DIFERENTES HEURÍSTICAS PARA LA SOLUCIÓN DE PROBLEMAS DE MÁQUINA ÚNICA EN ENTORNOS DINÁMICOS

de San Pedro M., Pandolfi D., Lasso M., Villagra A.
Laboratorio de Tecnologías Emergentes (LabTEm)
Proyecto UNPA-29/B064[1]
Unidad Académica Caleta Olivia
Universidad Nacional de La Patagonia Austral
Ruta 3 Acceso Norte s/n
(9011) Caleta Olivia – Santa Cruz - Argentina
e-mail: {edesanpedro,dpandolfi,mlasso,avillagra}@uaco.unpa.edu.ar
Tel/Fax : 0297 4854888 int 144

Esquivel S.
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 – Local 106
(5700) San Luis – Argentina
e-mail: esquivel@unsl.edu.ar
Tel : 02652 420823 / Fax : 02652 430224

RESUMEN

Existen dos conceptos de importancia en el contexto de problemas dinámicos y en particular de *scheduling* dinámicos: la búsqueda de soluciones robustas y flexibles. El concepto de robustez de las soluciones se refiere a un tipo de soluciones que pueden ser usadas de igual manera cuando se produce un cambio en el entorno y manteniendo su calidad relativa. En el caso de flexibilidad, se refiere a la posibilidad de que las soluciones encontradas puedan ser adaptadas sin mayores problemas cuando se produzca un cambio en el entorno. En consecuencia, soluciones robustas y flexibles son altamente deseables en este tipo de contexto.

Para un problema de *scheduling* se podrían presentar dos niveles de dinamismo: uno de ellos establece un dinamismo en el cual el problema puede ser dividido en varios problemas estáticos, llamado "dinamismo *off-line*" [33], [3]; el otro nivel de dinamismo apunta a estudiar el comportamiento de un algoritmo cuando se producen cambios en el entorno, pero durante el proceso de búsqueda y se lo denomina "dinamismo *on-line*".

Los problemas de *scheduling off-line* han sido estudiados por distintos investigadores usando para su resolución distintas metaheurísticas: *Simulating Annealing, Tabu Search*, *Algoritmos Evolutivos y Ant Colony Algorithms*. Pero pocos han encarado los problemas de *scheduling on-line;* para problemas de *job shop* [4], [5], [6], [7], [8] y para problemas de máquina única [9], [10], [11], [12], [13] y [14]. Este trabajo presenta la línea de investigación a través de la cual se pretende comparar los trabajos realizados hasta el momento sobre *scheduling* dinámico con algoritmos evolutivos para problemas de máquina única, con otra metaheurística diferentes como es el caso de la Colonia de Hormigas.

1. INTRODUCCIÓN

Un proceso de *scheduling* implica seleccionar y secuenciar actividades tal que ellas cumplan uno o más objetivos y satisfagan un conjunto de restricciones del dominio del problema. Durante este proceso se deberá seleccionar entre *schedules* (planes o planificaciones) alternativos y asignar recursos y tiempos a cada actividad de manera tal que dichas asignaciones respeten las restricciones temporales de las actividades (*jobs*) y las capacidades limitadas de un conjunto de recursos compartidos, de manera que ciertas funciones objetivo (por ejemplo *tardiness, makespan*, etc.) sean minimizadas.

En general, dentro del ámbito de *scheduling*, los modelos más estudiados fueron los modelos conocidos como estáticos u *off-line*, es decir, donde las actividades, los recursos, los tiempos de procesamiento están predefinidos, no se modifican durante el proceso, y con un objetivo involucrando

[1] El Grupo de Investigación cuenta con el apoyo de la Universidad Nacional de La Patagonia Austral.

la minimización del tiempo de finalización y los costos de operación. Pero en los problemas del mundo real existen otra serie de decisiones que interactúan con el modelo clásico, como por ejemplo cambiar la cantidad y/o configuración interna de los recursos mientras el proceso de *scheduling* está en avance para balancear los cambios en la carga de los *jobs* que arriban al sistema; o puede aumentar o disminuir la cantidad de operarios en el sistema, en distintos momentos del día; o uno o más recursos pueden deshabilitarse temporalmente por razones de falla o mantenimiento. Si algunas de estas decisiones se adicionan a dicho modelo se obtienen los modelos de *scheduling* dinámicos u *on-line*.

