Víctor Parada Daza

Doctor en Ciencias de la Ingeniería de Sistemas y Computación
USACH

Héctor Pincheira Conejeros

Magíster en Ingeniería Informática
UTEM

Ricardo Corbinaud Pérez

Magíster en Ingeniería Informática
UTEM

MODELO CONSTRUCTIVO-EVOLUTIVO PARA EL RCPSP

RESUMEN

El RCPSP (ResourceConstrained Project SchedulingProblem) o problema de la planificación de actividades con restricciones de orden temporal y de recursos, constituye un modelo general de enorme trascendencia en el campo de la optimización combinatoria, que tiene por objetivo minimizar el tiempo máximo de duración total de un proyecto y que ha sido abordado con métodos tanto exactos como heurísticos. Aunque no garantizan la obtención de una solución óptima, los algoritmos heurísticos pueden entregar resultados satisfactorios en tiempos considerablemente inferiores a los demandados por las técnicas analíticas exactas. Como propuesta de solución al RCPSP, presentamos un modelo Constructivo-Evolutivo, resultante de la combinación de las capacidades de los modelos constructivos y la potencialidad de los algoritmos genéticos.

Palabras clave: **RCPSP, Scheduling, Métodos Heurísticos, Optimización Combinatoria.**

INTRODUCCIÓN

Los problemas de optimización combinatoria tienen como objetivo optimizar una determinada función, sujeta a un conjunto de restricciones. El conjunto de las soluciones posibles es, por lo general, extremadamente grande debido a una explosión combinatoria resultante de las muchas formas distintas de disponer los elementos básicos que conforman las soluciones.

Tal es el caso del problema de la planificación temporal de actividades bajo restricciones de precedencia y disponibilidad de recursos, más conocido como RCPSP (Resource Constrained Project Scheduling Problem), el cual ha motivado grandes discusiones de tipo matemático y mantiene un interés permanente por contribuir con propuestas de solución cada vez mejores.

El RCPSP consiste en programar de manera óptima un conjunto de n actividades, bajo relaciones de precedencia y limitación de recursos.

Algunas de las características de este problema son las siguientes:

• Las actividades no pueden ser interrumpidas una vez que han comenzado a ejecutarse.

• No existen plazos imperativos de término de las actividades. Es decir, no se pueden establecer a priori sus tiempos de finalización.

• Para cada actividad existe un único modo de operación. O sea, no es posible reducir su tiempo de ejecución asignando recursos adicionales.

• Es posible la ejecución simultánea de ciertas actividades.

La formalización del problema requiere definir los siguientes conjuntos:

$N = \{i / 0 \leq i \leq n+1\}$, actividades entre las cuales la 0 y la n+1 son ficticias.
$D = \{d_i / 0 \leq i \leq n+1 \wedge d_i \geq 0\}$, duraciones de las actividades.
$P_i = \{$predecesoras inmediatas de la actividad i$\}$
$R = \{R_k / 1 \leq k \leq m \wedge R_k \geq 0\}$, disponibilidades de cada uno de los m tipos de recursos.
$r_i = \{ r_{ik} / 1 \leq k \leq m\}$, unidades de cada recurso R_k requeridas por la actividad i.
$F = \{F_i / 1 \leq i \leq n\}$, tiempos de término de todas las tareas, clave para la minimización del tiempo total de ejecución del proyecto.
$A(t) = \{i \in N / F_i - d_i \leq t < F_i\}$, tareas activas en un instante t.

El problema así definido se puede expresar (Christofides, Alvarez-Valdez and Tamarit, 1987) como sigue:

Minimizar F_{n+1}, sujeto a (1)
$F_h \leq F_i - d_i \quad i = 1, ...,n+1; h \in P_i$ (2)
$\sum r_{ik} \leq R_k \quad i \in A(t) \wedge R_k \in R$ (3)
$F_j \geq 0$ (4)

La función objetivo (1) consiste en minimizar el tiempo de finalización de la última actividad del proyecto. Es decir, el tiempo total del proyecto (makespan). La restricción (2) establece las relaciones de precedencia entre las actividades. La restricción (3) valida que, en cada instante t, la cantidad de recursos utilizados por las actividades en ejecución no supere la disponibilidad máxima. Finalmente, la restricción (4) define las variables de decisión.

De la preocupación por resolver más óptimamente el RCPSP, han surgido dos categorías de métodos: los exactos y los heurísticos. Los métodos exactos intentan encontrar una solución óptima y demostrar que esa solución es la óptima global. Pero, el tiempo invertido por un método exacto para encontrar la solución óptima de un problema difícil, si es que existe tal método, es de un orden de gran magnitud, pudiendo, incluso, llegar a ser inaceptable.

Para resolver de manera exacta el RCPSP, se han diseñado diversos algoritmos basados en técnicas tales como: Branch and Bound, Relajación Lagrangiana y Programación Dinámica (Gorenstein, 1972; Fisher, 1973; Stinson, Davis and Kjumawala, 1978; Talbot and Patterson, 1978; Patterson, 1984; Christofides, Alvarez-Valdez and Tamarit, 1987; Demeulemeester, 1992; Mingozzi, Maniezzo, Ricciardelli and Bianco, 1995; Simpson and Patterson, 1996; Brucker and Knust, 2000; Dorndorf, Peschand Phan-Huy, 2000; Erenguc, Ahnand Conway, 2001, entre otros).

