Resolución del problema RCPS mediante lógica neuro-difusa con aprendizaje evolutivo.*

Joaquín Bautista[1], Jordi Pereira[2], Marcela de la Rosa[3], Ramón Companys[4]
[1]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona, bautista@oe.upc.es
[2]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona,, pereira@oe.upc.es
[3]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona,, delarosa@oe.upc.es
[4]ETSEIB-DOE-UPC, Diagonal 647, Planta 7, 08028, Barcelona, companys@oe.upc.es

RESUMEN

La programación de actividades con limitación de recursos (RCPSP: resource-constrained project scheduling problem) es un problema combinatorio clásico que aparece en las líneas de manufactura y producción. El problema consiste en establecer el orden de lanzamiento de las actividades y la asignación de los recursos con una menor duración total. En problemas de tamaño real, los algoritmos de exploración de entornos y las heurísticas greedy basadas en reglas de prioridad son las más empleadas para la resolución. En el presente trabajo se muestra un combinador de reglas de prioridad mediante lógica difusa cuyos parámetros han sido entrenados mediante un procedimiento evolutivo conocido como evolución diferencial.
***Palabras clave**: RCPSP, meta heurística, lógica difusa, evolución diferencial, soft computing.*

1 Introducción.

La secuenciación de tareas o actividades con limitación de recursos (RCPSP: Resource-Constrained Project Scheduling Problem) es un problema clásico de optimización combinatoria que ha sido ampliamente tratado en la literatura [1], [2] y [3]. El problema consiste en determinar el programa compatible de menor duración de un conjunto de actividades con duraciones conocidas que presentan restricciones de tipo potencial (relativas a las precedencias entre tareas) y acumulativas (relativas al consumo de unos recursos finitos limitados por su disponibilidad).

Para la resolución de ejemplares de tamaño real se han empleado diversas heurísticas, entre las que destacan las de tipo greedy basadas en la construcción progresiva de una solución mediante el uso de reglas de prioridad que condicionan el lanzamiento de las tareas, ya sea en serie o en paralelo [4] y [5]. Las reglas se refieren o combinan aspectos tales como la duración, la holgura total, el número de actividades siguientes, la solicitud de recursos, etc. y sirven para establecer una ordenación de tareas en cada iteración, con el propósito de seleccionar la tarea más prometedora según el índice utilizado, entre un conjunto de candidatas compatibles con la parte de la solución ya construida; las tareas candidatas son aquéllas cuyas precedentes han sido programadas y su requerimiento de recursos no supera las disponibilidades de los mismos.

La aplicación de este tipo de algoritmos heurísticos suele ofrecer soluciones aceptables, en promedio, tanto más cuanto mayor sea el número de aspectos que combina la regla. No obstante, no puede concluirse que exista una única regla que supere a todas las demás ante cualquier ejemplar de problema. Por otra parte, salvo que se incorpore el azar al procedimiento, siempre ofrecerán las mismas soluciones.

* El presente estudio se ha realizado en el marco del proyecto de investigación TAP98-0494 financiado por la CICYT.

En el presente trabajo se propone un procedimiento de construcción progresiva (un combinador de reglas mediante lógica difusa) que incorpora los siguientes aspectos: 1) el conocimiento sobre el RCPSP que ofrecen las reglas de prioridad específicas del problema y 2) la posibilidad deseable en todo problema combinatorio de generar diversas soluciones en el espacio de búsqueda.

2 Combinación de heurísticas en los procedimientos de búsqueda.

Existen diversas técnicas para la combinación de reglas heurísticas, ya sea en procedimientos de construcción progresiva o en métodos de búsqueda local.

En [6], se expone la aplicación de un procedimiento de combinación de heurísticas en que esta selección se realiza de forma aleatoria en cada ocasión que se debe tomar una decisión. Los resultados que se obtienen son mejores que los de la elección de una única heurística durante todo el proceso. Este método incorpora también y de forma paralela la obtención de diferentes soluciones al añadir un factor de azar en el proceso.

En el caso de la búsqueda local [7], se han propuesto otras alternativas para la definición de vecindarios [8], [9], [10] y [11] basadas en la relación que existe entre una heurística h y la solución s que se obtiene al aplicarla a un ejemplar del problema p: h(p) = s. Otra posibilidad para definir vecindarios en el espacio de las heurísticas consiste en desarrollar nuevas versiones parametrizadas del conjunto de heurísticas disponibles y específicas de un problema o en separar las decisiones de secuenciación y asociarlas a una regla, permitiendo caracterizar una solución a través de un vector de reglas: r = (ρ[1], ρ [2], ..., ρ [N]) donde N es el número de tareas y [k] es la regla que se aplica en la k-ésima decisión de secuenciación.

