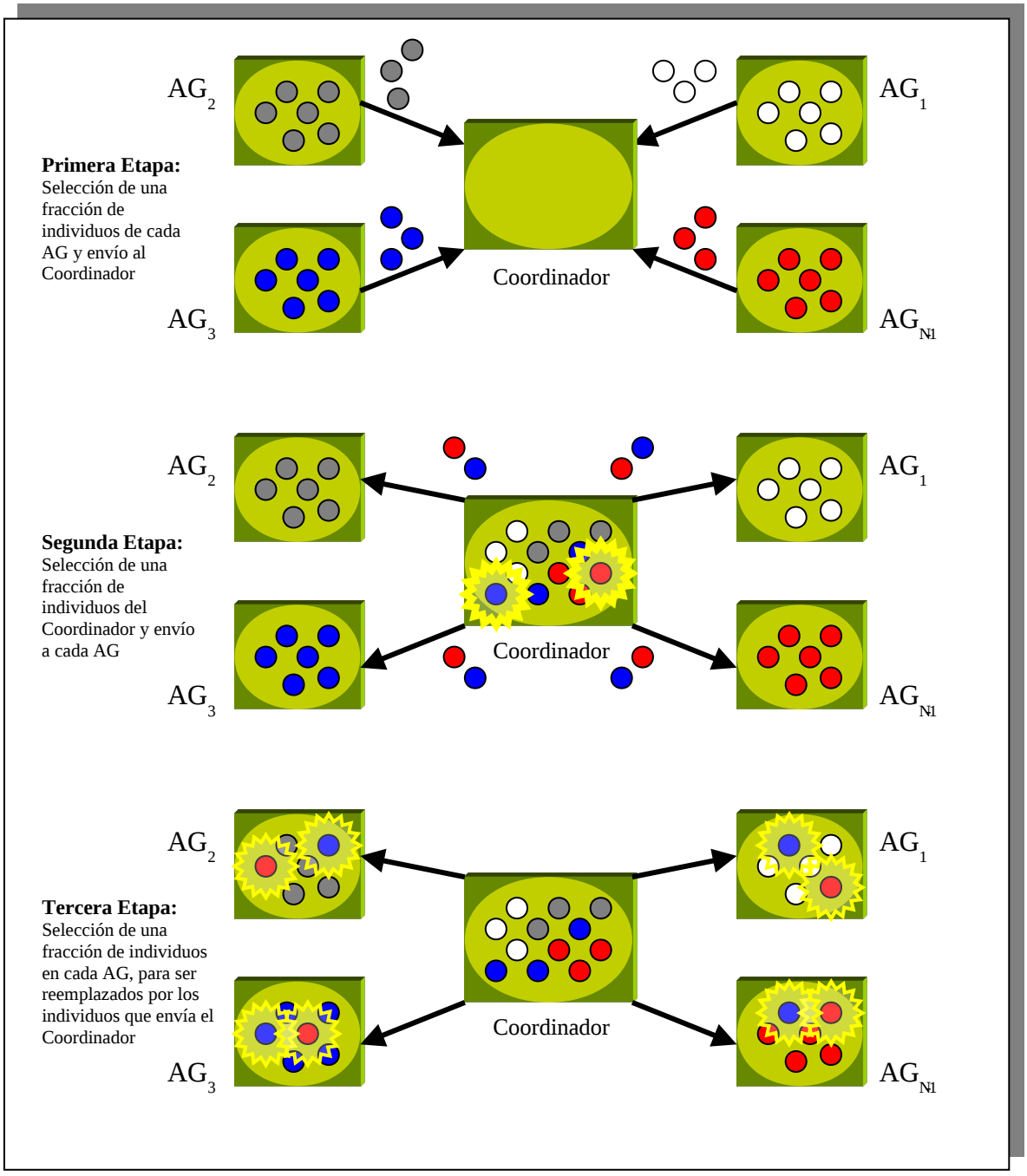


Figura N°4.17: Etapas de selección y reemplazo de individuos involucradas en el Proceso de Migración del Algoritmo Genético Paralelo.

El seudocódigo del Algoritmo Genético Paralelo se presenta a

continuación.

ALGORITMO GENÉTICO PARALELO

```
RANK = Inicializa_MPI();
IF (RANK == 0) THEN { //Rutinas a realizar por el Coordinador
        PARAMETROS = Lee_Parametros_Geneticos();
        Envia_Mensaje(“Parametros_Geneticos”, PARAMETROS_GENETICOS, Todos_Los_AG);
        DATOS_PROBLEMA = Lee_Datos_Del_Problema_A_Resolver();
        Envia_Mensaje(“Datos_Del_Problema_A_Resolver”, DATOS_PROBLEMA, Todos_Los_AG);
        Inicializa_Variables_Globales(DATOS_PROBLEMA, RANK);
        Dimensiona_Estructuras_Dinámicas(DATOS_PROBLEMA, RANK);
        Recibe_Mensaje(“Listo_Para_Comenzar”, MENSAJE, Todos_Los_AG);
        Envia_Mensaje(“Comience”, Todos_Los_AG);
        CONTADOR_AG = 0;
        CONDICIONES_DE_TERMINO_COORD = FALSE;
        SE_ALCANZA_NUMERO_GENERACIONES_PARA_MIGRAR = FALSE;
        WHILE (NOT CONDICIONES_DE_TERMINO_COORD) {
             IF (SE_ALCANZA_NUMERO_GENERACIONES_PARA_MIGRAR == FALSE) {
                 IF (PARAMETROS_GENETICOS.TIPO_MIGRACION == ASINCRONA){
                      Espera_Que_Al_Menos_Un_AG_Alcance_Numero_De_Generaciones_Para_Migrar();
                 ELSE
                      Espera_Que_Todos_Los_AG_Alcancen_Numero_De_Generaciones_Para_Migrar();
                 Envia_Mensaje(“ENVIE_INDIVIDUOS”, Todos_Los_AG);
                 SE_ALCANZA_NUMERO_GENERACIONES_PARA_MIGRAR = TRUE;
             }
             AG_QUE_ENVIA_INDIVIDUOS = Determina_Si_Algun_AG_Ha_Enviado_Individuos();
             IF (AG_QUE_ENVIA_INDIVIDUOS > 0) { //Algún AG a enviado Individuos hacia Coordinador
                 Recibe_Mensaje(“Individuos_A_Enviar”,INDIVIDUOS_AG, AG_QUE_ENVIA_INDIVIDUOS);
                 COORD_POBLACION = Almacena_Individuos_En_Coord(INDIVIDUOS_AG, AG_QUE_ENVIA_INDIVIDUOS);
                 CONTADOR_AG = CONTADOR_AG + 1;
             }
             IF (CONTADOR_AG = NUMERO_DE_AG_INVOLUCRADOS){ //Recibidos los Individuos de todos los AG’s
                 CONTADOR_AG = 0;
                 SE_ALCANZA_NUMERO_GENERACIONES_PARA_MIGRAR = FALSE;
                 Obtiene_Estadisticas_De_La_Poblacion_Del_Coordinador(COORD_POBLACION);
                 MEJOR_INDIVIDUO = Determina_Mejor_Individuo(COORD_POBLACION);
                 INDIVIDUOS = Selecciona_Individuos_A_Enviar_A_Cada_AG(COORD_POBLACION);
                 Envia_Mensaje(“Individuos_A_Recibir”, INDIVIDUOS, Todos_Los_AG);
             }
             CONDICIONES_DE_TERMINO_COORD = Actualiza_Condiciones_De_Termino_Coord();
        }
        Envia_Mensaje(“Finalice_Ejecucion”, Todos_Los_AG);
        Libera_Memoria_Estructuras_Dinamicas(RANK);
        Genera_Archivo_De_Resultados(MEJOR_INDIVIDUO);
} ELSE { //Rutinas a realizar por cada AG
        Inicializa_Semilla_Aleatoria();
        Recibe_Mensaje(“Parametros_Geneticos”, PARAMETROS_GENETICOS, Coordinador);
        Recibe_Mensaje(“Datos_Del_Problema_A_Resolver”, DATOS_PROBLEMA, Coordinador);
        Inicializa_Variables_Globales(DATOS_PROBLEMA, RANK);
        Dimensiona_Estructuras_Dinámicas(DATOS_PROBLEMA, RANK);
        OLD_POBLACION = Inicializa_Poblacion_Inicial();
        Envia_Mensaje(“Listo_Para_Comenzar”, Coordinador);
        Recibe_Mensaje(“Comience”, MENSAJE, Coordinador);
        CONDICIONES_DE_TERMINO_AG = FALSE;
        WHILE (NOT CONDICIONES_DE_TERMINO_AG) {
             j = 0;
             DO {
                 Indice_Padre1 = Selecciona_Padre(OLD_POBLACION);
                 Indice_Padre2 = Selecciona_Padre(OLD_POBLACION);
Cruza_Padres(OLD_POBLACION[Indice_Padre1], OLD_POBLACION[Indice_Padre2],
             NEW_POBLACION[j], NEW_POBLACION[j+1]);
                 Mutacion(NEW_POBLACION[j]);
                 Mutacion(NEW_POBLACION[j+1]);
                 Evalua_Individuo(NEW_POBLACION[j]);
                 Evalua_Individuo(NEW_POBLACION[j+1]);
                 j = j + 2;
             } WHILE (j < TAMAÑO_SUB_POBLACION -1);
             DEBE_MIGRAR = Examina_Si_Debe_Migrar_Individuos();
             IF (DEBE_MIGRAR == TRUE){  //Realiza Proceso de Migración
                 INDIVUDOS_A_MIGRAR = Selecciona_Individuos_A_Migrar(NEW_POBLACION);
                 Envia_Mensaje(“Individuos_A_Enviar”, INDIVIDUOS_A_MIGRAR, Coordinador);
                 Recibe_Mensaje(“Individuos_A_Recibir”, INDIVIDUOS_A_REEMPLAZAR, Coordinador);
                 Inserta_Individuos_En_Sub_Poblacion(INDIVIDUOS_A_REEMPLAZAR);
             }
             CONDICIONES_DE_TERMINO_AG = Actualiza_Condiciones_De_Termino_AG();
             OLD_POBLACION = NEW_POBLACION;
        }
        Libera_Memoria_Estructuras_Dinamicas(RANK);
        Envia_Mensaje(“AG_Ha_Finalizado_Evolucion”, Coordinador);
        Recibe_Mensaje(“Finalice_Ejecucion”, MENSAJE, Coordinador);
}
```

```
Finaliza_MPI();
```

El seudocódigo anterior presenta una visión global del Algoritmo Genético Paralelo que permite resolver el PCPGBR. Este seudocódigo es válido tanto para representar las funciones a realizar por cada AG, como por aquellas a realizar por el Coordinador. Es decir, se escribe sólo un programa y se ejecuta en múltiples procesadores (en Coordinador y en cada uno de los AG). Al ejecutar un programa en MPI, cada procesador tiene una copia del programa. Todos los procesadores comienzan a ejecutar el mismo programa, pero cada procesador ejecutará distintas sentencias del programa, por bifurcaciones introducidas basadas en el rango (RANK) del procesador (el rango identifica a cada procesador). Sólo el primer procesador (RANK = 0) ejecuta las rutinas asociadas al Coordinador, mientras que todos los demás procesadores ejecutan las rutinas asociadas a los AGs.

Capítulo 5. Resultados y Discusión

A continuación se muestran los experimentos realizados y los resultados obtenidos al evaluar el Modelo Paralelo, presentado en el Capítulo 4, que permite resolver el PCPGBR.

El Algoritmo Genético Paralelo se programa en Lenguaje C (Kernighan y Ritchie, 1991; Schildt, 2000) usando el Entorno Integrado de Desarrollo (IDE) "*Kdevelop 2.1 - C/C++*" (Kdevelop, 1998). Para realizar los experimentos numéricos se dispone de seis computadores. Cinco de ellos de iguales características: Pentium IV, cada uno con un Procesador de 1.4 GHz de velocidad, 128 Mb. de memoria RAM y Tarjeta de Red marca "*D-Link DFE-538TX 10/100 PCI Fast Ethernet Adapter*" de 100 Mbits por segundo; más un computador "*Pentium III*", con un Procesador de 700 MHz de velocidad, 128 Mb de memoria RAM y Tarjeta de Red marca "*D-Link DFE-538TX 10/100 PCI Fast Ethernet Adapter*" de 100 Mbits por segundo. Todos corren bajo el Sistema Operativo Linux (Negus, 1999; Goerzen, 2000; Wall, 2000) RedHat Versión 7.3 (RedHat, 2002), Kernel 2.4.18.

Para la realización de la aplicación paralela basada en paso de mensajes, se utiliza la implementación LAM Versión 6.5.6 (Lumsdaine,

2001) como herramienta de programación MPI.

La interconexión entre los seis computadores está constituida por una red "*Ethernet*" y un "*Switch D-Link*", Modelo "*DES-1024R+ 10/100 Fast Ethernet Switch*" de 100 Mbits por segundo.

La Figura N°5.1 muestra la configuración final utilizada.

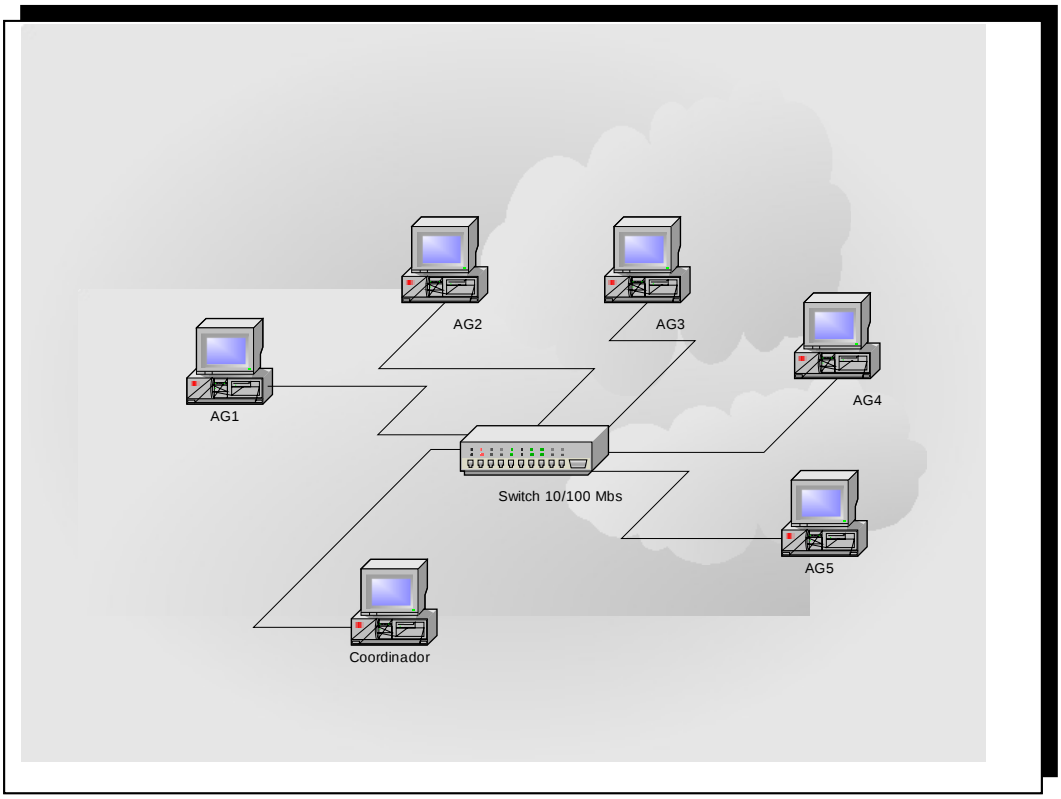


Figura N°5.1: Red de Computadores usada para implementar el Algoritmo Genético Paralelo que resuelve el PCPGBR.

En cada uno de los cinco computadores de idénticas características se corre un Algoritmo Genético (Capítulo 4), el Coordinador se ejecuta en el Computador de menores prestaciones (*Pentium III*).

Los datos de entrada necesarios para correr cualquier problema y la

información entregada por el Coordinador, como resultado del proceso evolutivo, se detallan en el Apéndice A.

Con el propósito de mostrar el proceso evolutivo que ocurre en el sistema cuando se resuelven los problemas de corte, se muestra en forma detallada el procesamiento de un problema de alta complejidad, del cual se conoce una solución óptima que consiste en un layout con pérdida cero. Este layout considera la totalidad de las piezas referenciadas en el archivo que representa la instancia del problema (ver Problema2026.txt en Apéndice C).

La instancia de prueba fue generada partiendo desde una lámina de tamaño arbitrario, a la que se le aplicaron sucesivos cortes guillotinas en forma vertical y horizontal, dando origen al conjunto de piezas de la instancia Problema2026.txt (Apéndice C).

Alimentando al modelo con una instancia generada mediante el procedimiento antes descrito, se pretende que, en el mejor de los casos, el modelo pueda generar un layout con pérdida cero, en el menor tiempo posible.

El problema de prueba se ejecuta considerando que las migraciones de individuos ocurren en instantes predefinidos, alcanzando todos los AGs la misma barrera antes de poder continuar (Migración Síncrona), o bastando que sólo un AG alcance el número de generaciones necesarias para realizar el proceso de migración (Migración Asíncrona).

Los datos de la instancia de prueba y los parámetros genéticos utilizados para resolverlo mediante el algoritmo diseñado se resumen en la Tabla N°5.1.

Los gráficos de las Figuras N°5.2, N°5.3 y N°5.4 muestran el comportamiento evolutivo de la población de soluciones para cada uno de los AGs, al considerar Migración Asíncrona con un procesador (Figura N°5.2-a), dos procesadores (Figura N°5.2-b), tres procesadores (Figura N°5.2-c), cuatro procesadores (Figura N°5.3) y cinco procesadores (Figura N°5.4). La Figura N°5.5 muestra el comportamiento registrado por el Coordinador al considerar desde uno a cinco

procesadores y Migración Asíncrona. Las Figuras N°5.6 y N°5.7 muestran los layout obtenidos en este experimento.

Tabla N°5.1: Parámetros Genéticos y datos de la instancia de prueba.

PARÁMETROS GENÉTICOS		Número de Procesadores				
		1	2	3	4	5
Tamaño de la Sub-Población de cada AG	:	5000	2500	1666	1250	1000
N° de individuos a enviar desde cada AG al Coordinador	:	500	250	166	125	100
N° de individuos a recibir en cada AG desde Coordinador	:	500	250	166	125	100
Intervalo de Migración (en generaciones)	:	25	25	25	25	25

DATOS DEL PROBLEMA DE PRUEBA

Nombre Problema	Problema2026.txt
Número Total de Piezas	225
Número Total de Piezas distintas	42
Dimensiones de la Lámina	69 x 77 $(\text{unidades})^2$

Número Total de Individuos en la Población Global	5000
Cantidad de AGs	5
Probabilidad de Cruzamiento	90 %
Probabilidad de Mutación	10 %
Número Máximo de Generaciones	1000
Modelos de Migración considerados	Asíncrona y Síncrona

Política de Selección de Individuos a enviar desde cada AG al Coordinador	Se seleccionan los mejores individuos
Política de Selección de Individuos a enviar desde el Coordinador a cada AG	Se seleccionan los mejores individuos
Política de Reemplazo de Individuos en cada AG	Se reemplazan los peores individuos

Los gráficos de las Figuras N°5.8, N°5.9 y N°5.10 muestran el comportamiento evolutivo de la población de soluciones para cada uno de los AGs, al considerar Migración Síncrona con un procesador (Figura N°5.8-a), dos procesadores (Figura N°5.8-b), tres procesadores (Figura N°5.8-c), cuatro procesadores (Figura N°5.9) y cinco procesadores (Figura N°5.10). La Figura N°5.11 muestra el comportamiento registrado por el Coordinador al considerar desde uno a cinco

procesadores y Migración Síncrona. Las Figuras N°5.12 y N°5.13 muestran los layout obtenidos en este experimento.

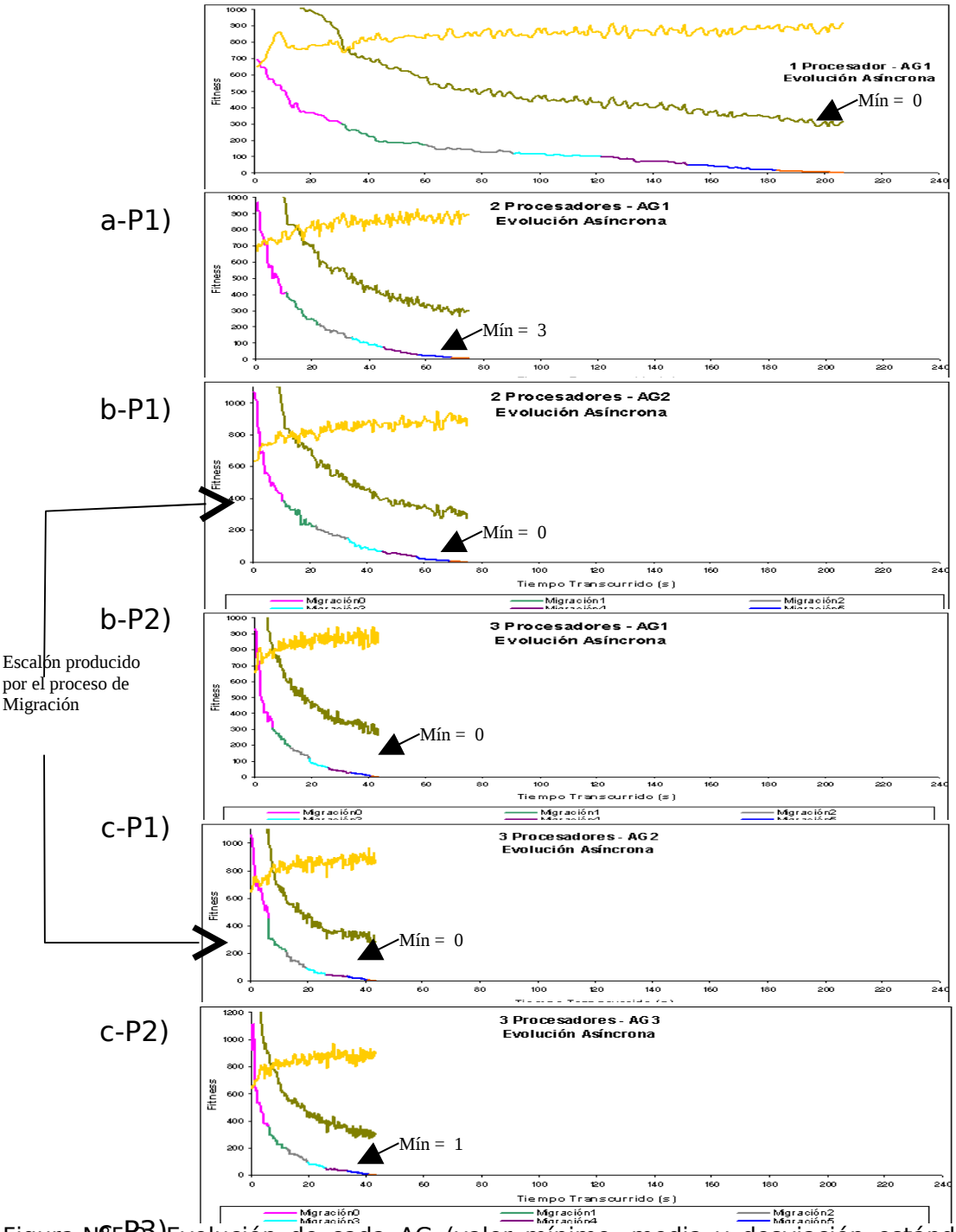


Figura N° 5.2 Evolución de cada AG (valor mínimo, media y desviación estándar) para problema de prueba, considerando (a) uno, (b) dos y (c) tres Procesadores con Migración Asíncrona.

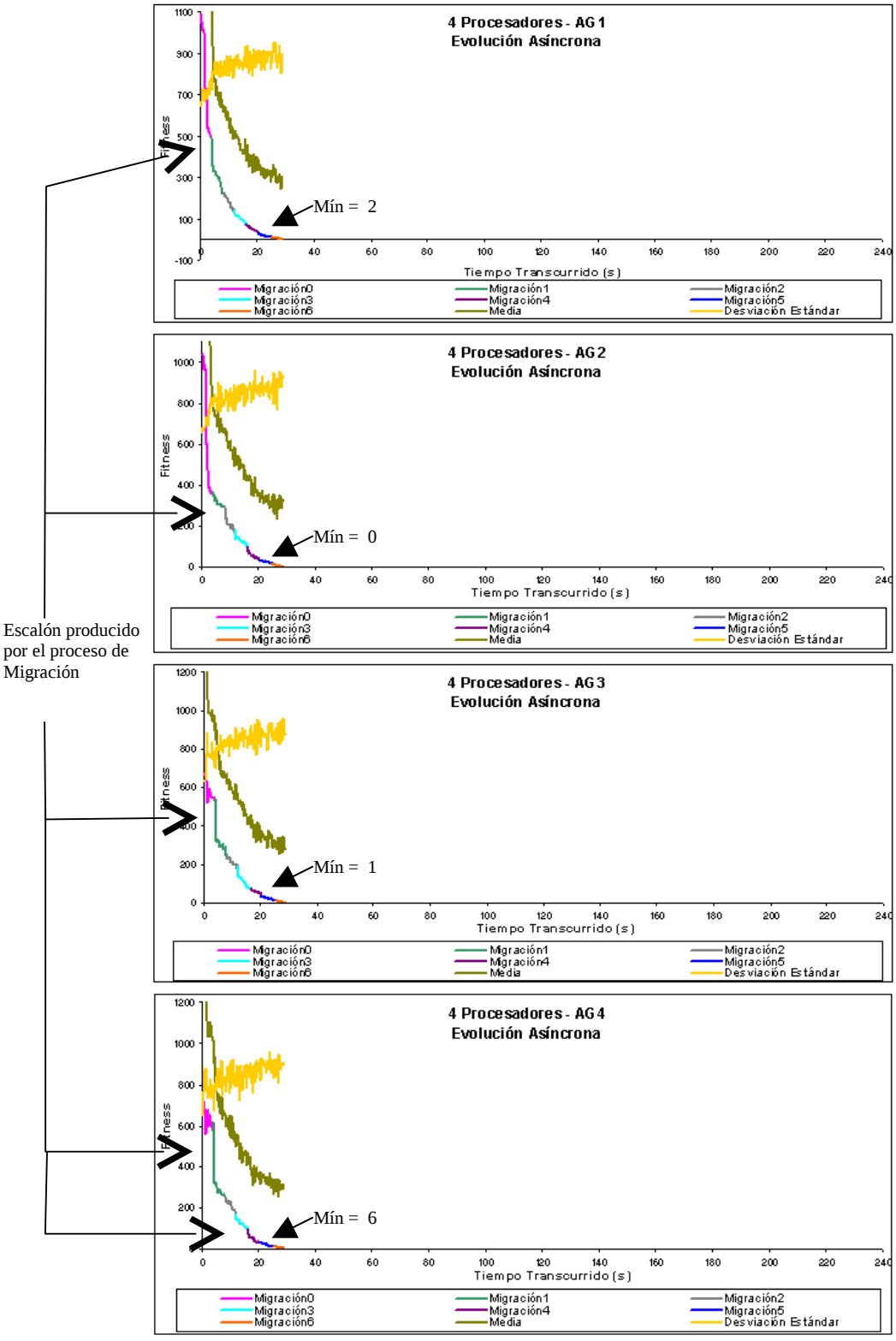


Figura N°5.3: Evolución de cada AG (valor mínimo, media y desviación estándar) para problema de prueba, considerando cuatro Procesadores con Migración Asíncrona.

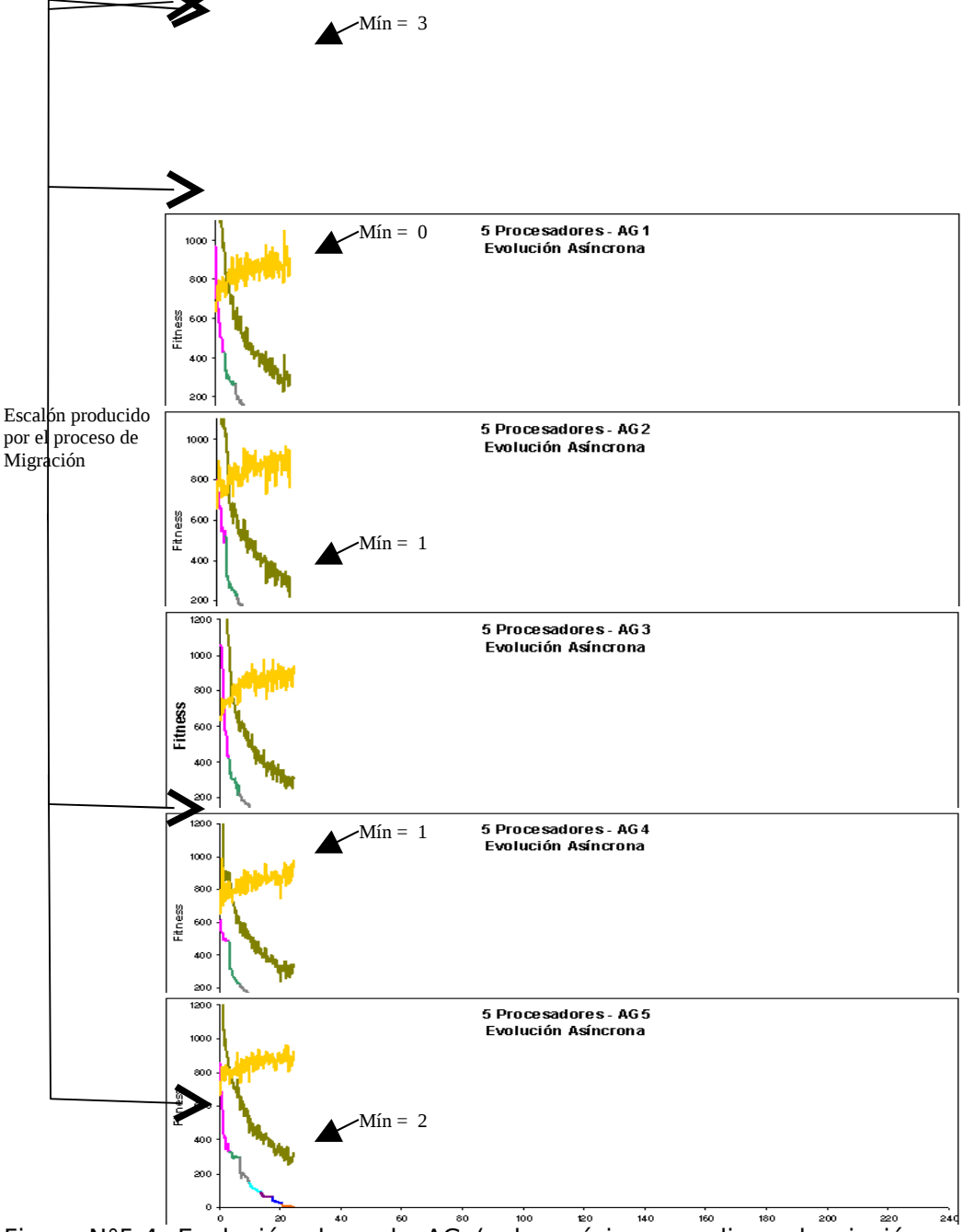
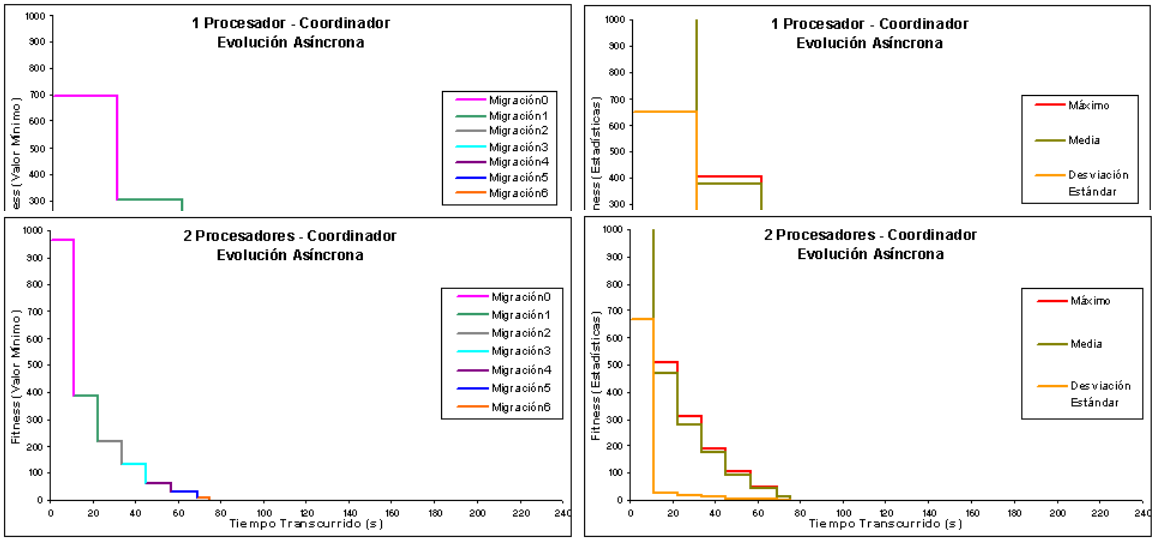


Figura N°5.4: Evolución de cada AG (valor mínimo, media y desviación estándar) para problema de prueba, considerando cinco Procesadores con Migración Asíncrona.



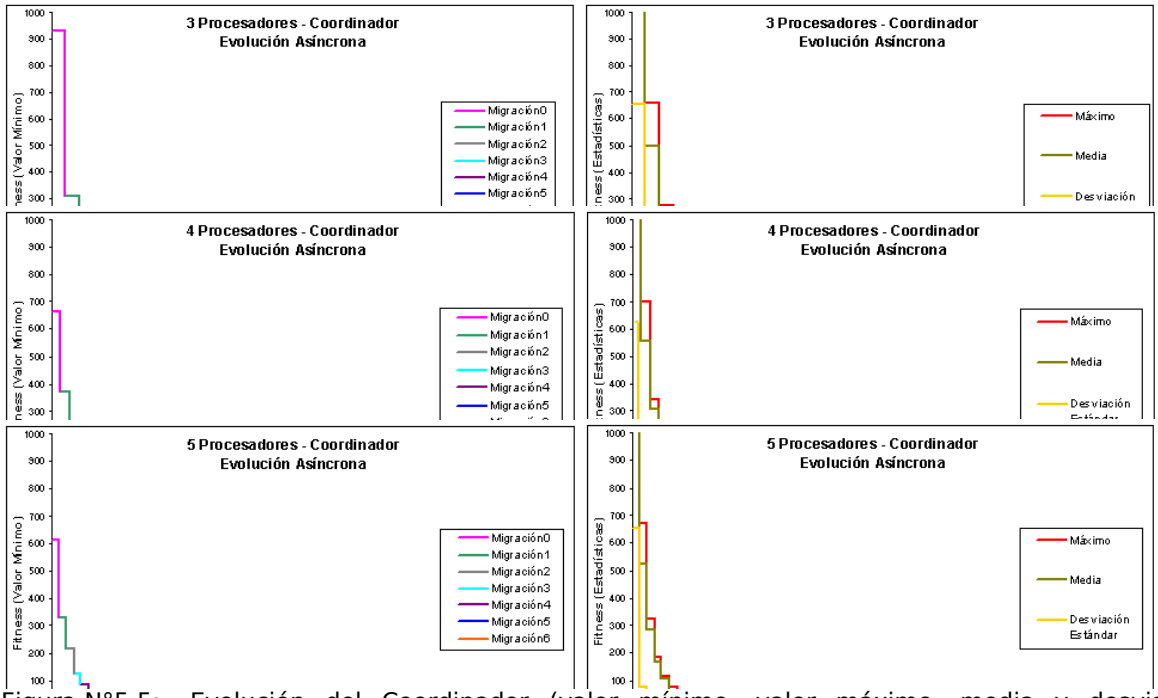
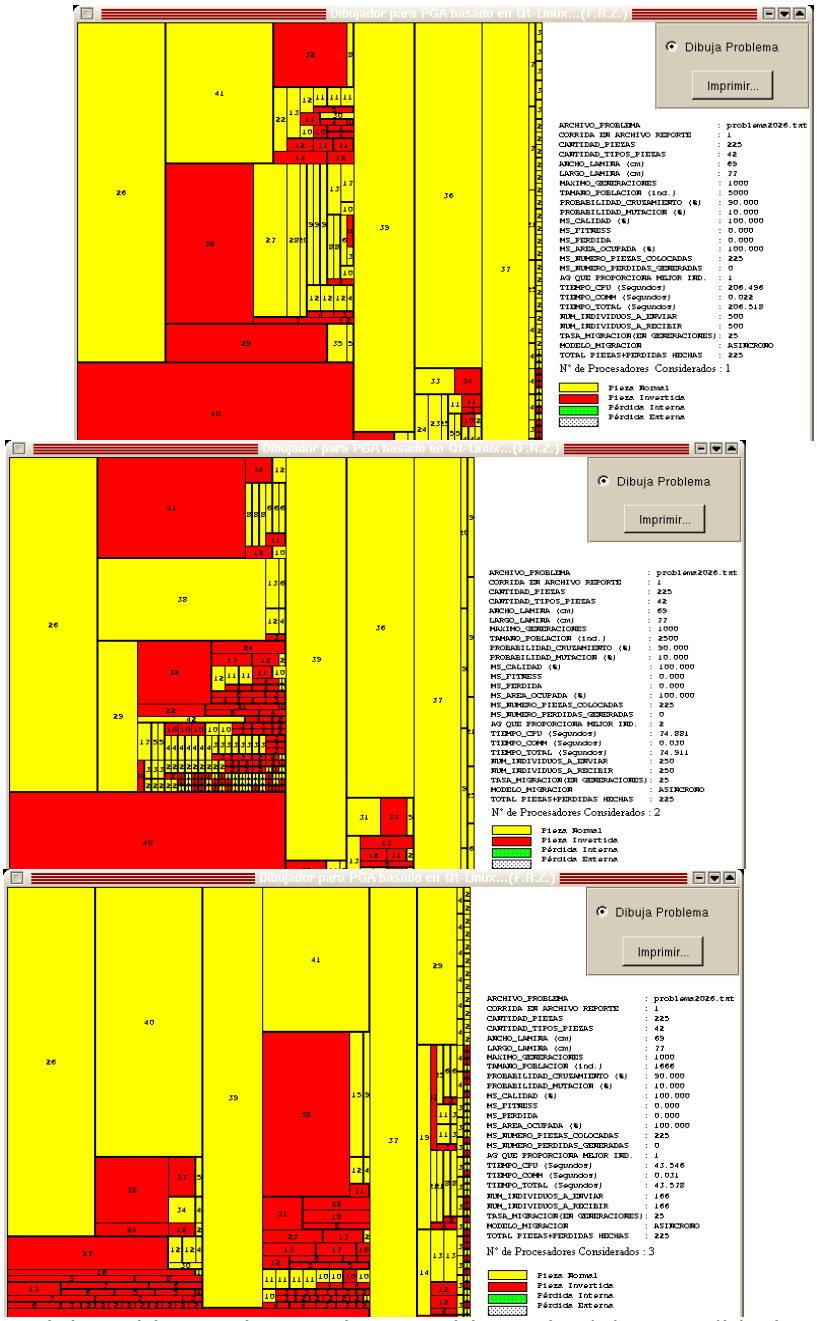


Figura N°5.5: Evolución del Coordinador (valor mínimo, valor máximo, media y desviación estándar) para problema de prueba, considerando Migración Asíncrona.

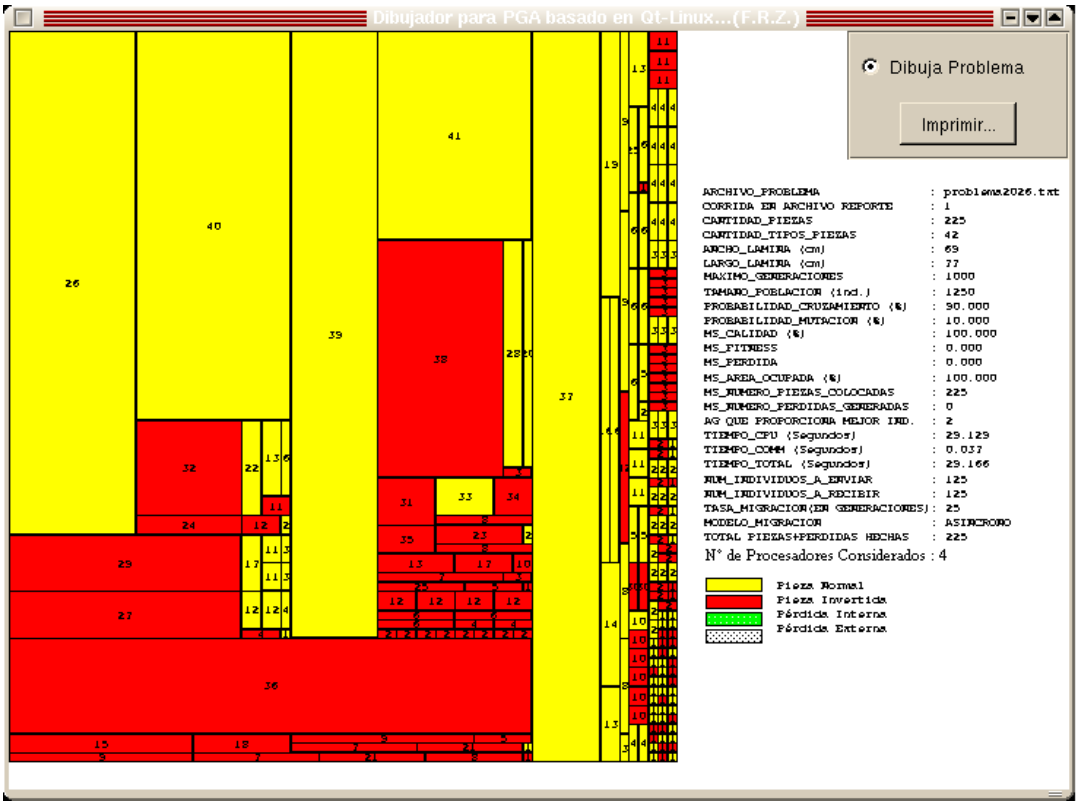


b)

Figura N°5.6: Layout del problema de prueba considerando (a) uno, (b) dos y (c) tres procesadores con migración asíncrona.

c)

a)



b)

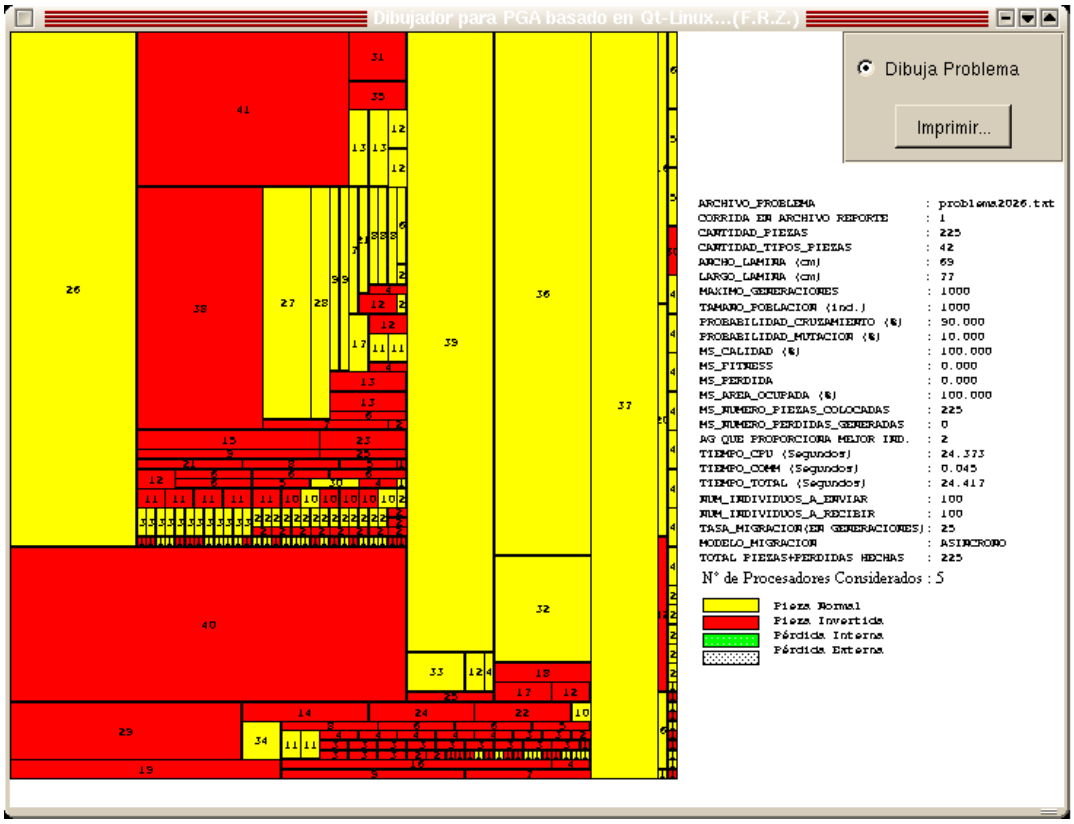


Figura N°5.7: Layout del Problema de prueba al considerar (a) cuatro y (b) cinco procesadores con migración asíncrona.

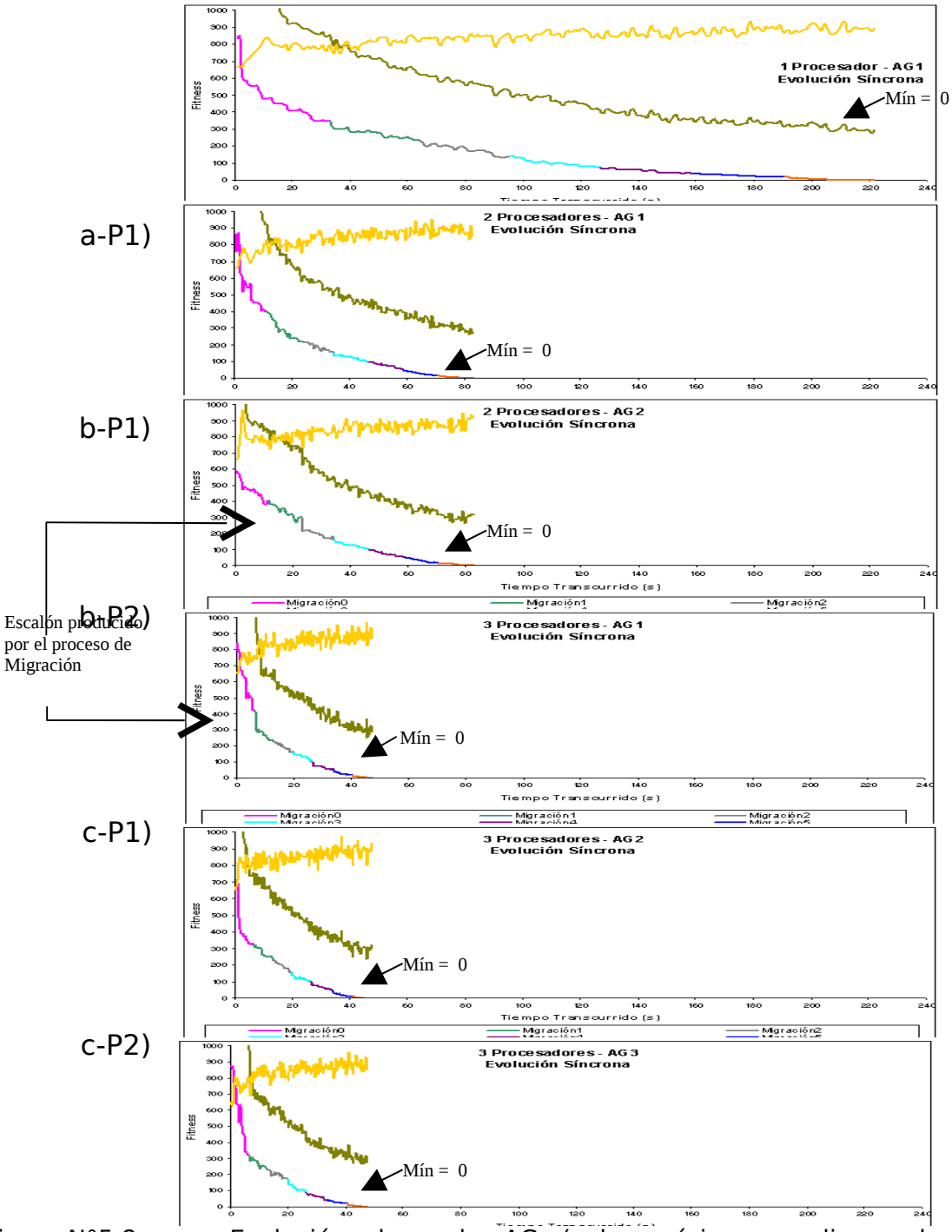


Figura N°5.8: Evolución de cada AG (valor mínimo, media y desviación estándar) para problema de prueba, considerando (a) uno, (b) dos y (c) tres Procesadores con Migración Síncrona.

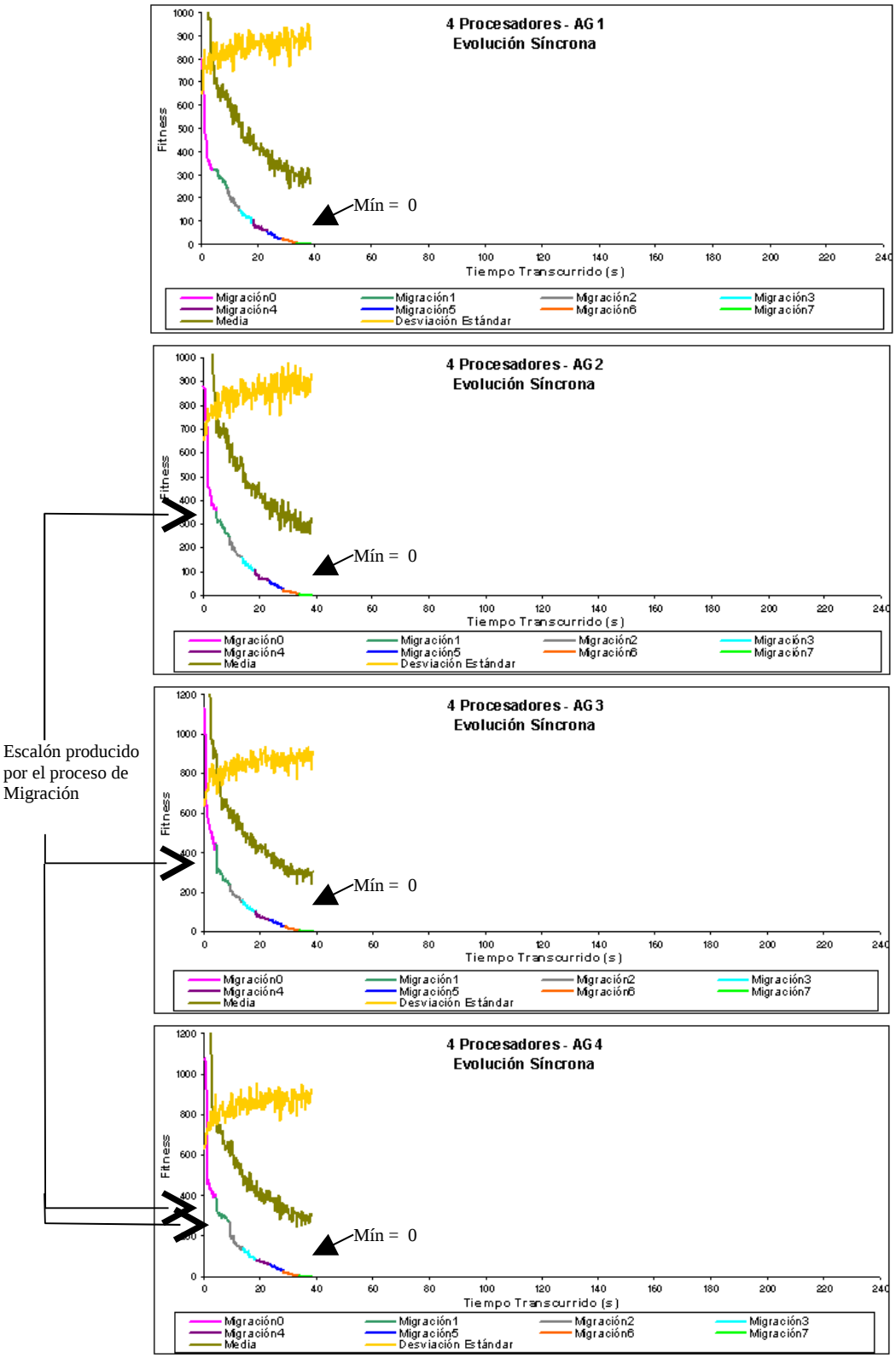
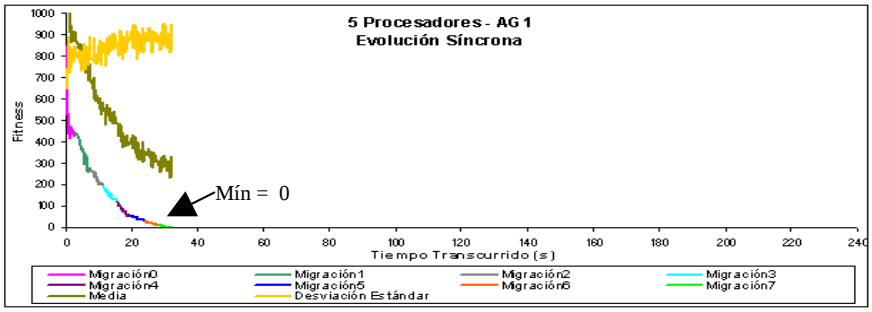


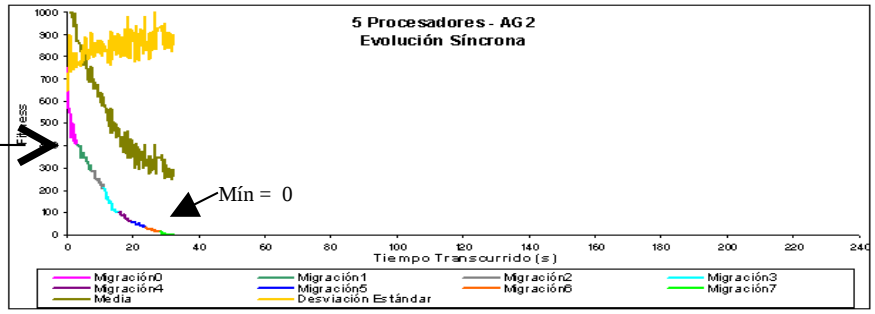
Figura N°5.9: Evolución de cada AG (valor mínimo, media y desviación estándar) para problema de prueba, considerando cuatro

Procesadores con Migración Síncrona.



