RESOLUCIÓN DE PROBLEMAS DE CORTE DE PIEZAS RECTANGULARES MEDIANTE ALGORITMOS GENÉTICOS.

De la Fuente García, D.; Gómez Gómez, A.
Universidad de Oviedo.

RESUMEN:

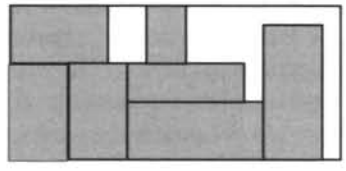
En este trabajo se presenta una forma de resolver el llamado, en la literatura inglesa, problema de packing por medio de una técnica heurística llamada Algoritmos Genéticos. En el trabajo, se compara los resultados obtenidos con esta nueva técnica con los obtenidos con técnicas basadas en programación entera. Así mismo, se comparan diferentes modelos de Algoritmos Genéticos en orden a obtener la tipología más idónea para estos problemas, para este fin se proponen diferentes tipos de cruce, mutaciones y funciones de aptitud. Como última aportación del trabajo, se efectúa una aproximación a los Algoritmos Genéticos Paralelos, con el fin de averiguar si aportan algún tipo de ventajas añadidas.

INTRODUCCIÓN.

En este trabajo se va a abordar un problema industrial real, consistente en la colocación de unas figuras rectangulares dentro de una superficie base con el fin de aprovechar al máximo el material de dicha superficie. Problemas como estos se originan en muchas industrias como las textiles, las de cuero, las navieras y la industria del metal donde la obtención de piezas a partir de una placa base es una actividad recurrente. Desde los años 60, numerosos investigadores han estudiado este problema con el fin de desarrollar algoritmos que permitan resolverlo.

La materia prima usada en la industria suele estar disponible en ciertos tamaños estándares; normalmente es necesario cortar estas hojas en piezas de unas dimensiones fijas antes de poder ser utilizadas en el resto del proceso productivo. El objetivo obvio en esta fase, es poder utilizar al máximo la materia prima. Entre las primeras investigaciones realizadas en este terreno, cabe destacar las realizadas por Gilmore and Gomory (1961-1963) para la resolución de problemas en una dimensión, y la ampliación que estos mismos autores realizaron en 1965 para problemas en dos dimensiones. Desde entonces, el análisis de estos problemas se ha extendido rápidamente; sin embargo, no existe una solución global para todos ellos, debido a la gran complejidad que presentan. Ante esto se ha decidido dividir el problema en varios subproblemas (Dyckhoff,90), e intentar resolver cada uno de ellos por separado. Precisamente, en este trabajo nos vamos a concentrar en uno de estos subproblemas, el problema que se pretende resolver es la colocación de piezas rectangulares en una superficie de igual forma, a la cual llamaremos chapa. En la figura 1, se puede ver un ejemplo de un problema de este tipo.

FIGURA 1: EJEMPLO DE UBICACIÓN DE PIEZAS.



El objetivo perseguido es colocar en la chapa una serie de grupos de piezas rectangulares. Las condiciones del problema exigen que en la solución final deba haber por lo menos 'mi' piezas de un tipo y como máximo 'ma'. Los valores mínimo y máximo dependen de los grupos, cada uno tendrá unos límites diferentes.

A lo largo de los últimos 30 años varios autores se han aproximado a estos problemas, entre los trabajos realizado en este campo, conviene destacar los realizados por Beasley (Beasley,1985). El autor presentó un algoritmo exacto que resolvía los problemas en dos dimensiones de tipo no-guillotina, siendo la primera vez que se presentaba un algoritmo exacto para resolver este problema. El método permite que problemas de tamaños moderados se pueden resolver en tiempos razonables.

ALGORITMOS GENÉTICOS.

Los algoritmos genéticos (abreviadamente AGs), surgieron a finales de los años 60', de la mano de Holland (1975) para dar solución a numerosos problemas, planteados en ciertos sectores de las industrias, y que resultaban de difícil resolución por los métodos que se conocían hasta entonces.

