X Congreso de Ingeniería de Organización
Valencia, 7 y 8 de septiembre de 2006

Desarrollo de algoritmos genéticos, de recocido simulado e híbridos para la planificación de un taller flexible

Sara Lumbreras Sancho[1], Ángel Sarabia Viejo[1]

[1] Dpto. de Organización Industrial. Universidad Pontificia Comillas. C/ Alberto Aguilera, 25, 28015, Madrid. sara.lumbreras@icai.es, angel.sarabia@doi.upco.es .

Resumen

El problema del Jobshop ha sido objeto de investigación durante décadas debido a la diversidad de situaciones en las que aparece y a la complejidad computacional que presenta. Debido a esta última no resulta conveniente aplicar para su resolución los métodos clásicos de optimización, que resultan tan eficientes en otros casos. Los algoritmos metaheurísticos son preferidos para este tipo de problemas. Este artículo describe varios algoritmos creados para su resolución, basados en modelos como los algoritmos genéticos o el recocido simulado, pero que presentan peculiaridades originales que les confieren una eficiencia particular.

Palabras clave: algoritmo genético, recocido simulado, secuenciación, jobshop

1. Introducción

Los métodos clásicos de optimización no resultan adecuados para resolver los problemas NP-completos de la planificación, para los que consumen tiempos de cálculo en muchas ocasiones inaceptables. En estos casos es conveniente aplicar técnicas heurísticas, en las que se renuncia al óptimo global a cambio de obtener soluciones muy satisfactorias en tiempos asequibles. Una selección de estas técnicas puede encontrarse en Pham (1998).
La secuenciación de tareas en un taller es uno de las manifestaciones más interesantes de esta realidad, tanto por la diversidad de entornos en los que se presenta como por su complejidad, que ha motivado la aparición de una extensa literatura.
El objetivo de este trabajo es el desarrollo y estudio de nuevos algoritmos capaces de llevar a cabo la optimización de la planificación de un taller en tiempos que quedan muy por debajo de los empleados para la resolución mediante métodos clásicos. Estas diferencias de tiempos de cómputo son estudiadas y la posible desviación con respecto al óptimo global es acotada y estimada. Por último, se realiza un análisis de la solución aceptada para apoyar la toma de decisiones con respecto a posibles ampliaciones de capacidad en las instalaciones.

2. Un taller generalizado

El problema del taller, conocido como jobshop, consiste en la secuenciación de tareas para un conjunto de trabajos compuestos por varias operaciones distintas, empleando el conjunto de máquinas disponibles.
El caso en estudio es inusitadamente general, abarcando la mayoría de las posibles complejidades de los modelos deterministas. El taller es de tipo flexible, lo que implica que una operación puede realizarse en varias máquinas distintas, por supuesto con diferentes tiempos de ejecución. Además de tiempos de cambio dependientes de la secuencia, se han

considerado tiempos de transporte entre centros que pueden depender del tamaño de lote o del trabajo en particular.
La función objetivo a optimizar permite trabajar con magnitudes sencillas (como el tiempo total de proceso o el máximo retraso) o definir una función multicriterio en la que es posible, por ejemplo, establecer prioridades entre clientes para evitar retrasar los pedidos más importantes.
Además, se ha incluido una generalización adicional que permite trabajar con tiempos aleatorios mediante la definición de una función de distribución coherente, posibilitando así la resolución de un taller estocástico.
La posibilidad de fallo de alguna de las máquinas está también considerada, permitiendo la obtención de soluciones robustas.

3. Comparación con los métodos clásicos. Desarrollo de un modelo de programación lineal

Con el único objetivo de comparar los resultados obtenidos con los que arrojaría la aplicación de los métodos clásicos, se ha desarrollado un modelo de programación lineal implementado en GAMS que ya para tamaños del problema pequeños –cuatro trabajos con seis operaciones cada uno a realizar sobre tres máquinas- requirió tiempos de alrededor de más de 24 horas para su ejecución.

