V Congreso de Ingeniería de Organización
Valladolid-Burgos, 4-5 Septiembre 2003

Índice

Creación de una herramienta software para el Scheduling y Planificación de operaciones basada en algoritmos genéticos

Gutiérrez Cosío[1] y Guillermo Parodi Hernandorena[2]
[1]Doctora Informática, Universidad Camilo José Cela, Villafranca del Castillo (Madrid),
celiagutierrez@hotmail.com
[2]Ing. de Telecomunicaciones, Universidad Camilo José Cela, Villafranca del Castillo (Madrid),
gparodi@ucjc.edu

RESUMEN

El trabajo desarrollado estudia el problema de la optimización en la asignación de tiempos y recursos para completar un conjunto de tareas mientras se satisface una serie de restricciones del dominio. Los problemas de Scheduling reales suponen manejar cantidades importantes de tipos diferentes de relaciones y restricciones entre variables, sobretodo de recursos y tiempo. Estas relaciones se llaman restricciones y tienen diferentes formas: fechas de entrega de pedidos, capacidad de recursos, precedencia de operaciones, restricciones fisicas (tamaño), etc. En la práctica, existe un conflicto fuerte entre dos objetivos: consecución de un óptimo y satisfacción de todas las restricciones. En esta tesis, siguiendo las directrices de Goldberg,, se ha aplicado un Algoritmo Genético, que es una herramienta de optimización usando la técnicas de la penalización. Se ha elegido este método (técnicas de penalización) frente a otros, por su simplicidad y aplicabilidad a problemas de diferente tipo. Además, se le han añadido otras peculiaridades que lo hacen totalmente distinto de lo hecho hasta el momento.

Palabras clave: *Planificación, Scheduling ,tiempos de comienzo, algoritmo genético.*

1 Introducción.

El presente trabajo presenta el problema de la asignación de recursos y tiempos para la ejecución de una serie tareas de manera que se pueda obtener un plan que sea "óptimo" desde un criterio. El método empleado para resolver el problema se basará en el uso combinado de Algoritmos Genéticos (AGs) artificiales y técnicas reparadoras heurísticas.
En los problemas de resolución de tareas, existen conflictos entre el binomio restricciones-optimización de función, tal y como ocurre en el mundo real (no existe un problema a resolver sin tener recursos limitados para su solución). En este trabajo, este problema se ha abordado desde el punto de vista de la penalización de la función objetivo en el grado en que no se cumplan las restricciones, aunque existen otras maneras de incorporar restricciones a los Algoritmos Genéticos.

Existen estudios previos sobre la aplicación combinada de los AG´s y técnicas heurísticas en tareas de planificación, como en [1] y [2].

En el 2º apartado se habla de los métodos heurísticos de optimización: algoritmos genéticos y técnicas de reparación de búsqueda local. En el 3er. apartado se explica el caso de planificación concreto a resolver. En el 4º apartado se detalla la solución diseñada para abordar el problema. En el 5º apartado se muestran los resultados experimentales obtenidos.

En el 6º se explican las conclusiones. Por último se adjuntan los agradecimientos y unas referencias bibliográficas.

2 Métodos heurísticos de optimización.

Las técnicas tradicionales de optimización (Investigación Operativa,...) resultan ineficientes para problemas donde hay que realizar búsqueda en espacios amplios, esto es en problemas de optimización muy complejos, debido a que computacionalmente no se pueden llevar a cabo. Esta problemática también se extrapola a problemas de Scheduling complejos. De esta manera surgen técnicas de búsqueda heurística, como se postula en [3]. En vez de realizar la búsqueda en el espacio del problema, las técnicas heurísticas modernas se concentran en guiar la búsqueda hacia regiones prometedoras del espacio de búsqueda, como se postula en [4]. Existen varios métodos heurísticos de optimización, de entre las cuales destacan los Algoritmos Genéticos y Búsqueda Local.

2.1 Algoritmos Genéticos.

Los algoritmos genéticos (AGs) han surgido fundamentalmente como consecuencia de los estudios realizados por Holland y su equipo de la Universidad de Michigan durante la década de los 70. Ya más recientemente, durante la década de los 90, ha comenzado su explotación industrial, que primero se ha realizado a nivel de laboratorios o centros de investigación, y ya en este momento está empezando a extenderse a nivel industrial.
Entre muchas de las áreas en las que se pueden aplicar soluciones basadas en los algoritmos genéticos está la planificación o programación de tareas.