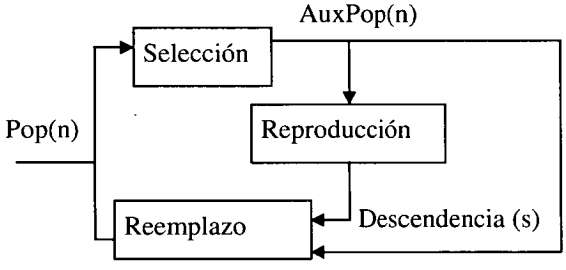
Entre las aplicaciones realizadas con AGs, un elevado número está orientado a la resolución de problemas de optimización, logrando alcanzar una solución, que no siendo la óptimo, si resulta ser una buena aproximación. Los AGs trabajan siguiendo un bucle, llamado generación. Parten de una población inicial (normalmente generada de forma aleatoria) de soluciones, y se va ejecutando dicho bucle sucesivamente. El bucle consta de tres etapas fundamentales: *Selección*, *Reproducción* y *Reemplazo* (Goldberg,89) (Michalewicz,94).

La etapa de *Selección*, consiste en realizar un muestreo de la población de partida, de manera que se obtenga una nueva población, la población auxiliar, con el mismo número de individuos que la población de partida. Esta etapa busca mejorar la población, es decir, favorecer a los mejores individuos. Hay muchas formas de realizar este muestreo, pero lo más habitual es realizar un muestreo por sorteo. Su principio de funcionamiento se base en asignar a cada individuo una probabilidad de selección que dependa de la calidad del individuo, la población auxiliar se genera a partir de estas probabilidades.

En la etapa de *Reproducción*, se aplican los llamados operadores genéticos, siendo los más habituales el cruce y la mutación. En líneas generales, el operador de cruce, actúa tomando dos progenitores e intercambiando parte de sus cadenas, las cruza para generar dos nuevos individuos, los descendientes. Por su parte el operador mutación, se aplica a un progenitor, alterando su cadena de algún modo, por ejemplo cambiando alguno de sus genes.

Al finalizar la reproducción se tienen dos poblaciones independientes, la de los progenitores y la de los descendientes, el último paso se basa en el llamado *reemplazo* consistente en la formación de una nueva población como consecuencia de la mezcla de las dos iniciales. En la figura 2 se puede ver una representación del ciclo completo de una generación.

FIGURA 2: BUCLE BÁSICO DE UN AG



ALGORITMOS GENÉTICOS PARALELOS.

Probablemente el primer intento de introducir algoritmos genéticos en arquitecturas de ordenadores en paralelo fue realizado en 1981 por John Grefenstette. A partir de esa fecha, se han realizado numerosas implementaciones de algoritmos genéticos paralelos (PGA), entre estas se pueden citar:

- **Global parallelization**: El modelo está compuesto por una única población. Su funcionamiento se basa en que el cálculo del fitness requiere gran esfuerzo computacional, para disminuirlo se trabaja en cada ordenador con una parte de la población total. En algunos casos los operadores genéticos también se realizan en paralelo.
- **Fine-Grain GA**: Se asigna un individuo a cada procesador, éste es procesado en paralelo con los demás. Cada individuo es parte de un sistema de múltiples subpoblaciones, cuyas relaciones están determinadas por la topología de la red sobre la que se implemente el PGA.
- **Coarse-Grain GA**: Es similar al anterior, con la diferencia de que cada procesador trabaja con una subpoblación en lugar de con un único individuo.

La principal característica que diferencia a los PGA es el método de migración, es decir, las condiciones que se deben cumplir para que se produzcan intercambios entre las subpoblaciones. Existen varias posibilidades.

