matemática: Una publicación de FCNM – ESPOL
2014, Vol. 12, No. 1

IMPLEMENTACIÓN DE UN ALGORITMO GENÉTICO PARA RESOLVER EL PROBLEMA DE PROGRAMACIÓN DE PROYECTOS CON RECURSOS LIMITADOS

Narváez Gabriela[1], Saltos Ramiro[2]

Resumen. *Mayoritariamente en las empresas de producción industrial y en aquellas que brindan servicios de mantenimiento a las maquinarias, cotidianamente se enfrentan a un problema de índole operativo que debe ser resuelto por las personas competentes en el menor tiempo posible. Este equipo de trabajo tiene en frente un proyecto que está compuesto por una serie de actividades y procesos que tienen una duración determinada. Los procesos y actividades consumen los recursos escasos de la compañía y obedecen una serie de relaciones de precedencia. El equipo de trabajo deberá elaborar una secuencia según la cual se deben ejecutar las diferentes actividades y procesos, respetando todas las limitantes consideradas de tal manera que el tiempo de finalización del proyecto sea el menor posible. Este tipo de problemas suelen ser conocidos como Programación de Proyectos con Recursos Limitados (RCPSP por sus siglas en inglés). En este trabajo se hablará un poco sobre la teoría detrás de los algoritmos genéticos, conoceremos sus ventajas y, se diseñará e implementará uno de ellos basado en la secuenciación en serie que nos permita encontrar soluciones óptimas al problema planteado en el menor tiempo posible.*

Palabras Claves: Programación de Proyectos con Recursos Limitados, Algoritmos Genéticos.

Abstract. *Mainly in industrial production companies and those that provide maintenance services to machinery, daily face a problem of operational nature that must be resolved by the people involved in the shortest time possible. This team has in front a project that consists of a series of activities and processes that have a definite duration. The processes and activities consume scarce resources of the company and obey a set of precedence relationships. The team should develop a sequence according to which we should implement the various activities and processes, respecting all the constraints considered so that the project completion time is minimum. Such problems are usually known as Resource Constrained Project Scheduling (RCPSP). In this paper we will talk a little about the theory behind genetic algorithms, know their strengths and, design and implement one based on sequencing in series to allow us to find optimal solutions to the problem in the shortest time possible.*

Keywords: Resource Constrained Project Scheduling, Genetic Algorithms.

Recibido: Octubre 2013
Aceptado: Diciembre 2013

1. INTRODUCCIÓN

Mayoritariamente en las empresas de producción industrial y en aquellas que brindan servicios de mantenimiento a las maquinarias, cotidianamente se enfrentan a un problema de índole operativo que debe ser resuelto por las personas competentes en el menor tiempo posible. Este equipo de trabajo tiene en frente una obra, como usualmente se denominan a los proyectos industriales, o proyecto que está compuesto por una serie de actividades y procesos que tienen una duración determinada. Los procesos y actividades consumen los recursos escasos de la compañía. El equipo de trabajo deberá elaborar una secuencia según la cual se deben ejecutar las diferentes actividades y procesos, respetando la limitación de los recursos junto con un conjunto de restricciones adicionales de tal manera que el tiempo de finalización de la obra sea el menor posible.

Este tipo de problemas suelen ser conocidos como Programación de Proyectos con Recursos Limitados en donde las restricciones adicionales son las relaciones de precedencia de las actividades que conforman el proyecto.

2. OBJETIVOS

Los objetivos que se persiguen con el presente trabajo son:

1. Dar a conocer la importancia que tiene la programación de proyectos en las empresas.
2. Diseñar un algoritmo genético eficiente para resolver el RCPSP (Problema de Programación de Proyectos con Recursos Limitados por sus siglas en inglés).

Dar a conocer la necesidad que existe actualmente de desarrollar programas de optimización.

