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

Índice

Algoritmo genético para la planificación conjunta de la producción y distribución de pedidos desde múltiples plantas y ventanas temporales de entrega

J. M. García [1], M. Calle[2], G. Villa[3]
[1]Ingeniero Informático, Dpto. Organización Industrial y Gestión de Empresas. Escuela Superior de Ingenieros. Universidad de Sevilla. Camino de los Descubrimientos, s.n., 41092 - Sevilla , jmgs@esi.us.es
[2]Ingeniero de Organización Industrial. mcalle@esi.us.es
[3]Ingeniero Industrial. gvilla@esi.us.es

RESUMEN

Este trabajo se ocupa del problema de seleccionar y planificar un conjunto de pedidos para ser fabricados e inmediatamente distribuidos. Para dicho proceso existen m plantas de producción y V vehículos para el transporte de los pedidos. Otras restricciones a ser consideradas son la capacidad limitada de producción en cada fábrica y las ventanas temporales dentro de las que debe ser servido cada pedido. Para el problema se describe un algoritmo genético y un método de solución exacta. Este método exacto es utilizado para obtener las soluciones óptimas de los problemas y medir el comportamiento del algoritmo genético. El método puede únicamente ser usado con instancias pequeñas. Los resultados demuestran que el algoritmo heurístico propuesto encuentra soluciones satisfactorias en un tiempo aceptable.

***Palabras clave**: Planificación, Producción y Distribución conjunta, Algoritmo Genético, Ventana temporal.*

1. Introducción.

En ciertos entornos industriales el abastecimiento de productos debe estar acorde con la demanda debido a la ausencia de inventarios del producto. Esta característica suele estar motivada por el carácter perecedero de los productos fabricados. Ejemplos se pueden encontrar en la producción y distribución de hormigón y en el reparto y preparación de comida rápida. Debido a la naturaleza de estos productos, los pedidos que se hagan del producto necesitan ser fabricados inmediatamente antes de ser repartidos al lugar de destino.

Este trabajo considera el problema de seleccionar y planificar un conjunto dado de pedidos. Los pedidos seleccionados deben ser preparados en una cualquiera de las plantas de producción existentes, e inmediatamente distribuidos al lugar del cliente. La entrega del pedido debe producirse dentro de una ventana temporal. Existen un conjunto de m plantas de producción $K=\{1...m\}$, cada una con una capacidad de producción limitada C_k. Consideramos capacidad de producción como el número de pedidos que pueden ser preparados simultáneamente, es decir, la producción de un pedido se considera un proceso continuo que requiere una unidad de capacidad durante su tiempo de proceso. Si la planta k tiene C_k unidades de capacidad, entonces a lo sumo C_k pedidos pueden ser preparados simultáneamente en dicha planta. Además, cada pedido i puede únicamente ser preparado en un subconjunto K_i del conjunto de plantas K.

Existe un número fijo de vehículos de idénticas características. Una vez que un pedido se prepara en la planta, se distribuye sin esperas al lugar de destino. Asumimos que el tamaño de cada pedido es inferior a la capacidad de un vehículo y que sólo un pedido puede ser distribuido en cada viaje que realice un vehículo. Cuando un pedido es descargado en el destino, el vehículo retorna a cualquiera de las plantas existentes y queda disponible para el reparto de otro pedido. Inicialmente, cada vehículo se encuentra en una planta específica. Se considera v_k como el número de vehículos inicialmente en la planta k.

El número de pedidos, sus tiempos de proceso en planta y los tiempos de viaje desde las localizaciones de destino a cualquiera de las plantas son datos conocidos y fijos. De ese modo, con cada pedido i, del conjunto de pedidos $P=\{1...n\}$, se asocia una fecha de entrega e_i, un tiempo desde cada planta k a su localización ti_{ik}, un tiempo de viaje desde esa localización hasta las plantas tr_{ik}, un tiempo de proceso o producción tp_i independiente de la planta, y un tiempo de descarga tu_i. Puesto que todos los tiempos de proceso (tiempos de producción y distribución) son conocidos de antemano, cada ventana temporal de entrega $[A_i, B_i]$ puede ser trasladada al comienzo del procesamiento en cada planta, de forma que contemos con una ventana temporal de comienzo $[a_{ik}, b_{ik}]$ en cada planta. La Figura 1 muestra el proceso gráfico asociado a la actividad de un pedido.