- **Sistema de islas**: No existe migración entre las subpoblaciones. Cada una de ellas evoluciona de forma independiente, está demostrado que esta solución no aporta grandes ventajas con respecto a los sistemas no paralelos.
- **Sistema sincronizado**: Cada cierto número de generaciones, las subpoblaciones se intercambian un número de individuos. Este número es fijado por una variable que se conoce como migration rate. Así mismo, los intercambios entre las subpoblaciones no se producen continuamente, sino cada cierto número de generaciones, a la variable que controla este fenómeno se le llama *migration interval.*
- **Sistemas no sincronizados**: Debido a que normalmente las máquinas donde se implementan los PGAs tienen diferentes cargas de trabajo, es imposible sincronizar las comunizaciones entre las subpoblaciones por lo que se recurre a realizar la comunicación en diferentes instantes temporales.

PROBLEMA TRATADO.

El problema que se va a analizar en este trabajo consiste en la colocación en una chapa, de dimensiones rectangulares, de unos grupos de piezas rectangulares; cada uno de estos grupos está formado por rectángulos con dimensiones idénticas, y las condiciones del problema piden que se inserte un número variable entre un mínimo (mi) y un máximo (ma) de estos rectángulos en la chapa. Los valores mínimo y máximo dependen de los grupos, cada uno tendrá unos límites diferentes.

Este problema surge como una continuación del presentado en este congreso el año pasado (Gómez et al,98). La diferencia entre ambas comunicaciones radica en que en el anterior trabajo no se contemplaba la posibilidad de tener que introducir más de una vez la misma pieza y de que existiera una limitación en las dimensiones de la chapa.

En los sucesivos subapartados se van a describir las principales características del AG usado en el trabajo. Inicialmente, se comenta la forma de codificar la ubicación de las piezas rectangulares dentro de la chapa, esta codificación es la base del trabajo y está basada en las ideas de Jakobs (1996), en los demás apartados se comentan las características que presentan los cruces y las mutaciones usadas en el trabajo, asimismo se justifica los dos tipos de función de aptitud usadas.

CODIFICACIÓN.

Primero se crea un cromosoma de longitud igual a la suma de todos los máximos. Cada una de las posiciones del cromosoma va a representar un rectángulo; por ejemplo, si se tiene un problema con ocho rectángulos, el individuo viable será:

1 2 3 4 5 6 7 8

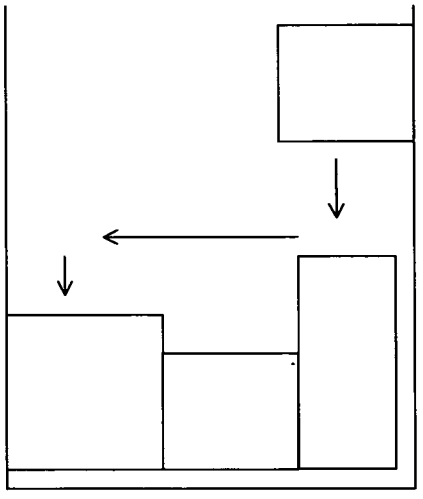
El individuo representa la secuencia usada para empaquetar los rectángulos en el contenedor (chapa). Además, con esta estructura de datos, resulta fácil la elaboración de nuevos individuos por cambios en el orden de la secuencia, el problema se plantea al transformar esta codificación a una estructura física. Para explicar esta transformación se va a usar un ejemplo: Considérese que se tienen 4 rectángulos, para insertar en un contenedor, estos rectángulos estarán numerados del 1 al 4, para distinguirlos. Un individuo (genotipo) que podría ser solución al problema, sería el dado por la siguiente permutación: Π = (1,3,4,2), esto quiere decir que el rectángulo 1 es el primero en introducirse en el contenedor, seguido del 3, del 4, y finalmente del 2.