Los modelos de este tipo implican una reconfiguración interna dinámica del proceso de *scheduling* para adaptarlo a la nueva situación del contexto. También pueden existir causas externas que necesiten una reconfiguración, por ejemplo cambio, por parte de los clientes, en las fechas de entrega en función de sus *stocks* y demandas [1]. Uno de los problemas más importantes de *scheduling* es el de máquina única que consiste en el secuenciamiento de un conjunto de tareas para ser procesadas por un único recurso. El estudio de este tipo de problemas es importante porque, a menudo, aparece embebido en un problema de *scheduling* más complejo [2]. Por ejemplo, en un entorno de planificación de tareas sobre múltiples máquinas existe una máquina crítica, cuello de botella, cuya capacidad de procesamiento es más lenta que el requerido por el resto. Un tratamiento de la máquina crítica como un problema de *scheduling* de máquina única puede mejorar notablemente la solución del problema más complejo

2. AMBIENTES DINÁMICOS

Siempre que ocurre un cambio en un problema de optimización dinámico, ya sea cuando cambia el objetivo de optimización, la instancia del problema, o algún cambio de restricciones, el óptimo probablemente también cambiará, por lo que seria necesaria una adaptación de la vieja solución.

Un enfoque estándar para tratar con esos dinamismos, es considerar a cada cambio como el arribo de un nuevo problema de optimización que tiene que ser resuelto desde el principio [15]. Sin embargo, resolviendo un problema desde el principio sin re-usar información del pasado, consume demasiado tiempo, debido a que un cambio puede no ser identificado directamente, o porque la solución del nuevo problema no debería diferir demasiado de la solución del viejo problema.

Considerando que no todos los ambientes dinámicos son equivalentes, y que dinámicas diferentes probablemente requieren diferentes enfoques de optimización, se pueden definir un número de criterios a través de los cuales los ambientes dinámicos puedan ser categorizados [25]. Basados en esta categorización, uno podría eventualmente caracterizar clases de ambientes dinámicos para el cual un algoritmo es mas conveniente que otro.

- *Frecuencia de cambio*: ¿Cada cuánto tiempo o con qué frecuencia cambia el ambiente? Algunas veces muchas comparaciones dependen de los detalles de implementación y hardware, y ya que usualmente el número de evaluaciones en los AEs es el factor de determinación de tiempo, el número promedio de evaluaciones entre los cambios debería ser en una medida apropiada. Solamente si existen otros aspectos relevantes para el tiempo de computación de la evaluación, debería necesitarse para las comparaciones entre enfoques, el tiempo real entre los cambios del entorno.
- *Severidad del cambio*: ¿El sistema tuvo sólo un cambio leve o generó una situación completamente nueva? Esto está determinado principalmente por la distancia genotípica desde el viejo al nuevo óptimo, pero pueden haber otros aspectos como por ejemplo, si el nuevo óptimo puede ser obtenido desde el viejo óptimo por un *hill climbing* simple, el tamaño del espacio de búsqueda, o la probabilidad que el nuevo óptimo se alcanzará desde el viejo óptimo por una mutación simple.
- *Predecibilidad del cambio*: ¿Se genera un patrón o tendencia en los cambios, o son sólo aleatorios? ¿Es posible predecir la dirección, tiempo o severidad del siguiente cambio considerando los cambios encontrados hasta ahora?
- *Duración del ciclo / Precisión del ciclo:* ¿El óptimo retorna a las situaciones anteriores o por lo menos se acerca a ellos? Y si es así, ¿Cuanto de cerca? Aquí uno debería medir el número promedio de estados del ambiente entre dos encuentros consecutivos del mismo estado (o uno muy similar). Si el nuevo estado no es exactamente el mismo, pero varía levemente, entonces la distancia adicional del nuevo estado a la solución previamente encontrada, es

importante. La duración y precisión del ciclo podría determinar si la memorización de las soluciones previas es una estrategia útil.

Parece difícil medir algunas de estas características, o más aún, pueden estar fuera del alcance las comparaciones entre diferentes problemas de optimización, pero al menos éstas características pueden ser variadas en un problema simple tal que puede ser examinada su influencia de calidad sobre un AE específico. Además de las propiedades del ambiente, puede ser interesante para el diseño de un AE, considerar los siguientes aspectos: ¿Las ocurrencias de un cambio son conocidas explícitamente por el sistema o tienen que ser detectadas?, ¿Es necesario cambiar la representación genética ya que ésta está afectada por un cambio?, la característica que cambia, ¿Es equivalente a uno en la función de optimización, la instancia del problema o algunas restricciones?, y a menudo, ¿Las soluciones producidas por el AE pueden influir en el ambiente?.

