UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Sistemas Inteligentes
y Aplicaciones Numéricas en la Ingeniería

ESTUDIO E IMPLEMETACIÓN DE UN ALGORITMO PARA LA PROGRAMACIÓN ÓPTIMA DE PROYECTOS

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Máster Universitario en Sistemas Inteligentes y Aplicaciones Numéricas en la Ingeniería

Autor:

Rubén Santana Rodríguez

Tutores:

Luis Alberto Padrón Hernández
David Juan Greiner Sánchez

Las Palmas de Gran Canaria, Diciembre 2014

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: La Voluntad.

Albert Einstein

Agradecimientos

Agradecerles a *mis tutores Dr. D. Luis Alberto Padrón Hernández y Dr. D. David Juan Greiner Sánchez quienes han hecho posible el presente trabajo. Agradecerles nuevamente el haberme dado la oportunidad de iniciarme en el mundo de la investigación y compartir toda su experiencia para realizar con rigor este trabajo.*

Resumen

La gestión de proyectos, y en particular, la planificación, es hoy en día un aspecto importante en muchas empresas. Se demuestra que una empresa con una buena organización y planificación de sus recursos (humano, maquinaria y materiales) hacen que la misma sea más eficiente y rentable.

Es un área de conocimiento bastante compleja, teniendo que disponer de un equipo técnico cualificado y competente capaz de conocer el potencial de la empresa y sus debilidades para tomar decisiones. El inconveniente de esa toma de decisiones es que en muchas ocasiones surgen imprevistos o condiciones que están fuera de alcance para poder tomar medidas correctoras.

Mis experiencias en el campo de la ingeniería, concretamente en las obras de ejecución de obras civiles, han motivado el plantearme otras formas de planificar estas obras que la mayoría son de gran envergadura, teniendo muchas veces que coordinar diferentes subcontratas para realizar diferentes actividades de la obra.

Basándonos en los trabajos del Dr. D. Jorge Magalhães Mendes (Politécnico de Oporto, Portugal), [1-5], el objetivo principal de este documento es minimizar el periodo de ejecución de un proyecto gracias a un algoritmo de programación de proyectos, teniendo en cuenta solamente la duración de las actividades y sus cantidades de recursos tipo. En este caso, no se ha tenido en cuenta la minimización de los costes.

La metodología empleada para este trabajo ha sido:

- Utilizar Red de precedencias para identificar cada una las actividades que están involucradas en el proyecto.
- Proceso de construcción de una programación activa parametrizada que sea capaz de colocar cada una de las actividades en función de sus restricciones de recursos, tiempos de inicio y fin, retraso de las actividades y sus precedencias.

- Utilización de un algoritmo genético, cuya función objetivo es minimizar el periodo de ejecución del proyecto.

Teniendo definida la metodología, se tomaron tres casos distintos de programación en las cuales se modificaron los parámetros de entradas para obtener diferentes resultados (número de actividades, cantidad de recursos disponibles, cantidad de recursos tipo, número de generaciones, población del algoritmo genético,…).

Finalmente los resultados fueron satisfactorios, cumpliendo el objetivo de implementar un programa que permite una programación óptima de proyectos utilizando algoritmos genéticos.

Este algoritmo implementado es un primer paso y será, en líneas futuras, una herramienta potente para la toma de decisiones en un proyecto y ayudará al director del mismo a controlar en todo momento el número de recursos de que dispondría la empresa en cada instante y a conocer, de una forma eficaz, la viabilidad del proyecto.

Summary

Nowadays, the project management, and in particular the planning schedule is an important aspect in a lot of companies. It is shown that a company with a good organization and planning of its resources (human, machines and equipment) make it more efficient and profitable.

It is a very complex area of knowledge, having to have a qualified and competent technical team capable of understanding the potential of the company and its weaknesses to make decisions. The main inconvenient of this decision making is that in several cases an unexpected event or condition could happen. This force that in some cases the correctives actions can be out of our scope.

Based on my own experiences at the engineering field, particularly in the scope of civil works, planning these types of works has motivated me to wonder other ways of planning. Most of them have a large magnitude. Due to that, it is usual the need of having to coordinate different subcontractors to perform different activities of the work.

Based on the investigation of Dr. D. Jorge Mendes Magalhães (Polytechnic of Oporto, Portugal), [1-5], the main objective of this document is to minimize the period of project implementation through a project scheduling algorithm, taking only into account the duration of its activities and the amount of different resources. In this case, is not taken into account the minimization of costs.

The methodology for this study has been:

- Using the precedence network to identify each activities that are involved in the project.
- Process of making a parameterized active schedule to be able to place each of the activities in terms of their resource constraints, start and end time, its delays and their precedence.
- Using a genetic algorithm, whose objective function is to minimize the period of project implementation.

Having defined the methodology, three different cases of programming were taken in which input parameters were modified to obtain different results (number of activities, available resources, amount of different kind of resources, number of generations, population genetic algorithm...).

Finally the results were satisfactory, fulfilling the objective of implementing a program that enables optimal project scheduling using genetic algorithms.

The implementation of this algorithm is a first step, and will be in future a powerful tool for decision making in a project and will help the manager to control in every moment the number of resources that would have the company at every instant and to know in an effective manner, the viability of the project.

Índice general

1 INTRODUCCIÓN 1

1.1 MOTIVACIÓN 1
1.2 OBJETIVO DEL TRABAJO 2
1.3 ESTRUCTURA DEL DOCUMENTO 3

2 METODOLOGÍA 4

2.1 INTRODUCCIÓN 4
2.1.1 CONCEPTO DE PROYECTO 6
2.2 ALGORITMO PARA LA GENERACIÓN DE PROGRAMACIONES ACTIVAS PARAMETRIZADAS 8
2.2.1 FUNDAMENTOS Y GENERALIDADES 8
2.2.2 TIPOS DE PROGRAMACIONES 10
2.2.3 PSEUDO-CÓDIGO 13
2.2.4 TRAZA DEL ALGORITMO 17
2.3 ALGORITMO GENÉTICO 25
2.3.1 REPRESENTACIÓN DEL CROMOSOMA 25
2.3.2 ESTRATEGIA EVOLUTIVA 26
2.4 FUNCIÓN OBJETIVO 27

3 RESULTADOS 28

3.1 CASO 1 29
3.1.1 DESCRIPCIÓN 29
3.1.2 PARÁMETROS DEL ALGORITMO GENÉTICO Y DE LA PROGRAMACIÓN 30
3.1.3 RESUMEN DE RESULTADOS 31
3.1.4 MEJOR CROMOSOMA POR CADA LANZAMIENTO 33
3.2 CASO 2 35

3.2.1 DESCRIPCIÓN 35
3.2.2 PARÁMETROS DEL ALGORITMO GENÉTICO Y DE LA PROGRAMACIÓN 37
3.2.3 RESUMEN DE RESULTADOS 37
3.2.4 MEJOR CROMOSOMA POR CADA LANZAMIENTO 39
3.3 CASO 3 41
3.3.1 DESCRIPCIÓN 41
3.3.2 PARÁMETROS DEL ALGORITMO GENÉTICO Y DE LA PROGRAMACIÓN 43
3.3.3 RESUMEN DE RESULTADOS 44
3.3.4 MEJOR CROMOSOMA POR CADA LANZAMIENTO 46

4 CONCLUSIONES Y LÍNEAS FUTURAS 50

4.1 CONCLUSIONES 50
4.2 LÍNEAS FUTURAS 52

5 BIBLIOGRAFÍA 53

ANEXO 1 – ALGORITMO (EJEMPLO J30) 54

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

El presente Trabajo Fin de Máster está integrado en la línea de investigación que llevan a cabo los miembros de la División de Computación Evolutiva y Aplicaciones (CEANI) del Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en la Ingeniería (SIANI) de la Universidad de Las Palmas de Gran Canaria en el campo de la computación evolutiva y, en particular, en la optimización de problemas de ingeniería (ej: [6]).

Mis experiencias en el campo de la ingeniería, concretamente en las obras de ejecución de obras civiles, han motivado el plantearme otras formas de planificar estas obras que la mayoría son de gran envergadura, teniendo muchas veces que coordinar diferentes subcontratas para realizar diferentes actividades de la obra.

Un alto porcentaje de dichas obras no cumplen los plazos de ejecuciones previstos, ya sea por falta de financiación, problemas en la coordinación de las actividades que se realizan muchas de ellas en paralelo, cargas de trabajo abusivas a los jefes de obras, teniendo muchas veces que optar por la improvisación cada vez que exista alguna situación inesperada en la obra y la falta de stock de materiales a pie de obra debido a la ineficiente planificación de la logística y de los recursos humanos.

La importancia en la gestión de proyectos ha ido en aumento, sobre todo desde su reconocimiento como una actividad fundamental en las organizaciones, sujetas a la competencia en una economía de mercado. Esta importancia se refleja en la capacidad de organización de la competencia, un aumento de productividad y el aumento de la calidad.

La gestión del proyecto comprende un complejo proceso de toma de decisiones, en particular a nivel de planificación y programación, en la que se presta especial atención al cumplimiento de los plazos de entrega y los recursos.

El planeamiento general implica el conjunto de actividades, las restricciones de dependencia entre las actividades y las cantidades de recursos necesarios para realizar las actividades relacionadas en el proyecto.

1.2 OBJETIVO DEL TRABAJO

Para llevar a cabo este estudio, es necesario conocer en profundidad cada uno de los procesos de proyección y ejecución de un proyecto de ingeniería. El autor de este documento, Ingeniero Civil, se ha especializado en la gestión de proyectos de ingeniería y formado, durante estos 2 últimos años, en conocer diferentes metodologías de programación de proyectos, metodologías de optimización y en programación en lenguajes C y C++.

Basándonos en el libro [5] y en el artículo [1] del Dr. D. Jorge Magalhães Mendes (Politécnico de Oporto, Portugal), especialista en metodologías de programación de proyectos y optimización, este documento estudia en profundidad e implementa un algoritmo que permite realizar una programación óptima de proyectos cuyos recursos son limitados. Este algoritmo es capaz de programar las diferentes actividades de recursos limitados que el proyecto exija y devuelva el esquema óptimo de actividades, para ello el programa se basará en unos criterios de selección para la colocación en el cronograma de dichas actividades.

Por tanto, el objetivo principal de este documento es minimizar el periodo de ejecución del proyecto, teniendo en cuenta solamente la duración de las actividades y sus cantidades de recursos tipo. No se ha tenido en cuenta la minimización de los costes.

1.3 ESTRUCTURA DEL DOCUMENTO

En este punto se tratará de resumir los diferentes apartados del presente trabajo, resaltando los aspectos más importantes en cada uno de ellos.

En el apartado 2 Metodología, se explicarán los fundamentos, generalidades, pseudo-código y algunos ejemplos acerca del algoritmo para la generación de programaciones activas parametrizadas. Además se hará una breve descripción de la generalidad y de los operadores utilizados del algoritmo genético, además de la función objetivo elegida.

Continuando con el apartado 3 se muestran los Resultados obtenidos en los diferentes casos. En primer lugar se definen los parámetros de entrada del problema y luego se muestran los resultados obtenidos en cada ejemplo.

Finalmente, en el apartado 4, se presentan las diferentes conclusiones y análisis de resultados obtenidos descritos en el apartado anterior. Se comentan también, las líneas futuras que nacen de este documento.

2 METODOLOGÍA

2.1 INTRODUCCIÓN

En el planeamiento de proyectos son frecuentemente utilizados los métodos clásicos siguientes:

- **Diagrama de Gantt**. Es una útil herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. A pesar de esto, el Diagrama de Gantt no indica las relaciones existentes entre actividades.
- **CPM (Critical Path Method)**. Es un algoritmo utilizado para el cálculo de tiempos y plazos en la planificación de proyectos. En administración y gestión de proyectos, una ruta crítica es la secuencia de los elementos terminales de la red de proyectos con la mayor duración entre ellos, determinando el tiempo más corto en el que es posible completar el proyecto. La duración de la ruta crítica determina la duración del proyecto entero. Cualquier retraso en un elemento de la ruta crítica afecta a la fecha de término planeada del proyecto, y se dice que no hay holgura en la ruta crítica.
- **PERT (Program Evaluation and Review Technique - Programa de Evaluación y Revisión Técnica)**. Es un método para analizar las tareas involucradas en completar un proyecto dado, especialmente el tiempo para completar cada tarea, e identificar el tiempo mínimo necesario para completar el proyecto total.

A pesar de sus grandes divulgaciones, estos métodos tienen serias limitaciones porque asumen que los recursos son ilimitados y sólo se aplican a un solo proyecto y no a múltiples proyectos.

En este trabajo, se analiza e implementa la metodología del artículo del Dr. Mendes [1]. Esta metodología se basa en tres partes fundamentales:

- **Utiliza Red de precedencias** para identificar cada una las actividades que están involucradas en el proyecto. Esta Red permite conocer qué actividades son predecesoras y sucesoras a cada actividad y sus duraciones. Se añade a esta red la cantidad de recursos tipo necesarios para cada actividad. Al saber que los recursos disponibles son limitados, es importante conocer qué cantidad total de cada recurso tipo se dispone y cuántos recursos de cada tipo necesita cada actividad.
- **Se realiza un proceso de construcción de una programación activa parametrizada**. Una vez conocido la red de precedencia de las actividades del proyecto, se implementa un algoritmo que sea capaz de colocar cada una de las actividades en función de sus restricciones de recursos, tiempos de inicio y fin, retraso de las actividades y sus precedencias.
- **Utilización de un algoritmo genético**, cuya función objetivo es minimizar el periodo de ejecución del proyecto. De esta forma se consigue optimizar los espacios vacíos de tiempo y recursos y la colocación óptima de las actividades en la línea de tiempo (optimización del cromosoma - vectores PRIORITY y tdelay).

En la ejecución de un proyecto de ingeniería civil, existen numerosas variables que dificultan poder definir con exactitud las duraciones de tiempo de las actividades (factor humano, maquinaria, meteorología, etc.). Es por ello que se utilizará en el futuro el método PERT, ya que las redes PERT trabajan con tiempos probabilísticos, a diferencia las redes CPM, que usan tiempos ciertos (reales o determinísticos).

2.1.1 CONCEPTO DE PROYECTO

Haciendo mención al diccionario de la Real Academia Española de la lengua (RAE), en su descripción 4ª se define **Proyecto** como *"Conjunto de escritos, cálculos y dibujos que se hacen para dar idea de cómo ha de ser y lo que ha de costar una obra de arquitectura o de ingeniería"*. En un sentido más amplio, se puede definir **Proyecto** como *un proceso único, que consiste en un conjunto de actividades coordinadas y controladas con fechas de inicio y fin, que compiten para realizar los objetivos de acuerdo a las necesidades específicas, incluidas las limitaciones de tiempo, coste y recursos.*

Todos los proyectos, en mayor o menor extensión, tienen algunas características en común, pudiéndolas definir de la siguiente forma:

- Un objetivo a alcanzar.
- Unicidad. Los proyectos son únicos e irrepetibles.
- Complejidad. Las relaciones entre las actividades.
- Carácter temporal. Fechas de inicio y fin para realizar el proyecto.
- Incertidumbre. Elementos de riesgos difíciles de predecir.
- Ciclo de vida. Diferentes fases del proyecto.

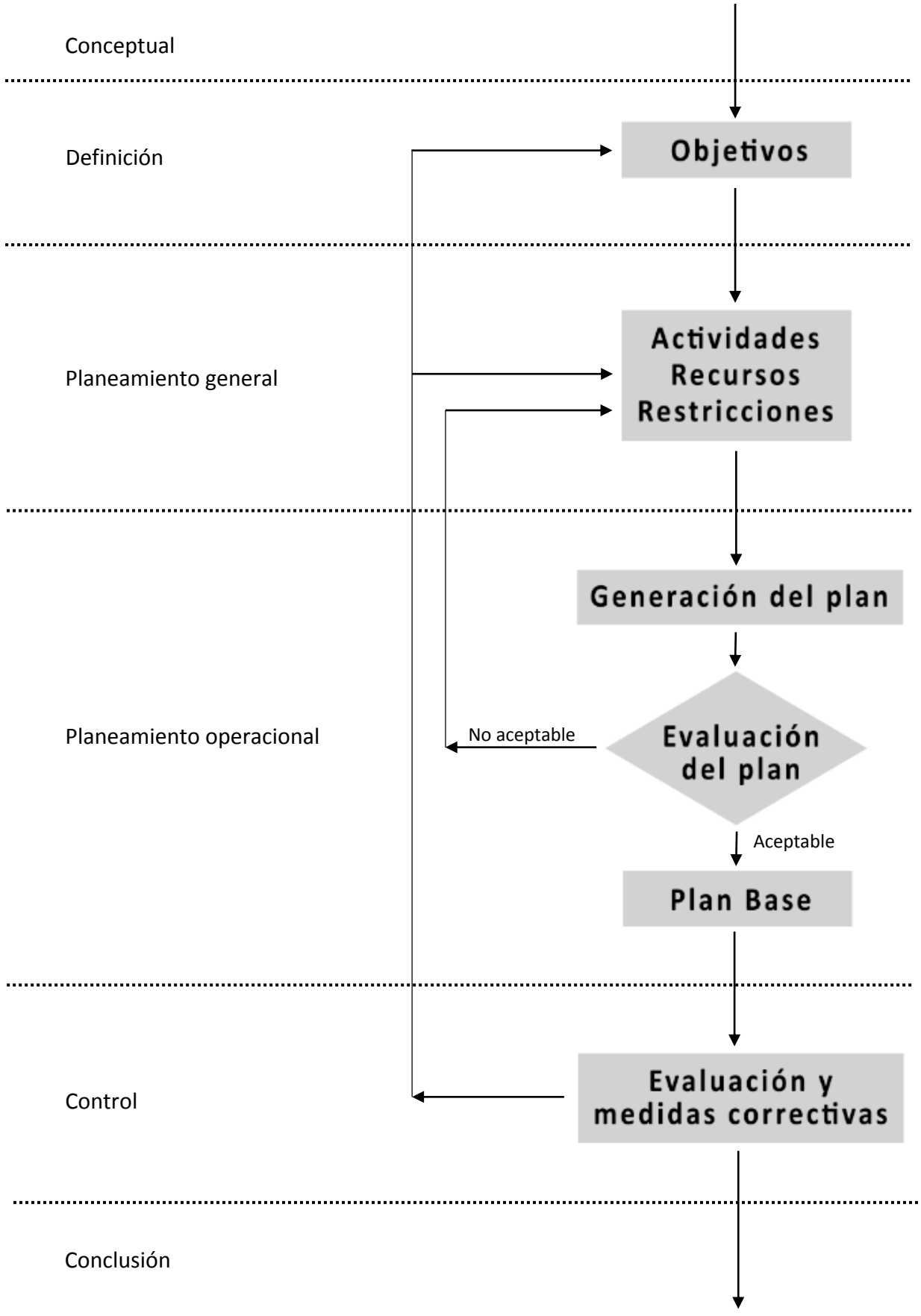

Fig. 2.1 Fases de un Proyecto según Mendes [5].

