Universidad de las Ciencias Informáticas
Vertex, Entornos Interactivos 3D, Facultad 5

Algoritmo Genético para la planificación de tareas de un proyecto de software

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Esmaykel Vázquez Avila

Tutores: Ing. Susej Beovides Luis

Ing. Gabriel Pedraza Rodríguez

La Habana, 2015

"La planificación a largo plazo no es pensar en decisiones futuras, sino en el futuro de las decisiones presentes."

Peter Drucker

Dedicatoria

A mis padres Héctor y Elaine, por enseñarme a batallar y aspirar siempre a lo más alto.

A mis abuelos Pilar y Héctor, mi otra suerte de padres.

A mi hermano Daniel, de quien me siento muy orgulloso.

A mi tía Blanca, porque desde la distancia siempre ha estado presente.

Agradecimientos

A los que a lo largo de estos años me han hecho crecer como ser humano y como profesional, la lista sería extensa.

A mis padres y abuelos por darme la confianza y el apoyo necesario en cada momento, por siempre creer en mí.

A mi tía Blanca, por hacerme más fácil el tránsito por esta experiencia universidad, y por su consejo siempre certero.

A mi hermano, a mi familia y amigos por estar siempre cuando los necesito.

A Yoivys por su cariño y preocupación.

A Susej, más que tutora, compañera de tesis; por dedicarme tanto tiempo aún cuando no lo tenía.

A Gabriel por sus minuciosas revisiones y sus valiosos consejos.

A todos aquellos que colaboraron con idea, acción o energía positiva en este trabajo.

A mi piquete, desde el 5106 hasta el 5503, por tantos buenos ratos en estos cinco años. A Erduin, Yerson, Mirnerys, Luis, Kiki, Lisandra, Orson, Yandry, Marín y Yanet.

"La gratitud debería ser un acto constante de cada hora, de cada día, de toda la vida."
Nancy Leigh DeMoss

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Esmaykel Vázquez Avila
Autor

Ing. Susej Beovides Luis
Tutora

Ing. Gabriel Pedraza Rodríguez
Tutor

Resumen

En la Universidad de las Ciencias Informáticas existen varios centros dedicados al desarrollo de software, los cuales gestionan sus productos a través de proyectos. En la planificación de los mismos se utiliza GESPRO, una plataforma orientada a la coordinación, seguimiento de las tareas y a la comunicación entre los participantes del proyecto. Pero la planificación se realiza de forma manual y se evidencian deficiencias en la misma. Esta investigación tiene por objetivo desarrollar una estrategia de planificación que minimice el tiempo de ejecución y optimice la asignación de recursos en un proyecto de software. El aporte más significativo de este trabajo es la solución del Problema de Planificación de Proyectos con Recursos Limitados a través de la técnica metaheurística Algoritmos Genéticos. Se comprueba el desempeño del algoritmo propuesto mediante instancias de pruebas de PSPLib. Como conclusión se plantea que el Algoritmo Genético diseñado obtiene soluciones óptimas o cercanas a las óptimas para el Problema de Planificación con Recursos Limitados. Además, se logra un adecuado balance de carga entre los recursos de un mismo tipo.

Palabras clave: Algoritmo Genético, Planificación de Proyectos de Software.

Índice general

Índice de figuras

Índice de tablas

Lista de algoritmos

Lista de códigos fuentes

Introducción

La gestión del tiempo de un proyecto de software comienza con un conjunto de actividades que globalmente se denomina planificación del proyecto. El establecimiento de una programación es uno de los aspectos más importantes para planificar un proyecto. Esto permite establecer un guion temporal de las tareas a desarrollar durante la realización del software. Además, tiene en cuenta todos los factores decisivos en la ejecución de los trabajos, desde el orden necesario hasta la disponibilidad de los recursos.

Es difícil que un equipo de especialistas realice la mejor alternativa de planificación de un proyecto desde su primera versión. Esto se debe a que el número de alternativas a evaluar es una función factorial que depende de la cantidad de actividades a planificar y de los recursos disponibles, lo que aumenta considerablemente el tiempo de cómputo. Es por ello que la planificación de las tareas de proyectos de software se considera un problema cuya solución algorítmica está acotada por tiempos no polinomiales (NP-hard). Al avanzar el proyecto algunas actividades pueden requerir mayor o menor tiempo para completarse, en dependencia de la disponibilidad de recursos que se posee, lo cual hace cambiar la planificación de las tareas.

Al aumentar la complejidad de un proyecto de software, se hace imperiosa la necesidad de planificar. La planificación garantiza, en gran medida, la eficiencia en el desempeño del equipo de proyecto. Además, establece métodos de utilización racional de los recursos y del tiempo. De igual manera proporciona los elementos necesarios para llevar a cabo el control y establecer un sistema adecuado para la toma de decisiones, que hagan frente a las contingencias que pueden presentarse.

Situación problémica

En la Universidad de las Ciencias Informáticas (UCI) existen varios centros dedicados al desarrollo de software, los cuales gestionan sus productos a través de proyectos. En la planificación de los mismos se utiliza *GESPRO*[1] en su versión 13.5. Esta plataforma fue desarrollada a partir de herramientas de software libre, contiene funcionalidades que garantizan la gestión de datos de los proyectos y la ayuda a la toma de decisiones. La planificación se lleva a cabo usando esta plataforma orientada a la coordinación, seguimiento de las tareas y a la comunicación entre los participantes del proyecto. Sin embargo, se realiza de forma manual, lo que puede provocar deficiencias en la misma, ya que:

[1] *Ecosistema de software para la Dirección Integrada de Proyectos*

- Al planificar una tarea, el planificador del proyecto debe tener conocimiento de las demás tareas que dependen de su realización, además de las que deben haber finalizado para que esta pueda iniciarse.
- Los tiempos de inicio y de fin se asignan manualmente y se puede violar la precedencia entre tareas.
- La asignación de los recursos a las tareas se realiza de forma intencionada, en función de las habilidades que posee el recurso; pero en la mayoría de los casos se descuida el hecho de que tengan asignadas otras tareas, o se utilizan más de los necesarios.
- El proceso de planificación es altamente dependiente del conocimiento del planificador del proyecto.

Estas deficiencias en el proceso de planificación actual atentan contra el tiempo de ejecución del proyecto e influyen negativamente en el balance de carga entre los recursos. Por ello se impone la búsqueda de una solución para estas problemáticas, que conlleva al planteamiento del siguiente **problema de investigación**. ¿Cómo automatizar la planificación de las tareas y asignación de recursos para disminuir el tiempo de desarrollo de un proyecto de software?

El problema anterior se enmarca dentro del **objeto de estudio** los métodos de optimización aplicados al Problema de Planificación de Proyectos con Recursos Limitados (RCPSP, por sus siglas en inglés). En la solución del mismo se traza como **objetivo general** desarrollar un algoritmo para la planificación de tareas y asignación de recursos de forma automática en un proyecto de desarrollo de software.

Mientras el **campo de acción** se centra en los métodos de optimización basados en Algoritmos Genéticos (GA, por sus siglas en inglés) aplicados al problema RCPSP.

Para alcanzar dicho resultado se definieron las siguientes tareas de investigación:

- Elaboración del marco teórico de la investigación a partir del estado del arte existente actualmente.
- Caracterización del problema RCPSP para determinar qué elementos intervienen en él y cómo representarlo.
- Caracterización de las metaheurísticas aplicadas a los problemas de planificación de proyectos para determinar aspectos relevantes que puedan ser utilizados en la solución.
- Definición de la codificación, los operadores de selección, cruzamiento y mutación así como el mecanismo de reemplazo de la población del GA para representar el problema en cuestión.
- Diseño e implementación de un GA para la planificación de tareas en un proyecto de software.
- Comprobación del desempeño del GA propuesto mediante casos de estudio que verifiquen su efectividad.

Para dar cumplimiento a estas tareas de investigación se emplearon **métodos científicos teóricos y empíricos**.

Métodos Teóricos:

Histórico-Lógico: Este método teórico se utilizó para el estudio crítico del estado del arte. Además, permitió conocer las tendencias más relevantes que existen sobre métodos de optimización que se han aplicado para resolver el problema RCPSP con el uso de GA.

Analítico-Sintético: Se utilizó para estudiar los múltiples elementos que conforman el RCPSP, establecer mentalmente la unión entre las partes previamente analizadas, descubrir sus características generales y las relaciones esenciales entre ellas.

Métodos Empíricos:

Experimental: Este método se utilizó para comprobar y fundamentar con casos de estudio que el uso de un GA para la planificación de tareas y asignación de recursos de forma automática en un proyecto de desarrollo de software, reduce el tiempo de ejecución del proyecto y mejora el balance de carga entre los recursos.

El documento está estructurado en 3 capítulos. En el Capítulo 1 se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. Además, se hace un estudio de técnicas metaheurísticas utilizadas para la resolución del problema RCPSP y se profundiza en el estudio de la metaheurística basada en GA. En el Capítulo 2 se propone un método de solución basado en GA y se exponen elementos del diseño del algoritmo desarrollado. Finalmente, en el Capítulo 3 se muestran y discuten los resultados obtenidos. Además, se exponen los resultados de las pruebas de aceptación realizadas al GA.

CAPÍTULO 1

Fundamentación Teórica

1.1. Introducción

En el presente capítulo se realiza un estudio sobre la planificación de proyectos de software, se analiza el RCPSP como una variante del Problema de Planificación de Proyectos (PSP, por sus siglas en inglés), se explican los elementos que lo conforman y el modelo matemático de su representación. Se describen métodos de optimización para la solución del RCPSP, basados en metaheurísticas, además de brindar especial atención a los GA.

1.2. Planificación de Proyectos de Software

Una de las actividades más necesarias en un proyecto es la planificación. Esta se encarga de organizar de manera coherente un grupo de acciones y actividades encaminadas a lograr un producto final con elevado valor de uso, que satisfaga las necesidades del cliente y que esté dentro de los plazos de tiempo estimados (SANTIESTEBAN QUINTANA, 2011).

El objetivo de la planificación de proyectos de software es proporcionar un marco de trabajo que permita al gestor de planificación hacer estimaciones razonables de recursos, costos y secuencia temporal de las tareas. La planificación ayuda en gran medida a aumentar la eficiencia del equipo de desarrollo, pues permite hacer un uso racional de los recursos y del tiempo.

La planificación define qué se debe hacer, en qué momento, así como los medios necesarios. Además, todo control es poco efectivo si no se compara con un plan previo (SALVADOR, 2001). Actualmente la planificación se sustenta en el uso de herramientas informáticas, específicamente en proyectos de gran alcance, en los que resulta imprescindible su uso.

1.2.1. Problema de Planificación de Proyectos de Software

Entre las técnicas de modelado de proyectos más empleadas en la actualidad se encuentra el PSP, el cual está formado por actividades, recursos, relaciones de precedencia y una función objetivo. Las primeras aproximaciones fueron desarrolladas en los años 1950. Algunos clásicos son (KELLEY JR, 1961; MALCOLM; ROSEBOOM; CLARK y FAZAR, 1959; MODER y PHILLIPS, 1964; PRITSKER; WAITERS y WOLFE, 1969). Seguidamente se describen cada uno de los elementos que conforman el PSP:

Actividades

Los proyectos cuentan con un número finito de actividades (tareas, trabajos u operaciones) que se necesitan ejecutar (CAPETILLO CORBEA, 2012). Para terminar un proyecto es necesario ejecutar cada actividad. Por otra parte, las actividades determinan los requerimientos de recursos para su ejecución. Lo que se debe hallar en un PSP es cuándo y cómo se va a procesar cada actividad.

Relaciones de precedencia

En los proyectos la planificación de una actividad generalmente está condicionada a la ejecución de otra actividad. Esta relación entre las actividades se denomina relación de precedencia. La forma más simple en que pueden aparecer las relaciones de precedencia es cuando una actividad no puede comenzar hasta que otra(s) finalice(n) (DEMEULEMEESTER, 2002). Las relaciones de precedencia se representan mediante un grafo dirigido y sin ciclos, en formato Actividad En Nodo (AON, por sus siglas en inglés) (MALCOLM; ROSEBOOM; CLARK y FAZAR, 1959). Cada actividad representa un nodo del grafo y cada arco dirigido una relación de precedencia entre dos actividades. Cada arco tiene asociado el tipo de relación de precedencia que existe entre las dos actividades que separa.

Recursos

Para realizar una actividad, generalmente es necesario tener algún tipo de consumo. En los PSP se modela ese consumo a través de los recursos que utiliza cada actividad. Los recursos se clasifican según su categoría, tipo y valor (KOLISCH y PADMAN, 2001). Las categorías más generales de los recursos son las de recursos renovables y no renovables.

Recursos renovables

La disponibilidad de estos recursos está limitada en cada unidad de tiempo. Como restricción, cada recurso renovable está disponible en una cierta cantidad constante en cada unidad de tiempo y su utilización no puede exceder esa cantidad (MEDRANO BROCHE, 2012). Como ejemplo podemos citar: mano de obra, máquinas, herramientas. En esta investigación solo se tendrán en cuenta los recursos que responden a esta categoría.

Recursos no renovables

Recursos que están limitados a lo largo del ciclo de vida del proyecto y una vez usados en el procesamiento de una actividad no pueden ser asignados a otra (BALLESTÍN GONZÁLEZ, 2002). Ejemplo de ellos lo constituyen: presupuesto, materia prima, energía.