Con objeto de considerar el carácter perecedero de esta clase de productos, asumimos un instante ideal de entrega e_i dentro de cada ventana temporal. Podemos lógicamente trasladar también ese instante ideal a un instante ideal de comienzo en cada planta s_{ik}.

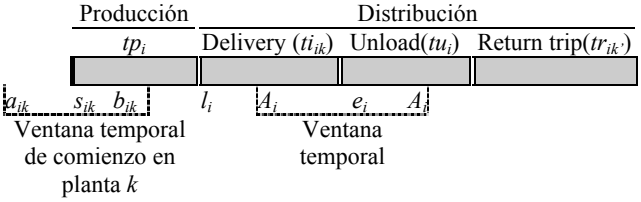

Figura 1: Actividad de un pedido

Debido a que tenemos una capacidad limitada C_k en cada planta así como un número finito de vehículos V, podría suceder que no fuera admisible servir todos los pedidos solicitados. Por ello, el objetivo en el problema será maximizar el beneficio asociado a la selección de pedidos realizada, asumiendo que cuando un pedido no es servido en su instante ideal de entrega, se produce un decremento del valor del pedido proporcional a la desviación. Sea W_i el beneficio asociado con servir el pedido i en su instante ideal e_i y sean w_i^- y w_i^+ penaltis por entrega anticipada y retrasada del pedido, respectivamente. Estos penaltis son usados para decrementar el valor del pedido cuando es servido antes o después de su instante ideal. De ese modo, cuando un pedido se sirve en el instante r, el beneficio del pedido viene a ser:

$$w_i = \begin{cases} W_i - (r - e_i)w_i^+ & \text{si } r \geq e_i \\ W_i - (e_i - r)w_i^- & \text{si } r < e_i \end{cases}$$

El trabajo está organizado como sigue. En la Sección 2 proponemos un método exacto basado en grafos para la obtención de soluciones óptimas. Sin embargo, este método sólo es válido para instancias pequeñas y de un número de recursos mínimo. Por ello, en la Sección 3

proponemos un algoritmo genético para resolver el problema. Los resultados computacionales se muestran en la sección 4. Finalmente, en la Sección 5 se presentan las conclusiones de este estudio.

2. Método de solución exacta basado en grafos

El enfoque que presentamos está basado en un método exacto para el mismo problema pero únicamente con una planta de producción [1]. El método construye un grafo *G* que colecciona las soluciones admisibles del problema a través de un método de evaluación de estados admisibles en la secuenciación de los pedidos. El camino máximo desde el nodo de inicio al nodo final en *G* proporciona la solución óptima del problema.

El proceso puede ser descrito como sigue: Consideremos los eventos correspondientes a los instantes de inicio de cada pedido en cada planta. Sea *D* el número total de esos eventos. El conjunto $\{t_d / d = 1,\ldots, D\}$ es usado para representar esos eventos en orden cronológico. Es decir, $\{t_d / d = 1,\ldots, D\} = \{r^j_{ik} : r^j_{ik} \in [a_{ik}, b_{ik}]; j = 1\ldots (b_i - a_i + 1), i = 1\ldots n, k = 1\ldots m\}$ y $t_{d-1} \leq t_d$ para $d = 1\ldots D$. Por tanto, $D = nm\sum_{i=1}^{n}(b_i - a_i + 1)$.

Para cada evento se crean nodos correspondientes a combinaciones legales de pedidos que están siendo procesados en una planta o distribuidos sobre uno cualquiera de los vehículos disponibles en ese instante en la planta correspondiente al evento. Cada nodo contiene dos vectores representando el estatus de las *m* plantas y de los *V* vehículos (Figura 2). El vector correspondiente a las plantas indica qué pedidos están siendo procesados en el instante correspondiente al evento. Cada pedido necesita una unidad de capacidad, por tanto el subvector asociado a cada planta *k* contiene C_k posiciones, es decir, C_k pedidos podrían ser procesados en un mismo instante en esa planta. Por otra parte, el vector asociado con los vehículos incluye la planta donde se localiza cada vehículo en el instante del evento (*Origen*) y, para los vehículos que están distribuyendo algún pedido, la información asociada con el envío (el evento correspondiente r^j_{ik} que define el pedido, el instante de inicio y la planta, además de la planta destino donde finalizará el vehículo después de servir el pedido).