2.2 ALGORITMO PARA LA GENERACIÓN DE PROGRAMACIONES ACTIVAS PARAMETRIZADAS

2.2.1 FUNDAMENTOS Y GENERALIDADES

Según Mendes [5], un Proyecto consta de n + 2 actividades donde cada actividad forma una parte fundamental para la finalización del proyecto. Siendo J = {0, 1, …, n, n + 1} el conjunto de actividades a programar en el calendario y, K = {1, …, k} el conjunto de recursos. Las actividades 0 y n + 1 son actividades no utilizadas (sin duración) ya que sólo sirven para definir el inicio y el fin de las actividades del proyecto. Las actividades están interrelacionadas por 2 tipos de restricciones:

1) **Restricciones de precedencia** fuerzan a cada actividad j a ser programada después de que terminen todas las actividades predecesoras P_j.
2) Las actividades requieren recursos con **capacidades limitadas**.

Cada activad j requiere $r_{j,k}$ unidades de recursos tipo $k \in K$ durante todo el instante de tiempo que dura la actividad (d_j). El recurso tipo k tiene una limitada capacidad de R_k en cada uno de los instantes del tiempo. Para iniciar y terminar las actividades, las duraciones $d_0 = d_{n+1}=0$ y $r_{0,k}=r_{n+1,k}=0$, para todo $k \in K$. El problema consiste en encontrar una programación de actividades que, tomando en cuenta los recursos y las restricciones de precedencia, hagan minimizar el valor del tiempo de ejecución del proyecto (C_{max}).

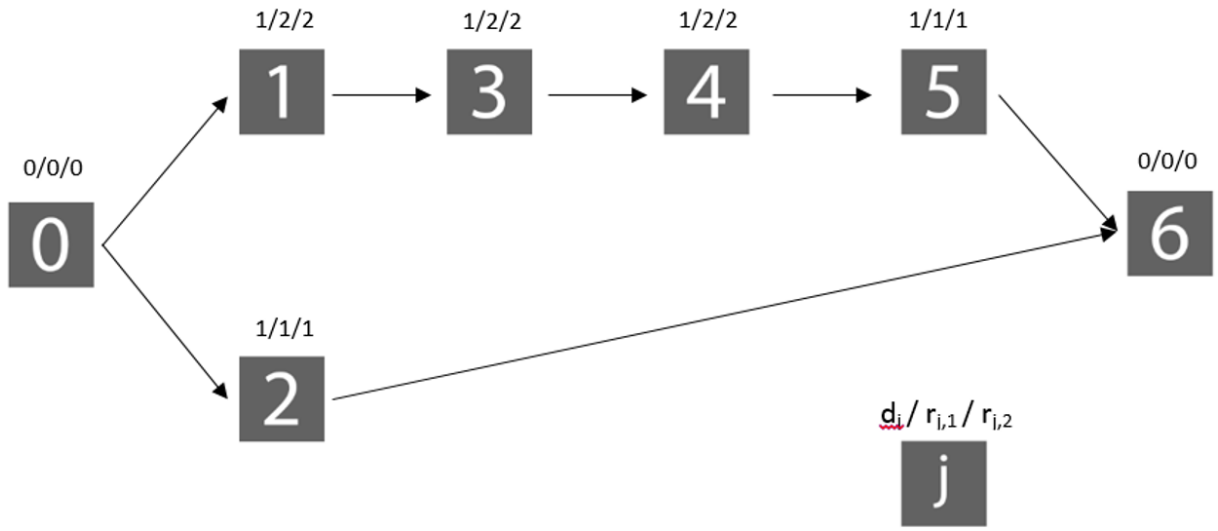

Fig. 2.2 Ejemplo de una red de Proyecto.

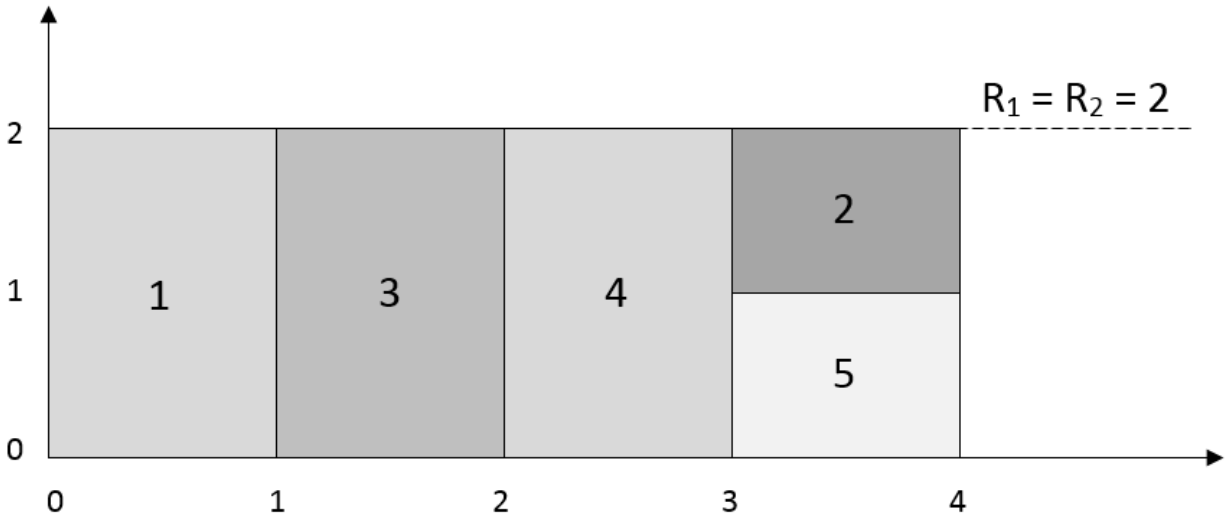

Fig. 2.3 Programación viable con un tiempo de ejecución del proyecto óptima de 4 para la red de Proyecto de la Fig. 2.2.

Siendo F_j los tiempos finales de las actividades j, una programación se representa por un vector de tiempos finales (F_1, ..., F_m, ..., F_{n+1}). La Fig. 2.2 muestra un ejemplo de un proyecto comprendido por $n = 5$ actividades a programar, sujeto a dos recursos tipo renovables con una capacidad de 2 unidades ambas. En la Fig. 2.3 se representa un calendario factible con un óptimo de tiempo de ejecución del proyecto de 4 unidades de tiempo.

2.2.2 TIPOS DE PROGRAMACIONES

Los calendarios son clasificados de la siguiente forma:

1) ***Programación Semiactiva***. Estas son programaciones obtenidas por las actividades en secuencia tan pronto como sea posible. En una programación Semiactiva, ninguna actividad puede empezar antes alterando las secuencias de procesamiento.
2) ***Programación Activa***. Son programaciones en las que ninguna actividad puede empezar anteriormente sin retrasar algunas actividades o romper una restricción de precedencia. Las programaciones activas son también programaciones Semiactivas. Una programación óptima siempre es Activa, de este modo, el espacio de búsqueda puede ser limitado con seguridad para el conjunto de todas las actividades de la programación.
3) ***Programación sin retrasos***. Son programaciones en las que ningún recurso se mantiene inactivo cuando podría comenzar el procesamiento de alguna actividad. Las programaciones sin retrasos son Activas y por lo tanto son también Semiactivas.

2.2.2.1 PROGRAMACIONES ACTIVAS PARAMETRIZADAS

Como se mencionó anteriormente, la programación óptima está en el conjunto de todas las programaciones activas. Sin embargo, el conjunto de actividades suele ser muy extenso y contiene muchas programaciones con grandes tiempos de retraso, dando poca calidad al valor del tiempo de ejecución del proyecto. El tiempo de retraso es un valor probabilístico que

determina la holgura de tiempo máximo en el que una actividad puede retrasarse. Para reducir el espacio de soluciones y controlar los tiempos de retraso, se usa el concepto de programación activa parametrizada.

En la siguiente Fig. 2.4, se muestra dónde se encuentra el conjunto de programaciones activas parametrizadas relativo a la clase de Semiactivas, Activas y sin retraso. Mediante el control del tiempo de retraso máximo permitido, se puede reducir o aumentar el tamaño del espacio de soluciones.

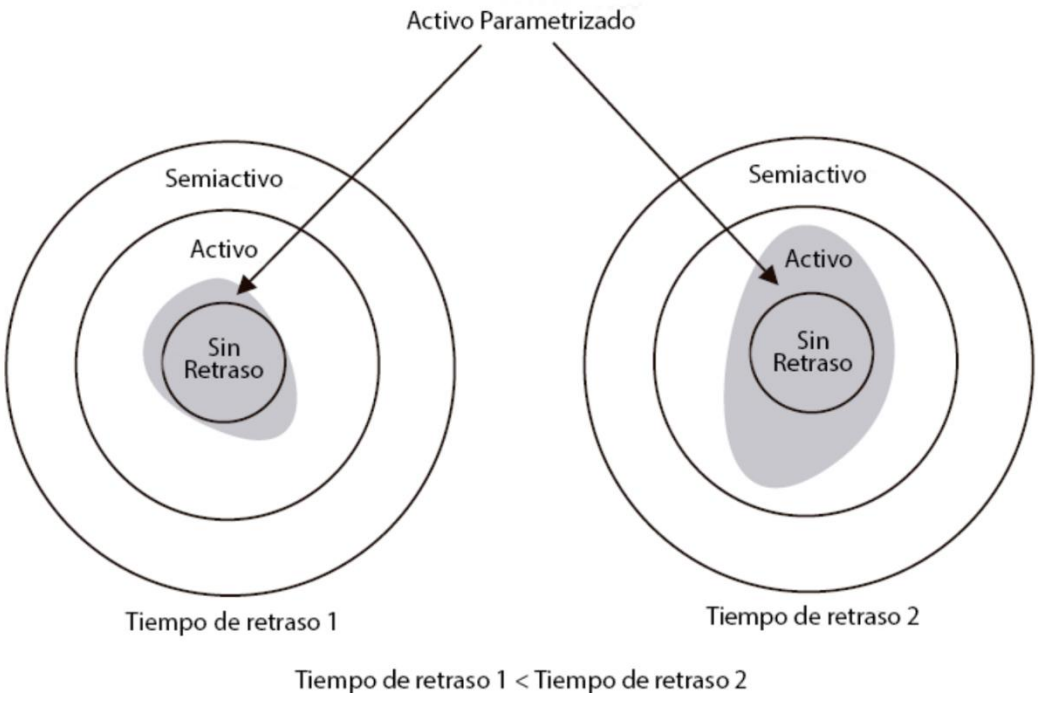

Fig. 2.4 Programaciones parametrizadas activas para diferentes valores de tiempo de retraso.

Haciendo mención a todo lo anterior, para generar las programaciones activas parametrizadas se combina un algoritmo genético y un planificador. Dicha combinación calcula además un valor del tiempo de ejecución modificado (MM) que se usa como medida de calidad y feedback para el algoritmo genético. El algoritmo genético evoluciona los cromosomas que representan las prioridades (PRIORITY) de las actividades y los tiempos de retrasos (tdelay). Para cada cromosoma se han de aplicar las siguientes fases:

1) ***Descodificación de prioridades y tiempos de retraso***. Esta fase es responsable de transformar el cromosoma suministrado por el algoritmo genético en las prioridades y tiempos de retraso de las actividades.
2) ***Generación de la Programación***. Esta fase hace uso de las prioridades y los tiempos de retraso definidos en la primera fase y construye las programaciones activas parametrizadas.

Una vez obtenida la programación, la correspondiente medida de calidad (MM) sirve de feedback para el algoritmo genético (función objetivo a evaluar).

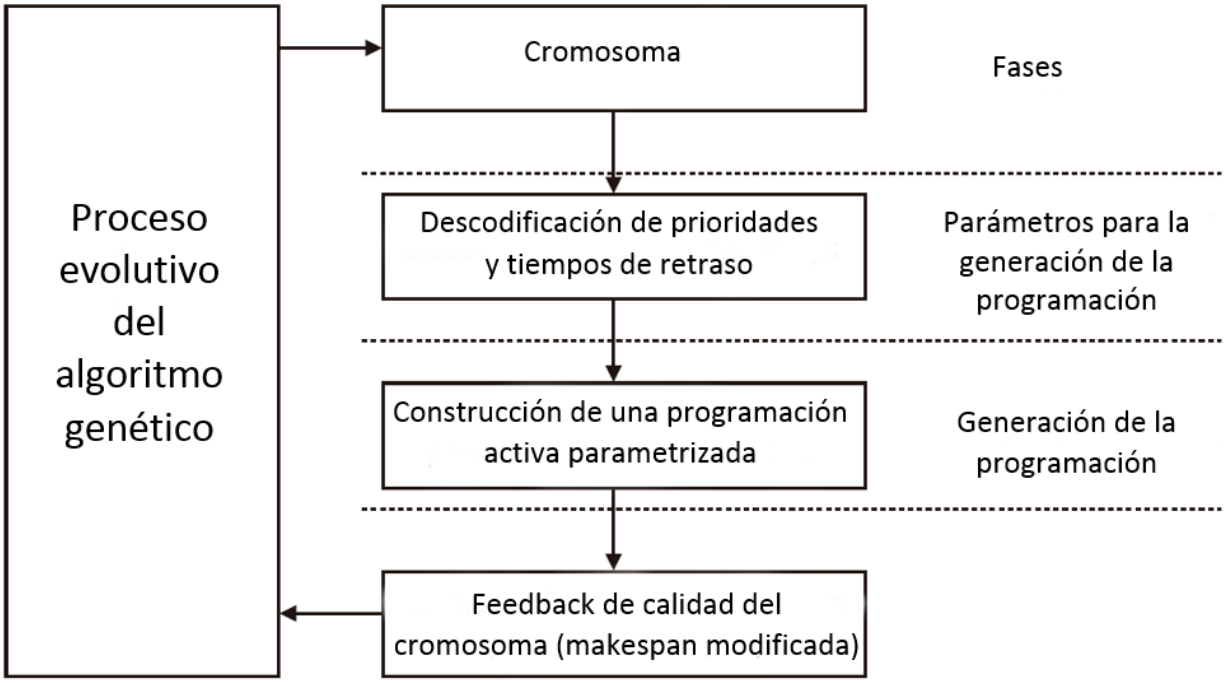

Fig. 2.5 Proceso evolutivo del algoritmo genético.

2.2.3 PSEUDO-CÓDIGO

Haciendo mención al artículo [1] y el libro [5] publicados por el Dr. Mendes, existen algunas erratas en el proceso de construcción de una programación activa parametrizada. En la siguiente ilustración, se muestra el código tal y como debería ser:

Initialize: $g = 1;\ t_1 = 0;\ A_0 = \{0\};\ T_0 = \{0\};\ S_0 = \{0\};$

$for\ \ k \epsilon K\ \ do\ \ \{initialize\ RD_k(0) = R_k(k \epsilon K)\};$

$while\ \ |S_g| < n + 2\ \ do\ \ \{$

$while\ \ E_g \neq \{\ \}\ \ do\ \ \{$

$j^* = argmax\{PRIORITY_j\} \qquad j \epsilon E_g;$

$EF_{j^*} = \max\{F_i\} + d_{j^*} \qquad i \epsilon P_{j^*};$

$F_{j^*} = \min\{t \in [(EF_{j^*} - d_{j^*}), \infty] \cap T_{g-1} \mid r_{j^*,k} \leq RD_k(\tau),\ k \epsilon K \mid r_{j^*,k} \geq 0,\ \tau \epsilon [t, (t + d_{j^*})]\} + d_{j^*};$

$S_g = S_{g-1} \cup \{j^*\}\ \ (Update);$

$T_g = T_{g-1} \cup \{F_{j^*}\}\ \ (Update);$

$g = g + 1;$

$A_g = \{j \in J \mid F_j - d_j \leq t_g < F_j\}\ \ (Update);$

$E_g = \{j \in J \setminus S_{g-1} \mid F_i \leq t_g + tdelay_g\}\ \ for\ i \in P_j;$

$RD_k(t) = R_k(t_g) - \sum_{j \in A_g} r_{j^*,k} \mid t \in [(F_{j^*} - d_{j^*}), F_{j^*}], k \in K \mid r_{j^*,k} \geq 0\ \ (Update);$

$\}$

$t_g = min\{t \in T_{g-1} \mid t > t_{g-1}\};$

$\}$

Fig. 2.6 Pseudo-código del proceso de construcción de la programación activa parametrizada (CORREGIDO).