La naturaleza combinatoria del RCPSP conlleva su inherente intratabilidad y consecuente pertenencia a la clase NP-duro (Blazewicz, Lenstra and Rinnooy Kan, 1983). De otra manera, la inexistencia de limitación de recursos reduciría este problema a un caso de CPM (Critical Path Method) el cual puede resolverse en tiempo polinomial (Elmaghraby, 1977).

Revisando sólo una parte del dominio, los métodos heurísticos proporcionan buenas soluciones en breve tiempo computacional, aunque no necesariamente las óptimas (Osman and Kelly, 1996). Estos métodos proveen un marco general para la creación de algoritmos híbridos basados en la combinación de diferentes conceptos derivados de la inteligencia artificial, la evolución biológica, la comunicación química entre animales y la formulación estadística.

Así, es posible advertir un cúmulo de recientes contribuciones a la solución del RCPSP, entre las cuales se destacan los métodos heurísticos basados en reglas de prioridad (Xu, McKee, Nozickand Ufomata, 2008). Una regla de prioridad define un procedimiento para seleccionar una entre un conjunto de actividades elegibles (Ballestín, 2002). Estas reglas se han probado con algoritmos genéticos, tabu search, simulated annealing, colonias de hormigas, búsqueda local y parallel SGS (Schedule Generation Scheme) (Hartmannand Kolisch, 2006).

Buenos resultados han entregado los algoritmos de optimización por colonias de hormigas (ACO), combinando dos métodos de evaluación de feromona (Merkle, Middendorf and Schmeck, 2002) o integrándolos a una metaheurística híbrida denominada ANGEL (Ant colony optimization, Genetic algorithm and Local search strategy) (Tsengand Chen, 2006). ACO provee la población inicial para un algoritmo genético que, actualizando el feromona, obtiene una solución mejorada; luego, ambos algoritmos intervienen alternada y cooperativamente para generar una solución a partir de la cual una búsqueda local eficiente produce una solución final de mejor calidad.

La justificación técnica generalmente utilizada para encauzar procesos de búsqueda en conjuntos de secuencias activas y para reducir la duración de secuencias al interior de grafos (Ballestín, 2002), ha sido incorporada en 22 diferentes algoritmos (Valls, Ballestín and Quintanilla, 2005), 15 de los cuales usan justificación doble, reportando importantes mejoras en los resultados sin aumentar los tiempos de cómputo. Por otra parte, el HGA (Hybrid Genetic Algorithm) (Valls, Ballestín and Quintanilla, 2008), usa un operador de cruzamiento exclusivo para el RCPSP y un operador de justificación doble destinado a la mejora local de todas las soluciones generadas.

No menos importante ha sido el denominado algoritmo genético self-adapting, para el cual tanto la solución del problema como el algoritmo mismo son objetos del proceso de optimización genética (Hartmann, 2002). Meritorio también es el aporte del método de optimización basado en la permutación de enjambres de partículas para el RCPSP (Zhang, Li and Tam, 2006) y de las variaciones denominadas estrategia del límite inferior y estrategia de la duración mínima, ambas destinadas a mejorar la eficiencia de los algoritmos tabu search y simulated annealing (Moreno, Díaz, Pena and Rivera, 2007).

En las publicaciones revisadas se advierte una marcada tendencia al uso de métodos híbridos para resolver eficientemente el RCPSP. Y como una de las principales dificultades inherentes al problema del scheduling radica en factibilizar el espacio de búsqueda, parece razonable optar por navegar siempre a través de espacios factibles constructivos.

Por lo tanto, en este trabajo nosotros experimentamos con una hibridización entre las estrategias constructivas y los algoritmos evolutivos. Para ello, se formaliza el modelo híbrido constructivo-evolutivo propuesto (sección 2), se describen aspectos de representación necesarios para su implementación (sección 3) y se comprueba la efectividad del modelo mediante el análisis de los resultados de una experiencia computacional (sección 4).

MODELOHÍBRIDO CONSTRUCTIVO-EVOLUTIVO

El modelo propuesto constituye un híbrido que combina las características constructivas de los grafos and/or y los conceptos evolutivos de los algoritmos genéticos (Pearl, 1984; Goldberg 1989).

• Métodos constructivos

Los métodos constructivos generan, de manera incremental, la solución de un problema, enumerando implícitamente el dominio de las soluciones factibles debido a lo cual, generalmente, requieren de un gran esfuerzo computacional, tanto en espacio como en tiempo. Un método constructivo opera realizando una búsqueda de la mejor solución sobre un grafo que representa constructivamente el conjunto de soluciones factibles de cierto problema. En cada paso, el método selecciona el mejor de los nodos mediante una función de evaluación f(n) aplicada sobre cada nodo n (Pearl, 1984). Al nodo seleccionado se le aplica alguna regla (preestablecida y adecuada a la representación del problema) de generación de nodos descendientes. Así, nuevamente se está en condiciones de evaluar y continuar repitiendo el proceso hasta encontrar la solución del problema, que no es otra que un camino en el grafo construido, el cual se caracteriza por ser acíclico y dirigido. Cada nodo de este grafo representa un estado y pertenece a un camino que constituye una potencial solución del problema. La función de evaluación está compuesta por las funciones g(n) y h(n) que corresponden, la primera, al costo de generación del nodo, y la segunda, a una estimación del costo futuro en el cual se ha de incurrir para alcanzar un estado meta a partir del nodo inicial (Gómes and Parada, 1996).

• Algoritmos evolutivos

La teoría de la evolución de Darwin, el seleccionismo de Weismann y la genética de Mendel, confluyen en un conjunto de argumentos que actualmente configuran el paradigma del neodarwinismo, para el cual la vida en el planeta es el resultado de tres procesos que operan sobre las especies, a saber: la selección, la reproducción y la mutación (Kolisch and Hartmann, 1999).