4. Codificación empleada

La codificación matricial que suele emplearse en este tipo de problemas –como en Mesoughni (2004)- presenta el inconveniente de permitir secuencias infactibles. De manera más específica, pueden aparecer referencias circulares entre operaciones, que resultan computacionalmente muy costosas de identificar y eliminar. La figura 1 muestra una secuencia aparentemente normal pero que no podría llevarse a cabo debido a la existencia de una referencia circular entre los trabajos 1 y 2.

M1:	O(1,2)	O(2,1)	O(3,3)
M2:	O(2,2)	O(1,1)	
M3:	O(3,1)	O(3,2)	

Figura 1. ***Ejemplo de referencia circular en una secuencia***

Las soluciones, en todos los heurísticos desarrollados, son codificadas en forma de una doble lista:

- Una *lista de trabajos*, que representa únicamente a éstos y no a sus operaciones. Cuando aparece el símbolo de un trabajo se pretende notar que su primera operación no realizada comenzará a trabajarse en la máquina correspondiente tan pronto como ésta se encuentre libre.
- Una *lista de máquina*, en paralelo con la anterior, que complementa la información previa con el dato de la máquina sobre la que ha de realizarse la operación que simboliza cada elemento de la *lista de trabajos*.

A partir de una solución codificada es relativamente sencillo obtener todos los datos de interés de la secuencia que representa. Por ejemplo, una solución válida para un taller con tres trabajos de dos operaciones y tres máquinas podría ser:

Trabajos: {1 3 1 2 2 3}

Máquinas:{2 3 1 1 2 3}

y la secuencia resultante podría representarse mediante un diagrama de Gantt como el de la figura 2, en la que a cada trabajo le ha sido asociado un color y las etiquetas representan cada operación realizada en el formato *(trabajo,operación)* (por ejemplo, "(3,2)" significaría la segunda operación del tercer trabajo).

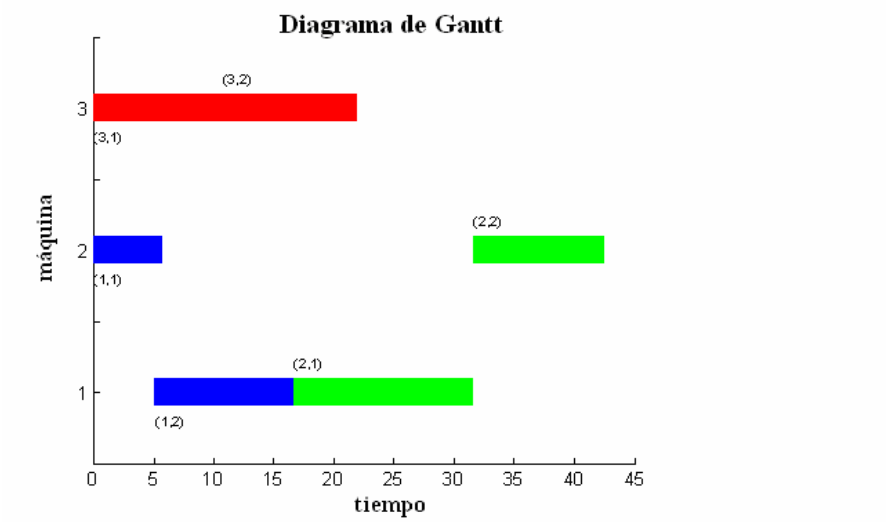

Figura 2. ***Representación de la secuencia ejemplo en un diagrama de Gantt***

Para que una solución codificada de esta manera sea factible, únicamente debe cumplir que la *lista de trabajos* sea una permutación con repetición de los símbolos empleados para los trabajos en el que cada uno aparezca tantas veces como operaciones lo componen. La *lista de máquinas* sólo debe respetar la flexibilidad de las máquinas del taller.

Estas comprobaciones son muy sencillas y pueden tenerse en cuenta en el algoritmo como una guía para la generación de la solución o soluciones aleatorias que se tomarán como punto de partida. Así, la estructura en doble lista imposibilita que aparezcan infactibilidades, reduciendo el espacio de soluciones a únicamente las factibles. Además, facilita considerablemente el desarrollo de los operadores básicos de los algoritmos y es interpretable de manera sencilla. Debido a esto, los tiempos de cálculo se reducen.