3. FUNDAMENTOS TEÓRICOS

El problema considerado en este trabajo es el de secuenciación de un proyecto con limitación de recursos y sin posibilidad de interrupción del proceso de las actividades que lo componen una vez

[1] *Narváez Gabriela, Ingeniera en Logística y Transporte, Escuela Superior Politécnica del Litoral (ESPOL). (e_mail: gabriela.narvaez.m@hotmail.com).*
[2] *Saltos Ramiro, Ingeniero en Logística y Transporte, Escuela Superior Politécnica del Litoral (ESPOL). (e_mail: ramirojsaltos@hotmail.com).*

iniciadas, y las relaciones de precedencia entre las tareas son del tipo fin – inicio, que quiere decir que una actividad no puede iniciar mientras todas sus predecesoras no hayan concluido. También suponemos que la duración de las actividades, los requerimientos de recursos y la disponibilidad de cada tipo de recurso son cantidades no negativas, conocidas y constantes a lo largo del tiempo en el que se realiza el proyecto. El objetivo es minimizar el tiempo total de ejecución [1].

3.1 DEFINICIÓN MATEMÁTICA DEL RCPSP

El Problema de Secuenciación de Proyectos (RCPSP) puede ser definido matemáticamente de la siguiente manera:
Un proyecto consiste en un conjunto $A = \{a_0, a_1, a_2, \dots, a_{n+1}\}$ de actividades y un conjunto $R = \{r_1, r_2, \dots, r_m\}$ de recursos renovables limitados. Las actividades a_0 y a_{n+1} son ficticias, tienen una duración de cero unidades de tiempo, no consumen ningún recurso y representan el inicio y fin del proyecto. Luego cada actividad a_i para $i = 1,2, \dots, n$ tienen una duración $d_i \geq 0$ y consumen una cantidad $q_{ij} \geq 0$ de los recursos siendo $Q_j \geq 0$ la disponibilidad máxima de cada recurso $r_j \in R$ en cada instante de tiempo. Adicionalmente existen restricciones de precedencia para las actividades del proyecto de tal manera que cada actividad a_i no puede ser iniciada mientras todas sus actividades predecesoras P_i no hayan sido finalizadas. De la misma manera se suele denotar S_i como el conjunto de actividades sucesoras de la actividad a_i para $i = 0,1, \dots, n+1$. El objetivo consiste en encontrar un conjunto $T = \{t_0, t_1, t_2, \dots, t_{n+1}\}$de tiempos de inicio de cada actividad que cumpla con las restricciones de precedencia y disponibilidad de los recursos, y minimice el tiempo total de duración del proyecto.

3.2 MODELO MATEMÁTICO DEL RCPSP

Sea A_t el conjunto de todas las actividades en proceso en el tiempo t, entonces el RCPSP puede ser modelado matemáticamente de la siguiente manera:

$$Min\ z = t_{n+1} \qquad (1)$$

St.

$$t_i > t_p + d_p \quad \forall a_i \in A\ \forall a_p \in P_i \qquad (2)$$

$$\sum_{a_i \in A_t} q_{ij} \leq Q_j \quad \forall r_j \in R\ \forall t \geq 0 \qquad (3)$$

La función objetivo (1) busca minimizar el tiempo de inicio de la actividad ficticia que representa el fin del proyecto, la ecuación (2) garantiza que se cumpla las restricciones de precedencia de las actividades y la última (3) asegura que no se sobrepase la disponibilidad máxima de los recursos en cada instante de tiempo.

3.3 COMPLEJIDAD COMPUTACIONAL DEL RCPSP

De acuerdo con la teoría de la complejidad computacional, el RCPSP es uno de los más difíciles de resolver entre los problemas de optimización combinatoria. El RCPSP pertenece a la clase de problemas NP - *hard* en sentido estricto [2].