Una categorización alternativa ha sido propuesta recientemente en [16]. Aquí, los ambientes pueden ser clasificados como constante (el mismo cambio en cualquier período), estacionario (no cambia), periódico (retorna al estado anterior), homogéneos (el *landscape* completo moviéndose coherentemente, como opuesto a partes diferentes comportándose diferentemente), y alternativos (saltos óptimos desde un componente/máximo del *landscape* hacia otro).

Además de las posibles clasificaciones y/o alternativas de cambios en el entorno, existe una cuestión muy importante a tener en cuenta. Dicha cuestión es la manera en que se evalúan los algoritmos propuestos para un problema dinámico. En este sentido, las comparaciones no son tan directas como lo podrían ser en un entorno estático. Por lo tanto, es también es necesario analizar cuidadosamente estos aspectos al momento de hacer afirmaciones acerca de la calidad de los algoritmos propuestos. En el caso de problemas de *scheduling* dinámico, se pueden considerar diferentes aspectos que acercan a la formulación general los problemas reales como: cambio de las fechas límites de terminación de las tareas (*due dates*) o cambios en el tiempo de procesamiento o duración de una tarea. Las alternativas anteriores son interesantes, pero están más relacionadas a aspectos más teóricos del problema de dinamismo en general. Si bien se podrían considerar para su implementación, existen otras que son más específicas de los problemas de *scheduling*. Por ejemplo: de nuevas tareas durante el proceso de asignación de recursos (mayor es el número de tareas, mayor es la severidad del cambio), igual al anterior pero eliminando de tareas, rotura de máquinas, etc. Aunque estos tipos de cambios son más complejos de manejar, son más cercanos a situaciones encontradas en el mundo real.

3. METAHEURÍSTICAS PARA AMBIENTES DINÁMICOS

Considerando las características de los ambientes dinámicos, sería deseable contar con un algoritmo de optimización que sea capaz de ir adaptando continuamente la solución a los cambios del entorno, re-usando la información obtenida en el pasado. Los Algoritmos Evolutivos parecen ser los candidatos apropiados ya que tienen mucho en común con la evolución natural, y la adaptación en la naturaleza es un proceso continuo. La primera aplicación conocida de AEs a entornos dinámicos, fue en 1966 [20].

El problema principal con los Algoritmos Evolutivos estándar, es que ellos convergen eventualmente a un óptimo y pierden así su diversidad que es necesaria para explorar eficientemente el espacio de búsqueda. Así, una vez que la población del algoritmo evolutivo converge, ésta también pierde su habilidad para adaptarse a un cambio en el entorno cuando tal cambio ocurre. En consecuencia, se requiere de mecanismos adicionales que provean permanente diversidad en la población sin perturbar el proceso de búsqueda.

Para resolver de manera práctica instancias grandes, frecuentemente se usan métodos aproximados que retornan soluciones cercanas al óptimo en un tiempo relativamente corto. Los algoritmos de este tipo son llamados en términos generales *heurísticas*. Ellos usan a menudo un poco de conocimiento específico del problema para construir o mejorar soluciones.

Estas heurísticas pueden clasificarse dentro de las categorías de ***explotativas***, llamadas así porque inspeccionan el espacio de búsqueda en el vecindario de una solución posible [18]. En general consisten en obtener una solución factible de partida y luego, ir modificándola levemente, es decir, permaneciendo en el vecindario de dicha solución y evaluar cada nuevo *schedule* y cuando no exista mejora el método finaliza; ***exploративas***, que pueden ser pensadas que actúan, en determinado momento del proceso al no avanzar hacia nuevas soluciones, diciendo “es tiempo de un cambio” [21] lo cual se traduce en realizar un salto a otro punto del espacio de búsqueda (no del vecindario); y las ***exploративas/explotativas*** que son las tendencias

más modernas y que tratan de hallar heurísticas con estrategias de explotación/exploración mixtas. Dentro de ellas cabe mencionar *Tabu Search* [19], *Simulating Annealing* [22], Algoritmos Evolutivos [23] y Algoritmos de Colonia de Hormigas (ACO) [24].