Una vez conocida la secuencia de entrada de los rectángulos en el contenedor, se va a explicar los pasos del algoritmo propuesto por Jakobs, para saber la ubicación de un rectángulo genérico dentro del contenedor:

1) Se coloca el rectángulo, situado en primer lugar en la permutación, en la esquina inferior izquierda del contenedor.

2) Se van situando el resto de los rectángulos, tomados en el orden en que vienen dados en la permutación, de manera que para situar un rectángulo, se parte de la esquina superior derecha del contenedor, bajándolo tanto como sea posible hasta tocar otro rectángulo y no poder avanzar más. A continuación, se debe llevar tan a la izquierda como sea posible, igualmente hasta alcanzar algún límite que impida el avance (figura 3).

FIGURA 3: SECUENCIA DE EMPAQUETADO.



Este algoritmo base es necesario modificarlo para adaptarlo a los problemas que se analizan en este trabajo, pues aquí las dimensiones del contenedor son fijas, y puede suceder que no se puedan introducir en la chapa todas las piezas. La solución consiste en leer el cromosoma hasta que al introducir un nuevo rectángulo se sobrepase la altura que tiene el rectángulo base. Con esta solución propuesta se puede producir la circunstancia de que no todos los rectángulos se puedan alojar dentro del contenedor.

VARIANTES DEL CRUCE.

Para decidir el tipo de cruce más idóneo para estos problemas, se ha tomado como base el problema del viajante de comercio y se han implementado diferentes tipos de cruce, cuyos resultados en el campo combinatorio son ampliamente reconocidos. En concreto se han implementado cuatro tipos de cruces.

Cruce simple (one-point crossover): selección de un punto de cruce, e intercambio de la información de los dos padres.

Cruce PMX (partial matching crossover), propuesto por Goldberg y Lingle en 1989. Dados dos cromosomas padres, el operador copia una subcadena de uno de los padres directamente a las mismas posiciones en el hijo. Las posiciones restantes se llenan con los valores que aún no han sido utilizados en el mismo orden que se encuentran en uno de los padres. Por ejemplo, si tenemos dos cadenas p1(1,2,4,6,3,7,5,8) y p2(5,4,1,7,2,6,8,3) y si la subcadena seleccionada al azar de p1 para ser insertada en p2 es la (4,6,3) esto establece una relación con la subcadena (1,7,2) que ocupa las mismas posiciones en p2. Entonces la secuencia de operaciones transformarían p2 en (5,4,4,6,3,6,8,3), y luego, se elimina las repeticiones quedando (5,*,4,6,3,*,8,*) y reeemplazando queda (5,1,4,6,3,*,8,*) ya que el 4 había ocupado el lugar del 1 en la subcadena. Continuando obtenemos (5,1,4,6,3,7,8,2).

Cruce OX (order crossover), propuesto por Davis en 1985.

Cruce CX (cycle crossover), propuesto por Oliver en 1987.

TIPOS DE FITNESS.

A la hora de calcular la bondad de una determinada solución, se platearon una serie de problemas, los cuales han motivado que existan diferentes tipos de fitness. Como ya se ha comentado anteriormente, se decidió establecer como base para el cálculo del fitness, la altura que alcanzan dentro del contenedor los rectángulos. El algoritmo descodificador se detiene cuando la altura alcanzada supera la altura de la placa base; en ese instante, se comprueba si todos los rectángulos caben o no dentro de la chapa. A la hora de implementar estas ideas se vio la necesidad de incluir ciertas modificaciones al fitness inicial, estas modificaciones dieron lugar a la existencia de dos fitness, los cuales se comentan a continuación:

Fitness 1: La función encargada de calcular el fitness asigna un valor de uno al fitness si todos los rectángulos que deben obligatoriamente formar parte de la solución final están en ella, sino se suma al valor del fitness un número proporcional al número de rectángulos que falten en la solución final. La fórmula utilizada es:

Fitness1 = número de rectángulos no introducidos * penalización+1.