4. ESQUEMAS ALGORÍTMICOS PARA EL RCPSP

Kelley y Walker distinguen dos formas básicas de generar una secuencia posible, que denominan *en serie y en paralelo.* En ambos procedimientos, una vez que una actividad ha sido introducida en la secuencia, nunca es resecuenciada [1].
La secuenciación en serie comienza numerando los vértices de forma que para cada arco del grafo de precedencia, su vértice inicial tiene un número menor que su vértice final. Este esquema de numeración tiene la propiedad de que si se secuencian las actividades en el orden indicado por su número, entonces ninguna actividad aparecerá antes que ninguna de sus predecesoras. Se puede construir una secuencia posible considerando las actividades en este orden y secuenciando cada una de ellas tan pronto como las relaciones de precedencia y las restricciones sobre los recursos lo permitan. El procedimiento de numeración no es, en general, único. El orden entre las mismas puede obtenerse de forma aleatoria o mediante una función de prioridad tal como el requerimiento de recursos, duración de las actividades, etc. Cada regla de prioridad define un algoritmo en serie diferente [1].
En la secuenciación en paralelo se construye una secuencia posible procediendo hacia delante en el tiempo. En cada momento durante la construcción, se determina el conjunto de actividades que pueden ser secuenciadas de acuerdo con las restricciones de precedencia y de recursos. Este conjunto se ordena mediante una regla de prioridad y las actividades se secuencian en ese orden mientras no se supere la capacidad de los recursos y así sucesivamente. Si las actividades reciben la prioridad independientemente de la secuencia ya existente, el

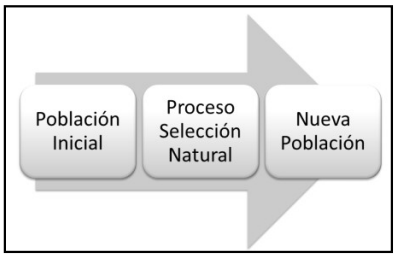
algoritmo se denomina estático. Caso contrario se denomina dinámico [1].
Una forma completamente diferente de obtener una secuencia posible es el *método de muestreo*. Los métodos de muestreo forman un conjunto de secuencias posibles usando técnicas de aleatorización y eligen la mejor secuencia obtenida [1]. La desventaja de utilizar este método radica en que no se dirige la búsqueda, simplemente se toma el mejor individuo de un vecindario y obtenemos óptimos locales que no necesariamente están cerca del óptimo global.

5. ALGORÍTMOS GENÉTICOS

5.1. FILOSOFÍA DE LOS ALGORITMOS GENÉTICOS

Los algoritmos genéticos son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Charles Darwin (1859). Por imitación de este proceso, los algoritmos genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas [4].

FIGURA 1
Implementación de un algoritmo genético para resolver el problema de programación de proyectos con recursos limitados
Proceso de Evolución de Darwin



El poder de los algoritmos genéticos proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el algoritmo genético encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria [4].

5.2. VENTAJAS DE LOS ALGORITMOS GENÉTICOS

Son intrínsecamente paralelos. La mayoría de los otros algoritmos son en serie y sólo pueden explorar el espacio de soluciones en una dirección al mismo tiempo, y si la solución que descubren resulta subóptima, no se puede hacer otra cosa que abandonar todo el trabajo hecho y empezar de nuevo. Sin embargo, ya que los algoritmos genéticos tienen descendencia múltiple, pueden explorar el espacio de soluciones en múltiples direcciones a la vez. Si un camino resulta ser un callejón sin salida, pueden eliminarlo fácilmente y continuar el trabajo en avenidas más prometedoras, dándoles una mayor probabilidad en cada ejecución de encontrar la solución [4].
Debido al paralelismo que les permite evaluar implícitamente muchos esquemas a la vez, los algoritmos genéticos funcionan particularmente bien resolviendo problemas cuyo espacio de soluciones potenciales es realmente grande – demasiado vasto para hacer una búsqueda exhaustiva en un tiempo razonable [4].
Los algoritmos evolutivos han demostrado su efectividad al escapar de los óptimos locales y descubrir el óptimo global incluso en paisajes adaptativos muy escabrosos y complejos [4].