La ciencia de la computación concibió la evolución como un proceso de optimización que podía simularse y dar apoyo a la solución de problemas de ingeniería en los cuales se debe optimizar el uso de recursos escasos. La idea fundamental de este proceso consiste en evolucionar una población de posibles soluciones para un problema dado, usando operadores inspirados en la variación genética y en la selección natural. La evolución es, en efecto, un método de búsqueda entre una inmensa cantidad de «soluciones» posibles. En el ámbito biológico, el enorme conjunto de posibilidades está dado por una colección de potenciales secuencias genéticas, y las soluciones deseables son los organismos altamente aptos. Es decir, los organismos mejor dotados para sobrevivir y reproducirse en su ambiente.

Actualmente existen tres vertientes derivadas de los principios del neodarwinismo: las estrategias evolutivas, la programación evolutiva y los algoritmos genéticos. Conjuntamente, a estas técnicas se les denomina computación evolutiva.

Un individuo es una potencial solución a un determinado problema. Un conjunto de individuos se denomina población. La aptitud individual se evalúa mediante una función que indica lo adecuado que es el individuo (es decir, la solución al problema) con respecto a los demás. Usualmente, la función aptitud es igual a la función objetivo del problema.

• Bases para el modelo híbrido

Un grafo and/or es un grafo en el cual cada par de nodos se conecta mediante un arco and, o bien mediante un arco or. Consecuentemente con el método de descomposición de un problema, un grafo and/or puede utilizarse como estrategia de solución de la siguiente manera: el nodo de inicio corresponde a la especificación del problema original, en tanto los nodos restantes representan los respectivos subproblemas (Pearl, 1984). Los arcos and establecen una relación de descomposición, los arcos or una relación de descomposición opcional, como se puede observar en la Figura.1. Para resolver un problema descompuesto de esta forma, las soluciones de los subproblemas se van agregando mediante reglas de descomposición inversa.

Una solución al RCPSP es un camino, en el correspondiente grafo and/or, entre el nodo inicial y el nodo final del proyecto. Una importante ventaja de esta representación radica en la independencia de las actividades potencialmente ejecutables en paralelo.

Los métodos constructivos actúan en el dominio de los caminos a las soluciones óptimas, es decir, comienzan con un estado elemental inicial y generan estados parciales que constituyen un camino que puede, o no, conducir a un estado meta con carácter de solución óptima. En cambio, los métodos evolutivos exploran el dominio de las soluciones manteniendo, en todo momento, una o varias soluciones finales, y para encontrar respuestas de mejor calidad, aplican operadores que modifican las soluciones actuales para obtener otras nuevas.

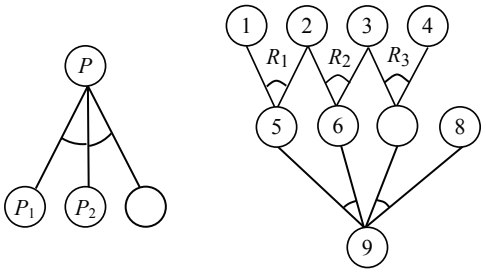


FIGURA 1. DOS EJEMPLOS DE GRAFO AND/OR

En el contexto del RCPSP, una población de individuos coexiste en un determinado entorno con recursos limitados. La competición por los recursos provoca la selección de aquellos individuos mejor adaptados a ese entorno, los cuales se combinan para lograr el cruzamiento de nuevos individuos y la mutación de ciertas características de sus progenitores. Los nuevos individuos pasan a competir por su supervivencia y, con el paso del tiempo, esta selección natural provoca el incremento en la "calidad" de los individuos de la población.

• Algoritmo constructivo-evolutivo

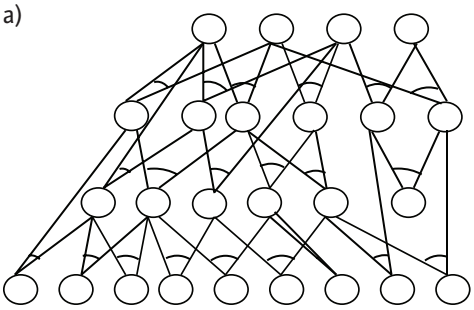
Para la construcción de caminos a las soluciones, se contempla el uso de cuatro conjuntos (poblaciones). El primero incluye la configuración inicial de una instancia del problema; el segundo contiene todos los estados parciales derivados de una selección de elementos; el tercero incluye una selección de estados parciales desde los derivados de una selección de elementos; el cuarto, de carácter elitista, contiene aquellos estados parciales correspondientes a las mejores construcciones logradas, las cuales son candidatas a formar parte de una solución.

Existe un ciclo generador de soluciones, que se repite hasta cumplir algún criterio de término, en el cual se seleccionan y combinan estados pertenecientes a los cuatro conjuntos. La combinación se realiza mediante operadores constructivos de cruzamiento y mutación.

La estrategia de utilizar conjuntos de construcciones parciales y operadores evolutivos pretende recorrer sólo parte del espacio de búsqueda que recorrería un método constructivo puro, expandiendo exclusivamente las construcciones candidatas a formar parte de una solución de buena calidad.