A continuación se explica paso a paso cada uno de los apartados que compone el pseudo-código:

$\boldsymbol{Initialize}$: $\boldsymbol{g = 1;\ t_1 = 0;\ A_0 = \{0\};\ T_0 = \{0\};\ S_0 = \{0\};}$

Inicializamos las siguientes variables:

- **g:** número de iteraciones.
- **t_g:** tiempo de programación en la iteración g.
- **A_0:** A_g es el conjunto de actividades que están activas en el instante t_g.
- **T_0:** T_g representa el conjunto de tiempos finales de actividades calculadas para la iteración g.
- **S_0:** S_g representa el conjunto de actividades que han sido colocadas en la programación en la iteración g.
- E_g: El conjunto de todas aquellas actividades que pueden ser programadas en esa iteración g. $\boldsymbol{E_g = \{j \in J \setminus S_{g-1} \mid F_i \leq t_g + tdelay_g\}}$ $for\ i \in P_j$;

$\boldsymbol{for\ \ k \epsilon K\ \ do\ \ \{initialize\ RD_k(0) = R_k(k \epsilon K)\};}$

Para todos los recursos tipo que pertenecen al conjunto de recursos, se debe inicializar en el instante 0 cada una de las cantidades de recursos tipo en un vector de recursos disponibles (RD_k). Este valor se irá consumiendo y liberando en función de la actividad y sus cantidades de recursos tipo.

$\boldsymbol{while\ \ |S_g| < n + 2\ \ do\ \ \{}$

Bucle while. Mientras que el tamaño del vector S_g (debiendo ser siempre mayor que -1) sea menor el número de actividades (n) + 2 (siendo dos las actividades no utilizadas de inicio y fin), se sigue haciendo lo que está dentro del bucle.

$\boldsymbol{while\ \ E_g \neq \{\ \}\ \ do\ \ \{}$

Segundo bucle while interno. Mientras que el tamaño del vector E_g sea distinto del conjunto vacío, se sigue haciendo lo que está dentro del bucle.

$$\boldsymbol{j^* = argmax\{PRIORITY_j\}} \qquad j \in E_g;$$

El algoritmo escoge aquella actividad j que se encuentre dentro del conjunto E_g que tenga el mayor valor $PRIORITY_j$.

$$\boldsymbol{EF_{j^*} = \max\{F_i\} + d_{j^*}} \qquad i \in P_{j^*};$$

Conocida la actividad a programar (j*), se toma el valor máximo de todos aquellos tiempos finales de actividades (F_i) pertenecientes al conjunto de precedencias de dicha actividad (P_{j*}), más la duración de la actividad a programar (d_{j*}).

$$\boldsymbol{F_{j^*} = \min\left\{t \in \left[(EF_{j^*} - d_{j^*}), \infty\right] \bigcap T_{g-1} \mid r_{j^*,k} \leq RD_k(\tau), \quad k \in K \mid r_{j^*,k} \geq 0, \quad \tau \in \left[t, (t + d_{j^*})\right]\right\} + d_{j^*}};$$

Para calcular el tiempo final de la actividad a programar, se debe tomar el mínimo valor t de la intersección entre el conjunto $\left[(EF_{j^*} - d_{j^*}), \infty\right]$ y el conjunto T_{g-1}. De tal forma que se cumpla en todo momento, que los recursos tipo exigidos por dicha actividad sean menores que los recursos tipo disponibles en todo el intervalo de tiempo, desde el valor t hasta el valor $(t + d_{j^*})$, más la duración de la actividad a programar (d_{j*}).

$$\boldsymbol{S_g = S_{g-1} \bigcup \{j^*\}} \quad (\boldsymbol{Update})$$

Se actualiza el conjunto de actividades que han sido colocadas en la programación en la iteración g.

$$\boldsymbol{T_g = T_{g-1} \bigcup \{F_{j^*}\}} \quad (\boldsymbol{Update});$$

Se actualiza el conjunto de tiempos finales de actividades calculadas para la iteración g.

$$\boldsymbol{g = g + 1};$$

Actualización de la iteración g.

$$\boldsymbol{A_g = \{j \in J \mid F_j - d_j \leq t_g < F_j\}} \quad (\boldsymbol{Update});$$

Se actualiza el conjunto de actividades que están activas en el nuevo instante t_g. Para saber cuáles están activas, se observan todas las actividades que pertenecen al conjunto de

actividades. Se comprueba la condición $\boldsymbol{F_j - d_j \leq t_g < F_j}$, de modo que si el cálculo es satisfactorio, se incluye dentro del conjunto A_g dicha actividad. Cada vez que llegamos a este punto del algoritmo, se realiza una limpieza del conjunto para que esté vacío.

$$\boldsymbol{E_g = \{j \in J \setminus S_{g-1} \mid F_i \leq t_g + tdelay_g\}} \quad for\ i \in P_j\ ;$$

Para calcular el conjunto E_g se observan todas las actividades pertenecientes al conjunto de actividades que no se encuentren en el conjunto S_{g-1}, de tal forma que debe satisfacer la ecuación $\boldsymbol{F_i \leq t_g + tdelay_g}$ para todas las F_i que se encuentren dentro del conjunto de precedencias de cada actividad (P_j). En caso afirmativo, se incluye dicha actividad en el conjunto E_g.

$$\boldsymbol{RD_k(t) = R_k(t_g) - \sum_{j \in A_g} r_{j^*,k}} \mid t \in \left[\left(F_{j^*} - d_{j^*}\right), F_{j^*}\right], k \in K \mid r_{j^*,k} \geq 0 \quad (\boldsymbol{Update})\ ;$$

Se actualiza $\boldsymbol{RD_k(t) = R_k(t_g) - \sum_{j \in A_g} r_{j^*,k}}$ para el intervalo de tiempo $\left[\left(F_{j^*} - d_{j^*}\right), F_{j^*}\right]$

$$\boldsymbol{t_g = min\{t \in T_{g-1} \mid t > t_{g-1}\}}\ ;$$

Una vez que el conjunto E_g sea el conjunto vacío, se actualizará el instante de tiempo de la iteración actual. Para ello se tomará el mínimo valor de t que pertenezca al conjunto $\boldsymbol{T_{g-1}}$, de tal forma que satisfaga la inecuación $\boldsymbol{t > t_{g-1}}$.

Cuando se deje de cumplir la condición del primer bucle while, se dará por finalizado la programación, puesto que todas las actividades habrán sido programadas.

2.2.4 TRAZA DEL ALGORITMO

Definido el algoritmo, se explicará la traza para un proyecto de 5 actividades y 2 recursos tipo, como se muestra en las Fig. 2.2 y Fig. 2.3. Para ello, se inicializará con los siguientes Datos de Entrada, cuyo vector cromosoma (PRIORITY y tdelay) es el resultado del algoritmo genético que se explicará más adelante.

```
======================================================
                Inicializamos Datos de Entrada
======================================================

Matriz M (j, PRIORITY, tdelay, duraciones):
0 1 2 3 4 5 6
0 0.62 0.2 0.7 0.72 0.3 0
0 1.98 2.31 3.96 4.13 8.05 0
0 1 1 1 1 1 0

Matriz P (Precedentes):
0 -1 -1 -1
0 -1 -1 -1
1 -1 -1 -1
3 -1 -1 -1
4 -1 -1 -1
2 5 -1 -1

Dimension F: 7
F[] = 0 20000 20000 20000 20000 20000 20000
Dimension F1: 1
```

F[] = 0

Matriz RD (Recursos disponibles):

2 2 2 2 2 2

2 2 2 2 2 2

Matriz K (Cantidad de recursos tipo por actividad):

2 2

1 1

2 2

2 2

1 1

==========

Calculamos Calendario de Actividades

==========

---

j* = 1

EF = 1

Vector interseccion: 0

Comprobamos F = 0 1 20000 20000 20000 20000 20000

Comprobamos F1 = 0 1

F[1] = 1

Sg[] = 0 1

Tg[] = 0 1

g = 2

Vector tg[] = 0 0 0

t[g] = 0

Ag[] = 0 1

Eg[] = 2 3

0 2 2 2 2 2

0 2 2 2 2 2

---

j* = 3

EF = 2

Vector interseccion: 1

Comprobamos F = 0 1 20000 2 20000 20000 20000

Comprobamos F1 = 0 1 2

F[3] = 2

Sg[] = 0 1 3

Tg[] = 0 1 2

g = 3

Vector tg[] = 0 0 0 0

t[g] = 0

Ag[] = 0 1

Eg[] = 2 4

0 0 2 2 2 2

0 0 2 2 2 2

---

j* = 4

EF = 3

Vector interseccion: 2

Comprobamos F = 0 1 20000 2 3 20000 20000

Comprobamos F1 = 0 1 2 3

F[4] = 3

Sg[] = 0 1 3 4

Tg[] = 0 1 2 3

g = 4

Vector tg[] = 0 0 0 0 0

t[g] = 0

Ag[] = 0 1

Eg[] = 2 5

0 0 0 2 2 2

0 0 0 2 2 2

---
j* = 5

EF = 4

Vector interseccion: 3

Comprobamos F = 0 1 20000 2 3 4 20000

Comprobamos F1 = 0 1 2 3 4

F[5] = 4

Sg[] = 0 1 3 4 5

Tg[] = 0 1 2 3 4

g = 5

Vector tg[] = 0 0 0 0 0 0

t[g] = 0

Ag[] = 0 1

Eg[] = 2

0 0 0 1 2 2

0 0 0 1 2 2

---
j* = 2

EF = 1

Vector interseccion: 0 1 2 3 4

Comprobamos F = 0 1 4 2 3 4 20000

Comprobamos F1 = 0 1 2 3 4 4

F[2] = 4

Sg[] = 0 1 2 3 4 5

Tg[] = 0 1 2 3 4 4

g = 6

Vector tg[] = 0 0 0 0 0 0 0

t[g] = 0

Ag[] = 0 1

Eg[] =

0 0 0 0 2 2

0 0 0 0 2 2

t[g] = 0 0 0 0 0 0 1

t[g] = 1
t[g] = 0 0 0 0 0 0 2

t[g] = 2
t[g] = 0 0 0 0 0 0 3

t[g] = 3
t[g] = 0 0 0 0 0 0 4

t[g] = 4

Eg[] = 6

Dimension E = 1

---

j* = 6

EF = 4

Vector interseccion: 4

Comprobamos F = 0 1 4 2 3 4 4

Comprobamos F1 = 0 1 2 3 4 4 4

F[6] = 4

Sg[] = 0 1 2 3 4 5 6

Tg[] = 0 1 2 3 4 4 4

g = 7

Vector tg[] = 0 0 0 0 0 0 4 4

t[g] = 4

Ag[] =

Eg[] =

0 0 0 0 2 2

0 0 0 0 2 2

==========

Resultados

==========

Cmax = 4

Valor de F[]

F[] = 0 1 4 2 3 4 4

==========

==========

2.3 ALGORITMO GENÉTICO

Los algoritmos genéticos son métodos adaptativos que se usan para resolver problemas de búsqueda y optimización. Es importante que exista una buena codificación del problema antes de ejecutarse. Se requiere una función objetivo que asigne un valor (calidad) para cada solución codificada. Durante la ejecución del algoritmo, los padres deben ser seleccionados para la reproducción, y luego recombinados para generar la descendencia.

Se supone que una posible solución a un problema puede representarse como un conjunto de parámetros. Esos parámetros, conocidos como genes, se unen entre sí para formar una cadena de valores, llamada cromosoma. Al conjunto de parámetros representados por un cromosoma se le denomina un individuo (la programación). La aptitud de un individuo depende del cromosoma y es evaluado por la función objetivo.

Los individuos, durante su fase reproductiva, son seleccionados de la población y recombinados, produciendo la descendencia, siendo por tanto la siguiente generación. Los padres son seleccionados de forma aleatoria en la población usando un esquema por torneo, que favorece al individuo más apto. Habiendo seleccionado dos padres, sus cromosomas son recombinados usando mecanismos de cruce y mutación. En general, la mutación se aplica en algunos individuos para garantizar la diversidad de la población.

2.3.1 REPRESENTACIÓN DEL CROMOSOMA

El algoritmo genético seleccionado para este trabajo, usa un alfabeto random-key que se compone de números reales aleatorios entre 0 y 1 para la primera parte del cromosoma (vector PRIORITY) y para la segunda parte (tdelay), valor mínimo y máximo de tiempos de retraso.

Se ha utilizado una codificación binaria de 8 bits de memoria para el vector cromosoma del algoritmo genético.

En una representación directa, un cromosoma representa una solución del problema original, mientras que en una representación indirecta no es así y se necesitan procedimientos especiales para obtener una solución, llamada fenotipo.

Cada solución del cromosoma tiene una dimensión de 2n genes, donde n es el número de actividades.

$$Cromosoma = (\underbrace{gen_1, \ldots, gen_n}_{Prioridades}, \underbrace{gen_{n+1}, \ldots, gen_{2n}}_{Tiempos\ de\ retraso})$$

Los primeros n genes se usan para determinar la prioridad de cada una de las actividades. Los genes comprendidos entre n + 1 y 2n, se usan para determinar los tiempos de retraso usados en cada una de las n iteraciones del procedimiento de programación.

2.3.2 ESTRATEGIA EVOLUTIVA

Para generar buenas soluciones, el vector random-key de población es operado por un algoritmo genético. Variaciones del algoritmo genético se obtienen por elecciones en sus operadores de reproducción, cruce y mutación. El operador de selección determina qué padres tendrán descendencia (selección por torneo en este caso), y cómo el material genético se intercambia entre los padres para crear la descendencia queda definido por el operador de cruce (cruce en un punto). Por otro lado, la mutación permite introducir diversidad genética en el proceso, incentivando la exploración del espacio de búsqueda (Probabilidad de mutación bit por bit – FlipMutation).

En este documento, se usó la librería GALib de código abierto.

2.4 FUNCIÓN OBJETIVO

Diferentes programaciones pueden dar mismos valores del tiempo de ejecución del proyecto (Cmax). Sin embargo, el potencial de mejora de las programaciones con el mismo tiempo de ejecución del proyecto pueden no ser los mismos. Para diferenciar el potencial de mejora entre las programaciones que tengan el mismo tiempo de ejecución del proyecto se implementa una nueva medida de aptitud, llamado tiempo de ejecución del proyecto modificado (MM).

El tiempo de ejecución del proyecto modificado combina el tiempo de ejecución del proyecto de la programación con una medida de potencial de mejora de la programación. Esta medida de potencial de mejora tendrá valores entre 0 y 1. La justificación de esta nueva medida es si se tienen dos programaciones con el mismo tiempo de ejecución del proyecto, la que tenga el menor número de actividades terminadas cerca del valor del tiempo de ejecución del proyecto tendrá más posibilidades de mejora.

Para definir el tiempo de ejecución del proyecto modificado, se introduce el concepto de la distancia de una actividad a la actividad final (n + 1). La distancia dist(j) es igual al número de actividades que existen por el camino más corto desde la actividad j hasta la actividad n + 1.

El tiempo de ejecución del proyecto modificado de distancia L se expresa de la siguiente forma:

$$MM = F_{n+1} + \frac{\sum_{d=1}^{L}\left[\sum_{i|dist(i)=d} F_i\right]}{\sum_{d=1}^{L}\left[\sum_{i|dist(i)=d} F_{n+1}\right]}$$

Cuando L = 0, el tiempo de ejecución del proyecto modificado es el propio tiempo de ejecución del proyecto. Siendo L el valor de la distancia, parámetro de libre elección que establece la definición de la función objetivo (MM).

3 RESULTADOS

En este apartado se exponen los diferentes resultados obtenidos de este estudio. Para poder obtener diferentes resultados y compararlos entre sí, se han modificado los siguientes parámetros:

- Número de actividades (j).
- Número de recursos tipo (k).
- Cantidad máxima de recursos tipo (R_k).
- Valor de la distancia (L).
- Tiempos de retraso min. y max. (tdelay).
- Tamaño de la población.
- Probabilidad de cruce.
- Probabilidad de mutación.
- Número de generaciones.

Los problemas han sido ejecutados en el clúster del SIANI.

3.1 CASO 1

3.1.1 DESCRIPCIÓN

El primer caso está compuesto de 5 actividades y 2 recursos tipo renovables. Cada recurso tipo tiene disponible 2 unidades de recursos. En la figura 3.1 se muestra el esquema de precedencias de las actividades. Las actividades 0 y 6 son actividades de inicio y fin del proyecto.