Funciones objetivo

Todos los problemas de optimización consisten en encontrar una mejor solución respecto a un criterio determinado. Una función objetivo cuantifica la calidad de una solución, es el criterio por el cual se mide su bondad. Según las clasificaciones dadas en BRUCKER; DREXL; MÖHRING; NEUMANN y PESCH (1999), cada función objetivo define un modelo distinto, aunque el conjunto de soluciones posibles sea el mismo. Diferentes funciones objetivos pueden requerir técnicas distintas para resolver el problema (BALLESTÍN GONZÁLEZ, 2002).

La minimización de la duración del proyecto es probablemente la función objetivo más tratada y aplicada en el dominio de la planificación de proyectos (HARTMANN y BRISKORN, 2008; KOLISCH y PADMAN, 2001). La duración del proyecto se define como el tiempo que ha de transcurrir entre la ejecución de la primera y última tarea. Como el inicio del proyecto se sitúa usualmente en $t = 0$, minimizar la duración del proyecto equivale a minimizar el máximo de los tiempos finales de las actividades. Las funciones objetivo que dependen del tiempo en los PSP, son clasificadas como *regulares*[1] (MEDRANO BROCHE, 2012). En esta investigación se emplea siempre una función objetivo regular.

Variantes de los Problemas de Planificación de Proyectos de Software

La familia de los PSP presenta distintas variantes. El problema básico es conocido como RCPSP y se suele hacer referencia a él como la versión de "modo único". Esto se debe a que las actividades tienen un único modo de ejecución, con una duración determinada y un consumo dado de recursos. Este problema consiste en minimizar la duración del proyecto cuando las actividades que lo componen no pueden interrumpir su ejecución y están sujetas exclusivamente a relaciones de precedencia de tipo Fin-Inicio. Esto significa que hasta que no finalice la actividad precedente, no puede comenzar la siguiente.

Otra clase de problemas se genera cuando las actividades presentan distintos modos de ejecución. Esto quiere decir que cada una de las actividades puede presentar distintas posibilidades de realización. Un modo es una forma de ejecutar la actividad, que lleva asociada una duración y un consumo de recursos. A esta variante de los PSP se hace referencia como la versión "multi-modo" del problema (CERVANTES POSADA, 2010). En esta investigación sólo se analizará el problema conocido como RCPSP.

[1] *Una función objetivo regular, es aquella que al comparar dos secuencias S y $\hat{S}$ para un problema dado, si S difiere de $\hat{S}$ únicamente en el tiempo de finalización de una actividad j y se cumple que $f_j < \hat{f}_j$ para $f_j \in S$ y $\hat{f}_j \in \hat{S}$, se puede asegurar que la secuencia con el menor tiempo de finalización en esa actividad es al menos tan buena como la otra secuencia (tomando la evaluación de la función objetivo como criterio) y se dice que S domina a $\hat{S}$*

1.2.2. Problema de Planificación de Proyectos con Recursos Limitados

Dentro de los problemas de planificación de actividades, uno de los más estudiados es el de planificación de tareas con recursos limitados (BALLESTÍN GONZÁLEZ, 2002). El RCPSP consiste en establecer la secuencia de un conjunto de actividades $J = \{1, \ldots, n\}$ de un proyecto, sujetas a dos tipos de restricciones, las relaciones de precedencia y la cantidad de recursos disponibles para ejecutar las actividades en cada instante de tiempo. Se asumen que los recursos son renovables de diferentes tipos. Como función objetivo se tiene la minimización del tiempo de terminación del proyecto. Todas las magnitudes son asumidas enteras.

Representación del Problema de Planificación de Proyectos con Recursos Limitados

Un proyecto es representado por un grafo dirigido sin ciclo $G = (J, E)$ donde J es el conjunto de los nodos (actividades) y $E = \left\{(i, j) : i \in A_j\right\}$ el conjunto de los arcos (relaciones de precedencia), donde $A_j \subset J$ es el conjunto de las actividades antecesoras de j y por ende tienen que ejecutarse antes que esta. El conjunto de las actividades sucesoras inmediatas de j denotado por Suc_j. Cada actividad j tiene una duración d_j (peso del arco (i, j)) con $d_0 = d_n = 0$ ya que $j = 1$ y $j = n$ son nodos ficticios creados para representar donde comienza y termina el proyecto. Se denota como R_k al conjunto de recursos disponibles del tipo $k \in K$. Cada actividad j para procesarse consume de una cantidad q_{kj} de recurso. De ahora en adelante se denota por $s_j\left(f_j\right)$ al tiempo de inicio (fin) de la actividad j en una secuencia $(s_1, s_2, \ldots, s_n), (f_1, f_2, \ldots, f_n)$ (MEDRANO BROCHE, 2012). En la figura 1.1 se representa la red de un proyecto con siete actividades y dos recursos del mismo tipo disponibles.

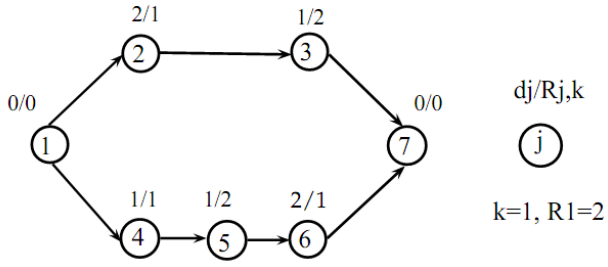

Figura 1.1. Red de un proyecto (tomado de (BALLESTÍN GONZÁLEZ, 2002))

Modelo matemático del Problema de Planificación de Proyectos con Recursos Limitados

Sea $P_t = \left\{j \in J : s_j \leq t \leq s_j + d_j\right\}$ el conjunto de las actividades que se están procesando en el instante de tiempo t y $U(S, t) = \sum_{j \in P} q_{jk}$ la cantidad de recurso que consumen estas. El RCPSP se puede definir de la siguiente manera:

$$mins_n \tag{1.2.1}$$

Sujeto a:

$$s_j - s_h \geq d_j \forall h \in A_j, j \in J \tag{1.2.2}$$

$$U(S,t) \leq |R_k|, t = 0,1,2,...,T-1, \forall k \in K \tag{1.2.3}$$

$$s_j \geq 0 \forall j \in J \tag{1.2.4}$$

La expresión 1.2.1 define la minimización de la duración del proyecto, que está dada por el tiempo de inicio de la actividad ficticia que pone fin a la red del proyecto. El grupo de restricciones 1.2.2 controla las relaciones de precedencia en la secuencia y las restricciones 1.2.3 controlan que no se exceda la cantidad de recurso disponible para cada instante t. Finalmente la restricción 1.2.4 define la no negatividad de los tiempos de inicio (BALLESTÍN GONZÁLEZ, 2002).

1.3. Metaheurísticas para la solución de Problemas de Planificación de Proyectos con Recursos Limitados

Los enfoques que se han usado para la solución del RCPSP son el exacto y el de aproximación. El primero garantiza una solución óptima, siempre que esta exista; sin embargo, en problemas grandes o muy complejos puede hacerse inviable ya que el tiempo de cómputo crece en forma exponencial con respecto al tamaño del problema. El segundo se traduce en metodologías heurísticas o metaheurísticas que si bien no garantizan la optimalidad, pueden dar muy buenas soluciones en tiempos razonables (MORILLO; MORENO y DÍAZ, 2014).

Las metodologías metaheurísticas son estrategias de alto nivel para explorar espacios de búsqueda mediante el uso de diferentes procedimientos. Normalmente, estos procedimientos son heurísticos. Las metaheurísticas generalmente se aplican a los problemas de optimización combinatoria para los que se sabe no existe una solución en tiempo polinómico, clasificados según su complejidad computacional como NP-hard (BLUM y ROLI, 2003). Ejemplos de metaheurísticas son: Búsqueda Tabú, Recocido Simulado, GA y Algoritmos Evolutivos.

Estas metodologías tienen como función tomar inicialmente una solución factible, para luego mejorarla usando heurísticas de mejoramiento embebidas en una estructura más general. La característica común de estos enfoques es el uso de mecanismos para evadir óptimos locales (DAZA; MONTOYA y NARDUCCI,

2009). En GLOVER y LAGUNA (1999) se define el término metaheurística como una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquellas que son normalmente generadas en una vecindad por optimalidad local.

Algunas de las razones de la popularidad y éxito de estos algoritmos metaheurísticos son su gran versatilidad, que permite su implementación para la solución de diversos tipos de problemas, la relativa simplicidad de sus conceptos subyacentes y un amplio espectro de variantes en sus aplicaciones (SANTANA; RODRÍGUEZ; LÓPEZ et al., 2004).

A continuación se enuncian y analizan los algoritmos metaheurísticos más relevantes que son usados en la resolución del RCPSP.

1.3.1. Recocido Simulado

El algoritmo Recocido Simulado (Simulated Annealing) fue introducido por (KIRKPATRICK; GELATT; VECCHI et al., 1983). Se inspira en el proceso de templado del metal, específicamente en el comportamiento molecular de un metal fundido cuando es sometido a un enfriamiento súbito. El algoritmo empieza desde una solución inicial aleatoria x_0. A partir de esta, se define una vecindad de soluciones a su alrededor y se evalúa x_0 en la función objetivo $f(x_0)$. Se genera una nueva solución x_1 perteneciente a la vecindad de x_0 y se evalúa la nueva solución $f(x_1)$. Si $f(x_1)$ es mejor que $f(x_0)$, entonces se acepta la nueva solución y se continúa el proceso a partir x_1. De lo contrario, la nueva solución se acepta con una cierta probabilidad, la cual depende de la diferencia de las funciones objetivo, $f(x_0)$ y $f(x_1)$, y de un parámetro T llamado temperatura, la cual debe definirse al inicio del algoritmo y que varía adaptativamente durante todo el proceso. Este procedimiento se repite para n iteraciones o hasta que se cumpla un criterio de parada (MORILLO; MORENO y DÍAZ, 2014).

En la expresión 1.3.1 se representa el esquema general de la probabilidad de aceptar una solución peor, en el caso de un problema de minimización, donde n es el número de iteraciones y $f(x_n)$ es la función objetivo evaluada en la solución x_n.

$$P(\text{aceptar una solución peor}) = e^{\frac{f(x_{n-1})-f(x_n)}{T}} \tag{1.3.1}$$

La expresión 1.3.1 muestra que mientras mayor sea la diferencia entre $f(x_{n-1})$ y $f(x_n)$ menor será la probabilidad de aceptación de una nueva solución que no mejore la actual. Para un problema de maximización, el numerador del exponente en la expresión 1.3.1 deberá escribirse como $f(x_n) - f(x_{n-1})$.

La temperatura, por lo general, empieza con un valor grande, lo que permite aceptar con mayor probabilidad soluciones que no mejoran, y con ello explorar el espacio factible en las primeras etapas del algoritmo, evitando estancamientos en óptimos locales. Esta temperatura se reduce de manera gradual con la evolución del algoritmo, mediante un parámetro r, menor que 1, cuyo valor representa la velocidad de enfriamiento del algoritmo, cambiando su perfil de exploración a uno de explotación. Este procedimiento se repite hasta

que se cumpla un criterio de parada. Generalmente, en el Recocido Simulado se define una temperatura mínima, denominada temperatura de congelación. El algoritmo debe parar cuando esta se alcance por primera vez. En BROOKS y MORGAN (1995) y KIRKPATRICK; GELATT; VECCHI et al., (1983) se exponen los resultados de emplear un algoritmo Recocido Simulado en la solución de problemas RCPSP.

1.3.2. Búsqueda Tabú

El desarrollo de esta metaheurística es atribuido a GLOVER (1989), es una extensión de la búsqueda local que trata de no quedar atrapado en un óptimo local, mediante el uso de información que recauda a medida que el algoritmo avanza. Por esta razón, la búsqueda tabú es uno de los enfoques más representativos de los algoritmos con memoria (MORILLO; MORENO y DÍAZ, 2014).

El algoritmo empieza con una solución inicial aleatoria x_0. Posteriormente, se define un vecindario inicial de soluciones alrededor de x_0 y una lista tabú. Luego, mediante reglas de movimiento, se generan nuevas soluciones las cuales se analizan para determinar si se seleccionan. La lista tabú se construye con las soluciones históricamente visitadas, las cuales quedan prohibidas (definidas como tabú), de manera que restringen el vecindario actual. Esta estrategia garantiza, hasta cierto punto, no volver a seleccionar aquellas soluciones visitadas antes, y así evitar quedar atrapado en un óptimo local. Vale destacar que la lista tabú es de tamaño limitado, cuando este se alcanza, se elimina la solución más antigua (que deja de ser tabú) y se adiciona la nueva.

Para aumentar la eficiencia en la etapa de explotación, generalmente, en la lista tabú no se almacenan directamente las soluciones, sino sus características inherentes. Así, cualquier solución que comparta algunas características que estén en la lista tabú no se selecciona. Esta regla solo tiene una excepción: cuando la nueva solución supera la mejor solución encontrada hasta el momento, se le asigna una probabilidad de ser seleccionada, denominada nivel de aspiración (ibíd.).

