Una propuesta de una herramienta flexible para problemas de scheduling

Guillermo Ordoñez y Guillermo Leguizamón

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional*
Línea: Metaheurísticas

Departamento de Informática
Universidad Nacional de San Luis
San Luis, Argentina
{gordonez,legui}@unsl.edu.ar

Resumen

El problema general de scheduling representa un desafío para su resolución dado que es un problema inherentemente difícil. Por otro lado, existe una gran variedad de enfoques cuya efectividad depende del tipo, tamaño y otras características del problema de scheduling. Tales enfoques incluyen métodos tradicionales de investigación operativa, búsqueda local y sus diferentes versiones (por ejemplo, simulated annealing y tabu search) y más recientemente diversas meta heurísticas bio-inspiradas (computación evolutiva y ant colony optimization, entre otras). La propuesta presentada aquí incluye una herramienta flexible basada en algoritmos evolutivos para ser aplicada a un amplio conjunto de tipos de problemas de scheduling.

1. Introducción

El problema de scheduling es de gran importancia práctica, sin embargo gran parte de los problemas de scheduling son del tipo $\mathcal{NP}$-Completos. En un problema de scheduling existen *t* tareas($t = |T|$) y *m* máquinas($m = |M|$) y cada una de estas tareas debe ejecutarse en una máquina perteneciente a $m(t_i)$ incluido en *M*. Un scheduling es factible si no existe superposición de intervalos de tiempo de distintas máquinas (dos tareas no pueden ser procesadas en una misma máquina al mismo tiempo), si cada tarea es procesada exactamente por exactamente una máquina a la vez (la cual debe pertenecer a $m(t_i)$) y además, si se cumplen las restricciones adicionales especificadas para el problema particular que se está resolviendo (por ejemplo, cada tarea podría tener asociado un tiempo mínimo de inicio).

Dependiendo de las restricciones, las cardinalidades de $m(t_i)$, M y las restricciones adicionales (principalmente el conjunto de tareas predecesoras) existen varias clasificaciones para los problemas de scheduling. La mayoría de las clases pertenecientes a estas clasificaciones describen problemas del tipo $\mathcal{NP}$-Completos[7] (por ejemplo Job Shop Scheduling y Open Shop Scheduling). Es por esto que para

*El laboratorio es dirigido por la Lic. Susana Esquivel y subvencionado por la Universidad Nacional de San Luis y la ANPCyT (Agencia Nacional para la Promoción de la Ciencia y la Tecnología).

gran parte de estas clases de problemas han sido aplicados diferentes enfoques que incluyen técnicas tradicionales de investigación operativa, búsqueda local y sus diferentes versiones (simulated annealing, tabu search)[1] y más recientemente diversas técnicas meta heurísticas que entre otras, incluyen a los algoritmos evolutivos[2, 3, 9, 10]. La característica común de estas herramientas es que en general están diseñadas para una clase particular de problemas de scheduling. Lo que se pretende con la herramienta flexible es poder resolver la mayor cantidad de problemas de scheduling y no una clase determinada (al menos la mayoría de los entornos de máquinas propuestos por Pinedo [14] y posibles combinaciones de éstos).

En función de la propuesta cabría preguntarse: ¿cuál es la necesidad de crear una nueva herramienta para resolver problemas de scheduling?, Como ya dijimos, las herramientas existentes son para problemas de scheduling específicos. Por lo tanto, la primer ventaja que brindará nuestra herramienta es la de su aplicabilidad a a una importante familia de problemas de scheduling. Sin embargo, esta no es la principal ventaja, sino que existen varios tipos de problemas de scheduling que no han sido estudiados o para los cuales no es fácil encontrar una herramienta que los resuelva. En síntesis, nuestra herramienta apunta a ser lo suficientemente robusta y efectiva para ser aplicada a diversos tipos de problemas de scheduling. Debido a que gran parte de los problemas de scheduling pertenecen a la familia de problemas $\mathcal{NP}$-Completos y donde una solución cercana a la óptima es aceptable, los algoritmos evolutivos conforman uno de los posibles enfoques apropiados para ser usados en este contexto.

Los AE (Algoritmos Evolutivos) son algoritmos de búsqueda inspirados en la genética de las poblaciones naturales. Ellos mantienen una población de individuos que representan las soluciones candidatas. Dicha población evoluciona en el tiempo a través de la competencia entre los individuos y una variación controlada de los mismos. La aplicación de AEs incluye un amplio rango de problemas de optimización y aprendizaje de máquina en variados dominios. Los Algoritmos Evolutivos han sido aplicados con diferente grado de éxito a problemas de scheduling de distintos tipos. Davis [6] fue el primero en proponer el uso de AEs para resolver JSS [5]. Al mismo tiempo, diferentes versiones de AEs aplicados a un problema de ordenamiento, tal como el Problema del Viajante de Comercio, dieron lugar a representaciones avanzadas, muchas de ellas adecuadas para ser usadas en muchos problemas de scheduling. Un ejemplo de estas representaciones es la permutación de tareas la cual tiene asociado un conjunto considerable de operadores de entrecruzamiento a ser aplicados: PMX, OX, CX, etc. [8, 10, 12, 13].