La Figura 2a) muestra un espacio de búsqueda generado por un método constructivo puro; en cambio, la Figura 2b) muestra el dominio generado por el método propuesto. En virtud de los antecedentes expuestos, el modelo híbrido constructivo-evolutivo se materializa en el algoritmo presentado.

a)



b)

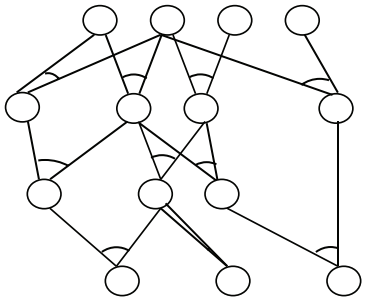


FIGURA 2. ESPACIO DE BÚSQUEDA: A) MÉTODO EXHAUSTIVO Y B) MÉTODO PROPUESTO

Algoritmo constructivo-evolutivo

Inicio
E← conjunto de todos los estados elementales
A(0) ←selección probabilística de elementos de E según f(ei)
B(0) ← Φ
t = 0

Repetir
t = t+1
C ← Φ
[1] A'(t)← selección desde A(t) según $f(a_i) = g(a_i) + h(a_i)$ $\forall_{ai} \in A(t)$
[2] C ← cruzamiento y mutación entre A'(t) y E
[3] C ← cruzamiento entre A'(t) y A(t)
[4] C ← cruzamiento y mutación entre A'(t) y B(t)
[5] Evaluar $f(ci) = g(ci) + h(ci)$ $\forall c_i \in C$
[6] B(t+1) ← actualización mediante elección determinística de individuos de C
[7] A(t+1) ← selección probabilística de mejores individuos de C según $f(c_i)$
Hasta cumplir criterio de término

Fin

Las poblaciones utilizadas por el algoritmo Constructivo-Evolutivo en cada etapa, cumplen las funciones que a continuación se detallan:
E: Conjunto de estados elementales de cualquier solución del problema.
A(t): Conjunto de construcciones parciales.
A'(t): Conjunto de los mejores individuos seleccionados desde A(t).
B(t): Conjunto de las mejores construcciones sobrevivientes hasta la t-ésima iteración.
C: Conjunto de las construcciones parciales resultantes de reproducir y mutar elementos de A'(t) con E, A(t) y B(t).
A(t+1): Conjunto de los mejores individuos seleccionados probabilísticamente desde C.
B(t+1): Conjunto de elementos elegidos determinísticamente desde C.

La cardinalidad de los conjuntos A y B puede variar mientras se ejecuta el algoritmo, pero se debe preestablecer un tamaño máximo dependiente de la instancia del problema a resolver. La cardinalidad del conjunto C puede determinarse a partir del número de sucesores resultantes del proceso de combinación. Al finalizar la ejecución, las soluciones deben encontrarse en el conjunto B. Alternativamente, B puede mantener los mejores candidatos a ser progenitores de nuevas construcciones. El criterio de selección aplicado para actualizar B debe considerar las características propias de la instancia del problema, pues un buen candidato a progenitor en un instante t no necesariamente es bueno en el instante t+1. Luego, podría ocurrir que, al cumplirse el criterio de término, B no contenga las soluciones. Por eso, se recomienda usar una variable independiente que almacene, en todo momento, la mejor solución obtenida.

El progreso de la construcción de elementos del conjunto A(t+1) depende principalmente de los criterios de selección definidos. Es decir, si éstos son adecuados, se evita que el algoritmo se entrampe en óptimos locales de mala calidad. Cuando el conjunto B no se ocupa, es decir cuando $B(t+1) \leftarrow \Phi \; \forall t$, el modelo se comporta como un algoritmo genético, salvo que las poblaciones no sean de soluciones finales, sino de estados parciales.

Con el avance del proceso van surgiendo construcciones potencialmente válidas como soluciones parciales al problema, las cuales se evalúan mediante la función $f(x) = g(x) + h(x)$.

• Convergencia para el algoritmo constructivo-evolutivo

Dado que el algoritmo contempla reglas de transformación de carácter constructivo, la evolución de los elementos del conjunto A(t+1) es de naturaleza incremental, es decir, en cada iteración sus elementos se van aproximando a una solución factible.

Si dos estados parciales generan uno o más estados parciales nuevos, entonces el operador de cruzamiento será de carácter constructivo. Luego, los descendientes son estados más evolucionados que sus progenitores, es decir, más cercanos a una solución. De igual manera, la modificación producida por mutación es de carácter constructivo, ya que implica la inserción de un nuevo elemento en un estado parcial.

Las instancias de selección definidas, sin importar cuales sean, actúan sobre estados parciales, por lo cual sólo influyen en la velocidad de convergencia y no afectan la tendencia evolutiva de esos estados a estados solución.

Los operadores de selección, cruzamiento y mutación permiten asegurar que la solución al problema estudiado existe y puede representarse como un grafo finito de estados, de manera que el algoritmo converge a una solución en un número finito de iteraciones (Pearl 1984).

ADAPTACIÓN DEL RCPSP AL MODELO CONSTRUCTIVO-EVOLUTIVO

Aspectos atingentes a la implementación hacen necesario adaptar el modelo general del RCPSP a las características de definición del algoritmo constructivo-evolutivo.

• Representación de una construcción parcial

Según el modelo general del RCPSP, cada tarea se puede concebir como una estructura (m+1)-dimensional, donde m indica el número de tipos de recursos requeridos, y la dimensión restante, la duración de la actividad. Una construcción parcial corresponde a la planificación de algunas actividades del proyecto. La Figura 3 muestra la representación tridimensional de una construcción parcial con cinco tareas y dos tipos de recursos. Pero, aunque la duración de cada tarea y los recursos que necesitan sean datos fijos, la representación gráfica de una

construcción parcial se complica cuando la cantidad de tipos de recursos es mayor que dos.