2.1.1 Conceptos sobre Algoritmos Genéticos.

Los algoritmos genéticos toman su nombre debido a la similitud de su comportamiento con el de la Naturaleza: al igual que en la Genética natural se trata de llegar a los individuos más perfectos, más selectos. Una función a optimizar será el criterio por el cuál decidiremos la población más perfecta. Así mismo, los mecanismos mediante los cuales se obtienen descendientes más perfectos son los mismos que postula la selección natural de Darwin: reproducción, cruce y mutación de los componentes genéticos de los individuos.
El algoritmo genético comienza explorando una parte del espacio de soluciones al azar y generación tras generación explora otras zonas donde se encuentran las soluciones más óptimas.

2.1.2 Ejecución de un AG.

Si el algoritmo está bien diseñado será convergente, lo cual quiere decir que al cabo de varias generaciones mejorará su función de adaptación ó fitness, hasta que ésta se estabilice.
En la figura (1), se muestra un ejemplo en el que el objetivo es la maximización de la función $F(x)=x^2$, con el dominio [0,1] para la variable x. Los mejores valores (óptimos) corresponden al máximo y se consiguen obviamente para valores F(x)=1, de ahí que la función se va acercando a este valor.

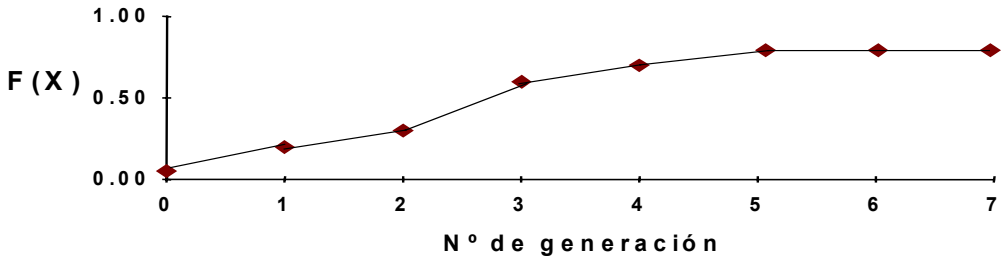

Figura 1 : Ejemplo de ejecución de un Algoritmo Genético

2.1.3 Codificación de las soluciones.

Los individuos de la población, también llamados cromosomas, se codifican en un alfabeto en el que sea fácil utilizar los operadores de selección, cruce y mutación. Los cromosomas se subdividen en genes, cada uno de los cuales tiene un significado concreto para el problema tratado. En nuestro caso un gen será el tiempo de comienzo de una operación y un cromosoma será el conjunto de tiempos de comienzo de las operaciones. La labor que realiza el algoritmo genético es la de extraer los valores para estos genes, que optimizan la función de adaptación. Son por tanto, las variables a manejar.

2.1.4 Incorporación de condiciones.

El AG puro y simple se adapta perfectamente a funciones de optimización sin condiciones; el problema surge cuando aparecen las restricciones. Una manera válida de abordar el problema de las restricciones sería aplicar este sencillo método de prueba y error a los cromosomas: descartar los individuos que no superen las restricciones, con lo cual perecen y no se reproducen; sin embargo, esta solución no es válida para problemas generalizados de restricciones., ya que es muy difícil que en la primera generación existan individuos ó cromosomas válidos.

El método de la penalización es una solución al problema anterior. Se basa en aplicar a la función de adaptación de cada individuo una penalización proporcional a la función violada. De esta manera, un individuo no será tan adaptado como lo que indique su función de adaptación si no supera todas las condiciones.

Las siguientes fórmulas (1) y (2) muestran una función de adaptación sin penalización y con penalización:

Sin penalización

$$\text{Minimizar } g(x), \quad \text{condiciones: } h_i(x) > 0 \tag{1}$$

Con penalización

$$\text{Minimizar } g(x) + r * \Sigma\Phi(h_i(x)) \tag{2}$$

donde r=coeficiente de penalización

Φ=función de penalización

El primer método, (1), que no contiene penalización, asigna un valor de función de adaptación a cada individuo y evalúa todas las condiciones; si no respeta la totalidad de las condiciones, el individuo no pasa a la siguiente generación.