5. Un algoritmo genético original

Este modelo aplica la interpretación de los mecanismos de la evolución de los algoritmos genéticos usuales adaptándolos al problema junto con nuevas ideas que mejoran sensiblemente sus resultados. Las secciones siguientes explicitan estas peculiaridades.

5.1. Creación de la generación inicial

La población inicial de soluciones representa el punto de partida del algoritmo. La codificación utilizada facilita notablemente este proceso:

- La *lista de trabajos* se obtiene como una permutación aleatoria de una cadena en la que el símbolo correspondiente a cada trabajo aparece repetido tantas veces como operaciones lo componen.

- A partir de ella, la *lista de máquinas* se construye seleccionando aleatoriamente para cada posición un centro de trabajo del conjunto de los posibles.

Este proceso garantiza que todos los individuos serán factibles de antemano, evitando así todo proceso de comprobación posterior.
Es importante notar que el tamaño de la población influenciará notablemente el desarrollo del proceso de optimización. Por una parte, el tiempo de cómputo total del algoritmo es proporcional al número de iteraciones y al tamaño de la población. Al mismo tiempo, y al igual que en poblaciones biológicas, la diversidad es un factor clave. La probabilidad de que el algoritmo quede atrapado en óptimos locales crece cuando la población es pequeña. Las pruebas realizadas sugieren que la mejor estrategia es fijar el tiempo de ejecución y considerar un número prudente de generaciones (de 100 a 200, independientemente del tamaño del problema, ya que este tipo de algoritmos suele *agotarse* o detenerse en su proceso de mejora tras un número de iteraciones de ese orden). Así se trabajaría con el mayor número posible de individuos en la población, lo que garantiza aprovechar el tiempo disponible de manera óptima en eficacia y eficiencia.

5.2. Selección natural

La supervivencia de los individuos más fuertes se modela mediante una selección en ruleta. A las mejores soluciones se les asignan sectores mayores, de tal manera que unos dardos lanzados al azar tengan más probabilidades de seleccionarlos para que se reproduzcan.
Con el objetivo de garantizar que la mejor solución obtenida hasta el momento –la denominada *reina* en este modelo-no se pierda, independientemente de que ésta sea escogida para formar parte del conjunto de *padres*, se le realiza una copia que pasa directamente a la generación siguiente sin cambios.

5.3. Cruce

Los individuos seleccionados para ser padres se organizan por parejas al azar.
El cruce se implementa en dos pasos, uno para cada uno de los componentes de la doble lista. Si en las *listas de tareas* se aplicase un cruce bipuntual simple, los individuos *hijos* podrían resultar *monstruos* o soluciones infactibles. Por ello, un proceso sencillo identifica las operaciones sobrantes o faltantes en cada bloque intercambiado. Las primeras son eliminadas en su primera aparición en el bloque a intercambiar, mientras que las segundas se añaden al final en el orden en el que aparecían en la secuencia original. Esta comprobación y posterior modificación de los *cromosomas* es sencilla y rápida, evitando los problemas de infactibilidades originadas en otros operadores de cruce.
Las *listas de máquinas* se traducen primero en matrices que almacenan el centro de trabajo asignado a cada trabajo y operación. Después, se determina aleatoriamente una submatriz que es intercambiada entre ambos padres.
Mediante este proceso se garantiza la obtención de dos cromosomas *hijos* factibles que resultan realmente una combinación de sus padres. Sus características serán similares en los fragmentos que comparten y será posible la mejora en tanto las mejores partes de cada *padre* pueden ser seleccionadas para un *hijo.*