No obstante, si se respetan las restricciones de precedencia y de recursos, es posible representar una planificación en tan sólo dos dimensiones, de modo que se observe la relevancia del tiempo de inicio de cada actividad. Por lo tanto, una construcción parcial será válida si se muestran los tiempos de inicio de las tareas en una representación gráfica bidimensional que, implícitamente, respete las restricciones de precedencia y de recursos.

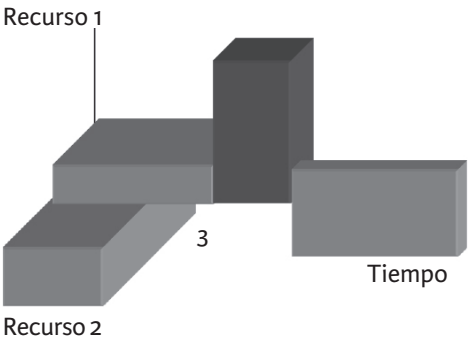


FIGURA 3. CONSTRUCCIÓN PARCIAL CON 5 TAREAS Y 2 TIPOS DE RECURSOS.

• Política de construcción

La elección de una tarea a insertar en una construcción parcial exige comprobar la previa planificación de aquellas que la preceden. Esto implica establecer un tiempo de inicio menor, es decir, lo más pronto posible.A su vez, el tiempo de inicio menor se determina a partir del tiempo de inicio mayor, o sea, el mayor tiempo de término de todas las tareas precedentes. Luego, se puede calcular el tiempo de término menor, como el tiempo de inicio menor más la duración de la tarea.

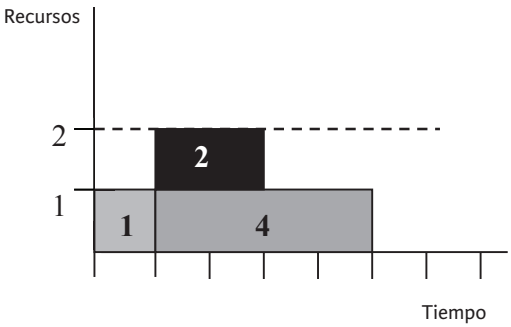
Seguidamente, se deben verificar las restricciones de recursos. Si los recursos son insuficientes, el tiempo de inicio menor de la tarea seleccionada se aumenta progresivamente hasta que existan recursos disponibles. Así, al insertar una tarea en una construcción parcial, se tiene la certeza de cumplir con el menor tiempo de inicio que permita la restricción de recursos.

La Tabla 1 muestra un ejemplo de proyecto simple, de cinco tareas, con sus duraciones, necesidades de recursos (de tan sólo dos unidades disponibles) y relaciones de precedencia. Suponemos, además, una construcción parcial como la descrita en la Figura 4. Si, para continuar la construcción, se elige la tarea 3, su tiempo de inicio mayor debiera ser el término de la tarea 2 (restricción de precedencia), como se aprecia en la Figura 4. Sin embargo, por no haber en ese instante disponibilidad del recurso, la tarea 3 debe posponerse según su tiempo de inicio menor, como se aprecia en la Figura 4.

TABLA 1. DEFINICIÓN DE TAREAS DE UN PROYECTORECURSOS.

Tarea	Duración	Recurso	Preced.
1	1	1	--
2	2	1	1
3	2	2	2
4	4	1	1
5	1	1	3,4

a)



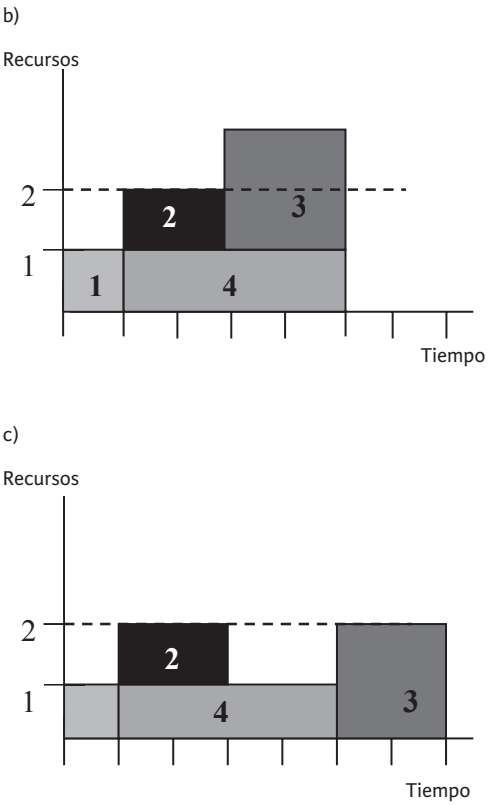


FIGURA 4. POLÍTICA DE CONSTRUCCIÓN

• Función de evaluación

El método tiene cierta similitud con el algoritmo A*, pues, a medida que el proceso de planificación avanza, van surgiendo construcciones virtualmente válidas como soluciones parciales al problema, las cuales se evalúan de acuerdo con fl(x)=g(x)+h(x), donde la función g(x) evalúa la calidad de la construcción parcial con base en su tiempo total de duración, y la función h(x) representa el tiempo estimado de duración de las actividades no consideradas aún en la construcción parcial.