Mín = 0

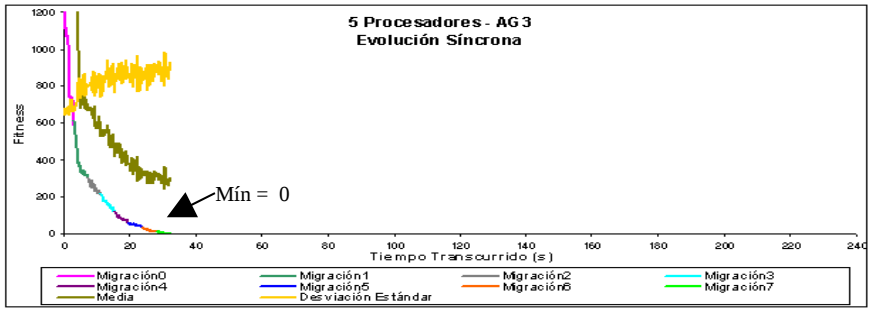
Mín = 0

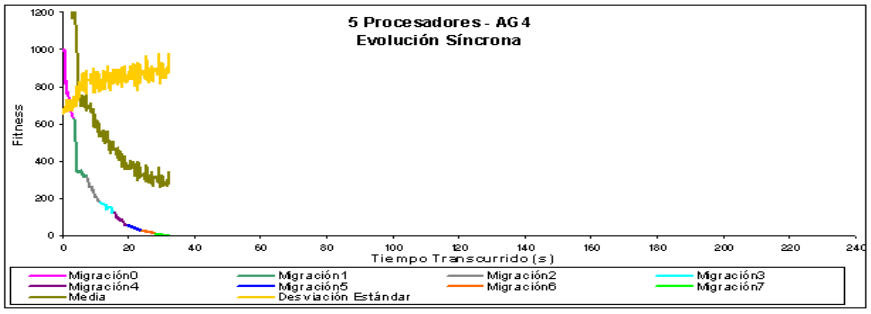


Escalón producido por el proceso de Migración

Mín = 0

Mín = 0





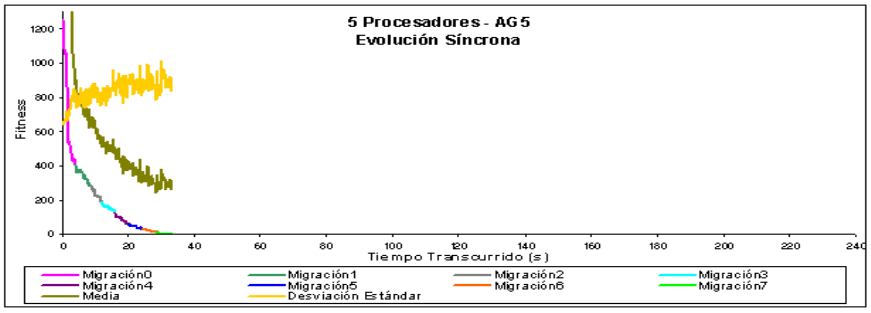
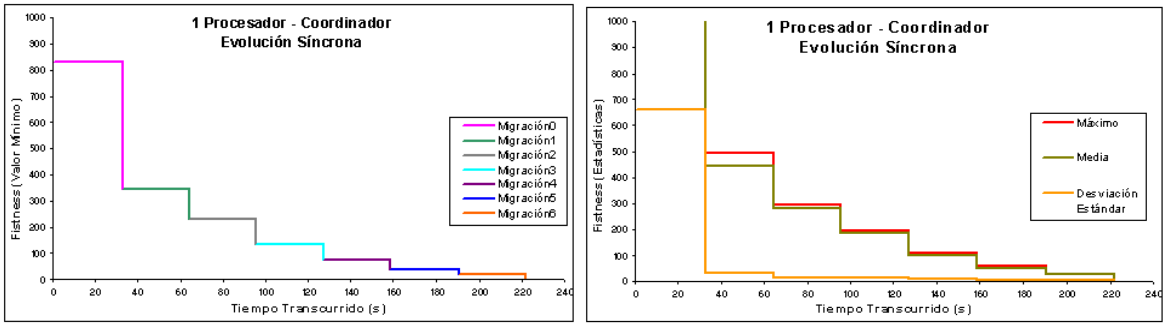


Figura N°5.10: Evolución de cada AG (valor mínimo, media y desviación estándar) para problema de prueba, considerando cinco Procesadores con Migración Síncrona.



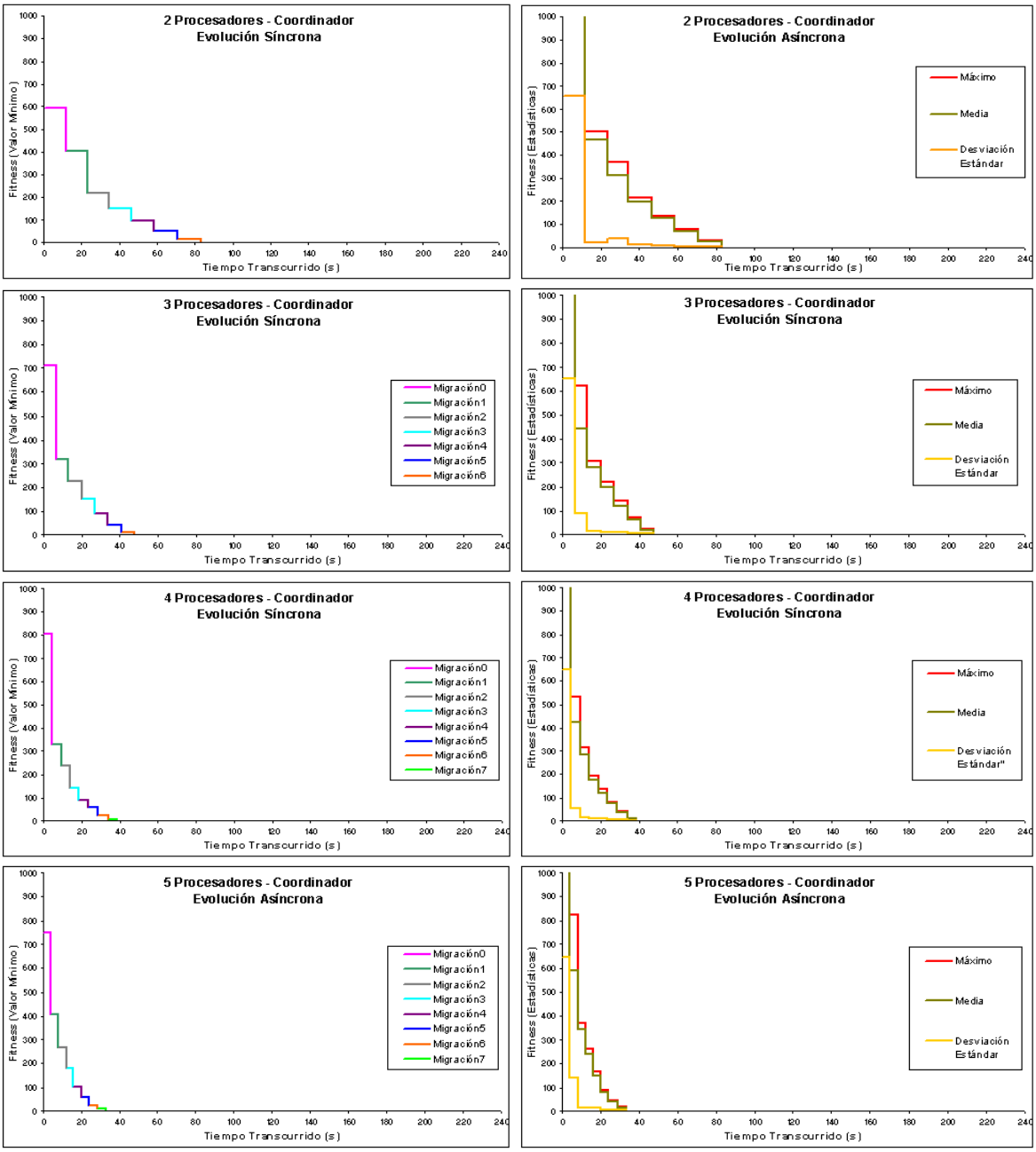
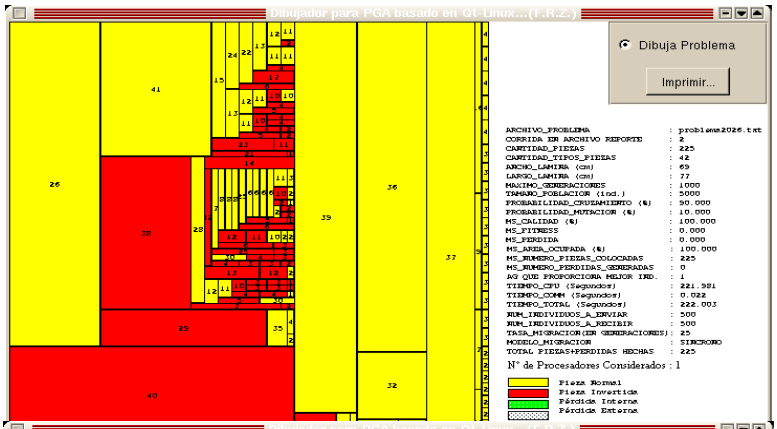
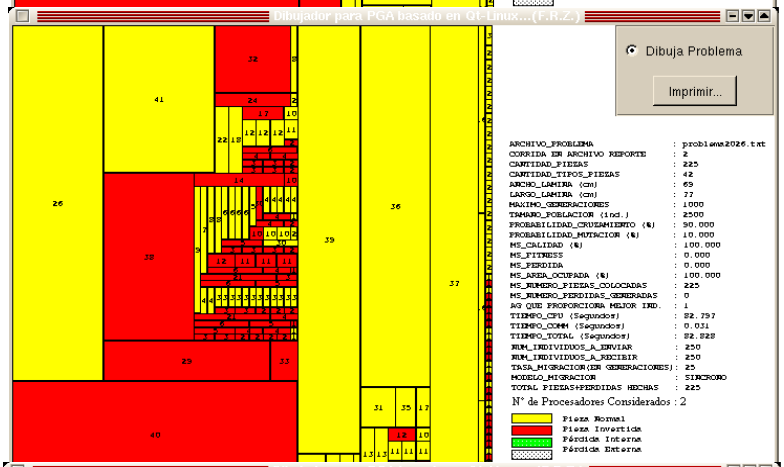


Figura N°5.11: Evolución del Coordinador (valor mínimo, valor máximo, media y desviación estándar) para problema de prueba, considerando Migración Síncrona.

a)



b)



c)

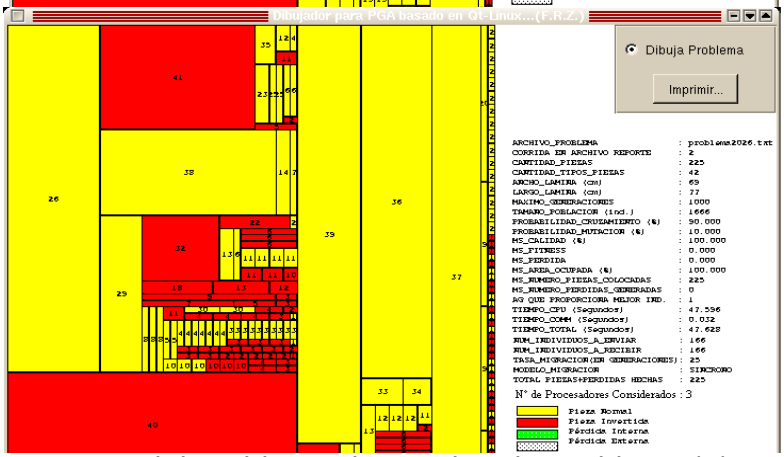
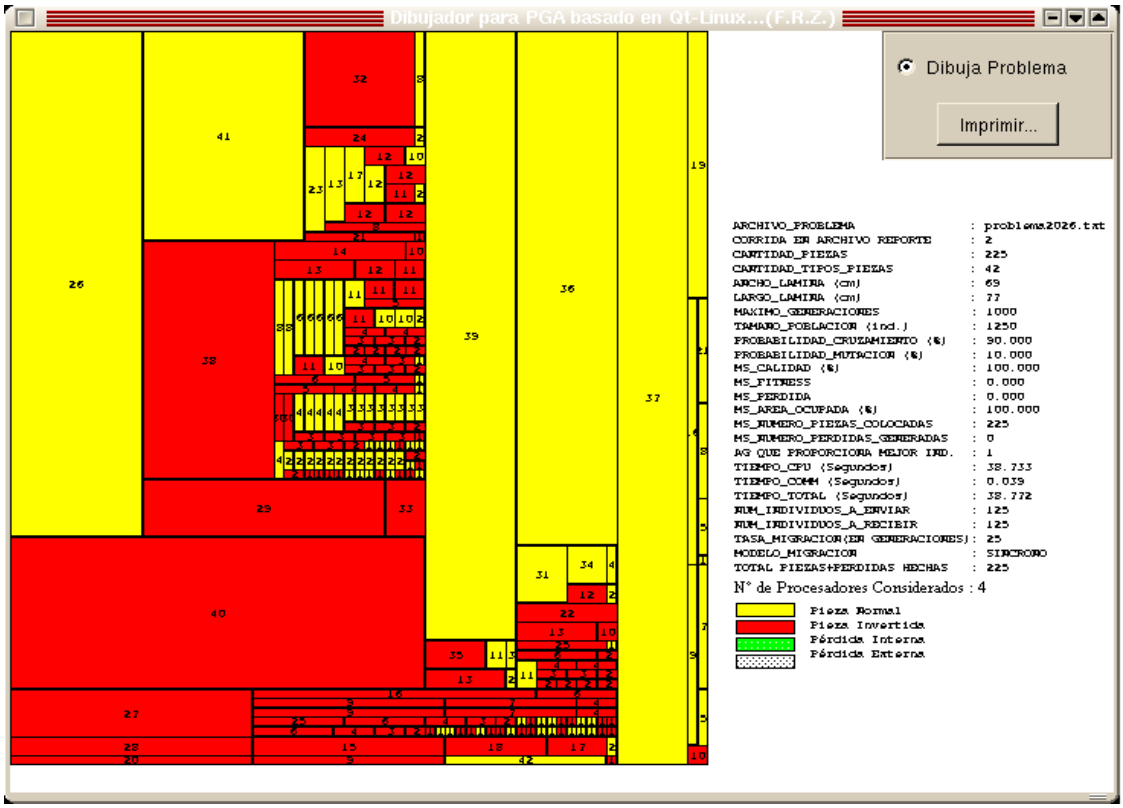
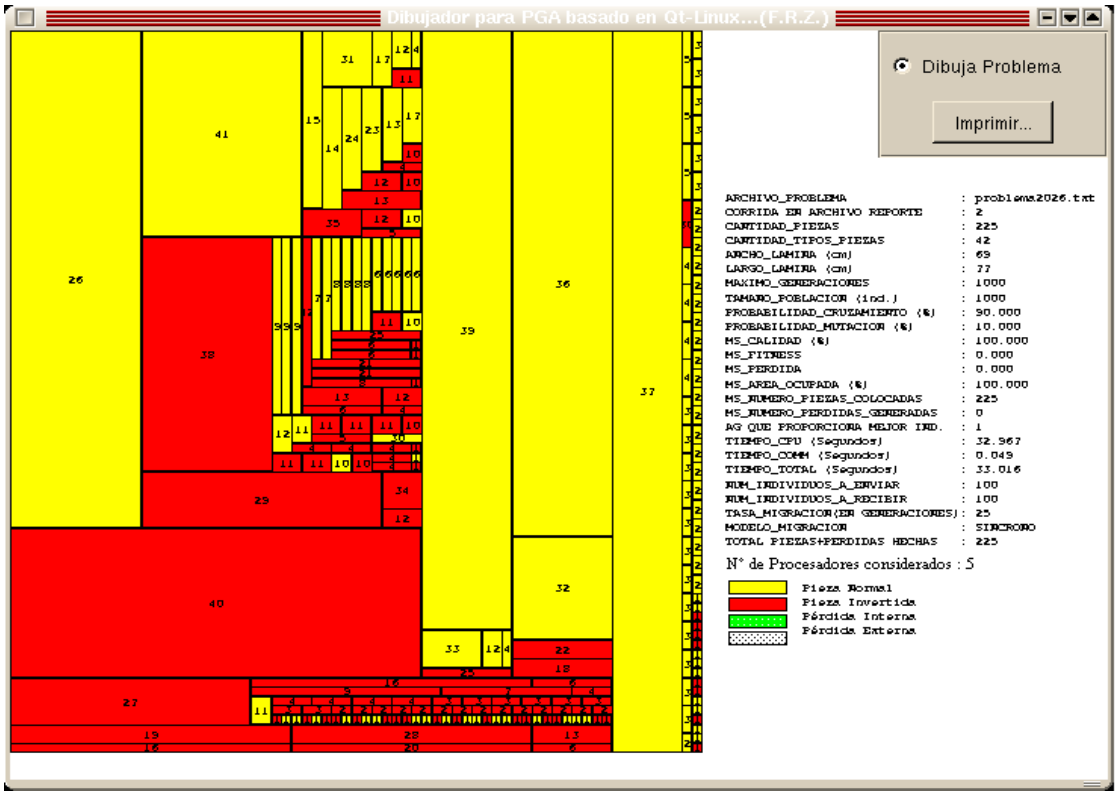


Figura N°5.12: Layout del Problema de prueba al considerar (a) uno, (b) dos, (c) tres procesadores con migración síncrona.



a)



b)

Figura N°5.13: Layout del Problema de prueba al considerar (a) cuatro, (b) cinco procesadores con migración síncrona.

Las Figuras N°5.2, N°5.3, N°5.4, N°5.8, N°5.9 y N°5.10 están conformadas por gráficos que contienen tres líneas. La línea verde representa el valor medio del "*fitness*" (pérdida) de los individuos de la sub-población en cada instante. La línea amarilla corresponde a la desviación estándar de la sub-población en cada instante. La línea multicolor corresponde al valor del "mejor individuo" de la sub-población (menor pérdida) en cada instante. Los cambios de color de esta línea se producen cuando se registran eventos de migración, es decir, un color determinado se mantiene hasta que se produce el siguiente evento de migración.

Con el fin de poder apreciar el efecto de la reducción en el tiempo, es que el eje horizontal de estas figuras corresponde al tiempo transcurrido (en segundos) y no al número de generaciones efectuadas.

En estas figuras se observa cómo el valor mínimo (de menor pérdida) de la sub-población disminuye hasta cero, encontrando en ese punto, la solución óptima

En estas figuras se señalan algunas disminuciones bruscas del valor mínimo (de menor pérdida) producidas por el proceso de migración de individuos.

Lo anterior se produce ya que algunas sub-poblaciones reciben individuos de otras sub-poblaciones con pérdidas menores a las que hasta en ese instante tenía la sub-población que recibe. Esta recepción de material genético beneficia al individuo con menor pérdida en la sub-

población, tal como se muestra, por ejemplo, en la sub-población AG2 de la Figura N°5.4.

Nótese que en la Figura N°5.2.b (caso 2 Procesadores con Migración Asíncrona) la sub-población AG2 es la que encuentra la solución óptima, mientras que la otra sub-población (AG1), en ese mismo instante, aún no ha encontrado el óptimo (Valor Mín= 3). Sin embargo, AG1 está muy cercano en encontrar la solución óptima.

En las Figuras N°5.2, N°5.3, N°5.4, N°5.8, N°5.9 y N°5.10 el valor medio de la población tiende a la baja. Esta baja la producen los individuos que son mejorados por el proceso evolutivo. La desviación estándar tiende a ser constante, lo que sugiere que hay un grupo de individuos que no mejora con el proceso evolutivo. Esto es consistente con la política de selección que dirige el proceso evolutivo de los mejores individuos (selección por ruleta, capítulo 4.2.2.1), marginando a los peores individuos de cada sub-población.

Por otro lado, las Figuras N°5.5 y 5.11 muestran la evolución de los individuos que recibe el Coordinador para el caso de uno a cinco Procesadores con Migración Asíncrona y Síncrona, respectivamente. En el gráfico de la izquierda de estas figuras, se muestra una línea multicolor que corresponde al valor del “mejor individuo” (menor pérdida) que posee el Coordinador en cada instante. Los cambios de color de esta línea se producen cuando se registran eventos de migración, es decir, un color determinado se mantiene hasta que se produce el siguiente evento de

migración. En el gráfico de la derecha de estas figuras, se muestran 3 líneas. La línea verde representa el valor medio del "*fitness*" (pérdida) de los individuos que recibe el Coordinador en cada instante. La línea amarilla corresponde a la desviación estándar de los individuos enviados desde los AGs hacia el Coordinador en cada instante.

La línea de color verde corresponde al valor del "peor individuo" (mayor pérdida) que recibe el Coordinador. Las líneas tipo "escalón" que aparecen en estas figuras, se deben a que los individuos en el Coordinador no varían hasta que se cumpla el siguiente evento de migración.

En las Figuras N°5.5 y 5.11 se observa que el Coordinador cada vez recibe "mejores individuos" (de menor pérdida), en la medida que se producen los eventos de migración. Esto lo confirma el hecho que tanto la media como el valor máximo de los individuos disminuye. Además, dado que la desviación estándar cada vez tiende más a cero, se puede afirmar que los individuos recibidos son cada vez más similares.

En las Figuras N°5.6, N°5.7 y N°5.12, N°5.13 se muestran los "*layout*" resultantes al resolver el problema de prueba desde uno a cinco Procesadores con Migración Asíncrona y Síncrona, respectivamente. Se observan piezas de color amarillo, lo que indica que dichas piezas han sido consideradas sin rotación en el patrón de corte. Una pieza con color rojo indica que ha sido incluida rotada en el layout final. Nótese que los layout obtenidos en todos los casos son distintos, lo que sugiere que el problema de prueba no posee una solución óptima absoluta, sino que posee varias

distribuciones de piezas que permiten encontrar pérdida cero. Sin embargo, se observa que en todos los casos el Algoritmo Genético desarrollado intenta agrupar las piezas de similares dimensiones, así como rellenar las pérdidas internas que pudieran producirse, dando como resultado un layout de gran complejidad, como los obtenidos en estos experimentos. Nótese, además, el grado de combinatoriedad que posee el problema de prueba, lo cual se puede apreciar en cada uno de los layout obtenidos.

En la Figura N°5.14 se despliega el "*Speedup*" alcanzado para el problema de prueba, considerando Migración Asíncrona y Síncrona.

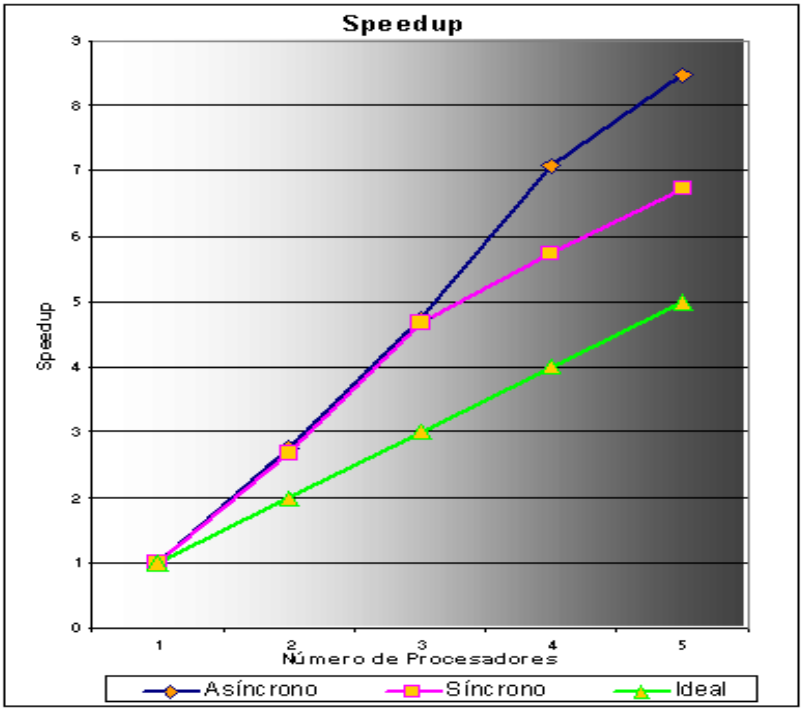


Figura N°5.14: "*Speedup*" alcanzado por el Algoritmo Genético Paralelo considerando Migración Asíncrona y Síncrona

El cálculo del "*Speedup*" de la Figura N°5.14 se realiza, tanto para

Migración Asíncrona como Síncrona, considerando la razón entre el tiempo secuencial (tiempo necesario para lograr pérdida cero al utilizar un procesador) versus el tiempo paralelo (tiempo necesario para lograr pérdida cero al utilizar dos, tres, cuatro y cinco procesadores).

En la Figura N°5.14 se muestra que el "*Speedup*" alcanzado con el Algoritmo Genético Paralelo diseñado es Superlineal[4], tanto para Migración Asíncrona como Síncrona.

Además, queda de manifiesto la superioridad, en cuanto al "*Speedup*", de resolver este problema considerando Migración Asíncrona frente a resolverlo con Migración Síncrona.

Según Wilkinson y Allen (1999) y Cantú-Paz (2000), el "*Speedup*" Superlineal puede ocurrir en algoritmos de búsqueda como los Algoritmos Genéticos Paralelos. Sin embargo, éste ha sido un tema controversial por muchos años en la comunidad relacionada con los Algoritmos Genéticos, del cual no se ha podido llegar a un consenso.

Algunos trabajos (Koza y Andre, 1995; Lin, Punch, y Goodman, 1994; Tongchim y Chongstitvatana, 2000) relacionados con la paralelización de los Algoritmos Genéticos, reportan "*Speedup*" Superlineal al usar un modelo paralelo de grano grueso. Tongchim y Chongstitvatana (2000) obtienen "*Speedup*" muy similares a los encontrados en este trabajo, argumentando que el "*Speedup*" Superlineal alcanzado es debido a la distribución de la población en diferentes procesadores incrementando la

[4] Speedup Superlineal significa que el Speedup es mayor que el número de procesadores usado.

probabilidad de encontrar una solución óptima, en la medida que aumenta el número de sub-poblaciones.

Otro elemento que contribuye a encontrar "*Speedup*" Superlineales puede ser la elección de una población de tamaño superior al mínimo necesario, para obtener soluciones con una calidad prefijada, al realizar las mediciones de tiempo con el algoritmo secuencial. Esto conlleva a que el algoritmo secuencial tiene que evaluar más individuos que los estrictamente necesarios para encontrar el nivel de calidad deseado de las soluciones. Esto se relaciona directamente con la definición de "*Speedup*", donde el tiempo paralelo se compara con el mejor algoritmo secuencial conocido, el cual no se logra si no es posible establecer el tamaño correcto de la población.

Considerando que sistemáticamente se ha reportado "*Speedup*" Superlineales en la literatura, y que en los experimentos se buscó exhaustivamente los tamaños mínimos de las poblaciones necesarias para encontrar la calidad deseada de las soluciones, se está en condiciones de constatar el comportamiento *Superlineal* de la implementación del modelo paralelo propuesto para la resolución del PCPGBR.

Debido a la componente aleatoria de los Algoritmos Genéticos, para medir la eficiencia del algoritmo paralelo, se requiere variar el número de procesadores y los resultados promediarlos por cada número de procesadores. Por lo tanto, es necesario realizar varias corridas de cada problema y luego obtener un valor promedio para el "*Speedup*". Pero,

¿cuántas corridas realizar? Tongchim y Chongstitvatana (2000) proponen promediar los resultados sobre veinte corridas. En el Apéndice B, se ha establecido experimentalmente como treinta el número de corridas necesarias a realizar en cada problema.

En el Apéndice C, se muestran los archivos de piezas que representan a los problemas utilizados para realizar las pruebas del AGP. Se considera variar la cantidad de procesadores desde uno a cinco, realizando treinta corridas para los problemas de pruebas en cada caso, y luego obtener los valores promedios del "*Speedup*". Los parámetros genéticos de los problemas2025.txt y problema2026.txt se muestran en la Tabla N°5.2.

Tabla N°5.2: Parámetros Genéticos y Datos del Problema2025.txt y Problema2026.txt.

PARÁMETROS GENÉTICOS		Número de Procesadores				
		1	2	3	4	5
Tamaño de la Sub-Población de cada AG	:	4000	2000	1334	1000	800
N° de individuos a enviar desde cada AG al Coordinador	:	400	200	133	100	80
N° de individuos a recibir en cada AG desde Coordinador	:	400	200	133	100	80
Intervalo de Migración (en generaciones)	:	200	100	66	50	40

DATOS DE LOS PROBLEMAS DE PRUEBA	
Nombre Problema	Problema2025.txt
Número Total de Piezas	109
Número Total de Piezas distintas	57
Dimensiones de la Lámina	59 x 57 $(unidades)^2$

Nombre Problema	Problema2026.txt
Número Total de Piezas	225
Número Total de Piezas distintas	42
Dimensiones de la Lámina	69 x 77 $(unidades)^2$

Número Total de Individuos en la Población Global	4000
Cantidad de AGs	5
Probabilidad de Cruzamiento	90 %
Probabilidad de Mutación	10 %
Número Máximo de Generaciones	200
Modelos de Migración considerados	Asíncrona y Síncrona

Política de Selección de Individuos a enviar desde cada AG al Coordinador	Se seleccionan los mejores individuos
Política de Selección de Individuos a enviar desde el Coordinador a cada AG	Se seleccionan los mejores individuos
Política de Reemplazo de Individuos en cada AG	Se reemplazan los peores individuos

Los tiempos promedios transcurridos en segundos, el porcentaje del área ocupada y el Speedup para el Problema2025.txt y el Problema2026.txt, considerando tanto Migración Asíncrona y Síncrona desde uno a cinco procesadores, se muestran en la Figura N°5.15.

De la Figura N°5.15 se puede observar que los tiempos transcurridos al considerar implementaciones asíncronas siempre son menores a las implementaciones síncronas para dos o más procesadores. Esto se aprecia también en el gráfico del "*Speedup*", para ambos problemas, en donde se manifiesta la Superlinealidad del Algoritmo Genético Paralelo. También, en estos casos, la Migración Asíncrona es superior a la Migración Síncrona.

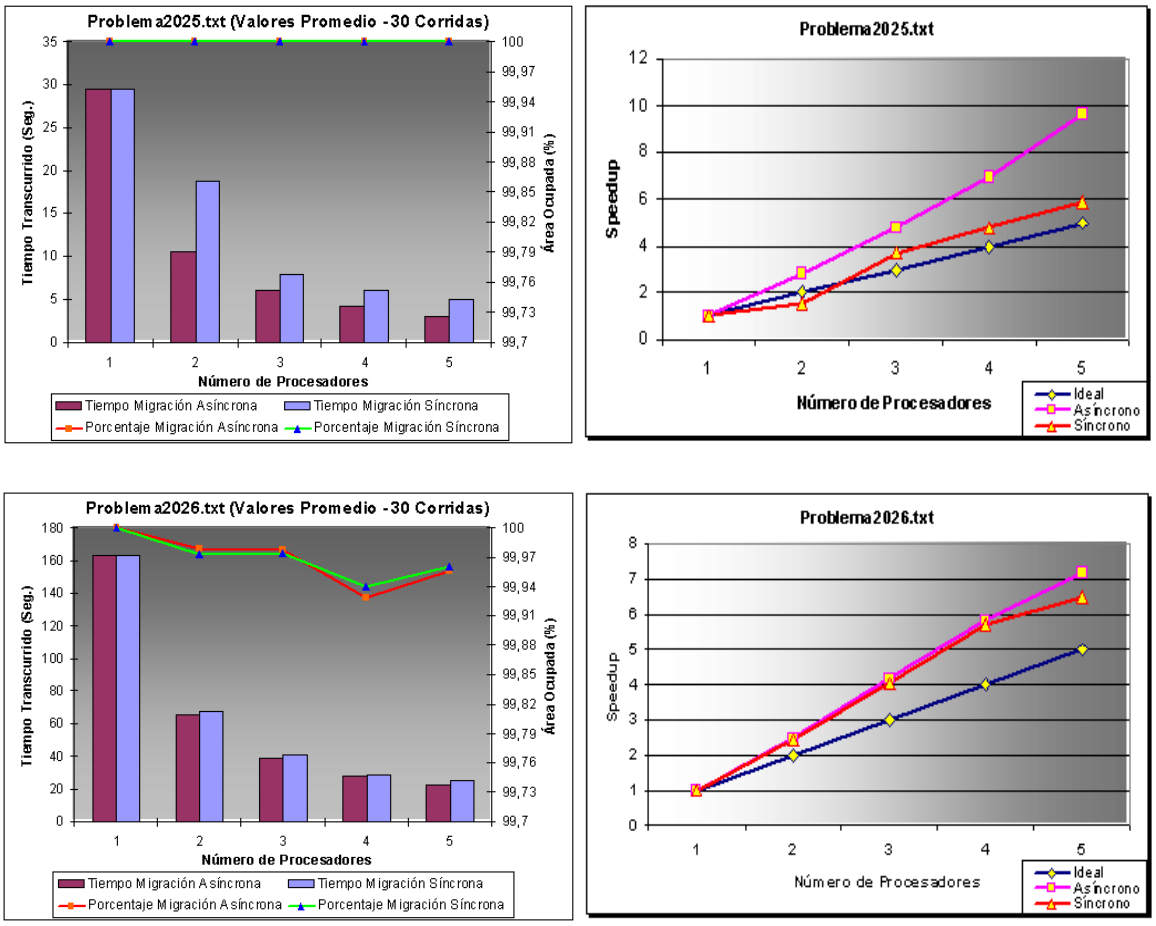


Figura N°5.15: Tiempos transcurridos, porcentaje del área ocupada y "*Speedup*" para treinta corridas considerando el Problema2025.txt y el Problema2026.txt, con Migración Asíncrona y Síncrona.

Con respecto al área ocupada, en el caso del problema2025.txt, al efectuar 30 corridas considerando desde uno a cinco Procesadores, en el 100% de las veces se obtuvieron soluciones óptimas (pérdida cero). Sin embargo, para el problema2026.txt y al considerar un Procesador, en el 100% de las corridas fue posible encontrar el óptimo. Para el caso de dos Procesadores y Migración Asíncrona, en el 66,6% de las corridas se encontró el óptimo (abarcando el 100% del área ocupada). Sin embargo, en el 33,3% de las corridas no se encontró el óptimo, logrando como promedio, ocupar el 99,98% del área. En el caso de Migración Síncrona, en el 50% de las corridas se encontró el óptimo. Sin embargo, en el 50% restante no se encontró el óptimo, logrando como promedio, ocupar el 99,97% del área.

Al considerar tres Procesadores y Migración Asíncrona, en el 73,3% de las corridas se encontró el óptimo. Pero, en el 26,6% de las corridas no se encontró el óptimo, logrando como promedio ocupar el 99,97% del área. En el caso de Migración Síncrona, en el 46,6% de las corridas se encontró el óptimo. Sin embargo, en el 53,4% restante no se encontró el óptimo, logrando como promedio ocupar el 99,97% del área.

Con cuatro Procesadores y Migración Asíncrona, en el 46,6% de las corridas se encontró el óptimo. Pero, en el 53,4% de las corridas no se encontró el óptimo, logrando como promedio ocupar el 99,93% del área. En el caso de Migración Síncrona, en el 36,6% de las corridas se encontró el óptimo. Sin embargo, en el 63,4% restante no se encontró el óptimo,

logrando como promedio ocupar el 99,94% del área.

Para el caso de cinco Procesadores y Migración Asíncrona, en el 76,6% de las corridas se encontró el óptimo (abarcando el 100% del área ocupada). Sin embargo, en el 23,4% de las corridas no se encontró el óptimo, logrando como promedio ocupar el 99,95% del área. En el caso de Migración Síncrona, en el 43,3% de las corridas se encontró el óptimo. Sin embargo, en el 56,7% restante no se encontró el óptimo, logrando como promedio ocupar el 99,96% del área.

En el Apéndice D, se muestran algunos layout obtenidos con el AGP, al considerar desde uno a cinco Procesadores, Migración Asíncrona y Síncrona para el problema2025.txt (ver Figuras N°A4-26 a N°A4-30) y problema2026.txt (ver Figuras N°A4-31 a N°A4-35).

Con el fin de obtener mayores resultados, se han considerado otros tipos de problemas de distintas complejidades (distintas piezas y dimensiones de la lámina), los cuales se detallan a continuación.

Los parámetros genéticos de los problemas1003.txt, problema1005.txt, problema1006.txt, problema1007.txt y problema1008.txt se muestran en la Tabla N°5.3.

Tabla N°5.3: Parámetros Genéticos y Datos de los Problema1003.txt, Problema1005.txt, Problema1006.txt, Problema1007.txt y Problema1008.txt.

PARÁMETROS GENÉTICOS		Número de Procesadores				
		1	2	3	4	5
Tamaño de la Sub-Población de cada AG	:	4000	2000	1334	1000	800
N° de individuos a enviar desde cada AG al Coordinador	:	800	400	266	200	160
N° de individuos a recibir en cada AG desde Coordinador	:	800	400	266	200	160
Intervalo de Migración (en generaciones)	:	25	25	25	25	25

DATOS DE LOS PROBLEMAS DE PRUEBA	
Nombre Problema	Problema1003.txt
Número Total de Piezas	97
Número Total de Piezas distintas	24
Dimensiones de la Lámina	350 x 260 $(unidades)^2$

Nombre Problema	Problema1005.txt
Número Total de Piezas	125
Número Total de Piezas distintas	42
Dimensiones de la Lámina	300 x 450 $(unidades)^2$

Nombre Problema	Problema1006.txt
Número Total de Piezas	128
Número Total de Piezas distintas	31
Dimensiones de la Lámina	300 x 450 $(unidades)^2$

Nombre Problema	Problema1007.txt
Número Total de Piezas	168
Número Total de Piezas distintas	16
Dimensiones de la Lámina	330 x 240 $(unidades)^2$

Nombre Problema	Problema1008.txt
Número Total de Piezas	95
Número Total de Piezas distintas	42
Dimensiones de la Lámina	30 x 45 $(unidades)^2$

Número Total de Individuos en la Población Global	4000
Cantidad de AGs	5
Probabilidad de Cruzamiento	90 %

Probabilidad de Mutación	10 %
Número Máximo de Generaciones	2000
Modelos de Migración considerados	Asíncrona y Síncrona

Política de Selección de Individuos a enviar desde cada AG al Coordinador	Se seleccionan los mejores individuos
Política de Selección de Individuos a enviar desde el Coordinador a cada AG	Se seleccionan los mejores individuos
Política de Reemplazo de Individuos en cada AG	Se reemplazan los peores individuos

Los tiempos promedios transcurridos en segundos, el porcentaje del área ocupada y el "*Speedup*", considerando tanto Migración Asíncrona y Síncrona desde uno a cinco Procesadores, se muestran en la Figura N°5.16 para los Problema1003.txt y Problema1005.txt, y en la Figura N°5.17 para los Problema1006.txt, Problema1007.txt y Problema1008.txt.

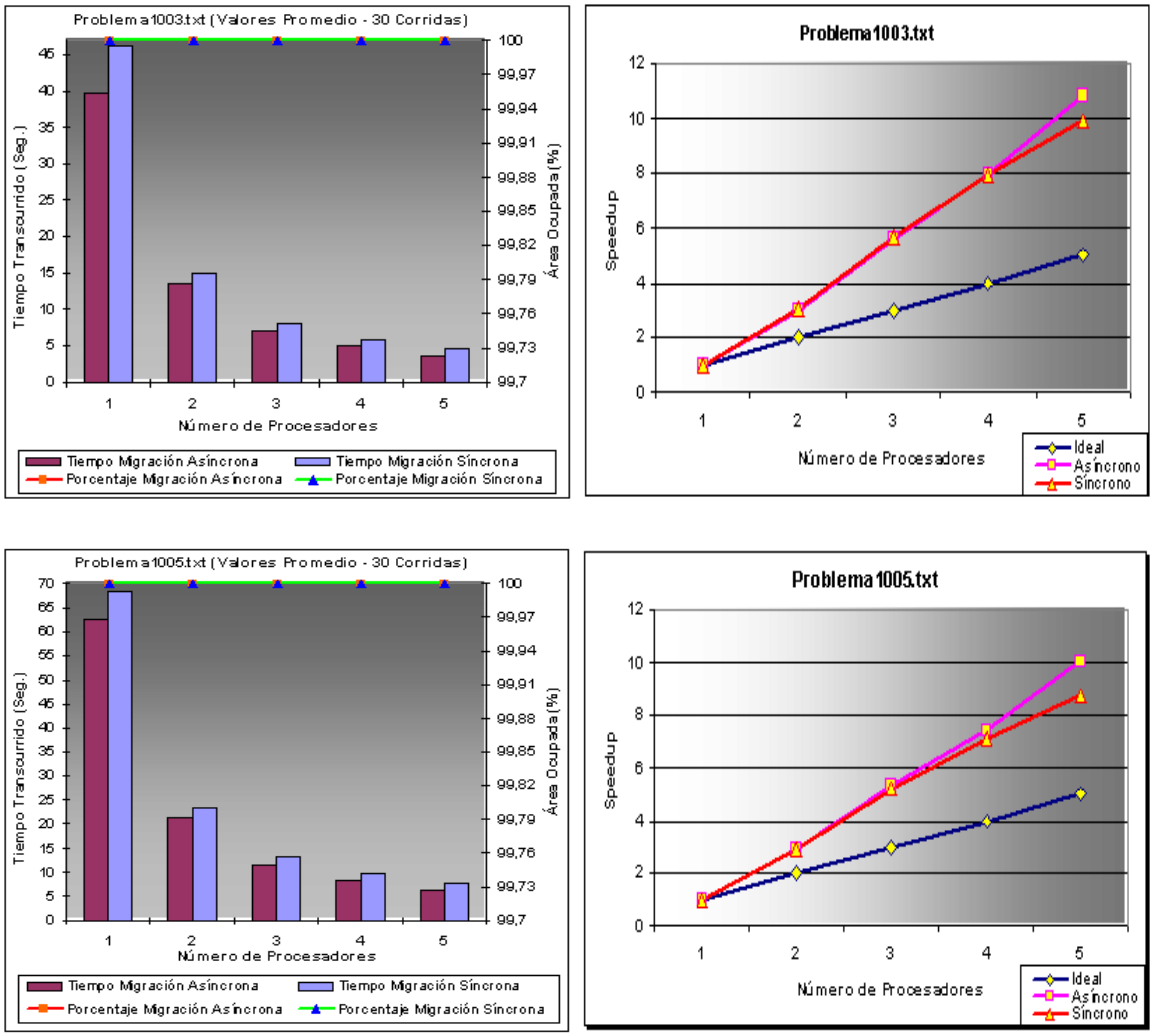
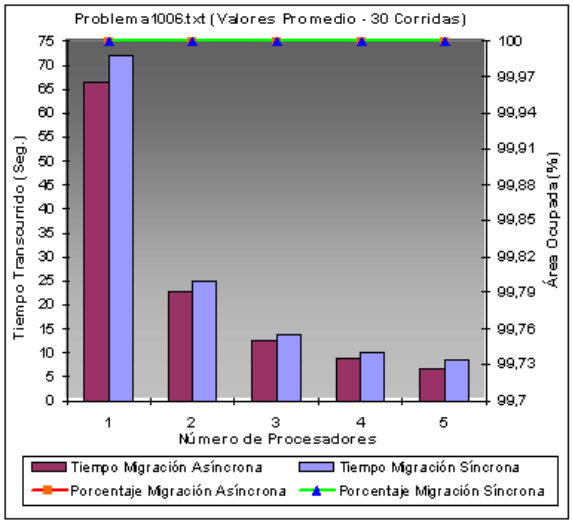
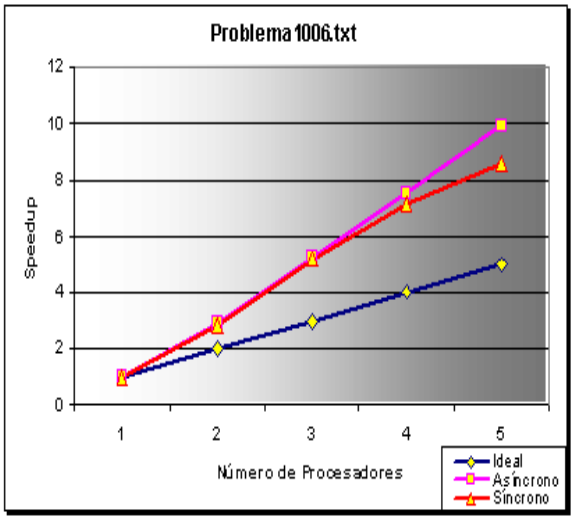
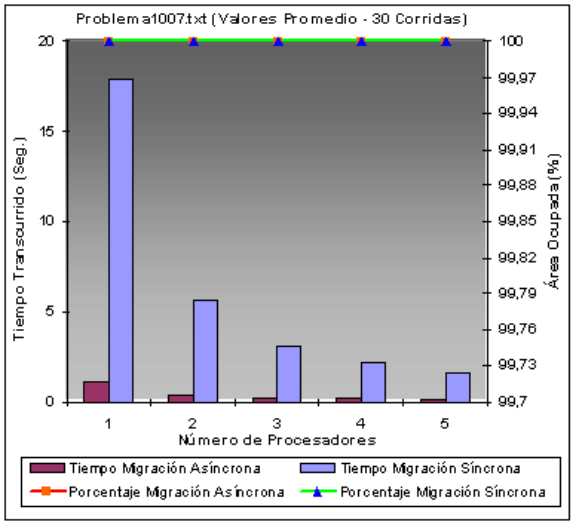


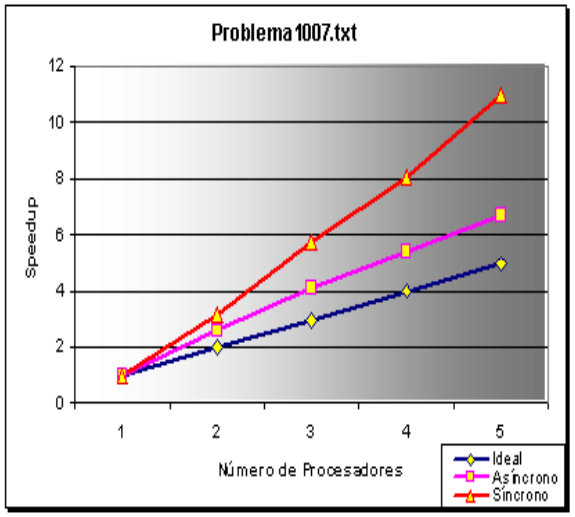
Figura N°5.16: Tiempos transcurridos, porcentaje del área ocupada y "*Speedup*" para treinta corridas considerando el

Problema1003.txt y el Problema1005.txt, con Migración Asíncrona y Síncrona.









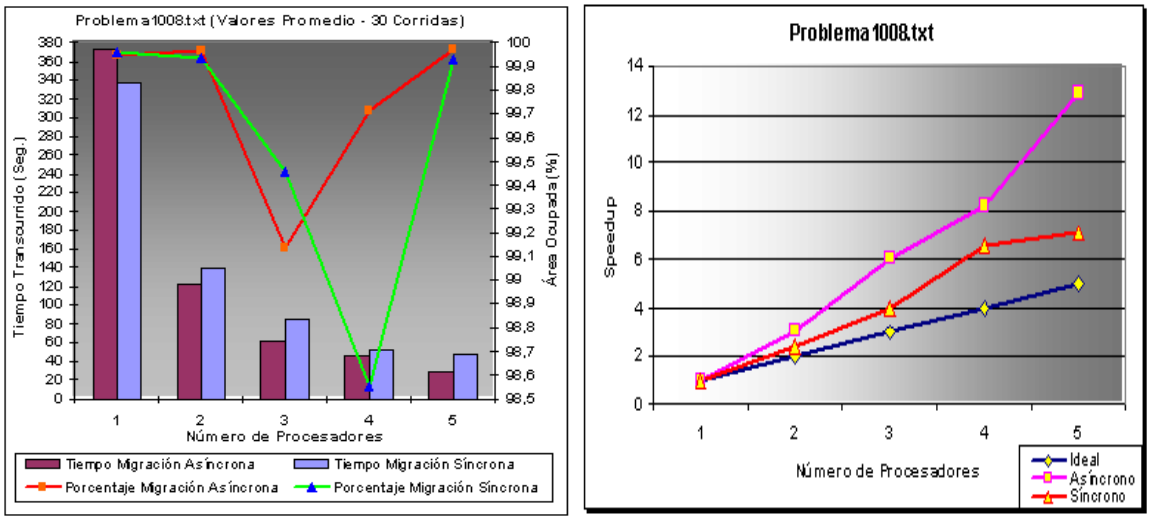


Figura N°5.17: Tiempos transcurridos, porcentaje del área ocupada y "*Speedup*" para treinta corridas considerando el Problema1006.txt, Problema1007.txt y el Problema1008.txt, con Migración Asíncrona y Síncrona.

De las Figuras N°5.16 y N°5.17 se observa un "*Speedup*" Superlineal alcanzado por el AGP, para los problemas de prueba de la Tabla N°5.3. En la mayoría de estos problemas se logra encontrar una solución óptima, ocupando el 100% del área de la lámina.

Sólo en el problema1008.txt, en algunas corridas no se encuentra una solución óptima. Sin embargo, el peor caso de la Migración Asíncrona ocurre al considerar tres procesadores, encontrando el 76,6% de las veces, una solución óptima. En la Migración Síncrona, el peor caso ocurre al considerar cuatro procesadores, en donde el AGP logra el 70% de la veces encontrar una solución óptima. Aún así, se considera aceptable los resultados obtenidos en ambos casos.

La Tabla N°5.4 muestra los tiempos totales en resolver problemas de prueba, los tiempos de comunicación total en enviar los mensajes y el

porcentaje de los tiempos de comunicación con respecto a los tiempos totales, utilizados para resolver algunos de los problemas de prueba. De esta tabla se desprende que los tiempos de comunicación, si bien es cierto que aumentan al aumentar el número de procesadores, pueden despreciarse ya que no alcanzan ni el 0,5% del tiempo total en resolver cada problema. Lo anterior confirma el grano grueso del Algoritmo Genético Paralelo diseñado.

Otro aspecto importante de señalar es que no se logró visualizar disminuciones en el *Speedup* en la medida que aumentaba el número de procesadores, debido a que el cluster utilizado contaba sólo con cinco nodos. Considerando el gran tamaño del grano y el paralelismo natural de los algoritmos genéticos, es posible pensar que los cinco nodos usados están lejos del punto donde comienza a disminuir el "*Speedup*".

La granularidad gruesa se logró mediante una implementación eficiente y un exhaustivo estudio de los mensajes intercambiados entre cada AG y el Coordinador. La información cromosomática se representa a nivel de bits, permitiendo minimizar el tamaño de los mensajes, aún para problemas de gran tamaño.

Tabla N°5.4: Tiempo utilizado en resolver problemas de prueba, utilizando desde uno a cinco Procesadores, con Migración Asíncrona y Síncrona.

		Migración Asíncrona			Migración Síncrona		
Problema	CPU	Tiempo Total (Seg.)	Tiempo Comunic. (Seg.)	% del Tiempo Total	Tiempo Total (Seg.)	Tiempo Comunic. (Seg.)	% del Tiempo Total
Problerma1003.txt	1	39,89680	0,01320	0,03309	46,17420	0,01243	0,02693
	2	13,41187	0,01520	0,11333	15,04633	0,01530	0,10169
	3	7,11660	0,01807	0,25387	8,13387	0,01863	0,22908
	4	5,02847	0,02130	0,42359	5,79617	0,02127	0,36691
	5	3,67960	0,02157	0,58611	4,64673	0,02003	0,43113
Problerma1005.txt	1	62,51157	0,01737	0,02778	68,28400	0,01633	0,02392
	2	21,20343	0,01980	0,09338	23,31203	0,02027	0,08694
	3	11,69433	0,02340	0,20010	13,10127	0,02307	0,17606
	4	8,38423	0,02387	0,28466	9,62817	0,02357	0,24477
	5	6,23963	0,02783	0,44607	7,80300	0,02613	0,33491
Problerma1006.txt	1	66,35077	0,01823	0,02748	72,03423	0,01703	0,02365
	2	22,81167	0,02130	0,09337	25,04533	0,02143	0,08558
	3	12,65707	0,02363	0,18672	13,90967	0,02343	0,16847
	4	8,85517	0,02403	0,27140	10,09837	0,02507	0,24822
	5	6,68570	0,02857	0,42728	8,41967	0,02783	0,33058
Problerma1008.txt	1	373,10910	0,08663	0,02322	337,42757	0,07690	0,02279
	2	122,78863	0,10813	0,08806	139,18050	0,11850	0,08514
	3	61,42297	0,10523	0,17133	84,97467	0,14413	0,16962
	4	45,43027	0,12083	0,26598	51,30753	0,13580	0,26468
	5	28,91107	0,10817	0,37414	47,26780	0,16057	0,33970

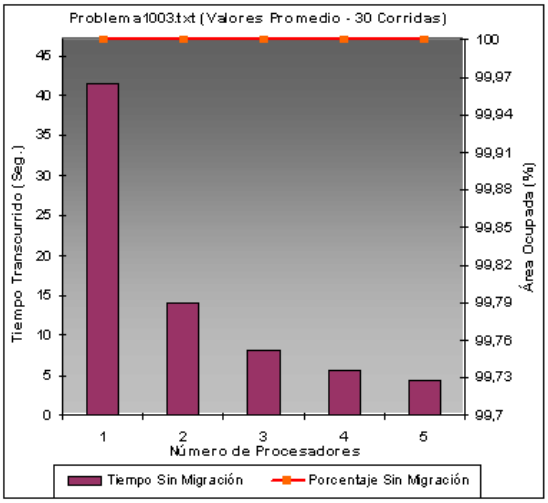
Con el fin de analizar el problema desde varios puntos de vistas, se consideran a continuación, los resultados obtenidos al resolver los problemas de prueba con los mismos parámetros genéticos que muestran las Tablas N°5.2 y N°5.3, pero sin migración.