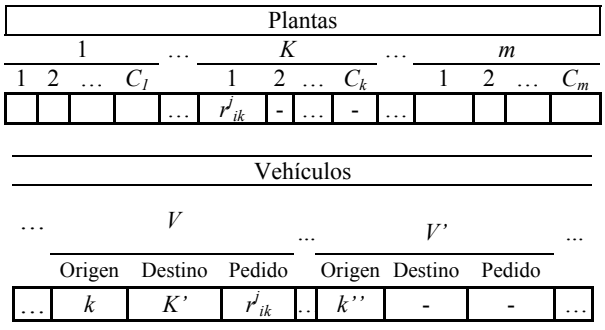

Figura 2: Representación de un nodo

En el grafo se considera un evento de inicio y finalización representados respectivamente por los nodos E_s y E_f. Estos nodos no poseen actividad alguna.

Supongamos que todos los nodos hasta el evento t_{d-1} han sido ya construidos. Sea el evento t_d el correspondiente al instante número j de inicio del pedido i en la planta k. Para construir el siguiente conjunto de nodos para el evento t_d se aplica la siguiente lógica: Para cada nodo n correspondiente a cada uno de los eventos anteriores, t_s s = 1...d-1, se actualiza el estado de las plantas y de los vehículos en el instante correspondiente al evento t_d y se evalúa si el subvector asociado a la planta k tiene posiciones libres para procesar el pedido i, y si existe algún vehículo libre localizado en la planta k. En caso afirmativo, se conecta el nodo n a m nuevos nodos que difieren en la planta de destino elegida para el vehículo que distribuye el pedido. Los nodos se conectan con arcos de peso igual al beneficio asociado a procesar el pedido i en el instante número j en la planta k.

Para el cálculo del número de nodos del grafo, nótese que el máximo número de nodos para el evento t_d es la suma de los nodos de los eventos previos multiplicado por el número de plantas m. Por tanto, existen $O(2m)^D$ nodos.

3. Algoritmo genético propuesto

Los algoritmos genéticos [2] son algoritmos de búsqueda y optimización basados en la teoría Darwiniana de la evolución. Esencialmente, un algoritmo genético es un procedimiento iterativo que mantiene una población de un conjunto de individuos que representan soluciones del problema. Una población inicial se genera aleatoria o heurísticamente. En cada iteración, se evalúan los individuos asignándoles un valor denominado *fitness*. Desde cada iteración a la siguiente, una nueva población con el mismo tamaño es generada mediante operadores evolutivos como son la selección, el cruce y la mutación. En lo que sigue describiremos la representación y operadores empleados en nuestro algoritmo.

3.1 Representación de un individuo: Genotipo y Fenotipo

Un concepto importante es la estricta separación de las restricciones del problema y el método evolutivo. Tal separación resulta en dos vistas completamente diferentes del individuo. Por un lado se encuentra la representación del fenotipo para evaluar el *fitness* del individuo. Por otro lado la representación del genotipo es aquélla con la que el algoritmo genético trabaja, es decir, es la representación codificada sobre la que se aplican los operadores evolutivos.

Un *string* será usado para representar el genotipo de un individuo. La longitud del *string* es el número de pedidos n. Cada posición en el *string*, denotada como gen, corresponde a cada pedido y contiene el instante en el que el pedido debería comenzar así como la planta donde sería procesado. Este instante se define con respecto al instante ideal de entrega del pedido. La figura 3 muestra el genotipo de un individuo para un problema con 5 pedidos y 2 plantas. Para ese individuo, el pedido 2 debería iniciarse en el instante s_{21}-1, el pedido 3 en el instante s_{32}+1 y el resto de pedidos en sus instantes ideales de inicio. Los pedidos 1, 2 y 5 serían procesados en la planta 1 y el resto en la planta 2.