En ARTIGUES; MICHELON y REUSSER (2003) se usan técnicas de inserción para desarrollar un procedimiento de búsqueda tabú en la solución del RCPSP que, de manera iterativa, selecciona una actividad de una solución y la elimina de la programación; luego, la reinserta mediante el uso de reglas de flujo de redes. Tanto la actividad seleccionada como sus predecesores y sucesores se adicionan a la lista tabú. En BAAR; BRUCKER y KNUST (1999) se diseñan dos esquemas de búsqueda tabú para resolver el RCPSP: el primero genera vecindarios con ayuda de la información de la ruta crítica y el segundo se basa en la generación de vecinos. Mientras que KLEIN (2000) propone la llamada Búsqueda Tabú Reactiva, basada en la representación serial de actividades mediante un Esquema Generador de Soluciones (SGS, por sus siglas en inglés). Las vecindades de soluciones se obtienen con movimientos de intercambio que incluyen desplazamientos de los predecesores y sucesores de las actividades intercambiadas.

1.3.3. Greedy Randomized Adaptive Search Procedure

Los Greedy Randomized Adaptive Search Procedure (GRASP, por sus siglas en inglés) fueron descritos en FEO y RESENDE (1989). Es útil en problemas que pueden descomponerse en una secuencia de

decisiones que deben tomarse en varias etapas. Este método consta de dos fases fundamentales, una fase constructiva y una de mejora (MORILLO; MORENO y DÍAZ, 2014).

En la fase constructiva se generan soluciones factibles de manera iterativa, donde en cada iteración se incorpora un elemento a la solución, cuya selección se basa en el valor de la función de aptitud parcial de cada alternativa disponible. En lugar de seleccionar directamente la mejor alternativa (metodología avariciosa, Greedy) selecciona de manera aleatoria el siguiente elemento a incorporar como parte de la solución, pero asigna una probabilidad en proporción a la función de aptitud. Una vez que se elige el elemento a adicionar en la solución, los valores de las alternativas se adaptan a la nueva situación.

Una vez construida la solución, se le aplica una búsqueda local o una función de mejora, estos dos pasos se repiten un número de veces predeterminado. Al emplear componentes aleatorios, en general se obtendrán diferentes soluciones en cada iteración (BALLESTÍN GONZÁLEZ, 2002).

1.3.4. Búsqueda Dispersa

La Búsqueda Dispersa (Scatter Search) fue propuesta en GLOVER (1998) y descrita en LAGUNA; MARTI y MARTÍ (2003). Esta metaheurística se considera un proceso evolutivo donde se construye un conjunto de referencia de soluciones buenas, pero dispersas, es decir, soluciones distantes entre sí o significativamente diferentes. Este conjunto evoluciona combinando dos o más soluciones para producir otras mejores, pero mantiene un significativo grado de dispersión (MORILLO; MORENO y DÍAZ, 2014).

Según la descripción que se ofrece en BALLESTÍN GONZÁLEZ (2002), la búsqueda dispersa consta de 5 elementos:

- Un método de generación diversificada para generar una colección de soluciones de prueba diversas.
- Un método de mejora que transforma una solución de prueba en una o más soluciones mejoradas.
- Un método para construir, mantener y actualizar el conjunto conformado por un número pequeño de las mejores soluciones encontradas.
- Un método que, a partir del conjunto de referencia, genera subconjuntos con el fin de combinarlos para crear nuevas soluciones.
- Un método de combinación de soluciones que transforma un subconjunto dado de estas en una o más soluciones. Este método de combinación es el equivalente al operador genético de cruce en los GA.

Según DEBELS; DE REYCK; LEUS y VANHOUCKE (2006), este algoritmo se ha implementado para resolver el RCPSP con buenos resultados, situándose en la literatura como un método tan competitivo como los GA para este tipo de problemas.

1.3.5. Algoritmos Genéticos

Los GA introducidos por Holland en HOLLAND (1975), proporcionan un mecanismo de búsqueda robusto, para resolver problemas de optimización. Se inspiran en la teoría darwiniana de la evolución, que se trasladada al campo de optimización. Se puede interpretar así: cada individuo (solución) tiene asociado un valor de su aptitud, representado por su valor en la función objetivo. Las diferentes generaciones de individuos evolucionan mediante la selección y operadores genéticos. Se toman varias soluciones simultáneamente para lograr un proceso de búsqueda en paralelo (MORILLO; MORENO y DÍAZ, 2014).

La implementación de estos algoritmos se basa en la representación de un conjunto de soluciones en un código genético, expresado en lenguaje computacional. En la literatura se evidencia que las dos formas de codificar las soluciones más empleadas son: la representación binaria, propuesta en HOLLAND (1975), y la representación en valor real. La diferencia de estos dos métodos, consiste en la manera de definir los denominados operadores genéticos.

En este algoritmo se parte de una población inicial, que generalmente se crea de manera aleatoria. De esta población inicial se seleccionan los padres de la siguiente generación, para lo cual, usualmente se eligen los individuos de menor valor en la función de aptitud (para problemas de minimización) con un alta probabilidad. Sin embargo, también se da cabida a soluciones de mayor valor (soluciones malas), aunque con menor probabilidad, para permitir la exploración en diferentes zonas de la región factible y dar diversidad al algoritmo de búsqueda. El operador genético más importante es el cruce (MORILLO; MORENO y DÍAZ, 2014). La mutación cumple un papel secundario aportando, casualmente, cambios aleatorios para permitir la exploración de la búsqueda. Luego, se asignan parejas de padres de manera aleatoria y se procede a usar los operadores genéticos previamente definidos para hallar nuevas soluciones (hijos). Después de la trasferencia de genes y la generación de hijos se usa algún criterio de selección para elaborar la nueva generación. El proceso se repite hasta cumplir con un criterio de parada. Los algoritmos genéticos son muy utilizados para resolver problemas de planificación de proyectos. En LANCASTER y OZBAYRAK (2007) se describe detalladamente la implementación de estos algoritmos en este tipo de problemas.

1.3.6. Optimización de la Colonia de Hormigas

La idea en la que se sustenta este algoritmo fue planteada en DORIGO (1992). Su algoritmo de optimización es un metaheurístico de poblaciones que intenta reproducir el mecanismo utilizado por las hormigas para localizar el alimento e informar a la colonia donde se encuentra. La ruta seguida por las hormigas tiene la característica de ser la de menor distancia desde su nido hasta el alimento. Este algoritmo se cataloga como un procedimiento de construcción seudo-aleatoria (MORILLO; MORENO y DÍAZ, 2014).

Estableciendo una analogía con el comportamiento natural de las hormigas, inicialmente, estas se desplazan de manera aleatoria, dejando por cada camino recorrido una cierta cantidad de feromonas. Las siguientes hormigas se desplazarán aleatoriamente pero con mayor probabilidad de ir por los caminos que tengan mayor concentración de esta sustancia. Estas, a su vez, dejan su propia feromona, la cual, con el tiempo, se evapora paulatinamente; es decir, aquellas rutas que no se visiten recurrentemente tienden a desaparecer. De

esta manera, algunos caminos se vuelven más atractivos por su alta concentración de feromona.

El principio de este algoritmo consiste en que la probabilidad de que una hormiga seleccione una ruta está condicionada por el número de hormigas que la haya seleccionado previamente y de su distancia (BALLESTÍN GONZÁLEZ, 2002). Paulatinamente, las hormigas prefieren usar las rutas más cortas. El concepto de mínima distancia entre el nido y la fuente de alimento se adapta a la función de aptitud, cuyo mínimo valor indicará la ruta a seleccionar. En este algoritmo se construyen soluciones iterativamente.

En MORILLO; MORENO y DÍAZ (2014) se define la probabilidad de incorporar el elemento j de la solución elaborada hasta el elemento i, de la siguiente manera:

$$P_{i,j} = \begin{cases} \frac{[\tau_{i,j}(t)]^{\alpha} \cdot [\eta_{i,j}]^{\beta}}{\sum_{u \notin M_k} [\tau_{iu}(t)]^{\alpha} \cdot [\eta_{iu}]^{\beta}} & \text{si } j \notin M_k \\ 0 & \text{en otro caso} \end{cases} \tag{1.3.2}$$

donde:

$\eta_{i,j}$: parámetro que representa una cierta "visibilidad", relacionada con la función de aptitud.

$\tau_{i,j}(t)$: rastro de feromona, que depende del tiempo.

α: importancia relativa de la feromona.

β: importancia relativa de la "visibilidad".

M_k: memoria asociada a la hormiga k.

En MERKLE; MIDDENDORF y SCHMECK (2002) se desarrolló por primera vez una aplicación de la optimización de la colonia de hormigas al RCPSP. Bajo este enfoque, una sola hormiga se define como una aplicación completa para elaborar una solución factible. En cada iteración, la siguiente actividad a seleccionar depende del valor promedio del tiempo de inicio más tardío de cada actividad. Dicho promedio será la feromona que representa el aprendizaje y el efecto de las anteriores hormigas (MORILLO; MORENO y DÍAZ, 2014).

1.4. Generación de Soluciones del Problema de Planificación con Recursos Limitados

Los SGS son el núcleo de la mayoría de los procedimientos heurísticos para la solución del RCPSP (KOLISCH y S. HARTMANN, 1999). Estos construyen una solución factible extendiendo paso a paso una secuencia parcial de actividades, que inicialmente asigna el tiempo de inicio igual cero a la primera actividad planificada de la secuencia. Una secuencia parcial es aquella donde únicamente un subconjunto de las n actividades de un proyecto han sido planificadas (MEDRANO BROCHE, 2012).

Existen dos SGS: el Esquema Generador de Soluciones en Serie (SSGS, por sus siglas en inglés) y el Esquema Generador de Soluciones en Paralelo (PSGS, por sus siglas en inglés). Ambos esquemas permiten encontrar soluciones factibles para el RCPSP. Según KOLISCH (1996), el SSGS construye soluciones

activas[2], tiene mejor ejecución ante instancias grandes y el conjunto de soluciones generadas por este algoritmo siempre contendrá la solución óptima si se considera únicamente el problema no restringido en cuanto a recursos. En cambio PSGS genera soluciones sin retraso, el conjunto de soluciones sin retraso es un subconjunto de las soluciones activas por lo que tiene, en general, un cardinal menor. Pero tiene una desventaja importante, y es que puede no contener ninguna solución óptima para una medida regular, algo que no ocurre con el conjunto de las soluciones activas. Por ello en este trabajo se abordará el SSGS.

1.4.1. Esquema Generador de Soluciones en Serie

El SSGS, ver Algoritmo 1, consiste en $g = 0, 1, 2, ...$ estados, en cada uno de ellos las actividades de un proyecto son planificadas sin violar las restricciones de recursos y las relaciones de precedencia. Asociado a cada estado g existen dos conjuntos disjuntos de actividades. Un conjunto $D_g = \left\{ j \in J \setminus S_g : A_j(t) \subseteq S_g \right\}$ de las actividades que pueden ser seleccionadas para planificarse, donde $A_j(t)$ es el conjunto de las actividades antecesoras de la actividad $j \in J$ planificadas en el instante $t = 1, \ldots, T$. El otro conjunto S_{ec_g}, es el de las actividades que ya han sido planificadas en el estado g. Se tiene que $\bar{R}_k(t) = |R_k| - U_k(S, t)$ es el número de recursos del tipo $k \in K$, disponibles en el instante t, y que $F_g = \left\{ f_j : j \in S_g \right\}$ es el conjunto de los tiempos de fin de todas las actividades que se han planificado (MEDRANO BROCHE, 2012).

Algoritmo 1 Esquema Generador de Soluciones en Serie (SSGS)

1: INITIALIZE: $g = 0, Sec_0 = \{1\}$
2: **while** $S_g \neq J$ **do**
3: $g = g + 1$
4: CALCULATE: $D_g, F_g, \bar{R}_k(t)$ TAL QUE $t \in F_g, k \in K$
5: SELECT: $j \in D_g$
6: CALCULATE: $es_j = \max\{f_h\}, h \in A_j$
7: CALCULATE: $s_j = \max\left\{ t : t \geq es_j, r_{jk} \leq \bar{R}_k(\tau), \forall k \in K, \tau \in \left[t, t + d_j\right] \cap F_g \right\}, h \in A_j$
8: CALCULATE: $f_j = s_j + d_j$
9: INSERT: $Sec_g = Sec_{g-1} \cup \{j\}$
10: $C_{max} = max_j\left(f_j\right), j \in J$
11: **end while**

1.5. Trabajos relacionados

Existen varios trabajos donde se relaciona la metaheurística GA con el problema RCPSP (LANCASTER y OZBAYRAK, 2007). Todos usan los operadores básicos de los GA: selección, mutación y cruzamiento.

[2] *Las soluciones activas son aquellas donde una actividad no puede comenzar a ejecutarse más tarde de lo que lo hace, sin que retrase el inicio de otra. Cuando se tiene un RCPSP donde se minimiza la duración del proyecto, la solución óptima del problema es una solución activa, lo que representa una ventaja importante al usar SSGS*

Las variantes de utilización de estos operadores son muy diferentes, así como la definición de la función objetivo, pero todas coinciden en que con el uso de este algoritmo se obtienen buenos resultados.

Un ejemplo de la relación entre las restricciones que plantea el RCPSP y la codificación del GA se observa en NARVÁEZ MOLINA (2010). En la codificación cada valor del gen en el cromosoma representa el número de la tarea y la posición del gen dentro de la cadena especifica el orden de ejecución de dicha tarea. En este se obtiene una planificación de las tareas donde solo se tiene en cuenta la restricción impuesta por la precedencia entre actividades. Por lo que, una vez obtenida la mejor solución parcial, se describe otro procedimiento no heurístico para determinar los tiempos de inicio y de fin de las tareas del cronograma. Este enfoque, aunque simplifica el tiempo de cómputo, compromete su efectividad en proyectos con más de 30 tareas.