El segundo método, (2), es el de la función de adaptación penalizada. El primer sumando es igual que el método sin penalización, es decir, la función de adaptación pura; el segundo sumando es la penalización y se calcula para cada condición: se le resta o suma a cada individuo una cantidad proporcional a la cantidad que supera el límite impuesto por la condición. El coeficiente de penalización (r) es el que dota de proporcionalidad y en su caso más simple r=1. La función de penalización (Φ se aplica a cada cantidad que no respeta la condición. Muchas veces $\Phi(h_i(x))\ h_i^{\ 2}(x)$.

Un ejemplo claro de función de adaptación penalizada es el aplicado al caso que se va a tratar en el siguiente capítulo: una operación se tiene que ajustar a un rango de tiempo de comienzo para que el conjunto total de operaciones realicen un producto a su debido tiempo. Suponiendo que para la operación A el tiempo de comienzo es t_A y que el rango establecido es de $[t_B\text{-}t_C]$ y $t_A < t_B$, hay que penalizar la función de adaptación. Si se aplican los valores que se han explicado anteriormente para el coeficiente y la función de penalización, esto resulta en una penalización de $(t_B\text{-}t_A)^2$.

El criterio de sumar ó restar la cantidad de penalización se aplica teniendo en cuenta si se trata de optimizar minimizando (se le resta) ó maximizando (se le resta).

2.2 Técnicas de reparación de búsqueda local.

Diversos algoritmos de búsqueda local se han desarrollado compartiendo la idea básica de vecindad. Una solución vecina se deriva de su solución originaria mediante una modificación parcial predefinida, llamada movimiento. Un movimiento produce una solución vecina que se diferencia solo ligeramente de la solución originaria. Se espera que una solución vecina produzca un valor de función objetiva de parecida calidad (algo mejor, ya que se trata de mejorar) que la solución originaria, ya que comparten la mayoría de características. De esta manera, se concentra en la búsqueda dentro de vecindades.

3 Planteamiento de un caso generalizado de Scheduling.

Una planta recibe múltiples pedidos, los cuales se componen de múltiples productos, los cuales se deben fabricar varias veces (puede haber varias ocurrencias de un producto), que deben ser fabricados por varias operaciones. Por último estas operaciones son realizadas por un recurso determinado (máquina ó persona) en un determinado tiempo.
Son precisamente estas las variables a manejar: tiempos de comienzo de las operaciones (función de la Planificación) y recursos donde se van a llevar a cabo las operaciones (función del Scheduling). La función a optimizar es en un principio la minimización del tiempo total de fabricación de los pedidos, por tanto de todas las operaciones necesarias para llevarlo a cabo.

Las condiciones son:

- Las operaciones se deben ajustar a un rango de tiempo de comienzo determinado.
- Suponiendo que los recursos tienen capacidad 1, es decir, solo pueden estar realizando una operación a un tiempo, las operaciones no se pueden solapar en un recurso.

4 Resolución del caso con una combinación de Algoritmo Genético y Heurísticas.

La solución se ha implementado sobre un generador de algoritmos genéticos llamado GAGS, que utiliza programación orientada a objetos.
Después de realizar múltiples pruebas, se ha llegado a la conclusión que la solución más viable es dividir el algoritmo en cuatro partes:

- Cálculo de parámetros
- Algoritmo Genético de asignación de recursos
- Algoritmo Genético de asignación de tiempos
- Heurísticas reparadoras de tiempos

La figura 2 muestra la descomposición del algoritmo genético:

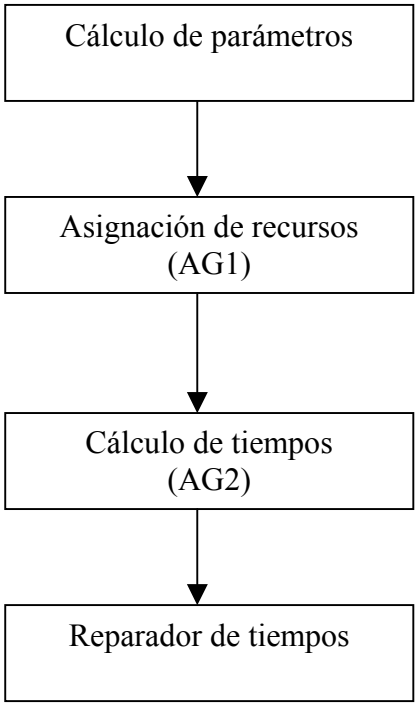

Figura 2 : Descomposición del algoritmo genético

4.1 Cálculo de parámetros.

Los parámetros que se van a calcular son los tiempos mínimo y máximo de comienzo de cada operación. Los datos a tener en cuenta para el cálculo son la duración de cada operación y la fecha de entrega, así como el orden de precedencia de las operaciones.