pedido 1		pedido 2		pedido 3		pedido 4		pedido 5	
0	1	-1	1	1	2	0	2	0	1

Figura 3. Ejemplo de genotipo

Debido a la limitación de recursos (capacidad de producción y número de vehículos), no tenemos garantía de que todos los pedidos puedan ser servidos usando esos instantes de inicio y esas plantas. Por ello, se busca un subconjunto de pedidos que maximice el beneficio total. Esta situación representa un caso especial del problema de la producción y distribución conjunta con instantes fijos de entrega y varias plantas (PDPM) [3]. En dicho problema, es también complicado la obtención de soluciones óptimas, por lo que nos centramos en una heurística rápida que alcance soluciones satisfactorias. La heurística que a continuación presentamos explota la observación de que PDPM puede ser modelado como un problema de flujo a coste mínimo si la capacidad de producción es suficiente para procesar todos los pedidos que pueden ser distribuidos. Por ello, no se consideran las restricciones asociadas con no preparar simultáneamente en cada planta un número de pedidos mayor a la capacidad de producción.

La construcción del grafo directo G usado para resolver este problema se describe a continuación: Para cada planta, existe un nodo en G. Representamos esos nodos como $\{n_1 \ldots n_k \ldots n_m\}$. Para cada pedido i, creamos un subgrafo G_i con la siguiente estructura: Nodos n_{is} y n_{ie} definen el inicio del procesamiento en planta definido en el genotipo y el fin de la descarga, respectivamente. El valor del pedido i se representa como un coste negativo $-w_i$ en el arco que conecta n_{is} y n_{ie}. El valor w_i se calcula con respecto al instante de comienzo definido en el genotipo. Nodos n^i_{rk}, $k = 1 \ldots m$, representan la planta de retorno para el vehículo que sirve el pedido i. Puesto que un pedido es procesado a lo sumo una vez, la capacidad para cada arco en G_i es uno. Para cada nodo de planta n_k, existen arcos con una unidad de capacidad y coste cero, que conectan todos los nodos n_{is} que tienen que ser preparados en la planta k. Además, existe un arco desde cada nodo n^i_{rk} a todo nodo $n_{i'k}$ siempre que el instante de comienzo del pedido i' sea mayor que el instante en el que el vehículo retorna a la planta k después de servir el pedido i. Estos arcos tienen también coste nulo y pueden transportar una unidad de flujo. Todos los nodos n^i_{rk}, $i = 1 \ldots n$, $k = 1 \ldots m$, se conectan a un nodo final e con coste cero y una unidad de capacidad.

Pedido	W_i	w_i^-	w_i^+	w_i	s_{ik}	f_{i1}	f_{i2}
1	12	1	1	12	3	11	13
2	20	1	1	19	4	17	16
3	10	1	1	9	4	13	15
4	13	1	1	13	13	22	21
5	10	1	1	10	15	21	18

Tabla 1: Ejemplo

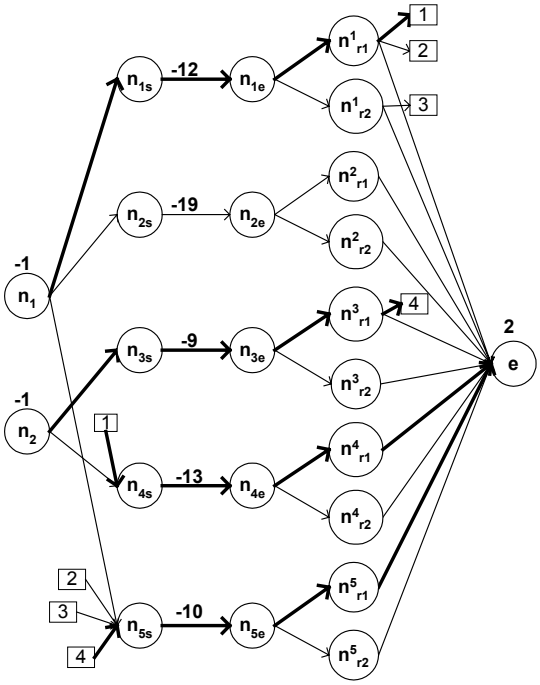

Figura 4: Grafo *G* correspondiente al genotipo de Figura 3.