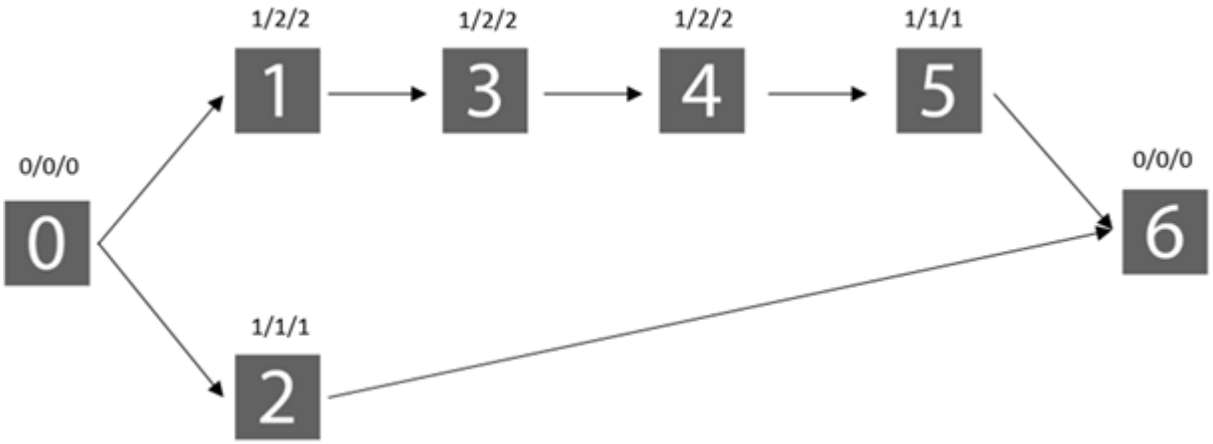

Fig. 3.1 Red Proyecto de 5 actividades y 2 recursos tipo.

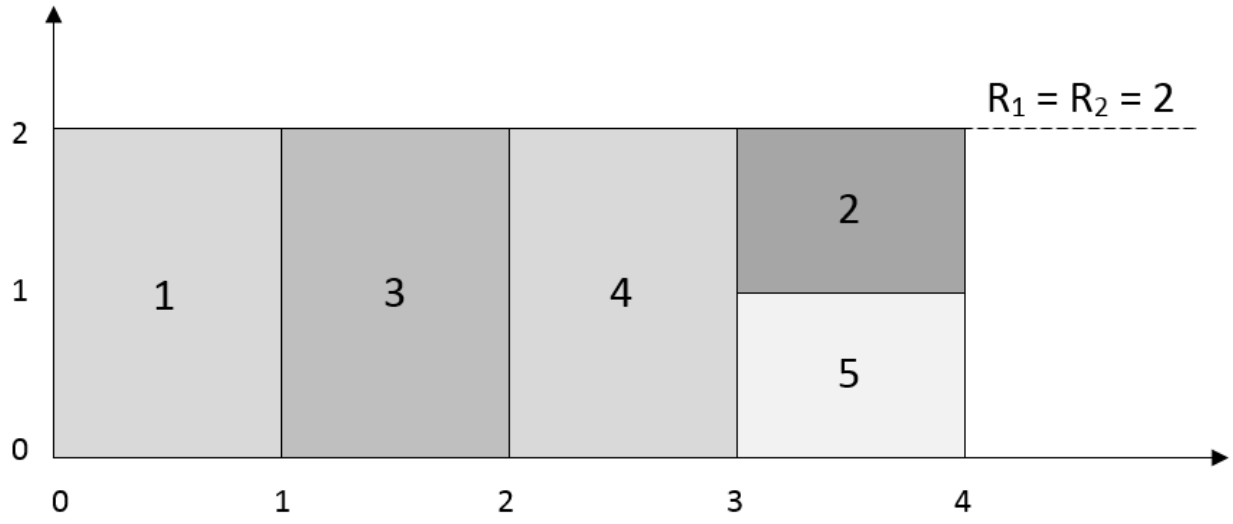

Fig. 3.2 Programación con un tiempo de ejecución del proyecto óptima (Cmax – F_{n+1}) de 4 para la red de Proyecto de la Fig. 3.1.

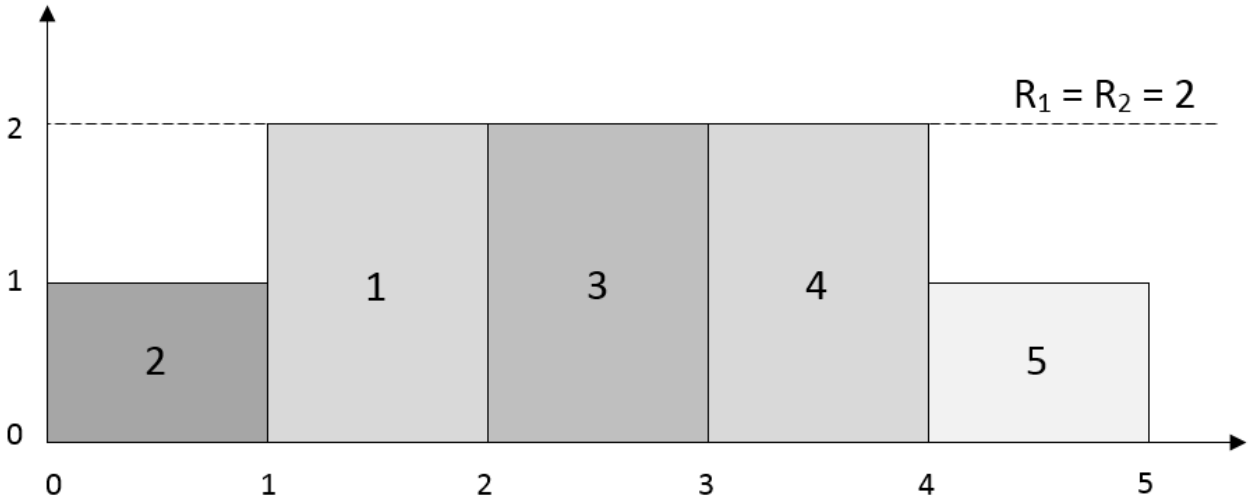

Fig. 3.3 Programación con un tiempo de ejecución del proyecto no óptima (Cmax – F_{n+1}) de 5 para la red de Proyecto de la Fig. 3.1.

Este primer caso tiene dos soluciones. La solución óptima, Fig. 3.2, tiene un tiempo de ejecución de proyecto (Cmax – F_{n+1}) de 4. Se caracteriza porque las actividades 2 y 5 se realizan de forma simultánea, lo que reduce el valor del tiempo de ejecución del proyecto en 1 unidad de tiempo. Sin embargo, en algunos lanzamientos que se han hecho al algoritmo para buscar el cromosoma óptimo, resultó ser que no fueron eficientes, siendo la Fig. 3.3 el resultado de una optimización poco precisa, dando un valor del tiempo de ejecución del proyecto de 5 unidades de tiempo.

3.1.2 PARÁMETROS DEL ALGORITMO GENÉTICO Y DE LA PROGRAMACIÓN

A continuación se describen los parámetros que se han utilizado:

- **j** = 5.
- **k** = 2.
- **R_k** = {2,2}.
- **L** = 1.
- **tdelay** = [5,00 – 9,00]
- **Población** = 3
- **Prob. Cruce** = 0,90.
- **Prob. Mutación** = 0,02.
- **Nº Generaciones** = 30.

Los valores de los parámetros de probabilidad de cruce, mutación y el valor de tdelay son recomendaciones de Dr. Mendes [1].

3.1.3 RESUMEN DE RESULTADOS

Definidos los parámetros de entrada, se observan los siguientes resultados en la Fig. 3.4. Nótese que los resultados serán mostrados en términos de la función objetivo (MM) de manera que, haciendo mención a la fórmula del tiempo de ejecución del proyecto modificado (MM), el primer sumando corresponde al valor Cmax – F_{n+1} y el segundo sumando al término que depende de las F_i de las actividades 2 y 5. En este caso, ese término adicional vale 1 y por eso la MM óptima, correspondiente a una duración Cmax - F_{n+1} = 4, es MM = 5.

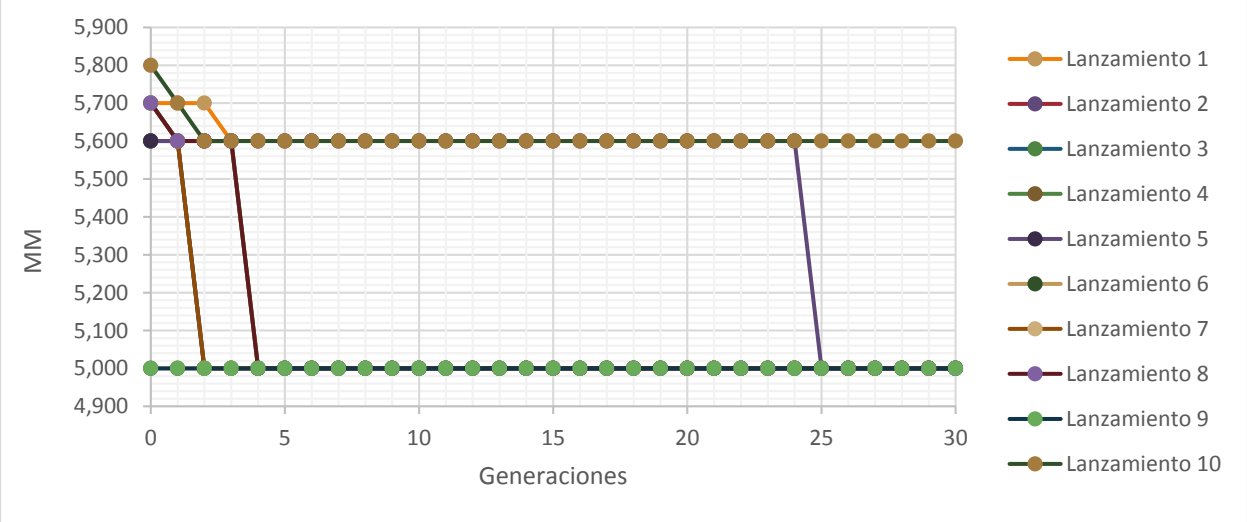

Fig. 3.4 Valor de MM por cada generación para cada lanzamiento.

En la Fig 3.4 se muestra los valores del tiempo de ejecución del proyecto modificado (MM) para cada generación. Se observa que algunos lanzamientos no consiguen encontrar la solución óptima. Se necesitan más generaciones para encontrar la solución óptima.

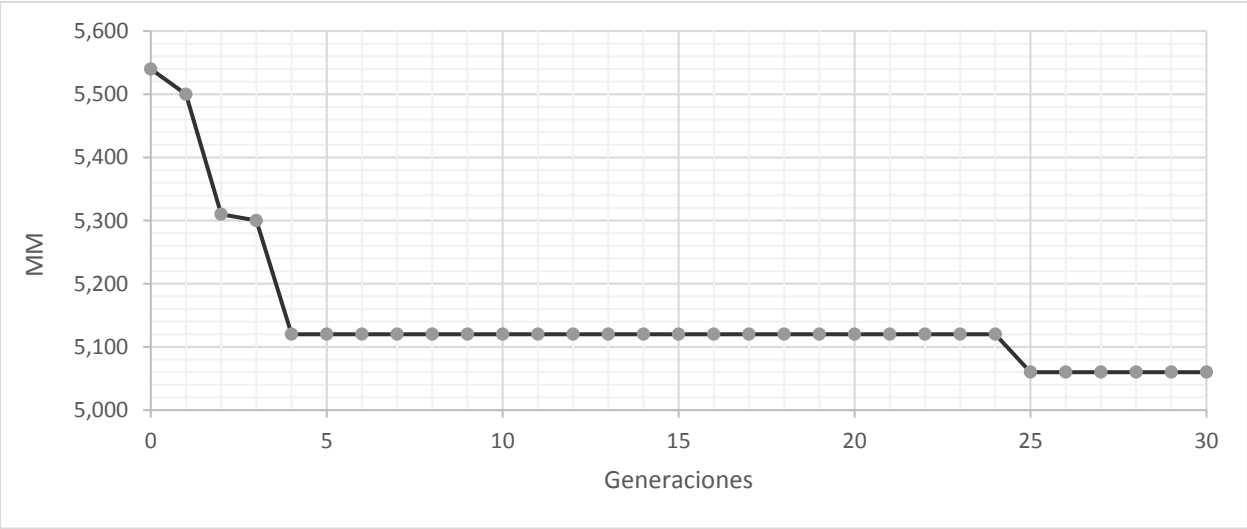

Fig. 3.5 Valor medio de MM por cada generación.

Haciendo una media de cada valor del tiempo de ejecución del proyecto modificado con su respectiva generación (Fig. 3.4), se obtiene el valor medio del tiempo de ejecución modificado (MM) según la Fig 3.5.

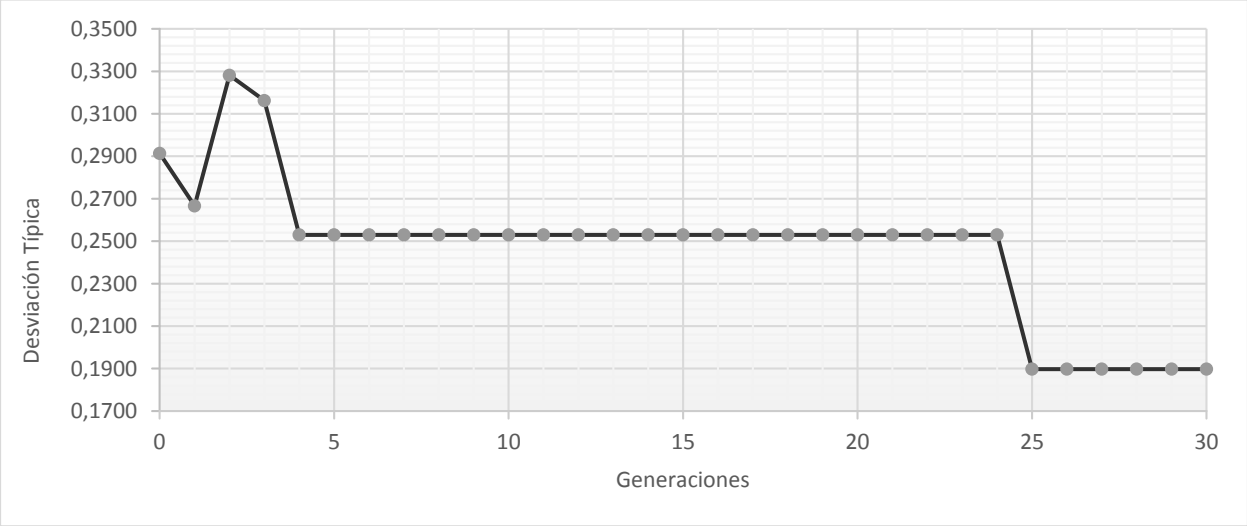

Fig. 3.6 Desviación Típica por cada generación.

En la Fig 3.6 se representa la desviación típica. En este caso, el valor disminuye a medida que se aumenta el número de generaciones, lo que hace aproximarse a su media aritmética.

3.1.4 MEJOR CROMOSOMA POR CADA LANZAMIENTO

En este apartado es importante conocer no solamente el resultado del mejor cromosoma de cada lanzamiento, sino saber si las soluciones de los mejores cromosomas obtienen la misma solución de programación. Para este caso se observa en la Fig. 3.4 que el lanzamiento 10 obtiene un valor diferente y de peor calidad que el resto de lanzamientos, cuyos valores han sido los óptimos.

A continuación se muestra una tabla con los diferentes resultados del mejor cromosoma en cada lanzamiento, dividiéndose el vector cromosoma de dos partes: PRIORITY (prioridades de las actividades) y tdelay (tiempo de retraso de las actividades).

CROMOSOMA										
	PRIORITY					tdelay				
L1	0.60	0.19	0.88	0.41	0.11	6.00	6.44	8.80	5.00	5.25
L2	0.96	0.71	0.81	0.76	0.92	7.18	8.67	5.74	7.79	8.17
L3	0.64	0.40	0.87	0.54	0.93	5.22	6.51	6.71	5.69	6.19
L4	0.28	0.18	0.58	0.66	0.19	8.67	7.53	8.78	7.10	7.92
L5	0.68	0.66	0.8	0.68	0.26	7.56	7.54	6.15	6.57	5.58
L6	0.14	0.12	0.18	0.88	0.75	5.20	8.42	7.64	5.64	5.82
L7	0.45	0.05	0.17	0.08	0.53	8.40	5.89	5.56	6.25	7.96
L8	0.44	0.32	1.00	0.41	0.14	8.18	6.87	8.00	8.50	6.76
L9	0.76	0.61	0.89	0.71	0.47	7.20	6.18	5.55	6.52	5.00
L10	0.37	1.00	0.20	0.36	0.33	5.14	8.23	5.74	6.51	8.28

Tabla 3.1 Vector cromosoma – Caso 1.

3.2 CASO 2

3.2.1 DESCRIPCIÓN

El segundo caso está compuesto de 6 actividades y 2 recursos tipo renovables. Cada recurso tipo tiene disponibles 4 y 2 unidades de recursos respectivamente. En la figura 3.7 se muestra el esquema de precedencias de las actividades. Las actividades 0 y 7 son actividades de inicio y fin del proyecto.