El objetivo que debe buscar el AG es minimizar este fitness hasta conseguir que su valor sea la unidad. Esta codificación presenta un problema, una vez superado el número mínimo de rectángulos de un

tipo, el AG no se ve motivado para introducir más rectángulos de ese grupo y, en estos problemas, no sólo se busca cubrir los mínimos, sino que se intenta introducir la mayor cantidad de rectángulos posible. Esto motiva que esta primer función de aptitud se deba modificar.

Fitness 2: El fitness es un número que depende del número de rectángulos que hay en la chapa. Se usa la siguiente fórmula a la hora de establecer el fitness:

Fitness = número de rectángulos total - número de rectángulos introducidos en la chapa+ fitness1+1.

MUTACIONES.

Se han implementado dos tipos de mutaciones:

Order-based Mutation: Basada en los trabajos de Davis de 1991, consiste en permutar dentro de un individuo la posición de dos rectángulos.

Mutación variable: Se usa una mutación idéntica a la anterior con la salvedad de que la probabilidad de mutación se va modificando a medida que pasan las generaciones. Se comienza con una probabilidad de mutación muy elevada, para poder rastrear el mayor rango posible, y se va disminuyendo esta probabilidad.

RESULTADOS.

Para poder medir la calidad de las soluciones aportadas por cada uno de los AGs presentados en este trabajo ha sido necesario generar una serie de problemas de test. Estos se han agrupado en cuatro tipos de problemas, su diferencia viene dada por el área de la superficie base que pueden cubrir; en el primer grupo la suma de las áreas de todas las piezas es el 50% del área de la superficie base, en el segundo es el 80%, en el tercero el 110% y en el cuarto el 120%. Como se puede deducir fácilmente, cuando se trata un problema del grupo tres o cuatro, no todas las piezas van a caber en la superficie base. Asimismo se generaron para cada uno de los grupos, problemas con 50, 100 y 200 piezas, con el fin de comprobar la influencia del número de rectángulos en la velocidad de cálculo y en la calidad de las soluciones.

A continuación se describen algunos de los experimentos realizados, así como los resultados más destacados obtenidos. Las investigaciones han seguido dos líneas en primer lugar se han comparado los resultados de los AGs con una técnica plenamente contrastada como es la programación entera (Beasley,85) y, en segundo lugar se han comparado diferentes AGs entre sí, incluyendo los PGA. En concreto, se ensayaron varios tipos de AG, con el objetivo de buscar la mejor combinación de cruce, mutación y fitness para este tipo de problema.

Como se acaba de comentar, en primer lugar, se analizó la posibilidad de comparar los resultados que se obtienen al simular un algoritmo genético con los que se obtienen mediante técnicas de programación entera (Beasley,85). Esta comparación sólo se pudo hacer para simulaciones con menos de 20 piezas, cuando el número de piezas a colocar superaba esta cifra, la solución basada en programación entera se volvía muy lenta (varias horas), sin embargo el tiempo necesario para la simulación con AG seguía siendo de unos pocos segundos.

Con menos de 20 piezas, los resultados ofrecidos por ambos métodos son muy similares, obviamente el AG, en algunas simulaciones, aprovecha menos el materia, pero este menor aprovechamiento no pasa de un 5%. Estos resultados nos permiten concluir que la solución que planteamos con AG se puede aplicar en estos problemas, siendo conscientes que su uso implica un menor aprovechamiento del material.

En la segunda parte del trabajo, nos planteamos analizar que modelo de AG se ajustaba mejor a estos problemas, en la bibliografía relacionada con los AGs están documentados numerosas tipologías de algoritmos, y no es fácil determinar de antemano cuál de ellas se ajusta mejor a nuestro problema. Los resultados obtenidos con los diferentes experimentos realizados se comentan a continuación, conviene en primer lugar, comentar las características de los ensayos realizados, pues dado el carácter aleatorio de los AGs, se consideró oportuno realizar 30 ensayos para cada experimento, y analizar la solución media.