Si inyectamos v_k (es decir, el número de vehículos inicalmente en la planta k) unidades de flujo en el nodo n_k, k=1…m, la solución óptima al problema de flujo a coste mínimo en G devolverá la secuenciación de un subconjunto de pedidos de valor total máximo. Un pedido i es servido si y sólo si en la solución del problema de flujo, una unidad de flujo pasa a través del arco (n_{is}, n_{ie}). La figura 4 muestra el grafo G correspondiente al genotipo mostrado en la figura 3 y referido al problema representado en la tabla 1, para un número de vehículos $V = 2$. Los datos f_{ik} representan los instantes en los cuales los vehículos retornarían a la planta k tras servir el pedido i. Las líneas gruesas en G denotan el camino de flujo a coste mínimo. Los pedidos seleccionados son el 1, 2, 4 y 5.

Una vez que se obtiene una solución para G, se chequea admisibilidad en cada una de las plantas ya que la solución obtenida únicamente garantiza una asignación admisible de vehículos. Para ello, hacemos uso del lema de Kroon para el problema de la secuenciación de trabajos fijos (*Fixed Job Scheduling Problem, FSP*) [4], el cual proporciona una condición necesaria y suficiente para la existencia de un plan admisible para todos los pedidos seleccionados en cada planta:

Sea O_k el conjunto de pedidos seleccionados para ser procesados en la planta k. Un plan admisible que incluya todos los pedidos pertenecientes a O_k existe si y sólo si el máximo grado de simultaneidad de O_k es menor o igual a la capacidad C_k en la planta.

Supongamos que los pedidos a ser procesados en el intervalo [0,T], el máximo grado de simultaneidad se define como sigue: $L = \max \{L_t: 0 \le t \le T\}$ con $L_t = \{i \in O_k: r_i \le t \le r^j_{ik} + tp_i\}$, , $r^j_{ik} \in [a_{ik}, b_{ik}]$.

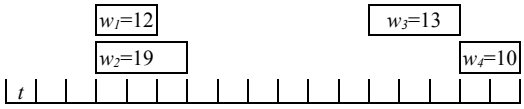

Figura 5: Fase de producción para un ejemplo con 4 pedidos a ser procesados en una planta

La Figura 3 presenta un ejemplo de un problema con cuatro pedidos. En la figura las barras indican el tiempo de producción (tp_i) de los pedidos. En este ejemplo L es igual a 2. Es claro que sólo si $C_k \geq 2$ podrán procesarse todos los pedidos en esa planta.

Si el máximo índice de solapamiento de O_k excede de C_k entonces la solución proporcionada por el problema de flujo a coste mínimo no es admisible. En este caso, la heurística intentará encontrar un subconjunto de pedidos con beneficio máximo que pueda ser procesado con una capacidad de C_k. Este problema es equivalente al problema de maximizar la secuenciación de trabajos fijos (Max. FSP). Max. FSP ha sido considerado por diversos autores [3] [4] [5], quienes muestran que el problema es resuelto por un algoritmo de flujo a coste mínimo.

La construcción del grafo *G'* que usamos en este trabajo es más directo que las construcciones propuestas por esos autores, y puede ser descrita como sigue. El conjunto $R=\{t_q: q = 1...Q\}$ se usa para representar todos los tiempos de comienzo de los trabajos que pertenecen a O_k en orden cronológico. El conjunto de nodos del grafo se corresponde uno a uno con el conjunto R, además de un nodo final. Existe un arco desde cada nodo al siguiente con coste cero y capacidad ilimitada. Por otra parte, existen arcos desde cada nodo al nodo correspondiente al primer pedido que podría ser producido en la planta, una vez que hubiera finalizado la producción del pedido representado por el nodo que es origen del arco. Estos arcos poseen capacidad para contener una unidad de flujo y el coste es igual a menos el beneficio por producir el pedido en la planta k, es decir, un coste igual a $-w_i$. En el primer nodo del grafo o nodo origen se inyectan C_k unidades de flujo, las cuales deben alcanzar el nodo final. Como ejemplo, en la figura 6 se muestra el grafo correspondiente al ejemplo de la figura 5.