Para estimar la duración de las actividades no consideradas aún en la construcción parcial, se asume la peor planificación de las tareas faltantes, o sea la planificación secuencial, razón por la cual h(x) corresponde a la suma de las duraciones de las tareas no programadas. Tal estimación evita que una construcción con menos tareas programadas sea mejor evaluada que una construcción con más tareas programadas. No obstante, cuando se presenta el peor caso, esto es cuando las restricciones de recursos provocan una solución secuencial, puede ocurrir que una construcción parcial sea mejor evaluada que una construcción total. Este inconveniente se supera aplicando una leve ponderación a la función h(x).

• Operadores Constructivo-Evolutivos

El operador de selección, que utiliza la función f(x) para evaluar los individuos, puede ser de naturaleza tanto elitista como probabilística. En cada iteración, la selección elitista (asociada al conjunto B) escoge los mejores individuos, y la selección probabilística (asociada al conjunto A) intenta aumentar la probabilidad de supervivencia de los mejores individuos.

La denominada selección por torneo binario hace competir individuos aleatoriamente emparejados y elige a los ganadores para la siguiente iteración. Así, se garantizan poblaciones heterogéneas y se excluyen mínimos locales. La probabilidad de selección de un individuo es $P(x) = (1 + s)/2$, donde $0 \leq s \leq 1$ representa el valor normalizado de ese individuo. Por lo tanto, aquel individuo más apto tiene una mayor probabilidad de ser seleccionado.

El operador de cruzamiento combina pares de individuos, aleatoriamente elegidos, para generar otros más evolucionados, de manera que las tareas de una construcción se insertan en otra que tenga programadas las que le preceden. En la Figura 5 se aprecia la generación de dos construcciones descendientes a partir de un par de construcciones progenitoras.

El uso de una probabilidad de cruzamiento reduce la combinación de individuos de los conjuntos A consigo mismo, A con B y A con E (compuesto de tareas individuales).

El operador de mutación consiste en insertar una tarea en una construcción parcial que tenga programadas todas las que le anteceden, tal como lo hace el operador de cruzamiento que combina los conjuntos A y E. La existencia de una probabilidad de mutación, distinta a la probabilidad de cruzamiento, establece la diferencia entre ambos operadores.

• Parámetros y criterio de término

El algoritmo necesita tres parámetros. El primero corresponde a la probabilidad de cruzamiento; el segundo, a la probabilidad de mutación; y el tercero, al multiplicador de población. El producto entre el número de tareas y el multiplicador de población establece la cardinalidad de los conjuntos A y B, y, consecuentemente, del conjunto C que contiene las construcciones resultantes de la aplicación de los operadores definidos.

Como elemento del criterio de término, el algoritmo utiliza un contador de iteraciones. Así, cada vez que se encuentra una solución mejor que la anterior, el contador toma el valor cero; en cambio, si la calidad de la solución no mejora después de cierto número de iteraciones, el algoritmo se detiene.

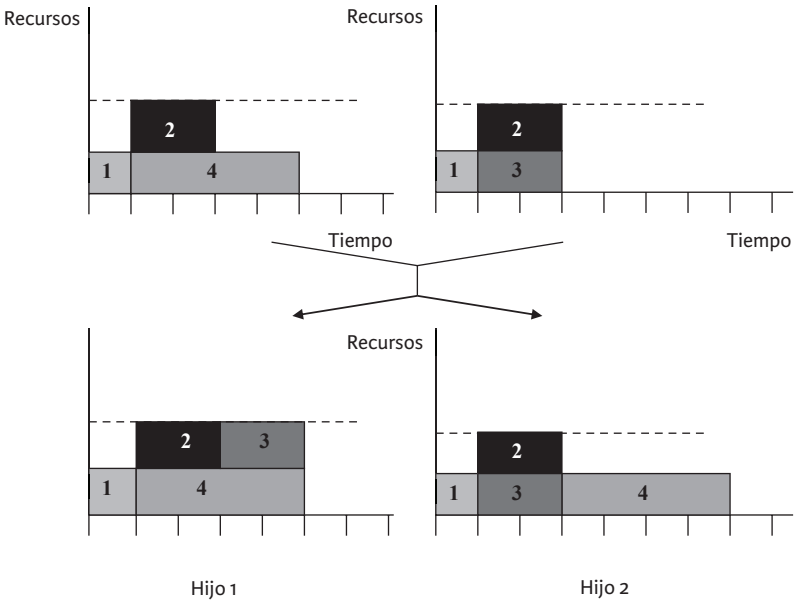


FIGURA 5. ESQUEMA DE CRUZAMIENTO

RESULTADOS

Las pruebas del algoritmo constructivo-evolutivo (ACE) se realizaron en un computador con CPU AMD Athlon X2 Dual-Core de 3.5GHz y RAM de 4Gb. Para el experimento se utilizaron 150 instancias del problema obtenidas desde Kolisch R. and A. Sprecher: 50 de 30 tareas, 50 de 60 tareas y 50 de 120 tareas. Dado que el algoritmo utiliza una componente aleatoria, la experiencia se realizó diez veces para cada instancia.

• Fase preliminar

En la primera fase de pruebas, para las instancias de 30 tareas, en el 29% de los casos se logró una solución óptima. En el 71% de los casos restantes el error promedio no excedió el 4.7%. Para las instancias de 60 tareas, se encontró una solución óptima en el 21% de los casos; en el 79% de los casos restantes el error promedio

no excedió el 8,9%. Para las instancias de 120 tareas, se obtuvo la solución óptima en el 14% de los casos. En el 86% de los casos restantes el error promedio no excedió el 19,9%

De lo anterior, se advierte que al incrementar la cantidad de tareas(tamaño del problema), aumenta el espacio de búsqueda factible, haciéndose cada vez más difícil encontrar un camino conducente al valor óptimo, lo que se traduce en el aumento de la diferencia entre los resultados obtenidos y los valores de referencia. No obstante, el error promedio producido evidencia que los resultados continúan siendo de buena calidad.