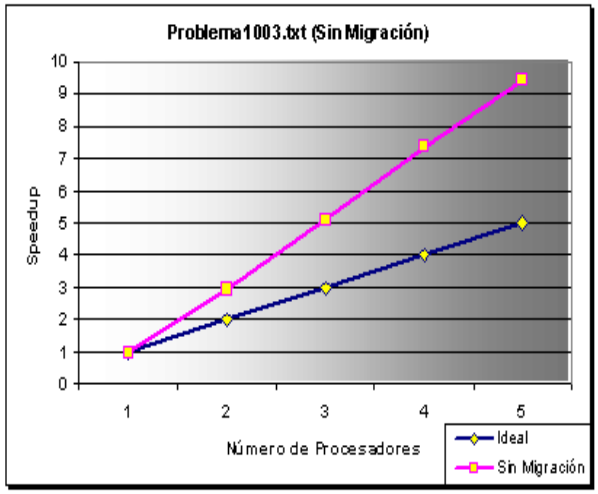
Los tiempos promedios transcurridos en segundos, el porcentaje del área ocupada y el "*Speedup*" para 30 corridas, sin migración y considerando desde uno a cinco Procesadores, se muestran en la Figura N°5.18 para el Problema1003.txt y Problema1005.txt; en la Figura N°5.19 para el Problema1006.txt, Problema1007.txt y Problema1008.txt; y en la Figura N°5.20 para el Problema2025.txt y Problema2026.txt.

En las Figuras N°5.18, N°5.19 y N°5.20 se observa que para algunos problemas se obtienen "*Speedup*" Superlineales (problema1003.txt, problema1005.txt, problema2025.txt), pero siempre con "*Speedup*" menores a los obtenidos al considerar Migración. En el Problema1006.txt y

Problema1007.txt se observa que el "*Speedup*" disminuye al utilizar cuatro y cinco procesadores, respectivamente, lo cual no ocurre al considerar migración. En el Problema1008.txt el "*Speedup*" disminuye por debajo del caso ideal. Finalmente, en el caso del Problema2026.txt se obtiene un "*Speedup*" sublineal, lo cual no ocurre al considerar migración.

En definitiva, el comportamiento del Algoritmo Genético Paralelo es más inestable al resolver el problema sin considerar migración que al considerarla.





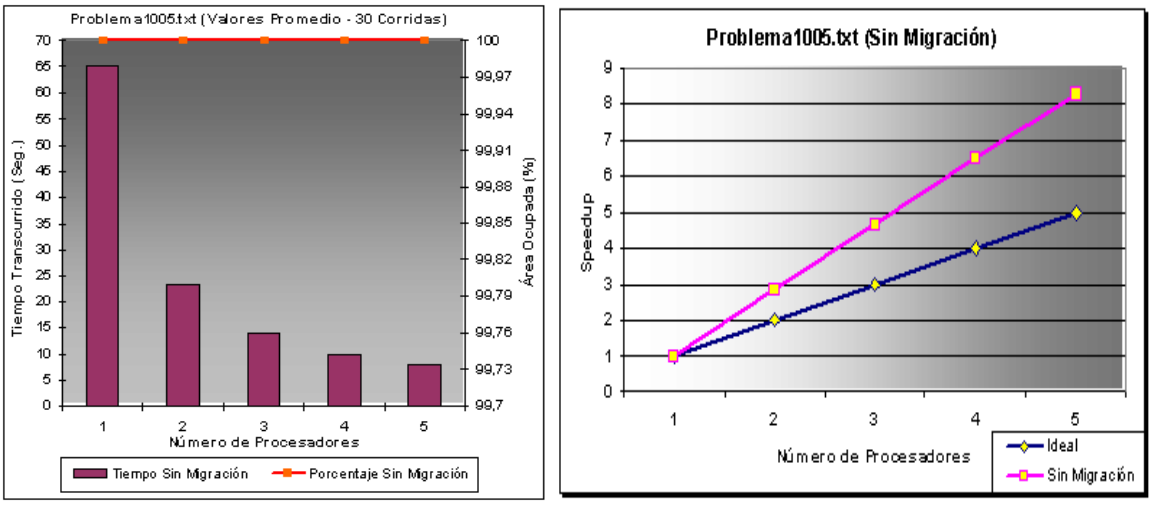


Figura N°5.18: Tiempos transcurridos, porcentaje del área ocupada y "*Speedup*" para treinta corridas considerando el Problema1003.txt y el Problema1005.txt sin migración.

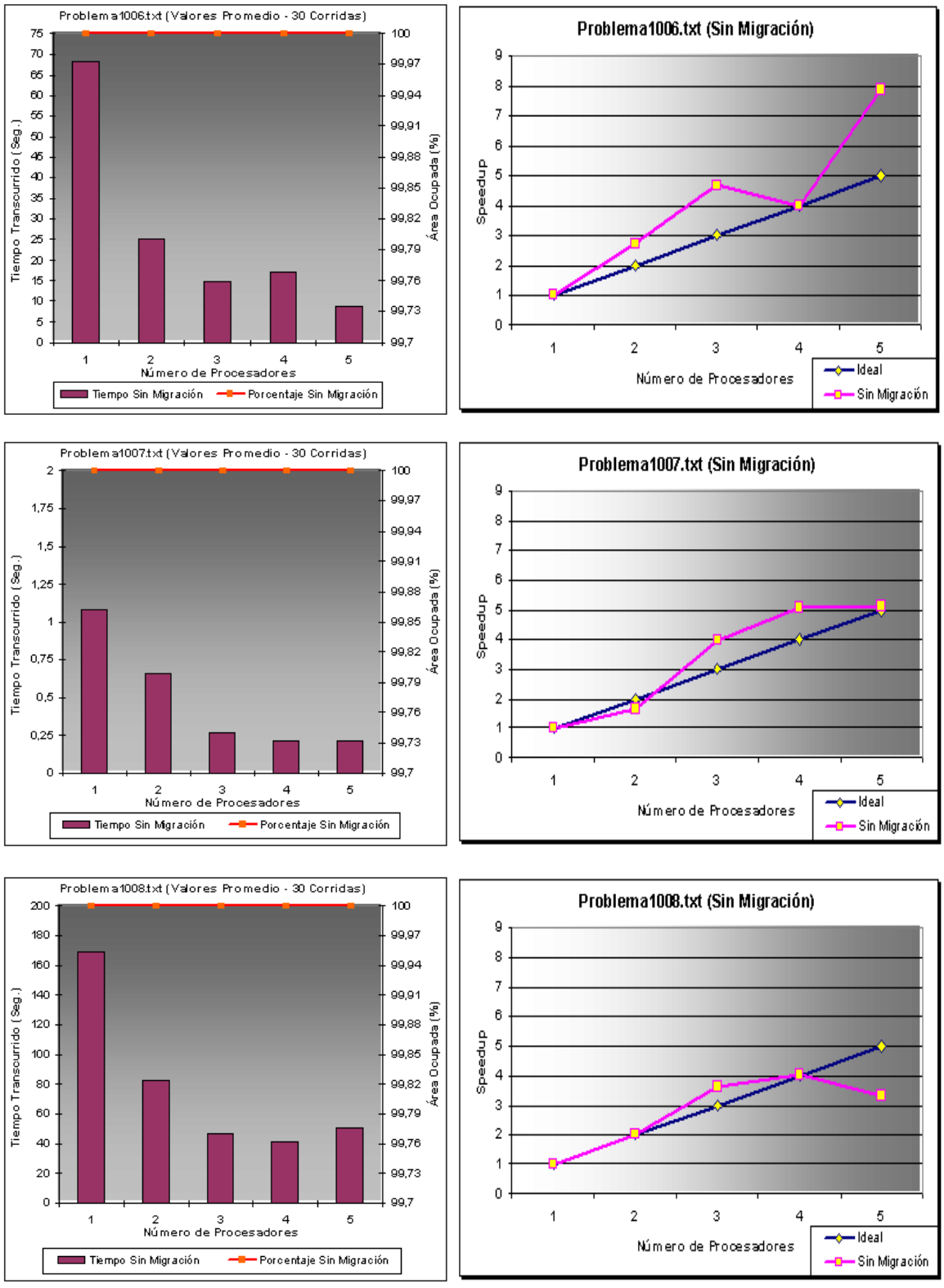


Figura N°5.19: Tiempos transcurridos, porcentaje del área ocupada y "*Speedup*" para treinta corridas considerando el Problema1006.txt,

Problema1007.txt y el Problema1008.txt sin migración.

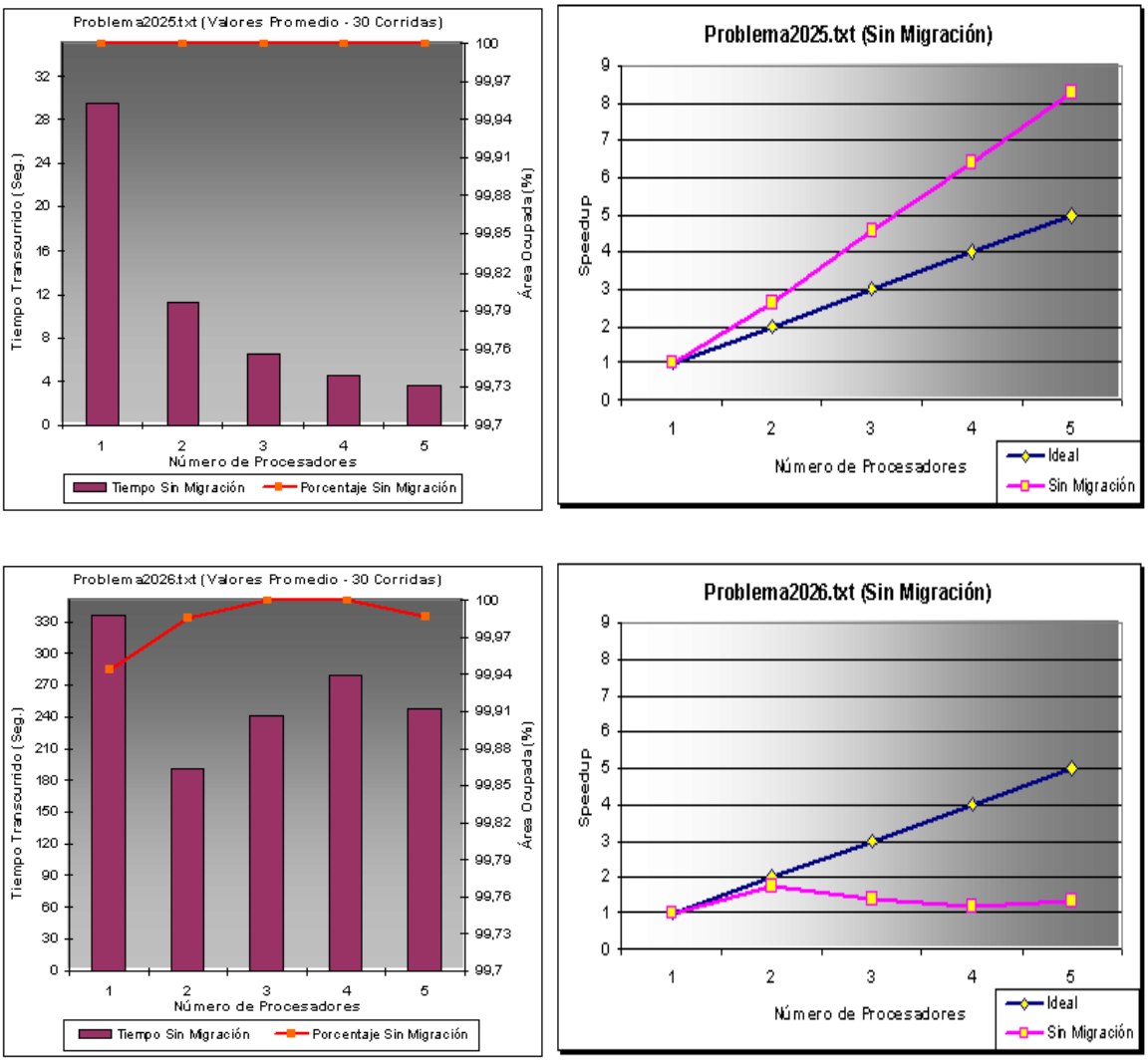


Figura N°5.20: Tiempos transcurridos, porcentaje del área ocupada y "*Speedup*" para treinta corridas considerando el Problema2025.txt y el Problema2026.txt sin migración.

Los tiempos promedios transcurridos (30 corridas) en segundos para cada problema de las Figuras N°5.15 a N°5.20 al considerar desde uno a cinco Procesadores, con Migración Asíncrona, Síncrona y sin migración, se muestran en la Tabla N°5.5.

Tabla N°5.5: Tiempos promedios transcurridos (treinta corridas) en segundos para cada problema de prueba, considerando desde uno a cinco Procesadores, con Migración Asíncrona, Síncrona y sin Migración.

		Tiempos Promedios Transcurridos en Segundos (Valores considerando 30 corridas)				
Problema	**Tipo Migración**	**1CPU**	**2CPU**	**3CPU**	**4CPU**	**5CPU**
problema1003.txt	Sin Migración	41,5490	14,0654	8,1012	5,6105	4,3881
	Asíncrona	39,8968	13,4119	7,1166	5,0285	3,6796
	Síncrona	46,1742	15,0463	8,1339	5,7962	4,6467
problema1005.txt	Sin Migración	64,9937	23,0543	13,9227	9,9675	7,8405
	Asíncrona	62,5116	21,2034	11,6943	8,3842	6,2396
	Síncrona	68,2840	23,3120	13,1013	9,6282	7,8030
problema1006.txt	Sin Migración	68,2153	25,0730	14,6797	17,0917	8,6606
	Asíncrona	66,3508	22,8117	12,6571	8,8552	6,6857
	Síncrona	72,0342	25,0453	13,9097	10,0984	8,4197
problema1007.txt	Sin Migración	1,0822	0,6600	0,2717	0,2128	0,2108
	Asíncrona	1,0911	0,4254	0,2655	0,2014	0,1626
	Síncrona	17,8571	5,6453	3,1066	2,2129	1,6273
problema1008.txt	Sin Migración	168,5481	83,1076	46,6868	41,4655	50,6055
	Asíncrona	373,1091	122,7886	61,4230	45,4303	28,9111
	Síncrona	337,4276	139,1805	84,9747	51,3075	47,2678
problema2025.txt	Sin Migración	29,5004	11,2689	6,4963	4,6175	3,5731
	Asíncrona	29,5416	10,5336	6,1690	4,2730	3,0603
	Síncrona	29,5416	18,7749	7,9499	6,1600	5,0157
problema2026.txt	Sin Migración	336,2899	191,5791	240,7065	278,2460	247,8777
	Asíncrona	162,9858	65,6685	39,0448	28,0342	22,7600
	Síncrona	162,9858	67,3073	40,4997	28,5098	25,2414

Cada valor de la Tabla N°5.5 corresponde al tiempo en segundos por cada procesador en el gráfico de barra (gráfico de la izquierda) de las Figuras N°5.15 a N°5.20, según se considere o no migración.

De los valores de la Tabla N°5.5 se puede responder la pregunta: ¿En qué porcentaje el Algoritmo Genético Paralelo con Migración es más rápido o más lento que el modelo paralelo sin Migración? Para ello, la Tabla N°5.6

muestra por cantidad de Procesadores utilizados y por tipo de migración considerada, el porcentaje en que el AGP con migración es más rápido o más lento que el AGP sin migración. La Tabla N°5.6 considera valores promedios (30 corridas) entre los problemas de prueba de la Tabla N°5.5.

Tabla N°5.6: Porcentaje en que el AGP con migración es más rápido o más lento que el AGP sin Migración.

	Porcentaje en que el AGP con migración es más rápido o más lento que el AGP sin migración				
Tipo Migración	1CPU	2CPU	3CPU	4CPU	5CPU
Asíncrona	8,81% más lento	0,36% más lento	6,04% más rápido	24,56% más rápido	33,25% más rápido
Síncrona	14,07% más lento	17,56% más lento	10,39% más lento	5,07% más rápido	4,78% más lento

De la Tabla N°5.6 se desprende que, por ejemplo, si se utilizan dos Procesadores y Migración Asíncrona, el AGP será 0,36% más lento que el mismo AGP con dos Procesadores y sin Migración. Sin embargo, si se utilizan tres Procesadores y Migración Asíncrona, el AGP será 6,04% más rápido que el mismo AGP con tres Procesadores y sin Migración.

También, se aprecia que al considerar Migración Asíncrona y en la medida que aumenta el número de Procesadores, se obtienen cada vez mejores resultados respecto al mismo modelo paralelo sin migración.

En la Tabla N°5.7 se muestra de acuerdo a la cantidad Procesadores utilizados, el porcentaje en que el AGP con Migración Asíncrona es más rápido o más lento que el AGP con Migración Síncrona. Esta tabla considera valores promedios (30 corridas) entre los problemas de prueba de la Tabla N°5.5.

Tabla N°5.7: Porcentaje en que el AGP con Migración Asíncrona es más rápido o más lento que el AGP con Migración Síncrona.

Porcentaje en que el AGP con Migración Asíncrona es más rápido o más lento que el AGP con Migración Síncrona				
1CPU	2CPU	3CPU	4CPU	5CPU
3,99% más rápido	9,55% más rápido	15,66% más rápido	17,16% más rápido	30,33% más rápido

De la Tabla N°5.7 se observa que el modelo paralelo con migración asíncrona, en promedio, es más rápido que utilizar el mismo modelo pero con migración síncrona. Incluso si se utiliza el modelo paralelo con cinco Procesadores, usar Migración Asíncrona será un 30,33% más rápido que usar Migración Síncrona.

En la Tabla N°5.8 se muestra el "*Speedup*" promedio obtenido al resolver desde uno a cinco Procesadores los problemas de prueba sin Migración, con Migración Asíncrona y Síncrona.

La Figura N°5.21, muestra gráficamente los resultados de la Tabla N°5.8.

Tabla N°5.8: "*Speedup*" promedio obtenido al resolver los problemas de prueba considerando desde uno a cinco Procesadores sin Migración, con Migración Asíncrona y Síncrona.

	Speedup Promedio (Valores considerando 30 corridas)				
Tipo Migración	**1CPU**	**2CPU**	**3CPU**	**4CPU**	**5CPU**
Sin Migración	1,0000	2,3621	3,9964	4,9523	6,2444
Asíncrona	1,0000	2,8173	5,0486	7,0344	9,6022
Síncrona	1,0000	2,6367	4,7896	6,7644	8,2432
Valores Promedios Finales:		**2,6054**	**4,6115**	**6,2503**	**8,0299**

De la Tabla N°5.8 y la Figura N°5.21 se desprende que para dos a cinco Procesadores, los mejores "*Speedup*" (promedios) ocurren al utilizar Migración Asíncrona. Luego, le sigue la Migración Síncrona, y en último lugar resolver los problemas sin considerar migración.

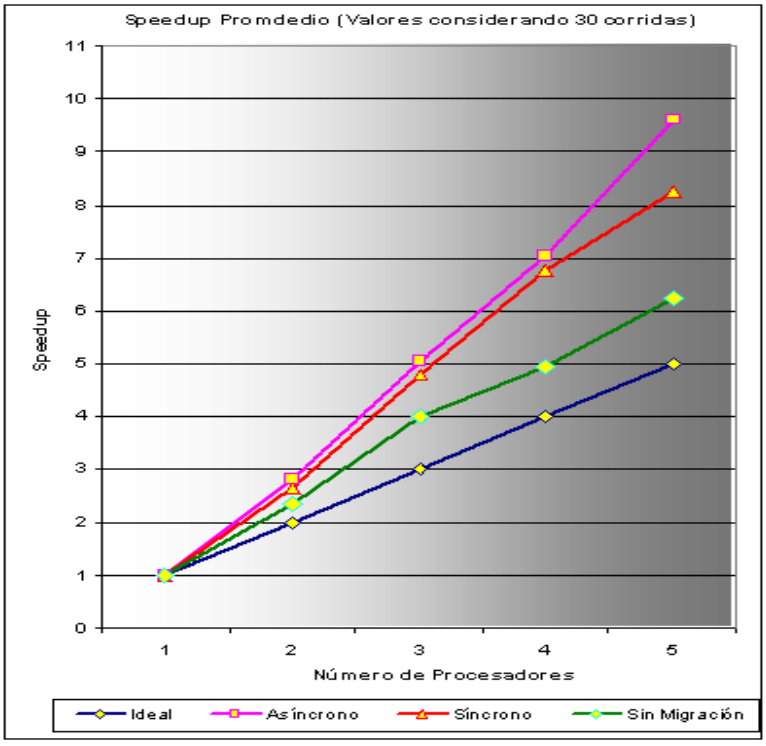


Figura N°5.21: Gráfico del "*Speedup*" promedio obtenido al resolver los problemas de prueba considerando desde uno a cinco Procesadores sin Migración, con Migración Asíncrona y Síncrona.

Finalmente, de todos los resultados generados, se puede decir que el Algoritmo Genético Paralelo diseñado permite obtener, en promedio, un "*Speedup*" de 2,6054 para dos Procesadores, 4,6115 para tres Procesadores, 6,2503 para cuatro Procesadores y 8,0299 para cinco Procesadores.

Capítulo 6. Conclusiones

En este trabajo se ha propuesto e implementado un modelo paralelo basado en Algoritmos Genéticos con Múltiples Sub-Poblaciones que ha permitido resolver el Problema de Corte de Piezas Guillotinadas Bidimensional Restricto. Para ello, se ha construido un *Cluster* de cinco PC interconectados, que intercambian información mediante paso de mensajes. Se han implementado y evaluado los métodos de Migración Asíncrona y Síncrona, bajo un esquema de conexión estático.

Considerando los resultados obtenidos sin migración mostrados en las Figuras N°5.18, N°5.19 y N°5.20 y comparándolos con los resultados obtenidos bajo el esquema de múltiples sub-poblaciones con migración (Figuras N°5.15 a N°5.17) queda en evidencia que la mayor velocidad con que se encuentran las soluciones de la calidad deseada por la implementación paralela, es debido a que la población total de búsqueda se divide en los múltiples procesadores disponibles en el cluster, sacando partido al paralelismo intrínseco de los AGs más que por los efectos de la migración. Para esta comparación se pudo constatar que en el

experimento con Migración Asíncrona se obtuvieron las soluciones con la calidad deseada, en promedio, un 6,04% más rápido con tres Procesadores, 24,56% más rápido con cuatro Procesadores y 33,25% más rápido con cinco Procesadores que al realizar los experimentos sin migración.

El Modelo Asíncrono mostró, en todos los casos, mejores resultados que el Modelo Síncrono. En promedio, el Modelo Asíncrono obtiene las soluciones con la calidad deseada, un 9,55% más rápido con dos Procesadores, 15,66% más rápido con tres Procesadores, 17,16% más rápido con cuatro Procesadores y 30,33% más rápido con cinco Procesadores que al realizar los experimentos con Migración Síncrona.

Lo anterior es debido a que en la comunicación asíncrona no existen barreras de sincronización presentes en el modelo síncrono, por lo que se eliminan tiempos ociosos de los procesadores que terminan antes el procesamiento de una generación. Es importante destacar que con las herramientas usadas, los dos modelos son equivalentes en cuanto a la complejidad de sus implementaciones, en términos de las funciones y las cantidades de líneas de código usadas.

El Speedup Superlineal reportado para los Algoritmos Genéticos Paralelos por varios autores (Cantú-Paz, 2000; Koza y Andre, 1995; Lin, Punch, y Goodman, 1994; Tongchim y Chongstitvatana, 2000; Wilkinson y Allen, 1999) ha sido constatado en este trabajo, obteniendo en promedio, un “*Speedup*” de 2,60 para dos Procesadores, 4,61 para tres Procesadores, 6,25 para cuatro Procesadores y 8,03 para cinco Procesadores.

Las herramientas utilizadas en la implementación y evaluación del modelo paralelo son adecuadas para el uso industrial y comercial. Lo notable son los bajos costos involucrados tanto en software como en hardware para obtener sistemas computacionales con importante poder de

cómputo. Lo anterior se debe a que la plataforma de hardware paralela está constituida por un conjunto de computadores personales de escritorio, interconectados por una red estándar, corriendo un sistema operativo de distribución gratuita, al igual que todas las herramientas y librerías de programación paralelas utilizadas. La alternativa a estos sistemas son máquinas paralelas de altos costos, tanto en hardware especializado como en el sistema operativo y herramientas de programación paralela.

Algunos pilares fundamentales de los Algoritmos Genéticos Paralelos, tales como selección, migración, etc., pueden ser incorporados a otras meta-heurísticas sirviendo como fuente inspiradora para futuros trabajos, en el campo de la optimización combinatoria.

La extensión del modelo paralelo propuesto para resolver otros problemas de corte de piezas bidimensionales, resulta bastante natural, debido a que se puede realizar utilizando la misma representación cromosomática y operadores genéticos, siendo necesario extender los módulos de cálculos geométricos. Para el caso particular del problema no guillotina, la extensión resulta alentadora, pues se deben eliminar solamente restricciones en el procesamiento geométrico.

Respecto a la superlinealidad lograda por el modelo paralelo propuesto, se constatan dos potenciales componentes causales, éstas son:

- El aumento de la probabilidad de encontrar mejores soluciones en forma anticipada respecto de su contraparte secuencial, debido a la exploración simultánea de distintas regiones del

espacio de soluciones, llevadas a cabo por el conjunto de las sub-poblaciones resultantes de la división de la población original.

- La diversidad genética aportada por el proceso de migración. En los resultados experimentales se constatan mejoras, respecto a la calidad, producidas tras un proceso de migración.

Se constata como una fuente distorsionadora de la superlinealidad, el hecho de sobredimensionar la población de individuos que se utiliza en el Algoritmo Genético Secuencial. Para calcular el "*Speedup*", el mejor algoritmo secuencial conocido, lo constituye el AG que contiene la mínima cantidad de individuos necesarios para encontrar soluciones de una calidad dada.

Si bien es cierto, los AG y otras meta-heurísticas conocidas, permiten obtener resultados razonables en la resolución de problemas combinatoriales, éstos presentan una debilidad debido a que un conjunto de parámetros de control de la meta-heurística puede funcionar excelente para ciertas instancias de un determinado problema, a la vez que puede funcionar en forma deficiente para otras instancias. Esta debilidad sugiere la necesidad de incorporar a las meta-heurísticas, nuevos e innovadores mecanismos orientados a la autosintonización de éstas. Una alternativa que parece atractiva son los elementos de computación evolutiva, que pueden ser aplicados para modificar el comportamiento de la meta-heurística, cambiando los parámetros de control, de modo de obtener buenos resultados al resolver distintas instancias.

Referencias Bibliográficas

Adamidis, P. (1994) Review of Parallel Genetic Algorithms Bibliography, Internal Technical Report, Automation and Robotics Laboratory, Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Greece.

Adamidis, P., y Petridis, V. (1996) Co-operating Populations With Different Evolution Behaviours, En Bäck, T., Kitano, H., y Michalewicz, Z. (eds.), Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, Piscataway, NJ: IEEE Service Center, pp. 188-191.

Adamowicz, M., y Albano, A. (1976) A Solution of the Rectangular Cutting Stock Problem, IEEE Transaction on Systems, Man and Cybernetics 6, pp. 302-310.

Alasdair R., Bruce A., Mills J.G. y Smith A.G. (1994) CHIMP-MPI User Guide, Technical Report EPCC-KTP-CHIMP-V2-USER 1.2, Edinburgh Parallel Computing Centre, The University of Edinburgh, Scotland.

Alba, E. y Troya, J. M. (1999) A Survey Of Parallel Distributed Genetic

Algorithms, Complexity Journal 4:4, pp. 31-52.

Albano, A. y Osrini, O. (1980) A Heuristic Solution of the Rectangular Cutting Stock Problem, The Computer Journal 23, pp. 338-343.

Almasi, E. J. y Gottlieb, A. (1994) Highly Parallel Computing, Second edition, Benjamin/Cummings Publishing Company, Redwood City, CA.

Anderson, J. E., y Ferris, M. C. (1990) A Genetic Algorithm for Assembly Line Balancing Problem, Technical Report TR 926, Computer Science Department, University Of Wisconsin-Madison, USA.

Baluja, S. (1992) A Massively Distributed Parallel Genetic Algorithm (mdpGA), Technical Report CMU-CS-92-196, Computer Science Department, Carnegie Mellon University, USA.

Barnett, S. y Kynch, G. J. (1967) Exact Solution of a Simple Stock Problem, Operations Research 15, pp. 1051-1056.

Beasley, J. E. (1985a) An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure, Operations Research 33, pp. 49-64.

Beasley, J. E. (1985b) Algorithms for Unconstrained Two-Dimensional Guillotine Cutting, Journal of the Operational Research Society 36, pp. 297-306.

Beasley, D., Bull, D.R. y Martin, R.R. (1993) An Overview of Genetic Algorithms: Part 1, Fundamentals, University Computing 15:2, pp. 58-69.

Belding, T. C. (1995) The Distributed Genetic Algorithm Revisited, En

Proceedings of the Sixth International Conference on Genetic Algorithms, L. Eschelman (ed.), Morgan Kaufmann, San Francisco, CA, U.S.A., pp. 114-121.

Bianchini, R. y Brown, C. (1993) Parallel Genetic Algorithms on Distributed-Memory Architectures, Proceedings of Transputers: Research and Applications 6, pp 67-82.

Biethahn, J. y Nissen, V. (eds.) (1995) Evolutionary Algorithms in Management Applications, Springer-Verlag, Berlin.

Bischoff, E. E. y Wäscher, G. (1995) Editorial: Cutting and Packing, European Journal of Operations Research 84:3, pp. 503-505.

Bossert, W. (1967) Mathematical Optimization: Are there Abstract Limits on Natural Selection?, En Moorehead, P. S., y Kaplan, M. M. (eds.), Mathematical Challenges to the Neo-Darwinian Interpretation of Evolution, The Wistar Institute Press, Philadelphia, USA.

Bremermann, H. J. (1958) The Evolution of Intelligence. The Nervous System as a Model of its Environment, Technical Report N°1, Contract N°477 (17), Department of Mathematics, University of Washington, Seattle, USA.

Brooks, R. L., Smith, C. A. B., Stone, A. H. y Tutte, W. T. (1940) The Dissection of Rectangles into Squares, Duke Math. J. 7, pp. 312-340.

Burns, G., Daoud, R. y Vaigl, J. (1994) LAM: A Open Cluster Environment for MPI, Proceeding of Supercomputing Symposium 94, pp. 379-386.

Cantú-Paz, E. y Majia-Olvera, M. (1994) Experimental Results in Distributed Genetic Algorithms, En International Symposium On Applied Corporate Computing, Monterey, pp. 99-108.

Cantú-Paz, E. (1995) A Summary of Research on Parallel Genetic Algorithms, Technical Report 95007, Computer Science Department and The Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana-Champaign, USA.

Cantú-Paz, E. (1997) Designing Efficient Master-Slave Parallel Genetic Algorithms, Technical Report 97004, Computer Science

Department and The Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana-Champaign, USA.

Cantú-Paz, E. (1998a) A Survey of Parallel Genetic Algorithms, Calculateurs Paralleles, Reseaux et Systems Repartis 10:2, pp. 141-171.

Cantú-Paz, E. (1998b) Designing Scalable Multi-Population Parallel Genetic Algorithms, Technical Report 98009, Computer Science Department and The Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana-Champaign, USA.

Cantú-Paz, E. (1999) Topologies, Migration Rates, and Multi-Population Parallel Genetic Algorithms, GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, USA, pp. 91-98.

Cantú-Paz, E. (2000) Efficient and Accurate Parallel Genetic Algorithms, Kluwer Academic Publishers, U.S.A.

Clarke J., Fletcher A., Trewin M., Bruce A., Smith A. y Chapple R. (1994) Reuse, Portability and Parallel Libraries, Technical Report TR9413, Edinburgh Parallel Computing Centre, The University of Edinburgh, Scotland.

Coley, D.A. (1999) An Introduction to Genetic Algorithms for Scientists and Engineers, World Scientific Publishing Company, Singapore.

Collins, R. J. (1992) Studies in Artificial Evolution, Ph.D. Thesis, Department of Computer Science, University of California, Los Angeles, CA, USA.

Crainic, T. G. y Toulouse, M. (1997) Parallel Metaheuristics, Fleet Management and Logistics, T.G. Crainic, G. Laporte (ed.), Kluwer Academic Publishers, Norwell MA.

Christofides, N. y Whitlock, C. (1977) An Algorithm for Two-Dimensional

Cutting Problems, Operations Research 25, pp. 30-44.

Christofides, N. y Hadjiconstantinou, E. (1995) An Exact Algorithm for Orthogonal 2D Cutting Problems using Guillotine Cuts, European Journal of Operations Research 83, pp. 21-38.

Dagli, C. H. y Tatoglu, M. Y. (1987) An Approach to Two-Dimensional Cutting Stock Problem, International Journal of Production Research 25, pp. 175-190.

Dagli, C. H. y Poshyanonda, P. (1997) New Approaches to Nesting Rectangular Patterns, Journal of Intelligent Manufacturing 8:3, pp. 177-190.

Dantzig, G. B. (1951) Maximization of a Linear Function of Variables Subject to Linear Inequalities, Activity Analysis of Production Allocation, T. C., Koopmans, (ed.) Cowles Commission Monograph, 13, Wiley, New York.

Davis, L.D. (ed.) (1987) Genetic Algorithms and Simulated Annealing, Research Notes in Articial Intelligence, Pitman Publishing, London.

Davis, L. D. (ed.) (1991) Handbook of Genetic Algorithms, VNR Computer Library, Van Nostrand Reinhold, New York, U.S.A.

Díaz A., Glover F., Ghaziri H. M., González J. L., Laguna M., Moscato P. y Tseng F. T. (1996) Optimización Heurística y Redes Neuronales en Dirección de Operaciones e Ingeniería, Editorial Paraninfo, Madrid.

Dyckhoff, H. (1981) A New Linear Programming Approach to the Cutting Stock Problem, Operations Research 29, pp. 1092-1104.

Dyckhoff, H. y Wäscher, G., (eds.) (1990) Special Issue on Cutting and Packing, European Journal of Operational Research 44:2.

Dyckhoff, H. (1990) A Typology of Cutting and Packing Problems, European Journal of Operational Research 44, pp. 145-159.

Dyckhoff, H. y Schster, K. P. (eds) (1991) Verpackungslogistik/Packaging logistics, OR Spektrum 13:4.

Dyckhoff, H. y Finke, U. (1992) Cutting and Packing in Production and Distribution, Physica-Verlag, Heidelberg, Germany.

Eilon, S. y Christofides, N. (1971) The Loading Problem, Management Science 17, pp. 259-268.

Eisemann, D. (1957) The Trim Problem, Management Science 3, pp. 279-284.

Fogel, D.B. y Anderson, R. W. (2000) Revisiting Bremermann's Genetic Algorithm: I. Simultaneous Mutation of All Parameters, Proceedings of Congress on Evolutionary Computation 2000 (CEC 2000), La Jolla Marriot Hotel, La Jolla, California, U.S.A., pp. 1204-1209.

Fogel, D.B. y Fraser, A.S. (2000) Running Races with Fraser's Recombination, Proceedings of Congress on Evolutionary Computation 2000 (CEC 2000), La Jolla Marriot Hotel, La Jolla, California, U.S.A., pp. 1217-1222.

Forrest, S. (1993) Genetic Algorithms: Principles of Natural Selection Applied to Computation, Science 261, pp. 872-878.

Fowler, R.J., Paterson, M. S. y Tatimoto, S. L. (1981) Optimal Packing and Covering in the Plane are NP-Complete, Information Processing

Letters 12:3, pp. 133-137.

Fraser, A.S. (1957) Simulation of Genetic Systems by Automatic Digital Computers. II. Effects of Linkage on Rates under Selection, Australian Journal of Biological Sciences 10, pp. 484-491.

Fraser, A.S. (1960) Simulation of Genetic Systems by Automatic Digital Computers. IV. Epistatis, Australian Journal of Biological Sciences 13, pp. 329-346.

Garey, M. R. y Jonson, D. S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, USA.

Ghandforoush, P. y Daniels, J. J. (1992) A Heuristic Algorithm for the Guillotine Constrained Cutting Stock Problem, ORSA Journal on Computing 4, pp. 351-356.

Gilmore, P. C., y Gomory, R. E. (1961) A Linear Programming Approach to the Cutting Stock Problem, Operations Research 9, pp. 848-859.

Gilmore, P. C., y Gomory, R. E. (1963) A Linear Programming Approach to the Cutting Stock Problem, Operations Research 11, pp. 863-888.

Gilmore, P. C. y Gomory, R. E. (1965) Multistage Cutting Stock Problems of Two and More Dimensions, Operations Research 13, pp. 94-120.

Gilmore, P. C., y Gomory, R. E. (1966) The Theory and Computation of

Knapsack Functions, Operations Research 14, pp. 1045-1074.

Goerzen, J. (2000) Linux Programming Bible, Wiley Publishing, Inc., USA.

Goldberg, D. E. (1989a) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley Publishing Company, Massachusetts.

Goldberg, D. E. (1989b) Sizing Populations for Serial and Parallel Genetic Algorithms, Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, San Mateo, pp. 70-79.

Goldberg, D. E., Deb, K. y Thierens, D. (1993) Toward A Better Understanding Of Mixing In Genetic Algorithms, Journal Of the Society Of instrument And Control Engineers 32:1, pp. 10-16.

Goldberg, D., E. (1994) Genetic And Evolutionary Algorithms Come Of Age, Communications of the ACM 37:3, pp. 113-119.

Goldberg, D., Zakrzewski K., Sutton B., Gadient R., Chang C., Gallego P., Miller B. y Cantú-Paz E. (1997) Genetic Algorithm: A Bibliography, Technical Report 97011, Computer Science Department and The Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana-Champaign, USA.

Gorg-Schleuter, M. (1990) Explicit Parallelism of Genetic Algorithms Through Population Structures, Proceedings of Parallel Problem Solving from Nature, 1st Workshop, H. P. Schwefel & R. Manner (eds.), Springer-Verlag, Berlin, Germany, pp. 150-159.

Grefenstette, J. J., Leuze, M. R. y Pettey, C. B. (1987) A Parallel Genetic Algorithm, Proceedings of the Second International Conference on Genetic Algorithms, John J. Grefenstette (ed.), Lawrence Erlbaum Associates, Publishers, pp. 155-161.

Gropp W., Lusk E. y Skjellum A. (1995) Using MPI, Portable Parallel Programming with the Message-Passing Interface, First Edition, MIT Press, Cambridge.

Gropp W., Lusk E., Doss N. y Skjellum A. (1996) A High Performance, Portable Implementation of the MPI Message Passing Interface Standard, Parallel Computing 22, pp. 789–828.

Grosso, P. B. (1985) Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model, Ph.D. Thesis, University Of Michigan, Ann Arbor, Michigan.

Haessler, R. W. (1971) A Heuristic Solution to a Nonlinear Cutting Stock Problems, Management Science 17:12, pp. 793-803.

Haessler, R. W. (1980) A Note on Some Computational Modifications to the Gilmore-Gomory Cutting Stock Algorithm, Operations Research 28, pp. 1001-1005.

Herrera, F. y Lozano, M. (2000) Gradual Distributed Real-Coded Genetic Algorithms, IEEE Transactions On Evolutionary Computation 4:1, pp. 43-63.

Herz, J. C. (1972) A Recursive Computing Procedure for Two-Dimensional Stock Cutting, IBM Journal of Research and Development 16,

pp. 462-469.

Hines, J., Thorpe, J. T., Winiecki, K. B. y Harris, F. C. (1995) Solving Quadratic Assignement Problems with Parallel Genetic Algorithms, S. Louis, (ed.), Proc. Of The ISCA Int. Conf. on Intelligent Systems, San Francisco, CA, pp. 11-16.

Hinxman, A. (1980) The Trim-Loss and Assortment Problems: A Survey, European Journal of Operational Research 5, pp. 8-18.

Holland, J. H. (1975) Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, Michigan.

Holland, J. H. (1992) Adaptation in Natural and Artificial Systems, Second Edition, University of Michigan Press, Ann Arbor, Michigan.

Holthöfer, N. y Tschöke, S. (1995) A New Parallel Approach to the Constrained Two-Dimensional Cutting Stock Problem, Parallel Algorithms for Irregularly Structured Problems, Proceedings, Lecture Notes in Computer Science 980 A. Ferreira, J. Rolim, (ed.), Springer Berlin, Heidelberg, New York

Hopper, E. (2000) Two-Dimensional Packing utilizing Evolutionary Algorithms and other Meta-Heuristic Methods, Ph.D. Thesis, School of Engineering, University of Wales, Cardiff.

Kantorovich, L.V. y Zalgaller V.A. (1951) Optimal Calculation for Subdivision in the Material Industry, Lenzidat, Leningrad.

Kantorovich, L.V. (1960) Mathematical Methods of Organizing and Planning Production, Science 6, pp. 366-422.

Kdevelop P. (1998) Kdevelop Project HomePage, Fachschaftsrat Informatik, Universität Potsdam, German, (Febrero, 2002), http://www.kdevelop.org.

Kernighan, B.W. y Ritchie, D.M. (1991) El Lenguaje de Programación C,

Segunda Edición, Prentice Hall, México.

Koza, J. R. (1992) Genetic Programming: on the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge.

Koza, J. R. y Andre, D. (1995) Parallel Genetic Programming on a Network of Transputers, Technical Report CS-TR-95-1542, Computer Science Department, Stanford University, U.S.A.

Kri, A. F. (1996) Modelos Paralelos para Algoritmos Genéticos con Memoria Distribuida, Trabajo de Título de Ingeniería Civil en Informática, Universidad de Santiago de Chile, Chile.

Lumsdaine, D. (2001) LAM / MPI Parallel Computing, Indiana University, Bloomington, USA, (Abril, 2002), http://www.lam-mpi.org.

Lin, S., Punch, W. F. y Goodman, E. D. (1994) Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach, Proceedings of the Sixth IEEE Symposium on Parallel and Distributed Processing, Michigan State University, pp. 28-37.

Lirow, Y. (ed.) (1992) Cutting Stock: Geometric Resource Allocation, Math. Comput. Mod. 16:1.

Luke, S. (1998) Genetic Programming Produced Competitive Soccer Softbot Teams for Robocup 97, Genetic Programming 1998: Proceedings on the Third Annual Conference, Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H. y Riolo, R. (eds.), Morgan Kaufmann Publishers, pp. 214-222.

MacLeod, B., Moll, R., Girkar, M. y Hanifi, N. (1993) An Algorithm for the 2D Guillotine Cutting Stock Problem, European Journal of Operations Research 68, pp. 400-412.

Man, K.F., Tang, K.S. y Kwong, S. (1999) Genetic Algorithms: Concepts and Design, Springer-Verlag, London.

Martello, S. (ed.) (1994) Knapsack, Packing and Cutting, INFOR 32, pp. 3-4.

Maruyama, T., Hirose, T. y Konagaya, A. (1993) A Fine-Grained Parallel Genetic Algorithm for Distributed Parallel Systems, Proceedings of the Fifth International Conference on Genetic Algorithms, Stephanie Forrest (ed.), Morgan Kaufmann Publishers, San Mateo, U.S.A., pp. 184-190.

Michalewicz, Z. (1996) Genetic Algorithms + Data Structures = Evolution Programs, Third Edition, Springer-Verlag, Berlin.

Michalewicz, Z y Fogel, D.B. (2000) How To Solve It: Modern Heuristics, Springer-Verlag, Berlin.

Mitchell, M. (1996) An Introduction to Genetic Algorithms, The MIT Press, Cambridge, MA.

MPI Forum, (1994) MPI: A Message-Passing Interface Standard, Technical Report UT-CS-94-230, Message Passing Interface Forum, University of Tennessee, USA, http://www.mpi-forum.org/docs/mpi-11.ps.

MPI Forum, (1997) MPI-2: Extensions to the Message-Passing Interface, Technical Report, Message Passing Interface Forum, University of Tennessee, USA, http://www.mpi-forum.org/docs/mpi-20.ps.

Mühlenbein, H. (1989) Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization, Proceeding of the Third

Internacional Conference on Genetic Algorithms, J. D. Schaffer (ed.), Morgan Kaufmann Publishers, San Mateo, pp. 416-421.

Muñoz, R., H. (1994) Un Algoritmo Genético Particular para el Problema del Cutting Stock, Trabajo de Título de Ingeniería Civil en Informática, Universidad de Santiago de Chile, Chile.

Muñoz, M. J. (1995) Desarrollo de un Algoritmo Genético para el Problema de Corte de Lámina con Piezas Irregulares, Trabajo de Título de Ingeniería Civil Informática, Universidad de Santiago de Chile, Chile.

Negus, C. (1999) RedHat Linux Bible, Wiley Publishing, Inc., USA.

Nicklas, L., Atkins R., Setia S. y Wang P. (1998) The Design and Implementation of a Parallel Solution to the Cutting Stock Problem, Journal Concurrency: Practice and Experience 10, pp. 783-805.

Nowostawski, M., y Poli, R. (1999) Parallel Genetic Algorithm Taxonomy, L. C. Jain, (ed.), Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES'99), Adelaide, pp. 88-92.

Oliveira, J. F., y Ferreira, J. S. (1990) An Improved Version of Wang's Algorithm for Two-Dimensional Cutting Problems, European Journal of Operational Research 44, pp. 256-266.

Otten, R. H. J. M. (1982) Automatic Floorplan Design, Proceedings of the

Nineteenth Design Automation Conference, pp. 261-267.

Parada, V., Gómes, A. y de Diego, J. (1995) Exact Solutions for Constrained Two-Dimensional Cutting Problem, European Journal of Operational Research 84:3, pp. 633-644.

Parada, V., Muñoz, R. y Gómes, A. (1995) A Hybrid Genetic Algorithms for the Two-Dimensional Guillotine Cutting Problem, Evolutionary Algorithms in Management Applications, Springer-Verlag, Bietahn, J. & Nissen, V., (eds), pp. 183-196.

Parada, V., Sepúlveda M., Solar M. y Gómes A. (1997) Solution for the Constrained Guillotine Cutting Problem by Simulated Annealing, Computers and Operations Research 25:1, pp. 37-47.

Paull, A. E. (1956) Linear Programming: A Key to Optimum Newsprint Production, Pulp and Paper Magazine of Canada 57, pp. 85-90.

RedHat (2002) RedHat, Linux, Embedded Linux and Open Source Solutions, RedHat Inc., (Febrero, 2002), http://www.redhat.com.

Reeves, C.R. (1995) Genetic Algorithms. In Modern Heuristic Techniques for Combinatorial Problems, Reeves C.R., (ed.), John Wiley & Sons, Inc., New York.

Rojas, R. G. (2000) Modelamiento y Resolución del Problema de Corte de Piezas Guillotinadas Bidimensional Restricto por medio del Algoritmo de Búsqueda Tabú, Tesis de Magíster en Ingeniería Informática, Universidad de Santiago de Chile, Chile.

Rojas W. V. (2001) Un Algoritmo Genético para el Problema de Corte no Guillotina, Tesis de Magíster en Ingeniería Informática, Universidad de Santiago de Chile, Chile.

Sales, D. Z. (1997) Una Contribución a la Resolución de Problemas de Optimización Combinatoria a través de Métodos Meta-Heurísticos, Tesis de Magíster en Ingeniería Informática, Universidad de Santiago de Chile, Chile.

Schwehm, M. (1992) Implementation of Genetic Algorithms on Various Interconnection Networks, Parallel Computing And Transputer Applications, IOS Press, Amsterdam.

Schildt, H. (2000) C Manual de Referencia, Mc Graw Hill, Cuarta Edición, España.

Shpitalni, M. y Manevich, V. (1996) Optimal Orthogonal Subdivision of Rectangular Sheets, Journal of Manufacturing Science and Engineering 118, pp. 281-288.

Solar, M (1992) Modelo Paralelo para Aprendizado Automático: Redes Neuronais e Algoritmos Genéticos, Tesis de Dr.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.

Spiessen, P. y Manderick, B. (1990) A Genetic Algorithm for Massively Parallel Computers, Parallel Processing in Neural Systems and Computers, Dusseldorf, Germany, North-Holland Publisher, Amsterdam.

Spiessen, P. y Manderick, B. (1991) A Massively Parallel Genetic Algorithm:

Implementation and First Analysis, Belew R. K. & Booker L. B., (ed.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, CA, pp. 279-286.

Stanley, T. J., y Mudge, T. (1995) A Parallel Genetic Algorithm for Multiobjective Microprocessor Design, Eschelman, L. (ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, pp. 597-604.

Starkweather, T., Whitley, D., y Mathias, K. (1991) Optimization using Distributed Genetic Algorithms, Schwefel, H. P. y Männer, R. (eds.), Parallel Problem Solving from Nature, Stringer-Verlag, Berlin.

Sweeney, P. E. y Paternoster, E. R. (1991) Cutting and Packing Problems: An Updated Literature Review, Working Paper N°654, School of Business, University of Michigan, USA.

Tanese, R. (1987) Parallel Genetic Algorithms for a Hypercube, Proceedings of the Second International Conference on Genetic Algorithms, Grefenstette J. J. (ed.), Lawrence Erlbaum Associates Publishers, New Jersey, USA, pp. 177-183.

Tanese, R. (1989a) Distributed Genetic Algorithms, Proceedings of the Third International Conference on Genetic Algorithms, Schaffer, J. D. (ed.), Morgan Kaufmann, San Mateo, CA, pp. 434-439.

Tanese, R. (1989b) Distributed Genetic Algorithms for Function

Optimization, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan.

Tomassini, M. (1999) Parallel and Distributed Evolutionary Algorithms: A Review, Evolutionary Algorithms in Engineering and Computer Science 7, pp. 113-133.

Tongchim, S. y Chongstitvatana, P. (2000) Comparison between Synchronous and Asynchronous Implementation of Parallel Genetic Programming, Proceedings of Fifth International Symposium on Artificial Life and Robotics (AROB), Oita, Japan, pp. 251-254.

Vajda, S. (1958) Trim Loss Reduction, Readings in Linear Programming 21, pp. 78-84.

Vaughan, P. L., Skjellum A., Reese D. S. y Chen Cheng F. (1995) Migrating from PVM to MPI, Part I: The Unify System, Fifth Symposium on the Frontiers of Massively Parallel Computation. IEEE Computer Society Technical Commitee on Computer Architecture, IEEE Computer Society Press, pp 488-495.

Viswanathan, K. V. y Bagchi, A. (1988) An Exact-Best-First Search Procedure for the Constrained Rectangular Guillotine Knapsack Problems, Proceedings of the American Association for Artificial Intelligence, St. Paul, MN, pp. 145-149.

Wall, K. (2000) Programación en Linux con Ejemplos, Primera Edición,

Prentice Hall, Buenos Aires.

Wang, P. Y. (1983) Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems, Operations Research 31, pp. 573-586.

Wilkinson, B. y Allen, M. (1999) Parallel Programming: Techniques and Applications using Networked Workstations and Parallel Computers, Prentice Hall, USA.

Zeigler, B. P., y Kim, J. (1993) Asynchronous Genetic Algorithms on Parallel Computers, Forrest S. (ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp. 600.

Zhaksilikov, M. y Harris, F. (1996) Comparison of Different Implementations of Parallelization of Genetic Algorithms, Proceedings of the ISCA Fifth International Conference on Intelligent Systems (IS '96), Reno, NV.

Apéndice A

Formatos de los Archivos de Entrada y Salida del Algoritmo Genético Paralelo

A continuación se presentan los formatos de los Archivos de Entrada y Salida que son manejados por el Coordinador para implementar el Algoritmo Genético Paralelo que resuelve el PCPGBR.

Formatos de los Archivos de Entrada

El Coordinador usa dos archivos de texto como entrada, uno que le indica los parámetros genéticos a usar y el otro los datos del problema a resolver.

- **Archivo de Entrada con parámetros genéticos.** El Coordinador puede resolver tantos problemas de corte de piezas como los especificados en este archivo, cuyo formato se muestra en la Figura N°A.1.

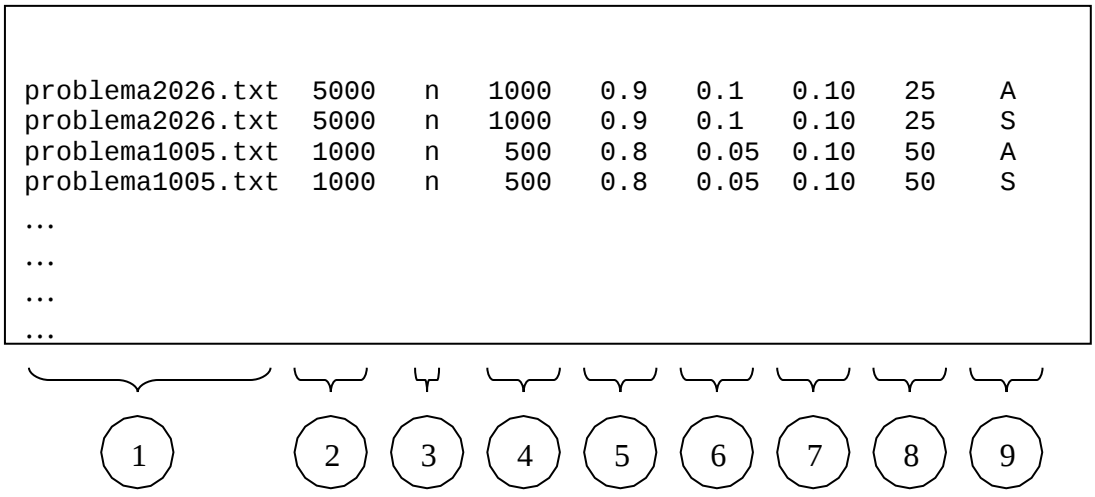


Figura N°A.1: Archivo de Entrada que indica al Coordinador los parámetros genéticos a usar para cada problema a resolver.