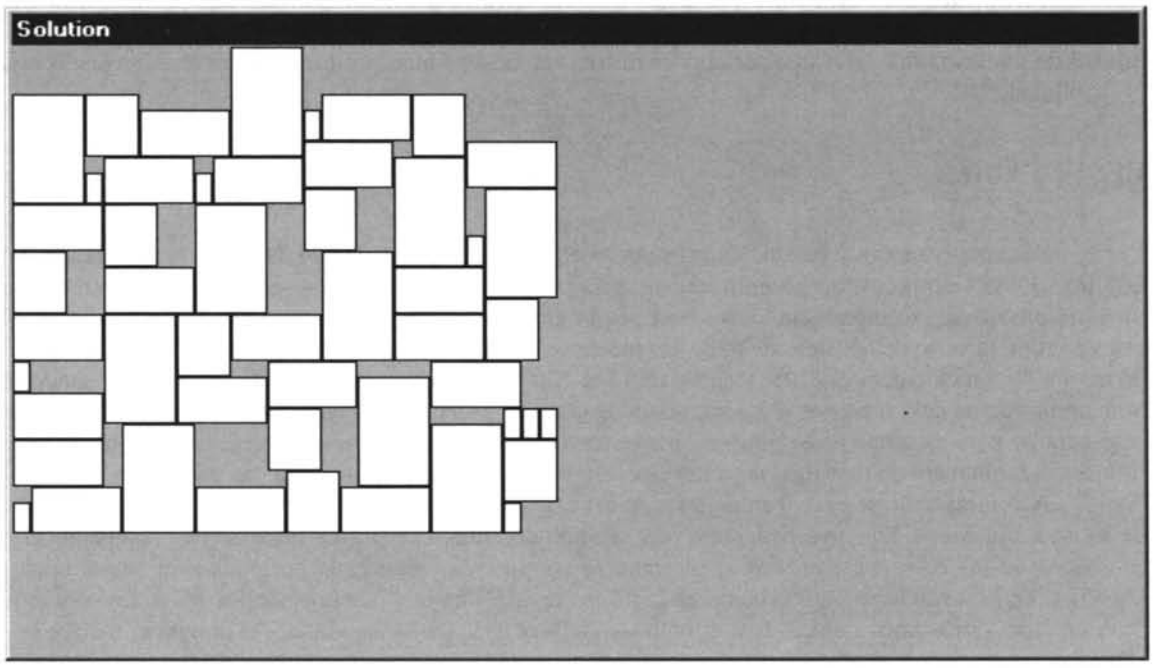
Con los primeros ensayos, se llegó a la conclusión que los dos primeros grupos (50% y 80%), no aportaban ninguna información de interés, pues el AG consigue colocar todas las piezas dentro de la superficie base en todos los ensayos. Luego, se decidió centrar la investigación en los otros dos grupos. Con ellos la información obtenida es más interesante, entre las conclusiones obtenidas conviene destacar que, como ya se esperaba, las mejores soluciones se obtienen cuando se usa el fitness 2, pues es el único que intenta introducir más piezas dentro de la superficie base de las estrictamente necesarias.

Los demás parámetros del AG no presentan ninguna sorpresa. El único resultado sorpresivo se obtiene a la hora de decidir cual es el mejor cruce. Pues, de los resultados experimentales se deduce que lo

mejor es usar un cruce simple, la razón de este resultado puede estar en que realmente no se utiliza todo el cromosoma, y los otros cruces operan con todo el cromosoma. Los resultados de la mutación variable son mejores al aumentar el número de piezas a ubicar, pues se aumenta la complejidad del problema y esto posibilita que esta mutación aporte información.