En la siguiente sección se presentan los objetivos del trabajo. La sección 3 describe el funcionamiento general de la herramienta y las interacciones entre las componentes de la misma. La sección 4 analiza una posible situación a considerar respecto de instancias existentes y la manera de resolverla. En la sección 5 se enumeran las etapas ya realizadas. Finalmente se presentan las conclusiones.

2. Objetivos del trabajo

El objetivo del trabajo es construir una herramienta integrada para ser aplicada a una gran variedad de problemas de scheduling. Para esto, primero fue necesario diseñar un lenguaje que permitiera especificar una instancia de un determinado tipo de problema de scheduling. Dicha instancia conforma la entrada de un algoritmo evolutivo encargado de realizar la búsqueda en el espacio de soluciones del problema. La característica fundamental del algoritmo evolutivo es el esquema de representación de las soluciones. Dicho esquema está diseñado para adaptarse a una importante variedad de problemas de scheduling especificados a través de un lenguaje *ad hoc*.

En la actualidad hay una gran variedad de herramientas y métodos para resolver distintos problemas de scheduling, pero éstas, en general están diseñadas para un tipo particular de problema de scheduling, la herramienta que proponemos presenta las siguientes ventajas:

- existen algunos paquetes que brindan la posibilidad de resolver varios de los problemas clásicos de scheduling, la herramienta que se propone también brindaría esta posibilidad, teniendo

así una única herramienta que puede ser aplicada a todos estos problemas. A diferencia de otras herramientas, ésta no es una colección de algoritmos, sino que se realiza una generalización de problemas de scheduling permitiendo resolverlos a través de un método flexible.

- hay varios tipos de problemas de scheduling que no han sido estudiados o para los cuales es difícil encontrar una herramienta que los resuelva, con esta herramienta sería posible realizar un estudio sobre estos problemas sin mayores complicaciones, simplemente deberá especificarse las instancia utilizando el lenguaje propuesto.

3. Funcionamiento general de la herramienta

La idea es bastante simple, se usa un algoritmo evolutivo que toma una instancia de un problema de scheduling. Dicha instancia está descripta en un archivo con un formato especial (este tipo de archivo que especifica en un nivel bajo un problema de scheduling utilizando tablas, lo denominaremos IS-T) y la salida la entrega en un archivo que incluye la mejor solución encontrada y un conjunto de datos estadísticos que nos sirven para evaluar el comportamiento del AE, (ver figura 1).

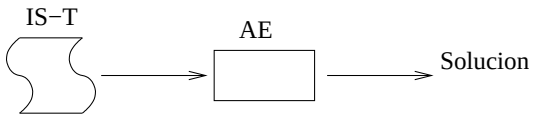

Figura 1: IS-T representa la instancia en un lenguaje de bajo nivel.

El problema es que este tipo de archivos (IS-T) está formado por un conjunto de tablas bastante complicadas, lo que torna difícil su especificación y lectura, sobre todo cuando las instancias son de gran tamaño y por consiguiente cualquier error puede llevar a alguna inconsistencia en la entrada. Por esto, y para facilitar la tarea de especificación de las instancias, se creó un segundo lenguaje bastante intuitivo, y junto con éste se implementó un compilador que traduce las instancias de este tipo de archivos (al que a partir de ahora denominaremos IS-LSch) a un archivo del tipo IS-T (quedando la herramienta como se muestra en la figura 2).

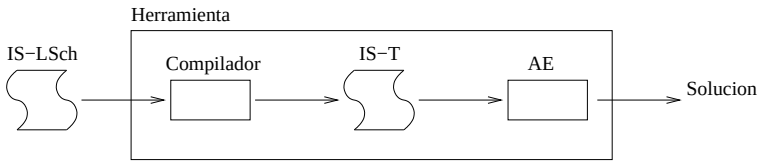

Figura 2: Incorporación de IS-LSch

Por el momento vamos a olvidarnos de los archivos IS-T, y cuando hagamos referencia a la herramienta nos referiremos al comportamiento combinado del compilador y el AE. De esta forma, para obtener la solución de un problema, un usuario simplemente debería armar el archivo IS-LSch e invocar a la herramienta (ver figura 3).

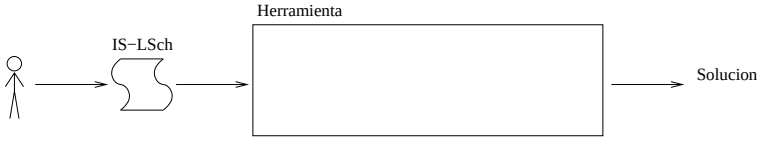

Figura 3: Herramienta desde una perspectiva global.