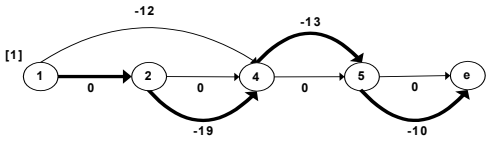

Figura 6: Grafo asociado al ejemplo de Figura 5

Una vez obtenida la solución a este problema de flujo, sea E el conjunto de los pedidos de O_k que no han sido seleccionados ($E=\{1\}$ para el ejemplo). Para cada pedido $i \in E$, se modifica el grafo inicial G, con objeto de prohibir que el pedido i pueda ser procesado en la planta k. Para este fin, simplemente necesitamos asignar capacidad cero al arco que une n_{is} con n_{ie}. Con este nuevo grafo G, se repite todo el proceso descrito hasta ahora hasta encontrar una solución admisible tanto en asignación de vehículos como en consumo de capacidad en cada planta.

3.2 Selección y reproducción de la población

Dos técnicas de reproducción son actualmente usadas en el campo de los algoritmos genéticos: generacional y *steady-state*. Brevemente, la reproducción generacional reemplaza todos los individuos de la población en cada iteración, mientras que la técnica *steady-state* reemplaza sólo unos pocos individuos.

En nuestro algoritmo hemos usado la técnica *steady-state*, reemplazando un individuo en cada iteración. Por tanto, en cada iteración se genera un nuevo individuo usando los operadores

descritos en la siguiente sección. En cada iteración un operador será seleccionado con una cierta probabilidad de ser elegido.

Para seleccionar el individuo a ser reemplazado, usamos un enfoque basado en un ranking exponencial [6] de los individuos de la población. El ranking exponencial asigna al peor individuo (peor *fitness*) una probabilidad de ser eliminado igual a p. Si éste no es seleccionado, entonces se asigna la misma probabilidad al siguiente con peor *fitness*, y así sucesivamente.

3.1 Operadores de cruce y mutación

El operador de cruce es el operador de combinación más importante para generar nuevos individuos, es decir, nuevos puntos de búsqueda. Se seleccionan dos individuos llamados padres y se produce, intercambiando partes de los padres, un nuevo individuo llamado *offspring* o hijo. Dados dos individuos padres *p1* y *p2* seleccionados aleatoriamente, el hijo se ha obtenido aplicando la siguiente regla: Sea $p1(j)$ el gen o posición j en *p1*. Para cada j desde 1 a $2n$ (número de posiciones), si $p1(j) = p2(j)$ entonces $hijo(j) = p1(j)$, en otro caso $hijo(j)$ es un valor aleatorio en el intervalo $[p1(j), p2(j)]$. En la figura 7 se puede ver un ejemplo del operador de cruce.

p1	-1	2	0	1	0	1
p2	-1	1	1	1	0	3
Hijo	-1	Aleat[1,2]	Aleat [0,1]	1	0	Aleat [1,3]

Figura 7: Operador de cruce

También se ha empleado un operador estándar de mutación que selecciona aleatoriamente un individuo y, aleatoriamente también, se elige un nuevo valor para una de sus posiciones.

4. Resultados computacionales

En esta sección presentamos y analizamos los resultados obtenidos en los experimentos llevados a cabo. La primera etapa en los experimentos involucró la generación aleatoria de una batería de problemas (sección 4.1). En la sección 4.2 se definen los parámetros del algoritmo y finalmente en la sección 4.3 se presentan los resultados obtenidos por la heurística respecto a las soluciones óptimas proporcionadas por el método de solución exacta.

4.1 Generación de problemas

Los tamaños usados fueron n = 10, 20, 30 y 40 pedidos. Diez instancias fueron aleatoriamente generadas para cada tamaño de problema. Las ventanas temporales $[a_i,b_i]$ fueron generadas aleatoriamente con tamaños entre 1 y 5 instantes de tiempo. El horizonte temporal de los problemas se consideró dependiente del número de pedidos, de acuerdo a los siguientes intervalos: [1,65] (10 pedidos); [1,80] (20 pedidos) ; [1,90] (30 pedidos); [1,100] (40 pedidos). Los valores W_i fueron también generados aleatoriamente dentro del intervalo [30,100]. De la misma forma, los penaltis por retraso w_i^+ y adelanto w_i^- fueron seleccionados en el intervalo [0,2]. La capacidad de producción C_k para cada planta en todos los problemas

fue 1. El número de vehículos inicialmente en cada planta fue generado también aleatoriamente.