5.3. ESQUEMA DE LOS ALGORITMOS GENÉTICOS

Los algoritmos genéticos tienen el siguiente esquema:

1. Existe una población inicial de individuos con determinadas características.
2. Los individuos se cruzan entre sí dando origen a los habitantes de la siguiente generación.
3. Estos habitantes heredarán las características de sus padres y si éstas son beneficiosas se irán propagando a través de las generaciones futuras, caso contrario desaparecerán por el proceso de selección natural que indica que sólo las especies más fuertes sobreviven.

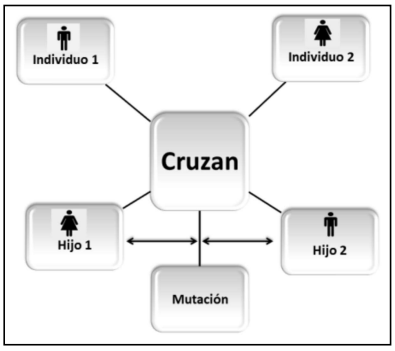
Durante el proceso de cruce puede darse la mutación del nuevo individuo, si esta mutación es mala seguramente el nuevo hijo morirá al nacer, pero si es beneficiosa se irá transmitiendo en las generaciones futuras. La mutación es un proceso mediante el cual se altera de forma significativa el código genético del individuo otorgándole nuevas características que nadie posee en la población, pero al mismo tiempo ocurre con muy poca frecuencia.

Posiblemente en una especie determinada uno de cada millón de habitantes ha mutado.

FIGURA 2

Implementación de un algoritmo genético para resolver el problema de programación de proyectos con recursos limitados

Esquema del Algoritmo Genético



5.3. ANALOGÍA CON LOS PROBLEMAS DE OPTIMIZACIÓN

Dado un problema de optimización, los algoritmos genéticos tienen la siguiente secuencia:

1. Codificar al individuo como un cromosoma perteneciente al espacio de soluciones factibles del problema.
2. Crear un conjunto de individuos como población inicial.
3. Aplicar el proceso de selección natural, que consiste en que, mediante algún criterio seleccionar las parejas que realizarán el cruce.
4. Realizar el cruce de las parejas formadas. El operador de cruce más usado es el punto de cruce.
5. Delimitar la nueva población al mismo tamaño de la población inicial seleccionando los mejores individuos.
6. Aplicar la mutación. La mutación clásica es el cambio de alelos. Se debe aplicar a menos del 1% de la población.
7. Actualizar la población inicial = nueva población.
8. Si se cumple el criterio de parada, finalizar y presentar la mejor solución encontrada, caso contrario volver al paso 3. El criterio de parada es libre.

6. DISEÑO DEL ALGORITMO GENÉTICO PARA EL RCPSP

El algoritmo genético implementado para resolver el RCPSP fue elaborado en base a los siguientes criterios de diseño.

6.1. CODIFICACIÓN

Las soluciones serán codificadas como una lista de tareas donde el orden de las actividades dentro de la misma indicará cuándo deben ser planificadas en el calendario de ejecución.

6.2. ELABORACIÓN DEL CROMOSOMA

El cromosoma será elaborado siguiendo la secuenciación en serie respetando las restricciones de precedencia que impone el problema. Para ello se tendrán en cada iteración dos conjuntos disjuntos, el conjunto de actividades secuenciadas y el conjunto de actividades elegibles.

Las actividades elegibles son todas aquellas actividades cuyos predecesores ya pertenecen al conjunto de actividades secuenciadas. De este conjunto se selecciona al azar una de ellas y se la agrega al final del conjunto de tareas secuenciadas. Este procedimiento se repite hasta completar el cromosoma.