3 Un procedimiento de construcción progresiva con un combinador de reglas

La construcción de la solución con un combinador de reglas mediante lógica difusa se basa en el algoritmo greedy de secuenciación en paralelo [5].Este procedimiento deja en el aire cómo escoger la regla. Esta elección normalmente corresponde a una regla fija, con lo que se obtiene el procedimiento de construcción progresiva estándar. Otra elección puede ser la elección de la regla al azar de un conjunto de reglas disponibles.

Al realizar una primera observación de las reglas heurísticas de la literatura, se observa que muchas de estas tratan de explotar diferentes conocimientos apriorísticos del problema. Por ejemplo, en el caso de disponer de muchos recursos, la intuición nos encaminaría a la elección de una actividad con un consumo elevado de ellos, mientras que con un número de candidatos bajo, intentaríamos escoger una actividad que ampliase el número de candidatas en la siguiente elección. Existen reglas que priorizan la elección de actividades con un alto consumo de recursos, y otras que priorizan la elección de tareas con muchas candidatas. No obstante, en caso de no explotar el conocimiento implícito en las reglas, pueden llevar a malas elecciones de secuenciación.

Aún así, el instante de elección de una regla se puede categorizar por diversos parámetros que afectan la elección de una actividad, tales como el número de candidatas existentes, la disponibilidad de recursos que se poseen o el margen de finalización de las tareas candidatas para no retrasar el conjunto del proyecto. La lógica difusa, introducida por Lotfi Zadeh [12] [13] en la Universidad de Berkeley, es un elemento válido para definir estas condiciones tal como las representa el lenguaje natural (muchos recursos, pocas candidatas) y a partir de él escoger una regla apropiada que asigne una actividad.

Esta técnica consiste en evaluar una serie de parámetros de entrada mediante condiciones difusas que determinen el estado del sistema, para su posterior categorización y, en función del subconjunto al que peretenezca, decidir qué medida debe de tomarse delante de la situación a la que se enfrenta. Así, en base a las condiciones de la solución en proceso de construcción, podemos categorizarla en un subconjunto donde unas heurísticas que exploten el conocimiento sobre características del problema que poseen sean las aplicadas.

En el caso implementado del problema, se han tenido en cuenta dos reglas de categorización, la disponibilidad de recursos y el número de candidatas existentes. El parámetro de veracidad de las premisas se enmarca respecto a los recursos totales disponibles y a la etapa del grafo con mayor amplitud. De la inferencia de ambas premisas obtenemos cuatro estados diferentes (pocos recursos y pocas candidatas, pocos recursos y muchas candidatas...) con un subconjunto de heurísticas asociadas a cada una de las condiciones.

La formalización del algoritmo de elección de la regla a utilizar es la siguiente:

Notación:

α: Variable de control de la disponibilidad de recursos
β: Variable de control del número de candidatas
$\alpha_{thresshold}$: Parámetro de corte de la disponibilidad de recursos
$\beta_{thresshold}$: Parámetro de corte del número de candidatas
R_t: unidades de recurso j disponibles en la t-ésima decisión.
Z_t: conjunto de tareas programables en la t-ésima decisión.
Anchura: (en este trabajo) Número máximo de vértices de una etapa en un grafo polietápico.

Algoritmo:

1.Determinar las variables de entrada

$$\alpha' = \sum_{1 \le j \le M} R_t(j) \qquad (1)$$

$$\beta' = cardinal(Z_t) \qquad (2)$$

2.Fuzzificación de las variables

$$\alpha = \frac{\alpha'}{\sum_{1 \le j \le M} R_0(j)} \qquad (3)$$

$$\beta = min\left[1, \frac{\beta'}{anchura(G)}\right] \qquad (4)$$

3.Inferencia de ambas premisas.

Si $\alpha<\alpha_t$ Y $\beta<\beta_t$
    escoger_grupo(1);
si $\alpha<\alpha_t$ Y $\beta>\beta_t$
    escoger_grupo(2);
si $\alpha>\alpha_t$ Y $\beta<\beta_t$
    escoger_grupo(3);
si $\alpha>\alpha_t$ Y $\beta>\beta_t$
    escoger_grupo(4);

4.Elección al azar una regla de la categoría: regla al azar del grupo correspondiente (anexo I).