En la Figura N°A.1 el significado de cada columna es el siguiente:

1 Nombre del Archivo donde se encuentra la información del problema a resolver. Este archivo debe indicar las dimensiones de la lámina a considerar, cuantas piezas están involucradas, sus dimensiones y las restricciones que posee cada una.

2 Indica el número de individuos total de la Población.

3 Indica si debe generarse (“y”) o no (“n”) archivo de salida que muestre en cada AG y por cada generación, los cromosomas de 1 y 0 que representan a los individuos de cada Sub-Población.

4 Indica el número de generaciones que debe ser considerada por el Coordinador y por cada AG.

5 Indica probabilidad de cruzamiento a usar por cada AG. Este valor se considera entre 0 y 1.

6 Indica probabilidad de mutación a usar por cada AG. Este valor se considera entre 0 y 1.

7 Indica fracción de individuos que deben ser considerados en el proceso de migración. Se considera la misma fracción para aquellos individuos que deben ser enviados desde cada Sub-Población hacia el Coordinador, como aquellos que serán enviados desde el Coordinador hacia cada Sub-Población. Este valor se considera entre 0 y 1.

8 Indica cantidad de generaciones necesarias para que se realice el

9

proceso de migración.

Indica el tipo de Migración a realizar. Se utiliza la letra “A” para indicar Migración Asíncrona y la letra “S” para Migración Síncrona.

- **Archivo de Entrada con datos del problema a resolver.** El Coordinador debe conocer los datos del problema de corte de pieza restricto a resolver, por lo que el formato de este archivo de texto se detalla a continuación.

Formato archivo de texto con datos del problema de corte de pieza restricto

a) Dimensiones (ancho, alto) de la lámina rectangular a considerar.
b) Número de Piezas (N) distintas a considerar en archivo.
c) Por cada pieza “*i*” (*i* = 1,..., N) del problema se considera: ancho de la pieza, alto de la pieza, número máximo de piezas del mismo tipo que pueden ser cortadas desde la lámina rectangular.

Un ejemplo de este archivo es el que se muestra en la Figura N°A.2.

```
99 66
15
10 13 6
12 38 6
75 64 1
90 41 3
19 44 5
22 22 1
93 41 9
46 29 4
26 32 3
91 47 4
82 11 10
46 48 7
72 64 7
13 4  1
83 47 5
```

Figura N°A.2: Archivo de Entrada que indica al Coordinador los datos del problema de corte de pieza restricto a resolver.

Formatos de los Archivos de Salida

El Coordinador proporciona varios archivos de texto de salida, siendo los más importantes aquellos que se detallan a continuación.

- **Archivo de texto con información del "*layout*" del problema planteado.** Archivo que entrega el "*layout*" resultante del problema planteado. La Figura N°A.3 muestra el formato del archivo y se explica cada una de las líneas de información hasta el carácter "@". Todas las líneas sobre este carácter se consideran como información a desplegar al usuario, mientras que las líneas por debajo de este carácter entregan información referente a las coordenadas, tipo rotación, área ocupada y si las coordenadas corresponden a pérdidas o piezas del "*layout*" final del problema. La columna "*Tipo_Pieza*" muestra el tipo de pieza que participa en el layout final como solución del PCPGBR. Un valor positivo en esta columna, indica que se está considerando un tipo de pieza sin rotación. Por el contrario, un valor negativo indica un tipo de pieza con rotación. En ambos casos, la sexta columna (*Estado*_Pieza) indica una "*G*" de *Ganancia*. Un valor cero en "*Tipo_Pieza*" indica que la pieza corresponde a una pérdida. Si "*Estado_Pieza*" tiene una "*P*" indica una *Pérdida Interna*, y una "*E*" indica una *Pérdida Externa*. Las coordenadas de cada tipo de pieza están representadas por *POS_X_INI*, *POS_Y_INI*, *POS_X_FIN* y

POS_Y_FIN. El área ocupada por cada tipo de pieza, en unidades cuadradas, se muestra en la última columna de la tabla de valores.

```
26 9 52 63       => Valores que indican la cantidad de líneas usadas para desplegar los parámetros, el
                                          número de piezas más pérdidas generadas, el ancho de la
                                          lámina y alto de la lámina, respectivamente.
ARCHIVO_PROBLEMA              : problema34.txt=> Nombre del archivo procesado.
CORRIDA EN ARCHIVO REPORTE    : 1             => Indica a qué línea del archivo de entrada con parámetros
                                          genéticos corresponde.
CANTIDAD_PIEZAS               : 88       => Cantidad de piezas total del problema a resolver.
CANTIDAD_TIPOS_PIEZAS         : 15       => Cantidad de piezas distintas del problema.
ANCHO_LAMINA                  : 52       => Ancho de la lámina.
LARGO_LAMINA                  : 63       => Largo de la lámina.
MAXIMO_GENERACIONES           : 50       => Máximo número de generaciones considerado por el Coordinador.
TAMANO_POBLACION (ind.)       : 200      => Tamaño de la Población Local del Coordinador.
PROBABILIDAD_CRUZAMIENTO (%)  : 90.000   => Probabilidad de cruzamiento considerada por cada AG.(considera
                                          3 decimales)
PROBABILIDAD_MUTACION (%)     : 10.000   => Probabilidad de mutación considerada por cada AG. (considera 3
                                          decimales)
MS_CALIDAD (%)                : 88.842   => Calidad de la solución encontrada. Considera un 85% del
                                          porcentaje del Área Ocupada de la lámina y un 15% del valor
                                          de Unificación de las pérdidas. Mientras menos pérdidas se
                                          generen más unificadas están las pérdidas (considera 3
                                          decimales)
MS_FITNESS                    : 141.000  => Fitness del mejor individuos de la población del Coordinador.
                                          (en unidades cuadradas y considera 3 decimales)
MS_PERDIDA                    : 141.000  => Pérdida del mejor individuos de la población del Coordinador.
                                          (en unidades cuadradas y considera 3 decimales)
MS_AREA_OCUPADA (%)           : 95.696   => Porcentaje del Área Ocupada de la lámina. (considera 3
                                          decimales)
MS_NUMERO_PIEZAS_COLOCADAS    : 6        => Número de piezas colocadas en la lámina.
MS_NUMERO_PERDIDAS_GENERADAS  : 2        => Número de pérdidas internas generadas.
AG QUE PROPORCIONA MEJOR IND. : 1        => AG que proporciona mejor individuo al Coordinador.
TIEMPO_CPU (Segundos)         : 5.785    => Tiempo de CPU total transcurrido. (considera 3 decimales)
TIEMPO_COMM (Segundos)        : 0.011    => Tiempo de Comunicación total generado entre todos los AG y el
                                          Coordinador. (considera 3 decimales)
TIEMPO_TOTAL (Segundos)       : 5.795    => Tiempo de CPU más Tiempo de Comunicación. (considera 3
                                          decimales)
NUM_INDIVIDUOS_A_ENVIAR       : 30       => Número de individuos a enviar desde cada AG al Coordinador.
NUM_INDIVIDUOS_A_RECIBIR      : 30       => Número de individuos a recibir en cada AG desde el Coordinador.
TASA_MIGRACION(EN GENERACIONES): 25       => Número de generaciones a esperar por cada AG antes de que se
                                          realice el Proceso de Migración.
MODELO_MIGRACION              : ASINCRONO=> Modelo de Migración utilizado.
TOTAL PIEZAS+PERDIDAS HECHAS  : 9        => Número de Piezas incluidas en la lámina más pérdidas internas y
                                          pérdidas externas generadas.
@       => Carácter que permite separar la información de los parámetros anteriores y la tabla de valores
                                          que a continuación se presenta, la cual representa el layout
                                          final del problema.

TIPO_PIEZA   POS_X_INI     TIPO_PIEZA        POS_Y_INI        POS_X_FIN        POS_Y_FIN     ESTADO_PIEZA
                                          AREA_OCUPADA
 0           0            61               52               63                E             104
 0          36            59               52               60                P              16
      15          36              54               52               59               G              80
      15          36              49               52               54               G              80
      15          36              44               52               49               G              80
       0          31              60               52               61               P              21
     -15          31              44               36               60               G              80
      11           0              44               31               61               G             527
       4           0               0               52               44               G            2288
```

Figura N°A.3: Archivo de salida del Coordinador que representa el “*layout*” final generado.

Se ha desarrollado una aplicación en C++, bajo Linux, que permite visualizar cualquier "*layout*" resuelto mediante el programa que utilice Algoritmos Genéticos Paralelos. La Figura N°A.4 muestra la ventana que despliega la aplicación, ésta considera como ejemplo el archivo de texto mostrado en la Figura N°A.3. A la derecha de la ventana se muestra el conjunto de parámetros ubicados arriba del carácter "@" en la Figura N°A.3. A la izquierda de la ventana se ubica el "*layout*" resultante al interpretar los datos de la tabla de valores ubicada por debajo del carácter "@" de la Figura N°A.3. Cada pieza muestra en su centro el tipo de pieza a que corresponde, desplegando en color amarillo aquellas piezas sin rotación, y en color rojo aquellas piezas rotadas. Las pérdidas internas se muestran de color verde, mientras que se utiliza un entramado gris para las pérdidas externas.

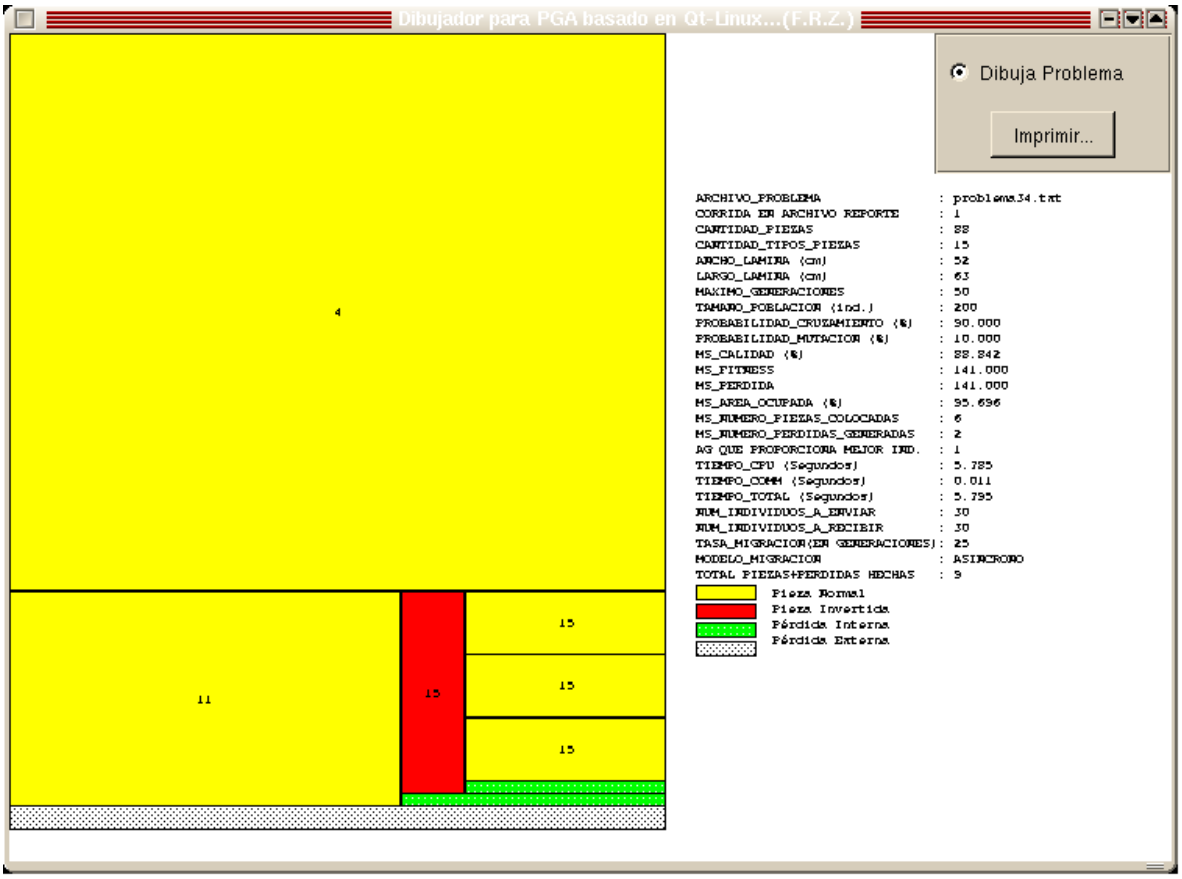


Figura N°A.4: Visualización de "*layout*" de un problema de prueba.

- **Archivo de texto de los individuos que recibe el Coordinador.**

  El siguiente archivo muestra los parámetros e individuos que maneja el Coordinador durante todo el proceso evolutivo. Inicialmente, se muestran los parámetros globales del Coordinador, tales como, número de individuos a enviar desde cada AG al Coordinador, número de individuos a enviar desde Coordinador a cada AG, tasa de migración, cantidad de AG que participan en el proceso, cantidad de individuos en Población del Coordinador, etc. Luego, se muestra, por cada etapa de migración, los individuos (cromosomas de piezas y de rotación) que recibe el Coordinador de cada AG y la pérdida asociada

a cada cromosoma de unos y ceros. Al final de cada Migración, se indican los datos del mejor individuo encontrado por el Coordinador después de la etapa de Migración. Los datos corresponden a :

- El AG que proporciona el mejor individuo encontrado por el Coordinador en la etapa de migración.
- El conjunto de unos y ceros correspondiente al “Cromosoma de Piezas”.
- El conjunto de unos y ceros correspondiente al “Cromosoma Rotación”.
- La Pérdida asociada a la evaluación del “Cromosoma de Piezas” y del “Cromosoma Rotación”.
- El porcentaje de pérdida asociada a la evaluación del “Cromosoma de Piezas” y del “Cromosoma Rotación”.
- El área ocupada por el conjunto de piezas.
- La cantidad de piezas incluidas en el patrón de corte.

El archivo muestra que en la medida que va ocurriendo el proceso evolutivo y las migraciones de individuos, la pérdida de los individuos que

recibe el Coordinador va disminuyendo. Esto indica que el proceso va convergiendo hacia el óptimo. Es importante notar que los individuos del “Cromosoma de Piezas”, en la medida que ocurren las migraciones, van transformándose en cadenas de unos más que de ceros; hasta que al encontrar el óptimo, el individuo asociado posee sólo unos. Lo anterior, se debe a que este ejemplo corresponde a la resolución del Problema1002.txt, el cual posee óptimo cero y todas las piezas caben dentro de la lámina. El “Cromosoma Rotación” del mejor individuo posee una combinación de unos y ceros ya que sólo entrega información referente a la rotación de las piezas en el “*layout*” final.

Archivo de texto de los individuos que recibe el Coordinador

```
Parámetros AGP para el PCPGBR
Desarrollado por Fernando Romero Z.
Magíster en Ingeniería Informática U S A C H - 2003.
-----------------------------------------------------------------------
Nombre del archivo de piezas a utilizar                  = problema1002.txt
Tamaño total de la Población de cada AG                  =   200
Largo de cada Cromosoma                                  =   109
Número Máx. de generaciones                              =   150
Probabilidad de Cruzamiento                              =    90,000 (%)
Probabilidad de Mutación                                 =    10,000 (%)
Número de Individuos a enviar desde cada AG al Coordinador=    10
Número de Individuos a enviar desde Coordinador a cada AG =    10
Tasa de Migración (en generaciones)                      =    25
Cantidad de AG que participan en Proceso                 =     5
Cantidad de Individuos en Población del Coordinador      =    50
-----------------------------------------------------------------------
Cantidad de tipos de piezas distintas                    =    24
Número de piezas del problema                            =   109
Lámina con dimensiones 260 (Alto) x 350 (Ancho)          = 91000 (unidades cuadradas)
```

Reporte de los individuos que maneja el Coordinator

MIGRACION NRO. 1

AG	CROMOSOMA DE PIEZAS	PÉRDIDA
1- 1)	1011111110111011110111101111111111111001110000101101001111010110001100110000001101111100011001100000101111110 0	27000,000
1- 2)	1100111011101111110101110011110111110001111010111111111011101111000110100001110111101100000110101001110010 11	27100,000
1- 3)	1011111110111011110111110110111111001101000111110000110111000011101111101111110111001010000110010100001100001	27100,000
1- 4)	1110110111101111110111101011110111111111000101101101001100111010001101010000111010111010101000101000011010111 0	27800,000
1- 5)	0110111011011011111110011010101110101111111111011111011111111011000011101111011001010110000100100010111011000	27800,000
1- 6)	1110011011101111110101110011110111110001111010110101001011111110110110001100111111011101010100011111101010110 01	28000,000
1- 7)	1111011110111011100011101011110111111111110011110101101000111001101111010011011110111010101000001101001100101 1	28000,000
1- 8)	1011111110111011110111101101111111001101000111011010011001110100011010100011001110100011101110110010011111011	28100,000
1- 9)	1110011011101111110101110011110111110001111010110101001011111110110000111011110110010101101011011111110100101 11	28300,000
1- 10)	1011111110111011110111101001010111111110110101100111100101100111000000010010110100111000011100111111101010 1	28600,000
2- 1)	1111111110111101100111001111111111111010010110111011100111100110001100110100011101000111011001000100001111101 0	25700,000
2- 2)	1111111110111101100111001111111111111010010110111011100111100110001100110100011101000111000101011010001000011 1	25800,000
2- 3)	1101111110111101100111001111111111111010010110111011100111100110001101101111000001011001101000111001001111010 0	27100,000
2- 4)	1011111110111101100111001111111101001111001100111001101001111111011101010101101011101001011010111001111111010	27300,000
2- 5)	1111001110110110111111001111111101001111001100101101101011111101100111100011010110111011101001101111100001110	27300,000
2- 6)	1111101101111101010111001111111101001111001100101101101111110110001100110100011111110111101110010000011110010 1	27400,000
2- 7)	1111001110110110111111001111111101001111001011101101100101111000111011110011001100011101111111111010001000011 1	27600,000
2- 8)	1111001110110110111111001111111101001111001001110111001111001100011011011110001101111111110111011011110000100	27600,000
2- 9)	1111001110110110111111001111111101001111001011101101100101111101101000100101000110111111111011101101111000111 1	27700,000
2- 10)	1011111110111101100111001111111111111010010101111111011100110011101010010010000001011001101001010111110000111 0	27800,000
3- 1)	1111111110111111101110110011101111110110111100011011101011111000011111001010100110101000110001011101000101 00	22900,000
3- 2)	0111111110111111101110110011101111110110111100100110110001111101111010111111100111100111000011111110110100 01	23100,000
3- 3)	0111111110111111101110110011101111110110111100011011101011111000111110110010111100110011101011011101100110 11	23200,000
3- 4)	0111111110111111101110110011101111110110111100011011101011111000111110110010111100110011101011011101100110 10	23300,000
3- 5)	0111111110111111101110110011101111110110111100011011101011111000111110110010111100110011101011011101100110 10	23300,000
3- 6)	0111111110111111101110110011101111110110111100011011101011111000111110110010111100110011101001100010111110 11	23500,000
3- 7)	0111111110111111101110110011111111100111100001111010110111101111111110110101011010011110000000101101100101 00	23500,000
3- 8)	1111111110111110011111011100111111001111111110111001101101111011111111101100111000000001110000111111100101 100	23900,000
3- 9)	1101111110111111101110111011110101011001000110001111101111110110111101011111110011110011100001111111011010 001	24100,000
3- 10)	0111111110111111101110110011101111110110111100011011101011111000011111100011100111101101001010000000001111 10	24500,000
4- 1)	0111111111110101110111111011101100101100110110111111101011010000010111100110111111101011011100000010000011 001	26100,000
4- 2)	1011111101111011011111101011101100111100110110111111111100000100000111100110111111101011010111100000010100 00	27000,000
4- 3)	1101111111111100101011011001110101110100001110111111111011000111111110010101111110000111100110010010111001 011	27400,000
4- 4)	1101111111110101010011001111111010100110101100111111111011000111111111010011111111001011000100110101010111 100	27400,000
4- 5)	0111011111110111011111111011101100101100110110111111101011010000010111100110111110000000000101110111110110 1011	27400,000
4- 6)	1111111110111001010011111011101100101000111111010010111101101111111011001101111110000000000101110111110011 11001	28100,000
4- 7)	1111111110111001010011111011110100110110011111010010111101000111111110001111010001110011010000111111000011 00	28100,000
4- 8)	0111111110111001010011001111111010100110101100111111111011000111111111010011111111001011000100110101010111 100	29400,000
4- 9)	1110011111111011011101001101110101110000101111000110111111000111111110001100111110101010100110111001010110 000	30100,000
4- 10)	0111011111110111011111111011101100100101010110011011111100000100000110011101011101011110011100100001011001 111	30200,000
5- 1)	0111111111001110111101101101011111111010001101111010111111101101011101000111101011100111011011010000111110 000	26100,000
5- 2)	1111101001100111111110110011101011100111010100111110110011101110000110100110110111110111001010011101110100 010	30200,000
5- 3)	1110011111001110111011111101101101010001110111111000110101000111001111010111011100111010000100010011110110 011	30500,000
5- 4)	0011111101011101110001110011111111110101010111111000110110100111101001111001111110111110001001010011001111 100	30700,000
5- 5)	1111011111100111111101011000001101011111010100111110110101000111001110000111001010110011011110011101011000 100	30700,000
5- 6)	1110011111011101110100011101111110111111010111001110101000000000111110001001011001111100111100111100110010 11001	31000,000
5- 7)	1110101111011101110100011101111101010001110111111001110010110111000011010011011011111011110000110100101100 0101	31200,000
5- 8)	1101110011011010111111011011101111000110100111100100110010011110011101111001111110111110001000011010110101 000	31400,000
5- 9)	1110011111011101110100011101111110111111010111001110101000000000111110001001011001111100111100110111000000 01000	31500,000
5- 10)	0011111100100111011111000110110011110111110111111101100101110101111101110010111000100111101100111101101010 010	32000,000

AG	CROMOSOMA ROTACIÓN
1- 1)	0010101111101000111011110110000100110110110000111101000110110010101111000111011100001000111110010100010001100
1- 2)	1010010100001011100001101111111001100101011000100001110111101101011100000010110001000011000000101001110001111 00
1- 3)	0010101111101000111011100100110010101100110110010101001111101111110100001001011101100010000110101110001001110 0
1- 4)	1100011100001011100101010111111011111000101000010001101010011010110101111100101000100001010101101011000111001
1- 5)	0100010100100101001001101101010101000011100000010000100110100000001001000110011101111001100010010001011011011 01
1- 6)	1000000100001011100001101111111001100101011101010011011101000000001010001111000000100001010101010111001011101 00
1- 7)	1110111111101000100100101111111011111000010000010100011101111110010101100001100000010000101000100011011010101 0
1- 8)	0010101111101000111011100100110010101100110110010001101010011010101010000110100001110000101110000001011011000 0
1- 9)	1000000100001011100001101111111001100101011101010011011101000000001001000110011101111001111011011000111001100 01
1- 10)	0010101111101000111011101111001100111000011100011001110110001010110001001010000101101010110110110110001010001
2- 1)	0000111001100110000110110001101010011111010110000011010011100000110000111100101101100101100101010001111111100
2- 2)	0000111001100110000110110001101010011111010110000011010011100000110000111100101101100101001111010001111100000
2- 3)	0110011001100110000110110001101010011111010110000011010011101011101111010101011010000100001111010001011111111
2- 4)	0000011001100110000100110000111011100000000110100101011111010000000010110110110000011101001100100000011111011 001
2- 5)	0000000000101111110000110000111011100001001101100100011110001110001011111001111110011101000110010101111100111
2- 6)	0011100100000110101000110000111011100000000110110010000000101100001100001111000011011111110100011000111000000 11
2- 7)	0000000000010111111000011000011101110000000010101111111000110101100101000000100100010000011110011011000111110 0000
2- 8)	0000000000010111111000011000011101110000000010100000110100111010111011110101010011010101000010000110011000110 11
2- 9)	0000000000010111111000011000011101110000000010101111111000110101000110111001111001101010100001000011001100011 110
2- 10)	0000011001100110000110110001101010011111011100001100111110100110110011001111011010000100001100010101111100111
3- 1)	1001001000110100011110101101010010110100101100000110000100001001110101000011110011110100101101101000111101110
3- 2)	1001001000110100011110101101010010110100101101110100001000100110000011011001100100011110000001010000011100100 0
3- 3)	1001001000110100011110101101010010110100101100000110000100001001100011000010100111101011111010001101000100111
3- 4)	1001001000110100011110101101010010110100101100000110000100001001100011000010100111101011111010001101000100111
3- 5)	1001001000110100011110101101010010110100101100000110000100001001100011000010100111101011111010001101000100111
3- 6)	1001001000110100011110101101010010110100101100000110000100001001100011000010100111101011111010111110101000001
3- 7)	1001001000110100011110101101010100110111101111001101001011001100100011000001110000101110001110011101000110110
3- 8)	1001001000110101111100100110110100011111001000000001100101100110010001100011111010100011000000101000000010011111
3- 9)	1100001000110100011110100100100001001011010101000111011110110110000011011001100100011110000001010000011100100 0
3- 10)	1001001000110100011110101101010010110100101100000110000100001001110100100110000111010111011011000000101110110 1
4- 1)	0010001001011101100010000000111111101110010111011010100111111010000010011001101101101000101001001111100101001 1
4- 2)	0010011011110010101010010101111111110010001011100111001001000110100110101001101101101000101111111011100100000 0
4- 3)	1000001001101000110111010011011010100001001110110101011000101010011100011101100011000111100110010111111011100
4- 4)	1000001001011101111011010000010011011111111001110101011000101010011100011101111000010101101100100100011011101
4- 5)	1110111011110110101010000000111111101110010111011010100111111010000010011001101001100001011111110010011011000 11
4- 6)	0011010000000110111101000000111111101101000010111000000100110011000011110101111000110000101111111001000001110000 1
4- 7)	0011010000000110111101000000110000110101000010111000000100101000100111000100100110001100000000101011010101010000
4- 8)	1111010000000110111101101000000100110111111111001110101011000101010011100011101111000010101101100100100011011101
4- 9)	1010000001110010001010100011101101010001001000111111111100010101001110001001000110110101000101110110001010011 1
4- 10)	1110111011110110101010000000111111101011101110000001100100100011010011000011111000100110001000000110000110011110
5- 1)	0011100100011110101101101000110100000110100101101111101010111110011000100100100111011000100000110000110000001 1
5- 2)	1110010111010111010000000101011101111010000110011101001010000110111000101111111000000110001010001101111111001
5- 3)	1011010011010011110010101101100111100001010011111000111100011111110111110001001111000010111100010010100101010
5- 4)	0101100011010010110010000110100111001010110011111000000001100110100110011111011110011100100000001011101001010 11
5- 5)	1000111100011100101100111100100111101010000110011101111110001111111111100000101011101001111110001101111111011 0
5- 6)	1011010011010010110011111101000101000111000001111101101100111000101011100111110111011111010010000110110001101
5- 7)	1101001100010010110011111101000111100001010011110001001000000110111000101111111000000101001010010010110111111 1
5- 8)	0111011001100000000010110100000111101010110110001100001010011111000100111110111100111001000011100100001010001
5- 9)	1011010011010010110011111101000101000111000001111101101100111000101011100111110111011111010010100110100100000
5- 10)	1110111111011111011110110010101110000000001101001011001110000000101000110101010011011101011001110110110010110111

DATOS DEL MEJOR INDIVIDUO ENCONTRADO POR EL COORDINADOR

AG que proporciona el Mejor Individuo	: 3
Cromosoma de Piezas	: 111111111011111110111011001110111111011011110001101110101111100001111100101010011010100011000101110100010100
Cromosoma Rotación	: 1001001000110100011110101101010010110100101100000110000100001001110101000011110011110100101101101000111101110
Pérdida resultante	: 22900,000 (Unidades Cuadradas)
Porcentaje de Pérdida resultante	: 25,165 (%)
Area Ocupada por las piezas	: 74,835 (%)
Cantidad de Piezas incluidas	: 69

```
-------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------  MIGRACION NRO.    2  -----------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
AG        CROMOSOMA DE PIEZAS                                                                                         PÉRDIDA
-------------------------------------------------------------------------------------------------------------------------------
 1-  1)  1011111111101111111111100111111111110101110111011111011111111010101110001011110110010101010101111101111000101  18200,000
 1-  2)  1111111110111111101101101111111111101101011011111100111111101001110110110001111101100110101011111111110101100  18400,000
 1-  3)  1111111110111111101101101111111111101101011011111100111111101001110110110001111101111111001000101111100101000  18600,000
 1-  4)  1111111111101111111111100111111111110101110011010010111111111011110010010010000101110101100111111110101100011  18700,000
 1-  5)  1111111110111011111110111101011111101001011011101111111111111010101110001011110110011101100111111110101010011  18900,000
 1-  6)  1111111111111110011111101111011111111010001110101110111111100011110111010001101110101010110010100111111010011  18900,000
 1-  7)  1111111111101111110100111111111111101100100111011111011111111010101110001011110101110110010101100110101010110  19300,000
 1-  8)  1111111110111111101101101111111111101101011011111100111111100011110110110000111011011001010000111011111100011  19300,000
 1-  9)  1011111111101111111111110111111111110101110011010010111111111011110010001011110110010101010101111101110000000  19500,000
 1- 10)  1011111111101111111111011111111111110100100011111110111010111110010111101100001101111010001010111111111110100011  19500,000
 2-  1)  1111101111110111101110111111111111110111111101111101011111010110101111101111111111110111101110001111011000101 1  14900,000
 2-  2)  1111111110111111101101111111111111110110111001110111111010010111111101111001111111011110111001101000100000010  15100,000
 2-  3)  1111111110111011110111101111111111111001110111111010111001111011111011100110111111111110101110001111011011011  15900,000
 2-  4)  1111101111110111101110111111111111110111110111111110111101110100001111111001011110011001110101101110110111011  16600,000
 2-  5)  0111111110111111101101111111111111110011111011111010111001111011111011111001011110110110010111001111111011111  17200,000
 2-  6)  1111111110111111101110111111111111101110011100111011111100101100011101001111100110111110011110011101110111100  17700,000
 2-  7)  1101111110111111101110101111111111111111110110110110111110110011111001111100110111011010001111111111000111  17800,000
 2-  8)  1110111110111110111111101111111111111111111011101011101011111100111001101101111001001101100010010100010110010  17900,000
 2-  9)  1101111110111011110111101111111111111001110111111010111001111011111011100110111111111110101110001111011010011 0  18100,000
 2- 10)  1011111110111011110111101111111111111001110111111110111111010110111011111001011110110110010111001111111111000  18100,000
 3-  1)  0111111110111111111111101111111011111101101111110111011110101011111011110100001101010111111011100101110000110 0  17400,000
 3-  2)  1111111110111111101111001011111111111101101111010011110101111011011111100100111110101011010111111111101100101 0  18000,000
 3-  3)  1111111110111111101111110011110111111011011110100111101011110111011101011100010110011101111111001011010100111  18400,000
 3-  4)  1111111110111111101111011101110111111011011110100111101011110110111111001111000101011010111101101010100111111  18400,000
 3-  5)  1111111110111111101111110011110111111011011110100111101011110111011101011100010110011101111011010111000001111  18600,000
 3-  6)  0111111110111111101111101111110111111011011111101110111101010111011111111111100100000011100010111100011001010  19200,000
 3-  7)  0111111110111111101111011101110111111011011110111111101011110110111100110110110100111111111001100011001010  19200,000
 3-  8)  1111111110111111101111001011111111111101101000011111101101111001001111111111111001101010011101101011110001110 11  19700,000
 3-  9)  1111111110111111101110110011110111111011011110100111101011110110111111001111000101001111111110011000110010 10  19700,000
 3- 10)  0111111110111111101111101111110111111011011111101110111101010111110111101000001011001010001101111110000010111 1  20100,000
 4-  1)  1111111111111111111110110001111011111101101111011111111111001111111111100011111011000100011111100011101111000 00  15700,000
 4-  2)  1111111111111111111011111101110111111011011110111111111110011111111101000110010000100000000101111000000100011 0  16400,000
 4-  3)  1111111111111111111110110001111011111101101111011111111111001111111111100011011111100000000010110111001100100 01  16700,000
 4-  4)  1111111111111111111110110001111011111101101110010111011101011010111111111111101110111001011011110111101111001011  16900,000
 4-  5)  1111111110111111101110110011110111111011011110111111111111001111111111011101011111100111110010100000000010000 00  17500,000
 4-  6)  1111111111111111111110110001111011111101000111111111111111001011111110011111110000111010101010110100000001011 10  17700,000
 4-  7)  1111111111111111110111101111101111011010001110111111111011010001111111011101010110111000000001011101111110110101 1  17900,000
 4-  8)  1111111110111111101110110011110111111011011110111111111111010001111111011101011101110011010111000011101110011 00  18000,000
 4-  9)  1111111111111111111110110001111011111101101111000110111101001111101101011111010111001110101101111111011010110 1  18000,000
 4- 10)  1111111111111111111110101111101111011010001110111111111110011111111010001100100000111100111001111000001000011  18000,000
 5-  1)  1110111101011111011111101111111111111101101011111011011100111101010111000101110011001011001101100011100111101 0  21600,000
 5-  2)  1110111101011111001110101111111111111111011100110111100011101111011110010101011001111111111111111111111111100010  21700,000
 5-  3)  1011111110111101100111101111111111111001110100111101111110011111111011101110010101100111100010011111100001011  21800,000
 5-  4)  1101111110111101101111011111111111111110101010111111011000111011100001010001110110011110110011000001011100010  22500,000
 5-  5)  1110111101011111011111101111111111111101101011111011001101110110010111110101001011000001100011000011111111000  22600,000
 5-  6)  1011111110111101100111101111111111111001110100111101101011111101001111101001110101010110011011000111101110101  23100,000
 5-  7)  1110111101011111011111101111111111111010010110001011101011101111001111110101000011110111110001101111110100101 0  23200,000
 5-  8)  1110111101011111011111000111111111111011010111110110011011101100101111101010110101111110110101100100110000000  23800,000
 5-  9)  1111111110111011001011101011110101111011010111111111111011000001111110110100000011101111111111111111011000010011  23900,000
 5- 10)  1011111110111101100111101111111111111001110100111101101011111101001111001010011000111110011011001010101011101  23900,000

AG        CROMOSOMA ROTACIÓN
-----------------------------------------------------------------------------------------------------------------
 1-  1)  0010101000000101110110111111000010110100111101011011100010010010101010000100010111000100101101111010111111000
 1-  2)  1001001000110100011110101001101010011101000101000001101100010100101100101011111110001111100101100011101111100
 1-  3)  1001001000110100011110101001101010011101000101000001101100010100101100101011111110000011000011110110101100110
 1-  4)  1000101000000101110110111111000010110100111101000110100110101000000101111101111011011110101000101111000001111 0
 1-  5)  1001001000111100111011000000011101011111010110000001100010010010101010000100010111000011101000101111000101110 0
 1-  6)  1001001000110101111110111010111111011101000110010100010011010111001011011010001100111000010110111111011100111110
 1-  7)  0001001000000101110000100010110101001110001101011011100010010010101010000100010100010011000010011101111001001 1
 1-  8)  1001001000110100011110101001101010011101000101000011010100010000101000011000101000110110010011000010101001110
 1-  9)  0010101000000101110110110111000010110100111101000110100110101000000010100010001011100010010110111101011100101 01
 1- 10)  0010101000000101110111100010110100010101101001000101000010000100100011001101101100111100110010110001110010111 0
 2-  1)  0011100000010000000111101001000000100110010111100001100110110110010000011101001001101111111010001001101001110101
 2-  2)  0001001111101100011110100110000110110100101101001010111101101100000011101000001101111111010001010001111100001
 2-  3)  1111011111100000111011110000000100110110000000100110011111010110001110101001101101101001000011100110100100101 0
 2-  4)  0011100000010000000111101001100001001100000000001100100111100010111110101010010100111101011111010001101101001010
 2-  5)  1001001000110100011110100000111000110110111100001100111101011000111010100101001110110000110110001101110100111
 2-  6)  0001001000110100011011110110000101110011011010110010100111101010110101000100111010001101011001010100101000011
 2-  7)  1010001000110100011110110001101011010000011001000100000001011000101111000010011101000010100111101000011100000 0
 2-  8)  1010111001100111110001110110000100110010111100001110011100011000100101111101110100000000100100101000111101011 1
 2-  9)  1010001000110000111011110000000100110110000000100110011111010110001110101001101101101001000011100110100100111 0
 2- 10)  0010101111101000111011110000000100110110000000110010011110100110101110101001010011101100001101100011000100001 1
 3-  1)  1101001000110100011100100101010010110100101101101010110010100111100101001001110010100001111001111111000010110
 3-  2)  1001001000110100011110111101010010110100101101000011111010110110000001101001011000110100010111010110000110110 1
 3-  3)  0001001001110100111100101101010010110100101100000001111101011010100001101100101011111111100010011111110110101 11
 3-  4)  1001001000110100011100100111010010110100101101000011111010110110000001101011101101010110110101101111011111001 1
 3-  5)  0001001001110100111100101101010010110100101100000001111101011010100001101100101011111111100010000110110101110 11
 3-  6)  1001001000110100111100100101010010110100101101101010110010100110011100110011001001001110000000110001100110110 1
 3-  7)  1001001000110100011100100111010010110100101101001111111010110110000001110111000101001000100011100000110011011 01
 3-  8)  1001001000110100011110111101010010110100101111001011001010100110011100110011001010010000110011101011010011010
 3-  9)  1001001000110100011110101101010010110100101101000011111010110110000001101001100101010000100011100000110011011 01
 3- 10)  1001001000110100111100100101010010110100101101101010110010100111100101001001101011100011100100000110011111110
 4-  1)  0010001001011111000101001101010010110100101100110101011010100100100100010010011000101101011011110001100111110
 4-  2)  0010001001011111110001000000111110011010010110011010101101010010010010001001001001100001011111111011100100111 0
 4-  3)  0010001001011111000101001101010010110100101100110101011010100100100100010011100011000010111111000101100100100 0
 4-  4)  1011001001011111000101001101010010110100101110101110110111100101101101111011100001101001110000000001100001100 1
 4-  5)  1011001000010100011110101101010010110100101100110101011010100100100010111111110001100010011011110010100100100 0
 4-  6)  1011001001011111000101001101010010110101110011110101010011000100010110100010001100110100110010010101011011110
 4-  7)  0010001001010100011100000000100010111010101100110101001101111111101111011111101111110001011111100100110110001 0
 4-  8)  1011001000010100011110101101010010110100101100110101011101111111011110111111100010010001001001010001000011000
 4-  9)  0010001001011111000101001101010010110100101100000111011000000000110010011001100011100101101111000110011100010 1
 4- 10)  0010001001011111000101000000100010111010101100110101011010100100100100010010011010101110111010100011010101010
 5-  1)  1101101110110110111101110101101100001111100100010100001110100010111111001001001010101010010011000111001100100
 5-  2)  1101101110110110110010110110000100001011000110111010001011010101100000000111011101010100001101111110010011110 0
 5-  3)  0110011001100110000110110110000100110110100110011100001010011011001011000100001011000000101110110101100100000 1
 5-  4)  0110011001100111111011110110000100001011100110111101101001100100011111001100011010000001101111001010001001110 0
 5-  5)  1101101110110110111101110101101100001111100100010100110001011110111000000011011100001100000100000001010011101 00
 5-  6)  0110011001100110000110110110000100110110100110011100011110001110111111111110000111001010100100110001011101100 01
 5-  7)  1101101110110110111101110101101010011111010100101101101011010100110111101111000000110011000010011101110000010
 5-  8)  1101101110110110111101101110000100001111100100010100110001011110111000000011111010000101000010101100101010111 0
 5-  9)  1000101111101000000111110100001111111100000001000111010001011110110010101011010011011110000110111111011100011111
 5- 10)  0110011001100110000110110110000100110110100110011100011110001110111111111101000111110001001001100010011001101 0

DATOS DEL MEJOR INDIVIDUO ENCONTRADO POR EL COORDINADOR
-------------------------------------------------------
AG que proporciona el Mejor Individuo          : 2
Cromosoma de Piezas                            : 11111011111101111011101111111111111101111111011111010111110101101011111011111111111110111101110001111011000101 1
Cromosoma Rotación                             : 0011100000010000000111101001000000100110010111100001100110110110010000011101001001101111111010001001101001110101
Pérdida resultante                             : 14900,000 (Unidades Cuadradas)
Porcentaje de Pérdida resultante               :    16,374 (%)
Area Ocupada por las piezas                    :    83,626 (%)
Cantidad de Piezas incluidas                   :    86
-------------------------------------------------------
```

-- MIGRACION NRO. 3 --

AG	CROMOSOMA DE PIEZAS	PÉRDIDA
1- 1)	111111111111111111111111011111111111101011111101111111111111111011111111101100111110111011011101010111000010010	8300,000
1- 2)	111111111111111111111111101111111111011101111011111111111111111011111011011000111110110110000101100011111110000	9800,000
1- 3)	11111111111111111111110111111111111111010101110011101111111111111011111111101100111110111011111111001011010100111	10200,000
1- 4)	111111111111111111111101111111111111110101111110111111111111111110001111011101000110110010101111110110111110111011	10500,000
1- 5)	1111111111111111111101111011111111111101101110011101111111111111011111111101100111110111011111111001011010100111	10600,000
1- 6)	111111111111111111111111011111111111101011100110111111111111111010111111011110011111111101111111001011010100011	10600,000
1- 7)	1111111111111111111101111011111111111101101110011101111111111111011111111101100111110111011111111001011010100111	10600,000
1- 8)	111111111111111111110111101111111111110110111001110111111111111101111111110110011111011101101110111101110011110	10600,000
1- 9)	111111111111111111111011111111111111110110111001111111111010010111111101111001111111011001011101001110011110	10700,000
1- 10)	11111111111111111111111001111111111111010111011111111111111111001111101101100011111011101110111000110101010111	10800,000
2- 1)	11111111111111111111111011111111111111111111110111110101111111011001111110111111111111101111011100011111000110001	8100,000
2- 2)	11111111111111111111111011111111111111111111110111110101111111011001111110111111111111101111011100011110011011000	8200,000
2- 3)	11111111111111111111111011111111111111111111110111110101111111011001111110111111111111101111011100011110011001100	8200,000
2- 4)	11111111111111111111111011111111111111101101110111111101011111011111111100111111111111011110111001111011011000	8500,000
2- 5)	11111111111111111111111011111111111111111111110111110111110111110111110101111001111111000101011111111100011001101	8500,000
2- 6)	111111111111111111111110111111111111111111111101111101011111110110011111101111111111111011110111000000101110001011	8500,000
2- 7)	1111111111111111111111101111111111111111111111011111011111011111011111010111100111111101110011110010001010100111	8500,000
2- 8)	111111111111111111111110111111111111111111111101111101011111110110011111101111111111111011110111000000101000100111	8600,000
2- 9)	111111111111111111111110111111111111111111111101111101011111110110011111101111111111111011110111000000101110000110	8600,000
2- 10)	111111111111111111111110111111111111111111111101111101011111110110011111101111111111111011110111000000110010000110	8700,000
3- 1)	11111111111111111111011111111111111101111111011111010111110101101011111111101001111001111000111110000100100110	11600,000
3- 2)	1111111111111111111101110111011111110110101110111111111110011111111011111101110100000111110011111101111001111	11900,000
3- 3)	11111111111111111111011111111111111101111111011111010111110101110110010110010100111111110111010101100011111010	12100,000
3- 4)	0111111111111111111101111111111111110111111101111111110111011101111111001001111101010110011010111100110111000	12400,000
3- 5)	111111111111111111111111101111011110111111101111101011110011111111010001111100010100111110011111101111001111	12400,000
3- 6)	1111111111111111111101110111011111110110101110111111111110011111111010111111100101001011111011100100100011001	12600,000
3- 7)	1111111111111111111101110111011111110110101110111111111110011111111010111111100101001011111011100100100010001	12700,000
3- 8)	01111111111111111111011111111111111101111111011111010111110111111111010111111100101001011111011100100101011110	12700,000
3- 9)	01111111111111111111011111111111111101111111011111010111110111111111010111111100101001011111011100100101011100	12800,000
3- 10)	111111111111111111110111011111111111101101110111111101011110110111111001111100010110001010101110000111110111	12900,000
4- 1)	111111111111111111111110111111111111111111111111011111111111111111101111101111111010111110011001000100001110101101	7500,000
4- 2)	111111111111111111011110111111111111111111111111011111111111111000111111001111111011111111011110011010110111111101	8500,000
4- 3)	1111111111111111111111101111111111111111111011111111111111111100011111100111111101100111010101011100001111000110	8700,000
4- 4)	1111111111111111110111101111111111111111111011111111111111111100011111111000111111111101011010101101111111111101	8900,000
4- 5)	111111111111111111011110111111111111111111111111011111111111111000111111001111111011111111011110011010110111111101	8900,000
4- 6)	1111111111111111111111101111111111111111011101111111101111101101011111111111101111011101010011100111111111001100	9100,000
4- 7)	11111111111111111111111011111111111111101101111111111111111110010111111001111011111110101111010010010110101100001	9400,000
4- 8)	11111111111111111111111011111111111111101101111111111111111110010111111001111011111110101111010010010110101100001	9400,000
4- 9)	11111111111111111101111011111111111111111111111101111111101111101011111011101011111110111111001010001101100000101	9500,000
4- 10)	111111111111111111011110111111111111111111111111011111111111111000111111110001111111111101011010101101101000101100	9500,000
5- 1)	111111111111111111011111101111111111011111110111111111111111011111011111101111111111111111111111111111111001011	6600,000
5- 2)	111111111111111111111110111111111111101111111011111011111010011111111111110111111111111111111111111111111001011	6700,000
5- 3)	111111111111111111011111101111111111111111111101111101111001111111111011011111111111111111111111111111110101110	6800,000
5- 4)	111111111111111111111110111111111111101110111111111111111100111110111100101011111111111111111111111111111010001	7400,000
5- 5)	111111111111111111011111111111111111101110111111111111111100111110111100101011111111111111111111111111111010011	7400,000
5- 6)	11111111101111111111111111111111111111011101101111111111111010011111111111101111111111111101111011101001011110101110	8500,000
5- 7)	111111111111111111011111111111111111111111101101111101011110011111111111101101011001111111111111111111111110001011	8600,000
5- 8)	1111111111111111110111111011111111111111111111011111011110101011111011110101111111111111111111111011111000001000	8600,000
5- 9)	111111111111111111011111101111111111011111110111111111111111001111101111001010111111111111111111111111111010011	8600,000
5- 10)	11111111111111111111111111011111111110111111111101111111010100110110111110111111111111101111011100011110110001011	9000,000

AG	CROMOSOMA ROTACIÓN
1- 1)	1001001000011111101101110101101001101001101100110110100100100100111101000111110000001110100110011101010100
1- 2)	1001001000111011101100000001100101101000101100110101000100100100111100101011111110000110011101110101101100000
1- 3)	0010001001010000111011000101101001101001101101001010100100100100100111101000111110000000001001111111011010111
1- 4)	0010001001010000111011000101101001101001101100110101001101011100101101101000111110001010010011111010001001010
1- 5)	1010001001011111100001101110000110110100101101001010100100100100100111101000111110000000001001111111011010111
1- 6)	1001001000011111101101110101101001101101111010110111011101010100000011101000001101101110001001111111011011110
1- 7)	0010001001011111100001101110000110110100101101001010100100100100100111101000111110000000001001111111011010111
1- 8)	1010001001011111100001101110000110110100101101001010100100100100100111101000111110000001110100100111000011111
1- 9)	00100010110111111000100100010000101100100101101000100111101101100100011101000001101100110000101101111000011111
1- 10)	1010001000111011101101111110000110110100101100110101001101011100100100101011111110000000110111111011110010011
2- 1)	001000100101111110001011101100001001100101111000011001100101100010100111010010011011111111010001001100001101000
2- 2)	001000100101111110001011101100001001100101111000011001100101100010100111010010011011111111010001001100011010001
2- 3)	001000100101111110001011101100001001100101111000011001100101100010100111010010011011111111010001001100001101101
2- 4)	0010001001111111100000001001000001101101001011001101100111001011000100110101110011011111111000001011101111010001
2- 5)	101100100101111110001011101100001001100101111000011101101001011000111111010000011011111100101110101000010101011
2- 6)	001000100101111110001011101100001001100101111000011001100101100010100111010010011011111111011100010010001110101
2- 7)	1011001001111111100000011101100001001100101111000011101101001011000111111010000011011111111011001011111101011100
2- 8)	0010001000011111100011111101100001001100101111000011001100101100010100111010010011011111111011100010010001011100
2- 9)	001000100101111110001011101100001001100101111000011001100101100010100111010010011011111111011100010010001100111
2- 10)	001000100101111110001011101100001001100101111000011001100101100010100111010010011011111111011100010011101100111
3- 1)	0010001001011111100010110010000001001100101111000011001101101100100000101100100001011111110100110010010100001001
3- 2)	0010001001011111100010100000011111001101001011001101010110101001011011011110111000010000001000011100101111101010
3- 3)	001000100101111110001011001000000100110010111100001100110110110101101101101011111111000110000001111001100000111
3- 4)	111100100101111110001011001000000100110010111100001111011110101011100101010010110001101000111110000111011110011
3- 5)	01110010010111111000010000101010000110010111100011100110010100100100100010011001010100001000011100101111101010
3- 6)	001000100101111110001010000001111100110100101100110101011010100100100100110011001011011100100000000001000101111
3- 7)	001000100101111110001010000001111100110100101100110101011010100100100100110011001011011100100000000001000100111
3- 8)	111100100101111110001011001000000100110010111100001100110110100100100100110011001011011100100000000001001000110
3- 9)	111100100101111110001011001000000100110010111100001100110110100100100100110011001011011100100000000001001000100
3- 10)	001000100101111110001010000001110010110100101101001111111010110110000011010011001010110011101000010001011101010
4- 1)	101100100101111110001111100000000100110010101100110101011000100110101010110010000011111111101110111010011000100
4- 2)	10110010010111111011011110110000100110010101100110101011000111110010110110010001010101001011101011001011110000
4- 3)	00100010010111111000111110110000101110000010011110101011000111110010110110010011000010100110110111010001101110
4- 4)	00100010010111111011011110110000111010000010011110101011000101100100100010001100011111011100101111110001110000
4- 5)	00010010010111111011011110110000100110010101100110101011000111110010110110010001010101001011101011001011110000
4- 6)	001000100101111110001111100000000110100110000001100100111101100101101101111011100111111011001001100100000011000
4- 7)	10110010010111111000111110110000101110100101101110101010011000100010110100000011010100111010001010000010100110
4- 8)	00100010010111111000111110110000101110100101101110101010011000100010110100000011010100111010001010000010100110
4- 9)	00100010010111111011011110110000100110010101100110101010100101000000010111111100011000100110111110101110000110
4- 10)	001000100101111110110111101100001001100101011001101010110001011001001000100011000111110111001011111100011011111
5- 1)	10110010010111111000100000011001001100101111000101010101101101011000111010010011011110000110111111100100000010
5- 2)	001000100101010001110010011000010111001011110000110101100000000110000111010010011011110000110111111100100000010
5- 3)	1011001001011111100010000110000100110010101100001010111010101011001111010010011011010000110111111100101011110
5- 4)	10110010010111111011100100101100100110000101101000000101101010110110000000011110011011010000110111111100100011101
5- 5)	1011001001011111100010000101100100110000101101000000101101010110110000000011110011011010000110111111100100011111
5- 6)	101100100001111110001100001100001001101100001010001010110000000011000011101001001101111111010001111111000011110
5- 7)	001000100101111110001000011000010011001011110000110011011000000110000111011101110101010000110111111100001110101
5- 8)	1011001001011111100010000110000100110010101100001010110010100111100101001001001101101000011010011111100001000
5- 9)	001000100101111110001000000110010011001011110000000101101010110110000000011110011011010000110111111100100011111
5- 10)	0010001001011111100011000000010001001100101011001100010010000000010000011101001001101111111010001001101001110101

DATOS DEL MEJOR INDIVIDUO ENCONTRADO POR EL COORDINADOR

AG que proporciona el Mejor Individuo : 5
Cromosoma de Piezas : 111111111111111111011111101111111111011111110111111111111111011111011111101111111111111111111111111111111001011
Cromosoma Rotación : 10110010010111111000100000011001001100101111000101010101101101011000111010010011011110000110111111100100000010
Pérdida resultante : 6600,000 (Unidades Cuadradas)
Porcentaje de Pérdida resultante : 7,253 (%)
Area Ocupada por las piezas : 92,747 (%)
Cantidad de Piezas incluidas : 99