Por otra parte en ZOULFAGHARI; NEMATIAN; MAHMOUDI y KHODABANDEH (2013) se presenta un GA que se utiliza para resolver el RCPSP a gran escala y mejorar las soluciones. En este trabajo la codificación se realiza a través de cromosomas en cuyos genes se representan los tiempos de inicio de las actividades. Se asume que la posición del gen dentro de la cadena hace referencia al número de la actividad. Los operadores de cruzamiento y mutación tienen en cuenta tanto las restricciones de precedencia como la disponibilidad de recursos. Los resultados mostrados con este procedimiento arrojan buenas soluciones para proyectos de hasta 30 tareas o más de 120 tareas. Sin embargo, para proyectos de aproximadamente 60 actividades, las soluciones se colocan en desventaja con respecto a las obtenidas con otros enfoques de solución.

En los trabajos publicados por FRANCO (2013) y HARTMANN (1997) se emplea un SGS para obtener los cromosomas de la población inicial. Esta característica acelera la convergencia del algoritmo (GIL LONDOÑO, 2006). Los operadores de cruzamiento utilizados son de un punto y de dos puntos. El algoritmo propuesto en FRANCO (2013) presenta mejores resultados con el operador de cruzamiento de dos puntos, sin embargo el tiempo de cómputo es considerablemente mayor que su variante de un solo punto.

1.6. Herramientas y Metodologías

1.6.1. Entorno de desarrollo

El Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) a utilizar es QtCreator. Las primeras versiones datan de 1996. Puede funcionar en varias plataformas: Windows, Linux y Mac. Con este IDE se han desarrollado aplicaciones tales como: Autodesk, Google Earth, Adobe Photoshop Elements, Skype, KDE, Avidemux, LyX, Mathematica (versión Linux), Doxygen, VirtualBox, entre otros. Se distribuye bajo ediciones comerciales o bien de código abierto. Es un entorno potente, con una amplia gama de funcionalidades que facilitan su uso. Utiliza el lenguaje de programación orientado a objetos C++, lo cual tiene concordancia con el lenguaje y la biblioteca que se utiliza. La versión a utilizar es QtCreator 3.0.0.

1.6.2. Metodología

Para el desarrollo del algoritmo se va a utilizar la Programación Extrema (XP, por sus siglas en inglés), una metodología de procesos ágiles para el desarrollo de software. Es de las metodologías más exitosas debido a que se centra en potenciar las relaciones entre los desarrolladores para lograr el éxito, promoviendo el trabajo en equipo y la comunicación con el cliente, lo cual propicia un buen entorno de trabajo. Algunas de las características que los distinguen son:

- Desarrollo iterativo e incremental.
- Integración del equipo de programación con el usuario.
- Propiedad del código compartida.
- Corrección de errores.

1.6.3. Lenguaje de programación

El lenguaje a utilizar es C++. Es una extensión del lenguaje C con mecanismos que permiten la manipulación de objetos. Contiene los paradigmas de programación estructurada y orientada a objetos, por lo que se suele decir que es un lenguaje multiparadigma. Además, es de los de alto nivel el más cercano al lenguaje de máquina. Esta característica le proporciona mayor velocidad de ejecución, con respecto al resto de los lenguajes de alto nivel.

1.6.4. Bibliotecas

Biblioteca de Inteligencia Artificial

Esta biblioteca fue desarrollada en la UCI y contiene un módulo con la implementación de un GA (PALOMINO, 2012), su objetivo principal es la resolución de problemas de optimización. La biblioteca está implementada en C++ y la UCI posee todos los derechos de autoría de esta investigación. La principal característica de este Algoritmo Genético Simple (SGA, por sus siglas en inglés) que se puede destacar, es su fácil adaptación a diferentes codificaciones. Por otro lado, este módulo se encuentra en su primera versión y no cuenta con todas las funcionalidades requeridas para este trabajo.

GAlib

GAlib es una biblioteca de GA desarrollada en C++. Ha sido utilizada en diferentes configuraciones: DOS/Windows, Windows NT/95, Mac y UNIX. Contiene los elementos para utilizar la programación en paralelo. Es una biblioteca de código abierto, que tiene la implementación de los operadores básicos de los GA y una gran cantidad de codificaciones. Además, permite obtener valores estadísticos como: desviación, generación donde fue encontrado el mejor individuo, entre otros.

Biblioteca de Algoritmos Genéticos

Es una biblioteca de código abierto implementada en C++. Tiene dos tipos de licencia una privativa y otra de código abierto, esta última en su versión 1.4. Esta biblioteca contiene las clases para la implementación de un SGA y la implementación de sus operadores básicos. Además, posee una gran cantidad de codificaciones y permite obtener estadísticas de su ejecución como son el mejor valor arrojado por la función objetivo de un individuo, así como el promedio de este valor para una población, entre otros.

UNGenético

UNGenético es una biblioteca para el desarrollo de GA (VELASCO, 2002). Su objetivo principal es fusionar codificaciones para resolver problemas de optimización. Esta biblioteca es de código abierto e implementada en C++. Es fácil de adaptar y configurar para representar la codificación del problema RCPSP que se propone en esta investigación. Cuenta con la implementación de los operadores clásicos de los GA. Asimismo, posee el cálculo de medidas que permiten observar el rendimiento del algoritmo implementado tales como la desviación, la generación donde fue encontrado el mejor individuo, entre otros. Por estas características se decide utilizar en esta investigación.

1.7. Conclusiones del capítulo

En este trabajo se siguen los fundamentos propuestos por la teoría de planificación de proyectos. Para la solución del RCPSP se elige la metaheurística GA, ya que actualmente es uno de los algoritmos más utilizados y reconocidos para este fin. Este algoritmo evita, a través del diseño de sus operadores, la convergencia prematura hacia una solución óptima local. Además, se valora el empleo de un SSGS para generar la población inicial del GA, basado en la experiencia descrita en los trabajos revisados.

CAPÍTULO 2

Propuesta de solución

2.1. Introducción

En este capítulo se brindan detalles de la representación del problema RCPSP para esta investigación. Se describen los elementos a tener en cuenta para la construcción de la solución aplicando GA. Además, se presentan detalles de la Ingeniería de Software y pseudocódigos del algoritmo propuesto.

2.2. Descripción de la organización

Un proyecto está compuesto por actividades y recursos humanos. De cada actividad se conoce el nombre, el tiempo de duración (en horas), el conjunto de sus actividades predecesoras y la cantidad de recursos por tipo necesarios para su ejecución. Resulta conveniente acotar que cada tipo corresponde al rol desempeñado por un recurso. Las relaciones de precedencia entre las tareas son del tipo fin - inicio.

Se asigna para la realización de cada actividad, uno o más recursos humanos. Cada recurso tiene un rol asociado, así como un fondo de tiempo (en horas) que representa la cantidad de tiempo que puede dedicar al desarrollo de las tareas del proyecto. Además del tiempo ocioso, que representa, para cada instante del cronograma de ejecución, el tiempo que el recurso se encuentra sin realizar ninguna labor.

Se asume que la duración de las actividades, los requerimientos de recursos y la disponibilidad de cada tipo de recurso son cantidades no negativas; además de que toda tarea tiene una demanda de recursos menor o igual al total de recursos con los que cuenta el proyecto, para cada tipo de recurso.

El GA propuesto es factible solo para proyectos que cumplan con las características anteriores, bajo otras condiciones requiere modificaciones para poder aplicarse. Como salida de este algoritmo se obtienen secuencias de planificación de tareas, con los recursos asociados de forma tal que la duración del proyecto sea la menor posible.

2.3. Representación de la Planificación de Proyectos a través de Algoritmo Genético

Existen varias representaciones de las soluciones del RCPSP, en especial a partir de la incorporación de las técnicas metaheurísticas como alternativa de resolución. Estas representaciones condicionan las técnicas que se van a poder emplear dentro de un algoritmo basado en una de ellas (BALLESTÍN GONZÁLEZ, 2002).

En la presente investigación se diseña un GA que maneja la planificación de proyecto como una lista de actividades. Una lista de actividades es una permutación de actividades $\lambda = (j_1, j_2, \ldots, j_n)$ válida respecto a las relaciones de precedencia, o sea, a cada actividad le corresponde una posición u orden mayor que el de cualquiera de sus predecesoras. De este modo, si $i = j_P$ diremos que la actividad i está en la posición p, o que el orden de i en λ es p. Se tiene además que $j_1 = 1$ y $j_n = n \; \forall \lambda$.

Habitualmente la población inicial de un GA se obtiene a partir de un conjunto de individuos generados al azar, sin embargo en esta investigación se propone como población inicial un grupo de soluciones válidas generadas a partir del SSGS descrito en 1. En los pocos trabajos que abordan el uso de técnicas heurísticas o de optimización local en la obtención de la población inicial, se constata que esta inicialización no aleatoria puede acelerar la convergencia del algoritmo (GIL LONDOÑO, 2006). Seguidamente se detallan los operadores genéticos que se emplean en el GA diseñado.

2.3.1. Codificación

La codificación de la planificación de un proyecto se representa a través de I (individuo), con una lista de valores enteros (cromosoma), donde cada uno de sus elementos (genes) hace referencia a una tarea. La dimensión de I coincide con la cantidad de actividades del proyecto en cuestión. La lista de actividades definida en 2.3.1 representa una planificación de las actividades del caso de estudio, Figura 1.1. Obsérvese que en este caso, por ejemplo, la actividad 3 se encuentra en la posición 4, o lo que es lo mismo, el orden de 3 en I es 4.

$$I = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \left[\begin{matrix} 1 & 2 & 4 & 3 & 5 & 6 & 7 \end{matrix} \right] \end{array} \tag{2.3.1}$$

2.3.2. Selección

El operador de selección establece cuántas copias se crearán de cada individuo en la siguiente generación (VELASCO, 2002). En el caso en cuestión se emplea el operador de selección estocástica con reemplazo, para lo cual se crea una ruleta 2.1 con tantas casillas como individuos tenga la población; el ángulo de cada casilla es proporcional a la probabilidad de supervivencia.

Se utiliza la ruleta tantas veces como individuos tenga la población y se crea una copia de cada individuo ganador. De este modo se mantiene la diversidad entre los individuos de la población y se evita la convergencia prematura del algoritmo.

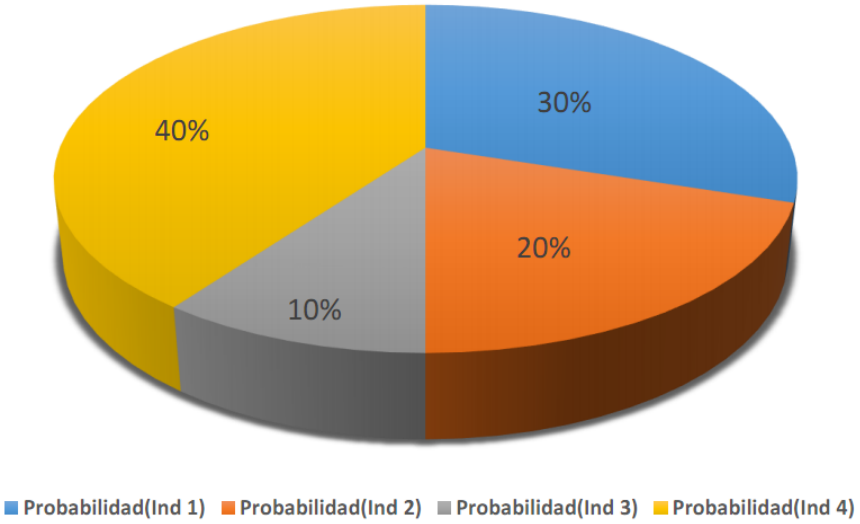

Figura 2.1. Ruleta con operador de probabilidad proporcional

2.3.3. Cruzamiento

Para el proceso de cruce se elige la generación de dos nuevos individuos (hijos) a partir de dos individuos de la actual generación (padres). El operador que se emplea fue propuesto en NARVÁEZ MOLINA (2010), es probabilístico y resulta ser una variación del operador de cruzamiento por un punto. Con su utlización se garantiza que los nuevos individuos obtenidos también cumplan las restricciones de precedencia impuestas por el problema. El procedimiento para su utilización es el que se describe a continuación:

- Se genera un número aleatorio u entre 1 y n, el cual representa el punto de cruce entre los cromosomas.
- El primer hijo hereda los genes del padre de izquierda a derecha hasta u, completando la secuencia genética con los genes de la madre, de izquierda a derecha, que aún no formen parte de la misma.
- De igual modo, el segundo hijo hereda los genes de la madre de izquierda a derecha hasta u, y completa la secuencia genética heredando los genes del padre, de izquierda a derecha, que aún no formen parte de la misma.