6.3. CREACIÓN DE LA POBLACIÓN INICIAL

Para crear la población inicial se repetirá el proceso mencionado en el apartado anterior hasta completar el tamaño deseado de la misma.

6.4. FITNESS

El fitness representa la fortaleza del individuo creado o en este caso, qué tan buena es la solución encontrada. En nuestro problema el fitness estará dado por el tiempo total de ejecución del proyecto y mientras menor sea este, mejor será la solución generada.

6.5. SELECCIÓN NATURAL

Para el proceso de selección natural, las parejas serán formadas ordenando los individuos de la población inicial desde el más fuerte al más débil y a continuación se parearán el primero con el segundo, el tercero con el cuarto y así sucesivamente.

6.6. OPERADOR DE CRUCE

El operador de cruce es aquella función mediante la cual se crean los nuevos individuos. Esta función recibe una pareja de cromosomas y la cruza generando los hijos de la misma. Estos hijos deben garantizar el cumplimiento de las restricciones de

precedencia y para ello utilizaremos el siguiente procedimiento:

1. Se genera un número entero aleatorio u entre 1 y n el cual representará el punto de cruce entre los cromosomas.
2. El primer hijo hereda los genes del padre de izquierda a derecha hasta u, y completa su secuencia genética heredando los genes de la madre, de izquierda a derecha, que aún no formen parte del nuevo individuo.
3. De manera similar, el segundo hijo hereda los genes de la madre de izquierda a derecha hasta u, y completa su secuencia genética heredando los genes del padre, de izquierda a derecha, que aún no formen parte de la misma.

6.7. MUTACIÓN

La mutación es un mecanismo que juega un papel fundamental en cualquier algoritmo genético debido a que brinda la posibilidad de salir de un entorno de búsqueda monótono y pasar a otro que probablemente tenga entre sus habitantes al individuo óptimo.

Cuando apliquemos la mutación a alguno de los nuevos habitantes, debemos asegurarnos que el individuo mutado aún pertenezca a la misma especie que los demás habitantes, esto es, que aún preserve las restricciones de precedencia que forman parte del problema de optimización. Para garantizar que esto se mantenga, elaboramos un procedimiento de mutación que consiste en intercambiar un par de alelos del nuevo individuo, siempre y cuando estos alelos no tengan relaciones de dependencia. Los pasos a seguir para lograr esto son:

1. Generar un número aleatorio u entre 1 y $n-1$.
2. Si la actividad ubicada en el puesto u de la secuencia genética no es predecesora de la actividad situada en la posición $u+1$ de la misma secuencia, entonces intercambiar ambas actividades. Caso contrario volver al paso 1.

6.8. CRITERIO DE PARADA

El criterio de parada utilizado en nuestro algoritmo genético consistirá en el alcance de un número determinado de generaciones, después de las cuales el mejor individuo encontrado durante todo el proceso evolutivo será considerado como la mejor solución al problema planteado.

6.9. CREACIÓN DEL CALENDARIO DE EJECUCIÓN

El calendario de ejecución de las tareas o mejor conocido como el diagrama de Gantt es aquel gráfico donde se indica el tiempo en el cual va a ser iniciada cada tarea y la duración de la misma ya considerando las restricciones impuestas por la escasez de los recursos. Nosotros utilizaremos la secuenciación dada por el algoritmo genético y la creación del calendario será hacia delante en el tiempo. Para cumplir con esto debemos:

1. Seleccionar la actividad que corresponda según el orden establecido en la secuenciación.
2. Ubicar el tiempo de inicio de la actividad seleccionada en el paso 1 en el primer día donde se satisfaga las restricciones de disponibilidad de recursos y todos los predecesores de dicha actividad hayan concluido.
3. Eliminar la actividad seleccionada de la secuenciación.

Si ya se han ubicado todas las actividades en el calendario, finalizar el procedimiento y el fitness de esta secuenciación será el tiempo de inicio de la última actividad secuenciada, que dado nuestro enfoque será la tarea ficticia FIN, caso contrario regresar al paso 1.