MIGRACION NRO. 4

AG	CROMOSOMA DE PIEZAS	PÉRDIDA
1- 1)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010011	500,000
1- 2)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010110	500,000
1- 3)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101110	700,000
1- 4)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101110	700,000
1- 5)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010111	700,000
1- 6)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010110	800,000
1- 7)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010111	800,000
1- 8)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010110	800,000
1- 9)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010011	800,000
1- 10)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111001110	800,000
2- 1)	11111111111111111111111111111111111111111111111111011111111111111111010111111011111111111110111111111111111111110000111	3400,000
2- 2)	11111111111111111111111111111111111111111111111111011111111111111111001111111111111111111111111111111111111111111010110	3400,000
2- 3)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111010100100	3600,000
2- 4)	11111111111111111111111111111111111111111111111111011111111111111111010111111011111111111111111111111111111111011011000	3600,000
2- 5)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111010100100	3600,000
2- 6)	11111111111111111111111111111111111111111111111111011111111111111111010111111011111111111101111011111111111111100111000	3600,000
2- 7)	11111111111111111111111111111111111111111111111111001111111111111111001111111011111111111110111111111111111111111111011000	3700,000
2- 8)	11111111111111111111111111111111111111111111111111011111111111111111011101100111111111111111011111111111111111111001001000	3700,000
2- 9)	11111111111111111111111111111111111111111111111111011111111111111111010111111011111111111110111111111111111111111001001110	3800,000
2- 10)	11111111111111111111111111111111111111111111111111011111111111111111010111111011111111111110111111111111111111111001001100	3900,000
3- 1)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001001011	2200,000
3- 2)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001001011	2200,000
3- 3)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001110101	2300,000
3- 4)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001001000	2400,000
3- 5)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111101111101111111111111110011011	2400,000
3- 6)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001001011	2500,000
3- 7)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001011011	2500,000
3- 8)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001001011	2500,000
3- 9)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001001011	2600,000
3- 10)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001101101	2600,000
4- 1)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001001010	1500,000
4- 2)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001001110	1500,000
4- 3)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001001110	1500,000
4- 4)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001001010	1500,000
4- 5)	11111111111111111111111111111111111111111111111111011111111111111111011111111011111111111110111111111111111111111001001010	1500,000
4- 6)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001001110	1500,000
4- 7)	11111111111111111111111111111111111111111111111111011111111111111111011111011111111111111111111111111111111111001001110	1500,000
4- 8)	11111111111111111111111111111111111111111111111111011111011111111111011111111011111111111110111111111111111111111001001010	1600,000
4- 9)	11111111111111111111111111111111111111111111111111011111011111111111011111111011111111111110111111111111111111111001001011	1600,000
4- 10)	11111111111111111111111111111111111111111111111111011111011111111111011111011111111111111111111111111111111111001001011	1700,000
5- 1)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001111	2800,000
5- 2)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001111	2800,000
5- 3)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001111	2800,000
5- 4)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001111	2800,000
5- 5)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001111	2800,000
5- 6)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001111	2800,000
5- 7)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001011	2900,000
5- 8)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001011	2900,000
5- 9)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001011	2900,000
5- 10)	11111111111111111111111111111111111111111111111111011111111111111111001111111011111111111110111111111111111111111001001011	2900,000

AG	CROMOSOMA ROTACIÓN
1- 1)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100011111
1- 2)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100011000
1- 3)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100011110
1- 4)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100011110
1- 5)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100001101
1- 6)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100101101
1- 7)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100101101
1- 8)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100101101
1- 9)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100011111
1- 10)	10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100001101
2- 1)	00100010011111110001000001010001001100101111000011101001001010100111010010011011100001101111100101101100
2- 2)	00100010010111110000100001010101001100101111000000101101010110000100111010010011011011000110111111001010111100
2- 3)	00100010011111110001000001010101011100100111000000001100101011001000111010010011011000011011111001010101100
2- 4)	00100010010111110001111001000001001100101111000011101001001001010011101001001101101000011011111001010100001
2- 5)	00100010011111110001000001010101011100100111000000001100101011001000111010010011011000011011111001010101000
2- 6)	00100010011111110001000001000010011001010110100000011001001001010011101001001101111001110111111001010111100
2- 7)	00100010000111110000100001000001001100000101100000101101010110001001110100100111011000011011111001010100000
2- 8)	00100010011111110001000001010001001100101111010000001111011000101001110100100111011010000011011001100001101000
2- 9)	00100010000111110000100001010101001100101111000011101001001001010011101001001101111000011011100010000110111
2- 10)	00100010000111110000100001010101001100101111000011101001001001010011101001001101111000011011001000011011101
3- 1)	00100010011111110001010001011000101110000010010000001011010110011000101111001011110111000011011111001000000010
3- 2)	00100010011111110001010001011001011100000100100000010110101100110010111100101111011100001101111110010000010
3- 3)	00010010011111110001010001011001011100000100110111011110001001101100111010010011011100001101111110010000011
3- 4)	10110010010111110001010001011001001100001011010000001011010110011000101111001011110100001101111110010000010
3- 5)	10110010011111110001010001011001011100001011010000011110001001110010111010010011011111101001111110010000010
3- 6)	00100010010111110001010001011001011100001111000110111110001001101010101100010011011100001101110100010000010
3- 7)	10110010010111110001010001011001011100000100111010101101100100101111001000001111100001011111110010000010
3- 8)	00010010010111110001010001011001011100000100110111011110101100100101111001000001111100001011111110010000010
3- 9)	10110010010111110001010001011001011100001011010000111010110011000101111001000001111100001011111110010000010
3- 10)	00100010011111110001010001011001011100000100100000010110101100110010111100100000111100001011111110010000011
4- 1)	10110010010111110001111001000001001100101111010101101100011000111001100101111010001101111110010011111110000110
4- 2)	10110010010111110001111001000001001100101100111101010101101101011100001100110101111010011011111100101011110
4- 3)	10110010010111110001111001000001001100101100111101010101101101011001000100110101111010011011111100101011110
4- 4)	10110010010111110001111001000001001100101100111101010101100010100000011101001001101111010011011111100100000100
4- 5)	10110010010111110001111001000001001100101100111101010101101101000001000100110111101001101111110010101111110
4- 6)	10110010010111110001111001000001001100101100111101010101101101000001000100110111101001101111110010101111110
4- 7)	10110010010111110001111001000001001100101100111101010101101101000001000100110111101001101111110010101111110
4- 8)	10110010010111110001111001000001001100101100111101010101101101011100001100110101111010011011111100101011110
4- 9)	10110010010111110001111001000001001100101100111101010101101101000001000100010111101100101111110010000010101
4- 10)	10110010010111110001111001000001001100101100111101010101101101011100001100110101111010011011111100100000101
5- 1)	10110010010111110001011001101001001100101111000001010110000000110000111010010011011100001101111110010000110
5- 2)	10110010010111110001011001000001001100101111000001010110000000110000111010010011011100001101111110010000110
5- 3)	10110010010111110101011001000001001100101111000001010110000000110000111010010011011100001101111110010000110
5- 4)	10110010010111110001100001000001001100101111000001010110000000110000111010010011011100001101111110010000110
5- 5)	10110010010111110001011001101001001100101111000001010110000000110000111010010011011100001101111110010000110
5- 6)	10110010010111110111111001101001001100101111000001010110000000110000111010010011011100001101111110010000110
5- 7)	10110010010111110001011001101001001100101111000001010110000000110000111010010011011100001101111110010000010
5- 8)	10110010010111110001011001101001001100101111000001010110000000110000111010010011011100001101111110010000010
5- 9)	10110010010111110001011001101001001100101111000001010110000000110000111010010010111100001101111110010000010
5- 10)	10110010010111110001011001101001001100101111000001010110000000110000111010010011011010000110111111001000000010

DATOS DEL MEJOR INDIVIDUO ENCONTRADO POR EL COORDINADOR

AG que proporciona el Mejor Individuo	: 1
Cromosoma de Piezas	: 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111010011
Cromosoma Rotación	: 10110010010111110001011001100001001100101111111011001101011000010111010110011010100001101111100100011111
Pérdida resultante	: 500,000 (Unidades Cuadradas)
Porcentaje de Pérdida resultante	: 0,549 (%)
Area Ocupada por las piezas	: 99,451 (%)
Cantidad de Piezas incluidas	: 105

```
------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------   MIGRACION NRO.    5   ---------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
AG       CROMOSOMA DE PIEZAS                                                                                           PÉRDIDA
------------------------------------------------------------------------------------------------------------------------------
 1-  1)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  2)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  3)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  4)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  5)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  6)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  7)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  8)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1-  9)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 1- 10)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111011111   300,000
 2-  1)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  2)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  3)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  4)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  5)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  6)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  7)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  8)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2-  9)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 2- 10)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 3-  1)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111   200,000
 3-  2)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111   200,000
 3-  3)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111   200,000
 3-  4)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111   200,000
 3-  5)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111   200,000
 3-  6)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 3-  7)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 3-  8)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111110111111   300,000
 3-  9)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111110111111   300,000
 3- 10)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   300,000
 4-  1)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  2)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  3)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  4)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  5)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  6)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  7)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  8)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4-  9)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 4- 10)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  1)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  2)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  3)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  4)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  5)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  6)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  7)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  8)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5-  9)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000
 5- 10)  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111   100,000

AG       CROMOSOMA ROTACIÓN
--------------------------------------------------------------------------------------------------------------------
 1-  1)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1-  2)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1-  3)  1001101001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1-  4)  1011001001011111000101100110000100110010111111110111001101011100100111101011001101101000011011111100100011101
 1-  5)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1-  6)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1-  7)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1-  8)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1-  9)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 1- 10)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 2-  1)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011100
 2-  2)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011000
 2-  3)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011100
 2-  4)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011000
 2-  5)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011100
 2-  6)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011100
 2-  7)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011100
 2-  8)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011100
 2-  9)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011000
 2- 10)  1011001001011111000101100110000100110010111111110111001101011100010111101001001101101000011011111100100011000
 3-  1)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100000111
 3-  2)  1011001001011111000101100110000100110010111111110111001101011100100111101001001101101000011011111100100000111
 3-  3)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100000111
 3-  4)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100000111
 3-  5)  1011001001011111000101100110000100110010111111110111001101011100100111101001001101101000011011111100100000111
 3-  6)  1011001001011111000101100110000100110010111111110111001101011100100111101001001101101000011011111100100001101
 3-  7)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100011111
 3-  8)  1011001001011111000101100110000100110010111111110111001101011100010111101011001101101000011011111100100000100
 3-  9)  1011001001011111000101100110000100110010101111110111001101011100010111101011001101101000011011111100100000100
 3- 10)  1011001001011111000101100110000100110010111111110111001101011100100111101001001101101000011011111100100011111
 4-  1)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101101010011011111100100011111
 4-  2)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111010011011111100100011111
 4-  3)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111010011011111100100011111
 4-  4)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111010011011111100100011111
 4-  5)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111010011011111100100011111
 4-  6)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111000011011111100100011111
 4-  7)  1011001001011111000111100110000100110010111111110111001101011100010111101001001101111010011011111100100011111
 4-  8)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111010011011111100100011111
 4-  9)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111010011011111100100011111
 4- 10)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101111010011011111100100011111
 5-  1)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101101000011011111100100011100
 5-  2)  1011001001011111000111100110000100110010111111110111001101011100010111101011001101101000011011111100100011101
 5-  3)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101101000011011111100100011101
 5-  4)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101101000011011111100100011101
 5-  5)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101101000011011111100100011101
 5-  6)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101101000011011111100100011101
 5-  7)  1011001001011111000111100100000100110010111111110111001101011100010111101011001101101000011011111100100011101
 5-  8)  1011001001011111000111100100000100110010111111110111001101011100100111101011001101101000011011111100100011101
 5-  9)  1011001001011111000111100100000100110010111111110111001101011100100111101011001101101000011011111100100011101
 5- 10)  1011001001011111000111100100000100110010111111110111001101011100100111101011001101101000011011111100100011101

DATOS DEL MEJOR INDIVIDUO ENCONTRADO POR EL COORDINADOR
-----------------------------------------------------
AG que proporciona el Mejor Individuo          : 4
Cromosoma de Piezas                            : 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111
Cromosoma Rotación                             : 1011001001011111000111100100000100110010111111110111001101011100010111101011001101101010011011111100100011111
Pérdida resultante                             : 100,000 (Unidades Cuadradas)
Porcentaje de Pérdida resultante               :   0,110 (%)
Area Ocupada por las piezas                    :  99,890 (%)
Cantidad de Piezas incluidas                   : 108
-----------------------------------------------------
```

MIGRACION NRO. 6

AG	CROMOSOMA DE PIEZAS	PÉRDIDA
1- 1)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111	0,000
1- 2)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 3)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 4)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 5)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 6)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 7)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 8)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 9)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
1- 10)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 1)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 2)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 3)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 4)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 5)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 6)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 7)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 8)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 9)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
2- 10)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
3- 1)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111	0,000
3- 2)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111	0,000
3- 3)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111	0,000
3- 4)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111	0,000
3- 5)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
3- 6)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
3- 7)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
3- 8)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
3- 9)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
3- 10)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 1)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 2)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 3)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 4)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 5)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 6)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 7)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 8)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 9)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
4- 10)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 1)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 2)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 3)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 4)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 5)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 6)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 7)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 8)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 9)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000
5- 10)	11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101111	100,000

AG	CROMOSOMA ROTACIÓN
1- 1)	10110010010111110001110010000010011001011111111011001101011000010111010110011011110000101111110010000111
1- 2)	10110010010111110001110010000010011001011111111011001101011000010111010110011011110100101111110010000111
1- 3)	10110010010111110001110010000010011001011111111011001101011000010111010110011010000010111111110010000111
1- 4)	10110010010111110001110010000010011001011111111011001101011000010111010110011011010000101111111110010000111101
1- 5)	10110010010111110001110010000010011001011111111011001101011000010111010110011010101001011111111110010000111
1- 6)	10110010010111110001110010000010011001011111111011001101011000010111010110011011010100101111111110010000111111
1- 7)	10110010010111110001110010000010011001011111111011001101011000010111010010011011110000101111111110010000111111
1- 8)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
1- 9)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
1- 10)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
2- 1)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
2- 2)	10110010010111110001110010000010011001011111111011001101011000010111010110011011010000101111111110010000111111
2- 3)	10110010010111110001110010000010011001011111111011001101011000010111010110011011010000101111111110010000111111
2- 4)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
2- 5)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
2- 6)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
2- 7)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111100
2- 8)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111100
2- 9)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
2- 10)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
3- 1)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000000100
3- 2)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000000111
3- 3)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000000111
3- 4)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000000111
3- 5)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
3- 6)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
3- 7)	10110010010111110001110010000010011001011111111011001101011000010011110100100110110100001011111111110000011101
3- 8)	10110010010111110001110010000010011001011111111011001101011000010111010010011010100001011111111110010000111101
3- 9)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111111
3- 10)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
4- 1)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
4- 2)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100101001011111111110010000111111
4- 3)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
4- 4)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
4- 5)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100101001011111111110010000111111
4- 6)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100101001011111111110010000111111
4- 7)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
4- 8)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
4- 9)	10110010010111110001110010000010011001011111111011001101011000010111010110011011101101001011111111110010000111111
4- 10)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100101001011111111110010000111111
5- 1)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111101
5- 2)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111101
5- 3)	10110010010111110001110010000010011001011111111011001101011000010011110100100110110100001011111111110010000111101
5- 4)	10110010010111110001110010000010011001011111111011001101011000010011110100100110110100001011111111110010000111101
5- 5)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111101
5- 6)	10110010010111110001110010000010011001011111111011001101011000010011110100100110110100001011111111110010000111101
5- 7)	10110010010111110001110010000010011000011111111011001101011000010111010110011010100001011111111110010000111101
5- 8)	10110010010111110001110010000010011001011111111011001101011000010011110100100110110100001011111111110010000111101
5- 9)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111111
5- 10)	10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111101

DATOS DEL MEJOR INDIVIDUO ENCONTRADO POR EL COORDINADOR

AG que proporciona el Mejor Individuo	: 1
Cromosoma de Piezas	: 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
Cromosoma Rotación	: 10110010010111110001110010000010011001011111111011001101011000010111010110011010100001011111111110010000111101
Pérdida resultante	: 0,000 (Unidades Cuadradas)
Porcentaje de Pérdida resultante	: 0,000 (%)
Area Ocupada por las piezas	: 100,000 (%)
Cantidad de Piezas incluidas	: 109

Apéndice B

Obtención del Número de Corridas necesario a realizar a cada problema para obtener el “Speedup” promedio del Algoritmo Genético Paralelo

A continuación se presenta el mecanismo utilizado para obtener el Número de Corridas necesario, de modo de conseguir el “Speedup” promedio de un problema (PCPGBR) que se resuelve mediante el Algoritmo Genético Paralelo diseñado.

Debido a la componente aleatoria de los Algoritmos Genéticos, para conocer el “Speedup” alcanzado por el AGP al resolver un problema dado, se requiere realizar varias corridas del problema. Con ello, se consigue un “Speedup” promedio, el cual permite obtener con mayor exactitud la eficiencia del Algoritmo Genético Paralelo.

Para un problema dado, del cual se conoce el óptimo con pérdida cero, se obtiene el tiempo que tomará el AGP en encontrar el óptimo. Se realizan corridas sucesivas, obteniendo en cada caso, tanto el tiempo promedio alcanzado como el error respecto al promedio. Entonces, experimentalmente, se mide el número de corridas necesaria para alcanzar tiempos promedios parecidos entre sí (mínimo error respecto al promedio). Esta prueba se realiza varias veces, de modo de confrontar los resultados obtenidos. Para todas las pruebas y con el fin de no distorsionar

los resultados, se requiere que los parámetros genéticos del AGP sean los mismos.

A continuación, se presenta un problema de prueba que permite obtener el número de corridas necesario para alcanzar un “Speedup” promedio representativo.

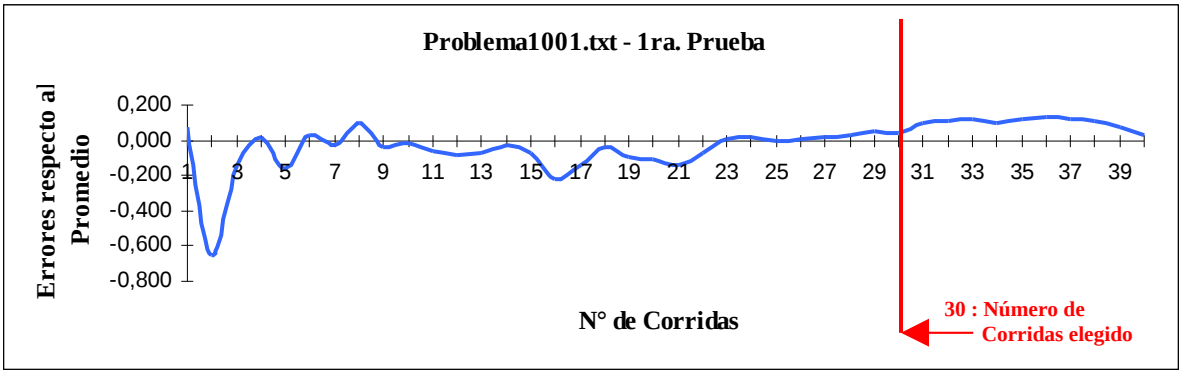
La Tabla N°B.1 muestra los parámetros genéticos del problema de prueba Problema1001.txt (Apéndice C), del cual se conoce que el óptimo es de pérdida cero.

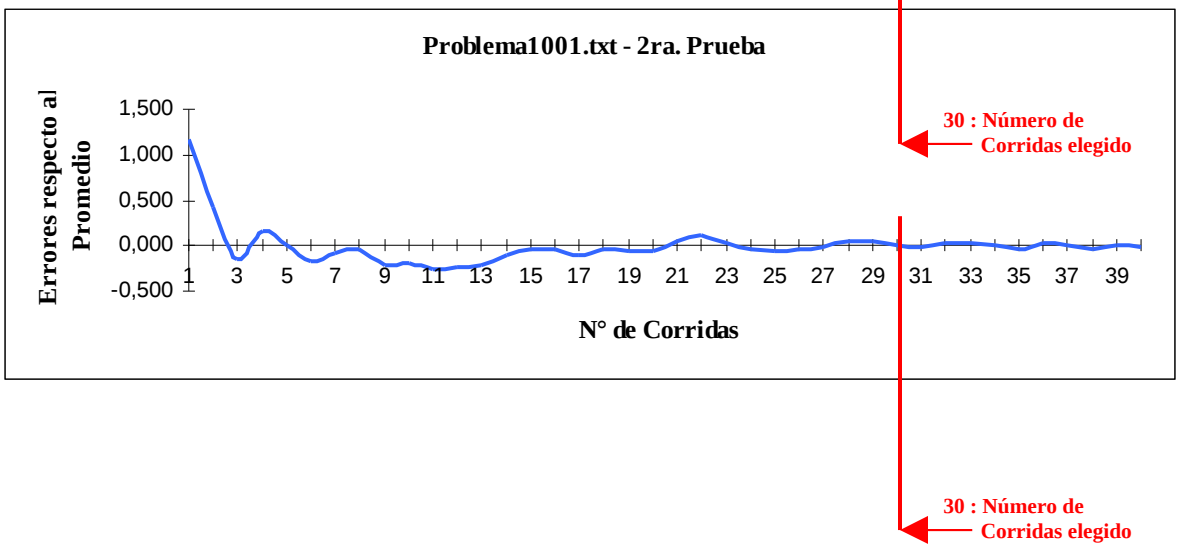
Tabla N°B.1: Parámetros Genéticos de un problema de prueba.

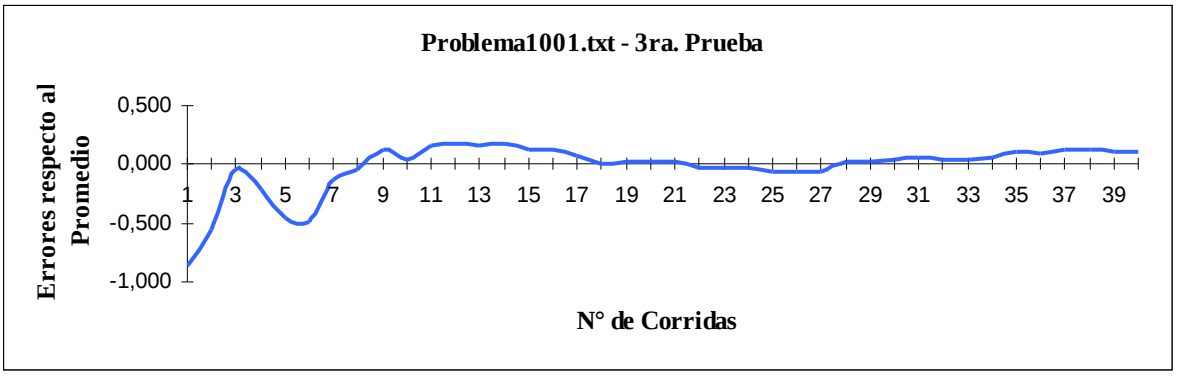
Nombre Problema	Problema1001.txt
Número total de Individuos	4000
Cantidad de AGs	4
Cantidad de Generaciones	200
Probabilidad de Cruzamiento	90 %
Probabilidad de Mutación	10 %
Tipo de Migración	Asíncrona
Cantidad de individuos en cada Sub-Población	1000
Número de Individuos a enviar a Coordinador	10 % de la Sub-Población
Número de Individuos a recibir desde Coordinador	10 % de la Sub-Población
Tasa de Migración	Cada 50 generaciones

Para este problema, se realizan 4 pruebas con 40 corridas cada una. Los gráficos de la Figura N°B.1 muestra los resultados obtenidos. De la Figura N°B.1 se observa que, en todos los casos, el mínimo error respecto al promedio ocurre alrededor del número de corrida treinta. Lo anterior equivale a decir que, alrededor del número de corrida treinta, el tiempo promedio alcanzado es representativo de todas las muestras que se han obtenido hasta el instante. Pruebas similares realizadas a otros problemas

del Apéndice C, arrojan resultados muy similares. Por lo tanto, se considera en treinta el Número de Corridas necesario, que permite conseguir "*Speedup*" promedio representativo de un problema (PCPGBR) que se resuelve mediante el Algoritmo Genético Paralelo diseñado.


Problema1001.txt - 1ra. Prueba
Errores respecto al Promedio
0,200
0,000
-0,200
-0,400
-0,600
-0,800
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
N° de Corridas
30 : Número de Corridas elegido


Problema1001.txt - 2ra. Prueba
Errores respecto al Promedio
1,500
1,000
0,500
0,000
-0,500
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
N° de Corridas
30 : Número de Corridas elegido


Problema1001.txt - 3ra. Prueba
Errores respecto al Promedio
0,500
0,000
-0,500
-1,000
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
N° de Corridas
30 : Número de Corridas elegido

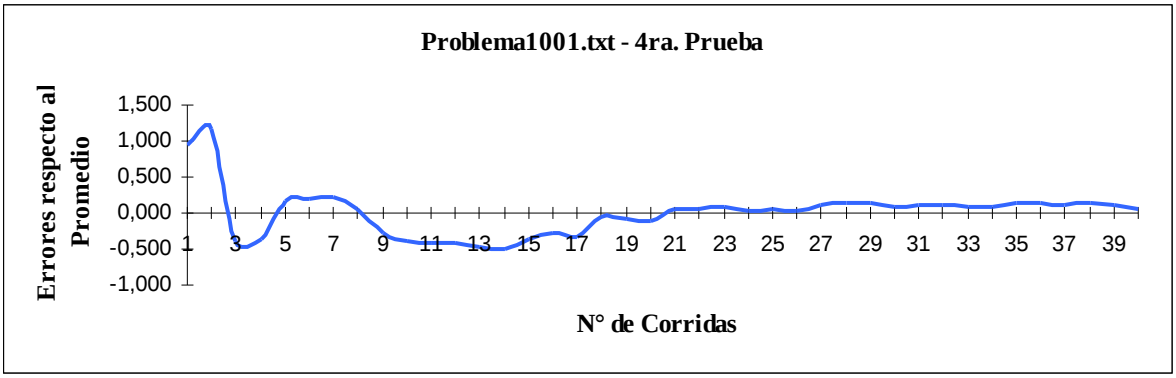


Figura N°B.1: Pruebas realizadas al Problema1001.txt para determinar el Número de Corridas necesarios para alcanzar un "*Speedup*" representativo.

Apéndice C

Problemas utilizados para realizar las pruebas

En las Tablas N°C.1 y N°C.2 se presentan cada uno de los problemas utilizados para realizar las pruebas del Algoritmo Genético Paralelo. El formato de los problemas se encuentra en el Apéndice A, sección "Formato archivo de texto con datos del problema de corte de pieza restricto". Estos problemas fueron diseñados considerando que el óptimo es de pérdida cero.

Tabla N°C.1: Problemas utilizados para realizar las pruebas del Algoritmo Genético Paralelo.

Problema1001.txt (84 piezas distintas)			Problema1002.txt (109 piezas distintas)
330 240 16 30 40 5 20 40 11 50 20 9 10 40 9 10 30 13 20 120 5 20 20 5 10 50 1	10 20 8 40 100 2 40 70 4 30 20 2 10 70 4 30 30 2 10 120 3 80 40 1	**30 : Número de Corridas elegido**	350 260 24

```
40 10 8
10 20 8
40 20 8
20 20 4
60 10 4
10 10 9
110 10 3
30 30 4
20 30 10
60 30 8
40 40 4
30 10 8
10 130 1
20 50 2
10 50 5
40 30 6
40 50 3
20 60 1
10 160 1
40 60 2
10 90 3
20 70 3
10 80 1
30 50 3
```

Problema1003.txt
(97 piezas distintas)

```
350 260
24
80 20 2
30 10 4
50 40 3
20 30 3
10 10 5
10 90 2
10 130 1
20 100 2
10 100 5
20 20 5
20 10 17
40 40 4
40 30 2
30 30 2
70 20 4
70 30 2
140 20 3
30 50 4
10 60 3
50 20 5
20 40 6
10 50 9
50 50 1
30 80 1
```

Problema1005.txt
(125 piezas distintas)

```
300 450
42
10 60 2
20 40 7
40 40 1
10 20 7
20 20 4
10 10 9
10 50 7
10 30 6
20 30 3
20 60 5
30 30 5
30 40 4
20 380 1
10 40 6
```

```
30 50 2
20 50 4
10 90 4
10 80 5
20 70 1
20 80 2
40 50 1
30 80 1
20 90 4
30 70 3
10 100 1
10 70 4
20 100 1
10 110 4
30 100 1
60 30 2
30 70 1
40 70 1
10 130 1
10 120 2
120 20 1
50 50 2
130 20 1
130 30 1
20 110 1
30 110 1
120 30 1
60 40 2
```

Tabla N°C.2: Continuación de los problemas utilizados para realizar las pruebas del Algoritmo Genético Paralelo.

Problema1006.txt
(128 piezas distintas)

```
300 450
31
20 40 6
20 20 3
10 20 14
30 20 5
10 10 7
30 10 5
40 10 7
60 10 6
50 10 4
70 10 2
30 30 3
30 50 4
20 50 8
30 40 6
70 40 2
20 60 4
60 30 3
70 20 5
90 10 3
40 40 2
50 40 1
50 50 3
30 70 4
80 10 4
80 20 3
80 30 1
```

```
50 60 1
120 10 8
60 80 2
20 90 1
90 70 1
```

Problema1007.txt
(168 piezas distintas)

```
330 240
16
30 40 10
20 40 22
50 20 18
10 40 18
10 30 26
20 120 10
20 20 10
10 50 2
10 20 16
40 100 4
40 70 8
30 20 4
10 70 8
30 30 4
10 120 6
80 40 2
```

Problema1008.txt
(95 piezas distintas)

```
30 45
42
1 1 5
1 2 3
1 3 1
1 4 3
1 5 4
1 6 3
1 7 4
2 2 5
2 6 4
1 21 2
3 5 2
1 11 1
2 11 2
3 2 4
3 4 3
3 21 1
2 23 1
4 5 3
4 2 1
14 2 1
5 2 3
27 1 1
24 1 2
15 1 2
16 1 2
8 1 3
8 2 1
8 3 2
4 4 2
```

```
14 1 3
1 18 2
25 1 1
5 5 2
5 7 1
2 17 2
7 2 3
3 11 1
4 11 1
9 1 1
9 2 2
1 12 4
5 12 1
```

Problema2025.txt
(109 piezas distintas)

```
59 57
59
2 2 8
2 23 1
3 5 3
3 3 3
3 2 5
8 3 1
2 10 1
16 3 1
6 1 1
1 24 1
1 3 3
1 21 1
1 8 4
1 16 2
1 11 1
1 13 1
1 23 1
1 1 15
2 8 2
4 3 4
1 2 7
5 1 1
6 2 1
6 3 1
2 5 1
4 25 1
4 5 3
4 8 1
4 4 4
```

```
4 2 1
4 10 1
4 25 1
4 8 1
4 10 1
5 57 1
6 11 1
6 51 1
9 5 1
9 57 1
23 2 1
8 55 1
6 8 1
5 8 1
4 7 2
4 9 1
4 1 1
1 11 1
2 5 1
1 5 2
2 14 1
13 9 1
2 9 1
15 3 1
15 12 1
10 10 1
1 9 1
4 10 1
```

Problema202

6.txt (225 piezas distintas)

```
69 77
42
1 1 57
1 2 36
1 3 26
1 4 18
1 6 5
1 8 10
1 13 3
1 10 5
1 19 4
2 2 7
2 3 9
2 4 7
2 8 4
2 13 1
2 19 1
1 28 2
2 6 2
2 10 1
2 28 1
1 24 1
1 11 2
2 10 1
2 9 1
2 11 1
```

```
1 9 2
13 53 1
5 24 1
2 24 1
6 24 1
5 1 2
5 6 1
10 11 1
6 4 1
4 4 1
3 6 1
10 54 1
7 77 1
25 13 1
9 64 1
16 41 1
16 22 1
16 1 1
```

Apéndice D

Layout de Problemas utilizados para realizar los experimentos

A continuación se presentan los layout resultantes, al ejecutar el Algoritmo Genético Paralelo a los problemas de corte de piezas: Problema1003.txt, Problema1005.txt, Problema1006.txt, Problema1007.txt, Problema1008.txt, Problema2025.txt, Problema2026.txt, mostrados en el Apéndice C, considerando desde uno a cinco Procesadores con Migración Asíncrona y Síncrona. La explicación

del formato de cada una de las figuras se encuentra en el Apéndice A, sección “Archivo de texto con información del *layout* del problema planteado”.

La Figura N°D.1 muestra el “*layout*” resultante del Problema1003.txt, al ejecutar el AGP considerando un procesador y Migración Asíncrona.

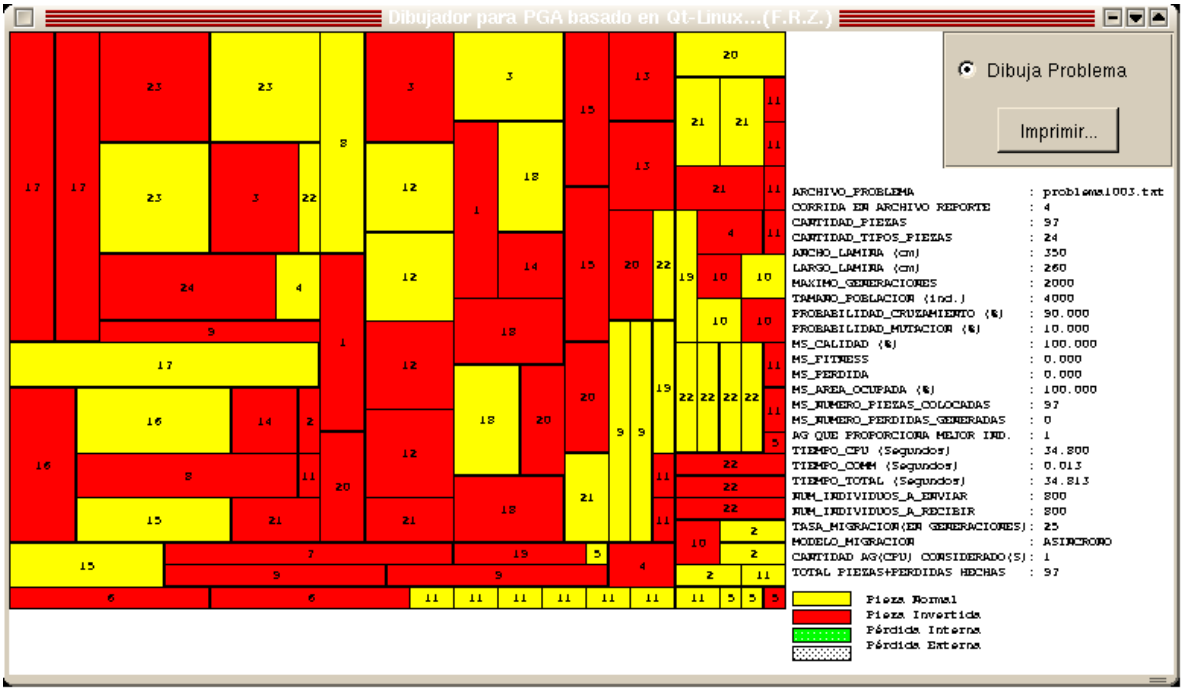
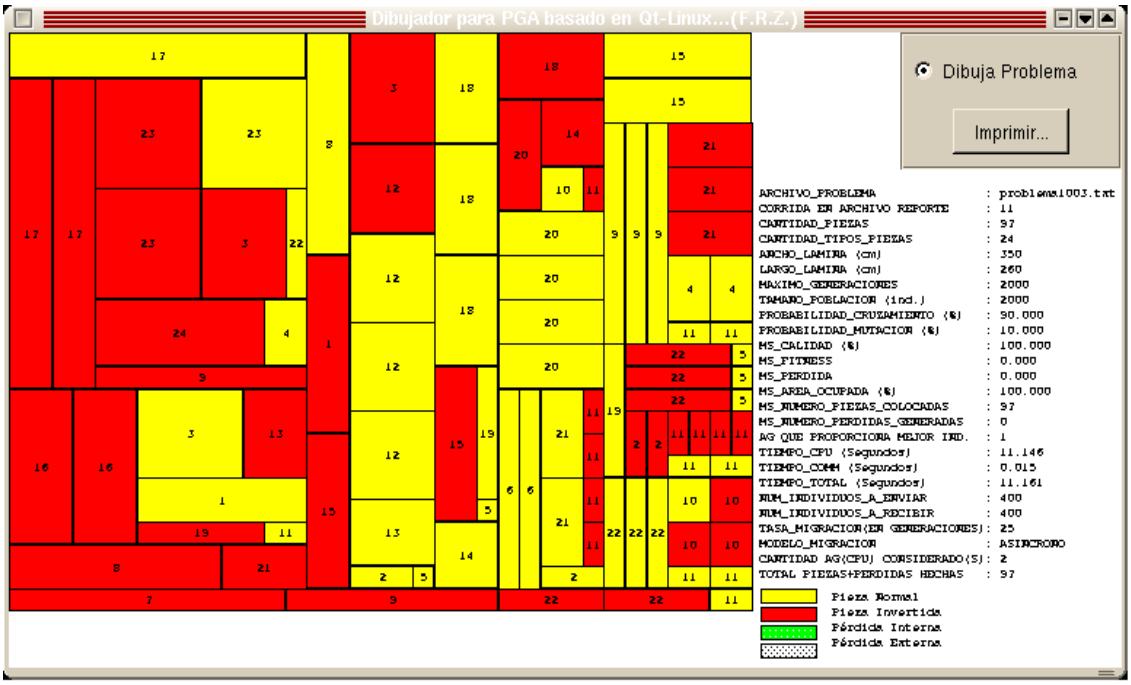


Figura N°D.1: Layout problema1003.txt, considerando un procesador y Migración Asíncrona.

La Figura N°D.2 muestra el “*layout*” resultante del Problema1003.txt, al ejecutar el AGP considerando dos procesadores con Migración Asíncrona y Síncrona.

a)



b)

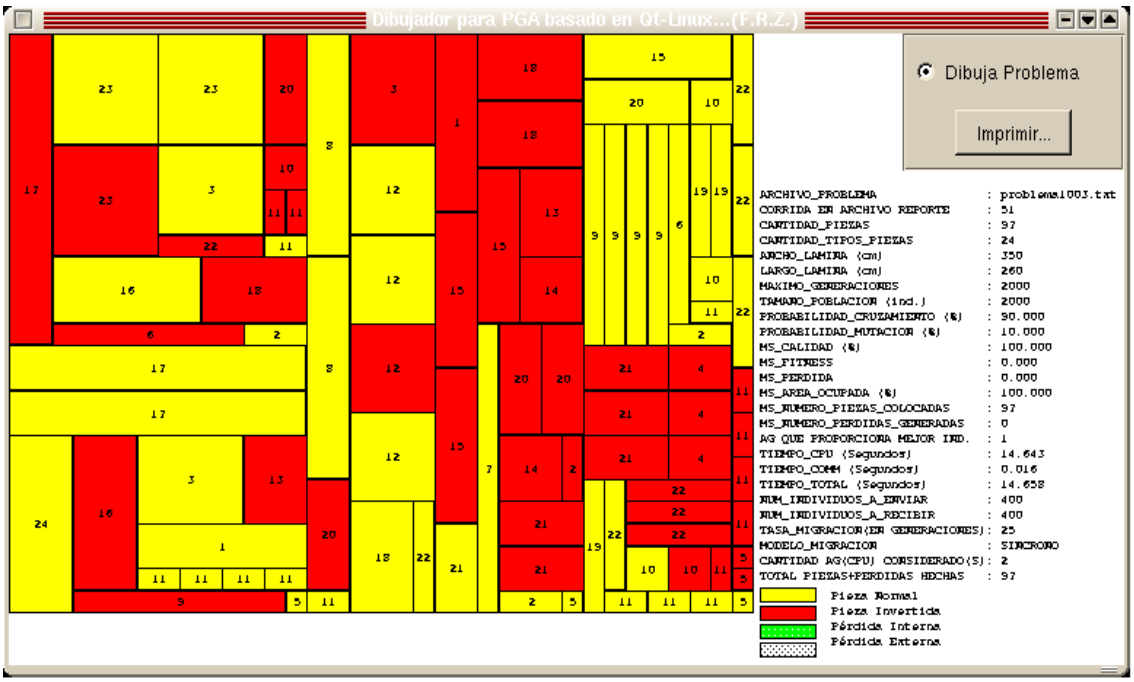
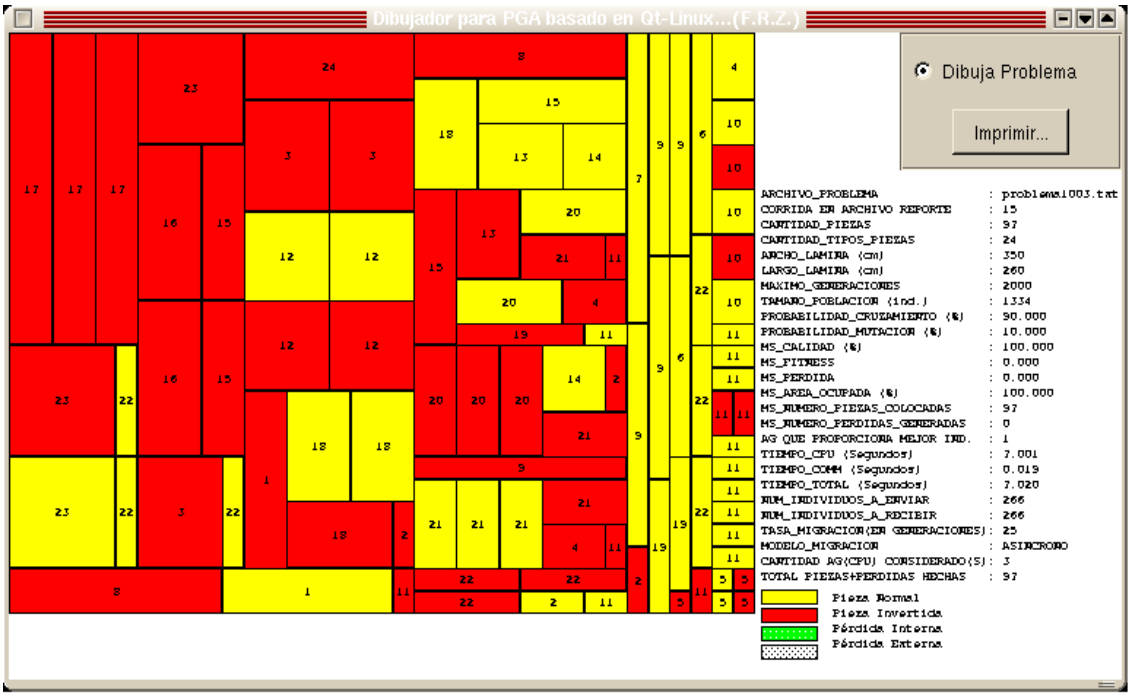
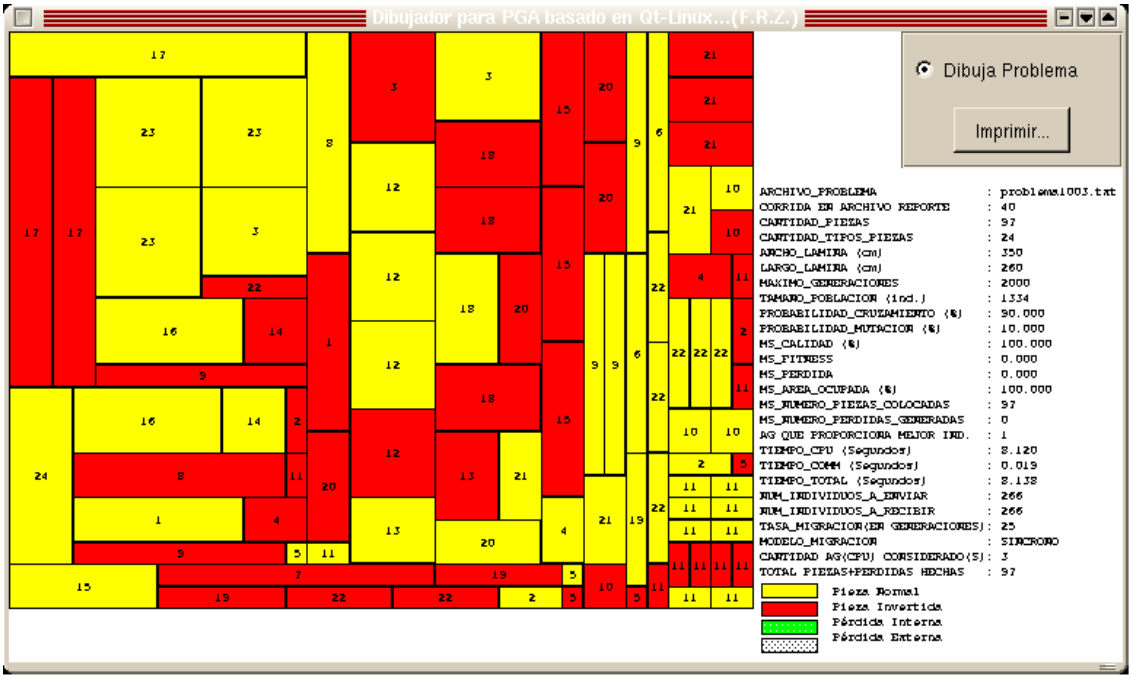


Figura N°D.2: Layout problema1003.txt, considerando dos procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.3 muestra el "*layout*" resultante del Problema1003.txt, al ejecutar el AGP considerando tres procesadores con Migración Asíncrona y Síncrona.



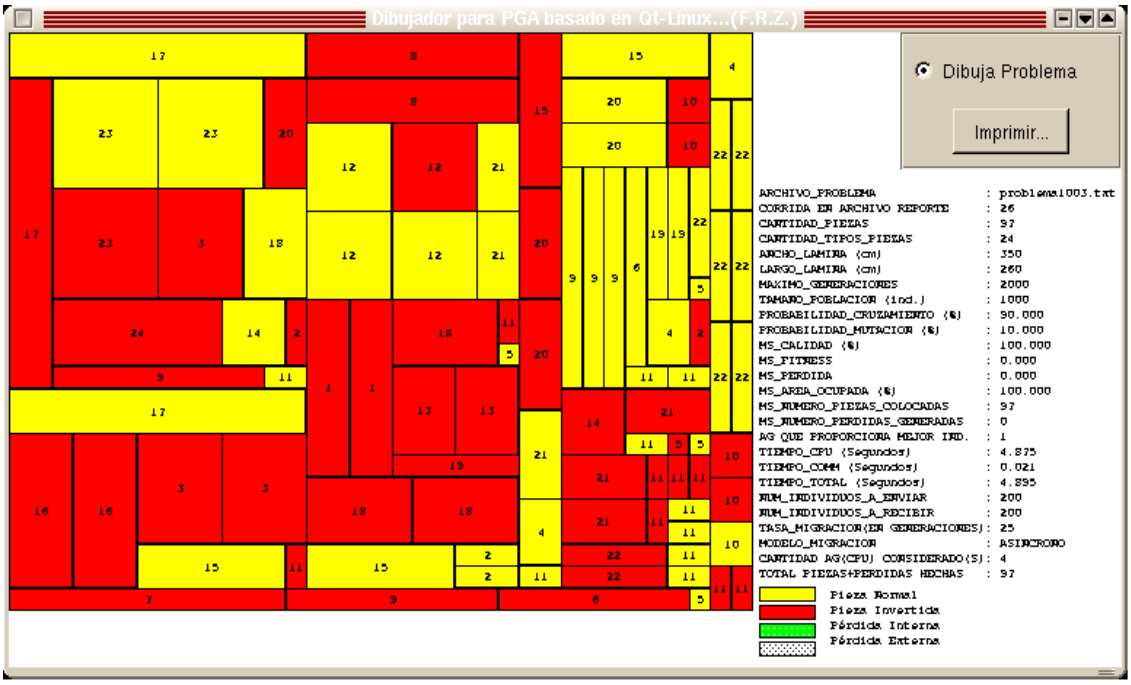
a)



b)

Figura N°D.3: Layout problema1003.txt, considerando tres procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.4 muestra el "*layout*" resultante del Problema1003.txt, al ejecutar el AGP considerando cuatro procesadores con Migración Asíncrona y Síncrona.

a) 

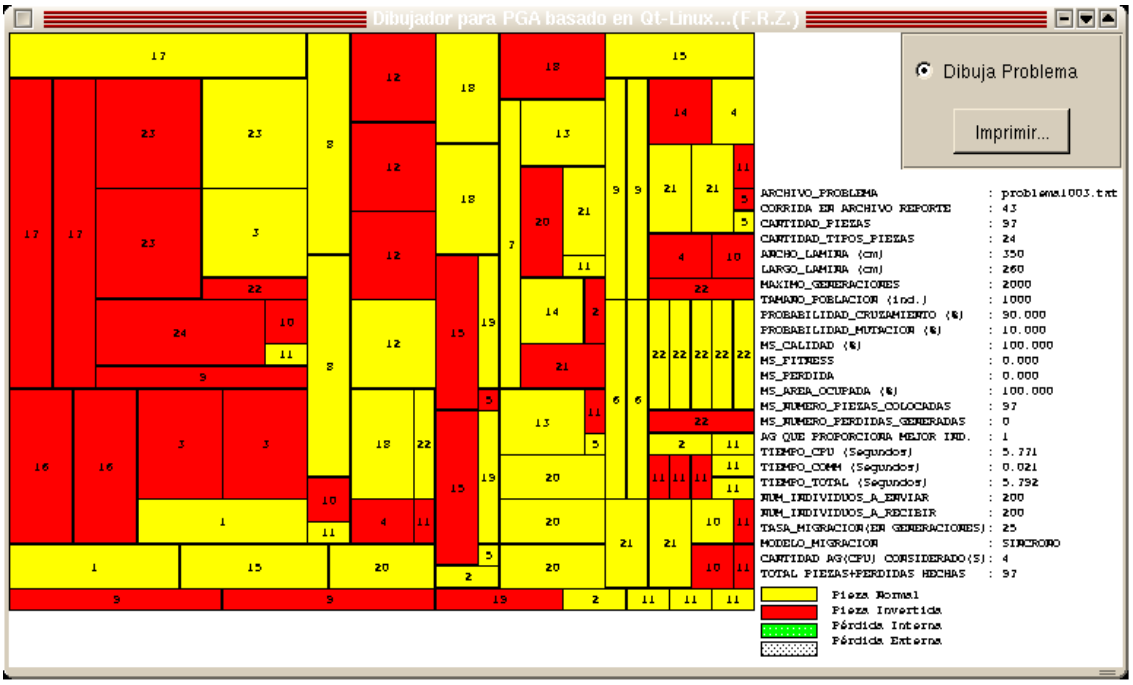
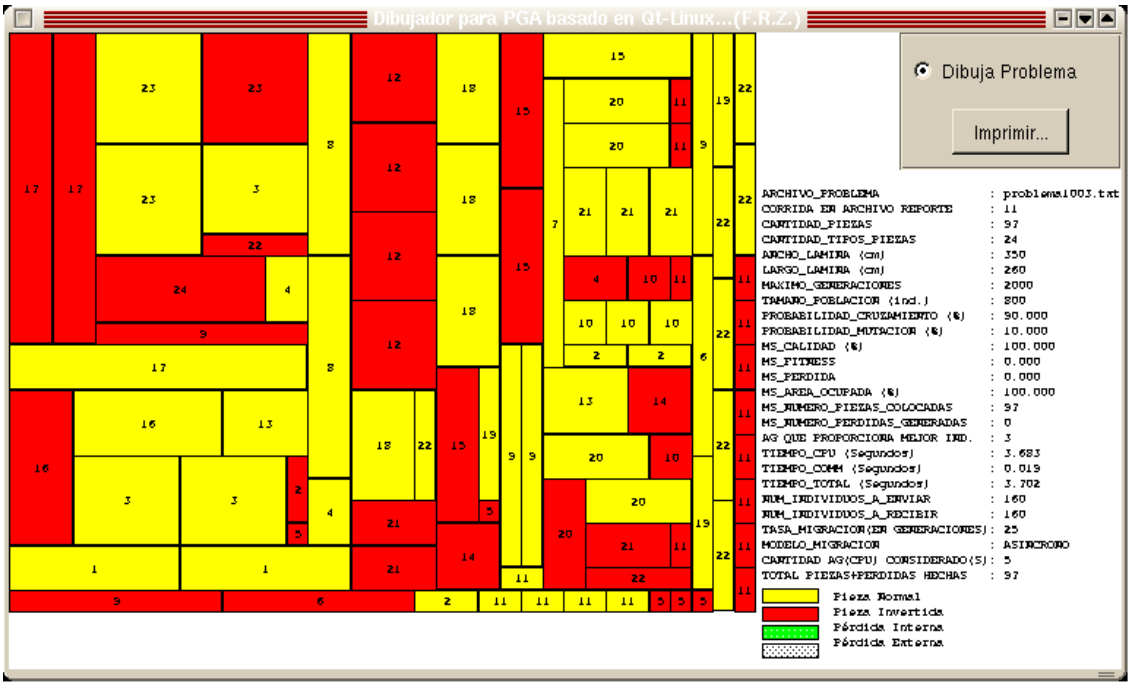
b) 

Figura N°D.4: Layout problema1003.txt, considerando cuatro procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.5 muestra el "*layout*" resultante del Problema1003.txt, al ejecutar el AGP considerando cinco procesadores con Migración Asíncrona y Síncrona.

a)



b)

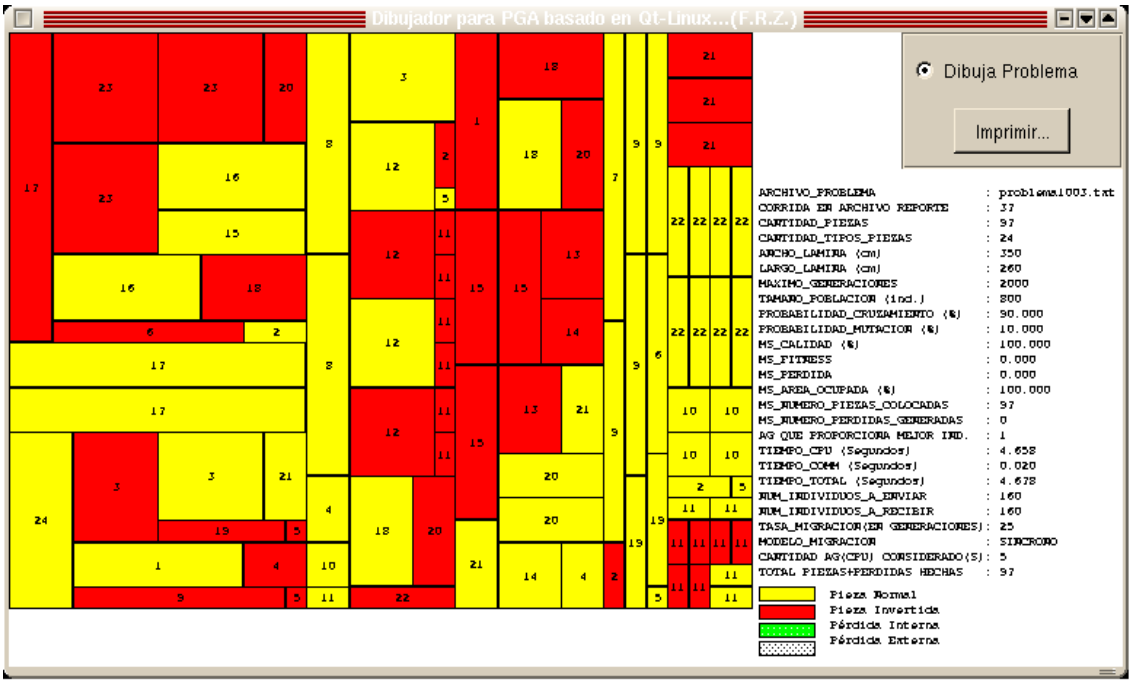


Figura N°D.5: Layout problema1003.txt, considerando cinco procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.6 muestra el "*layout*" resultante del Problema1005.txt, al ejecutar el AGP considerando un procesador y Migración Asíncrona.