4 Entrenamiento de los pesos mediante evolución diferencial

La evolución diferencial es una meta-heurística desarrollada en 1995 por los Rainer y Storn [14] de la unversidad de Berkeley que destaca por su sencillez de implementación y su rápida convergencia. Esta técnica ha sido utilizada con éxito en diversos campos y problemas diferentes como el diseño de filtros digitales, la resolución de problemas de programación lineal entera o el aprendizaje de redes neuronales.

Sus principales características son el mantenimiento de más de una solución, configuración de los pesos, durante el proceso de búsqueda, la representación de la solución mediante un vector de pesos reales asociando un valor real a cada elemento y la generación de nuevas soluciones partiendo de las soluciones actuales, mediante la ponderación de los pesos que forman la solución.

El algoritmo cuenta con tres parámetros de control, el número de generaciones, o iteraciones, que se permitirá realizar al algoritmo, el tamaño de la población que se mantendrá en paralelo y el peso que se utilizará para la generación de descendientes.

Cada vector solución presentará en este caso la información sobre el valor de $\alpha_{thresshold,}$ $\beta_{thresshold.\ y}$ la asignación de las regla a uno de los cuatro grupos definidos mediante la asignación de una frecuencia de elección de esta regla en cada una de las cuatro categorías. Para el procedimiento se han implementado un total de doce heurísticas, con lo que el número de parámetros que se pretende optimizar es de 50, dos parámetros de corte y 48 valores correspondientes a las doce reglas de decisión para cada uno de los cuatros grupos de tareas.

El valor de los parámetros seleccionados después de realizar diversas pruebas son:

- Tamaño de la población : 50.
- Peso de cruce: 0.3
- Max_gen: Número máximo de generaciones, 1000.

Estos parámetros fueron seleccionados midiendo el tamaño de la población y el peso de cruce que permitían la evolución de la población y posteriormente se marcó el número máximo de generaciones en que la desviación de la población era de valor reducido.

El algoritmo se describe a continuación:

Algoritmo II:

0. Inicialización: Generación de la población inicial mediante la generación de vectores de peso para cada pieza de mediante un valor aleatorio uniformemente distribuido entre 0 y 1. Num_generaciones:=0

Repetir mientras (num_generaciones< max_gen)

1. Seleccionar ascendentes y de parejas a cruzar: Escoger dos soluciones al azar.
2. Generación de descendientes: El vector solución "hijo" se compone como la suma-resta ponderada por el peso de cruce, entre los valores del vector padre y los del hijo. La figura 2 muestra este proceso. El vector u, adopta los valores de uno de los padres más la suma, resta ponderada de los dos valores que contienen ambos parámetros por el peso de cruze (0.3).
3. Evaluación: Se calcula el makespan de la suma de dieciseis proyectos escogidos al azar utilizando los pesos que conforman la solución para la selección de las reglas, mediante el algoritmo presentado en el apartado 3.
4. Regeneración de la población: Comparación del valor de la función de evaluación con el de otro elemento al azar de la solución y si el valor de la función objetivo es mejor sustituye a este elemento de la población.
5. Incrementar Num_generaciones;

Fin Repetir.

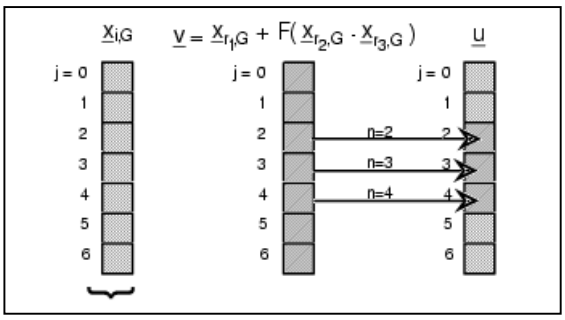

Figura 2: El nuevo vector se forma a partir de un vector padre y la suma de una resta ponderada entre los dos padres de determinados valores del vector.

5 Experiencia Computacional.

Para la experiencia computacional, se ha utilizado el juego PSPLIB disponible en Internet.