En 2.3.2 se muestra cómo se efectúa el cruce entre dos individuos que codifican soluciones válidas para el caso de estudio propuesto en 1.1. Nótese que el valor aleatorio u para el punto de cruce es 2.

$$
\begin{array}{rc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
Padre = & \left[\begin{array}{ccccccc} 1 & 2 & 4 & 3 & 5 & 6 & 7 \end{array}\right]
\end{array}
$$

$$
\begin{array}{rc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
Madre = & \left[\begin{array}{ccccccc} 1 & 4 & 5 & 2 & 3 & 6 & 7 \end{array}\right]
\end{array}
$$

$$
\begin{array}{rc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
Hijo1 = & \left[\begin{array}{ccccccc} 1 & 2 & 4 & 5 & 3 & 6 & 7 \end{array}\right]
\end{array}
\tag{2.3.2}
$$

$$
\begin{array}{rc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
Hijo2 = & \left[\begin{array}{ccccccc} 1 & 4 & 2 & 3 & 5 & 6 & 7 \end{array}\right]
\end{array}
$$

2.3.4. Mutación

La mutación juega un papel fundamental en cualquier GA debido a que brinda la posibilidad de salir de un entorno de búsqueda monótono y pasar a otro que probablemente tenga entre sus habitantes mejores soluciones. El operador propuesto en este trabajo se basa en el descrito por NARVÁEZ MOLINA (2010), al igual que el diseñado para el cruce es probabilístico y garantiza que el nuevo individuo que se obtenga cumpla con las restricciones de precedencia impuestas por el problema. Los pasos para lograr la mutación son los siguientes:

- Generar un número aleatorio u entre 1 y $n - 1$.
- Si la actividad ubicada en el orden u de la secuencia genética no es predecesora de la actividad situada en la posición $u + 1$ de la misma secuencia, entonces se intercambian ambas actividades. En caso contrario volver al paso anterior.

Este simple procedimiento se ejemplifica en 2.3.3 para el primer hijo obtenido en 2.3.2. En este caso el valor aleatorio u es 4.

$$
\begin{array}{rc}
 & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
Hijo1 = & \left[\begin{array}{ccccccc} 1 & 2 & 4 & 5 & 3 & 6 & 7 \end{array}\right]
\end{array}
$$

$$Hijo1Mutado = \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \left[1 \right. & 2 & 4 & 3 & 5 & 6 & \left. 7 \right] \end{matrix} \tag{2.3.3}$$

2.3.5. Función Objetivo

Un aspecto importante en los algoritmos genéticos es la determinación de la calidad o fortaleza de los individuos en la población, precisamente la función objetivo es la que determina la aptitud del individuo. Dada la naturaleza del RCPSP, la función objetivo empleada en esta investigación es minimizar el tiempo total de duración del proyecto. Su representación se plantea a continuación:

$$min f_n \tag{2.3.4}$$

Sujeto a:

$$f_n = \max \left\{ f_j, \forall f_j \in F_g \right\} \tag{2.3.5}$$

Para obtener el tiempo total de planificación del proyecto de cualquier individuo se ha utilizado una variante del SSGS, descrita en el Algoritmo 2. En este caso se omite la generación del conjunto de las actividades elegibles D_g para cada estado. Esto es debido a que ya se cuenta con la secuencia de actividades planificadas, denotada por P, por lo que solo es necesario encontrar los tiempos de finalización de cada una de las tareas. Además, se asignan los recursos necesarios para realizar las actividades, a partir de una estrategia de selección que prioriza aquellos recursos que llevan más tiempo sin realizar alguna tarea.

Algoritmo 2 Calcular Función Objetivo

1: INITIALIZE: $g = 0, Sec_0 = \{1\}$
2: **while** $S_g \neq J$ **do**
3: $g = g + 1$
4: CALCULATE: $F_g, \bar{R}_k(t)$ TAL QUE $t \in F_g, k \in K$
5: SELECT: $j \in P$
6: CALCULATE: $es_j = \max\{f_h\}, h \in A_j$
7: CALCULATE: $s_j = \max\left\{t : t \geq es_j, r_{jk} \leq \bar{R}_k(\tau), \forall k \in K, \tau \in \left[t, t + d_j\right] \cap F_g\right\}, h \in A_j$
8: CALCULATE: $f_j = s_j + d_j$
9: CALCULATE: R_{kj}
10: UPDATE LAST JOB: R_{kj}
11: INSERT: $Sec_g = Sec_{g-1} \cup \{j\}$
12: $C_{max} = max_j\left(f_j\right), j \in J$
13: **end while**

2.3.6. Elitismo

Uno de los métodos más utilizados para mejorar la convergencia de los GA es el elitismo. Consiste básicamente en incorporar una élite de r miembros de la población i a la población $i + 1$, sin pasar por la población intermedia. El elitismo puede mejorar el funcionamiento de los GA al evitar que se pierda la mejor solución entre dos iteraciones consecutivas (ARRANZ DE LA PEÑA y PARRA TRUYOL, 2014). Comúnmente, el tamaño de la élite es bastante pequeño (1 ó 2 para $n = 50$) para evitar la convergencia prematura. En esta investigación el tamaño para la élite se ha fijado en 1.

2.4. Pseudocódigo

Algoritmo 3 Optimización, de la clase Algoritmo Genético

```
1:  BeginOptimization()
2:  repeat
3:      procedure IterateOptimization
4:          Selection()
5:          AssignPartner()
6:          Crosover()
7:          Mutation()
8:          Elitism()
9:          AssignProbability()
10:         FindElite()
11:         generation := generation + 1
12:         UpdateMeasures()
13:     end procedure
14: until Stop() == true
```

2.5. Ingeniería de Software

2.5.1. Especificación de Requisitos

Requisitos funcionales

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos también declaran explícitamente lo que el sistema no debe hacer (SOMERVILLE, 2005).

El proceso de captura de requisitos concluyó con un total de 5 requisitos funcionales, los cuales se enumeran a continuación:

RF 1. Cargar base de datos del caso de estudio a probar

RF 2. Obtener secuencias de planificación aleatorias con el SSGS

RF 3. Cargar archivo con las secuencias obtenidas con el SSGS

RF 4. Determinar secuencias de planificación con el GA

RF 5. Crear reporte

Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el procesos de desarrollo y estándares. A menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (SOMERVILLE, 2005). Seguidamente se detallan los requisitos no funcionales a tener en cuenta en el desarrollo de la aplicación.

RnF 1. Requisitos de usabilidad:

- El sistema se debe desarrollar para plataformas de escritorio.
- Para el desarrollo: PC Intel Pentium 4 o superior, CPU 1.6 GHz o superior, 1 Gb RAM, teniendo en cuenta el nivel de procesamiento necesario que depende de la cantidad de tareas que contengan los proyectos.

RnF 2. Requisitos de soporte:

- Los usuarios deben tener conocimiento sobre la planificación de proyectos.
- Las normas de codificación estarán diseñadas para un fácil entendimiento del código de la planificación, guiadas al desarrollo de futuras versiones.

RnF 3. Requisitos de diseño de implementación:

- Para la implementación del sistema se debe utilizar el lenguaje de programación C++.
- Para la implementación del sistema se debe utilizar el entorno de desarrollo QtCreator versión 3.0.0.

2.5.2. Fase de Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (LETELIER, 2012).

Descripción de Historias de usuario

Las Historias de usuario son tarjetas de papel en las que el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Las Historias de usuario son dinámicas, ya que durante el desarrollo de la aplicación pueden cambiar constantemente el valor de prioridad de cada historia; así como su fecha de entrega. Las estimaciones de esfuerzo asociado a la implementación de las historias las establecen los programadores al utilizar como medida el punto. Un punto equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos (LETELIER, 2012). A continuación se muestran las Historias de usuario definidas para esta investigación:

Tabla 2.1. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Cargar base de datos
Usuario: Especialista	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 0.8	**Iteración asignada:** 1
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Permite cargar la información sobre el proyecto, contenida en la base de datos.	
Observaciones: En caso de que se produzca algún error al cargar la base de datos, el sistema debe mostrar un mensaje explicando el problema.	

Tabla 2.2. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Obtener secuencias con el SSGS
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2.0	**Iteración asignada:** 1
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Ejecuta el SSGS para obtener un conjunto de secuencias válidas y genera un archivo con este resultado.	
Observaciones: En caso de que se produzca algún error, el sistema debe mostrar un mensaje explicando el problema.	

Tabla 2.3. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Cargar secuencias obtenidas con el SSGS
Usuario: Especialista	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo

Continúa en la próxima página

Tabla 2.3. Continuación de la página anterior

Puntos estimados: 0.8	**Iteración asignada:** 2
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Permite cargar las secuencias obtenidas con el SSGS, para ser utilizadas como población inicial del GA propuesto.	
Observaciones: En caso de que se produzca algún error al cargar las secuencias, el sistema debe mostrar un mensaje explicando el problema.	

Tabla 2.4. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Determinar secuencias con el GA
Usuario: Especialista	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3.0	**Iteración asignada:** 2
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se ejecuta el GA que determina las mejores secuencias de planificación.	
Observaciones: En caso de que se produzca algún error, el sistema debe mostrar un mensaje explicando el problema.	

Tabla 2.5. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Crear reporte
Usuario: Especialista	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 0.8	**Iteración asignada:** 2
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se genera un reporte en formato PDF, con la información referente a las mejores soluciones obtenidas con el GA.	
Observaciones: Si se genera el reporte más de una vez se reescribe el archivo creado con anterioridad.	

2.5.3. Fase de Planificación

En esta fase el cliente establece la prioridad de cada Historia de usuario, Ver Tabla 2.6. En correspondencia, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Además, se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

El equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las Historias de usuario que

fueron terminadas en la última iteración (LETELIER, 2012).

Tabla 2.6. Prioridad de las Historias de usuario

Historias de usuario	Prioridad
Cargar base de datos	Baja
Obtener secuencias con el SSGS	Alta
Cargar secuencias obtenidas con el SSGS	Baja
Determinar secuencias con el GA	Alta
Crear reporte	Media

Plan de iteraciones y estimación de esfuerzo

Al concluir la definición de las Historias de usuario es necesario estimar el esfuerzo que se dedicará a cada una y planificar un sistema de trabajo, ver Tabla 2.7. Para esta aplicación se definió un conjunto de 2 iteraciones, conformadas de la siguiente manera:

- Iteración 1: Se implementa las Historias de usuario 1 y 2. Estas garantizan la creación del SSGS para generar la población inicial del GA. Al concluir esta iteración se obtendrá una versión 1.0 de la aplicación.
- Iteración 2: Se implementa las Historias de usuario 3, 4 y 5, que tienen especial relevancia en el desarrollo del GA y en la presentación de los resultados obtenidos por este. Estos resultados son el propósito de la aplicación y al concluir se obtendrá la versión 1.1. Con esta versión se pueden probar todas las funcionalidades que están definidas para el algoritmo.

Tabla 2.7. Estimación de esfuerzo por historia de usuario

Iteración		Historias de usuario	Puntos estimados (semanas)
1	1	Cargar base de datos	0.8
	2	Obtener secuencias con el SSGS	2.0
2	3	Cargar secuencias obtenidas con el SSGS	0.8
	4	Determinar secuencias con el GA	3.0
	5	Crear reporte	0.8
Total			**7.4**

Plan de entrega

A partir del plan de iteraciones anterior se realiza el plan de entregas, que tiene como objetivo definir las entregas que se deben realizar y el orden de las mismas, ver Tabla 2.8.

Tabla 2.8. Plan de entrega

Historias de usuario	**Final 1ra Iteración (3ra semana de marzo)**	**Final 2da Iteración (4ta semana de abril)**
Cargar base de datos	v1.0	
Obtener secuencias con el SSGS		
Cargar secuencias obtenidas con el SSGS		v1.1
Determinar secuencias con el GA		
Crear reporte		

2.5.4. Fase de Diseño

Arquitectura

La arquitectura de software de una aplicación es la estructura que incluye los componentes del software, las propiedades visibles externas de estos componentes y las relaciones entre ellos (PRESSMAN, 2005). La arquitectura tiene relevancia dentro del proceso de ingeniería de software, ya que permite medir el impacto del diseño de la aplicación y determina el éxito final de esta.

Para el desarrollo de la aplicación se escogió el estilo arquitectónico N-capas. Este estilo se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver (MOQUILLAZA HENRÍQUEZ; VEGA HUERTA y GUERRA GRADOS, 2014). Se utiliza este estilo para lograr una alta cohesión, un aislamiento de la parte lógica de la aplicación donde está programado el GA y de esta forma se logra que sea reutilizable el algoritmo. Se dividió en tres capas la aplicación y quedó conformada de la siguiente manera:

Capa de presentación: Es la capa que permite la comunicación con el usuario. Esta capa solo se relaciona con la capa de negocio.

Capa de negocio: Es donde reside la lógica del programa, en este caso el algoritmo diseñado. Esta capa se comunica con las capas de presentación y de datos.

Capa de datos: Se encarga de recuperar los datos que genera la capa de negocio y exportar un PDF con los mejores resultados obtenidos.

Patrones de Asignación de Responsabilidades y Pandilla de los cuatro

Un sistema orientado a objetos se compone de clases que se envían mensajes unas a otras para llevar a cabo operaciones. El diseño y la asignación de responsabilidades consecuente de la interacción entre estas presentan gran variación. Así, las decisiones poco acertadas de diseño dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender (*Patrones del Gang of Four* s.f.; VISCONTI y ASTUDILLO, s.f.).