En cuanto a las posibles aportaciones de los PGA, en este trabajo se han realizado diferentes pruebas, y no se ha podido encontrar ninguna mejora destacable con respecto a las soluciones obtenidas con los AGs simples. En la figura 4, se representa una solución generada por el programa diseñado para realizar los experimentos, en concreto en el ejemplo se representa una solución con 50 piezas rectangulares.

FIGURA 4: REPRESENTACIÓN DE UNA SOLUCIÓN.



CONCLUSIONES.

Este trabajo se ha centrado en buscar formas de resolver un problema de packing con la ayuda de técnicas heurísticas, en concreto se han usado algoritmos genéticos. En él se compara los resultados proporcionados por los AGs con los obtenidos con otros métodos basados en programación entera, y se busca además la tipología genética que mejor se adapte a los problemas de packing.

Entre las futuras líneas que podría seguir esta investigación conviene comentar dos, en primer lugar es necesario intentar mejorar los resultados ofrecidos por los algoritmos genéticos paralelos y, en segundo lugar se deben probar otras formas de codificar estos problemas.

BIBLIOGRAFÍA.

BEASLEY, J. E. (1985) AN EXACT TWO-DIMENSIONAL NON-GUILLOTINE CUTTING TREE SEARCH PROCEDURE. OPNS RES. 33, 49-64.

BELDING T (1995). THE DISTRIBUTED GENETIC ALGORITHM. PRESENTED AT THE SIXTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, SAN MATEO, CA: MORGAN KAUFMANN PUBLISHERS

DAVIS L (1991). HANDBOOK OF GENETIC ALGORITHMS. *VAN NOSTRAND REINHOLD*, NEW YORK,

DYCKHOFF, H. (1990) A TYPOLOGY OF CUTTING AND PACKING PROBLEMS. EUR. J. OPL RES. 44,145-159.

GILMORE, P.C., GOMORY, R.E. 1961. A LINEAR PROGRAMMING APPROACH TO THE CUTTING-STOCK PROBLEM. OPERATIONS RESEARCH, VOL 9, PP 849-859

GILMORE, P.C., GOMORY, R.E. 1963. A LINEAR PROGRAMMING APPROACH TO THE CUTTING-STOCK PROBLEM, PART II. OPERATIONS RESEARCH, VOL 11, PP 863-888

GILMORE, P.C., GOMORY, R.E. 1965. MULTISTAGE CUTTING STOCK PROBLEMS OF TWO AND MORE DIMENSIONS. OPERATIONS RESEARCH, VOL 13, PP 94-120

GOLDBERG, D. E. (1989). GENETIC ALGORITHMS IN SEARCH, OPTIMIZATION AND MACHINE LEARNING, ADDISON-WESLEY.

GÓMEZ GÓMEZ, A.; PARREÑO FERNÁNDEZ, J. Y FERNÁNDEZ QUESADA, I. (1998). "TÍTULO: RESOLUCIÓN DE PROBLEMAS DE PACKING EN UNA EMPRESA DE CALCAMONIAS MEDIANTE ALGORITMOS GENÉTICOS." EN EL XII CONGRESO NACIONAL Y VIII CONGRESO HISPANO-FRANCÉS DE AEDEM. P.:999-1008. BENALMÁDENA (MÁLAGA)

GREFENSTETTE J (1981). *PARALLEL ADAPTIVE ALGORITHMS FOR FUNCTION OPTIMIZATION.* TECHNICAL REPORT CS: 81-19, VANDERBILT UNIVERSITY, NASHVILLE, TN.

HOLLAND, J. J. (1975) ADAPTATION IN NATURAL AND ARTIFICIAL SYSTEMS, UNIVERSITY OF MICHIGAN PRESS.

JAKOBS S (1996). ON GENETIC ALGORITHMS FOR THE PACKING OF POLYGONS. *EUR. J. O. R.*. **88**: 165-181.

MICHALEWICZ, Z. (1994). GENETIC ALGORITHMS + DATA STRUCTURES = EVOLUTION PROGRAMS. SPRINGER VERLAG.