4.2 Parámetros del algoritmo genético

Se usaron los siguientes parámetros para el algoritmo:
Población inicial obtenida de un modo aleatorio
Tamaño de la población = 20
Probabilidad de cruce = 0.6
Probabilidad *p* en el ranking exponencial = 0.2
Número de iteraciones = 1000

4.3 Resultados

La tabla 2 muestra, para cinco instancias características, el comportamiento del método exacto en relación al número de nodos y arcos y al tiempo de computación. Todos los tiempos de ejecución vienen expresados en segundos sobre un Intel Pentium 1,5 Ghz.

Instancia	*n*	*m*	*V*	Nº de nodos	Nº de arcos	Tiempo
1	10	2	2	6764	41780	22
2	10	3	2	73996	231244	806
3	20	2	2	37937	561692	1081
4	30	2	2	64179	1347160	2530
5	40	2	2	140781	6814809	16740

Tabla 2. Comportamiento del método exacto

La Tabla 3 muestra el sumario de resultados obtenidos por el algoritmo genético. Los porcentajes de error han sido computados respecto a los valores óptimos obtenidos con el método exacto. En todos los datos se ha tomado promedio sobre las 10 instancias resueltas de cada tamaño de problema *n*. El número de plantas *m* y vehículos *V* en todas las ejecuciones fue de 2.

		AG		
n	Promedio pedidos servidos	Error (%)	Nº soluciones óptimas	Tiempo de computación
10	5.1	0.74	9	398
20	6.9	1.09	7	502
30	8.2	0.59	5	710
40	9	2.12	4	923

Tabla 3: Sumario de resultados

El algoritmo genético obtuvo éxito en 25 de los 40 problemas resueltos. El promedio de error nunca excedió del 2.2%.

Respecto a los tiempos de computación, el método exacto empleó un tiempo excesivamente mayor al del algoritmo genético, a pesar de que con 10 pedidos, el método exacto empleó menos tiempo debido a que el tamaño del grafo no es todavía considerable.

5. Conclusiones

En este trabajo hemos estudiado un problema de planificación conjunta de la producción y asignación de vehículos de reparto con *no-wait* en proceso y ventanas temporales de entrega. Se ha propuesto un algoritmo genético para resolver el problema. Las soluciones proporcionadas por el algoritmo genético han sido empíricamente comparadas con las soluciones óptimas obtenidas a través de un método de solución exacta basado en grafos. Los resultados computacionales indican que el algoritmo genético encuentra soluciones de muy buena calidad en tiempos de computación aceptables.

Referencias

[1] Garcia J. M., Smith K., Lozano S., Guerrero F. and Calle M., "Production and Delivery Scheduling Problem with Time Windows," *Proceedings of the 30-th International Conference on Computers and Industrial Engineering*, Publishing ZITI, Tynos Island, Greece, pp. 263-268, 2002.

[2] Goldberg, D. E., "Genetic Algorithms in Search"*, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.

[3] Garcia J. M., Lozano S., Guerrero F., Smith K., and Calle M., "Coordinated scheduling of production and delivery from multiple plants," *Proceedings of the 12-th International Conference on Flexible Automation & Intelligent Manufacturing*, Dresden, Germany , 2002.

[4] Kroon L.G., Salomon M. and Van Wassenhove L. N., "Exact and approximation algorithms for the operational fixed interval scheduling problem," *European Journal of Operational Research*, 82, pp. 190-205, 1995.

[5] Arking E.M., Silverberg E.B., "Scheduling jobs with fixed start and end times," *Discrete Applied Mathematics* 18, pp. 1-8, 1987.

[6] Kaufmann, M., "Fundations of Genetic Algorithms", Morgan Kaufmann Publishers, San Mateo, California, 1991.