Una de las buenas prácticas de diseño de clases es seguir patrones establecidos con anterioridad como es el caso de los Patrones de Asignación de Responsabilidades (GRASP, por sus siglas en inglés) y Pandilla de

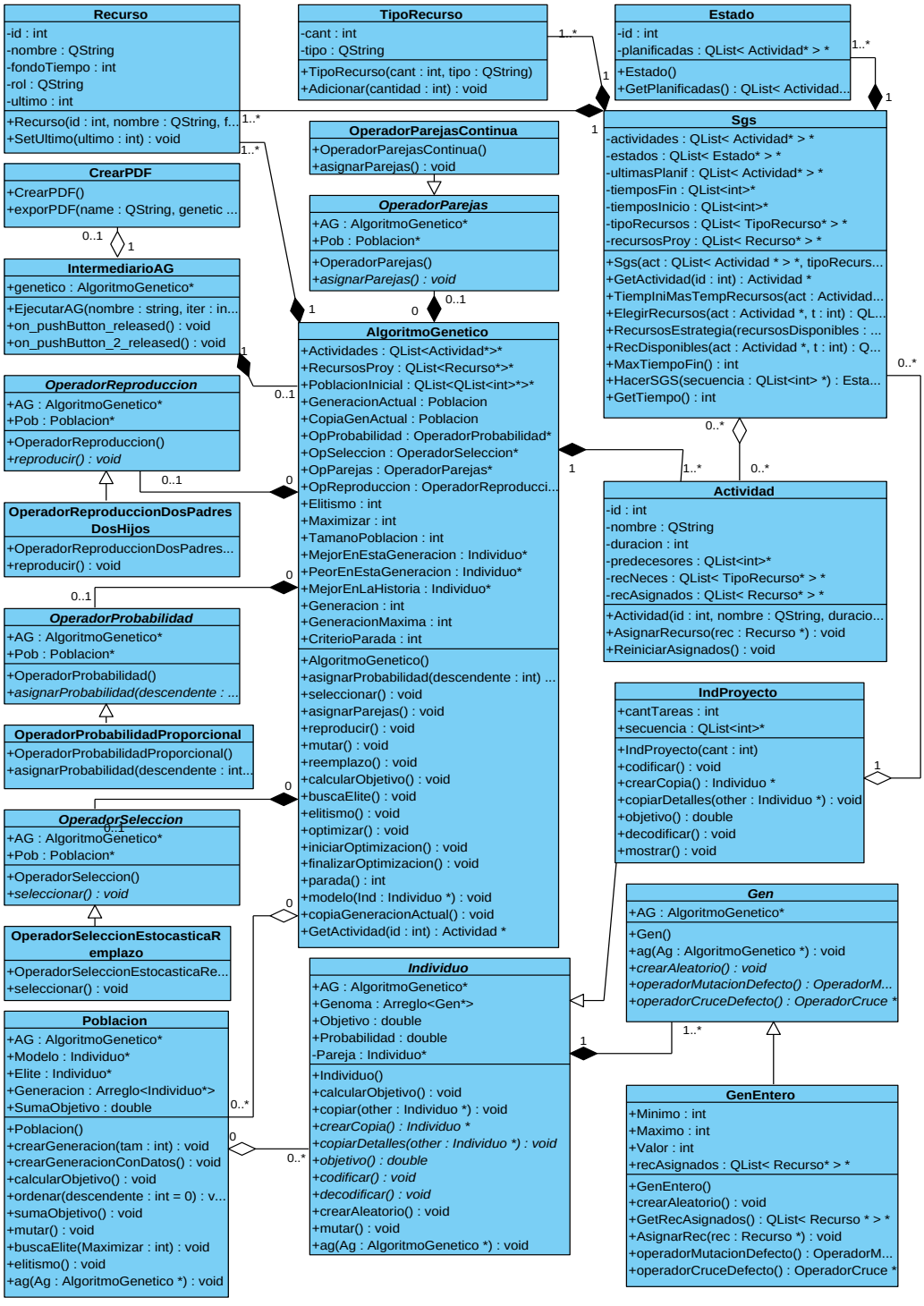

Figura 2.2. Diagrama de clases del GA

los cuatro (GoF, por sus siglas en inglés). Estos tipifican buenos principios de programación, cuya eficacia ha sido demostrada anteriormente en escenarios similares. A continuación se exponen los patrones utilizados en la aplicación desarrollada, estos pueden ser comprobados en el diagrama 2.2:

Creador: La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Una de sus principales ventajas es que permite lograr un bajo acoplamiento en el diseño de clases. Un ejemplo de su utilización es la clase AlgoritmoGenetico que tiene la responsabilidad de crear los operadores necesarios para ejecutar el GA diseñado.

Experto: Este patrón se utiliza con el objetivo de asignar una responsabilidad a la clase que posee toda la información necesaria para realizarla. Este patrón es el más utilizado debido a su esencia.

Controlador: Un controlador es un objeto de interfaz no destinado al usuario que se encarga de manejar un evento del sistema. En la aplicación hace función de intermediario entre el algoritmo diseñado y la interfaz de usuario desarrollada.

Alta cohesión: La cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Por tanto, una alta cohesión promueve la creación de clases con responsabilidades bien definidas y que estén estrechamente relacionadas. Este patrón se evidencia en las relaciones que poseen las clases implementadas para la realización del algoritmo.

Mediador: Es un patrón estructural que se encarga de establecer las comunicaciones entre un conjunto de objetos. Para este algoritmo se utilizó con el objetivo de establecer la relación entre las capas de la aplicación.

Tarjetas Clase-Responsabilidad-Colaboración

Las tarjetas Clase - Responsabilidad - Colaboración (CRC, por sus siglas en inglés) son fichas, una por cada clase, en las que se escriben brevemente las responsabilidades de la clase y una lista de los objetos con los que colabora para llevar a cabo esas responsabilidades. Permiten desprenderse del método basado en procedimientos y trabajar con una metodología basada en objetos, así el programador se centra y comienza a apreciar el desarrollo orientado a objetos (PUJOL MÉNDEZ, 2014). La información recopilada se puede enriquecer con el uso de diagramas de clases y de interacción.

Tabla 2.9. Tarjeta CRC # 1

Tarjeta CRC	
Clase: AlgoritmoGenetico	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.9. Continuación de la página anterior

Es la clase responsable de ejecutar la lógica del GA.	Población OperadorProbabilidad OperadorParejas OperadorReproducion OperadorSeleccion

Tabla 2.10. Tarjeta CRC # 2

Tarjeta CRC	
Clase: Poblacion	
Responsabilidad	**Colaboración**
Es la clase responsable de crear una población de individuos y ejecutar varias operaciones sobre ellos.	Individuo

Tabla 2.11. Tarjeta CRC # 3

Tarjeta CRC	
Clase: Individuo	
Responsabilidad	**Colaboración**
Es la clase responsable de crear el conjunto de genes necesarios para la creación de un individuo. Además de realizar el proceso de mutación y recombinación.	Gen

Tabla 2.12. Tarjeta CRC # 4

Tarjeta CRC	
Clase: Gen	
Responsabilidad	**Colaboración**
Es la clase responsable de almacenar la información genética.	Individuo

Tabla 2.13. Tarjeta CRC # 5

Tarjeta CRC	
Clase: OperadorParejas	
Responsabilidad	**Colaboración**
Es la clase encargada de crear las parejas de individuos que se reproducirán.	AlgoritmoGenetico

Tabla 2.14. Tarjeta CRC # 6

Tarjeta CRC	
Clase: OperadorReproduccion	
Responsabilidad	**Colaboración**
Es la clase encargada de ejecutar el operador para la reproducción entre individuos.	AlgoritmoGenetico

Tabla 2.15. Tarjeta CRC # 7

Tarjeta CRC	
Clase: OperadorProbabilidad	
Responsabilidad	**Colaboración**
Es la clase encargada de asignar una probabilidad de superviviencia a cada individuo.	AlgoritmoGenetico

Tabla 2.16. Tarjeta CRC # 8

Tarjeta CRC	
Clase: OperadorSeleccion	
Responsabilidad	**Colaboración**
Es la clase encargada de seleccionar los mejores individuos de la población.	AlgoritmoGenetico

Tabla 2.17. Tarjeta CRC # 9

Tarjeta CRC	
Clase: CrearPDF	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.17. Continuación de la página anterior

Es la clase encargada de generar un reporte con los resultados obtenidos.	IntermediarioGA

Tabla 2.18. Tarjeta CRC # 10

Tarjeta CRC	
Clase: IntermediarioGA	
Responsabilidad	**Colaboración**
Es la clase encargada de controlar las funcionalidades del algoritmo diseñado y actuar como intermediario entre la lógica del sistema y la interfaz.	AlgoritmoGenetico

Tabla 2.19. Tarjeta CRC # 11

Tarjeta CRC	
Clase: Recurso	
Responsabilidad	**Colaboración**
Es la clase que modela toda la información correspondiente a un recurso humano, así como las funcionalidades necesarias para interactuar con ella.	AlgoritmoGenetico Actividad SGS

Tabla 2.20. Tarjeta CRC # 12

Tarjeta CRC	
Clase: TipoRecurso	
Responsabilidad	**Colaboración**
Es la clase que modela una estructura de datos que contiene la cantidad de recursos por cada tipo, con los que cuenta el proyecto.	AlgoritmoGenetico Actividad SGS

Tabla 2.21. Tarjeta CRC # 13

Tarjeta CRC
Clase: Actividad

Continúa en la próxima página

Tabla 2.21. Continuación de la página anterior

Responsabilidad	**Colaboración**
Es la clase que modela la información relativa a una tarea de proyecto, además contiene los métodos necesarios para interactuar con ella.	AlgoritmoGenetico Recurso TipoRecurso SGS

Tabla 2.22. Tarjeta CRC # 14

Tarjeta CRC	
Clase: Estado	
Responsabilidad	**Colaboración**
Es la clase que modela un estado propuesto por el SGS. Contiene una lista de las actividades que han sido planificadas hasta ese intante.	SGS

Tabla 2.23. Tarjeta CRC # 15

Tarjeta CRC	
Clase: SGS	
Responsabilidad	**Colaboración**
Esta clase se encarga de calcular el tiempo de finalización de un proyecto, a partir de una secuencia predefinida de actividades. Además, asigna los recursos necesarios a cada actividad.	AlgoritmoGenetico Recurso TipoRecurso Actividad Estado

2.6. Reportes

El GA que se desarrolla tiene que generar, como resultado fundamental, secuencias de planificación optimizadas. Para cumplir con ello se diseña un reporte con formato PDF. Este reporte recoge las cinco mejores soluciones que obtenga el GA diseñado. El reporte está conformado por:

- En la cabecera se coloca el nombre del GA desarrollado.
- Se hace alusión al proyecto que da origen al reporte.
- Se muestran las mejores soluciones obtenidas. El formato de una solución está compuesto por una secuencia de tareas, los recursos asignados a cada tarea y el tiempo de finalización total del proyecto.

2.7. Conclusiones del capítulo

En este capítulo se definió una codificación apropiada para representar el problema. Se diseñó el GA propuesto, el cual utiliza como población inicial las secuencias de planificación obtenidas con una variante del SSGS. Esto permite concentrar la búsqueda en un área de posibles soluciones prometedoras. Se formalizó una función objetivo que permite evaluar la calidad del individuo a partir de la secuencia de planificación representada en su codificación. Se aplicaron operadores probabilísticos de cruzamiento y mutación que garantizan explorar el espacio de soluciones válidas. Además, se diseñó un reporte con formato PDF para mostrar los mejores resultados del algoritmo implementado.

CAPÍTULO 3

Resultados y Discusión

3.1. Introducción

En este capítulo se exponen las Tareas de ingeniería a partir de las Historias de usuario definidas. Se analizan los resultados obtenidos con la implementación del módulo propuesto. Para ello se emplean instancias de la biblioteca PSPLib. Además, se muestran las pruebas de aceptación realizadas para comprobar el cumplimiento de los requisitos establecidos por el cliente.

3.2. Fase de codificación

La fase de codificación es un proceso que se realiza en forma paralela al diseño del módulo (ECHEVERRY TOBÓN y DELGADO CARMONA, 2007; PRESSMAN, 2005). Las funcionalidades desarrolladas en esta etapa generan un entregable funcional que implementa las Historias de usuario asignadas a la iteración. Sin embargo, estas historias no contienen suficientes detalles para permitir su análisis y desarrollo. En consecuencia, se realizan las tareas necesarias para su análisis al principio de cada iteración. En la Tabla 3.1 se definen las Tareas de ingeniería para el desarrollo del algoritmo. Por otro lado, se define un estándar de código utilizado para las clases implementadas que no pertenecen a la biblioteca UNGenético.

Tabla 3.1. Tareas de ingeniería

Hist.	Tareas de ingeniería (número y nombre)		Pto. Est	Pto. Real
1	1	Diseño e implementación de la ventana principal	0.1	0.1
	2	Definición e implementación de la base de datos a cargar	0.3	0.3
	3	Validación de la base de datos cargada	0.2	0.2
	4	Desarrollo de las pruebas de aceptación	0.2	0.2
2	5	Diseño e implementación del SSGS	1.6	1.6
	6	Validación de las secuencias de planificación aleatorias	0.2	0.2
	7	Desarrollo de las pruebas de aceptación	0.2	0.2

Continúa en la próxima página

Tabla 3.1. Continuación de la página anterior

3	8	Implementación de la funcionalidad cargar secuencias generadas con el SSGS	0.5	0.5
	9	Validación de la funcionalidad cargar secuencias de planificación generadas	0.2	0.2
	10	Desarrollo de las pruebas de aceptación	0.1	0.1
4	11	Diseño e implementación del GA	2.6	2.6
	12	Validación de las secuencias de planificación obtenidas	0.2	0.2
	13	Desarrollo de las pruebas de aceptación	0.2	0.2
5	14	Implementación del reporte	0.4	0.4
	15	Validación de crear reporte	0.2	0.2
	16	Desarrollo de las pruebas de aceptación	0.2	0.2

Las Tareas de ingeniería se utilizan para describir las actividades que se realizan sobre el proyecto, que pueden ser de desarrollo, corrección o mejora y se definen para dar cumplimiento a las Historias de usuario. La Tablas 3.2 y 3.3 son ejemplos de las Tareas de ingeniería definidas para la realización del algoritmo. El resto se pueden observar en el Anexo A.1.