Otro factor que incide notablemente en los resultados es el conjunto de parámetros utilizados. Intuitivamente, es posible observar que un conjunto adecuado de parámetros es dependiente de cada tipo de instancias del problema. Sin embargo, en esta fase se utilizó el mismo conjunto de parámetros para las instancias de 30, 60 y 120 tareas, con la finalidad de facilitar la posterior labor de calibración derivada de un comportamiento preliminar del algoritmo.

• Calibración de parámetros

Dado que para cada tipo de instancia existe un conjunto singular de parámetros, en esta fase del experimento se intenta descubrir aquel conjunto más beneficiosamente representativo de todos.

Se experimentó con tres conjuntos de parámetros: el primero contempló un 98% para la probabilidad de cruzamiento, un 2% para la probabilidad de mutación y 2 como multiplicador de población (PR=98, PM=2, MP=2); el segundo conjunto considera un 85% para la probabilidad de cruzamiento, un 6% para la probabilidad de mutación y 4 como multiplicador de población (PR=85, PM=6, MP=4); y el tercer conjunto utiliza un 60% como probabilidad de cruzamiento, un 8% como probabilidad de mutación y 6 como multiplicador de población (PR=60, PM=8, MP=6).

Con la elección de estos conjuntos de parámetros se pretende contribuir al aumento de la diversidad de la población, es decir, que en una misma iteración se generen individuos que presenten diferentes niveles de desarrollo. En virtud del carácter constructivo del algoritmo, el nivel de desarrollo de un individuo lo establece el número de tareas programadas logrado en una iteración.

En esta segunda fase de pruebas, los mejores resultados se obtuvieron, para las instancias de 30 y de 60 tareas, con el segundo conjunto de parámetros; y para las instancias de 120 tareas, con el tercer conjunto de parámetros.

Complementariamente, sin perjuicio de que la generación de una mayor diversidad de soluciones aumente la probabilidad de encontrar una mejor entre ellas, en ocasiones puede llegar a ser contraproducente, tanto por los casos en que crece la desviación estándar como por el aumento del esfuerzo computacional derivado del incremento de la población.

• Convergencia

El aspecto constructivo del algoritmo asegura la obtención de una solución factible en un número finito de iteraciones directamente dependientes del conjunto de parámetros utilizados.

En primer lugar, se consideran las curvas de convergencia que resultan de la variación de la probabilidad de cruzamiento y de la mantención de las constantes probabilidad de mutación y multiplicador de población. Se observa que la solución se obtiene con una mayor cantidad de iteraciones cuando disminuye la probabilidad de cruzamiento, es decir, cuando disminuye la

cantidad de individuos que evolucionan en una iteración y, consecuentemente, incrementa la diversidad de la población.

Seguidamente se consideran las curvas de convergencia que resultan de la variación de la probabilidad de mutación y de la mantención de las constantes de probabilidad de cruzamiento y multiplicador de población. De manera similar, se aprecia que con el crecimiento de la probabilidad de mutación aumenta el número de individuos que evolucionan en una iteración y, por lo tanto, mejora la calidad de la solución.

Por último, se consideran las curvas de convergencia resultantes del aumento del multiplicador de población y de la mantención de las constantes de cruzamiento y de probabilidad de mutación. Con un comportamiento no muy distinto a los anteriores, se advierte un mejoramiento de los resultados en la medida en que se utiliza un multiplicador de población mayor. Sin embargo, se le debe prestar atención a este parámetro, pues un elevado multiplicador de población, para un reducido número de tareas, podría causar una alta redundancia de individuos en la población.

CONCLUSIONES

La trascendencia del problema de la planificación de actividades con restricciones de orden temporal y de recursos, en el ámbito organizacional, constituye un aliciente que compromete permanentes esfuerzos por encontrar cada vez mejores modelos de solución. Comúnmente, este proceso se aborda empíricamente, con lo cual se obtienen soluciones estimativas que, no obstante ser satisfactorias, carecen de la confiabilidad propia de los métodos formalmente desarrollados.

El modelo híbrido constructivo-evolutivo propuesto aprovecha la potencialidad de los enfoques que le otorgan su denominación, lo cual significa que la estrategia de utilizar conjuntos de construcciones parciales y operadores evolutivos tiene la finalidad de reducir el tamaño del espacio de búsqueda, expandiendo sólo aquellas construcciones susceptibles de contribuir a la obtención de soluciones de buena calidad.

La prueba experimental del modelo se basó en el análisis comparativo de los resultados obtenidos mediante la utilización de 150 instancias de un problema que había sido resuelto previamente. Se pudo observar que la diferencia entre los valores obtenidos y los valores de referencia se incrementa con el aumento del número de tareas, lo cual significa que la calidad de los resultados del algoritmo constructivo-evolutivo es inversamente proporcional a la cantidad de tareas consideradas. Este fenómeno se debe al crecimiento explosivo de las combinaciones posibles cuando aumenta el número de tareas de un proyecto. Sin embargo, se satisfacen las expectativas cifradas, ya que el error promedio presente en los resultados obtenidos no supera el 20% en el caso de las instancias con la mayor cantidad de tareas. Por lo tanto, se puede concluir que el modelo constructivo-evolutivo propuesto es pertinente y viable para resolver el problema de la asignación de actividades con recursos limitados.