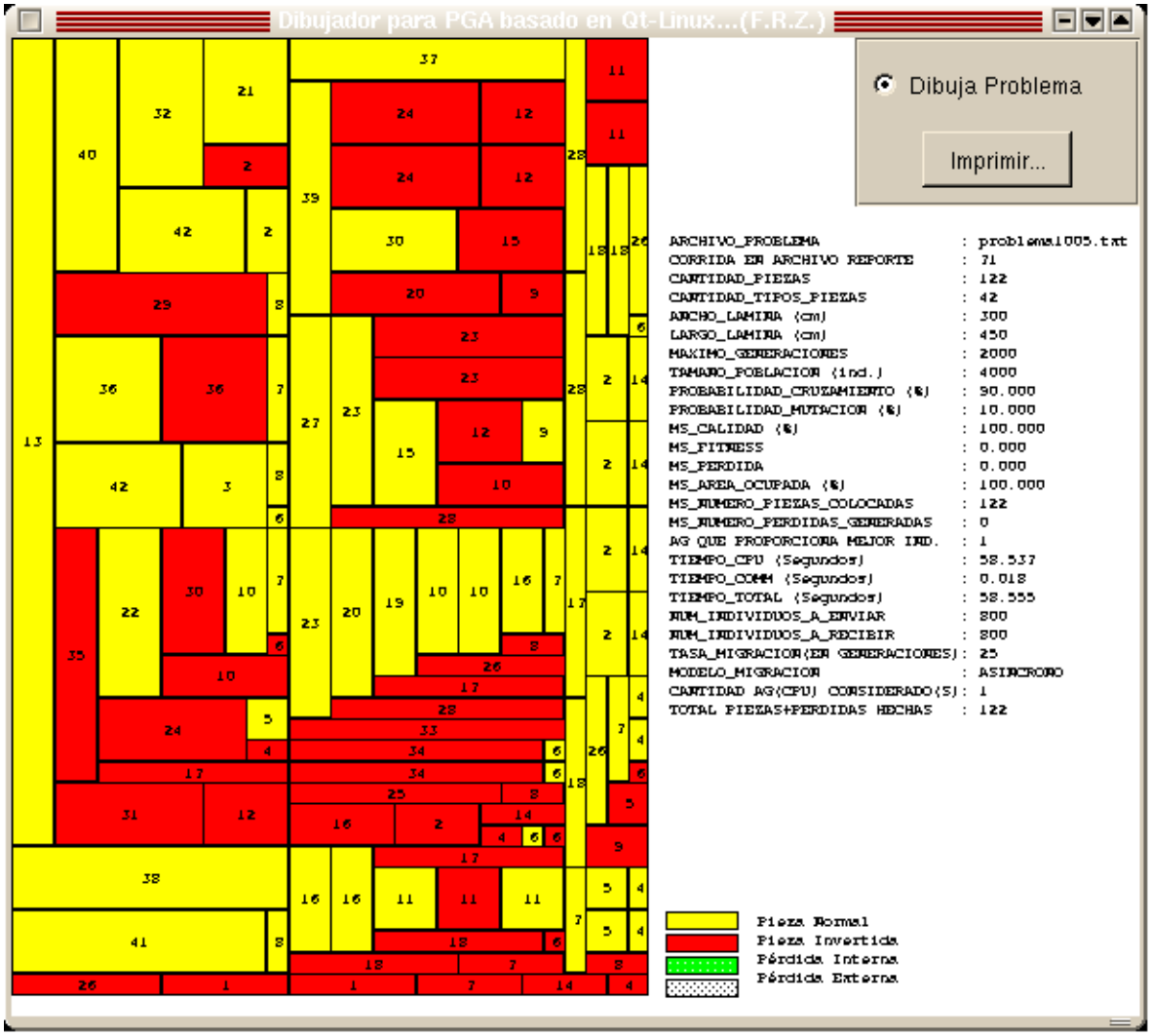
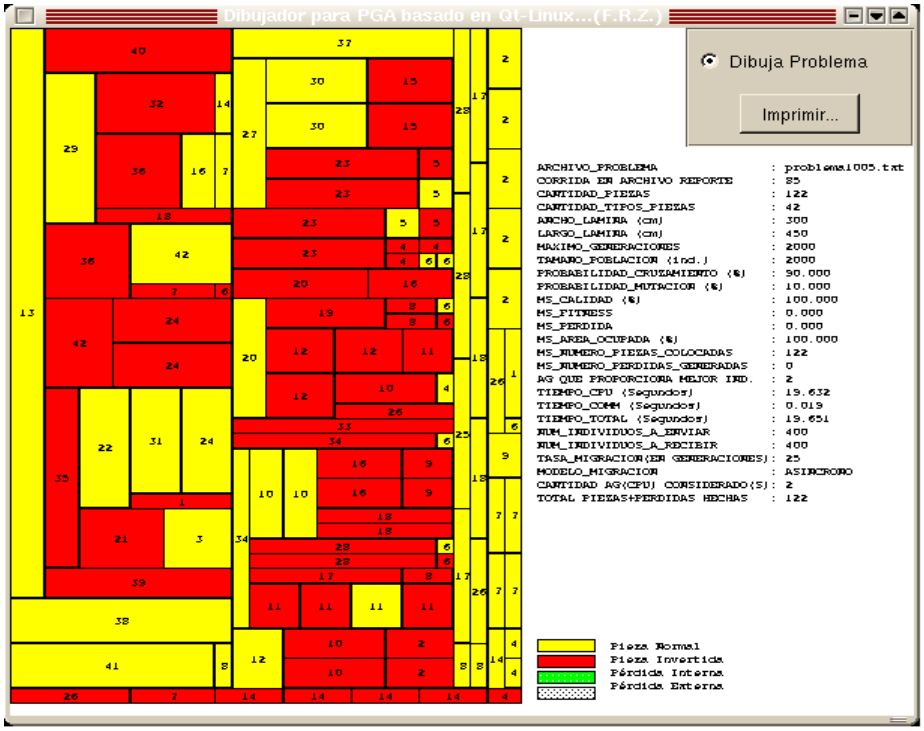


Figura N°D.6: Layout problema1005.txt, considerando un procesador y Migración Asíncrona.

La Figura N°D.7 muestra el "*layout*" resultante del Problema1005.txt, al ejecutar el AGP considerando dos procesadores con Migración Asíncrona y Síncrona.

a)



b)

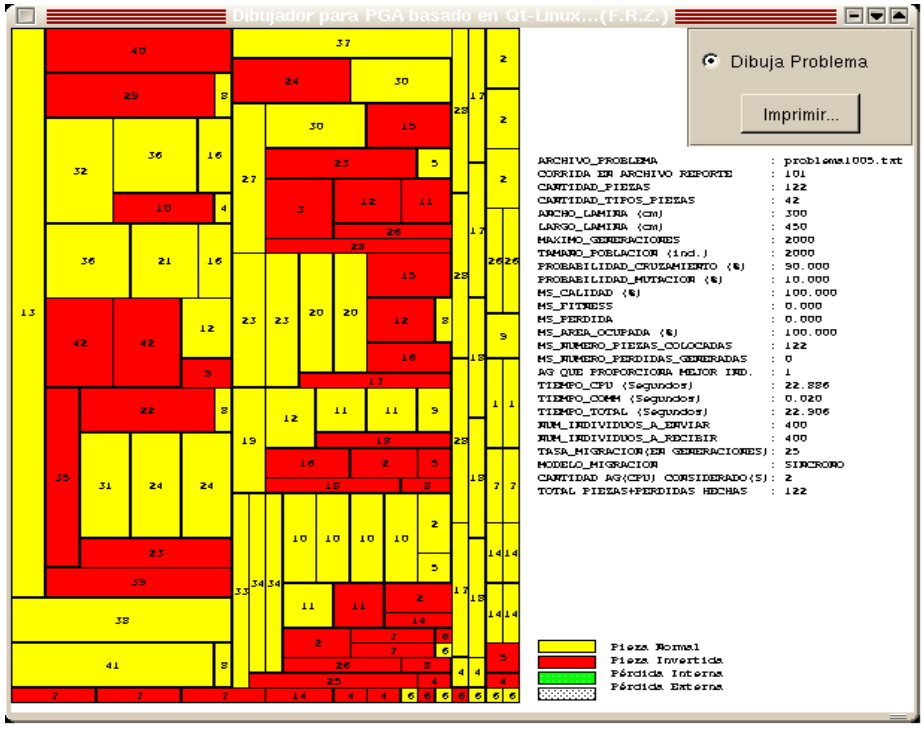
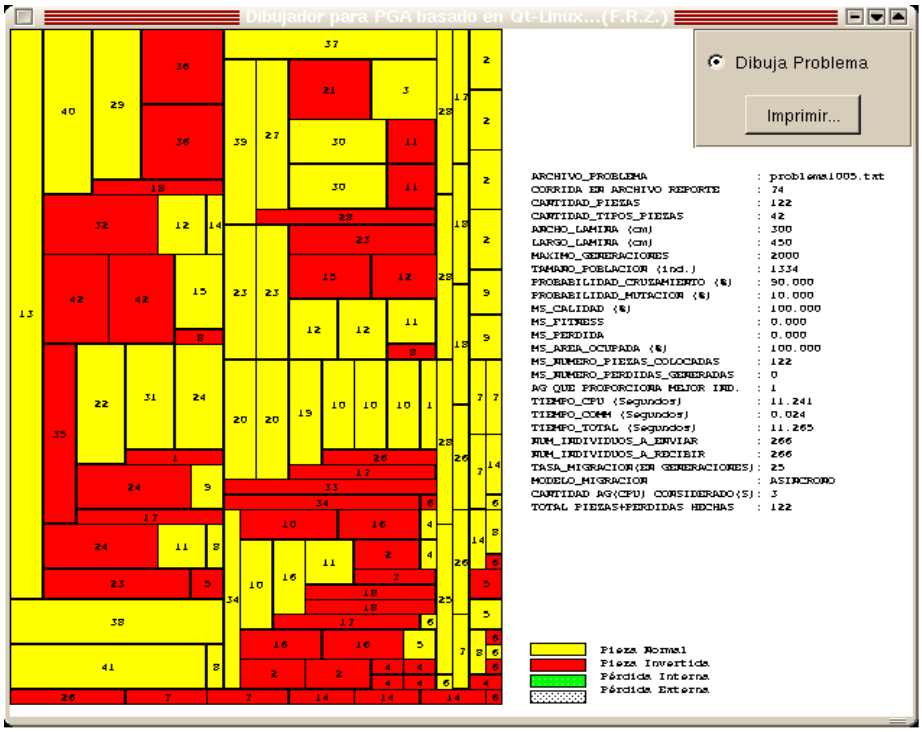


Figura N°D.7: Layout problema1005.txt, considerando dos procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.8 muestra el "*layout*" resultante del Problema1005.txt, al ejecutar el AGP considerando tres procesadores con Migración Asíncrona y Síncrona.

a) 

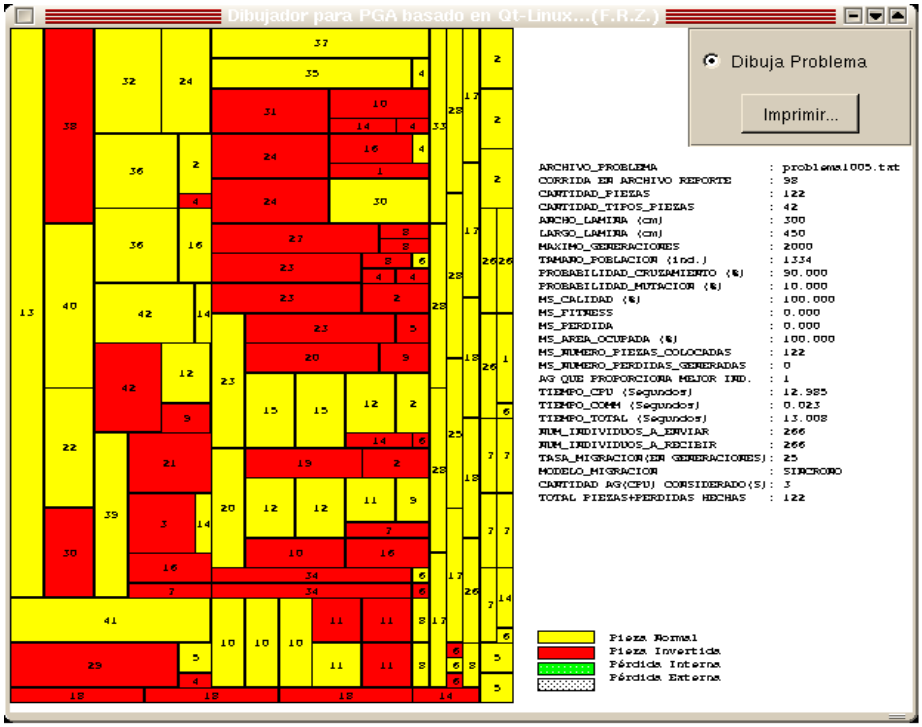
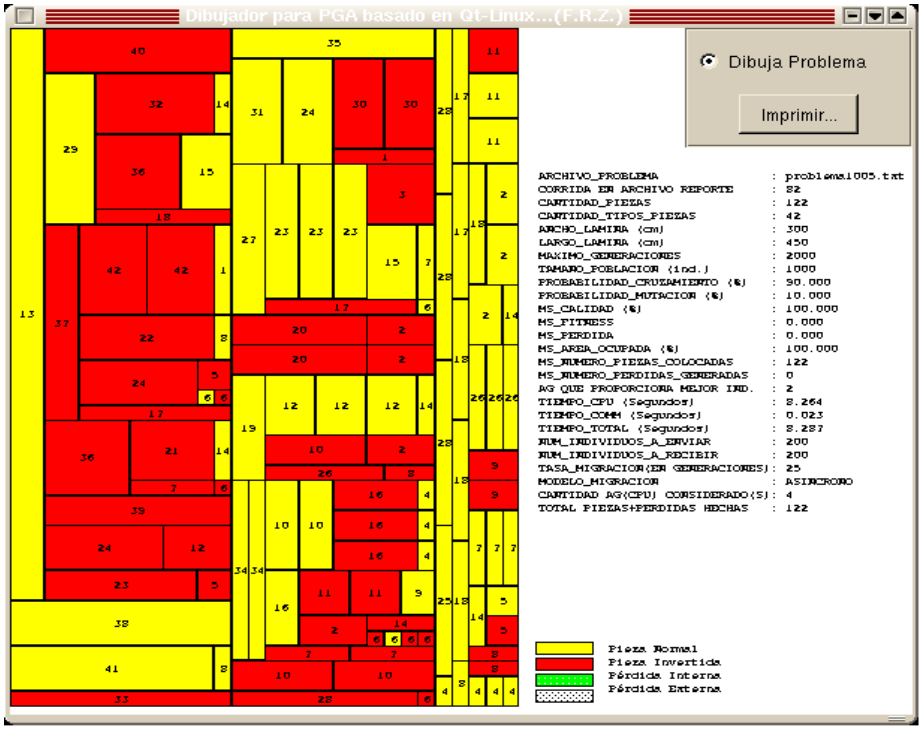
b) 

Figura N°D.8: Layout problema1005.txt, considerando tres procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.9 muestra el "*layout*" resultante del Problema1005.txt, al ejecutar el AGP considerando cuatro procesadores con Migración Asíncrona y Síncrona.

a)



b)

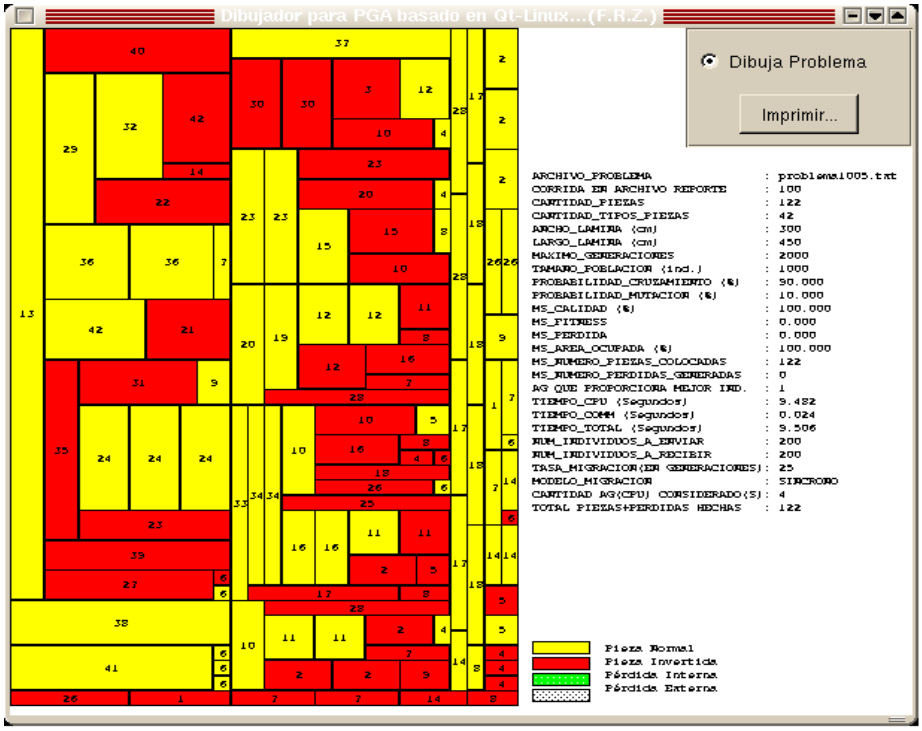
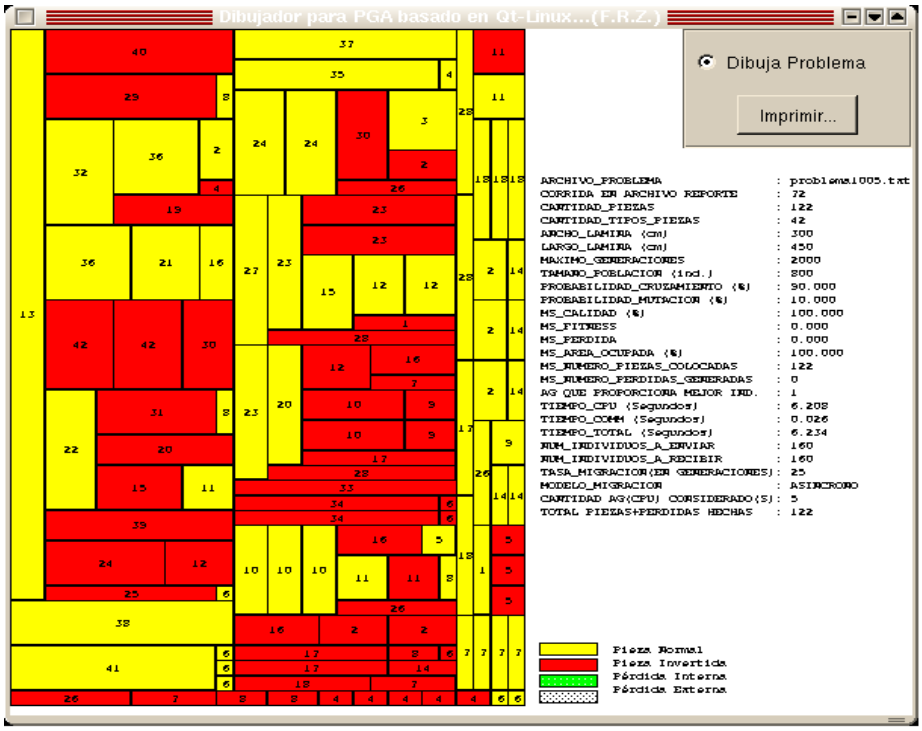


Figura N°D.9: Layout problema1005.txt, considerando cuatro procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.10 muestra el "*layout*" resultante del

Problema1005.txt, al ejecutar el AGP considerando cinco procesadores con Migración Asíncrona y Síncrona.

a)



b)

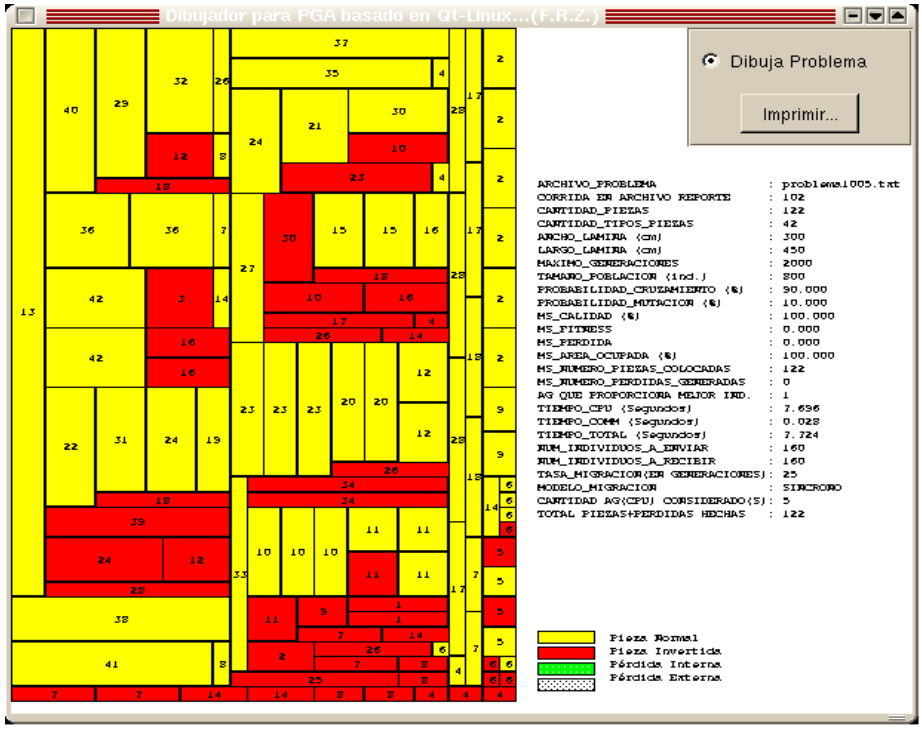


Figura N°D.10: Layout problema1005.txt, considerando cinco procesadores con Migración (a) Asíncrona y (b)

Síncrona.

La Figura N°D.11 muestra el "*layout*" resultante del Problema1006.txt, al ejecutar el AGP considerando un procesador y Migración Asíncrona.

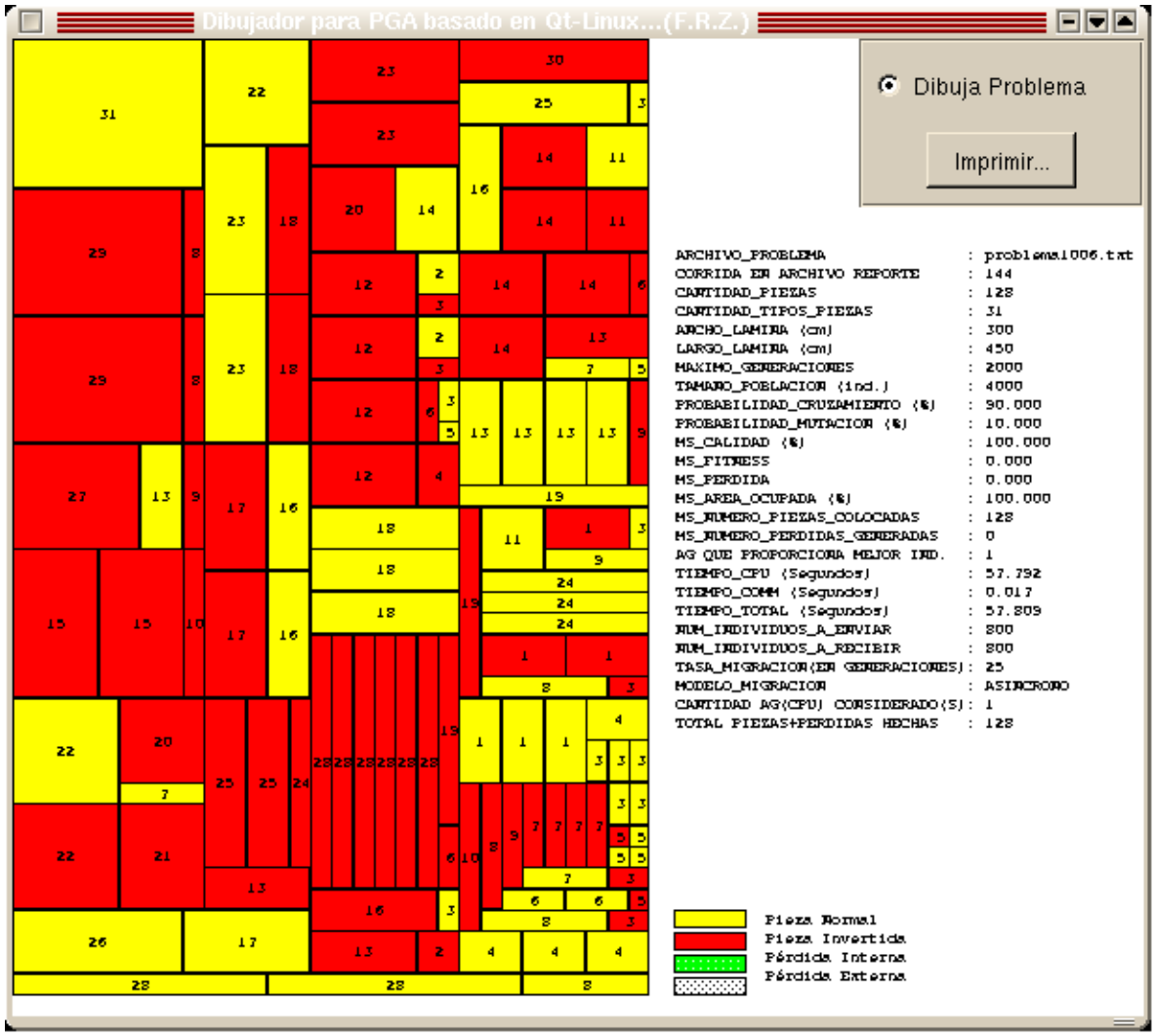
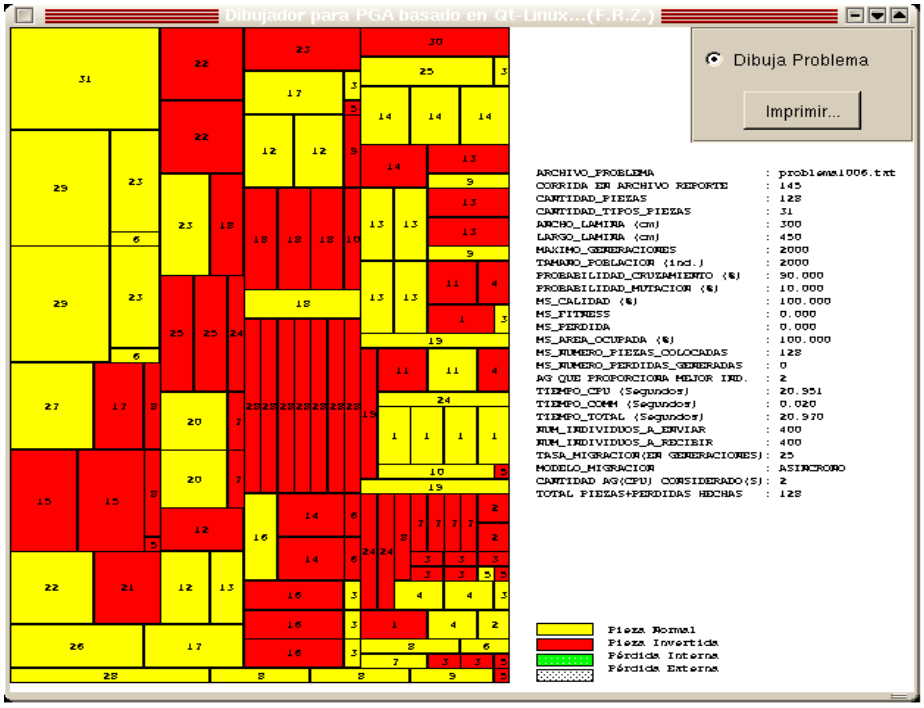


Figura N°D.11: Layout problema1006.txt, considerando un procesador y Migración Asíncrona.

La Figura N°D.12 muestra el "*layout*" resultante del Problema1006.txt, al ejecutar el AGP considerando dos procesadores con Migración Asíncrona y Síncrona.

a)



b)

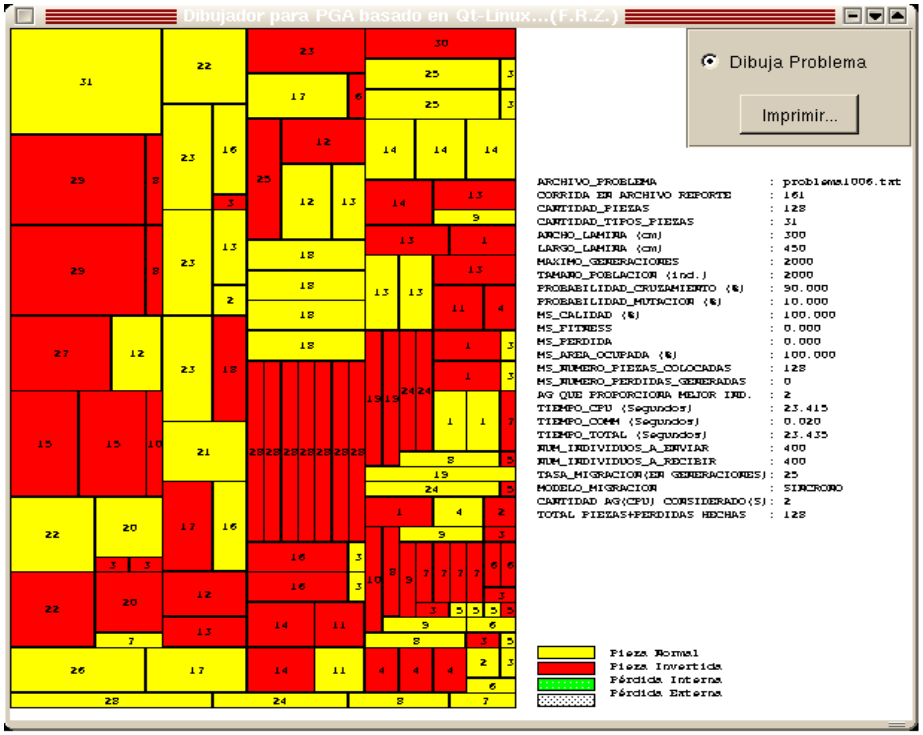
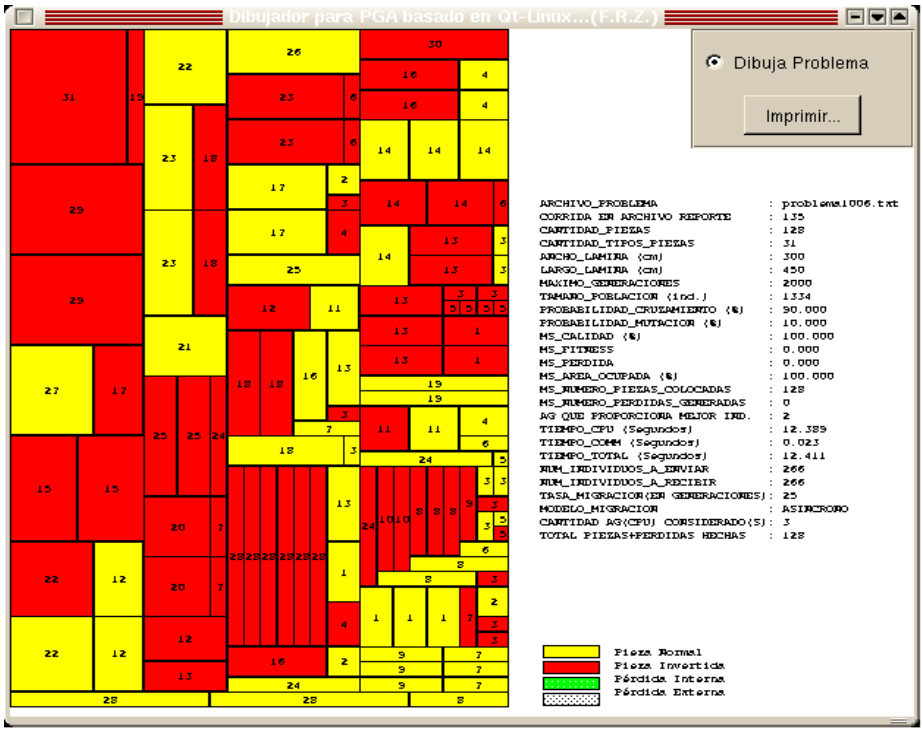


Figura N°D.12: Layout problema1006.txt, considerando dos procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.13 muestra el "*layout*" resultante del Problema1006.txt, al ejecutar el AGP considerando tres procesadores con Migración Asíncrona y Síncrona.

a) 

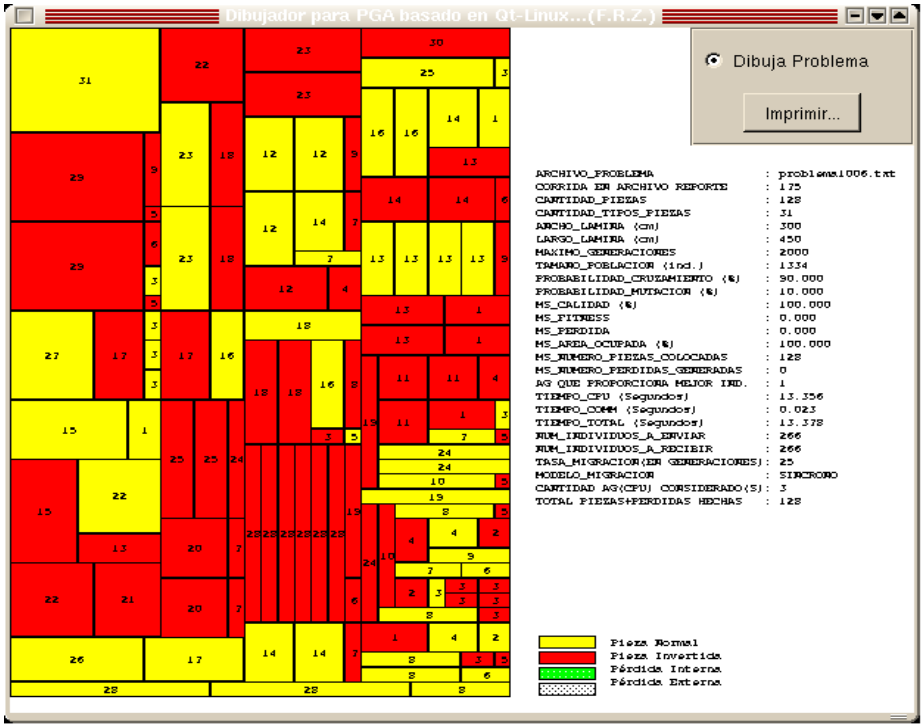
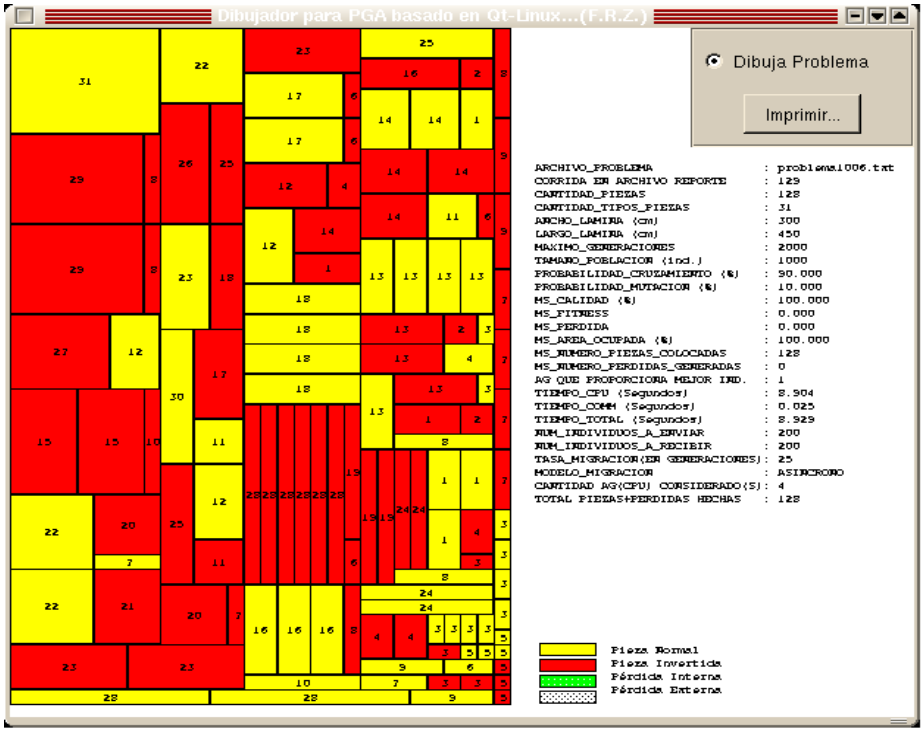
b) 

Figura N°D.13: Layout problema1006.txt, considerando tres procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.14 muestra el "*layout*" resultante del Problema1006.txt, al ejecutar el AGP considerando cuatro procesadores con Migración Asíncrona y Síncrona.

a)



b)

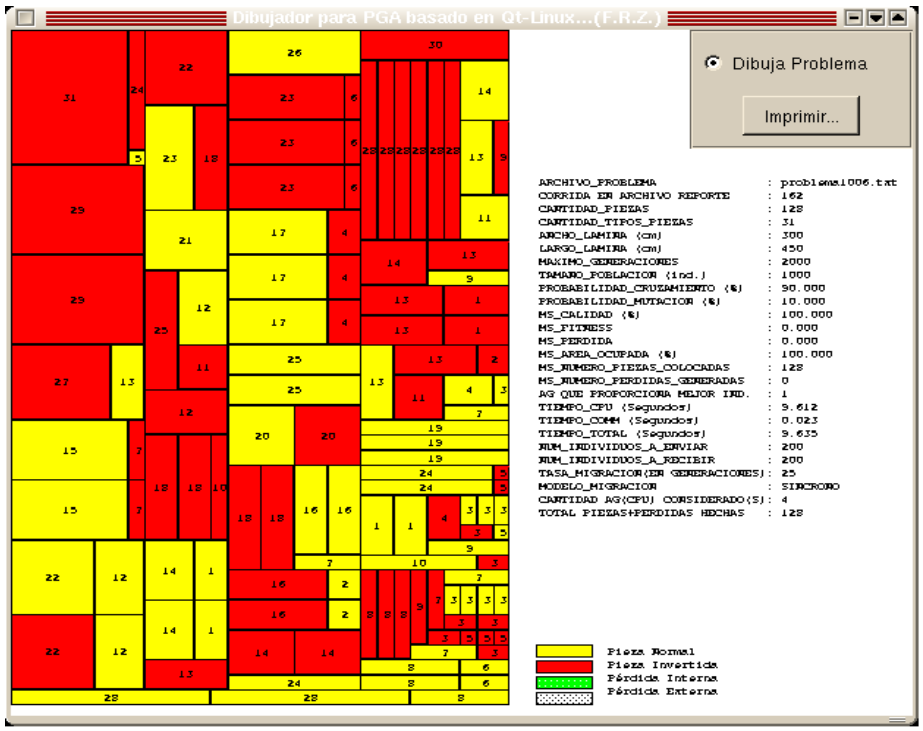
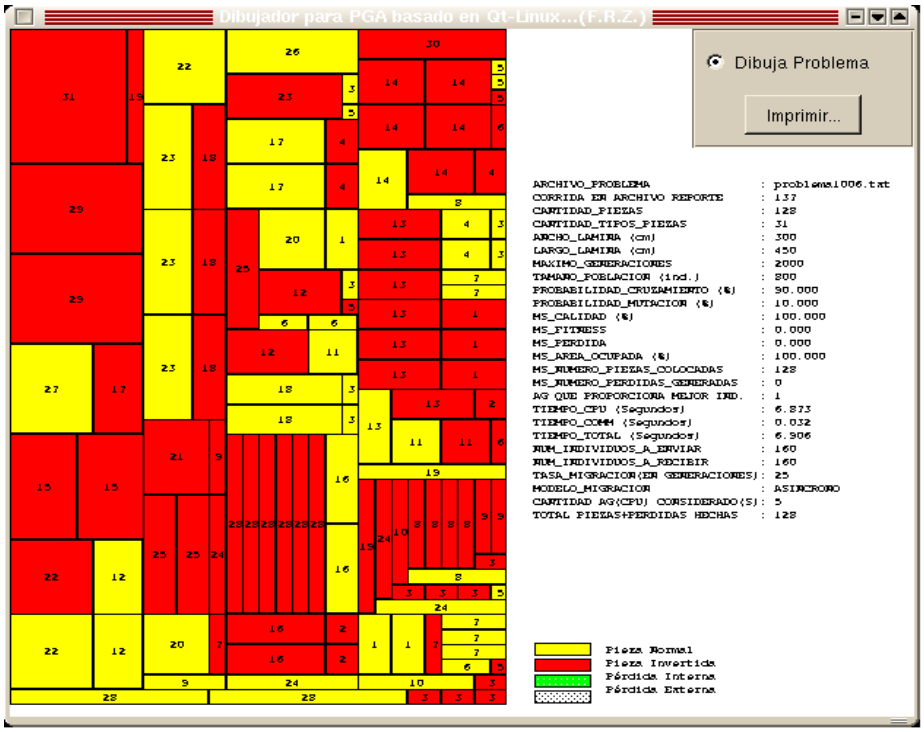


Figura N°D.14: Layout problema1006.txt, considerando cuatro procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.15 muestra el "*layout*" resultante del

Problema1006.txt, al ejecutar el AGP considerando cinco procesadores con Migración Asíncrona y Síncrona.

a)



b)

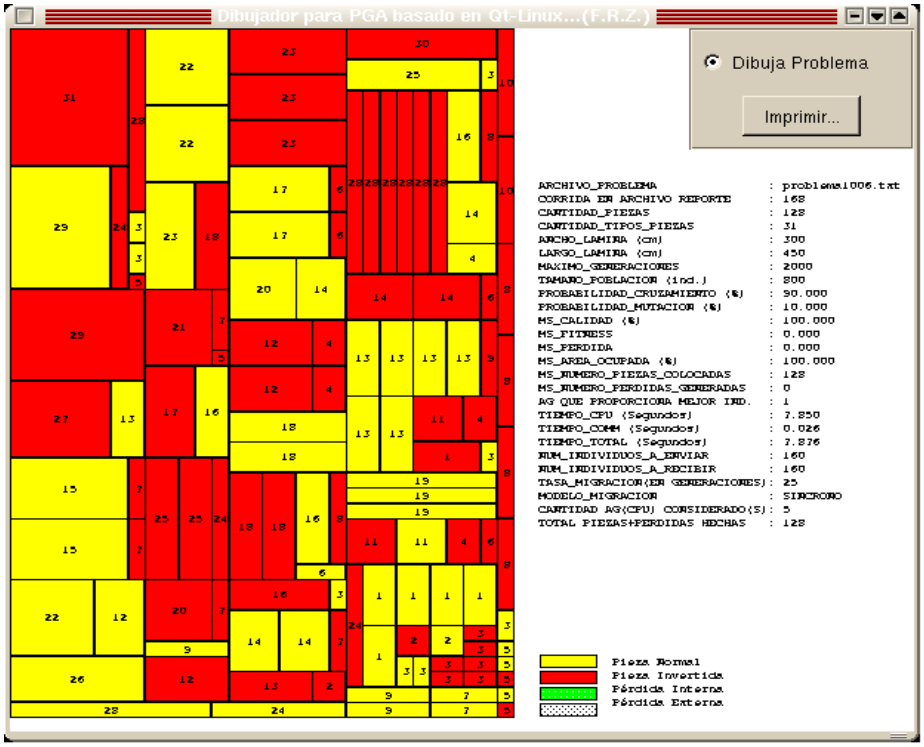


Figura N°D.15: Layout problema1006.txt, considerando cinco procesadores con Migración (a) Asíncrona y (b)

Síncrona.

La Figura N°D.16 muestra el "*layout*" resultante del Problema1007.txt, al ejecutar el AGP considerando un procesador y Migración Asíncrona.

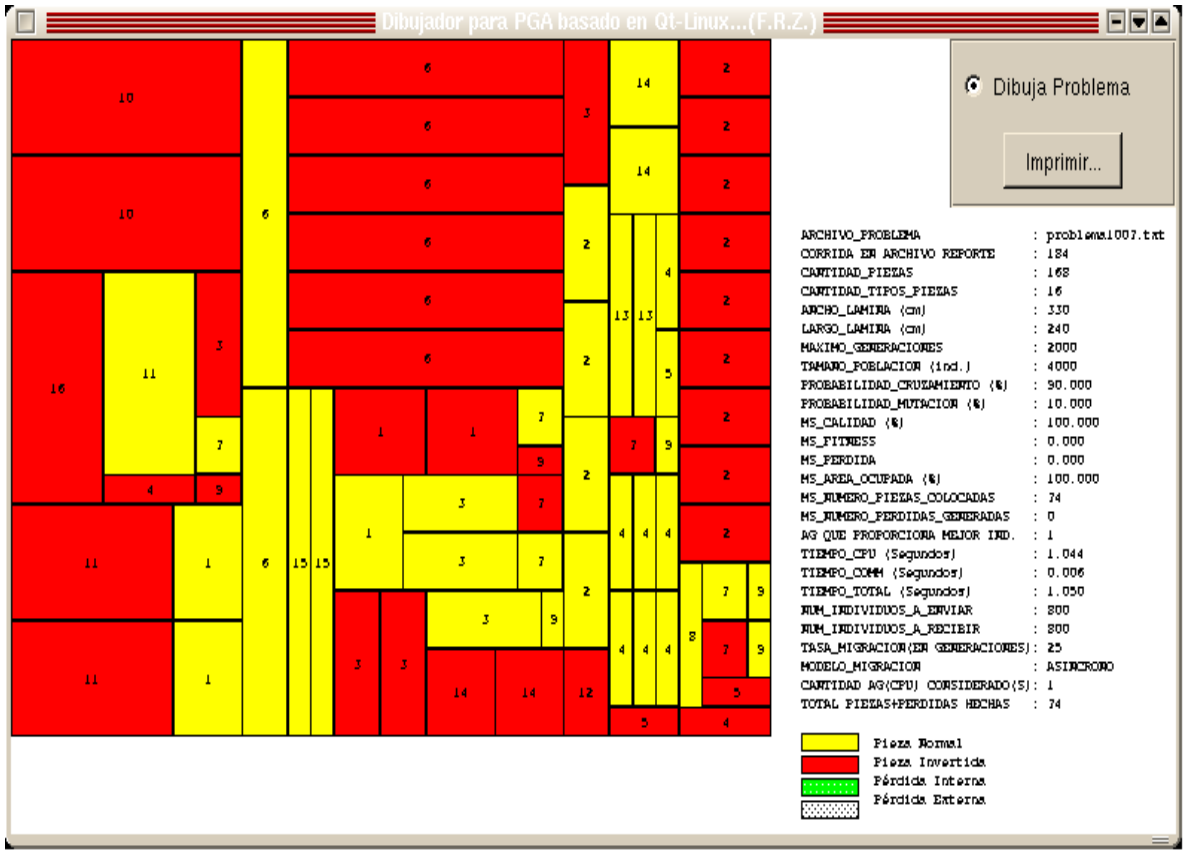
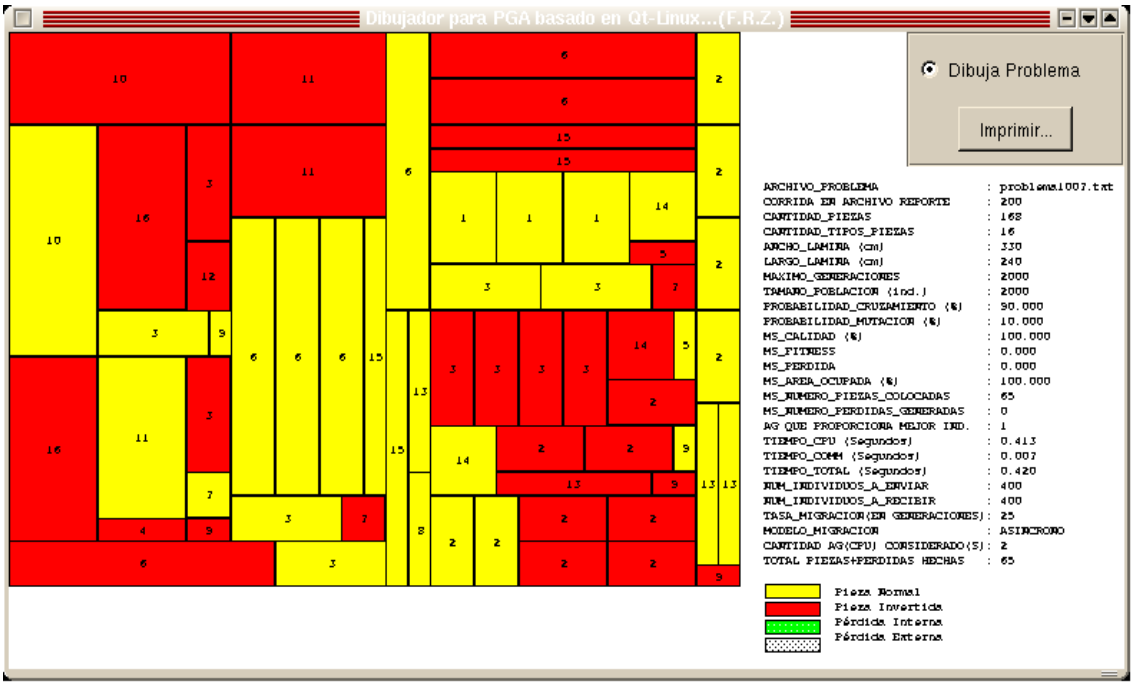


Figura N°D.16: Layout problema1007.txt, considerando un procesador y Migración Asíncrona.