5.4. Mutación

El proceso de generación de mutaciones forma parte del conjunto de mecanismos cuyo objetivo consiste en la intensificación de la exploración de nuevas soluciones. Se introducen

cambios al azar en los cromosomas, preservando la diversidad y previniendo la convergencia temprana del algoritmo.
La mutación se implementa también en dos pasos. Por un lado, la *lista de máquinas* puede ser modificada cambiando el centro de trabajo en el que se ejecuta una operación a otro que disponga también de esa posibilidad. Los cambios en la *lista de trabajos* se estructuran como una permutación entre dos de sus elementos. Esta definición es intuitiva y responde a un concepto de mínima distancia, garantizando así que la solución mutada sea muy similar a la original.
La probabilidad de mutación es un parámetro de ajuste delicado, ya que si resulta demasiado bajo sus efectos no serán relevantes pero si es muy elevado el algoritmo será más parecido a una búsqueda aleatoria, con la pérdida de eficiencia que de ello se seguiría. Las pruebas realizadas sugieren que las tasas de mutación entre el 3 y el 6% arrojan los mejores resultados. El modelo desarrollado escoge un valor en este intervalo dependiendo de la desviación típica de los ajustes de la generación anterior, dando valores mayores para poblaciones más uniformes.

5.5. Inmigración

Este mecanismo tiene también por objeto la conservación de la diversidad. Se generan nuevos *individuos* aleatorios que pueden o no ser incluidos en la población. El mecanismo diseñado considera que los *inmigrantes* seleccionan aleatoriamente una *presa* de entre el conjunto de los *individuos* más débiles. Cada pareja se bate entonces en un duelo en el que las probabilidades de sobrevivir son proporcionales a la bondad del ajuste, resultando así una implementación particular del conocido como método de selección por torneo. Si el *inmigrante* resulta vencedor ocupa el lugar del individuo original.
De la misma forma que la tasa de mutación, el número de aspirantes a *inmigrantes* debe ser escogido de manera que se aprovechen sus efectos positivos sobre la diversidad pero sin elevarlo hasta el extremo de comenzar a perder eficiencia. Las pruebas realizadas señalan los valores entre el 5 y el 15% como los más adecuados. De nuevo, este valor se modifica durante la evolución del algoritmo de manera que se incremente cuando la diversidad disminuya.

5.6. Componente asexual en la reproducción

A imitación de los sistemas reproductivos de algunos insectos sociales, que alternan ciclos de reproducción sexual y asexual (partenogénesis), un nuevo mecanismo es introducido en este modelo. Una proporción de la nueva generación es obtenida mediante un proceso que imita la partenogénesis. Este mecanismo toma la *reina* como punto de partida para generar un conjunto de soluciones modificadas mediante el operador mutación. De esta manera se intensifica la búsqueda en las regiones más prometedoras, incrementando la eficiencia.
De manera análoga a los casos anteriores, el porcentaje de *hijos* obtenidos mediante este proceso es un parámetro que debe seleccionarse cuidadosamente. Esta vez el problema es el opuesto: si es demasiado alto el algoritmo perderá efectividad, resultando más fácilmente atrapado en óptimos locales. Las pruebas señalan el 10-15% como valores entre los que debería oscilar esta proporción. La figura 3 ilustra el efecto que esta tasa tiene sobre el algoritmo.