Recientemente, muchos investigadores tienen enfocada su atención sobre una nueva clase de algoritmos, llamados metaheuristicas. Una ***metaheuristica*** es un conjunto de conceptos algorítmicos que se pueden usar para definir métodos heurísticos aplicables a un conjunto amplio de problemas de optimización en un tiempo razonable.

Una metaheuristica particularmente exitosa está inspirada para el comportamiento de las hormigas reales, conocida como la metaheurística ACO. Numerosos enfoques algorítmicos basados sobre las mismas fueron desarrollados y aplicados con éxito para una variedad de problemas de optimización [26], [27], [28], [29], [30], [31] y [32].

4. LÍNEAS DE INVESTIGACIÓN

Hasta el momento existen varias publicaciones realizadas por este grupo de investigación para diferentes problemas de *scheduling* dinámico en entornos de maquina única [11], [12], [13] y [14]. Todos estos problemas han sido desarrollados sin restricciones, por lo que se intentará incorporar a estos problemas, algunas restricciones que hasta el momento no han sido analizadas tales como *preemption* vs *nonpreempion*, inserción vs no inserción de tiempo ocioso, *set up* dependientes de la secuencia, dependencia entre jobs (*precedence constrainsts*) entre otros. Además de hacer un análisis profundo sobre los diferentes modelos dinámicos.

Actualmente la investigación está orientada al estudio de las diferentes heurísticas para ambientes dinámicos, desarrolladas hasta la fecha especialmente las evolutivas y de colonia de hormigas, orientado siempre al problema de máquina única dinámico. Se prevé el diseño e implementación de diferentes heurísticas alternativas, como así también de experimentos que permitan poder realizar la evaluación de las heurísticas desarrolladas y analizar los resultados obtenidos para cada una de ellas a los fines de poder realizar una comparación de resultados que nos indique cuál es la mejor heurística a utilizar para el modelo dinámico planteado (si es que sólo una existe).

5. BIBLIOGRAFÍA

[1] Morton T.E., Pentico D.W. – *Heuristic Scheduling Systems* – John Wiley & SoonsInc., New York, 1993.

[2] Baker, K. R. – *Introduction to Sequencing and Scheduling* – John Wiley & Sons Inc., New York, 1974.

[3] Shyh-Chang Lin and Erik D. Goodman and William F. Punch, *A Genetic Algorithm Approach to Dynamic Job-Shop Scheduling Problems*, Proceedings of the Seventh International Conference on Genetic Algorithms, 1997, Morgan-Kaufmann, Thomas Back (Ed.), pp. 481-489.

[4] C. Bierwirth, et. al – *Production Scheduling and Rescheduling with Genetic Algorithms* – Evolutionary Computation, 7, N° 1, págs. 1 - 17, 1999.

[5] Fang H. – *Genetic Algorithms in Timetabling and Scheduling* – Deparment of Artificial Intelligence, University of Edinburg, Scotland, 1994.

[6] Vazquez M. y Whitley , D – *A Comparision of Genetic Algorithms for the Dynamic Job Shop Scheduling Problem* – Parallel Problem Solving from Nature VI, 2000.

[7] Lin S., et. al – *A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems* – International Conference of Genetic Algorithms, 1997.

[8] Whitley, D. y Kauth, G. – *GENITOR: A Different Genetic Algorithm* – Proceedings Rocky Mountain Conference on Artificial Intelligence, 1988.

[9] Madureira A., Ramos C., do Carmo Silva, Silvio – *A Genetic Approach to Dynamic Scheduling for Total Weighted Tardiness Problem*, PLANSIG'99, 18th Workshop of the UK Planning and Scheduling Special Interest Group, Manchester, UK, 1999.

[10] Mao W., Kincaid R. y Rifkin, A. – *On-line Algorithms for a Single Machine Scheduling Problem*, en Impact of Emerging Technologies in Computer Science and Operation Research, págs. 157 – 173, 1995.

[11] Lasso, M., Pandolfi D., De San Pedro M., Villagra A., Vilanova G., Gallard, R. – *Algorithms to Solve the Dynamic Weighted Tardiness Problems*, VIII Congreso Argentino de Ciencias de la Computación, Bs. As., pág. 609 – 616, Octubre 2002.

[12] Lasso, M., Pandolfi D., De San Pedro M., Villagra A., Gallard, R. – *Heuristics to Solve Dynamic W-T Problems in Single Machine Environments*, Proceedings of the International Conference on Computer Science, Software Engineering Information Technology, e-Business and Applications, págs. 432 –437, Rio de Janeiro, June 2003, Brazil.