Como potenciales trabajos futuros, y con carácter de extensiones del presente, se sugiere adaptar el algoritmo a la asignación de tareas multi-modales, o bien a la inclusión de recursos no renovables.

Finalmente, es posible advertir que, sin perjuicio de la permanente preocupación por desarrollar modelos que mejoren la calidad de las soluciones para el RCPSP, aún queda bastante por hacer al respecto.

BIBLIOGRAFÍA

1 Xu, N., McKee, S., Nozick, L. and Ufomata, R. (2008). Augmenting priority rule heuristics with justification and rollout to solve the resource-constrained project scheduling problem. Computers & Operations Research, vol. 35, pags. 3284-3297.

2 Valls, V., Ballestín, F. and Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research, vol.185, pags. 495-508.

3 Moreno, F., Díaz, L., Pena, E. and Rivera, J. (2007). A comparative analysis between two heuristic algorithms for solving the Resource Constrained Project Scheduling Problem (RCPSP). Dyna-Colombia, vol. 74, pags.171-183.

4 Tseng, L. and Chen, S. (2006).A hybrid metaheuristic for the resource-constrained project scheduling problem. European Journal of Operational Research, vol. 175, pags. 707-721.

5 Hartmann, S. And Kolisch, R. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research, vol. 174, pags. 23-37.

6 Zhang, H., Li, H. and Tam, C. (2006). Permutation-based particle swarm optimization for resource-constrained project scheduling. Journal of Computing in Civil Engineering, vol. 20, pags.141-149.

7 Valls, V., Ballestín, F. and Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. European Journal of Operational Research, vol. 165, pags. 375-386.

8 Ballestín, F. (2002). Nuevos métodos de resolución del problema de secuenciación de proyectos con recursos limitados. Unpublished PhD Dissertation, Universidad de Valencia.

9 Hartmann, S. (2002). A Self-Adapting Genetic Algorithm for Project Scheduling under Resource Constraints. Naval Research Logistics, vol.49, págs. 433-448.

10 Merkle, D., Middendorf, M. and Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation, vol. 6, pags. 333-346.

11 Erenguc, S., Ahn, T. and Conway, D. (2001). The Resource Constrained Project Scheduling Problem with Multiple Crashable Modes: An Exact Solution Method. Naval Research Logistics, vol. 48(2), págs. 107-127.

12 Brucker, P. and Knust, S. (2000). A Linear Programming and Constraint Propagation-Based Lower Bound for the RCPSP. European Journal of Operational Research, vol. 127, págs. 355-362.

13 Dorndorf, U., Pesch, E. and Phan-Huy, T. (2000). A Branch and Bound Algorithm for the Resource Constrained Project Scheduling Problem. Mathematical Methods of Operations Research, vol. 52, págs. 413-439.

14 Kolisch, R. and Hartmann, S. (1999). Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. Project Scheduling: Recent Models, Algorithms and Applications, pags.147-178.

15 Palma, R. (1997). Solución al Problema de Corte de Piezas No Guillotina Utilizando un Método Híbrido Constructivo-Evolutivo. Tesis de Magíster, Departamento de Ingeniería Informática, Universidad de Santiago de Chile.

16 Gómes, A. and Parada, V. (1996). Métodos Meta-Heurísticos para Resolver Problemas de Corte de Piezas. Universidade Federal do Espírito Santo, Universidad de Santiago de Chile.

17 Osman, I. and Kelly, J. (1996). Meta-Heuristics: Theory and Applications. Ed. Kluwer Academic, Boston.

18 Simpson, W. And Patterson, J. (1996). A Multipletree Search Procedure for the Resource-Constrained Project Scheduling Problem. EJOR, Vol. 89, págs.525-542.

20 Mingozzi, A., Maniezzo, V., Ricciardelli, S. and Bianco, L. (1995). An Exact Algoritm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation. Department of Mathematics, University of Bologna, Bologna, Italia, Departmentof Electrical Engineering, University Tor Vergata, Roma, Italia.

21 Demeulemeester, E. (1992). Optimal Algorithms for Various Classes of Multiple Resource Constrained Project Scheduling Problems. Katholieke Universiteit Leuven, Bélgica.

22 Christofides, N., Alvarez-Valdez, R. and Tamarit, J. M. (1987). Project Scheduling with Resource Constraints: a Branch and Bound Approach. Europe Journal of Operations Research, vol. 29, págs. 262-273.

23 Patterson, J. (1984). A Comparison of Exact Approaches for Solving the Multiple Constrained Resources, Project Scheduling Problem. Management Science, vol. 30, págs.854-867.

24 Pearl, J.(1984).Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Publishing Company.

25 Blazewicz, J., Lenstra, J., and Rinnooy Kan A. (1983). Scheduling Subject to Resource Constraints: Classification and Complexity. Discrete Applied Mathematics, págs. 11-24.

26 Stinson, J., Davis, E. and Kjumawala, B. (1978) "Multiple Resource-Constrained Scheduling Using Branch and Bound. AIIE Transactions, vol. 10, págs. 252-259.

27 Elmaghraby, S. (1977). Activity Networks: Project Planning and Control by Network Models. Wiley, New York.

28 Fisher, M. (1973). Optimal Solution of Scheduling Problems Using Lagrange Multipliers. Part I, Operations Research, vol. 21, págs. 114-1127.

29 Gorenstein, S. (1972). An Algorithm for Project Sequencing with Resource Constraints. Operations Research, vol. 20, págs. 835-850.

30 Kolisch, R. and A. Sprecher (1996): PSPLIB - A project scheduling library, European Journal of Operational Research, Vol. 96, pp. 205--216.