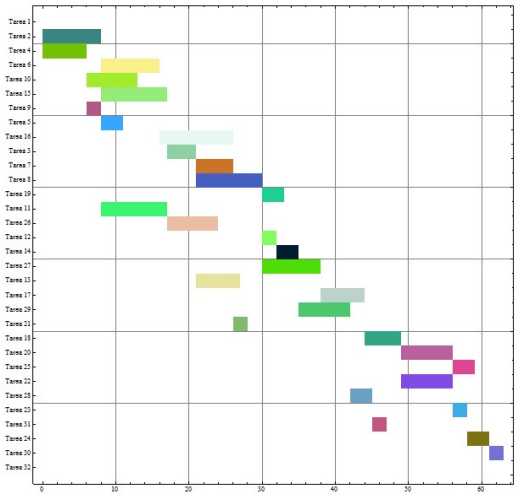
7. INFORME DE RESULTADOS

Al finalizar la ejecución del algoritmo genético, este presentará los resultados tal como se muestran en la figura 3:

FIGURA 3

Implementación de un algoritmo genético para resolver el problema de programación de proyectos con recursos limitados

Diagrama de Gantt



Donde el eje de las ordenadas muestra el orden en que son secuenciadas las tareas mientras que el eje de las abscisas indica el tiempo de inicio de cada una de ellas y la duración total del proyecto.

8. ANÁLISIS DEL ALGORITMO Y RESULTADOS

Para evaluar la eficacia del algoritmo genético desarrollado se corrieron 30 instancias de prueba obtenidas de [6] con un número de 30 tareas en el proyecto calibrando los parámetros con 50 individuos en la población inicial y el criterio de parada con 50 generaciones. Los resultados obtenidos se muestran en la siguiente tabla:

TABLA I

Implementación de un algoritmo genético para resolver el problema de programación de proyectos con recursos limitados

Resultados del Algoritmo Genético

Instancia	Sol AG	Opt	R-Gap	Time (s)
1	46	43	0,0697	130,18
2	51	47	0,0851	113,78
3	47	47	0	88,32
4	62	62	0	148,18
5	39	39	0	124,52
6	48	48	0	123,8
7	60	60	0	128,54
8	54	53	0,0188	105,03
9	49	49	0	129,98
10	45	45	0	109,27
11	38	38	0	105,64
12	51	51	0	146,56
13	43	43	0	92,18
14	43	43	0	124,83
15	51	51	0	121,15
16	47	47	0	116,34
17	48	48	0	116,69
18	54	54	0	117,27
19	65	65	0	123,11
20	59	59	0	137,85
21	49	49	0	106,49
22	60	60	0	119,25
23	47	47	0	119,51
24	57	57	0	119,55
25	59	59	0	121,39
26	45	45	0	115,86
27	56	56	0	132,14
28	55	55	0	121,23
29	38	38	0	111,28
30	48	48	0	113,98
Promedio			**0,0057**	**119,464**

Adicionalmente se corrieron 5 instancias con 60 tareas en el proyecto con los siguientes resultados:

TABLA II

Implementación de un algoritmo genético para resolver el problema de programación de proyectos con recursos limitados

Resultados del AG con 60 Tareas

Instancia	Sol AG	Opt	R-Gap	Time (s)
1	77	77	0	660,55
2	75	68	0,102941	676,78
3	73	68	0,073529	736,16
4	93	91	0,021978	674,46
5	78	73	0,068493	723,82
Promedio			**0,053388**	**694,354**

9. CONCLUSIONES Y RECOMENDACIONES

En este trabajo se desarrolló un algoritmo genético para resolver el problema de secuenciación de proyectos con recursos limitados y con relaciones de precedencia del tipo fin – inicio. A través de los experimentos se comprobó que nuestro algoritmo resultó ser una herramienta adaptativa muy eficaz para resolver este tipo de problemas hallando las soluciones óptimas para problemas de tamaño relativamente pequeños en un intervalo corto de tiempo.