Este juego de datos contiene 2040 ejemplares divididos por el número de actividades (480 ejemplares de 30, 60 y 90 tareas y 600 ejemplares de 120 tareas). Para la experiencia computacional se han programado los siguientes algoritmos: 12 heurísticas greedy seleccionadas de la literatura que denominaremos heurísticas (H), el procedimiento de mezcla de heurísticas presentado por Cooper [6] y que denominaremos MS(multistart) y un combinador de reglas mediante lógica difusa que denominaremos Fuzzy (F). Para este último caso, a su vez, se han probado la configuración obtenida mediante aprendizaje por evolución diferencial denominado (FE), y las siguientes tres configuraciones de alpha y beta (0,4 , 0,4) , (0,3 , 0,3) y (0,2 , 0,2) (F4, F3 y F2 respectivamente) con las reglas asignadas a priori tal como se puede ver en el apéndice 1. Se han resuelto todas las instancias con los procedimientos descritos, con unos tiempos de ejecución inferiores a un segundo para los cien lanzamientos en todos los casos. En el anexo 1 pueden visualizarse las heurísticas utilizadas y los pesos asignados a cada uno de ellos.

En la tabla 1 se muestran las diversas configuraciones del fuzzy que obtienen mejores resultados. Adicionalmente la columna F muestra la suma de todos los procedimientos fuzzy.

Tareas	H	F4	F3	F2	MS	F
30	38	10	5	24	73	142
60	34	64	28	57	43	220
90	20	81	66	59	26	261
120	25	202	148	128	27	530

Tabla 1: Número de ocasiones en que cada procedimiento obtiene el mejor resultado de la experiencia computacional.

Finalmente en la tabla 2 se comparan el número de mejores resultados obtenidos por la ejecución de cada una de las cuatro configuraciones fuzzy en el número de óptimos obtenidos. Cabe citar que el procedimiento entrenado tiene una mayor dispersión en media entre los resultados obtenidos en cada lanzamiento. Esto se debe a la inclusión de todas las reglas en cualquier grupo, aunque aquellas con más probabilidad de aparecer acostumbran a ser las mismas reglas que las dictadas en el algoritmo cuyos pesos han sido escogidos a priori. Este hecho conlleva a que en ocasiones se seleccionen algunas reglas de asignación que intuitivamente podrían considerarse negativas. Esta mayor dispersión, por otra parte, ayuda a encontrar mejores soluciones ya que se diversifica la búsqueda y por ello se mantienen los pesos a todas las heurísticas, pero empeora el resultado global del método si tenemos en cuenta la media de resultados obtenidos en cada lanzamiento.

Tareas	F4	F3	F2	FE	Mejor FE	Desviación FE
30	402	402	402	453	275	1.57 %
60	357	357	346	453	249	2.6 %
90	325	323	325	465	260	2.46 %
120	318	154	121	257	53	3.81 %

Tabla 2: Número de ocasiones en que se obtiene el mejor resultado, incluyendo empates, y número de mejores resultados obtenidos por el método fuzzy entrenado respecto a los mejores resultados conocidos para los problemas.

Como puede verse en la tabla 2, el número de mejores soluciones obtenido por el procedimiento con aprendizaje es comparable, y en determinados casos mejor, al del procedimiento entrenado manualmente. Finalmente, también se muestra el número de mejores resultados obtenidos por este procedimiento sobre los mejores resultados conocidos para los problemas para cada número de tareas, y su desviación media respecto a este mejor resultado.

6 Conclusiones

En el presente trabajo se comparan diversos procedimientos heurísticos constructivos para la resolución del problema de secuenciación de actividades en proyectos. En contraposición al uso de una única regla durante todo el procedimiento, los métodos de combinación de reglas heurísticas obtienen mejores resultados sin un aumento de coste computacional, siendo los mejores procedimientos aquéllos que explotan mejor los conocimientos del problema.

Los procedimientos heurísticos constructivos, a pesar de no ser los mejores procedimientos para la resolución del problema hasta la optimalidad, son interesantes para la resolución de problemas de gran tamaño, como solución inicial para procedimientos de exploración de entornos y como cota superior en procedimientos exactos.

Referencias

[1] Blazewicz, J., Lenstra, J.K., Rinnoy Kan, A.H.G. (1983)"Scheduling projects to resource constraints: Classification and complexity". *Discrete Applied Mathematics*, 5, 11-24.

[2] Hartmann, S., Kolisck, R. (1998) "Experimental Evaluation of State of Art Heuristics for the Resource Constrained Project Scheduling Problem". *Wp. IBUK*, No. 476.

[3] Weglarz, J. Ed.. (1998) *Handbook on Recent Advances in Project Scheduling*. Kluwer, Amsterdam.

[4] Álvarez-Valdés, R., Tamarit, J.M. (1989)"Heuristic algorithms for a resource constrained project scheduling: A review and an empirical analysis", *Advances in Project Scheduling*, R. Slowinski, J. Weglarz (Ed.). Elsevier, Amsterdam. pp. 113-134.