La Figura N°D.17 muestra el "*layout*" resultante del Problema1007.txt, al ejecutar el AGP considerando dos procesadores con Migración Asíncrona y Síncrona.

a)



b)

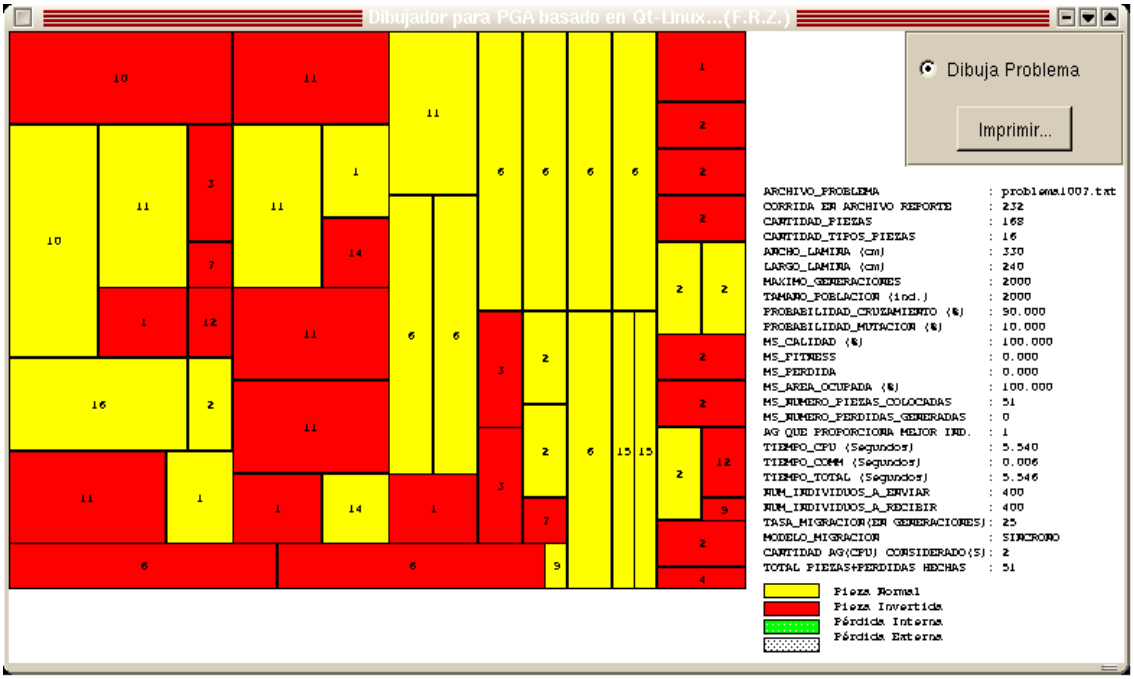
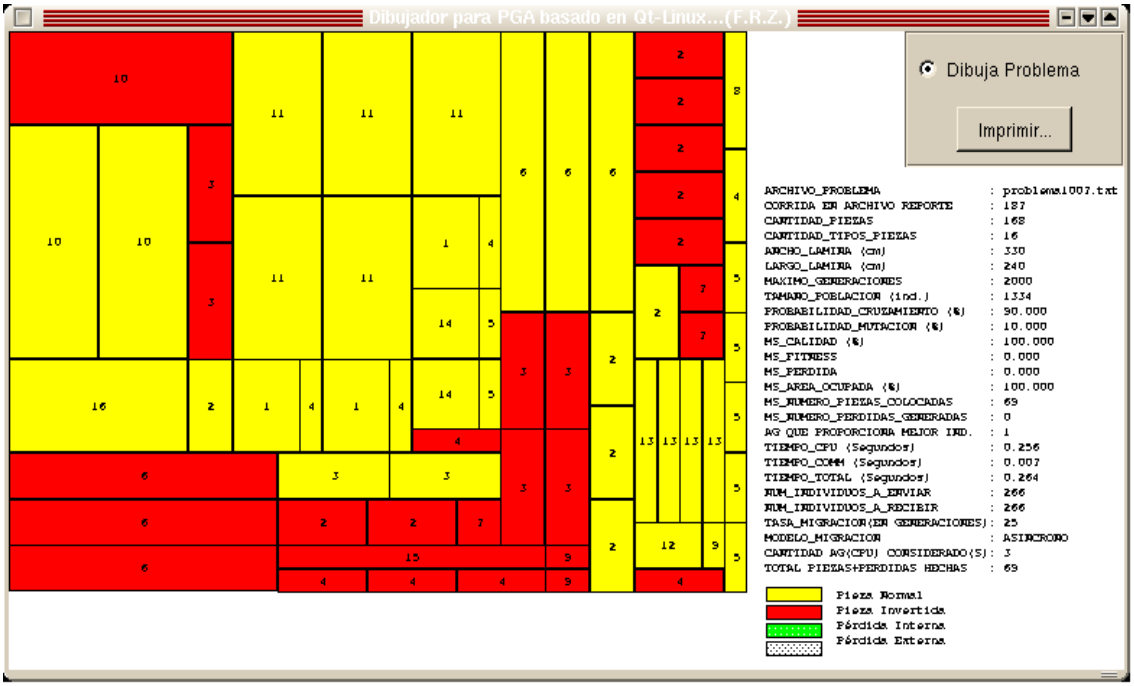


Figura N°D.17: Layout problema1007.txt, considerando dos procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.18 muestra el "*layout*" resultante del Problema1007.txt, al ejecutar el AGP considerando tres procesadores con Migración Asíncrona y Síncrona.

a)


Dibujador para PGA basado en Qt-Linux...(F.R.Z.)
Dibuja Problema
Imprimir...
ARCHIVO_PROBLEMA : problema1007.txt
CORRIDA EN ARCHIVO REPORTE : 187
CANTIDAD_PIEZAS : 168
CANTIDAD_TIPOS_PIEZAS : 16
ANCHO_LAMINA (cm) : 330
LARGO_LAMINA (cm) : 240
MAXIMO_GENERACIONES : 2000
TAMANO_POBLACION (ind.) : 1334
PROBABILIDAD_CRUZAMIENTO (%) : 90.000
PROBABILIDAD_MUTACION (%) : 10.000
MS_CALIDAD (%) : 100.000
MS_FITNESS : 0.000
MS_PERDIDA : 0.000
MS_AREA_OCUPADA (%) : 100.000
MS_NUMERO_PIEZAS_COLOCADAS : 69
MS_NUMERO_PERDIDAS_GENERADAS : 0
AG QUE PROPORCIONA MEJOR IND. : 1
TIEMPO_CPU (Segundos) : 0.256
TIEMPO_COMM (Segundos) : 0.007
TIEMPO_TOTAL (Segundos) : 0.264
NUM_INDIVIDUOS_A_ENVIAR : 266
NUM_INDIVIDUOS_A_RECIBIR : 266
TASA_MIGRACION(EN GENERACIONES) : 25
MODELO_MIGRACION : ASINCRONO
CANTIDAD AG(CPU) CONSIDERADO(S) : 3
TOTAL PIEZAS+PERDIDAS HECHAS : 69
Pieza Normal
Pieza Invertida
Pérdida Interna
Pérdida Externa

b)

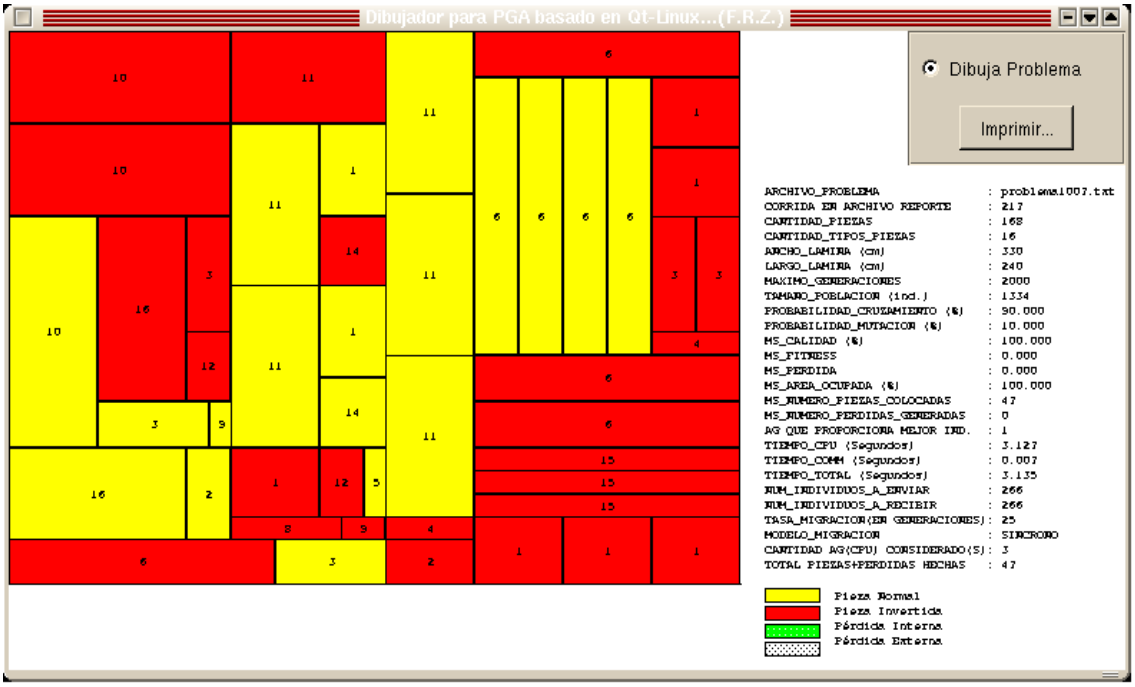
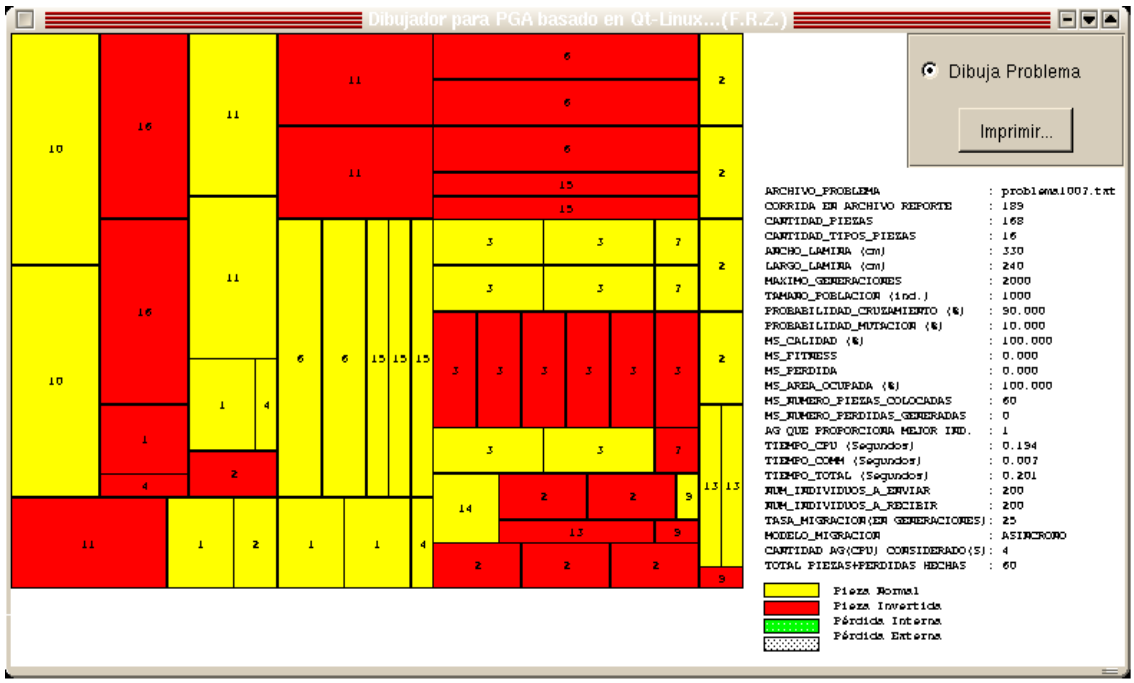


Figura N°D.18: Layout problema1007.txt, considerando tres procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.19 muestra el "*layout*" resultante del Problema1007.txt, al ejecutar el AGP considerando cuatro procesadores con Migración Asíncrona y Síncrona.

a)


Dibujador para PGA basado en Qt-Linux...(F.R.Z.)
Dibuja Problema
Imprimir...
ARCHIVO_PROBLEMA : problemas1007.txt
CORRIDA EN ARCHIVO REPORTE : 189
CANTIDAD_PIEZAS : 168
CANTIDAD_TIPOS_PIEZAS : 16
ANCHO_LAMINA (cm) : 330
LARGO_LAMINA (cm) : 240
MAXIMO_GENERACIONES : 2000
TAMANO_POBLACION (ind.) : 1000
PROBABILIDAD_CRUZAMIENTO (%) : 90.000
PROBABILIDAD_MUTACION (%) : 10.000
MS_CALIDAD (%) : 100.000
MS_FITNESS : 0.000
MS_PERDIDA : 0.000
MS_AREA_OCUPADA (%) : 100.000
MS_NUMERO_PIEZAS_COLOCADAS : 60
MS_NUMERO_PERDIDAS_GENERADAS : 0
AG QUE PROPORCIONA MEJOR IND. : 1
TIEMPO_CPU (Segundos) : 0.194
TIEMPO_COMM (Segundos) : 0.007
TIEMPO_TOTAL (Segundos) : 0.201
NUM_INDIVIDUOS_A_ENVIAR : 200
NUM_INDIVIDUOS_A_RECIBIR : 200
TASA_MIGRACION(EN GENERACIONES) : 25
MODELO_MIGRACION : ASINCRONO
CANTIDAD AG'S(CPU) CONSIDERADO(S) : 4
TOTAL PIEZAS+PERDIDAS HECHAS : 60
Pieza Normal
Pieza Invertida
Pérdida Interna
Pérdida Externa

b)

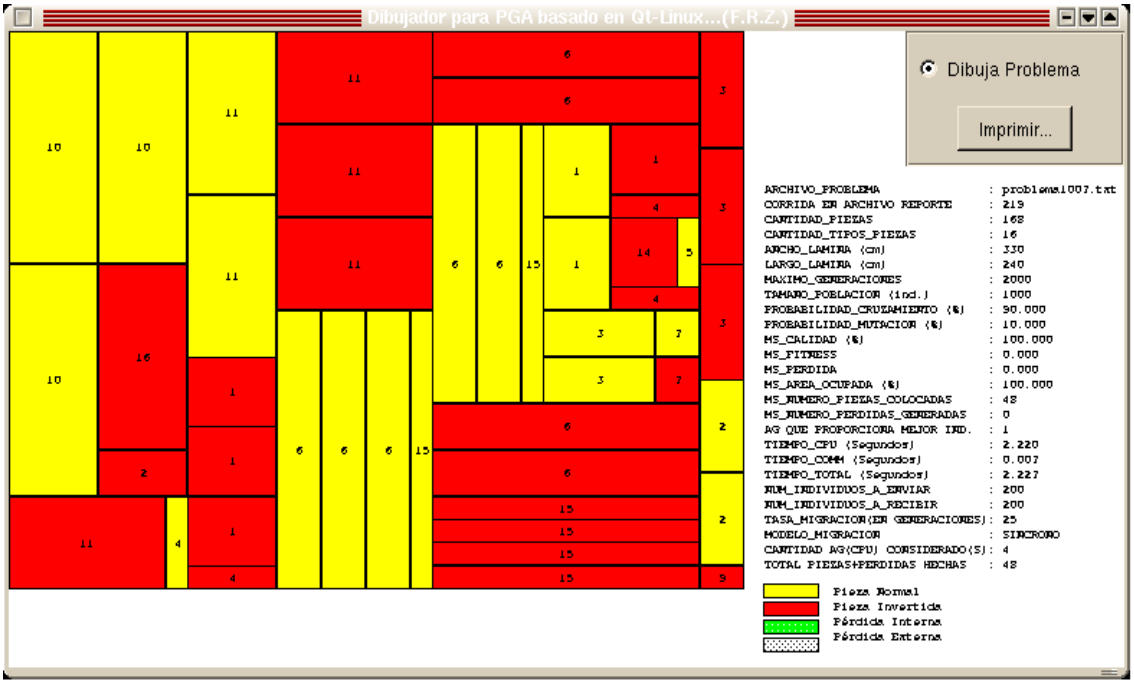
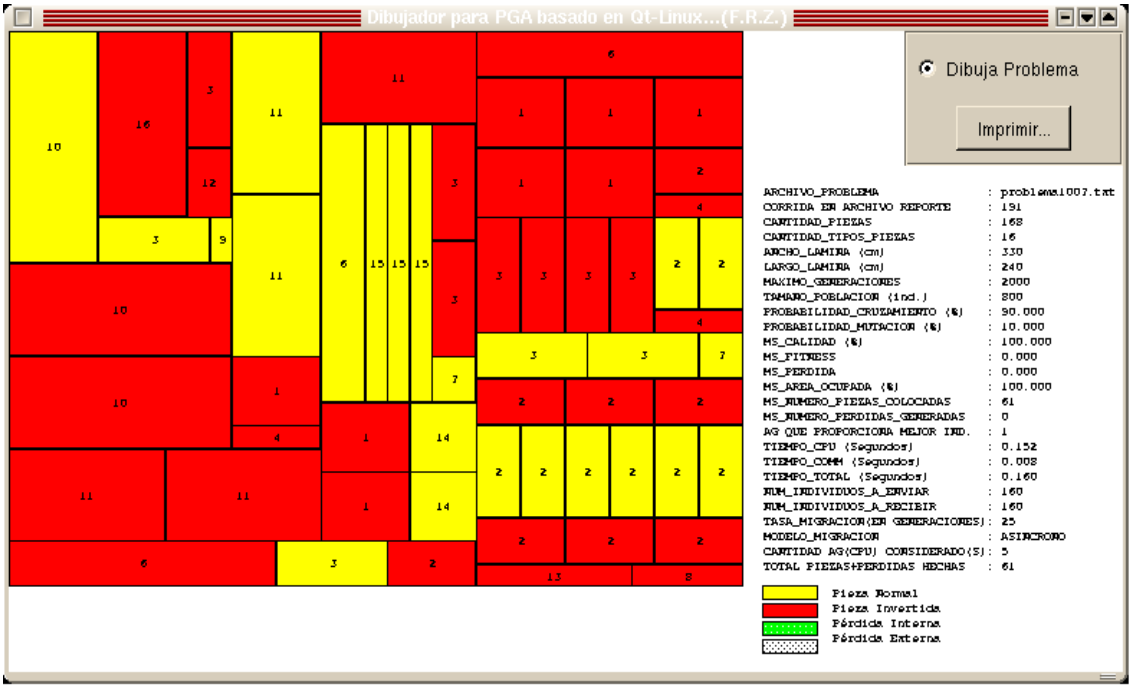


Figura N°D.19: Layout problema1007.txt, considerando cuatro procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.20 muestra el "*layout*" resultante del Problema1007.txt, al ejecutar el AGP considerando cinco procesadores con Migración Asíncrona y Síncrona.

a)


Dibujador para PGA basado en Qt-Linux...(F.R.Z.)
Dibuja Problema
Imprimir...
ARCHIVO_PROBLEMA : problema1007.txt
CORRIDA EN ARCHIVO REPORTE : 191
CANTIDAD_PIEZAS : 168
CANTIDAD_TIPOS_PIEZAS : 16
ANCHO_LAMINA (cm) : 330
LARGO_LAMINA (cm) : 240
MAXIMO_GENERACIONES : 2000
TAMANO_POBLACION (ind.) : 800
PROBABILIDAD_CRUZAMIENTO (%) : 90.000
PROBABILIDAD_MUTACION (%) : 10.000
MS_CALIDAD (%) : 100.000
MS_FITNESS : 0.000
MS_PERDIDA : 0.000
MS_AREA_OCUPADA (%) : 100.000
MS_NUMERO_PIEZAS_COLOCADAS : 61
MS_NUMERO_PERDIDAS_GENERADAS : 0
AG QUE PROPORCIONA MEJOR IND. : 1
TIEMPO_CPU (Segundos) : 0.152
TIEMPO_COMM (Segundos) : 0.008
TIEMPO_TOTAL (Segundos) : 0.160
NUM_INDIVIDUOS_A_ENVIAR : 160
NUM_INDIVIDUOS_A_RECIBIR : 160
TASA_MIGRACION(EN GENERACIONES) : 25
MODELO_MIGRACION : ASINCRONO
CANTIDAD AG(CPU) CONSIDERADO(S) : 5
TOTAL PIEZAS+PERDIDAS HECHAS : 61
Pieza Normal
Pieza Invertida
Pérdida Interna
Pérdida Externa

b)

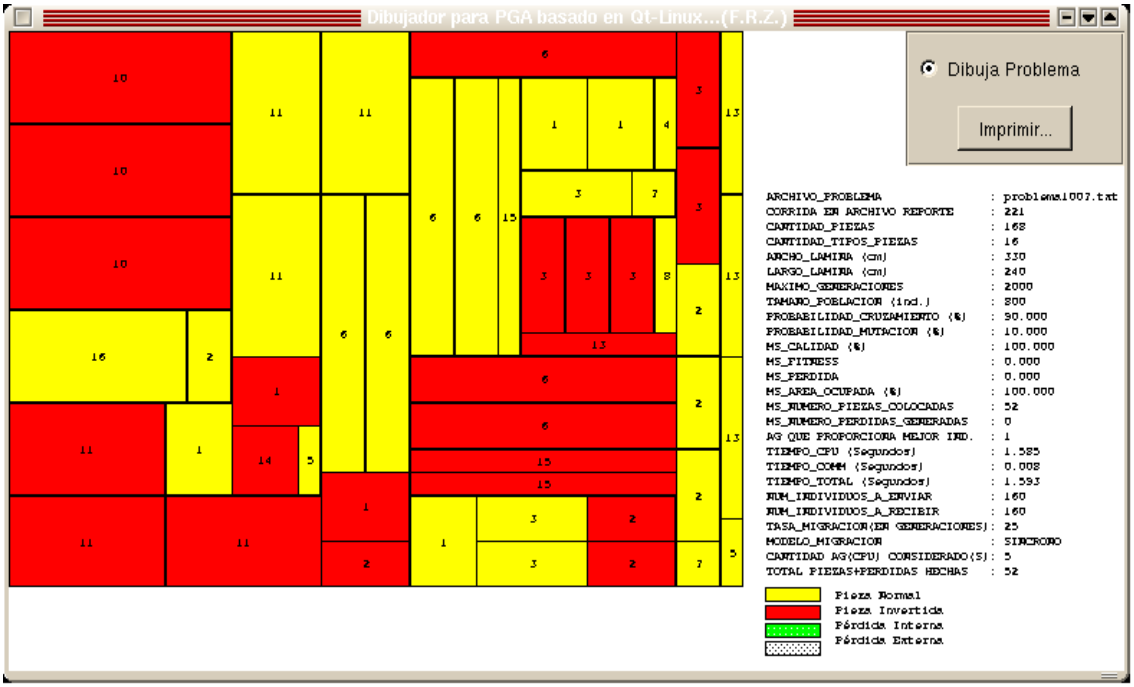


Figura N°D.20: Layout problema1007.txt, considerando cinco procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.21 muestra el "*layout*" resultante del Problema1008.txt, al ejecutar el AGP considerando un procesador y Migración Asíncrona.

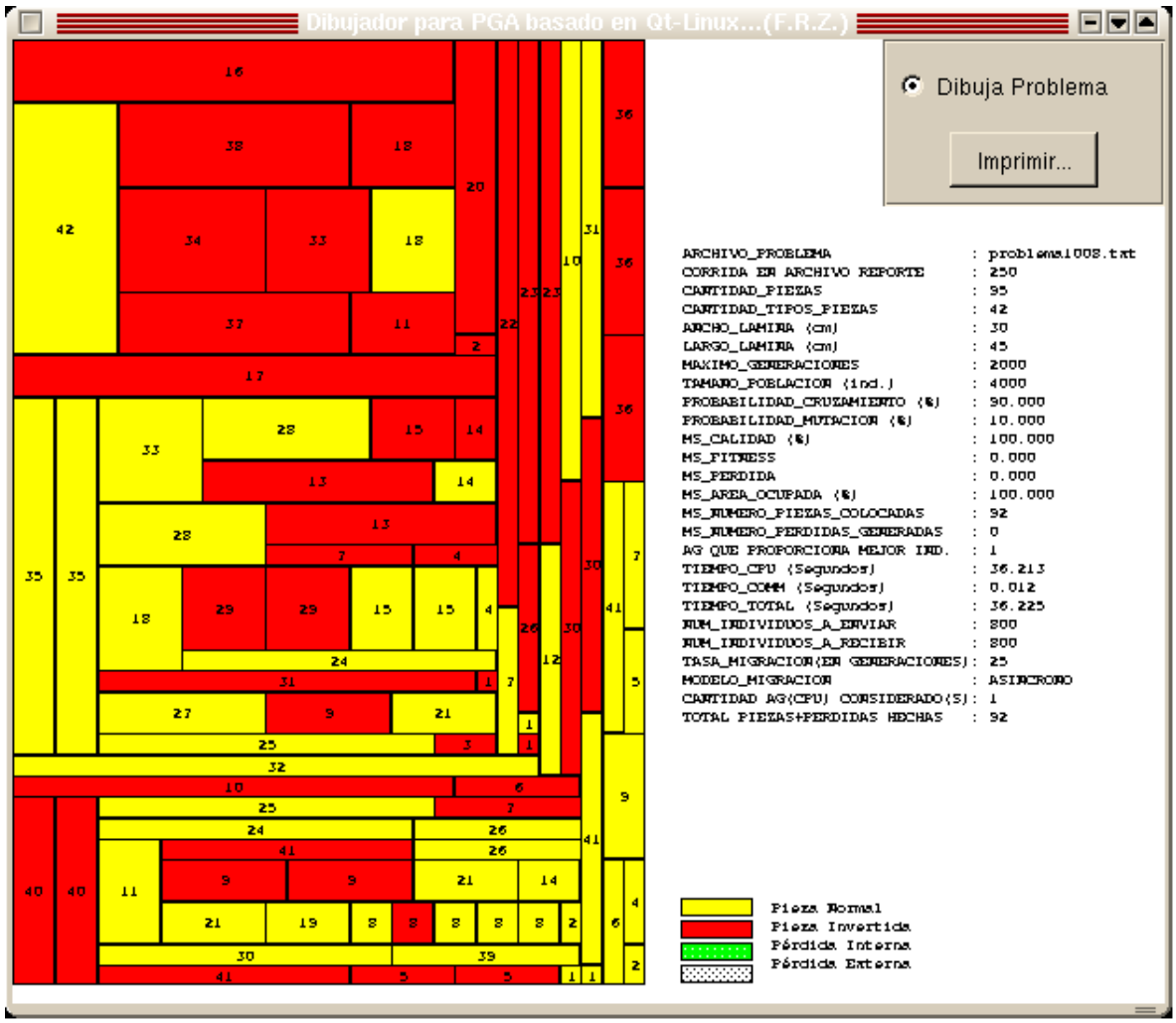
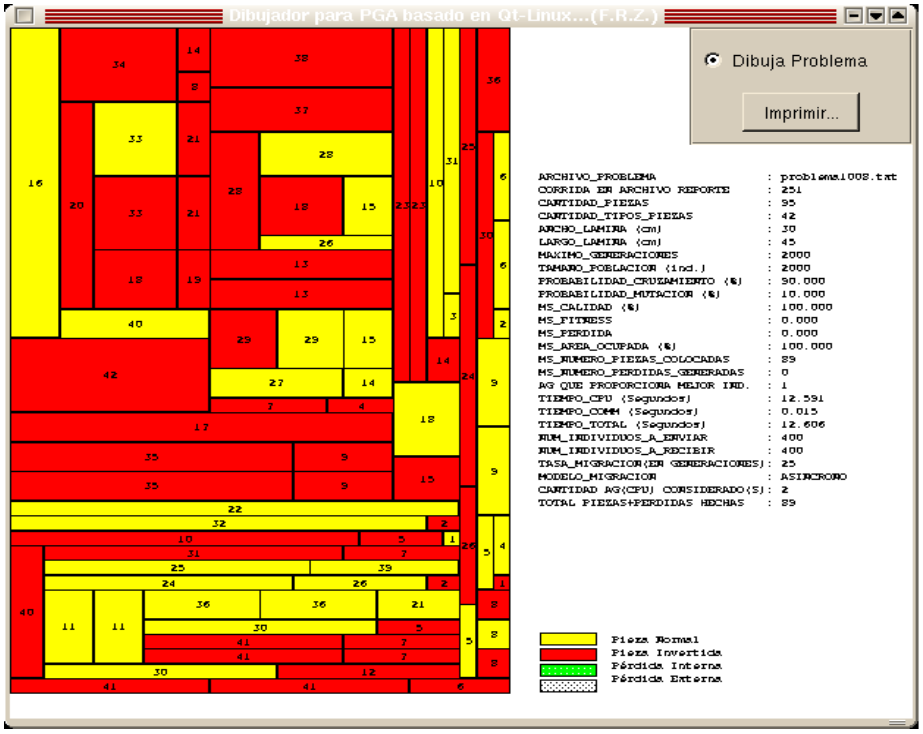


Figura N°D.21: Layout problema1008.txt, considerando un procesador y Migración Asíncrona.

La Figura N°D.22 muestra el "*layout*" resultante del Problema1008.txt, al ejecutar el AGP considerando dos procesadores con Migración Asíncrona y Síncrona.

a)



b)

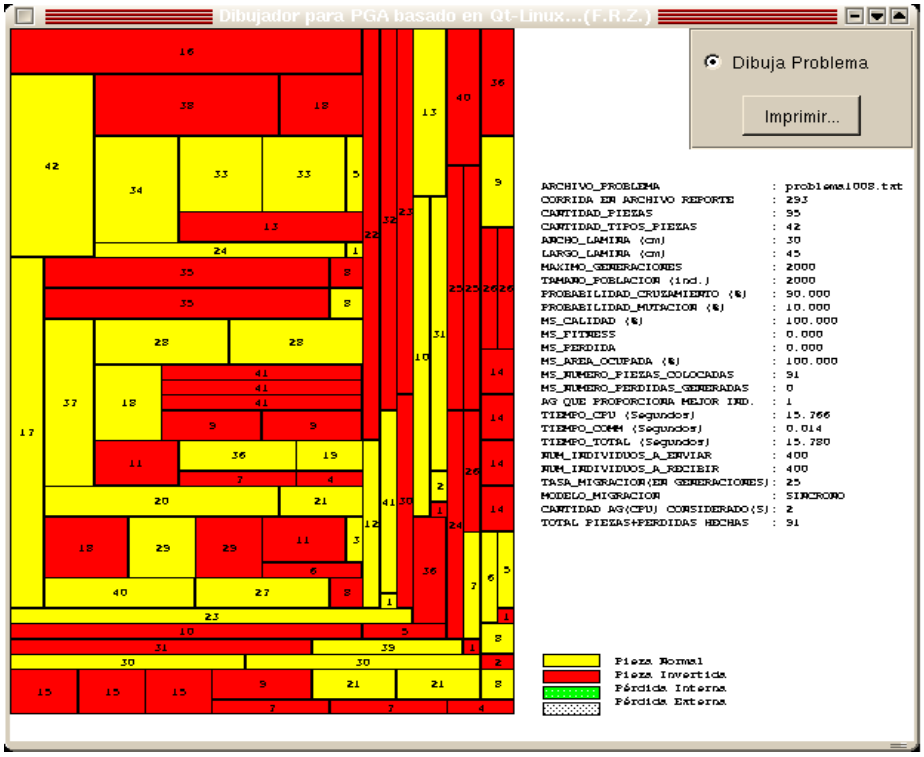
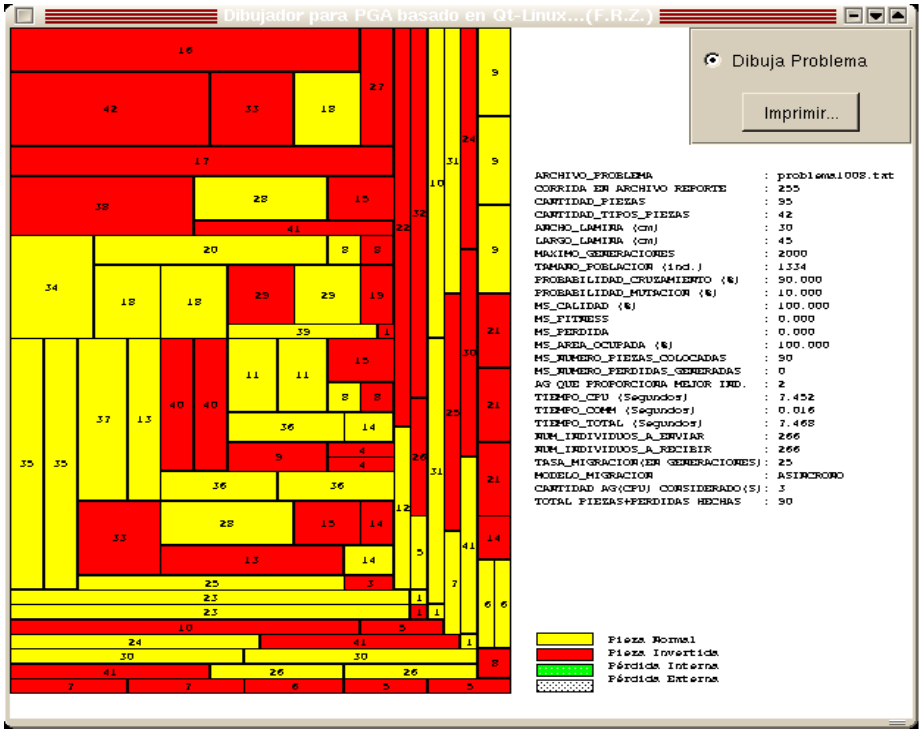


Figura N°D.22: Layout problema1008.txt, considerando dos procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.23 muestra el "*layout*" resultante del Problema1008.txt, al ejecutar el AGP considerando tres procesadores con Migración Asíncrona y Síncrona.

a) 

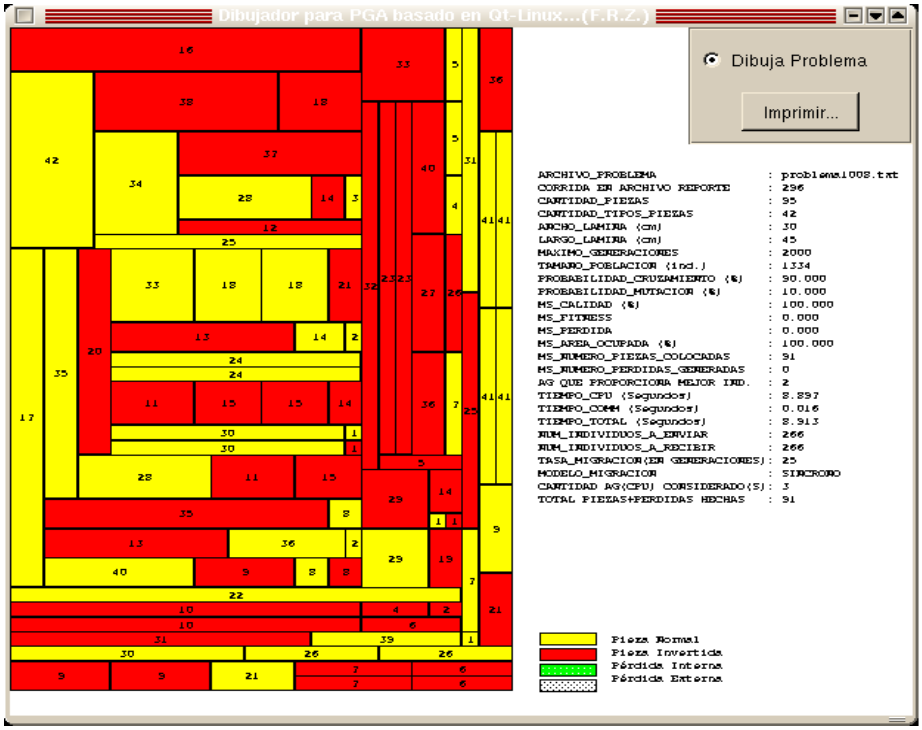
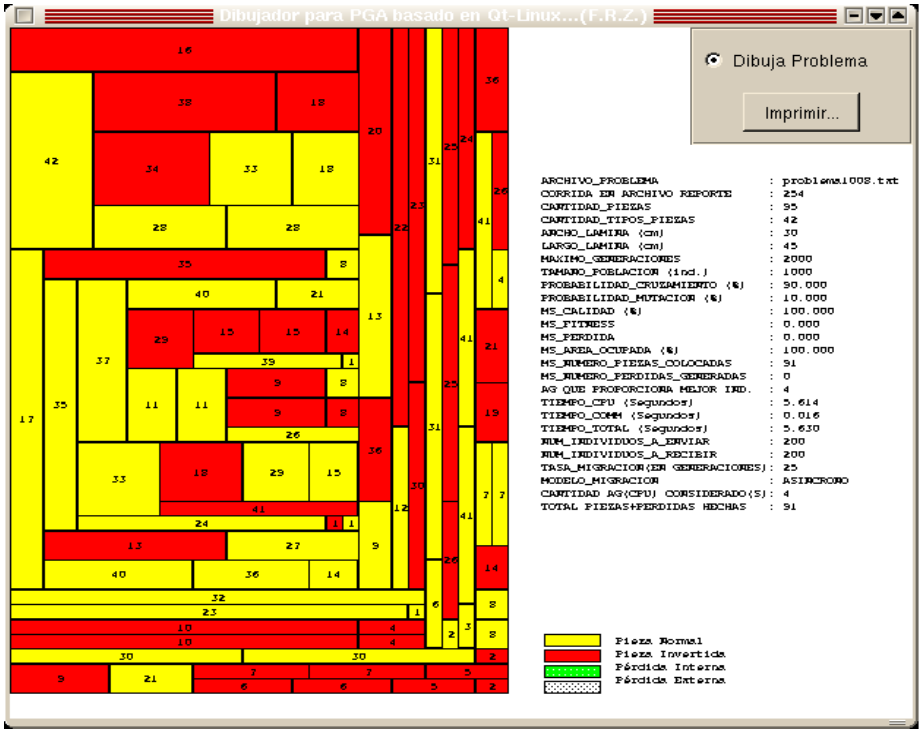
b) 

Figura N°D.23: Layout problema1008.txt, considerando tres procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.24 muestra el "*layout*" resultante del Problema1008.txt, al ejecutar el AGP considerando cuatro procesadores con Migración Asíncrona y Síncrona.

a)



b)

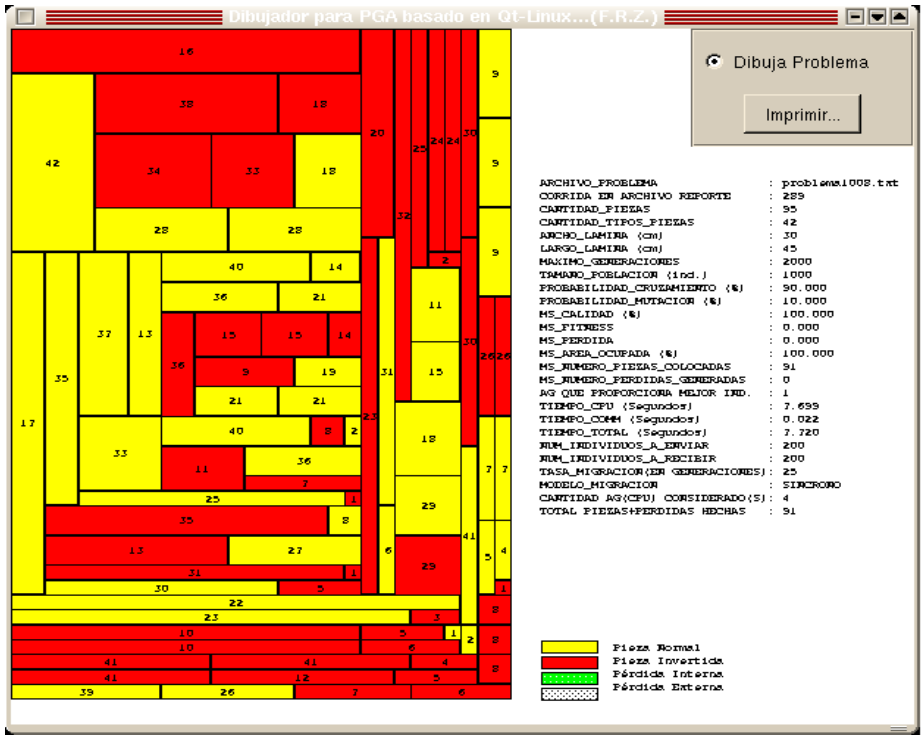
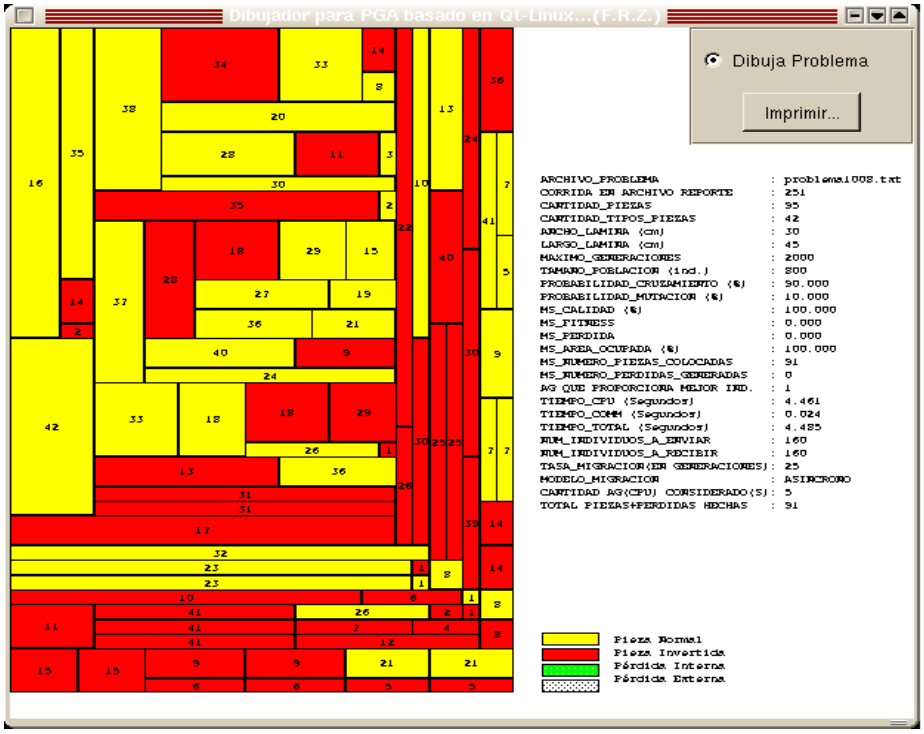


Figura N°D.24: Layout problema1008.txt, considerando cuatro procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.25 muestra el "*layout*" resultante del Problema1008.txt, al ejecutar el AGP considerando cinco procesadores

con Migración Asíncrona y Síncrona.

a)



b)

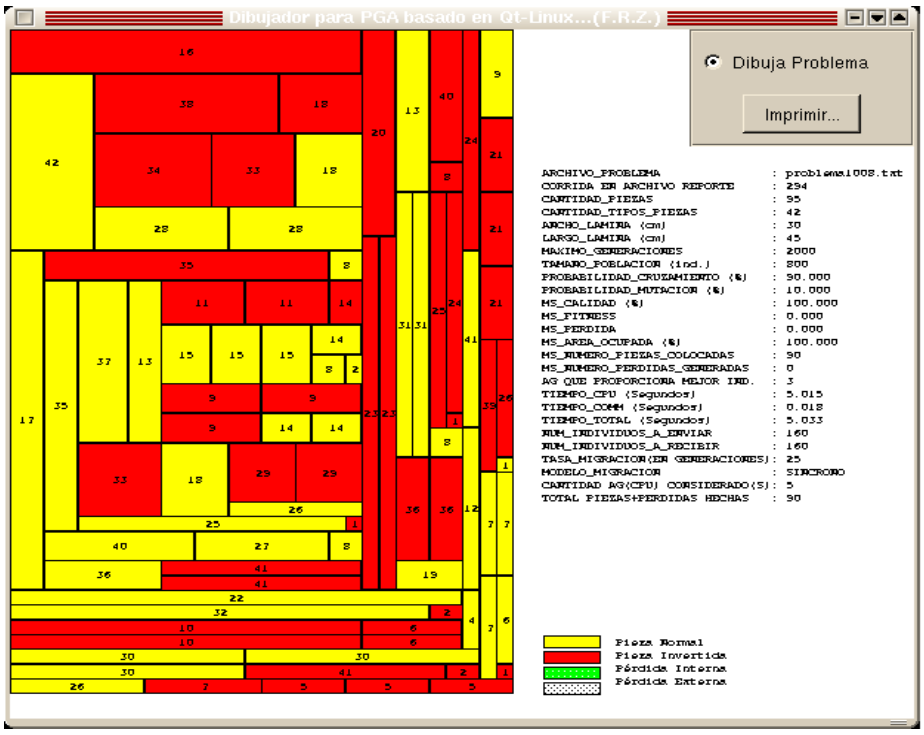


Figura N°D.25: Layout problema1008.txt, considerando cinco procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.26 muestra el "*layout*" resultante del Problema2025.txt, al ejecutar el AGP considerando un procesador y Migración Asíncrona.

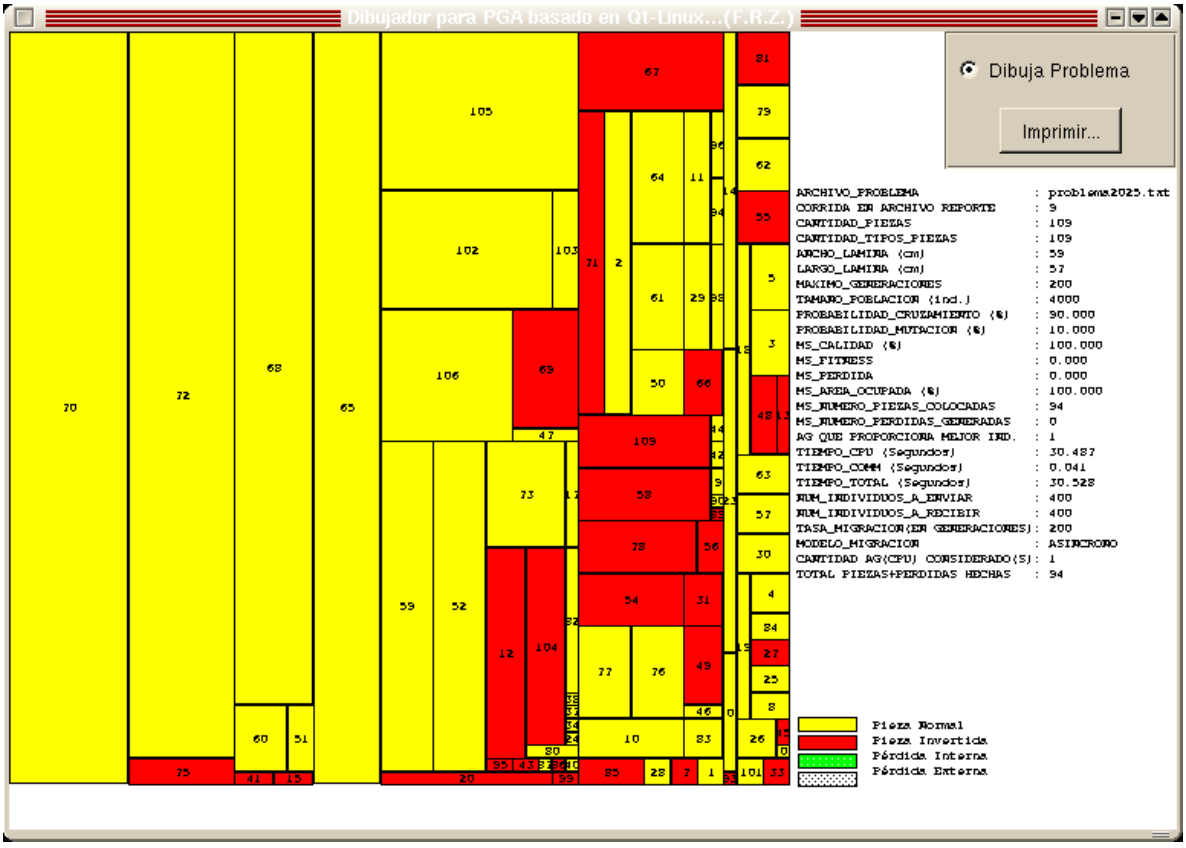
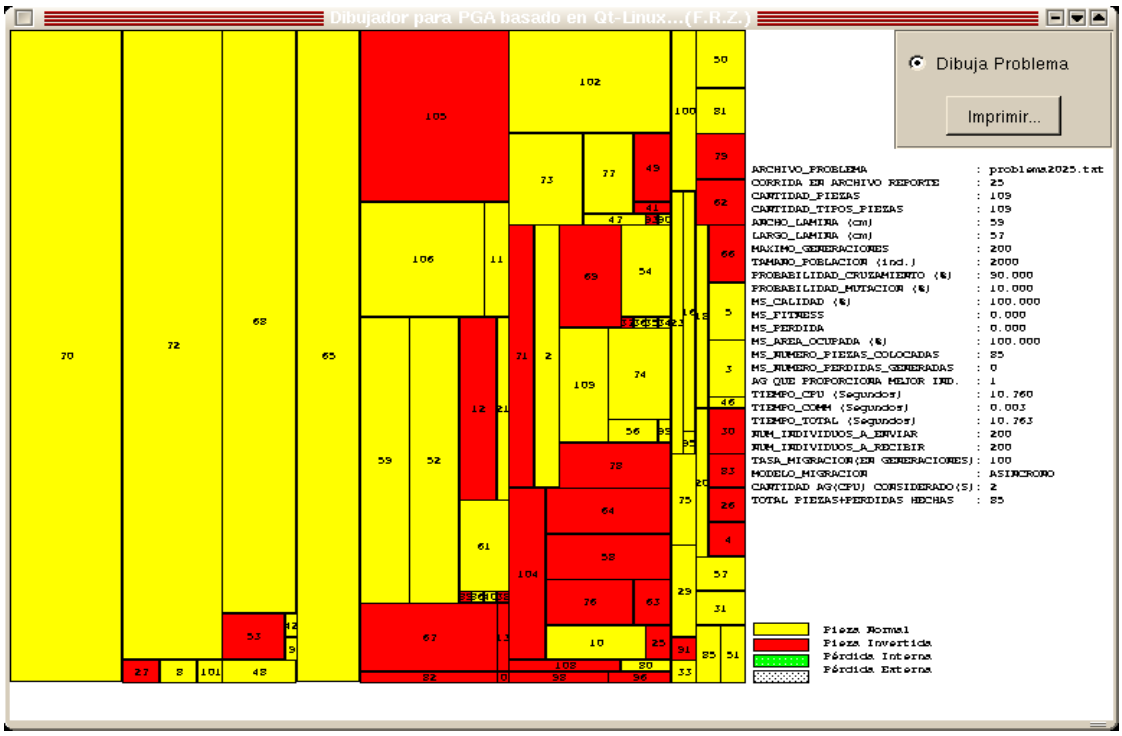


Figura N°D.26: Layout problema2025.txt, considerando un procesador y Migración Asíncrona.