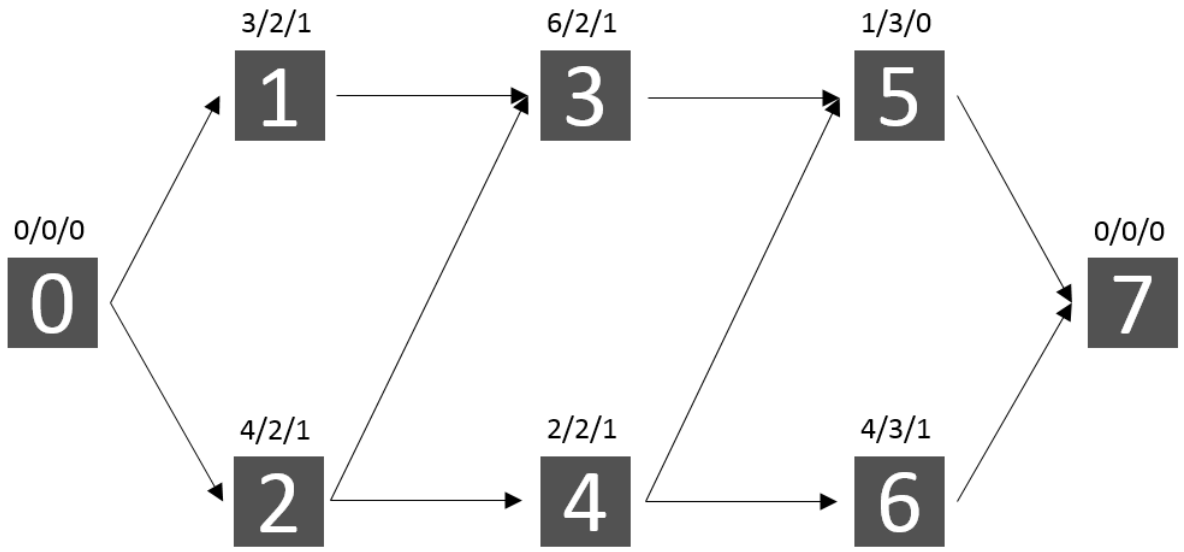

Fig. 3.7 Red Proyecto de 6 actividades y 2 recursos tipo.

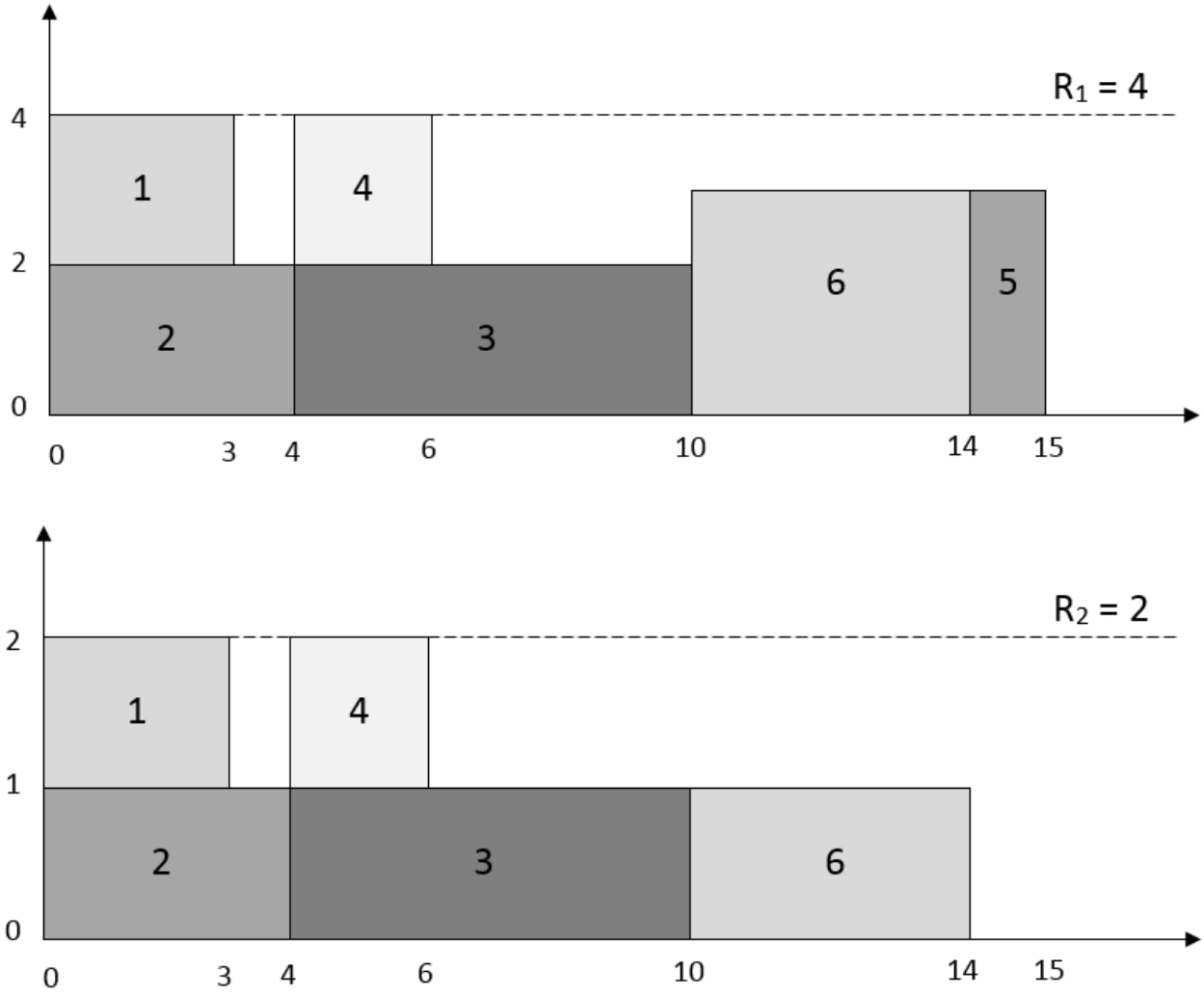

Fig. 3.8 Programación con un tiempo de ejecución del proyecto óptimo (Cmax – F_{n+1}) de 15 para la red de Proyecto de la Fig. 3.7.

Al ser dos recursos tipo con diferentes cantidades disponibles y que algunas actividades no necesitan de un recurso tipo, es necesario separarlos en dos diagramas. En este caso, se observó como el tiempo de ejecución del proyecto óptimo (Cmax – F_{n+1}) para todos los lanzamientos fue de 15 unidades de tiempo. El esquema refleja la colocación óptima de las actividades en la línea de tiempo.

3.2.2 PARÁMETROS DEL ALGORITMO GENÉTICO Y DE LA PROGRAMACIÓN

A continuación se describen los parámetros que se han utilizado:

- **j** = 6.
- **k** = 2.
- $\mathbf{R_k}$ = {4,2}.
- **L** = 1.
- **tdelay** = [5,00 – 9,00]
- **Población** = 3
- **Prob. Cruce** = 0,90.
- **Prob. Mutación** = 0,02.
- **Nº Generaciones** = 30.

Los valores de los parámetros de probabilidad de cruce, mutación y el valor de tdelay son recomendaciones de Dr. Mendes [1].

3.2.3 RESUMEN DE RESULTADOS

Definidos los parámetros de entrada, se observan los siguientes resultados en la Fig. 3.9. Nótese que los resultados serán mostrados en términos de la función objetivo (MM) de manera que, haciendo mención a la fórmula del tiempo de ejecución del proyecto modificado (MM), el primer sumando corresponde al valor Cmax – F_{n+1} y el segundo sumando al término que depende de las F_i de las actividades. En este caso, ese término adicional vale 0,97 y por eso la MM óptima, correspondiente a una duración Cmax - F_{n+1} = 15, es MM = 15,97.

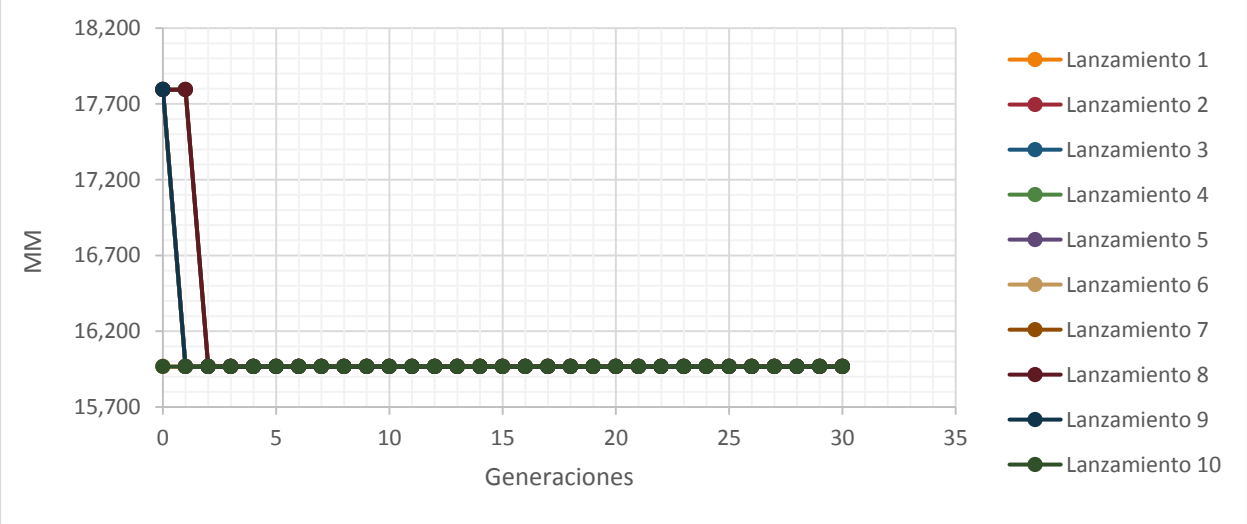

Fig. 3.9 Valor de MM por cada generación para cada lanzamiento.

En la Fig 3.9 se observa que todos lanzamientos consiguen encontrar la solución óptima, siendo rápida su convergencia con un número bajo de generaciones.

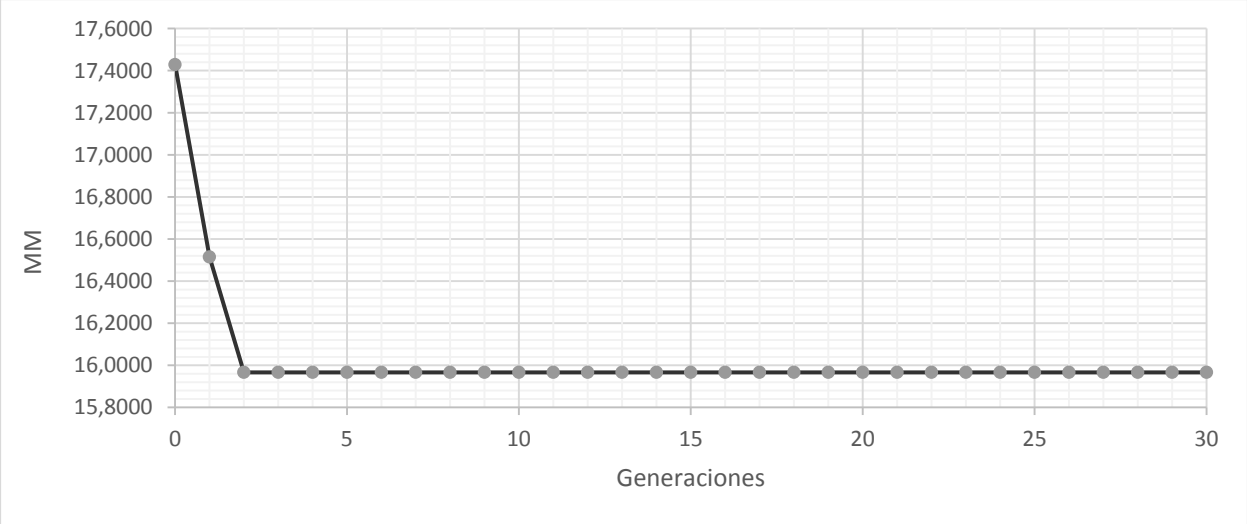

Fig. 3.10 Valor medio de MM por cada generación.

Haciendo una media de cada valor del tiempo de ejecución del proyecto modificado (MM) con su respectiva generación (Fig. 3.9), se obtiene el valor medio del tiempo de ejecución modificado (MM) según la Fig 3.10.

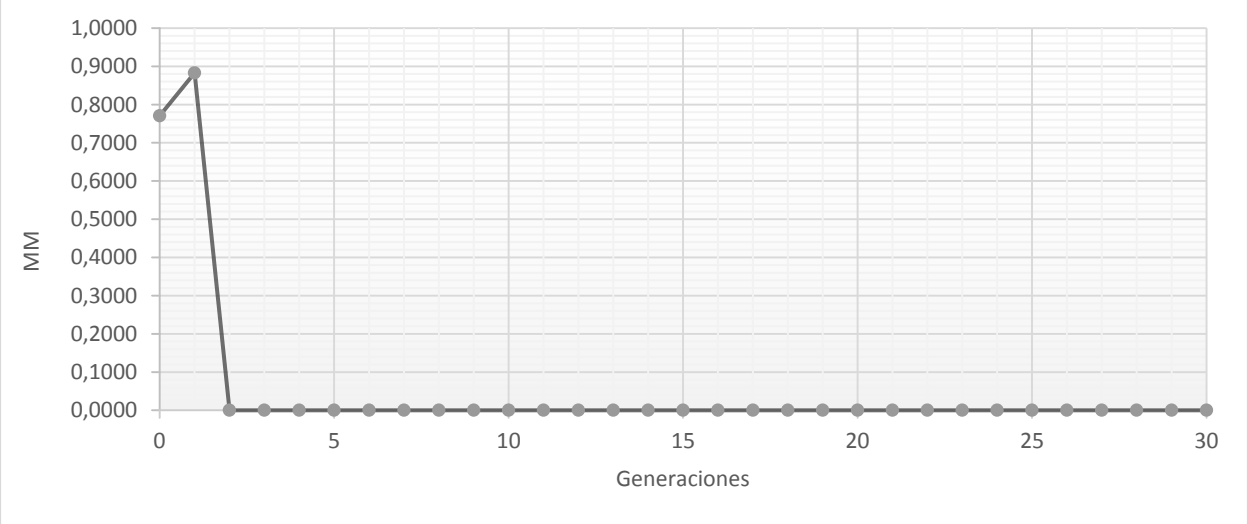

Fig. 3.11 Desviación Típica por cada generación.

En la Fig 3.11 se representa la desviación típica. En este caso, el valor disminuye a hasta alcanzar el valor 0 en la generación 3. En este punto quiere decir que la desviación típica coincide con su media aritmética.

3.2.4 MEJOR CROMOSOMA POR CADA LANZAMIENTO

En este caso se observa que todos los lanzamientos obtienen el valor óptimo de programación. El motivo por el cual se obtienen los mismos resultados, es debido a que el problema en cuestión es bastante sencillo, no teniendo un número significativo de alternativas de programación, ya que con solo 3 valores de población y pocas generaciones encuentran el óptimo.

A continuación se muestra una tabla con los diferentes resultados del mejor cromosoma en cada lanzamiento, dividiéndose el vector cromosoma de dos partes: PRIORITY (prioridades de las actividades) y tdelay (tiempo de retraso de las actividades).

CROMOSOMA												
	PRIORITY						tdelay					
L1	0.77	0.96	0.67	0.19	0.91	0.96	8.15	6.07	5.38	8.07	5.78	5.85
L2	0.53	0.74	0.70	0.15	0.72	0.26	7.37	5.91	7.42	8.22	5.13	7.49
L3	0.63	0.80	0.74	0.66	0.32	0.36	8.34	5.56	7.71	7.12	6.85	6.84
L4	0.75	0.49	0.02	0.58	0.53	0.49	6.84	8.86	8.56	5.75	5.52	5.09
L5	0.41	0.78	0.34	0.43	0.37	0.21	7.32	6.29	6.07	5.66	6.73	7.31
L6	0.31	0.84	0.21	0.81	0.96	0.08	6.98	5.67	5.06	8.62	8.98	6.55
L7	0.61	0.79	0.88	0.13	0.55	0.45	5.05	8.40	5.20	6.95	8.69	6.68
L8	0.69	0.72	0.33	0.99	0.55	0.45	5.55	5.09	6.22	5.71	5.55	7.64
L9	0.43	0.05	0.30	0.21	0.30	0.32	6.71	8.91	8.12	5.08	8.15	6.55
L10	0.79	0.87	0.79	0.98	0.53	0.14	7.79	5.36	7.70	7.02	5.83	6.54

Tabla 3.2 Vector cromosoma – Caso 2.

3.3 CASO 3

3.3.1 DESCRIPCIÓN

El último caso está compuesto de 30 actividades y 3 recursos tipo renovables. Cada recurso tipo tiene disponibles 6, 4 y 6 unidades de recursos respectivamente. En la figura 3.12 se muestra el esquema de precedencias de las actividades. Las actividades 0 y 31 son actividades de inicio y fin del proyecto.