Posteriormente se identifica cada operación de la planta según estos parámetros, teniendo en cuenta que una operación concreta se lleva a cabo para un pedido, un producto, una unidad de un producto y una operación del conjunto que puede realizarse en la planta:

Identificación de la operación: nº pedido, nº producto, nº unidad, nº operación

Esto da lugar a la aparición de las primeras clases con sus correspondientes atributos:
- Clase operación: nº pedido, nº producto, nº unidad, nº operación, tipo de recurso en el que se puede realizar, recurso asignado, tiempo mínimo de comienzo, tiempo máximo de comienzo, tiempo de comienzo.
- Clase recurso: nº recurso, tipo recurso.

De todos estos atributos, el recurso asignado y el tiempo de comienzo de la operación son las variables a calcular en las fases siguientes.

4.2 Asignación de recursos.

Esta tarea se realiza en un primer algoritmo genético, que calcula la asignación óptima de los recursos de la planta a cada operación. En la planta existen varios recursos que pertenecen a un tipo determinado, por ejemplo, Los recursos 1, 2 y 3 pertenecen al tipo A, ya que todos ellos son taladradoras. Se ha diseñado una función de adaptación que penaliza la asignación real de recursos respecto a la ideal. Se considera una asignación ideal aquella en la que los recursos del mismo tipo tienen asignadas el mismo número de operaciones. De esta manera, las operaciones están uniformemente asignadas en los recursos. La penalización consiste en la desviación de la asignación real respecto a la ideal por tipo de recurso, como se expresa en (3):

$$\text{fitness}=r * \sum^{\text{tipo}}_{i=0}[(\text{capacidad}_i / \text{capacidad total} * \text{n.ops.asignadas}) - (\text{capacidad}_i / \text{capacidad total} * \text{n. ops.}) \quad (3)$$

Cada gen contendrá encriptados los parámetros que definen:
- número de pedido
- número de producto
- número de operación
- número de ocurrencia
- número de recurso asignado

La variable obtenida en la salida de este algoritmo es el número de recurso asignado.

4.3 Asignación de tiempos.

El cromosoma es simplemente los tiempos de comienzo de cada operación. Los tiempos de comienzo deben estar ajustados a los rangos de comienzo y deben respetar la precedencia entre operaciones del mismo producto, y por supuesto, el no solapamiento entre operaciones de las mismo producto ó asignadas al mismo recurso. El ajuste a los rangos se consigue por la propia potencia de GAGS. El valor fitness es el inverso de la variable “valor”.
En el siguiente fragmento de código (Figura 3), aparece codificada la función fitness:

```
while (j<cant && num_prod[j]==num_prod[i] &&
num_pedido[j]==num_pedido[i] && num_ocur[j]==num_ocur[i]) {

      // Comparamos con cada operacion que pertenezca a la
      // misma ocurrencia de producto
      if (num_prod[j]==num_prod[i] && num_pedido[j]==num_pedido[i] &&
num_ocur[j]==num_ocur[i])
            // No pasamos a la siguiente generacion una operacion que
            // tenga lugar antes que otra  en la cadena de produccion
            if (myGen[j]<myGen[i])
                        valor+=1000000;
            j++;
           }
           i++;
      }
      i=0;
      while (i<cant){
           j=i+1;
           while (j<cant){
                 if (num_rec[j]==num_rec[i])
                 // Penalizamos las operaciones que se solapan con las
                 // restantes en toda la cadena de produccion
                       if (myGen[i]+sample3.leerDato(i) > myGen[j])
                             valor+=myGen[i]+sample3.leerDato(i)-myGen[j];
                 j++;
            }
            i++;
```

Figura 3: Código que representa la función de adaptación del algoritmo genético de tiempos

Como resultado de este algoritmo surgen cuatro clases más a tener en cuenta con sus respectivos atributos y métodos:

- Clase restricción: contiene tipo de restricción, método de evaluación de la restricción.
- Clase solapamiento: hereda de la clase restricción.
- Clase rango: hereda de la clase restricción.
- Clase precedencia: hereda de la clase restricción.

Se creará un objeto de clase solapamiento, rango y precedencia para cada operación de la planta.