4. Instancias de problema de scheduling

La especificación, para un estudio posterior, de nuevas instancias de problemas de scheduling deberían ser realizadas usando el lenguaje IS-LSch. Sin embargo, existe una gran cantidad de problemas de scheduling para los cuales están disponibles un conjunto muy importante de instancias con sus respectivos mejores valores conocidos u óptimos (lo que se conoce en la jerga como *benchmarks*)[4]. Estas instancias podrían ser utilizadas en nuestra herramienta como parte del estudio de su aplicabilidad, pero no de forma directa ya que los formatos en las que están especificadas son distintos a los usados en la herramienta bajo desarrollo. Como una manera de automatizar y acelerar el proceso de su aplicación, se implementaron varios traductores simples que pasan de varios formatos originales al formato de los archivos IS-LSch. La figura 4 muestra los traductores sombreados.

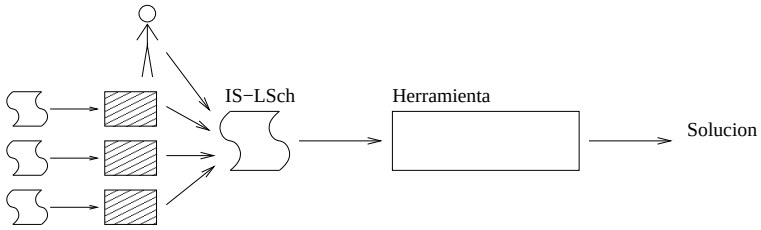

Figura 4: Componentes adicionales de la herramienta. Traductores de un determinado formato al formato IS-LSch.

5. Avance actual

Desde el punto de vista del diseño de la herramienta se ha realizado lo siguiente: diseño de las componentes principales y sus inter relaciones; diseño de los lenguajes de bajo y alto nivel, IS-T y IS-LSch respectivamente, para especificar problemas de scheduling; diseño de la forma generalizada que tendrá el cromosoma dentro del algoritmo evolutivo; diseño del algoritmo de *schedule builder* que permite obtener a partir de un cromosoma un único *schedule* para el problema en cuestión, y finalmente; diseño de una función de optimización que se aplicará, dependiendo de las características de los objetivos del problema en estudio.

6. Conclusiones

En este artículo hemos presentado las aspectos más relevantes de una herramienta flexible para problemas de scheduling. La flexibilidad de la herramienta está dada por la posibilidad de ser aplicada a distintos problemas de scheduling usando un enfoque evolutivo cuyo diseño contempla una representación generalizada de las soluciones. Dicha representación dentro del enfoque evolutivo permitirá su aplicación a un amplio espectro de problemas de scheduling. Detalles de cada una de las componentes de la herramienta y resultados de su aplicación, serán comunicados en futuras presentaciones acorde al grado de avance de la propuesta.

Referencias

[1] Aarts, E. & Lenstra, J.K. - Editores, (1997) - Local search in Combinatorial Optimizatio. John Wiley & Sons.

[2] Bäck T., Fogel D., Michalewicz Z. (1997) Handbook of Evolutionary Computation. Oxford University Press.

[3] Bäck T. (1997) Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.

[4] Beasley, J. 1990) - OR Library: distributing test problems by electronic mail. WWW site at `http://www.ms.ic.ac.uk/info.html`

[5] Bruns, R. (1993) - Direct representation and advanced genetic algorithms for production scheduling. Proceedings of the Fifth International Conference on genetic Algorithms. San Mateo, Morgan Kaufmann.

[6] Davis, L. (1985) - Job Shop Scheduling with Genetic Algorithms. Proceedings of the First International Conference on Genetic Algorithms, pages 136-140. San Mateo; Morgan Kaufmann.

[7] Garey, M. & Johnson, D. (1979) - Computers and Intractability -A guide to the theory of $\mathcal{NP}$-Completeness. Freeman.

[8] Goldberg, D. (1989) - Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

[9] Hart, E. (1997) - The State of the Art in Evolutionary Approaches to Timetabling and Scheduling, Department of Artificial Intelligence, University of Edinburgh. http://www.dcs.napier.ac.uk/evonet.

[10] Michalewicz, Z. (1996) - Genetic Algorithms + Data Structures = Evolution Programs. Third, revised and extended version. Springer-Verlag, USA.

[11] Nakano, R. & Yamada, T. (1991) - Conventional genetic algorithm for Job-Shop Scheduling. Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, Morgan Kaufmann.

[12] Ordóñez, G. & G. Leguizamón. (2001) - . Representaciones Indirectas en Algoritmos Genéticos y la Alcanzabilidad de Schedules Semi-Activos. VII Congreso Argentino de Ciencias de la Computación. Octubre de 2001, El Calafate, Santa Cruz, Argentina. Vol. II Pag. 1229.

[13] Ordóñez, G. & G. Leguizamón. (1999) - Representaciones Indirectas en Algoritmos Genéticos aplicados a un problema Scheduling. V Congreso Argentino de Ciencias de la Computación. Octubre de 1999, Tandil, Argentina.

[14] Pinedo, Michael.(1995) - Scheduling. Theory, Algorithms and Systems. Columbia University.