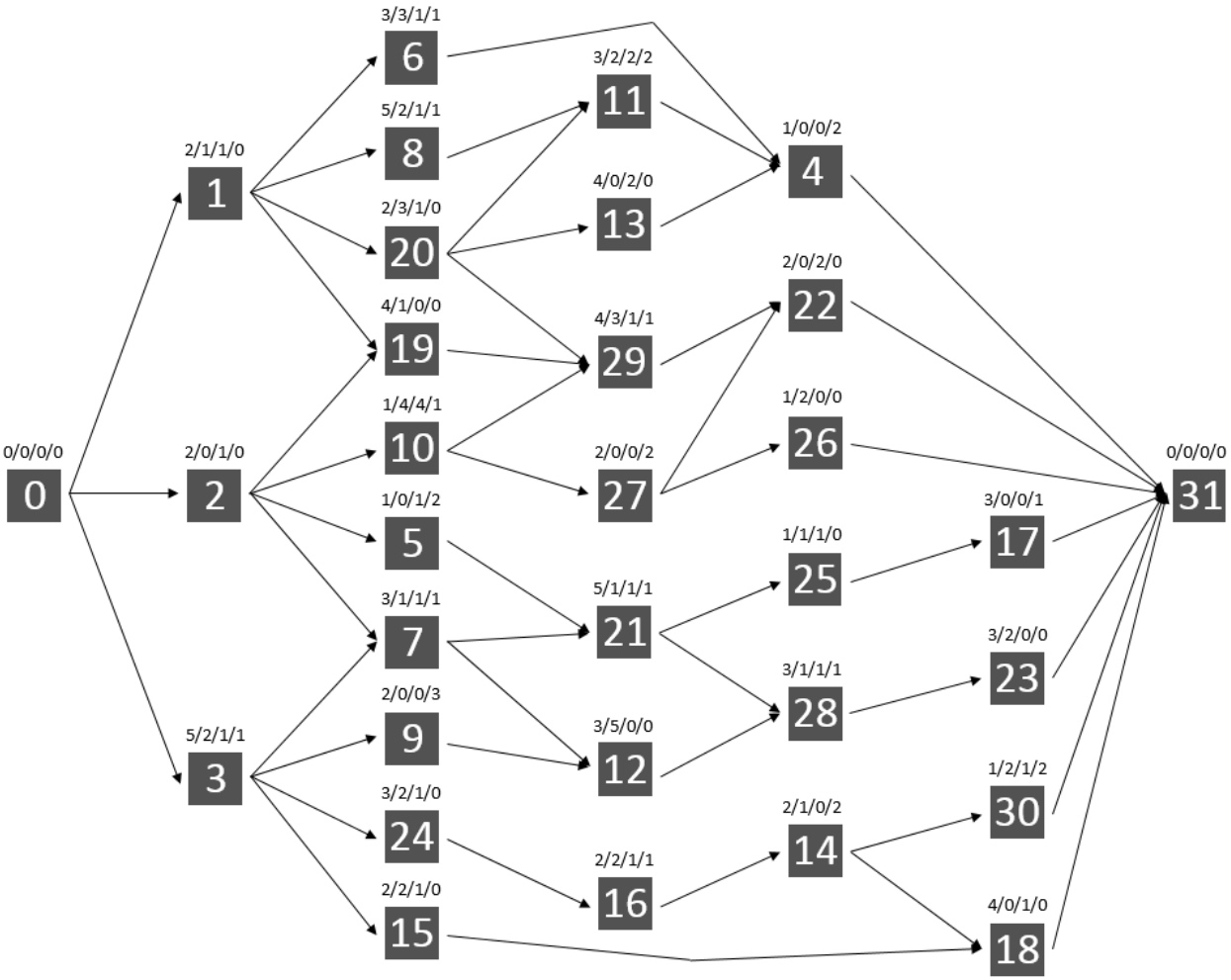

Fig. 3.12 Red Proyecto de 30 actividades y 3 recursos tipo.

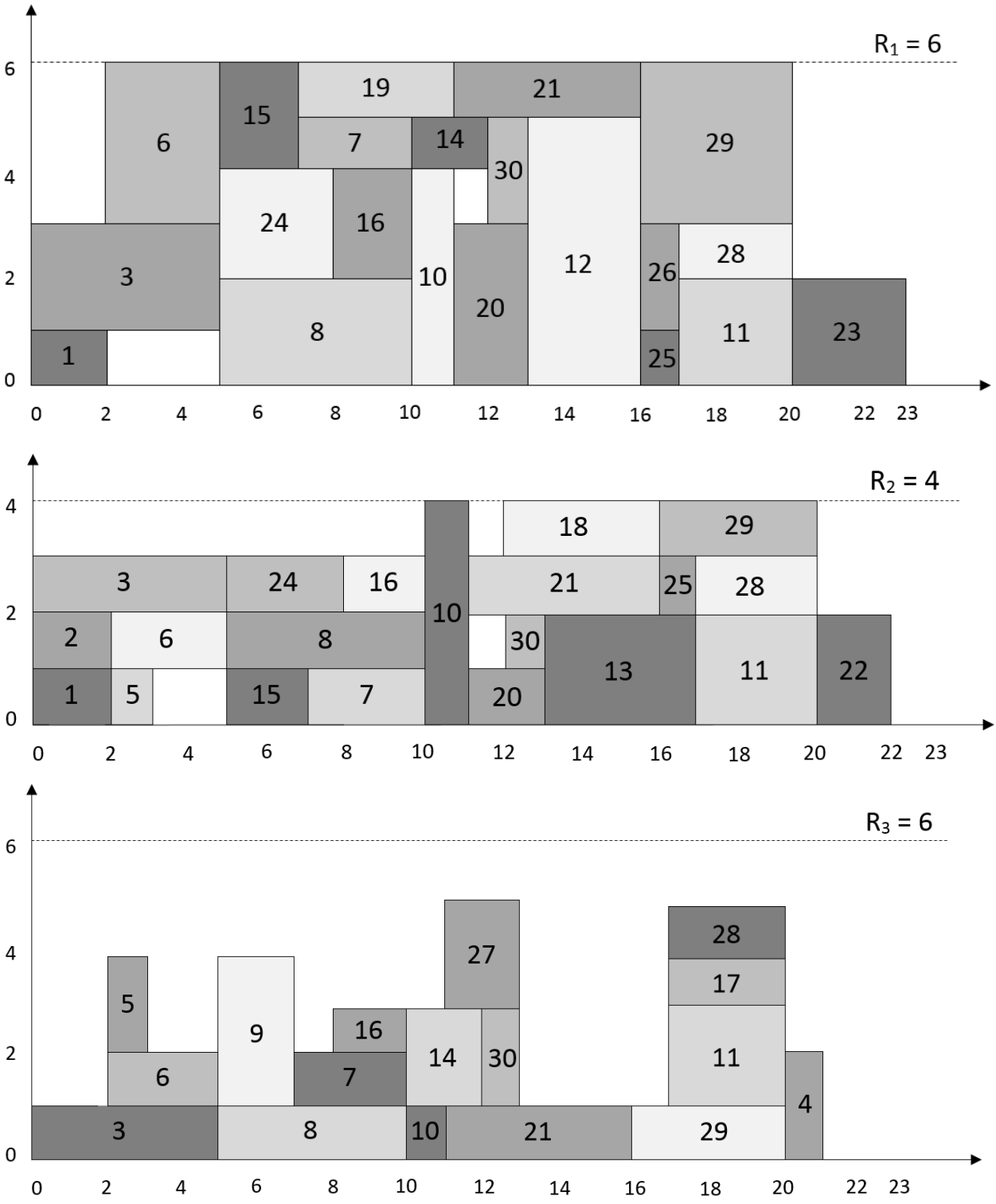

Fig. 3.13 Programación con una tiempo de ejecución óptimo (Cmax – F_{n+1}) de 23 para la red de Proyecto de la Fig. 3.10.

Al ser tres recursos tipo con diferentes cantidades disponibles y que algunas actividades no necesitan de un recurso tipo, es necesario separarlos en tres diagramas. En este caso, se observó como el tiempo de ejecución del proyecto óptimo (Cmax – F_{n+1}) para todos los lanzamientos fue de 23 unidades de tiempo. El esquema refleja la colocación óptima de las actividades en la línea de tiempo.

3.3.2 PARÁMETROS DEL ALGORITMO GENÉTICO Y DE LA PROGRAMACIÓN

A continuación se describen los parámetros que se han utilizado:

- **j** = 30.
- **k** = 3.
- $\mathbf{R_k}$ = {6,4,6}.
- **L** = 1.
- **tdelay** = [5,00 – 9,00]
- **Población** = 15
- **Prob. Cruce** = 0,90.
- **Prob. Mutación** = 0,02.
- **Nº Generaciones** = 150.

Los valores de los parámetros de probabilidad de cruce, mutación y el valor de tdelay son recomendaciones de Dr. Mendes [1].

Debido a que la convergencia fue de poca calidad con baja población (de valor 3) y número de generaciones (de valor 30), se han incrementado los valores y se han obtenido los siguientes resultados descritos en el siguiente apartado.

3.3.3 RESUMEN DE RESULTADOS

Definidos los parámetros de entrada, se observan los siguientes resultados en la Fig. 3.14. Nótese que los resultados serán mostrados en términos de la función objetivo (MM) de manera que, haciendo mención a la fórmula del tiempo de ejecución del proyecto modificado (MM), el primer sumando corresponde al valor Cmax – Fn+1 y el segundo sumando al término que depende de las Fi de las actividades. En este caso, ese término adicional vale 0,84 y por eso la MM óptima, correspondiente a una duración Cmax - Fn+1 = 23, es MM = 23,84.

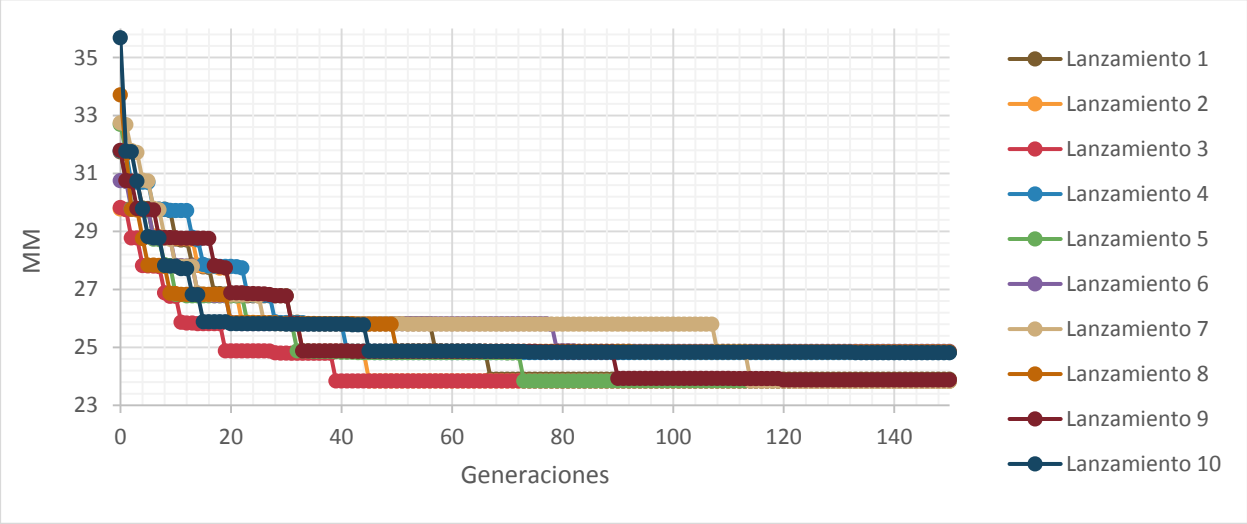

Fig. 3.14 Valor de MM por cada generación para cada lanzamiento.

En la Fig 3.14 se observa que algunos lanzamientos no consiguen encontrar la solución óptima. Se necesitan más generaciones para encontrar la solución óptima.

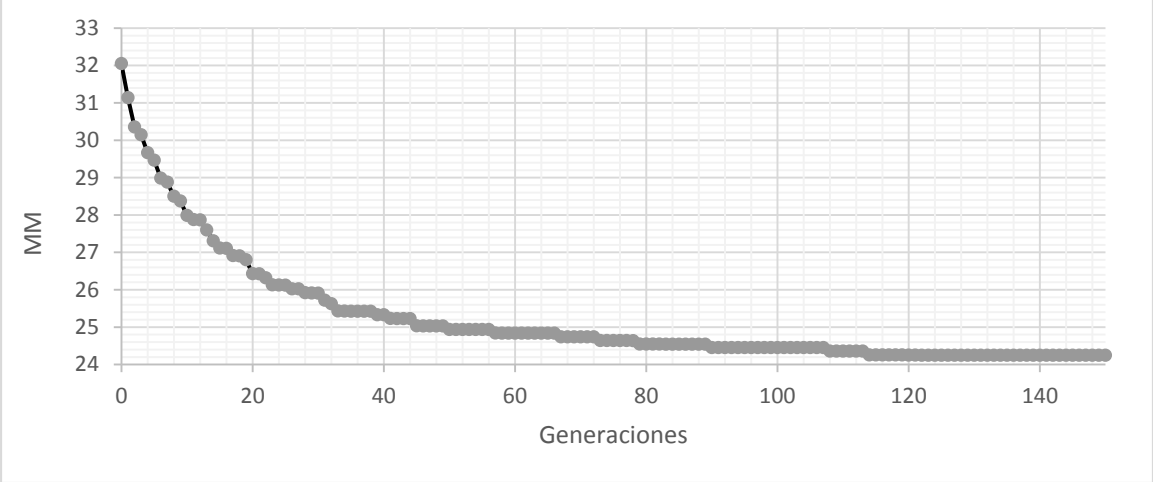

Fig. 3.15 Valor medio de MM por cada generación.

Haciendo una media de cada valor del tiempo de ejecución del proyecto modificado con su respectiva generación (Fig. 3.14), se obtiene el valor medio del tiempo de ejecución modificado (MM) según la Fig 3.15. Gracias a la media se puede observar como la convergencia es progresiva, siendo a partir de la generación 80 cuando empieza a descender muy lentamente, como si se estuviera aproximando a una asíntota horizontal. Este fenómeno revela que, aunque aumentemos el número de generaciones, no se consiguen mejorar los resultados, al haberse alcanzado la convergencia.

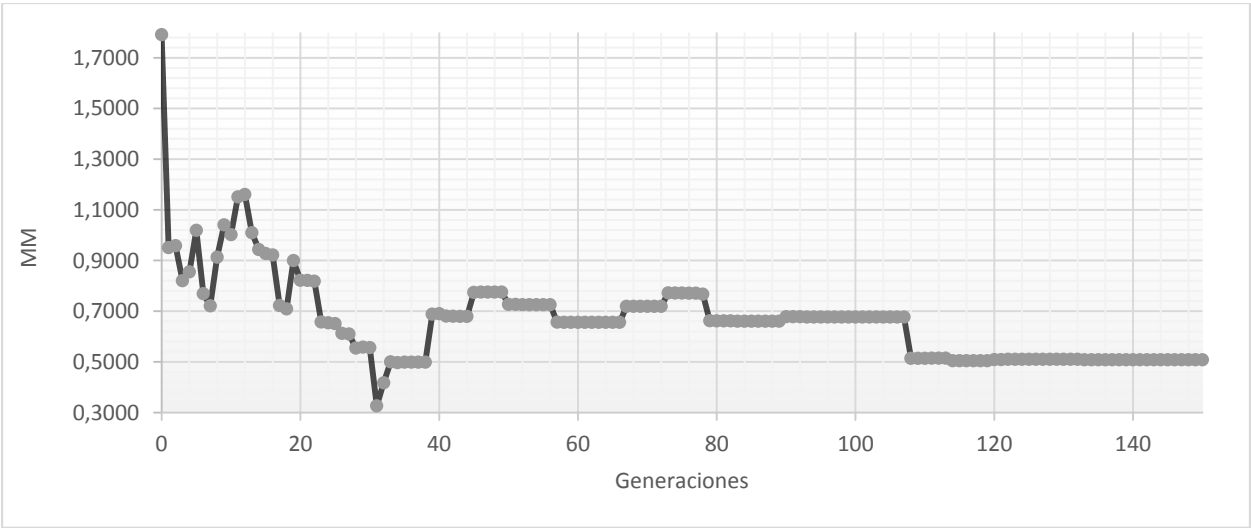

Fig. 3.16 Desviación Típica por cada generación.

En la Fig 3.16 se representa la desviación típica. En este caso, el valor disminuye hasta alcanzar el menor valor en la generación 32. Sin embargo, en las sucesivas generaciones se ve como la dispersión de los valores es más grande, volviendo a ser estabilizada a partir de la generación 108.

3.3.4 MEJOR CROMOSOMA POR CADA LANZAMIENTO

En este caso se observa que la gran mayoría de los lanzamientos obtienen el valor óptimo de programación. Esto es un dato importante, ya que se demuestra que esta metodología es eficaz y robusta para una programación de proyecto compleja. Aquellos proyectos cuya complejidad es simple, no es necesario usar este tipo de metodología porque existen pocas formas de colocar la programación de actividades.

Al ser el vector cromosoma con un número de genes elevado para su visualización en una tabla, se mostrará a continuación un resumen con los diferentes resultados del mejor cromosoma en cada lanzamiento, dividiéndose el vector cromosoma de dos partes: PRIORITY (prioridades de las actividades) y tdelay (tiempo de retraso de las actividades).