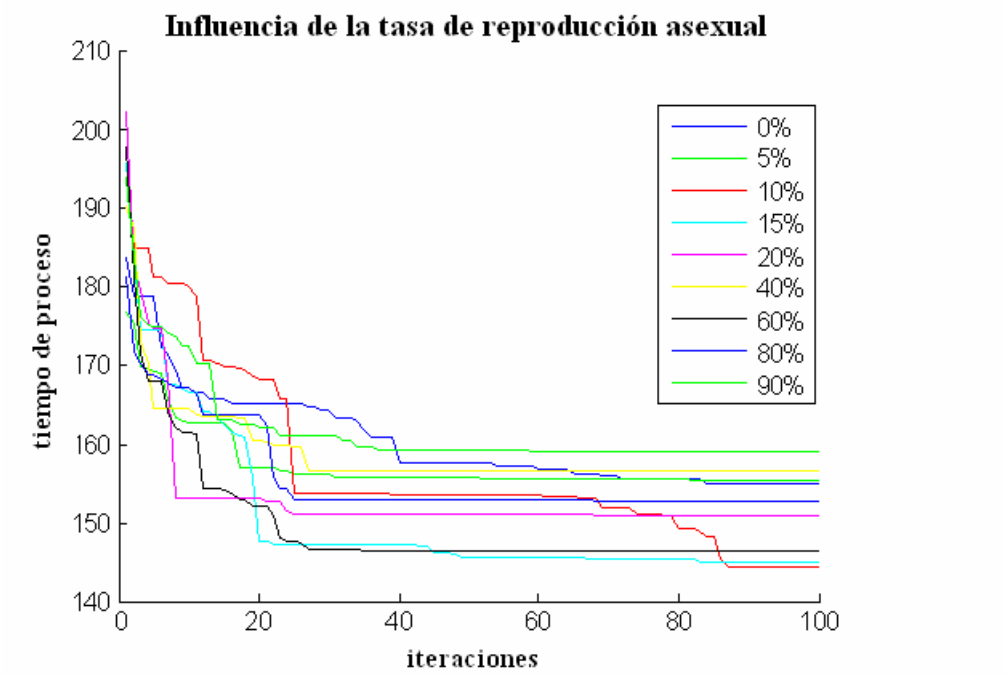

Figura 3. ***Comportamientos del algoritmo en sus primeras iteraciones para distintas tasas de reproducción asexual***

6. Algoritmo de recocido simulado

Los algoritmos de recocido simulado (S.A.) imitan los procesos metalúrgicos del recocido en el acero o el cobre. El modelo desarrollado aporta algunas ideas innovadoras a las implementaciones clásicas de estos algoritmos.

6.1. Obtención de soluciones *vecinas*

En estos heurísticos resulta de vital importancia conseguir generar *vecinos* que estén situados a una distancia mínima del punto de origen para que estén directamente relacionados con él pero puedan presentar valores de ajuste diferentes. Esto se consigue de manera análoga al procedimiento descrito en la sección dedicada a la mutación, conservando como ventaja importante que no es necesario hacer comprobaciones ni correcciones posteriores.

6.2. Mecanismo de búsqueda local

Para mejorar la eficiencia, el algoritmo de recocido contiene un mecanismo de búsqueda local que, al seleccionarse una solución, evalúa un conjunto de puntos de su entorno, desplazándose a ellos en el caso de que se encuentre una dirección de mejora. Si la mejora se produce, el proceso de búsqueda local se repite sin variación de la temperatura. Sin embargo, si no ha resultado satisfactorio se procede al mecanismo clásico de recocido en el que la probabilidad de un movimiento de empeoramiento se determina por su magnitud y por el valor de la temperatura según la ley de Boltzmann.
El tamaño del conjunto de *vecinos* que serán inspeccionados debe ser suficientemente grande como para tener impacto en los resultados pero suficientemente pequeño como para no disminuir la eficiencia. Las pruebas realizadas aconsejan valores en torno a los 30 puntos.

6.3. Planes de enfriamiento

De entre la extensa literatura publicada sobre la evolución que debería seguir la temperatura a lo largo del algoritmo puede concluirse que los planes demasiado sofisticados no resultan eficientes, siendo superados en la gran mayoría de los casos por una simple regla geométrica de determinados parámetros. Sin embargo, estos valores son difíciles de ajustar y el resultado es muy sensible a ellos, de manera que en muchas ocasiones se descarta aplicar la regla geométrica por esta dificultad.
El primero de los planes de enfriamiento propuestos intenta imitar el que resulta efectivamente escogido en la industria metalúrgica –resulta curioso comprobar que, mientras que el proceso del temple ha sido sometido a estudios exhaustivos para mejorar las propiedades de la pieza final, para el recocido se considera suficiente dejar enfriar el metal al aire o en el horno- que resulta ser el correspondiente a la convección natural. La discretización de este proceso lleva efectivamente a la regla geométrica, pero al tener en cuenta las propiedades de los transitorios de primer orden se obtienen pautas sencillas para determinar los parámetros que resultan en algoritmos eficientes y robustos:

$$c = 1 - \frac{1}{\tau} \tag{1}$$

$$6 \cdot \tau = n \tag{2}$$

donde c representa el parámetro de la geométrica, n simboliza el número máximo de iteraciones permitido y τ. la constante de tiempo del transitorio, que es empleada como un parámetro intermedio.
Una segunda alternativa es introducida en forma de plan dinámico, que fija el valor de la temperatura teniendo en cuenta el valor del ajuste de la solución en cada momento, de manera que la probabilidad de un salto en contra de la pendiente de determinada magnitud desciende mediante una regla geométrica que se define mediante las sencillas reglas descritas en (1) y (2). El plan de enfriamiento obtenido resulta en un algoritmo alternativo al anterior que resulta más eficiente en algunos de los casos simulados.