La Figura N°D.27 muestra el "*layout*" resultante del Problema2025.txt, al ejecutar el AGP considerando dos procesadores con Migración Asíncrona y Síncrona.

a)



b)

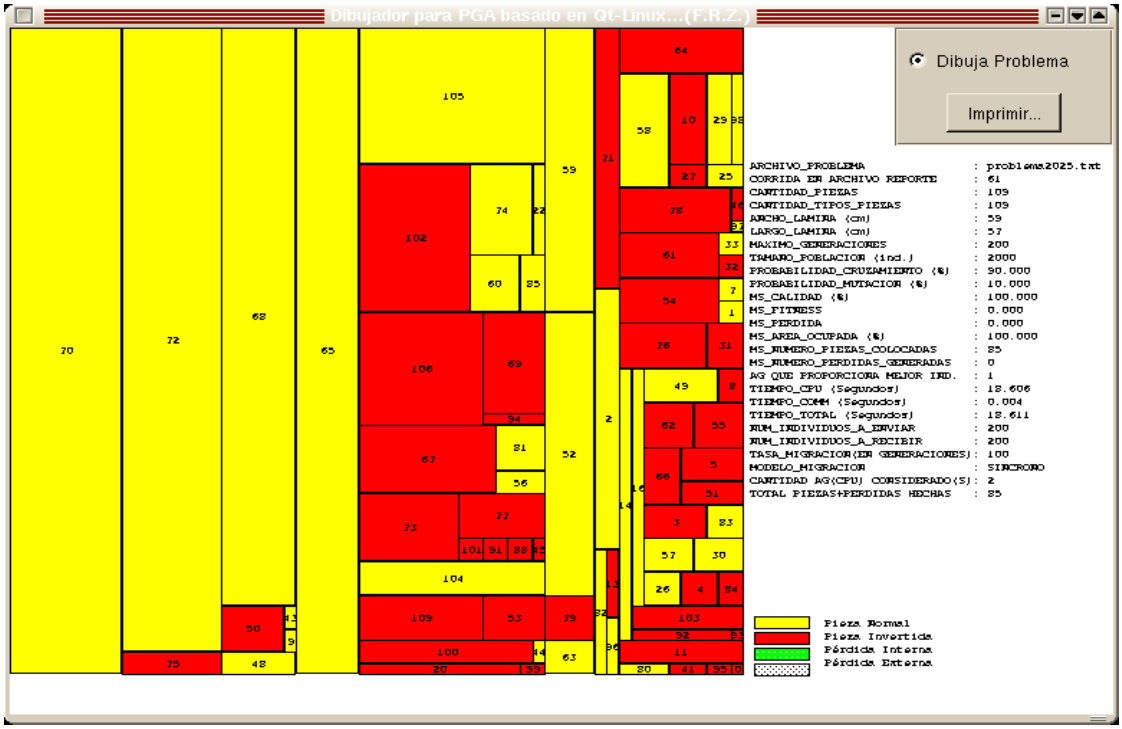
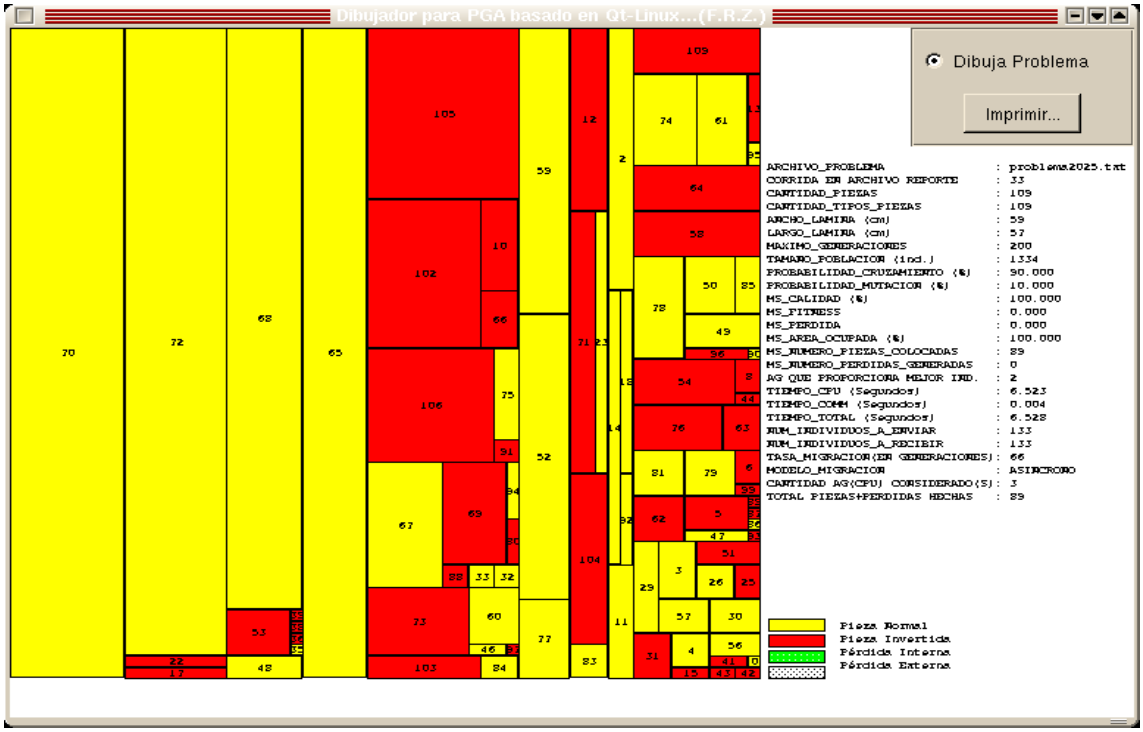


Figura N°D.27: Layout problema2025.txt, considerando dos procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.28 muestra el "*layout*" resultante del Problema2025.txt, al ejecutar el AGP considerando tres procesadores con Migración Asíncrona y Síncrona.

a)


Dibujador para PGA basado en Qt-Linux...(F.R.Z.)
Dibuja Problema
Imprimir...
ARCHIVO_PROBLEMA : problema2025.txt
CORRIDA EN ARCHIVO REPORTE : 33
CANTIDAD_PIEZAS : 109
CANTIDAD_TIPOS_PIEZAS : 109
ANCHO_LAMINA (cm) : 58
LARGO_LAMINA (cm) : 57
MAXIMO_GENERACIONES : 200
TAMANO_POBLACION (ind.) : 1334
PROBABILIDAD_CRUZAMIENTO (%) : 90.000
PROBABILIDAD_MUTACION (%) : 10.000
MS_CALIDAD (%) : 100.000
MS_FITNESS : 0.000
MS_PERDIDA : 0.000
MS_AREA_OCUPADA (%) : 100.000
MS_NUMERO_PIEZAS_COLOCADAS : 89
MS_NUMERO_PERDIDAS_GENERADAS : 0
AG QUE PROPORCIONA MEJOR IND. : 2
TIEMPO_CPU (Segundos) : 6.523
TIEMPO_COMM (Segundos) : 0.004
TIEMPO_TOTAL (Segundos) : 6.528
NUM_INDIVIDUOS_A_ENVIAR : 133
NUM_INDIVIDUOS_A_RECIBIR : 133
TASA_MIGRACION(EN GENERACIONES) : 66
MODELO_MIGRACION : ASINCRONO
CANTIDAD AG(CPU) CONSIDERADO(S) : 3
TOTAL PIEZAS+PERDIDAS HECHAS : 89
Pieza Normal
Pieza Invertida
Pérdida Interna
Pérdida Externa

b)

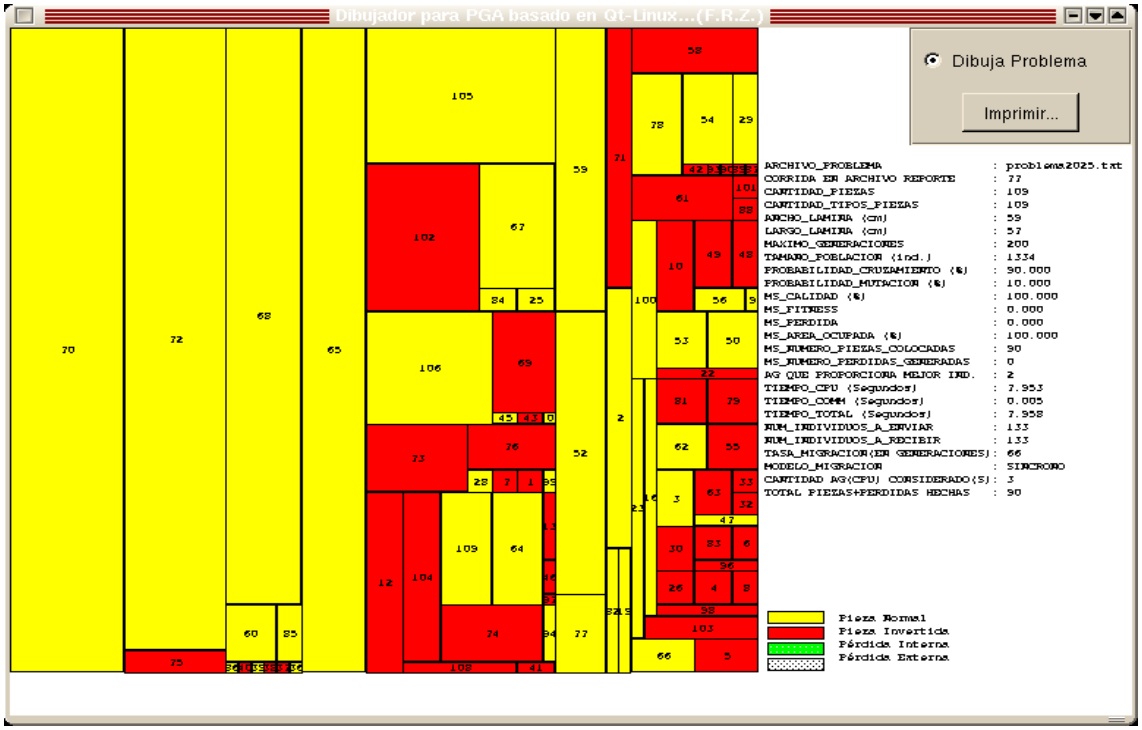
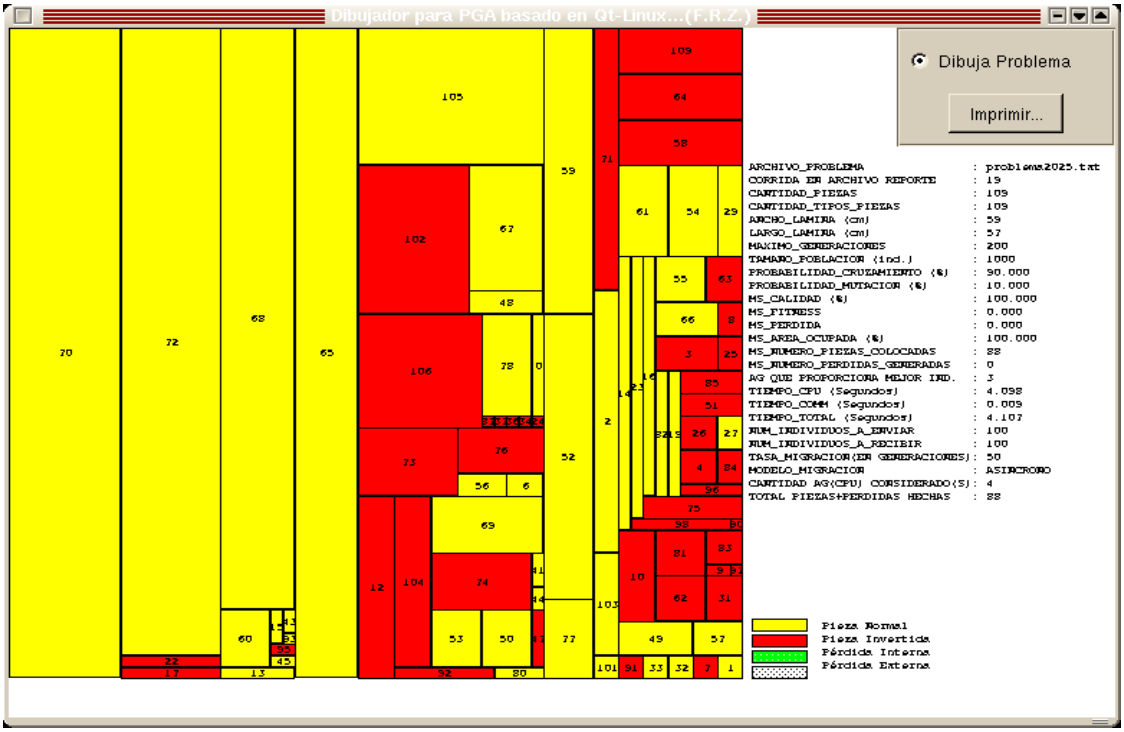


Figura N°D.28: Layout problema2025.txt, considerando tres procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.29 muestra el "*layout*" resultante del Problema2025.txt, al ejecutar el AGP considerando cuatro procesadores con Migración Asíncrona y Síncrona.

a)



b)

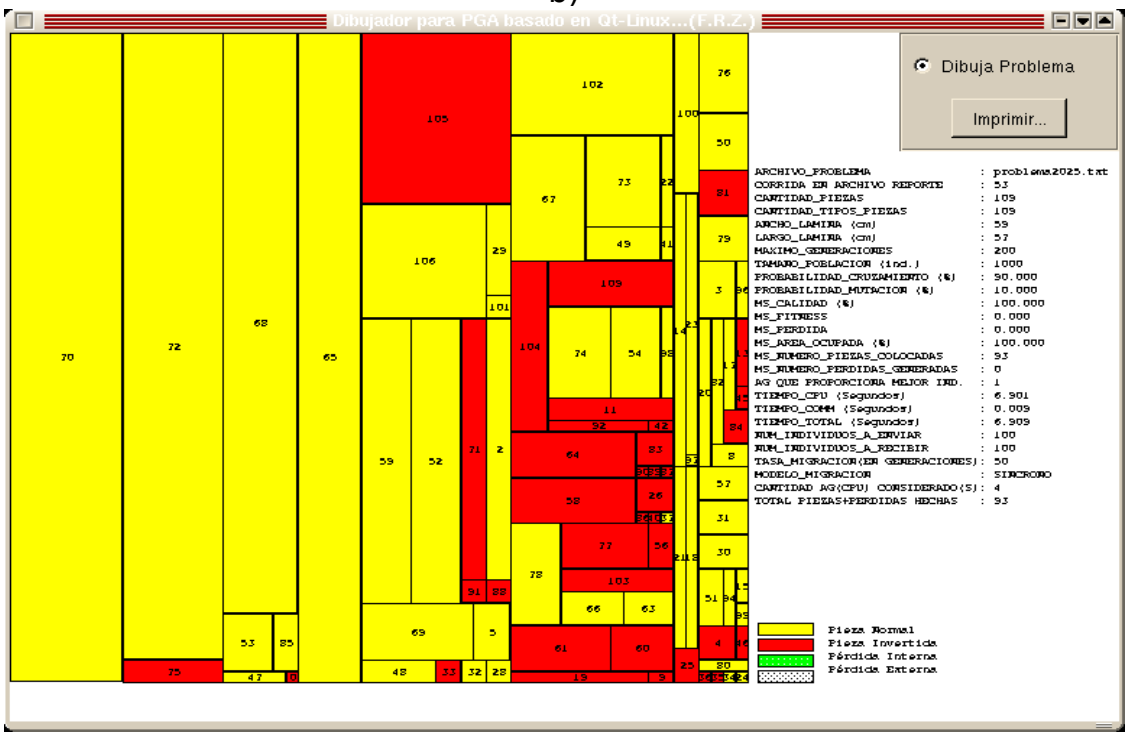
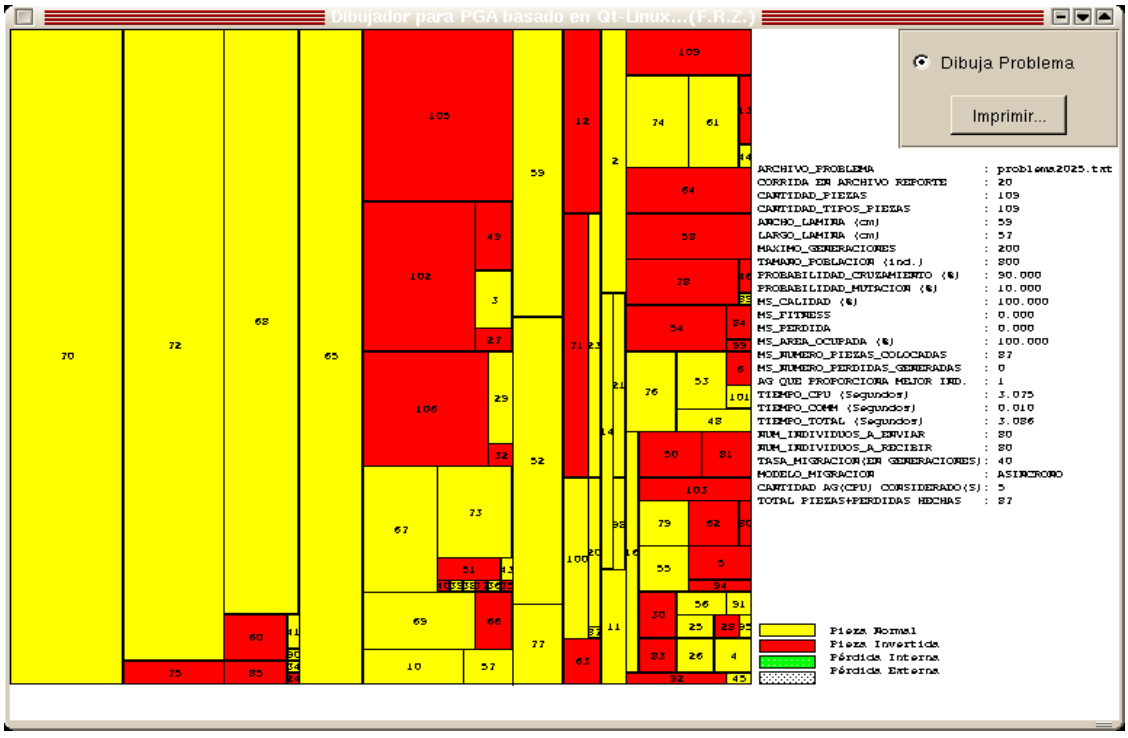


Figura N°D.29: Layout problema2025.txt, considerando cuatro procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.30 muestra el "*layout*" resultante del Problema2025.txt, al ejecutar el AGP considerando cinco procesadores con Migración Asíncrona y Síncrona.

a)



b)

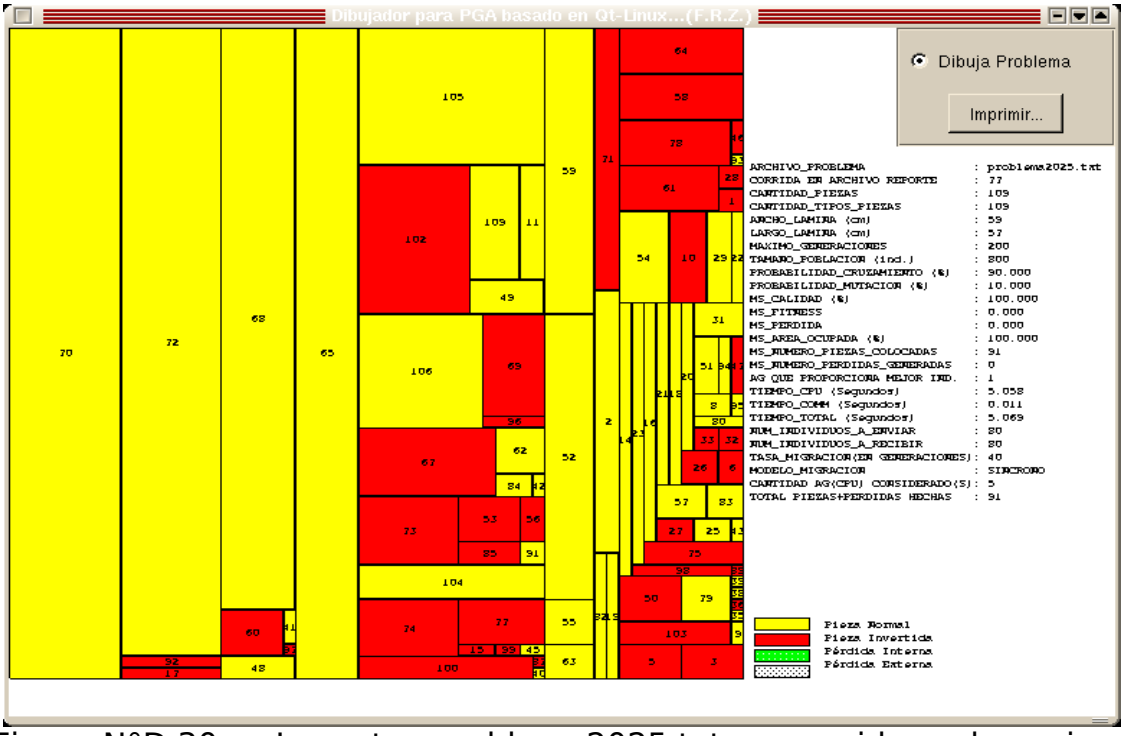


Figura N°D.30: Layout problema2025.txt, considerando cinco procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.31 muestra el "*layout*" resultante del Problema2026.txt, al ejecutar el AGP considerando un procesador y Migración Asíncrona.

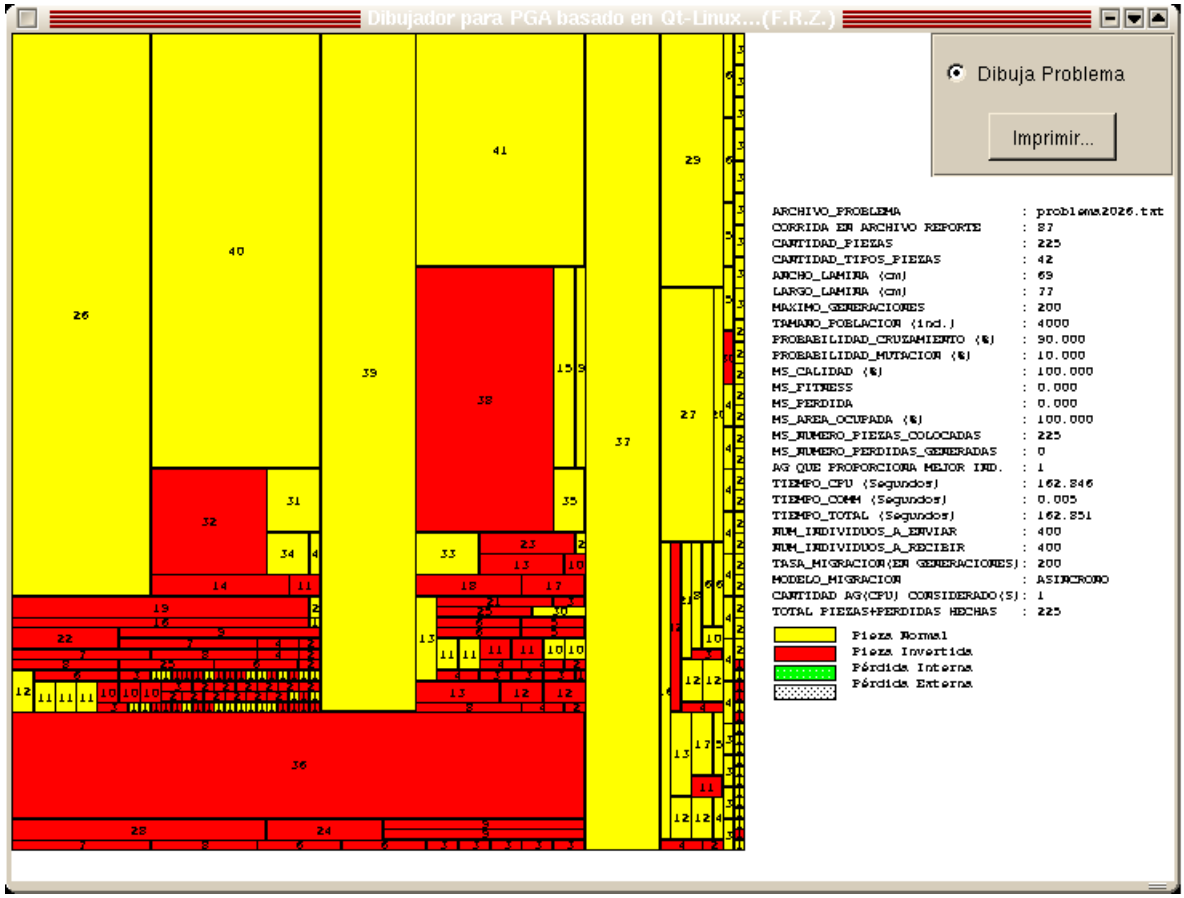
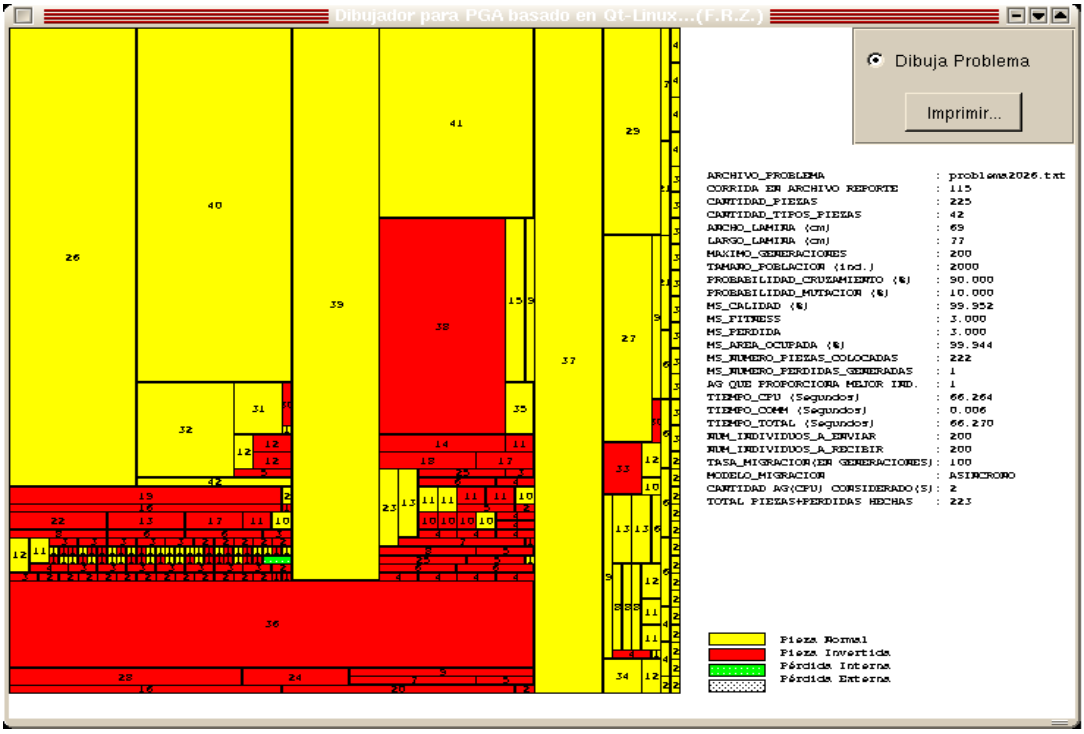


Figura N°D.31: Layout problema2026.txt, considerando un procesador y Migración Asíncrona.

La Figura N°D.32 muestra el "*layout*" resultante del Problema2026.txt, al ejecutar el AGP considerando dos procesadores con Migración Asíncrona y Síncrona.

a)



b)

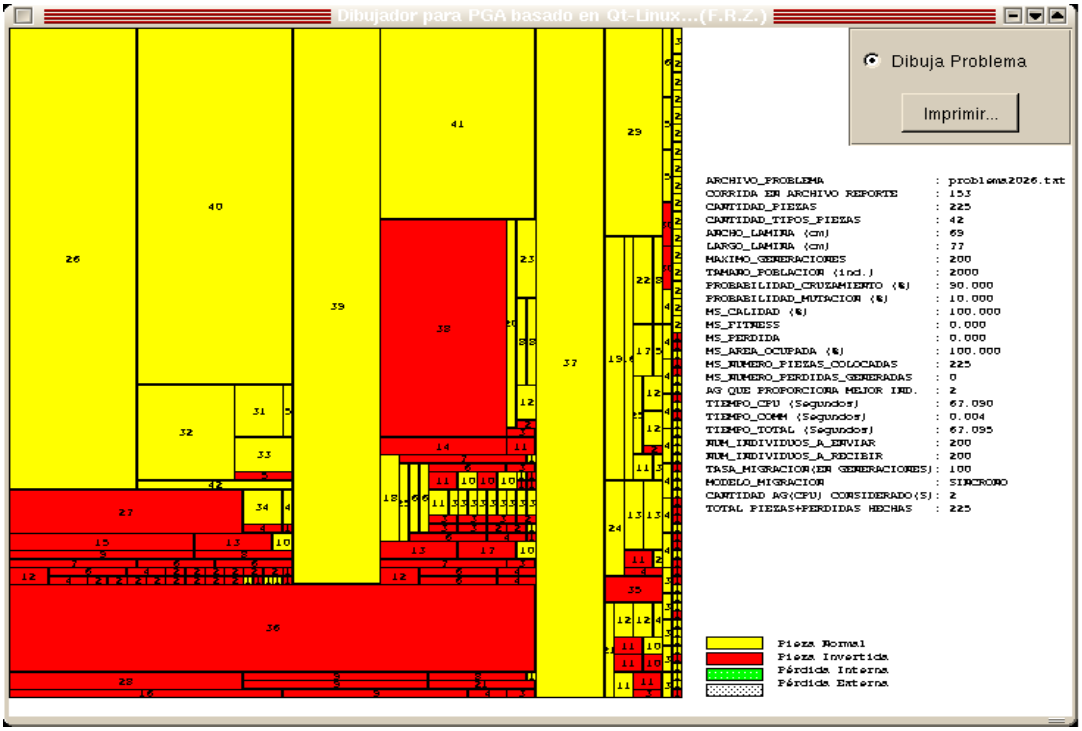
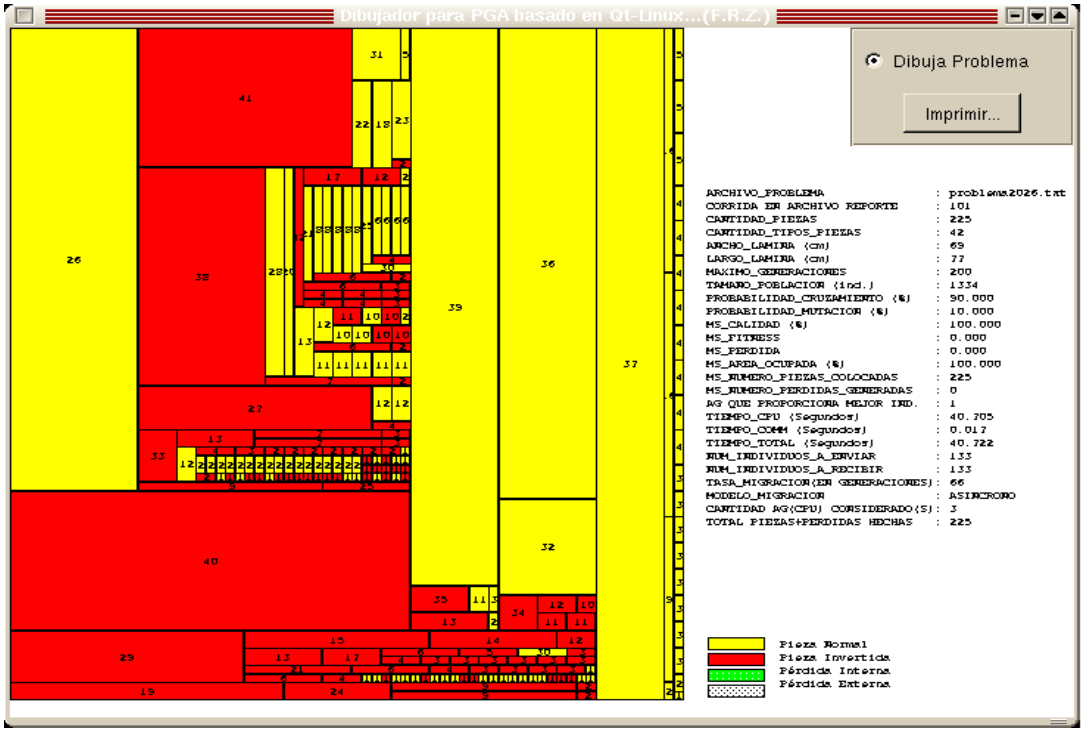
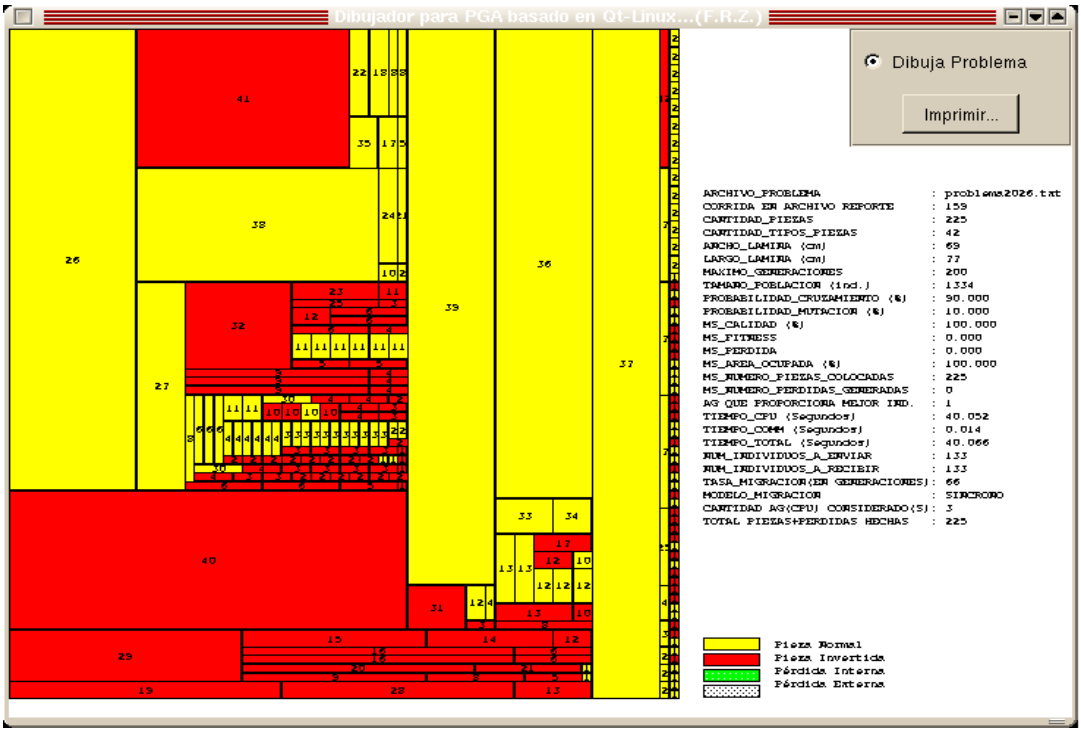


Figura N°D.32: Layout problema2026.txt, considerando dos procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.33 muestra el "*layout*" resultante del Problema2026.txt, al ejecutar el AGP considerando tres procesadores con Migración Asíncrona y Síncrona.



a)

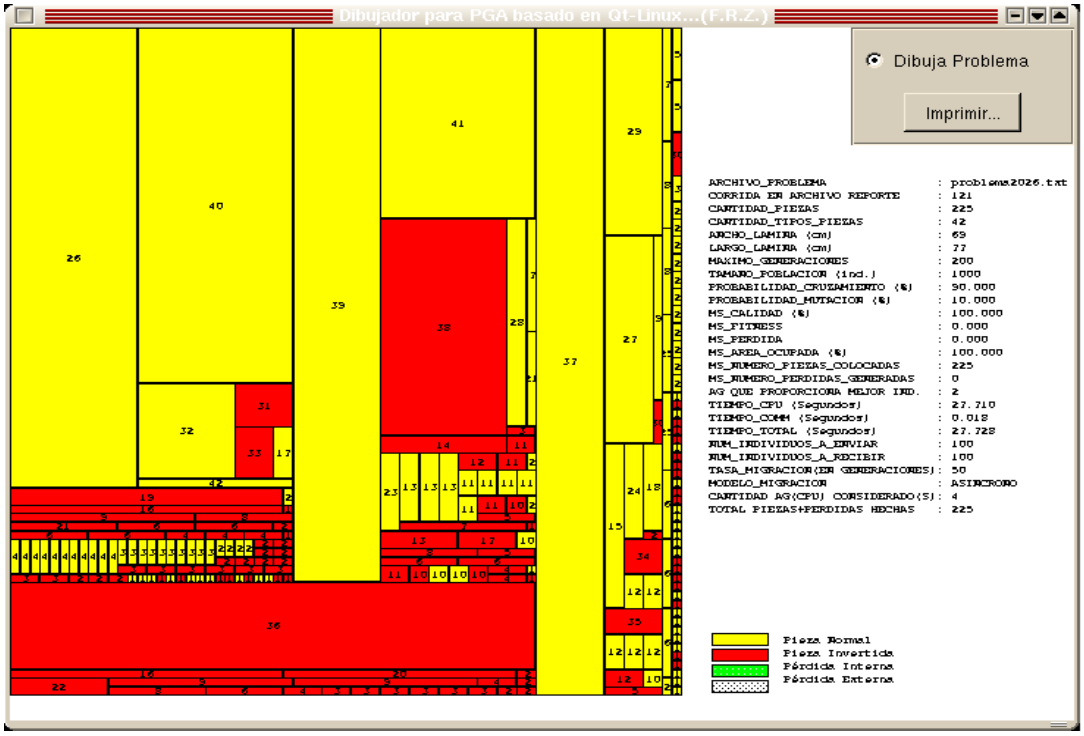


b)

Figura N°D.33: Layout problema2026.txt, considerando tres procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.34 muestra el "*layout*" resultante del Problema2026.txt, al ejecutar el AGP considerando cuatro procesadores con Migración Asíncrona y Síncrona.

a)



b)

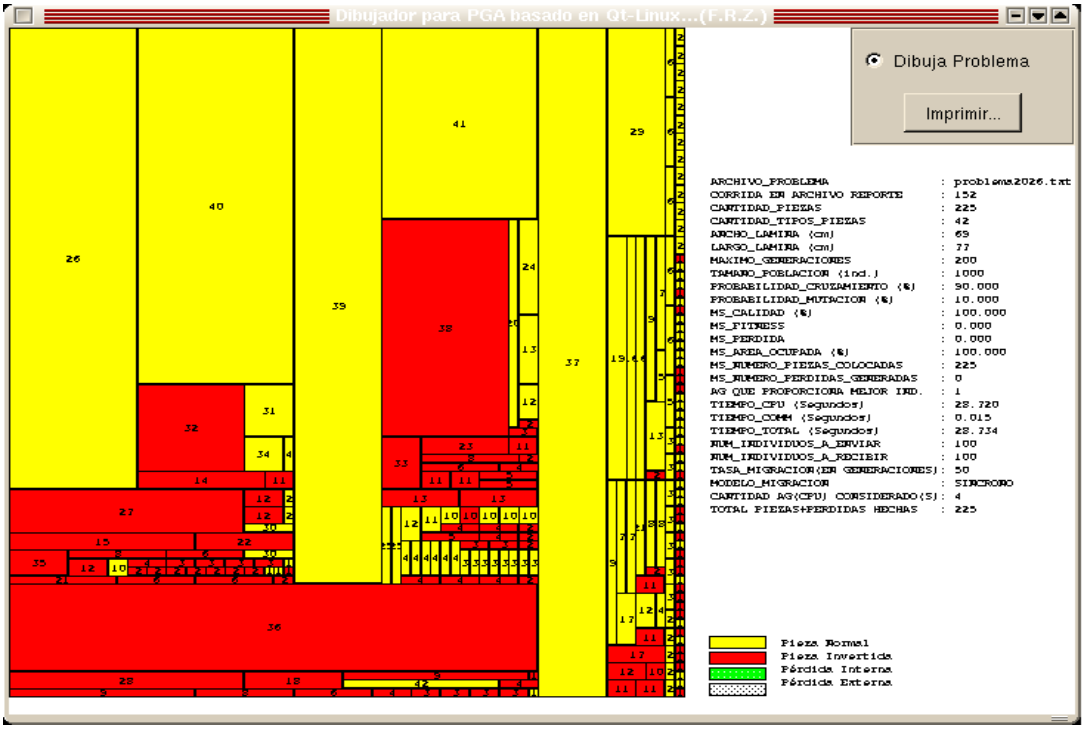
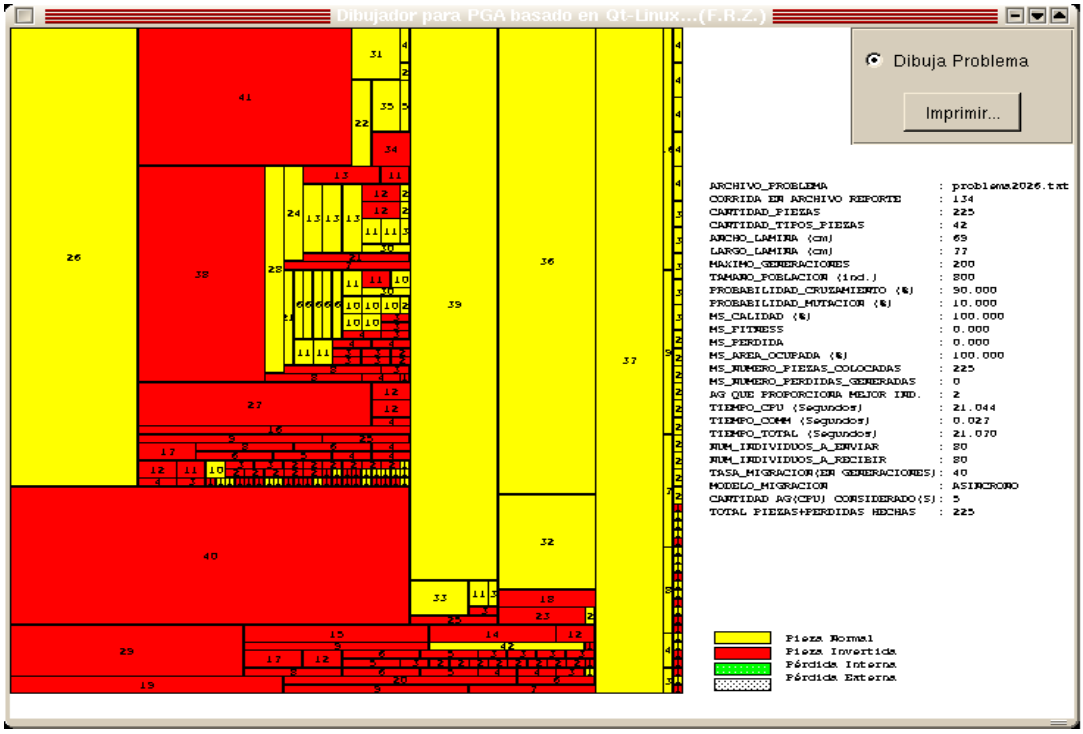


Figura N°D.34: Layout problema2026.txt, considerando cuatro procesadores con Migración (a) Asíncrona y (b) Síncrona.

La Figura N°D.35 muestra el "*layout*" resultante del Problema2026.txt, al ejecutar el AGP considerando cinco procesadores con Migración Asíncrona y Síncrona.

a)



b)

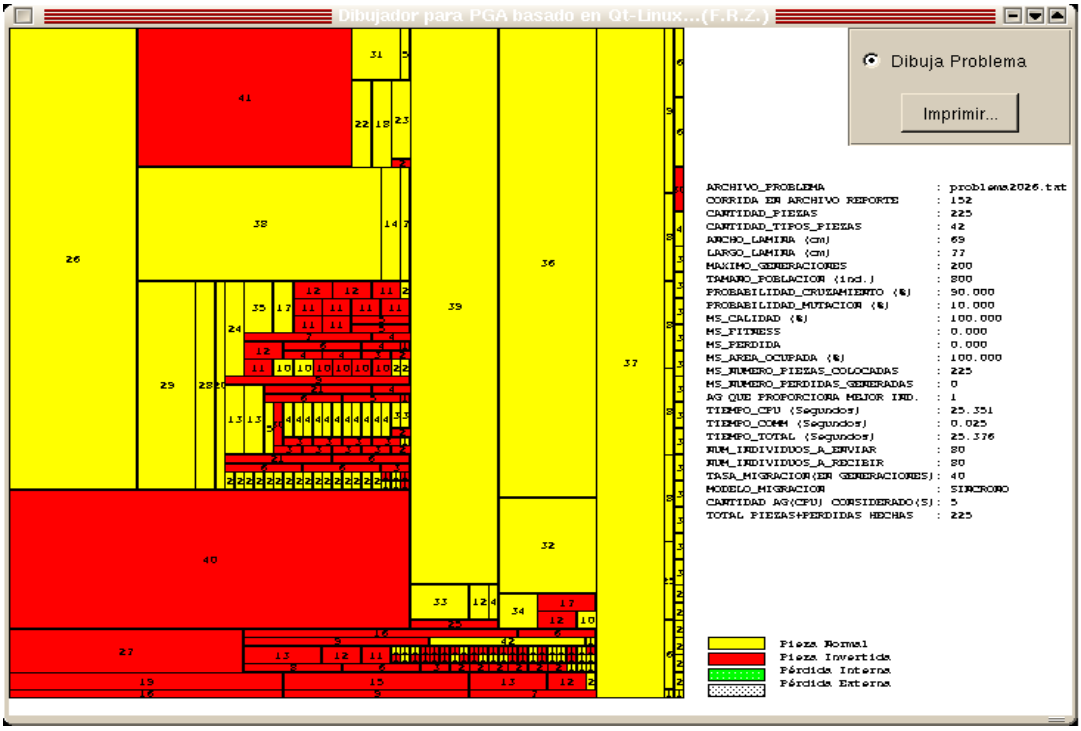


Figura N°D.35: Layout problema2026.txt, considerando cinco procesadores con Migración (a) Asíncrona y (b) Síncrona.

Apéndice E

Funciones principales del Algoritmo Genético Paralelo utilizado para resolver el PCPGBR

A continuación se describen algunas de las funciones principales mencionadas en el seudocódigo del Algoritmo Genético Paralelo del Apartado 4.4, abstrayéndose de los detalles de la implementación.

RUTINAS A EJECUTAR POR COORDINADOR

Lee_Parametros_Geneticos(): El Coordinador lee los parámetros genéticos desde archivo de entrada, cuyo formato se encuentra en el Apéndice A. Los parámetros genéticos son almacenados en una estructura de datos denominada "PARAMETROS_GENETICOS".

Envia_Mensaje("Parametros_Geneticos", PARAMETROS_GENETICOS, Todos_Los_AG): El Coordinador envía un mensaje a "Todos Los AG", etiquetado como "Parametros_Geneticos", transfiriendo a cada AG la estructura de datos PARAMETROS_GENETICOS.

Lee_Datos_Del_Problema_A_Resolver(): El Coordinador lee los datos del problema a resolver desde archivo de entrada, cuyo formato se encuentra en el Apéndice A. Las piezas del problema son almacenadas en una estructura de datos denominada "DATOS_PROBLEMA".

Envia_Mensaje("Datos_Del_Problema_A_Resolver", DATOS_PROBLEMA,

Todos_Los_AG): El Coordinador envía un mensaje a “Todos Los AG”, etiquetado como “Datos_Del_Problema_A_Resolver”, transfiriendo a cada AG la estructura de datos DATOS_PROBLEMA.

Recibe_Mensaje(“Listo_Para_Comenzar”, MENSAJE, Todos_Los_AG): El Coordinador queda esperando, hasta que reciba de parte de todos los AGs un mensaje etiquetado como “Listo_Para_Comenzar”.

Envia_Mensaje(“Comience”, Todos_Los_AG): El Coordinador envía un mensaje a “Todos Los AG”, etiquetado como “Comience”, dando inicio al “Proceso Evolutivo” en cada AG.

Espera_Que_Al_Menos_Un_AG_Alcance_Numero_De_Generaciones_Para_Migrar(): En caso de Migración Asíncrona, el Coordinador debe esperar hasta que al menos un AG alcance el número de generaciones necesarias para que se realice el “Proceso de Migración”.

Espera_Que_Todos_Los_AG_Alcancen_Numero_De_Generaciones_Para_Migrar(): En caso de Migración Síncrona, el Coordinador debe esperar hasta que todos los AG’s alcancen el número de generaciones necesarias para que se realice el “Proceso de Migración”.

Envia_Mensaje(“ENVIE_INDIVIDUOS”, Todos_Los_AG): El Coordinador envía un mensaje a “Todos Los AG”, etiquetado como “ENVIE_INDIVIDUOS”, dando así comienzo al “Proceso de Migración” en cada AG.

Determina_Si_Algun_AG_Ha_Enviado_Individuos(): Esta rutina devolverá el RANK (mayor a cero) del AG que envíe Individuos, de modo que el

Coordinador pueda recibirlos y almacenarlos donde corresponda. En caso de no haber individuos que recibir, la rutina devuelve el valor cero.

Recibe_Mensaje("Individuos_A_Enviar", INDIVIDUOS_AG, AG_QUE_ENVIA_INDIVIDUOS): El Coordinador recibe mensaje desde "AG_QUE_ENVIA_INDIVIDUOS", etiquetado como "Individuos_A_Enviar". Mediante este mensaje se recibe un grupo de individuos que son almacenados en una estructura de datos denominada "INDIVIDUOS_AG". El tamaño del mensaje y de la estructura depende del problema que se está resolviendo. Mientras mayor es la cantidad de piezas del problema, mayor es el tamaño del mensaje y, por lo tanto, el tamaño de la estructura que recibe la información.

Almacena_Individuos_En_Coordinador(INDIVIDUOS_AG, AG_QUE_ENVIA_INDIVIDUOS): Esta rutina guarda los "INDIVIDUOS_AG" recibidos por el Coordinador, en una estructura de datos denominada "COORD_POBLACION". La ubicación de los individuos en la estructura de datos depende del "AG_QUE_ENVIA_INDIVIDUOS". Lo anterior se hace para conocer claramente qué individuos corresponden a cada AG.

Obtiene_Estadisticas_De_La_Poblacion_Del_Coordinador(COORD_POBLACION): Una vez que el Coordinador recibe todos los Individuos de todos los AGs, genera estadísticas de los individuos almacenados, de modo

de conocer el comportamiento del “Proceso Evolutivo”.

Determina_Mejor_Individuo(COORD_POBLACION): De todos los individuos recibidos, el Coordinador selecciona aquel con menor pérdida y lo almacena en una estructura de datos denominada “MEJOR_INDIVIDUO”.

Selecciona_Individuos_A_Enviar_A_Cada_AG(COORD_POBLACION): El Coordinador, de acuerdo con alguna política de selección, obtiene los individuos que serán enviados a cada AG.

Envia_Mensaje(“Individuos_A_Recibir”, INDIVIDUOS. Todos_Los_AG): El Coordinador envía un mensaje a “Todos Los AG”, etiquetado como “Individuos_A_Recibir”, el cual transfiere el grupo de individuos seleccionados desde el Coordinador, a cada uno de los AGs.

Actualiza_Condiciones_De_Termino_Coord(“Individuos_A_Recibir”, INDIVIDUOS. Todos_Los_AG): Esta rutina actualiza la variable “CONDICIONES_DE_TERMINO_COORD” de acuerdo a ciertas condiciones.

Envia_Mensaje(“Finalice_Ejecucion”, Todos_Los_AG): El Coordinador envía un mensaje a “Todos Los AG”, etiquetado como “Finalice_Ejecucion”, permitiendo a cada AG finalizar la ejecución del programa.

Genera_Archivo_De_Resultados(MEJOR_INDIVIDUOS): Con el MEJOR_INDIVIDUO, el Coordinador crea archivo de resultados para que el usuario pueda ver el layout encontrado.

RUTINAS A EJECUTAR POR CADA AG

Inicializa_Semilla_Aleatoria(): Debido a que en los algoritmos genéticos es necesario utilizar un generador de números aleatorios, se requiere contar con una semilla aleatoria que, además, debe ser distinta para cada procesador. Esta rutina permite generar dicha semilla.

Recibe_Mensaje("Parametros_Geneticos", PARAMETROS_GENETICOS, Coordinador): Recibe desde el Coordinador, un mensaje etiquetado como "Parametros_Geneticos", el que se almacena en una estructura de datos llamada "PARAMETROS_GENETICOS". Esta estructura almacena parámetros genéticos tales como: probabilidad de cruzamiento, probabilidad de mutación, tamaño de población, número de generaciones, etc.

Recibe_Mensaje("Datos_Del_Problema_A_Resolver", DATOS_PROBLEMA, Coordinador): Recibe mensaje etiquetado como "Datos_Del_Problema_A_Resolver" desde el Coordinador. Este mensaje corresponde a las dimensiones de la lámina, cantidad de piezas distintas y dimensiones de cada pieza del problema. Se almacena en una estructura de datos denominada como "DATOS_PROBLEMA".

INICIALIZA_POBLACION_INICIAL(): Cada AG debe obtener su propia "Población Inicial", la cual se almacena en OLD_POBLACION.

Envia_Mensaje("Listo_Para_Comenzar", Coordinador): Una vez que el AG está listo para comenzar su "Proceso Evolutivo", envía al Coordinador

un mensaje etiquetado como “Listo_Para_Comenzar” y queda esperando hasta que el Coordinador envíe mensaje dando la orden para comenzar la evolución.

Recibe_Mensaje(“Comience”, MENSAJE, Coordinador): El AG queda esperando hasta que reciba de parte del Coordinador un mensaje etiquetado como “Comience”. Una vez recibido el mensaje, da comienzo al “Proceso Evolutivo”.

Selecciona_Padre(OLD_POBLACION): De la sub-población OLD_POBLACION, el AG obtiene el índice de un individuo que realizará las funciones de “Padre” de acuerdo al método de selección de la sección 4.2.2.1.

Cruza_Padres(Padre1, Padre2, Hijo1, Hijo2): El AG realiza el cruzamiento de padres, tanto para el cromosoma de piezas como para el cromosoma de rotación, obteniendo dos nuevos hijos, tal como se muestra en la sección 4.2.2.2.

Mutacion(Hijo): El AG realiza al azar la mutación de un hijo, tal como se muestra en la sección 4.2.2.3.

Evalua_Individuo(Hijo): El AG evalúa cada individuo “Hijo” que conforma la nueva sub-población, tal como se muestra en la sección 4.2.2.4.

Examina_Si_Debe_Migrar_Individuos(): El AG examina si debe migrar o no individuos. Los individuos serán migrados si el AG recibe mensaje “ENVIE_INDIVIDUOS” desde el Coordinador. En la sección 4.4 se muestran las condiciones a cumplir para que un AG realice el “Proceso de Migración”. Esta rutina devuelve “Verdadero” en caso de que el AG deba comenzar el “Proceso de Migración”, de lo contrario devuelve “Falso”.

Selecciona_Individuos_A_Migrar(): De los nuevos individuos obtenidos en la generación actual, el AG selecciona aquellos que cumplen con alguna política de selección establecida. Los individuos seleccionados

se almacenan en una estructura de datos denominada “INDIVIDUOS_A_MIGRAR”.

Envia_Mensaje(“Individuos_A_Enviar”, INDIVIDUOS_A_MIGRAR, Coordinador): El AG envía al Coordinador, mediante un mensaje etiquetado como “Individuos_A_Enviar”, los individuos almacenados en la estructura de datos INDIVIDUOS_A_MIGRAR. El tamaño del mensaje y el de la estructura dependen del problema que se está resolviendo. Mientras mayor es la cantidad de piezas del problema, mayor es el tamaño del mensaje y de la estructura que mantiene la información.

Recibe_Mensaje(“Individuos_A_Recibir”, INDIVIDUOS_A_REEMPLAZAR, Coordinador): El AG, mediante un mensaje etiquetado como “Individuos_A_Recibir”, recibe desde Coordinador aquellos individuos que deben ser insertados en la sub-población mediante alguna política de reemplazo. Los individuos son almacenados en una estructura de datos denominada INDIVIDUOS_A_REEMPLAZAR.

Inserta_Individuos_En_Sub_Poblacion(INDIVIDUOS_A_REEMPLAZAR): El AG, mediante alguna política de reemplazo, inserta los individuos en la Sub-Población.

Actualiza_Condiciones_De_Termino_AG(): Se actualizan las condiciones de término, de modo que el AG pueda finalizar el “Proceso Evolutivo” una vez que se cumpla alguna de ellas. En la sección 4.4 se examinan las condiciones de término.

Envia_Mensaje(“AG_Ha_Finalizado_Evolucion”, Coordinador): Una vez que el AG ha finalizado su “Proceso Evolutivo”, envía al Coordinador un

mensaje etiquetado como “AG_Ha_Finalizado_Evolución” y espera respuesta del Coordinador.

Recibe_Mensaje(“Finalice_Ejecucion”, MENSAJE, Coordinador): El AG, mediante un mensaje etiquetado como “Finalice_Ejecucion”, recibe desde Coordinador instrucción para finalizar la ejecución del programa.

RUTINAS COMUNES PARA LOS AGS COMO PARA EL COORDINADOR

Inicializa_MPI(): Invoca rutinas de inicialización MPI y permite obtener el rango (RANK) asociado a cada procesador.

Inicializa_Variables_Globales(DATOS_PROBLEMA, RANK): Define los largos de los cromosomas de piezas y de rotación, en términos de bytes de la máquina utilizada y la cantidad de piezas del problema. Inicializa generador de números aleatorios. Inicializa contadores y variables globales. Además, para el caso del Coordinador (RANK = 0), se inicializa archivo de resultados.

Dimensiona_Estructuras_Dinámicas(DATOS_PROBLEMA, RANK): Para el caso de cada AG (RANK ≠ 0), se reserva memoria para las sub-poblaciones de individuos OLD_POBLACION y NEW_POBLACION, así como para el grupo de individuos que se enviarán, y para aquellos que se recibirán desde el Coordinador. Para el caso del Coordinador (RANK = 0), se reserva memoria para almacenar el grupo de

individuos que se recibirá desde cada AG. Las ecuaciones (4.5) a (4.8) definen el tamaño de la sub-población de cada AG, la cantidad de individuos a enviar y recibir por cada AG, y el tamaño de la población del Coordinador. Además, el Coordinador debe reservar memoria para almacenar al “Mejor Individuo” de todo el sistema.

Libera_Estructuras_Dinámicas(RANK): Dependiendo de si RANK corresponde al Coordinador o a un AG, se libera la memoria asignada a las estructuras dinámicas utilizadas en cada caso.

Finaliza_MPI(): Invoca rutina para dar término a MPI.