4.4 Heurísticas reparadoras.

Esta fase consta de una fase evaluadora, en la cual se evalúa para cada operación, si cumple cada una de las restricciones. Si una operación no cumple una restricción, se crea una instancia de la clase solapamiento_violado, precedencia_violada ó rango_violado, según sea el tipo de restricción afectada.

Por tanto, aparecen en este apartado tres clases nuevas:

- Clase solapamiento_violado: hereda de solapamiento y contiene el método repara_solapamiento.
- Clase rango_violado: hereda de rango y contiene el método repara_rango.

- Clase precedencia_violada: hereda de precedencia y contiene el método repara_solapamiento.

La siguiente fase es la reparadora, en la que se reparan las restricciones de las operaciones afectadas, aplicando el método que corresponda. Mención aparte merece el reparador de solapamiento, por su grado de complejidad. Se han diseñado dos operadores, reparador de solapamiento puro y reparador de solapamiento híbrido y se han realizado pruebas, para comprobar cuál era el mejor:

4.4.1 Reparador puro de solapamiento.

El reparador se aplica siempre a la operación más crítica entre dos que estén solapadas. En este primer caso se aplicará un solo operador, que es la asignación de huecos.

4.4.1.1 Asignación de huecos.

Se busca un hueco en la línea de producción del mismo recurso, en el que pueda encajar la operación a tratar. Los límites de tiempo deben estar dentro del rango de tiempo de comienzo, con lo que se garantiza la no-violación de la restricción de rango. Si no existen huecos que la cumplan, se pasa a mover las operaciones que se puedan, para hacer un hueco en que se pueda encajar la operación. Si no se encuentra, se procede a la asignación de huecos en un recurso del mismo tipo al asignado por el algoritmo genético.

4.4.2 Reparador híbrido de solapamiento.

La figura 4 muestra el funcionamiento del reparador híbrido:

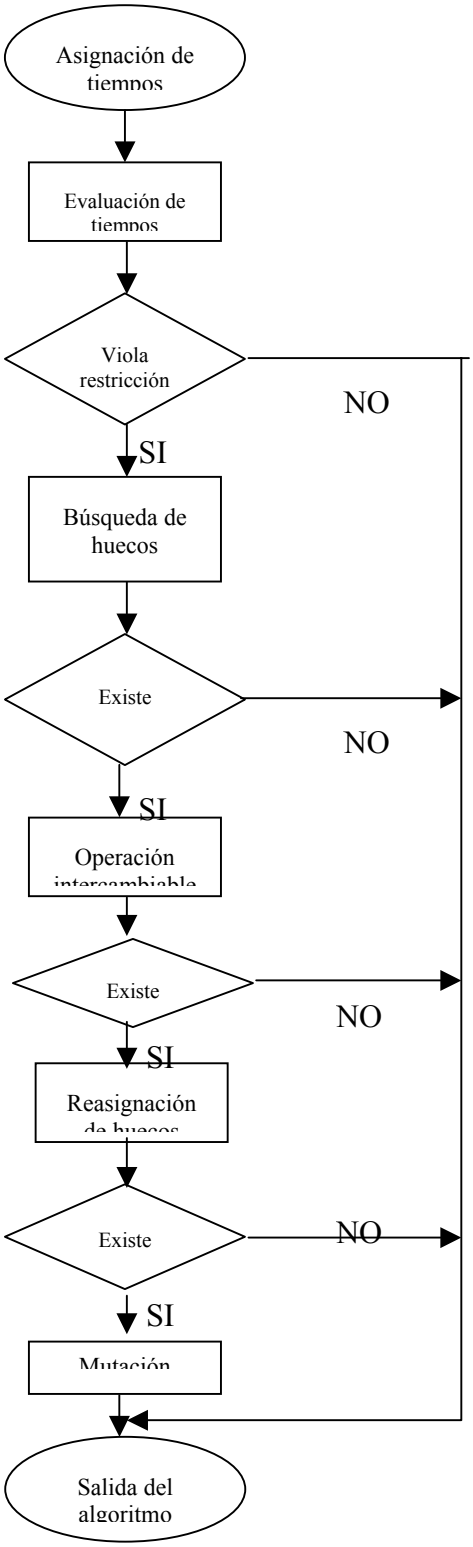

Figura 4 : Heurística reparadora del solapamiento (híbrido)

El reparador híbrido combina además del operador de asignación de huecos, otros dos operadores, que son los siguientes:

4.4.2.1 Intercambio de operaciones.

Se basa en intercambiar los tiempos de comienzo de la operación inmediatamente anterior en la línea de producción del recurso asignado.