7. Algoritmos híbridos

Las características positivas de los heurísticos descritos se han combinado para obtener híbridos que resultan interesantes desde las perspectivas de eficacia y efectividad.

7.1. Híbrido en dos fases

La forma más sencilla de combinar los algoritmos genéticos y de recocido descritos es ejecutarlos secuencialmente. Al agotarse antes, el genético será el que comenzará. Una vez la mejora comience a ser más lenta, pasará el relevo al recocido. Determinar el momento del cambio no es sencillo. Las pruebas sugieren que es conveniente fijar un número mínimo de iteraciones que por fuerza serán realizadas (unas 150-200 independientemente del tamaño del problema). Después, la alarma de *agotamiento* puede dispararse, y el cambio se realiza una vez se han sucedido 50-75 generaciones sin que se consiguiera mejora alguna.
Después, el recocido se realiza sobre una familia de soluciones que está formada por los miembros más prometedores de la etapa anterior. Trabajar con una población y no con una solución aislada mejora notablemente la efectividad del recocido. El tamaño de la familia ha de escogerse cuidadosamente, siendo preferidos los valores en el intervalo 6-12 para tamaños del problema de alrededor de 10 trabajos, pero deberían incrementarse para problemas mayores. La figura 3 muestra un ejemplo de ejecución, mientras que la figura 4 destaca el

principal problema de este algoritmo: aunque garantiza una mayor exploración que los anteriores, su eficiencia es mucho menor.

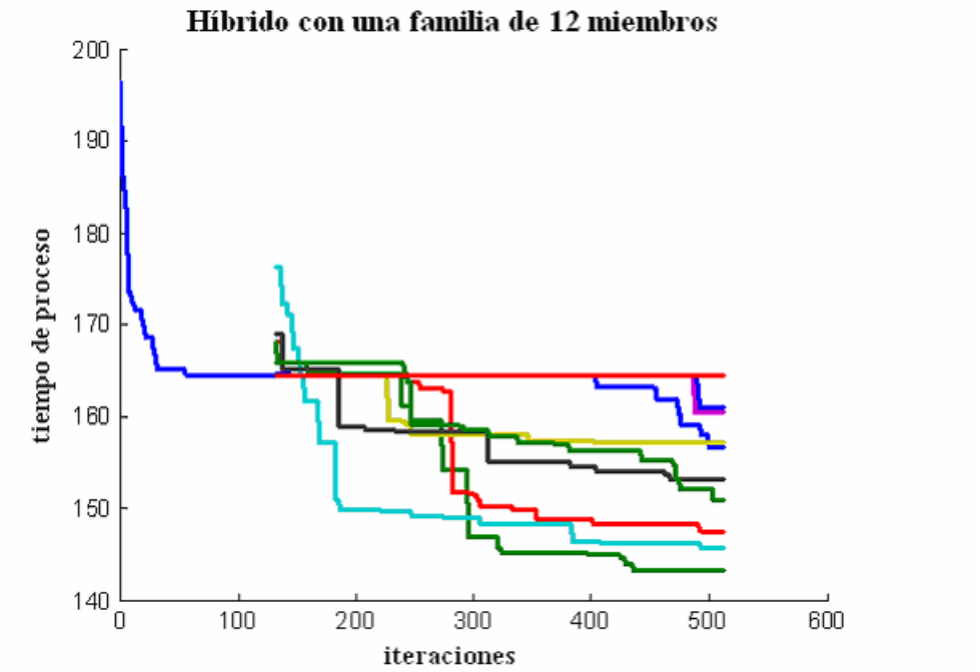

Figura 3. ***Ejemplo de ejecución del híbrido en dos fases con una familia de 12 miembros***