[33] De San Pedro M., Lasso, M., Villagra A., Pandolfi D., Gallard, R. – *Solutions to the Dynamic Average Tardiness Problem in Single Machine Environmets*, IX Congreso Argentino de Ciencias de la Computación, La Plata, págs. 729 – 739, Octubre 2003.

[14] Lasso, M., Pandolfi D., De San Pedro M., Villagra A., , Gallard, R. – *Solving Dynamic Tardiness Problems in Single Machine Environments*, IEEE Congress on Evolutionary Computation, Vol. I, págs. 1143 – 1149, Portland, USA, 2004.

[15] Raman N. and Talbot F.B. – *The job shop tardiness problem: a decomposition approach*. European Journal of Operational Research, 69:187-199, 1993.

[16] Weicker K. *An analysis od dynamic severity and population size*. In Schoenauer et al. [17].

[17] Schoenauer M., Deb K., Rudolph G. Yao X., Lutton E., Merelo J.J., and Schwefel H.P., editors. *Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of *LNCS*. Springer, 2000.

[18] Wilkerson, L. J. e Irwin, J. D . – *An Improved Algorithm for Scheuling Independent Tasks*, AOOE Transactions, 3, págs. 239 - 245, 1971.

[19] Van De Velde, S. L. - *Machine Scheduling and Lagrangian Relaxation*, PhD Thesis, Technische universiteit Eindhoven, 1991.

[20] Fogel D.B., Owens A.J., and Walsh M.J. *Artificial Intelligence through Simulated Evolution*. JoHn Wiley, 1996.

[21] Glover, F. y Laguna, M. – *Target Analysis to Improve a Tabu Search Method for Machine Scheduling*, Advanced Knowledge Research Group, US West Advanced Technologies, Boulder, CO., 1989.

[22] Kirkpatrick S.C. y Vecchi M.P. – *Optimization by Simulated Annealing*, Science 220, pp. 671 – 680, 1983.

[23] Michalewicz, Z. – *Genetic Algorithms + Data Structures = Evolutions Programs*, Second, Extended Editons, Springer-Verlag, 1994.

[24] Dorigo, M. y Stützle, T. – *Ant Colony Optimization*, The MIT Press, 2004.

[25] Branke J. – *Evolutionary Optimization in Dynamic Environments (Genetic Algorithms and Evolutionary Computation)*. Kluwer Academic Publishers (KAP), 2002.

[26] Colorni, A., Dorigo, M., Maniezzo V., Trubian, M. – *Ant System for Job Shop Scheduling*, JORBEL – Belgian Journal of Operations Research, Statistics and Computer Science, Vol. 34, N°19, págs. 39 – 53.

[27] Bauer, A., Bullheimer, B., Hartl R. F., Strauss, C. – *Minimazing Total Tardiness on a Single Machine using Ant Colony Optimization*, Central European Journal of Operations Research and Economics, Vol 8, N° 2, págs. 125-141, 2000.

[28] Blum C. – *ACO Applied to Group Shop Scheduling: A Case Study on Intensification and Diversification*, Proceedings of ANTS 2002, Lecture Notes in Computer Science Series, N° 2463, Springer-Verlag, 2002.

[29] Pfahringer, B. – *Multi-agent Search for Open Shop Scheduling. Adapting the Ant_Q Formalism*, Technical Report TR-96-09, Australian Research Institute for Artificial Intelligence, Vienna, 1996.

[30] Gagné C., Price, W. L., Gravel, M. – *Comparing an ACO Algorithm with other Metaheuristics for the Single Machine Scheduling Problem with Sequence-dependent setup times*, Journal of Operational Research Society, Vol. 53, págs. 895 – 906, 2002.

[31] Merkle, D., Middendorf, M. – *On Solving Permutation Scheduling Problems with Ant Colony Optimization*, Technical Report 415, Institute AIFB, University of Karlsruhe, 2002.

[32] Guntsch, M., Middendorf, M. – *Applying Population Based ACO to Dynamic Optimization Problems*, Proceedings of ANTS2002, Lecture Notes in Computer Science Series N° 2463, págs. 111 – 122, Springer-Verlag, 2002.

[33] F. Raman and M.T. Talbot, *The job shop tardiness problem: a decomposition approach*, European Journal of Operational Research, 69:187-199, 1993.