CROMOSOMAS:

Lanzamiento 1:

PRIORITY:

0.141176 0.94902 0.65098 0.886275 0.376471 0.87451 0.172549 0.862745 0.517647 0.145098 0.823529 0.592157 0.835294 0.54902 0.760784 0.623529 0.827451 0.290196 0.192157 0.294118 0.764706 0.223529 0.580392 0.403922 0.0627451 0.788235 0.396078 0.0941176 0.788235 0.823529

tdelay:

6.05098 5.95686 8.57647 7.24314 6.72549 7.63529 6.16078 8.21569 8.18431 8.27843 5.53333 6.47451 5.50196 6.89804 8.6549 7.7451 8.26274 7.14902 5.61176 6.91373 5.29804 5.78431 8.04314 6.99216 5.20392 5.62745 8.2 8.62353 6.72549 8.29412

Lanzamiento 2:

PRIORITY:

0.215686 0.0117647 0.929412 0.368627 0.980392 0.439216 0.301961 0.184314 0.188235 0.572549 0.388235 0.454902 0.909804 0.65098 0.454902 0.768627 0.929412 0.768627 0.6 0 0.0196078 0.411765 0.603922 0.52549 0.756863 0.235294 0.894118 0.603922 0.447059 0.682353

tdelay:

7.35294 5.47059 5.47059 7.33726 7.68235 8.07451 5.67451 6.4902 5.37647 5.15686 5.07843 5.51765 5.18824 6.74118 7.47843 5.28235 7.43137 5.28235 6.85098 8.70196 6.52157 5.01569 8.52941 5.31373 6.41176 5.70588 8.74902 7.16471 6.99216 7.77647

Lanzamiento 3:

PRIORITY:

0.392157 0.0627451 0.788235 0.835294 0.478431 0.188235 0.635294 0.0941176 0.894118 0.937255 0.505882 0.623529 0.298039 0.756863 0.0784314 0.0980392 0.627451 0.00392157 0 0.00392157 0.827451 0.654902 0.0431373 0.611765 0.203922 0.419608 0.109804 0.576471 0.0705882 0.164706

tdelay:

5.4549 8.67059 6.75686 7.30588 5.29804 5.42353 6.89804 7.90196 7.7451 6.77255 5.83137 8.84314 7.19608 5.26667 6.36471 7.35294 6.4902 6.97647 7.03922 6.6 5.42353 5.01569 6.61569 8.98431 7.11765 8.41961 7.80784 5.98824 5.09412 8.16863

Lanzamiento 4:

PRIORITY:

0.517647 0.556863 0.596078 0 0.403922 0.560784 0.431373 0.0941176 0.0784314 0.196078 0.219608 0.552941 0.243137 0.462745 0.290196 0.396078 0.301961 0.176471 0.603922 0.482353 0.588235 0.878431 0.239216 0.756863 0.929412 0.65098 0.745098 0.0901961 0.439216 0.027451

tdelay:

8.74902 5.50196 7.2902 7.61961 6.99216 5.40784 8.37255 7.57255 8.35686 6.33333 5.1098 7.19608 5.84706 7.63529 6.7098 6.69412 6.23922 6.41176 6.7098 6.17647 8.45098 8.78039 8.93725 7.7451 6.72549 7.90196 5.32941 7.0549 6.97647 5.42353

Lanzamiento 5:

PRIORITY:

0.243137 0.262745 0.439216 0.0941176 0.501961 0.858824 0.807843 0.705882 0.462745 0.811765 0.188235 0.180392 0.152941 0.262745 0.180392 0.968627 0.921569 0.760784 0.462745 0.694118 0.752941 0.886275 0.564706 0.984314 0.0627451 0.984314 0.247059 0.858824 0.419608 0.556863

tdelay:

5.87843 7.14902 5.07843 8.26274 7.76078 7.43137 8.82745 8.43529 8.43529 8.35686 5.03137 6.36471 5.3451 8.71765 6.28627 8.60784 7.00784 8.84314 5.53333 6.55294 5.42353 5 7.82353 6.38039 7.19608 6.09804 8.12157 6.2549 5.40784 8.40392

Lanzamiento 6:

PRIORITY:

0.0156863 0.0705882 0.670588 0.0627451 0.839216 0.607843 0.831373 0.0196078 0.0666667 0.388235 0.0627451 0.647059 0.984314 0.211765 0.72549 0.972549 0.556863 0.937255 0.858824 0.211765 0.556863 0.0862745 0.113725 1 0.14902 0.188235 0.388235 0.411765 0.772549 0.152941

tdelay:

8.56078 5.39216 7.18039 5.43922 5.31373 7.24314 8.8902 8.93725 7.80784 6.30196 6.23922 7.79216 7.77647 8.5451 8.05882 7.69804 7.03922 7.2902 5.01569 6.30196 6.83529 7.4 5.6902 8.46667 7.22745 6.19216 8.16863 7.58824 6.50588 8.67059

Lanzamiento 7:

PRIORITY:

0.2 0.239216 0.984314 0.729412 0.145098 0.87451 0.192157 0.435294 0.639216 0.462745 0.203922 0.654902 0.356863 0.811765 0.560784 0.984314 0.529412 0.756863 0.427451 0.243137 0.941176 0.443137 0.266667 0.839216 0.317647 0.133333 0.788235 0.223529 0.505882 0.968627

tdelay:

6.88235 7.38431 8.32549 5.14118 7.36863 6.03529 8.45098 5.62745 8.46667 5.40784 6.58431 5.56471 8.51373 6.53725 8.05882 6.11373 5.14118 7.30588 7.52549 6.12941 8.02745 6.12941 8.35686 7.18039 6.39608 5.76863 8.57647 6.31765 7.54118 7.36863

Lanzamiento 8:

PRIORITY:

0.0431373 0.00784314 0.678431 0.564706 0.333333 0.0627451 0.792157 0.196078 0.0745098 0.368627 0.576471 0.243137 0.678431 0.870588 0 0.447059 0.596078 0.494118 0.827451 0.862745 0.764706 0.341176 0.737255 0.419608 0.72549 0.0117647 0.980392 0.105882 0.301961 0.976471

tdelay:

5.78431 6.31765 5.94118 6.19216 6.01961 6.88235 7.63529 7.79216 5.50196 7.90196 8.63922 6.17647 5.42353 8.27843 6.75686 8.90588 6.30196 6.81961 7.47843 5.72157 6.33333 6.69412 6.22353 5.83137 6.85098 8.6549 8.24706 8.43529 8.57647 6.66274

Lanzamiento 9:

PRIORITY:

0.105882 0.529412 0.647059 0.870588 0.964706 0.980392 0.65098 0.372549 0.639216 0.780392 0.756863 0.894118 0.211765 0.611765 0.933333 0.984314 0.729412 0.792157 0.0901961 0.894118 0.505882 0.67451 0.972549 0.784314 0.45098 0.341176 0.780392 0.0980392 0.0627451 0.529412

tdelay:

7.69804 6.53725 5.59608 6.53725 5.59608 7.66667 6.47451 8.24706 7.14902 6.20784 6.50588 6.16078 8.35686 8.24706 7.98039 8.62353 5.43922 8.84314 6.6 8.32549 8.24706 8.32549 8.8902 7.00784 6.16078 8.32549 5.32941 6.23922 5.15686 8.20000

Lanzamiento 10:

PRIORITY:

0.0392157 0.733333 0.980392 0.396078 0.215686 0.8 0.25098 0.517647 0.65098 0.764706 0.713726 0.517647 0.392157 0.388235 0.0901961 0.94902 0.701961 0.847059 0.552941 0.737255 0.113725 0.733333 0.886275 0.180392 0.933333 0.65098 0.247059 0.388235 0.411765 0.933333

tdelay:

7.32157 8.35686 8.46667 5.29804 8.43529 7.94902 7.54118 6.23922 7.41569 5.6902 7.46275 8.5451 6.80392 8.63922 7.55686 6.23922 5.39216 7.90196 8.43529 6.00392 5.95686 6.80392 5.04706 5.07843 8.01176 7.35294 5.01569 7.16471 5.89412 5.76863

4 CONCLUSIONES Y LÍNEAS FUTURAS

4.1 CONCLUSIONES

En el presente trabajo se estudia e implementa un algoritmo para la programación óptima de proyectos de ingeniería. Su metodología en primer lugar fue conocer las actividades del proyecto y sus respectivas condiciones de precedencia en una red PERT. Luego, se calculó el cromosoma de la programación, que sirvió como dato de entrada para el procedimiento de la programación activa parametrizada (planificador). Finalmente el resultado de dicha programación sirvió de feedback de calidad del cromosoma (función objetivo) al algoritmo genético (tiempo de ejecución del proyecto modificado).

Sabiendo la metodología a emplear, se eligieron tres casos de programaciones distintas para analizar los diferentes resultados. Las principales conclusiones a las que se ha llegado en este documento se resumen de la siguiente forma:

- Se cumple el objetivo de implementar un programa que permite calcular una programación de actividades, cumpliendo las condiciones de precedencia, cantidad de recursos tipos para cada actividad y los parámetros de cálculo del algoritmo citado por el Dr. Mendes [1].
- Se cumple, además, el ensamblado de un algoritmo genético que es capaz de buscar el tiempo óptimo posible (tiempo de ejecución del proyecto modificado - minimizar). Esto significa, una mejora considerable en la organización y toma de decisiones de una empresa para realizar un proyecto en el menor tiempo posible y por ende, una posible reducción de los costes.
- Se comprueba con las pruebas de programación que se han realizado, aquellos proyectos cuyo número de actividades estén por debajo de 20 actividades, la convergencia es prácticamente inmediata. Esa eficaz convergencia es debido a que las

posibilidades de colocación de las actividades son muy limitadas, y el algoritmo genético obtiene rápidamente el óptimo.

- El procedimiento de programación sólo es capaz de calcular la programación si las actividades son no recursivas.

4.2 LÍNEAS FUTURAS

Haciendo referencia a las líneas futuras, se pueden realizar las siguientes:

- Mejorar el programa para que sea capaz de: Introducir por pantalla los parámetros necesarios para el cálculo de la programación, representar el diagrama de actividades colocadas por recurso tipo y sus tiempos finales y representar las gráficas de cada uno de los lanzamientos (Generaciones frente a MM) con su media y desviación típica.
- Inclusión de incertidumbres.
- Inclusión de costes. De esta forma se podría observar si un proyecto es viable.
- Implementar en el algoritmo el uso de metodologías multiobjetivo para la optimización simultánea de tiempo y coste.
- En la mayoría de los proyectos existen numerosos imprevistos que hacen retrasar las actividades o desestimar algunas de ellas. Además, no se hace hincapié en los recursos de los que dispone la empresa para contratar recursos humanos, medios mecánicos y materiales. Es interesante que el algoritmo sea capaz, por un lado, de recalcular el diagrama de actividades en el instante de tiempo que se precise (por posibles retrasos o imprevistos en el proyecto) y, por otro lado, que sea capaz de comprobar la viabilidad del proyecto, controlando los recursos y realizar un cálculo que permita saber si es rentable en tiempos y costes, la contratación de nuevos recursos y materiales.

5 BIBLIOGRAFÍA

[1] J.J.M. Mendes, J.F. Gonçalves, M.G.C. Resende. A random key based genetic algorithm for the resource constrained Project scheduling problem. Computer & Operations Research 36 (2009) 92-109.

[2] J.F. Gonçalves, J.J.M. Mendes, M.G.C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research 167 (2005) 77-95.

[3] J.F. Gonçalves, J.J.M. Mendes, M.G.C. Resende. A genetic algorithm for the resource constrained multi-project scheduling problem. European Journal of Operational Research 189 (2008) 1171-1190.

[4] J.F. Gonçalves, M.G.C. Resende , J.J.M. Mendes. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. J Heuristics (2011) 17:467-486.

[5] J.J. Magalhães Mendes (2008). *Planeamento de projectos com recursos limitados*. Edições Politema, Porto.

[6] D. Greiner, B. Galvan, J. Periaux, N. Gauger, K. Giannakoglou, G. Winter, "Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences", Computational Methods in Applied Sciences, Vol. 36, Springer, (2015).