[5] Kolisch, R. (1996) "Efficient priority rules for the resource-constrained project scheduling problem". *Journal of Operations Management, 14*, 179-192.

[6] Cooper, D. F. (1976) "Heuristics for scheduling resource-constrained Projects". *Management Sc*., 22, 1186-1197.

[7] Díaz, A., Glover, F. & Ghaziri, H. & González, J.M. & Laguna, M. & Moscato, P. & Tseng, F. (1996) *Optimización heurística y redes neuronales*. *Paraninfo*.

[8] Bautista, J. Lusa, A. Suárez, R. Mateo, M. Pastor, R. Corominas, A. (1999a)"Application of Genetic Algorithms to Assembly Sequence Planning with Limited Resources". *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning, ISATP'99* pp. 411-416

[9] Bautista, J. Suárez, R. Mateo, M. Lusa, A. (1999b) "Un algoritsme genètic en l'espai d'heurístiques per a cèl-lules de muntatge amb recursos limitats". *2n. Congrés Català d'Intel-ligència Artificial, CCIA'99, Butlletí de la ACIA, Núm. 18-19 (1999)* pp. 232-238

[10] Bautista J., Suárez R., Mateo M., Companys R.(2000), "Local Search Heuristics for the Assembly Line Balancing Problem with Incompatibilities Between Tasks", *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, San Francisco, CA, USA, April 24-28,* pp. 2404-2409.

[11] Storer, R.H., Wu, S.D., Vaccari, R. (1992) "New Search spaces for sequencing problems with application to Job Shop Scheduling". *Management Sc.*, 38-10, 1495-1509.

[12] Dubois,Didier, Prade, H. (1980) *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York.

[13] Kruse R., Gebhardt J., Klawonn F. (1994) *Foundations of Fuzzy Systems*. John Wiley and Sons Ltd., Chichester.

[14] Storn R. (1996), "On the Usage of Differential Evolution for Function Optimization", *Proceedings of the 1996 North American Fuzzy Information Processing Society (NAFIPS), Berkeley, CA, IEEE Press, 1996, pp. 519-523. IEEE Cat. No: 96TH8171.*

ANEXO I: Reglas heurísticas utilizadas

Número	Nombre	*Regla*
1	SIO *Shortest Imminent Operation*.	$v(i) = -P(i)$
2	GRD *Greatest Resource Demand*	$v(i) = P(i)\sum_{j=1}^{M} R(i,j)$
3	GRPW *Greatest Rank Positional Weight*.	$v(i) = P(i)\sum_{i \Rightarrow h} P(h)$
4	WRUP *Weighted Resource Utilization Ratio and Precedence*	$v(i) = w_p ns(i) + w_r \sum_{j=1}^{M} \frac{R(i,j)}{R_0(j)}$ $w = 1 - w_r; \quad w_r = (0.0, 0.1, ..., 0.9, 1.0)$
5	WRUP2	$v(i) = w_p \sum_{i \to h} P(h) + w_r \sum_{j=1}^{M} \frac{R(i,j)}{R_0(j)}$
6	MTS *Most Total Successors*	$v(i) = nst(i)$
7	WRUP3	$v(i) = w_p P(i) + w_r \sum_{j=1}^{M} \frac{R(i,j)}{R_0(j)}$
8	WRUP5	$v(i) = w_p nst(i) + w_r \sum_{j=1}^{M} \frac{R(i,j)}{R_0(j)}$
9	MIT *Most Immediate Successors*	$v(i) = ns(i)$
10	Batcharjee-Sahu	$v(i) = SIO(i) + d(i)$
11	GPW *Greatest Positional Weight*.	$v(i) = \sum_{i \Rightarrow h} P(h)$
12	Duración ponderada con consumo	$v(i) = SIO(i) + \sum_{j=1}^{M} d(i) * r(i,j)$

Peso de las asignación de las reglas heurísticas a cada grupo y método:

	Pesos fuzzy				Pesos fuzzy entrenados			
	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 1	Grupo 2	Grupo 3	Grupo 4
1	1	0	1	0	9	7	12	4
2	0	0	1	1	5	6	10	8
3	0	0	1	1	10	7	15	13
4	1	0	0	0	12	12	5	9
5	0	1	0	0	10	11	8	4
6	0	0	1	0	5	9	9	12
7	1	0	0	0	5	3	4	8
8	0	0	1	0	6	8	6	1
9	0	0	1	0	7	6	7	10
10	1	0	0	0	7	6	11	7
11	0	0	0	1	11	8	5	12
12	0	1	0	1	7	11	8	11