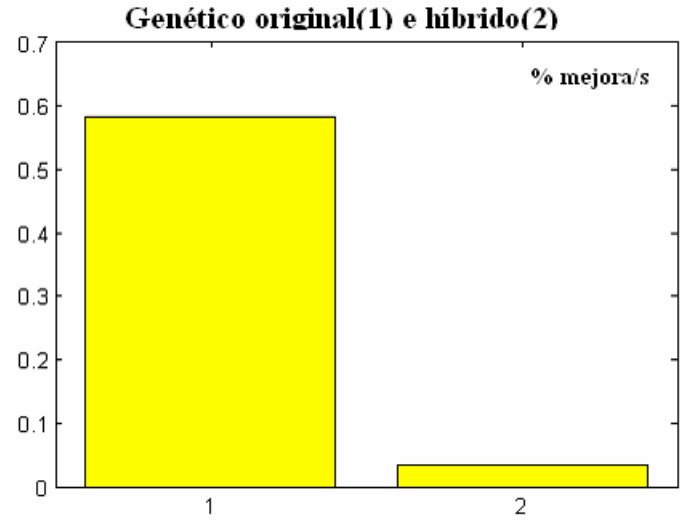

Figura 4. ***Eficiencia comparada del genético original y el híbrido secuencial***

7.2. Mutación no darwiniana

El segundo híbrido propuesto se aparta de la teoría clásica de la evolución –el estándar seguido por los algoritmos genéticos- para considerar que los cambios aleatorios que se introducen en forma de mutaciones no son tales, sino que la probabilidad de que resulten en una mejora es mayor que la de que el resultado empeore. Así, se introduce una pequeña ejecución de recocido en el operador mutación, que garantiza que la solución obtenida habrá explorado sus vecinos y pasado por el mecanismo de búsqueda local descrito y el método de recocido clásico. Un ejemplo de ejecución se incluye en la figura 5.

Este elemento intensifica la búsqueda en las regiones más prometedoras, resultando en un algoritmo más eficiente que ninguno de los anteriores en las primeras generaciones, siendo después superado por el genético darwiniano (cuando se ha alcanzado un óptimo local, y para seguir avanzando, sería necesario inspeccionar puntos nuevos, incrementando la diversidad). Así, se diseño un tercer híbrido que conmuta del genético no darwiniano al más clásico después de unas cuantas generaciones (150-200), alcanzando los mejores resultados en cuanto a eficiencia de todos los modelos presentados. La figura 6 muestra una comparación en eficiencia de los tres híbridos desarrollados.

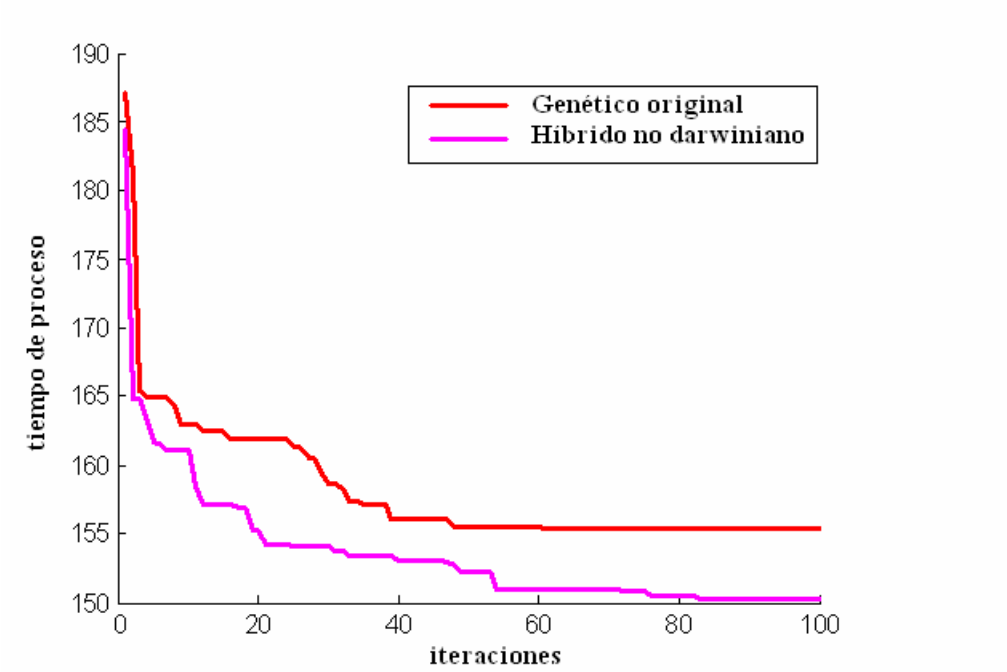

Figura 5. ***Ejemplo de ejecución de las primeras iteraciones del genético original y el no darwiniano***