ANEXO 1 – ALGORITMO (EJEMPLO J30)

```
#include <stdio.h>
#include <math.h>
#include <ga/GASimpleGA.h>
#include <ga/GABin2DecGenome.h>
#include <ga/GA1DBinStrGenome.h>
#include <ga/GABin2DecGenome.h>
#include <ga/std_stream.h>
#include <ga/ga.h>
#include <ga/GABaseGA.h>
#include <ga/GASStateGA.h>
#include <ga/GAStatistics.h>
#define cout STD_COUT
#include <algorithm>
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <vector>
//#include <conio.h> // Solo Windows

using namespace std;

vector<int> schedules(float PRIORITY[], float tdelay[]);

float Objective(GAGenome &);


int main() {

  int LChrom=60,i;
  float tdelaymin=5.0, tdelaymax=9.0;
  double CHROM[LChrom];

  ofstream outfile;
  ofstream Archivo1;
  ofstream Archivo2;

  GABin2DecPhenotype map;  //Create the 'map' phenotype for the 'genome' genome.

//Creates the phenotypes for the coordinates of the QRD barrier.
    map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(1))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(2))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(3))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(4))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(5))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(6))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(1))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(2))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(3))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(4))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(1))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(2))*/
    map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(3))*/
```

```
55      map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(4))*/
56      map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(5))*/
57      map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(6))*/
58      map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(1))*/
59      map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(2))*/
60      map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(3))*/
61      map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(4))*/
62      map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(1))*/
63      map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(2))*/
64      map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(3))*/
65      map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(4))*/
66      map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(5))*/
67      map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(6))*/
68      map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(1))*/
69      map.add(8,0,1);                 /*Se crea al fenotipo de eta8 (CHROM(2))*/
70      map.add(8,0,1);                 /*Se crea al fenotipo de eta10 (CHROM(3))*/
71      map.add(8,0,1);                 /*Se crea al fenotipo de eta6 (CHROM(4))*/
72      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(7))*/
73      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(8))*/
74      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(9))*/
75      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(10))*/
76      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(11))*/
77      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(12))*/
78      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(7))*/
79      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(8))*/
80      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(9))*/
81      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(10))*/
82      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(7))*/
83      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(8))*/
84      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(9))*/
85      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(10))*/
86      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(11))*/
87      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(12))*/
88      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(7))*/
89      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(8))*/
90      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(9))*/
91      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(10))*/
92      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(7))*/
93      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(8))*/
94      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(9))*/
95      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(10))*/
96      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(11))*/
97      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(12))*/
98      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(7))*/
99      map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(8))*/
100     map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(9))*/
101     map.add(8,tdelaymin,tdelaymax);               /*Se crea al fenotipo de eta6 (CHROM(10))*/
102
103
104   GABin2DecGenome genome(map, Objective);
105
106 /*------------------------------Se emplea el CÓDIGO GRAY-----------------------------------------*/
107 typedef int (*GAGrayDecode)(float& value, GABit* bits, unsigned int nbits, float min, float max);
108 typedef int (*GAGrayEncode)(float& value, const GABit* bits, unsigned int nbits, float min, float max);
```

```
/*-------------------------------------------------------------------------------------------*/

/*--------------------------Se establecen los PARÁMETROS DEL AG--------------------------------*/
GAParameterList params;
GASteadyStateGA::registerDefaultParameters(params);
params.set(gaNpopulationSize, 15);      // number of individuals in population
params.set(gaNnBestGenomes,3);
/*.........................................................................................*/
params.set(gaNpCrossover, 0.90);        // likelihood of doing crossover
params.set(gaNpMutation, 0.02);         // probability of mutation
/*.........................................................................................*/
params.set(gaNnGenerations, 150);       // number of generations
params.set(gaNpReplacement,0.02);       // replace 2% of population each generation
params.set(gaNscoreFrequency, 1);       // how often to record scores
params.set(gaNflushFrequency, 1);       // how often to flush scores to file
params.set(gaNselectScores,             // which scores should we track?
GAStatistics::Mean|GAStatistics::Maximum|GAStatistics::Minimum|GAStatistics::Deviation|GAStatistics::Diversity);
params.set(gaNscoreFilename, "Evol_AG.dat");
/*-------------------------------------------------------------------------------------------*/
/*--------------------------Tipo de AG EMPLEADO y otros parámetros-----------------------------*/
GASteadyStateGA ga(genome);
GASigmaTruncationScaling scaling;
ga.parameters(params);
/*-------------------------------------------------------------------------------------------*/
ga.minimize();                  // Minimizar Función Objetivo
GATournamentSelector selector;
ga.selector(selector);
/*-------------------------------------------------------------------------------------------*/


/*---------------------------------Se realiza la EVOLUCIÓN-------------------------------------*/
/*.....................................Manualmente.............................................*/
ga.initialize();

outfile.open("Best_Worst.dat",ios::out);
Archivo2.open("Evolution.dat",ios::out);

outfile << ga.generation() << "\t" << ga.statistics().numeval << "\t\t" << ga.population().best() << "\t\t" << ga.population().individual(0).score() << "\t\t\t\t" <<
                ga.population().worst() << "\t\t" << ga.population().individual(ga.population().size()-1).score() << endl;

Archivo2 << ga.generation() << "\t" << ga.statistics().numeval << "\t" << ga.population().ave() << "\t" << ga.population().max() << "\t" <<
                ga.population().min() << "\t" << ga.population().dev() << "\t" << ga.population().var() << endl;
/*-------------------------------------------------------------------------------------------*/

    while(!ga.done())
{
    ++ga;

    outfile << ga.generation() << "\t" << ga.statistics().numeval << "\t\t" << ga.population().best() << "\t\t" << ga.population().individual(0).score() << "\t\t\t\t" <<
                ga.population().worst() << "\t\t" << ga.population().individual(ga.population().size()-1).score() << endl;

Archivo2 << ga.generation() << "\t" << ga.statistics().numeval << "\t" << ga.population().ave() << "\t" << ga.population().max() << "\t" <<
                ga.population().min() << "\t" << ga.population().dev() << "\t" << ga.population().var() << endl;
}
```

```
163
164     if(ga.generation()==ga.nGenerations())
165         {
166         Archivo1.open("Last_Gen.dat",ios::out);
167
168             for(i=0;i<ga.population().size();i++)
169             {
170             Archivo1 << i+1 << "\t\t" << ga.population().individual(i) << "\t\t" << ga.population().individual(i).score() << endl;
171             }
172
173         Archivo1.close();
174         }
175
176     Archivo2.close();
177     outfile.close();
178 /*-------------------------------------------------------------------------------------------------*/
179
180 /*---------------------------------Se ESCRIBEN los valores-----------------------------------------*/
181 ga.flushScores();
182 /*-------------------------------------------------------------------------------------------------*/
183
184 /*----------------------------ESCRITURA DE VALORES INTERESANTES-----------------------------------*/
185
186 outfile.open("Best_Genome.dat",ios::out);
187 outfile << ga.statistics().bestIndividual();
188 outfile.close();
189
190 outfile.open("Statistics.dat",ios::out);
191 outfile << ga.statistics();
192 outfile.close();
193
194 outfile.open("nBestGenomes.dat",ios::out);
195 outfile << ga.statistics().bestPopulation()<< endl;
196 outfile.close();
197
198 /*-------------------------------------------------------------------------------------------------*/
199
200 return 0;
201 }
202
203 //This is the objective function to be call during the GA evolution process.
204
205 float
206 Objective(GAGenome& g)
207 {
208
209   GABin2DecGenome & genome = (GABin2DecGenome &)g;
210 //  GA1DBinaryStringGenome & genome = (GA1DBinaryStringGenome &)g;
211
212   int LChrom=60,i,j,L;
213   double CHROM[LChrom];
214   double ifobj,suma,sup,dwn;
215   float PRIORITY[32];
216   float tdelay[32];
```

```
217     vector<int> F;
218     vector<int> dist;
219
220     ifobj=0.0;
221
222     cout << "=========================================================" << endl;
223     cout << "                  Cálculos Previos" << endl;
224     cout << "=========================================================\n" << endl;
225     cout << "CHROM[LChrom] = ";
226     for(i=0;i<LChrom;i++)
227     {
228       CHROM[i]=genome.phenotype(i);
229       cout << CHROM[i] << " ";
230     }
231
232     PRIORITY[0]=0.0;
233     PRIORITY[(sizeof(PRIORITY)/sizeof(*PRIORITY))]=0.0;
234     tdelay[0]=0.0;
235     tdelay[(sizeof(tdelay)/sizeof(*tdelay))]=0.0;
236     for(i=1;i<31;i++) {
237       PRIORITY[i]=genome.phenotype(i-1);
238       tdelay[i]=genome.phenotype(29+i);
239     }
240     cout << "\n\nPRIORITY[32] = ";
241     for(i=0;i<32;i++) {
242       cout << PRIORITY[i] << " ";
243     }
244     cout << "\n\ntdelay[32] = ";
245     for(i=0;i<32;i++) {
246       cout << tdelay[i] << " ";
247     }
248     cout << "\n\n";
249
250     F = schedules(PRIORITY, tdelay);
251
252       for (i=0;i<F.size();i++) {
253           cout << F[i] << " ";
254       }
255       cout << "\n\nMakeSpan (Cmax) = " << F[F.size()-1];
256       cout << "\n";
257
258     dist.push_back(3);dist.push_back(4);dist.push_back(3);dist.push_back(1);dist.push_back(4);
259     dist.push_back(2);dist.push_back(4);dist.push_back(3);dist.push_back(4);dist.push_back(3);
260     dist.push_back(2);dist.push_back(3);dist.push_back(2);dist.push_back(2);dist.push_back(2);
261     dist.push_back(3);dist.push_back(1);dist.push_back(1);dist.push_back(3);dist.push_back(3);
262     dist.push_back(3);dist.push_back(1);dist.push_back(1);dist.push_back(4);dist.push_back(2);
263     dist.push_back(1);dist.push_back(2);dist.push_back(2);dist.push_back(2);dist.push_back(1);
264
265     L=1;sup=0;dwn=0;
266     for(i=0;i<L;i++) {
267       for(j=0;j<dist.size();j++) {
268           if(dist[j]==i+1) {
269               sup=sup+F[j+1];
270               dwn=dwn+F[F.size()-1];
```

```
        }
    }
  }
  suma=sup/dwn;
  ifobj=F[F.size()-1]+suma;

  cout << "\nMM = " << ifobj << "\n\n";


  return ifobj;

}

vector<int> schedules(float PRIORITY[], float tdelay[]) {

// Inicialización ============================================================
    vector <int> E; E.push_back(1); E.push_back(2); E.push_back(3);
    vector <int> S; S.push_back(0);
    vector <int> T; T.push_back(0);
    vector <int> t; t.push_back(0); t.push_back(0);
    vector <int> A;
    vector <int> vtg;
    vector <int> F1;
    F1.push_back(0);
// Actividades del proyecto
    int k[]={6,4,6},Cmax=100,n[32]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31};
    int d[]={0,2,2,5,1,1,3,3,5,2,1,3,3,4,2,2,2,3,4,4,2,5,2,3,3,1,1,2,3,4,1,0};
// Contadores
    int i,j,l,g=1;
// ===========================================================================
// ===========================================================================

// Matriz M ==================================================================
    typedef vector<float> a;
    typedef vector<a> b;

    b M(4,a(32));

    for(i=0;i<M.size();++i) {
        M[i].resize(4);
    }

    b M2(4,a(32));

    for(i=0;i<4;++i) {
        for(j=0;j<32;++j) {
            M[i][j]=M2[i][j]=0;
        }
    }

    for(i=0;i<4;++i) {
        for(j=0;j<31;++j) {
            M[0][j]=n[j];
            M[1][j]=PRIORITY[j];
```

```
325             M[2][j]=tdelay[j];
326             M[3][j]=d[j];
327         }
328
329     }
330 // ================================================================================
331
332 // Generar vector precedentes P[] ============================================
333     b P(31);
334
335     for(i=0;i<P.size();++i) {
336         P[i].resize(9);
337     }
338
339     b P2(31,a(9));
340
341     for(i=0;i<P.size();++i) {
342         for(j=0;j<P[i].size();++j) {
343             P[i][j]=P2[i][j]=-1;
344         }
345     }
346
347     P[0][0]=0;
348     P[1][0]=0;
349     P[2][0]=0;
350     P[3][0]=6;
351     P[3][1]=11;
352     P[3][2]=13;
353     P[4][0]=2;
354     P[5][0]=1;
355     P[6][0]=2;
356     P[6][1]=3;
357     P[7][0]=1;
358     P[8][0]=3;
359     P[9][0]=2;
360     P[10][0]=8;
361     P[10][1]=20;
362     P[11][0]=7;
363     P[11][1]=9;
364     P[12][0]=20;
365     P[13][0]=16;
366     P[14][0]=3;
367     P[15][0]=24;
368     P[16][0]=25;
369     P[17][0]=14;
370     P[17][1]=15;
371     P[18][0]=1;
372     P[18][1]=2;
373     P[19][0]=1;
374     P[20][0]=5;
375     P[20][1]=7;
376     P[21][0]=27;
377     P[21][1]=29;
378     P[22][0]=28;
```

```
379         P[23][0]=3;
380         P[24][0]=21;
381         P[25][0]=27;
382         P[26][0]=10;
383         P[27][0]=12;
384         P[27][1]=21;
385         P[28][0]=10;
386         P[28][1]=19;
387         P[28][2]=20;
388         P[29][0]=14;
389         P[30][0]=4;
390         P[30][1]=17;
391         P[30][2]=18;
392         P[30][3]=22;
393         P[30][4]=23;
394         P[30][5]=26;
395         P[30][6]=30;
396
397     // ============================================================================
398
399     // Generar vector tiempo final de las actividades F[] ========================
400         typedef vector<int> c;
401
402         c F(32);
403         for(i=1;i<F.size();i++) {
404             F[i]=20000;
405         }
406     // ============================================================================
407
408     // Inicializar matriz de recursos disponibles RD[][] =========================
409         std::vector< std::vector<int> > RD;
410         RD.resize(sizeof(k)/sizeof(*k));
411
412         for(i=0;i<(sizeof(k)/sizeof(*k));i++) {
413             RD[i].resize(Cmax);
414         }
415
416         for(j=0;j<Cmax;j++) {
417             for(i=0;i<(sizeof(k)/sizeof(*k));i++) {
418                 RD[0][j]=6;
419                 RD[1][j]=4;
420                 RD[2][j]=6;
421             }
422         }
423     //============================================================================
424
425     // Inicializar matriz de recursos tipo r[][] ==================================
426         int r[30][3];
427
428         r[0][0]=1;r[0][1]=1;r[0][2]=0;
429         r[1][0]=0;r[1][1]=1;r[1][2]=0;
430         r[2][0]=2;r[2][1]=1;r[2][2]=1;
431         r[3][0]=0;r[3][1]=0;r[3][2]=2;
432         r[4][0]=0;r[4][1]=1;r[4][2]=2;
```

```
433         r[5][0]=3;r[5][1]=1;r[5][2]=1;
434         r[6][0]=1;r[6][1]=1;r[6][2]=1;
435         r[7][0]=2;r[7][1]=1;r[7][2]=1;
436         r[8][0]=0;r[8][1]=0;r[8][2]=3;
437         r[9][0]=4;r[9][1]=4;r[9][2]=1;
438         r[10][0]=2;r[10][1]=2;r[10][2]=2;
439         r[11][0]=5;r[11][1]=0;r[11][2]=0;
440         r[12][0]=0;r[12][1]=2;r[12][2]=0;
441         r[13][0]=1;r[13][1]=0;r[13][2]=2;
442         r[14][0]=2;r[14][1]=1;r[14][2]=0;
443         r[15][0]=2;r[15][1]=1;r[15][2]=1;
444         r[16][0]=0;r[16][1]=0;r[16][2]=1;
445         r[17][0]=0;r[17][1]=1;r[17][2]=0;
446         r[18][0]=1;r[18][1]=0;r[18][2]=0;
447         r[19][0]=3;r[19][1]=1;r[19][2]=0;
448         r[20][0]=1;r[20][1]=1;r[20][2]=1;
449         r[21][0]=0;r[21][1]=2;r[21][2]=0;
450         r[22][0]=2;r[22][1]=0;r[22][2]=0;
451         r[23][0]=2;r[23][1]=1;r[23][2]=0;
452         r[24][0]=1;r[24][1]=1;r[24][2]=0;
453         r[25][0]=2;r[25][1]=0;r[25][2]=0;
454         r[26][0]=0;r[26][1]=0;r[26][2]=2;
455         r[27][0]=1;r[27][1]=1;r[27][2]=1;
456         r[28][0]=3;r[28][1]=1;r[28][2]=1;
457         r[29][0]=2;r[29][1]=1;r[29][2]=2;
458     //=============================================================================
459
460     //=============================================================================
461     //=============================================================================
462     // Inicio Main del Algoritmo ==================================================
463     //=============================================================================
464     //=============================================================================
465         int jast=0,EF;
466         int Ssize=(S.size()),Esize=E.size();
467         float prio,max=0;
468
469         while (S.size()<(sizeof(n)/sizeof(*n))+2) {
470             while(E.size()!=0) {
471     // Bucle prioridad (j*) ======================================================
472                 prio=-1;
473                 max=0;
474
475                 for(i=0;i<E.size();i++){
476                     if(M[1][E[i]]>prio){
477                         jast=E[i];
478                         prio=M[1][E[i]];
479                     }
480                 }
481     // EF ========================================================================
482                 for(j=0;j<P.size();j++) {
483                     if(F[P[jast-1][j]]==0 || P[jast-1][j]==-1) {
484                         break;
485                     }
486                     if(F[P[jast-1][j]]>max){
```

```
487                     max= F[P[jast-1][j]];
488                 }
489             }
490             EF=max+d[jast];
491 // Fj* ================================================================================
492             int tEFd,min[40];
493
494             tEFd=EF-d[jast];
495
496             min[0]=tEFd;
497
498             for (i=1;i<40;i++) {
499                 min[i]=min[i-1]+1;
500             }
501
502
503             std::vector<int> v((sizeof(min)/sizeof(*min))+(T.size()));
504             std::vector<int>::iterator it;
505
506             std::sort (min,min+(sizeof(min)/sizeof(*min)));
507             std::sort (T.begin(),T.end());
508
509             it=std::set_intersection (min, min+(sizeof(min)/sizeof(*min)), T.begin(), T.end(), v.begin());
510
511             v.resize(it-v.begin());
512
513             for (it=v.begin(); it!=v.end(); ++it)
514
515             for(i=0;i<v.size();i++) {
516                 for(l=v[i];l<(v[i]+d[jast]);l++) {
517                     for(j=0;j<3;j++) {
518                         if(r[jast-1][j]>RD[j][l]) {
519                         goto NoCumple;
520                         }
521                     }
522                 }
523                 F[jast]=v[i]+d[jast];
524                 goto Seguir;
525                 NoCumple:
526                 continue;
527             }
528             cout << "ERROR: No hay recursos suficientes\n";
529             exit(1);
530             Seguir:
531             F1.push_back(F[jast]);
532 // Sg ================================================================================
533             S.push_back(jast);
534             std::sort (S.begin(),S.end());
535 // Tg ================================================================================
536             T.push_back(F[jast]);
537             std::sort (T.begin(),T.end());
538 // Actualización g ===================================================================
539             g=g+1;
540             t.push_back(t[g-1]);
```

```
541  // Ag ========================================================================
542              A.clear();
543              if(t[g]==0) {
544                  A.push_back(0);
545              }
546
547              for (i=0;i<F.size()+1;i++) {
548                  if((F[i]-d[i])<=t[g] && t[g]<F[i]) {
549                      A.push_back(i);
550                  }
551              }
552  // Eg ========================================================================
553              E.clear();
554
555              int seguir,cumple;
556
557              for(j=0;j<(sizeof(n)/sizeof(*n)-1);j++) {
558                  seguir=1;
559                  cumple=0;
560                  for(i=0;i<S.size();i++) {
561                      if(S[i]==j) {
562                          seguir=0;
563                          break;
564                      }
565                  }
566                  if(seguir==1) {
567                      i=0;
568                      cumple=1;
569                      while(P[j-1][i]!=-1) {
570                          if(F[P[j-1][i]]>t[g]+tdelay[g])
571                              cumple=0;
572                              i++;
573                      }
574                  }
575                  if(cumple==1) {
576                      E.push_back(j);
577                  }
578              }
579  // RD (actualización) ========================================================
580              for(i=0;i<(sizeof(k)/sizeof(*k));i++) {
581                  for(j=F[jast]-d[jast];j<F[jast];j++) {
582                      RD[i][j]=RD[i][j]-r[jast-1][i];
583                  }
584              }
585          } // cierre while interior
586  // ===========================================================================
587  // Fin SubMain del Algoritmo =================================================
588  // ===========================================================================
589
590              if (S.size()==(sizeof(n)/sizeof(*n))) {
591              break;
592          }
593
594  // tg (actualización) ========================================================
```

```
// tg (actualización) ==========================================================
        int i_inic = 0;
        do {

        vtg.clear();

        for(i=i_inic;i<T.size();i++) {
            if(T[i]>t[g-1]) {
                break;
            }
        }
        t[g]=T[i];
        i_inic=i+1;

// Eg Nueva ====================================================================
        int seguir1,cumple1;

        for(j=0;j<(sizeof(n)/sizeof(*n));j++) {
            seguir1=1;
            cumple1=0;
            for(i=0;i<S.size();i++) {
                if(S[i]==j) {
                    seguir1=0;
                    break;
                }
            }
            if(seguir1==1) {
                i=0;
                cumple1=1;
                while(P[j-1][i]!=-1) {
                    if(F[P[j-1][i]]>t[g]+tdelay[g])
                        cumple1=0;
                        i++;
                }
            }
            if(cumple1==1) {
                E.push_back(j);
            }
        }

        } while (E.size()==0);
    } // cierre while exterior
//==============================================================================
//==============================================================================
// Fin Main del Algoritmo ======================================================
//==============================================================================
//==============================================================================
    cout << "============================================================" << endl;
    cout << "                    Resultados" << endl;
    cout << "============================================================\n" << endl;
    cout << "MakeSpan (Cmax) = " << F[F.size()-1];
    cout << "\n\n";
    cout << "F[] = ";

    return F;
} // cierre main
```