Tabla 3.2. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Diseño e implementación de la ventana principal	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 1 de marzo de 2015	**Fecha de fin:** 1 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se realiza el diseño e implementación de la ventana principal.	

Tabla 3.3. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 1
Nombre de la tarea: Definición e implementación de la base de datos a cargar	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 1 de marzo de 2015	**Fecha de fin:** 5 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se establecen como tablas de la base de datos las siguientes: **Actividad:** Contiene los datos correspondientes a una tarea de proyecto. **Recurso:** Almacena las características de los recursos humanos del proyecto. **Tipo Recurso:** Contiene el número de recursos por tipo, con los que cuenta el proyecto.	

3.3. Conjuntos de Problemas de PSPLib

La biblioteca PSPLib (http://www.om-db.wi.tum.de/psplib) contiene diferentes conjuntos de problemas para los distintos PSP, así como soluciones óptimas y heurísticas. Los conjuntos de datos pueden ser utilizados para la evaluación de los procedimientos de solución de los RCPSP. Los investigadores pueden descargar la referencia fija para evaluar sus algoritmos. Además, pueden enviar sus resultados para ser añadidos a la biblioteca (KOLISCH y SPRECHER, 1997). A continuación se describe el formato general de los archivos que propone PSPLib como instancias de problemas:

- Número de actividades (jobs).
- Número de modos en que cada actividad puede ejecutarse (#modes).
- Número de tipos de recursos renovables existentes en el problema (renewable).
- Disponibilidad máxima de cada tipo de recurso renovable (RESOURCEAVAILABILITIES).
- Número de tipos de recursos no renovables existentes en el problema (nonrenewable).
- Número de sucesores de cada actividad (#successors).
- Conjunto de actividades sucesoras de cada actividad (successors).
- Duración de cada actividad (duration).
- Número de recursos necesarios para cada actividad (R1, R2, R3, R4)

3.3.1. Instancia de 30 actividades

La instancia de prueba *j301-1* obtenida de PSPLib, contiene 30 tareas y 41 recursos, estos últimos están clasificados en cuatro tipos. En la Figura 3.1 se observa el formato del fichero *j301-1.sm*.

Tabla 3.4. Resultados para la instancia de prueba *j301-1*

n=30		**Tamaño de la Población**				
	Sol	**20**	**40**	**60**	**80**	**100**
Generaciones	**25**	55	50	53	52	46
	50	55	47	51	49	45
	75	54	50	47	53	47
	100	53	53	54	49	45

En la Tabla 3.4 y la Figura B.1 se observa el mejor resultado obtenido por el GA implementado, atendiendo al número de individuos de la población y la cantidad de generaciones.

```
jobs (incl. supersource/sink ):  32
RESOURCES
  - renewable                 :  4   R
  - nonrenewable              :  0   N
******************************************
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          3           2  3  4
   2        1          3           6  11 15
   3        1          3           7  8  13
   4        1          3           5  9  10
   5        1          1           20
   6        1          1           30
   7        1          1           27
   8        1          3           12 19 27
   9        1          1           14
  10        1          2           16 25
  11        1          2           20 26
  12        1          1           14
  13        1          2           17 18
  14        1          1           17
  15        1          1           25
  16        1          2           21 22
  17        1          1           22
  18        1          2           20 22
  19        1          2           24 29
  20        1          2           23  25
  21        1          1           28
  22        1          1           23
  23        1          1           24
  24        1          1           30
  25        1          1           30
  26        1          1           31
  27        1          1           28
  28        1          1           31
  29        1          1           32
  30        1          1           32
  31        1          1           32
  32        1          0
```

```
******************************************
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2  R 3  R 4
------------------------------------------
  1      1     0       0    0    0    0
  2      1     8       4    0    0    0
  3      1     4      10    0    0    0
  4      1     6       0    0    0    3
  5      1     3       3    0    0    0
  6      1     8       0    0    0    8
  7      1     5       4    0    0    0
  8      1     9       0    1    0    0
  9      1     2       6    0    0    0
 10      1     7       0    0    0    1
 11      1     9       0    5    0    0
 12      1     2       0    7    0    0
 13      1     6       4    0    0    0
 14      1     3       0    8    0    0
 15      1     9       3    0    0    0
 16      1    10       0    0    0    5
 17      1     6       0    0    0    8
 18      1     5       0    0    0    7
 19      1     3       0    1    0    0
 20      1     7       0   10    0    0
 21      1     2       0    0    0    6
 22      1     7       2    0    0    0
 23      1     2       3    0    0    0
 24      1     3       0    9    0    0
 25      1     3       4    0    0    0
 26      1     7       0    0    4    0
 27      1     8       0    0    0    7
 28      1     3       0    8    0    0
 29      1     7       0    7    0    0
 30      1     2       0    7    0    0
 31      1     2       0    0    2    0
 32      1     0       0    0    0    0
******************************************
RESOURCEAVAILABILITIES:
  R 1  R 2  R 3  R 4
   12   13    4   12
******************************************
```

Figura 3.1. Fichero j301-1.sm obtenido de PSPLib

Se observa que para esta instancia de 30 tareas el algoritmo converge a 45 unidades de tiempo. Un valor muy cercano al 43 propuesto en PSPLib como mejor solución para este caso. Por otra parte, la información anterior indica que los valores más cercanos a la solución óptima se logran con poblaciones de 40 o más individuos, después de las 25 generaciones.

Otras pruebas realizadas con esta instancia arrojaron como resultado que para una misma población inicial de n individuos, si se asigna un número de generaciones entre 25 y 100, el GA obtiene valores iguales o muy similares.

3.3.2. Instancia de 60 actividades

La instancia de prueba *j601-1* (Ver Figura 3.4), obtenida de PSPLib, cuenta con 60 tareas y contiene 49 recursos distribuidos en cuatro tipos. En la Tabla 3.5 y la Figura B.2 se observa el mejor resultado obtenido por el GA implementado, atendiendo al número de individuos de la población y la cantidad de generaciones.

Tabla 3.5. Resultados para la instancia de prueba *j601-1*

n=60		Tamaño de la Población				
	Sol	**20**	**40**	**60**	**80**	**100**
Generaciones	**25**	83	83	94	82	78
	50	90	83	82	83	77
	75	89	86	83	80	78
	100	89	87	90	78	77

Para esta instancia de 60 tareas, el algoritmo obtiene como mejor valor 77 unidades de tiempo, que coincide con la mejor solución propuesta por PSPLib. Las mejores soluciones se obtienen con poblaciones iniciales de 80 o más individuos, a partir de las 25 generaciones.

Al ejecutar el GA para esta instancia, con una misma población inicial de n individuos, si se varía el número de generaciones en el rango de 25 a 100, se obtiene el mismo valor o valores muy similares. Esto no se puede afirmar en caso de cambiar la población inicial para una misma cantidad de individuos y un mismo número de generaciones.

Resultados de la comparación entre casos de prueba

Luego de analizar por separado ambos casos, se evidencia que para una misma población inicial, independientemente del número de generaciones entre 25 y 100, el GA implementado obtiene soluciones iguales o muy similares.

Por otra parte, si se varía la población inicial se observa que para la instancia de 30 tareas la convergencia se obtiene con poblaciones de 40 o más individuos. Mientras que para la instancia de 60 tareas las mejores soluciones se alcanzan con mayor regularidad a partir de poblaciones de 80 individuos o más. En ambos casos se consideran estos resultados con un número de generaciones igual o superior a 25.

Lo anterior evidencia que, a medida que aumenta el número de tareas de un proyecto, el GA precisa de un número mayor de individuos en la población para obtener las mejores soluciones.

3.4. Convergencia del Algoritmo Genético

En la Figura 3.2 se realiza el análisis de los resultados obtenidos con la ejecución del GA para la instancia *j301-1*. En este se considera el valor de la función objetivo del mejor individuo de la población en el intervalo de generaciones de 0 a 100, con paso 5. Se representan los resultados de cinco corridas del algoritmo, cada una de ellas varía el tamaño de la población: 20, 40, 60, 80 y 100 individuos respectivamente.

En todos los casos se constata que el ritmo de convergencia del algoritmo es mayor en las primeras 40 generaciones que en las restantes. En estas últimas se mejora la solución únicamente en los casos de 60 y 100 individuos. En los cinco casos el GA implementado converge al valor óptimo o uno bastante cercano al óptimo, en un número pequeño de generaciones.