Para instancias de mayor tamaño el algoritmo encuentra buenas soluciones con un gap no mayor al 20% con un tiempo de ejecución aceptable en la práctica, esto es, no mayor a dos horas.

Por medio de los experimentos se constató que la convergencia al óptimo de las instancias depende en cierta medida de que tan buenos o malos eran los fitness de los individuos que conformaban la población inicial, esto es, si el mejor individuo en la población inicial ya era la solución óptima del problema, entonces el algoritmo con toda certeza llegaría al óptimo debido a que se decidió guardar el mejor individuo que se naciera en cada generación reemplazando al anterior si este no era mejor que la nueva secuencia generada.

En el caso que no estuviera la secuencia óptima en la población inicial, el algoritmo llegaría a generarla dependiendo de la cantidad de generaciones estén programadas así mismo como de los fitness de los individuos de la población inicial y el número de tareas del proyecto.

La adecuada programación de las tareas de un proyecto es de vital importancia en las áreas operativas de una empresa debido a que una mala planificación puede acarrear grandes pérdidas económicas para la compañía. Gracias a la tecnología existente se pueden evitar estos

problemas porque la planificación de los proyectos ya no se realizarán por el sentido común del planificador que hasta ahora lo hacía usando papel y lápiz, o en su defecto un software no optimizador como Microsoft Project, con el cual usando una instancia de 30 tareas nos demoramos media hora en sólo ingresar las precedencias de las tareas y no se consideran las limitaciones impuestas por la escasez de los recursos.

Por lo mencionado anteriormente salta a la luz la necesidad de desarrollar aplicaciones eficientes y fáciles de usar de manera que la planificación elaborada por el software sea sofisticada y optimice los recursos de la empresa. Esto es minimice el gasto que se incurre en la utilización de los recursos y el tiempo que se necesitaría para terminar el proyecto, lo cual mejora el nivel de servicio ofrecido por la empresa a sus clientes.

Finalmente, nuestro algoritmo genético puede ser utilizado como motor para la creación de este tipo de soluciones empresariales.

Para futuros trabajos de investigación dentro de este ámbito damos las siguientes recomendaciones:

- Cambiar el operador de cruce de uno a más puntos para mejorar las soluciones.
- Sugerimos usar dos puntos fijos para instancias de 50 a 70 tareas y 3 puntos fijos o más para instancias mayores.
- Recordar que el operador de cruce que se diseñe debe mantener las relaciones de precedencia de las tareas.
- Modificar el operador de mutación incrementando la probabilidad de que un individuo sea sujeto de una alteración en su secuencia genética.

REFERENCIAS BIBLIOGRÁFICAS Y ELECTRÓNICAS

[1]. **VALDÉS, R. Y TAMARIT, J.,** (1989). **"***Algoritmos Heurísticos Deterministas y Aleatorios en Secuenciación de Proyectos con Recursos Limitados*", Universidad de Valencia.

[2]. **ARTIGUES, C. ET AL.,** (2008). "*Resource Constrained Project Scheduling*", Iste and Wiley, Primera Edición, pp. 23 – 24.

[3]. **BUSTAMANTE, J.,** (2009). "*Apuntes de Metaheurísticas y Redes Neuronales*", Escuela Superior Politécnica del Litoral.

[4]. **LONDOÑO, N.,** (2006). "*Algoritmos Genéticos*", Universidad Nacional de Colombia.

[5]. **REEVES, C. Y ROWE, J.,** (2003). "*Genetic* **Algorithms Principles and Perspectives",** Kluwer Academics Publishers, Primera Edición.

[6]. PSP Lib., http://129.187.106.231/psplib/

[7]. Wolfram Research Inc., www.wolfram.com