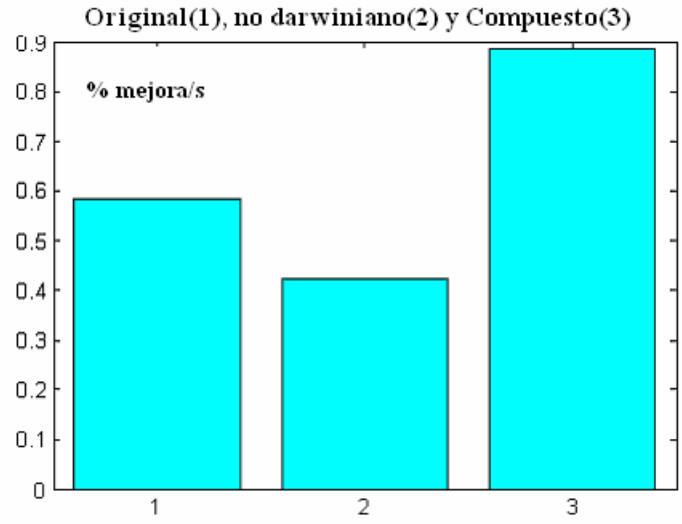

Figura 6. ***Eficiencia comparada entre los tres híbridos desarrollados***

8. Otras salidas de los modelos

Además de devolver los datos correspondientes a la solución obtenida, el avance del algoritmo en el tiempo e información estadística sobre la evolución de la población, los modelos desarrollados proporcionan otras salidas interesantes sobre el problema.

8.1. Acotamiento y estimación del error cometido

Además de una cota superior del error se facilita una estimación del mismo, que se basa en el ajuste de una distribución no paramétrica a los resultados obtenidos a lo largo de la ejecución del algoritmo. Este tipo de distribuciones no habían sido nunca aplicadas al problema de estimación de errores de heurísticos, arrojando esta herramienta resultados notablemente buenos.

8.2. Aplicación de la Teoría de las Limitaciones

La Teoría de las Limitaciones (Goldratt (1982)) se introduce en una herramienta que analiza la evolución del trabajo en curso, identifica cuellos de botella en el taller para la carga de trabajos considerada y aconseja dónde realizar nuevas inversiones de aumento de capacidad.

9. Otras generalizaciones de los modelos: taller estocástico y taller robusto

Los modelos desarrollados están preparados para trabajar con un taller estocástico, en el que los tiempos aleatorios se simulan a partir de una distribución que surge como combinación del dato del tiempo determinista y una componente aleatoria proporcional a él que se obtiene de una Weibull.
Además, los algoritmos tienen la posibilidad de considerar el fallo de alguna de las máquinas (encontrando la secuenciación de tareas alternativa óptima, así como indicando los trabajos cuya realización no será ya posible). De esta manera, mediante la introducción de una lista con los fallos más frecuentes y sus probabilidades respectivas es posible encontrar la solución robusta para el taller considerado siguiendo un proceso similar al presentado en Jensen (1999).

Referencias

Mesghouni, K, S Hammadi and P. Borne (2004): *Evolutionary Algorithms for Job-Shop Scheduling*. Int. J. Appl. Math. Comput. Sci. Vol. 14, No. 1, 91–103.

Jensen, T. and T. K. Hansen.‘*Robust Solutions to Job Shop Problems’*. Proceedings of the 1999 Congress on Evolutionary Computation, 1999.

Pham, D.T. and D. Karaboga (1998): *Intelligent Optimization Techniques*. Springer-Verlag

Goldratt, E.‘*The goal. An ongoing process’*(1982). North River.