4.4.2.2 "Mutación" del recurso.

Se basa en el cambio aleatorio del recurso asignado a una operación a otro recurso del mismo tipo, si es que existe.

5 Resultados obtenidos.

Los experimentos realizados van a ser todos para un pedido, luego se extrapolan los resultados obtenidos a cuatro pedidos. Con estos primeros ensayos se comprueban qué tipo de operadores se deben aplicar. Los de precedencia y rango se aplican siempre, y se varían la forma de aplicar los de solapamiento. El de asignación de huecos siempre debe aparecer, ya que es el más lógico.
- Primero, se aplica el operador de asignación de huecos, luego el de intercambio de operaciones y posteriormente el de mutación de recurso.
- Segundo, se aplica el operador de solapamiento híbrido, que combina la potencia de los tres operadores.

5.1 Operador puro para un pedido.

Aplicando un operador de asignación se consigue que el algoritmo se estabilice consiguiendo un porcentaje de acierto del 100%. Sin embargo, con los otros operadores no se logra que se estabilice, y los porcentajes son peores. Los resultados aparecen en la tabla 1:

	Convergente	% Efectividad de restricciones
Asignación de huecos	Si	100%
Intercambio de operaciones	No	-
Mutación del recurso	No	-

Tabla 1: Resultados obtenidos para un pedido (operador puro).

5.2 Operador puro para dos, tres y cuatro pedidos.

Los porcentajes empeoran ya que hay mayores interferencias entre las operaciones y es más fácil que no se cumplan las restricciones. Además, hay que aumentar hasta 200 y 300 el número de generaciones para que se estabilice en un número de operaciones en concreto. Los porcentajes están expuestos en la tabla 2:

	Dos pedidos	Tres pedidos	Cuatro pedidos
Asignación de huecos	94%	90,5%	89,75%
Intercambio de operaciones	92,2%	93,7%	93,6%
Mutación de recurso	92%	94%	93%

Tabla 2: Resultados obtenidos para más de un pedido (operador puro).

5.3 Soluciones para llegar al 100% (operador híbrido).

Con el operador de solapamiento híbrido se han realizado unas pruebas para los 4 pedidos con los siguientes resultados:

- 1 pedido: 99% de aciertos
- 2 pedidos: 93,2% de aciertos
- 3 pedidos: 93,7% de aciertos
- 4 pedidos: 93,6% de aciertos

6 Conclusiones.

La originalidad del trabajo radica en conseguir un Algoritmo Genético (partiendo de un generador de Algoritmos Genéticos ya existente) que resuelva problemas de Scheduling basándose en su propia naturaleza de algoritmo de optimización.

La versatilidad del trabajo radica en que el Algoritmo diseñado es aplicable a una planta industrial standard: realiza la asignación de tiempos y recursos procurando respetar al 100% las restricciones y además optimiza la función objetivo. Las peculiaridades de la planta pueden ser añadidas a ésta como clases aparte, generando así más restricciones y ampliando otras clases como la operación. La ventaja de la programación en C++ es que el código es perfectamente exportable y estructurado, de tal manera que las ampliaciones son fáciles de realizar. Es por esto que el trabajo realizado, además de tener fines científico-técnicos, tiene fines industriales y prácticos.

Otra de las peculiaridades del algoritmo es el diseño del cromosoma, de fácil manejo.

Agradecimientos.

Deseo agradecer la estimable ayuda del Dr. D. Anselmo del Moral, quien me dirigió la tesis doctoral que ha surgido de todo este trabajo; a Jose María Lázaro, del centro de investigación de Labein, que me dirigió la parte práctica del trabajo; al Dr. D. Javier Zubillaga, quien me recomendó la presentación de la presente ponencia.

Referencias.

[1] Calle, E., Fuchs, T.(1993) "Path Planning and Task Scheduling", *5th Workshop of the Artificial Intelligence and Knowledge Based Systems for Space*, pp.90-103.

[2] Hamada, T. et al. (1993) "Hybridizing a Genetic Algorithm with a rule-based reasoning for Productin Planning", *IEEE Expert*, pp.39-54.

[3] Morton, T.E., Pentico, D.W. (1993) "Heuristic Scheduling Problems", *Applications to Production Systems and Project Management*.

[4] Reeves, C. R. (1993) "Modern Heuristic Techniques for Combinatorial Problems", *Blackwell Scientific Publications, Oxford*.