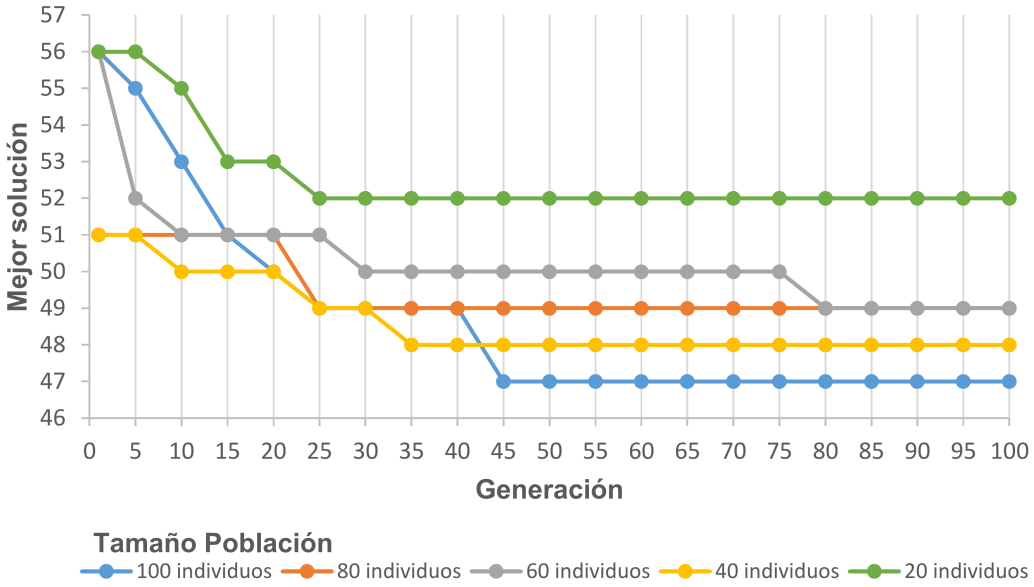

Figura 3.2. Convergencia del GA

3.5. Balance de carga entre recursos

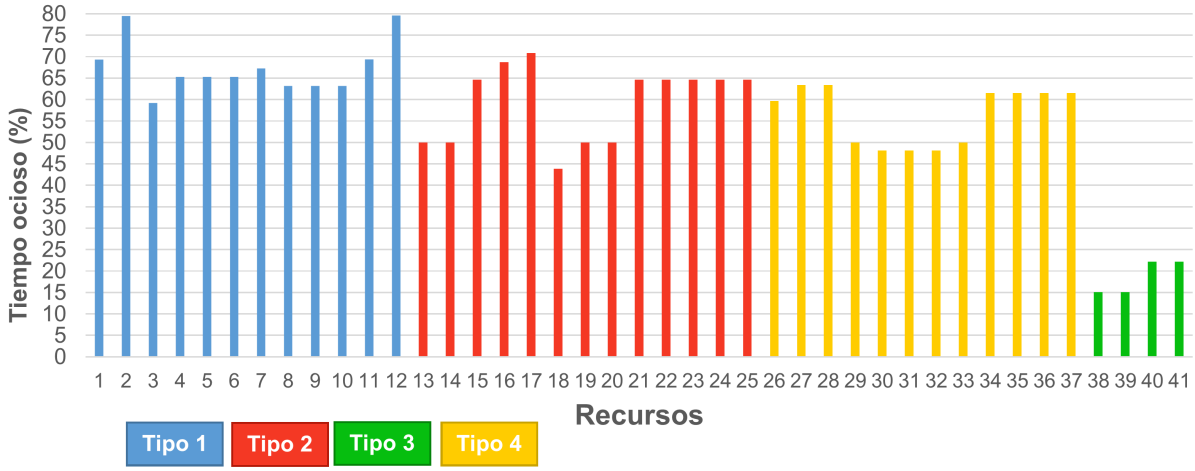

Figura 3.3. Balance de carga entre recursos del proyecto

La información representada en la Figura 3.3 corresponde a la instancia *j301-1* y se obtiene a partir de la ejecución del GA implementado. Se muestra el por ciento que representa el tiempo ocioso de cada recurso respecto al total de los tiempos de ejecución de cada una de las actividades que solicitan el empleo de recursos de su mismo tipo. En esta instancia de 30 tareas se cuenta con 4 tipos de recursos, en la gráfica se observa un favorable balance de carga entre los recursos de un mismo tipo.

```
jobs (incl. supersource/sink ):  62
RESOURCES
  - renewable                 :  4   R
  - nonrenewable              :  0   N
******************************************
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          3           2   3   4
   2        1          3           5  10  15
   3        1          3           7  14  29
   4        1          3           8  12  16
   5        1          3           6  22  24
   6        1          2          17  38
   7        1          1          23
   8        1          3           9  20  40
   9        1          2          13  35
  10        1          2          11  45
  11        1          3          26  37  44
  12        1          2          21  27
  13        1          1          18
  14        1          3          18  19  34
  15        1          2          25  59
  16        1          2          55  58
  17        1          2          32  43
  18        1          2          28  33
  19        1          1          28
  20        1          2          27  31
  21        1          1          39
  22        1          1          31
  23        1          1          51
  24        1          1          48
  25        1          1          40
  26        1          2          49  54
  27        1          2          30  51
  28        1          2          47  61
  29        1          2          41  57
  30        1          1          56
  31        1          1          43
  32        1          1          57
  33        1          1          39
  34        1          1          44
  35        1          2          36  39
  36        1          1          42
  37        1          1          58
  38        1          2          50  60
  39        1          1          53
  40        1          1          53
  41        1          1          46
  42        1          1          48
  43        1          2          51  58
  44        1          1          50
  45        1          1          53
  46        1          1          56
  47        1          1          52
  48        1          1          55
  49        1          1          60
  50        1          1          61
  51        1          1          60
  52        1          1          54
  53        1          1          56
  54        1          1          55
  55        1          1          57
  56        1          1          61
  57        1          1          59
  58        1          1          59
  59        1          1          62
  60        1          1          62
  61        1          1          62
  62        1          0
```

```
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2  R 3  R 4
------------------------------------------
  1      1     0       0    0    0    0
  2      1     8      10    0    0    0
  3      1     1       0    1    0    0
  4      1    10       0    9    0    0
  5      1     6       0    4    0    0
  6      1     5       0    0    0    1
  7      1     8      10    0    0    0
  8      1     9       0    0    6    0
  9      1     1       0    0    0    8
 10      1     9       0    6    0    0
 11      1     8       0    0    0    3
 12      1     3       0    7    0    0
 13      1     6       8    0    0    0
 14      1     2       0    0    0    1
 15      1     5       0    0    0    9
 16      1     1       6    0    0    0
 17      1     3       2    0    0    0
 18      1    10       0    0    2    0
 19      1     9       7    0    0    0
 20      1     1       5    0    0    0
 21      1     3       0    0    8    0
 22      1     6       0    4    0    0
 23      1     3       0    0    4    0
 24      1     3       0    5    0    0
 25      1     7       0    0    1    0
 26      1     6       9    0    0    0
 27      1    10       0    7    0    0
 28      1     9       3    0    0    0
 29      1     8       0    0    3    0
 30      1     4       0    0    7    0
 31      1     3       6    0    0    0
 32      1     3       0    0    0    4
 33      1     6       0    7    0    0
 34      1     1       0    0    0    4
 35      1     9       0    0    1    0
 36      1     9       9    0    0    0
 37      1     1       0    7    0    0
 38      1     2       5    0    0    0
 39      1     4       0    0    1    0
 40      1     9       0    0    0    5
 41      1    10       0    0    0    1
 42      1     8       0    0    0    9
 43      1     4       0    0    6    0
 44      1     3       0    0    0    1
 45      1     6       0    0    0    9
 46      1     6       0    0    0    7
 47      1     7       4    0    0    0
 48      1     3       0    8    0    0
 49      1     2       0    2    0    0
 50      1    10       0    0    7    0
 51      1     4       0    5    0    0
 52      1     2       2    0    0    0
 53      1     1       0    1    0    0
 54      1     4       0    0    6    0
 55      1    10       0    0    0    7
 56      1     8       0    0    3    0
 57      1     6       0    4    0    0
 58      1    10       0    0    9    0
 59      1     3       0    0    0    7
 60      1    10       0    3    0    0
 61      1     1       0    0    0    1
 62      1     0       0    0    0    0
******************************************
RESOURCEAVAILABILITIES:
  R 1  R 2  R 3  R 4
   13   11   12   13
```

Figura 3.4. Fichero j601-1.sm obtenido de PSPLib

3.6. Pruebas

Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos que se implementan. La calidad de los sistemas aumenta al disminuir la cantidad de errores de la aplicación. Las pruebas deben tener un conjunto de características para que se obtengan buenos resultados (LETELIER, 2012):

- Crear la prueba abstrayéndose del futuro código, de esta forma, se asegura la independencia de esta respecto al código que se evalúa.
- Observar la refactorización. La prueba permite verificar que un cambio en la estructura de un código no implica un cambio en su funcionamiento.
- Realizar pruebas a las funcionalidades generales que debe cumplir el programa especificado en las Historias de usuario.

Existen dos tipos de pruebas ampliamente utilizadas en la metodología XP, las unitarias y las de aceptación. Las pruebas unitarias se desarrollan por los programadores y verifican el código automáticamente. Las pruebas de aceptación verifican que el final de cada iteración de las Historias de usuario cumplen con la funcionalidad asignada y satisfacen las necesidades del cliente.

Se realizaron 5 pruebas de aceptación del cliente para validar la calidad de las funcionalidades implementadas. Además, durante el proceso de desarrollo de software y de refactorización de código, frecuentemente se realizaron pruebas a la aplicación por parte del desarrollador.

3.6.1. Pruebas de aceptación

Seguidamente se describen las pruebas de aceptación diseñadas a partir de las Historias de usuario:

Tabla 3.6. Caso de prueba de aceptación # 1

Caso de prueba de aceptación	
Código: $HU1_P1$	Historia de usuario: 1
Nombre: Cargar datos de proyecto	
Descripción: Prueba la funcionalidad cargar base de datos.	
Condición de ejecución: • El usuario debe tener la base de datos hospedada en el gestor para ser cargada por la aplicación.	
Paso de ejecución: • El usuario selecciona la opción **Cargar base de datos** que se encuentra en la barra de menú.	
Resultados esperados: • Si se establece la conexión con la base de datos se habilita el botón **Ejecutar SGS**. • Si no se establece la conexión con la base de datos el sistema lanzará la alerta *"Ha ocurrido un error mientras se cargaba la base de datos."*	
Evaluación de la prueba: Satisfactorio	

Tabla 3.7. Caso de prueba de aceptación # 2

Caso de prueba de aceptación	
Código: $HU2_P1$	Historia de usuario: 2
Nombre: Generar secuencias con el SSGS	
Descripción: Prueba la funcionalidad obtener secuencias con el SSGS.	
Condición de ejecución: • El usuario debe haber cargado la base de datos hospedada en el gestor.	
Pasos de ejecución: • El usuario selecciona la cantidad de secuencias que desea generar. • El usuario obtiene el fichero con las secuencias luego de hacer clic en la opción **Ejecutar SGS**.	
Resultados esperados: • Se crea el fichero con el número de secuencias solicitadas por el usuario. • Se lanza un mensaje indicando que el proceso se ha desarrollado correctamente.	
Evaluación de la prueba: Satisfactorio	

Tabla 3.8. Caso de prueba de aceptación # 3

Caso de prueba de aceptación	
Código: $HU3_P1$	Historia de usuario: 3
Nombre: Cargar secuencias para la población inicial del GA	
Descripción: Prueba la funcionalidad cargar secuencias obtenidas con el SSGS.	
Condición de ejecución: • El usuario debe poseer el archivo con el formato definido para ser cargado por la aplicación.	
Pasos de ejecución: • El usuario selecciona la opción **Cargar Secuencias**. • Luego selecciona el fichero con las secuencias.	
Resultados esperados: • Si el usuario selecciona un fichero con el formato correcto se habilita el botón **Ejecutar AG**.	
Evaluación de la prueba: Satisfactorio	

Tabla 3.9. Caso de prueba de aceptación # 4

Caso de prueba de aceptación	
Código: $HU4_P1$	Historia de usuario: 4
Nombre: Determinar secuencias con el GA	
Descripción: Prueba el GA implementado.	
Condición de ejecución: • El usuario debe haber cargado un fichero válido con anterioridad. • El usuario debe tener la base de datos hospedada en el gestor para ser cargada por la aplicación.	
Pasos de ejecución: • El usuario introduce un nombre para el caso a ejecutar.	

Continúa en la próxima página

Tabla 3.9. Continuación de la página anterior

• El usuario define el número máximo de generaciones para el GA. • Luego selecciona la opción **Ejecutar GA** para obtener las soluciones.
Resultados esperados: • Si el usuario ha introducido un nombre y un número de iteraciones para el caso a ejecutar, la aplicación mostrará los resultados fundamentales del GA. • Si el usuario no introduce un nombre para el caso a ejecutar la aplicación lanzará la alerta *"Debe introducir un nombre para el reporte."*
Evaluación de la prueba: Satisfactorio

Tabla 3.10. Caso de prueba de aceptación # 5

Caso de prueba de aceptación	
Código: $HU5_P1$	Historia de usuario: 5
Nombre: Generar reporte	
Descripción: Prueba la funcionalidad crear reporte.	
Condición de ejecución: El GA debe haber encontrado secuencias de planificación.	
Pasos de ejecución: • El usuario selecciona la opción **Crear Reporte**.	
Resultados esperados: • Se crea un archivo con formato PDF, este documento contiene las 5 mejores soluciones obtenidas por el GA y datos generales del proyecto	
Evaluación de la prueba: Satisfactorio	

3.7. Conclusiones del capítulo

Las instancias de prueba de PSPLib que se analizan en este capítulo demuestran que con la estrategia implementada se obtienen secuencias óptimas o cercanas a las óptimas para la solución del RCPSP.

Conclusiones

Al arribar a este punto se obtienen las siguientes conclusiones:

- Con el GA diseñado se obtienen soluciones óptimas o cercanas a las óptimas para el RCPSP.
- La creación de una población inicial con el SSGS permite la convergencia del GA en un número pequeño de generaciones.
- Con el GA implementado se logra un adecuado balance de carga entre los recursos de un mismo tipo.

Recomendaciones

Con interés de buscar mayor eficiencia en el proceso de planificación de tareas de un proyecto de software se proponen las siguientes recomendaciones:

- Analizar el uso de otros operadores que pudieran ser implementados en el GA para comprobar si mejoran las soluciones obtenidas.
- Desarrollar un mecanismo de asignación de recursos que, además de tener en cuenta el tiempo ocioso, utilice otros indicadores. Ejemplo: desempeño del rol.
- Introducir el concepto de múltiples modos de procesamiento en el algoritmo desarrollado.

Acrónimos

GA Algoritmos Genéticos. 2–4, 8, 11, 12, 14–19, 21, 23, 24, 26–31, 34, 35, 37–41, 44–47, 56, 57, 59

IDE Entorno de Desarrollo Integrado. 15

PSGS Esquema Generador de Soluciones en Paralelo. 14

PSP Problema de Planificación de Proyectos. 5, 6, 38

RCPSP Problema de Planificación de Proyectos con Recursos Limitados. 2–4, 6–11, 13–15, 17–19, 22, 38, 45, 46

SGA Algoritmo Genético Simple. 17

SGS Esquema Generador de Soluciones. 13, 15, 34

SSGS Esquema Generador de Soluciones en Serie. 13, 14, 17, 19, 22, 24–28, 35–37, 44, 46, 55, 56

UCI Universidad de las Ciencias Informáticas. 1, 16

XP Programación Extrema. 43

Referencias bibliográficas

ARRANZ DE LA PEÑA, Jorge y PARRA TRUYOL, Antonio. 2014. Algoritmos Genéticos. *Universidad Carlos III*. 2014.

ARTIGUES, Christian; MICHELON, Philippe y REUSSER, Stéphane. 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*. 2003, vol. 149, n.º 2, págs. 249-267.

BAAR, Tonius; BRUCKER, Peter y KNUST, Sigrid. 1999. *Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem*. 1999.

BALLESTÍN GONZÁLEZ, Francisco. 2002. *Nuevos métodos de resolución del problema de secuenciación de proyectos con recursos limitados*. 2002.

BLUM, Christian y ROLI, Andrea. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*. 2003, vol. 35, n.º 3, págs. 268-308.

BROOKS, SP y MORGAN, BJT. 1995. Optimization using simulated annealing. *The Statistician*. 1995, págs. 241-257.

BRUCKER, Peter; DREXL, Andreas; MÖHRING, Rolf; NEUMANN, Klaus y PESCH, Erwin. 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*. 1999, vol. 112, n.º 1, págs. 3-41.

CAPETILLO CORBEA, Jarsey. 2012. *Implementación de un algoritmo híbrido basado en optimización por Enjambre de Partículas para el problema de planificación de proyecto con múltiples modos de procesamiento*. 2012.

CERVANTES POSADA, Mariamar. 2010. *Nuevos métodos metaheurísticos para la asignación eficiente, optimizada y robusta de recursos Limitados*. 2010.

DAZA, Julio Mario; MONTOYA, Jairo R y NARDUCCI, Francesco. 2009. Resolución del problema de enrutamiento de vehículos con limitaciones de capacidad utilizando un procedimiento metaheurístico de dos fases. *Revista EIA*. 2009, vol. 12, págs. 23-38.

DEBELS, Dieter; DE REYCK, Bert; LEUS, Roel y VANHOUCKE, Mario. 2006. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*. 2006, vol. 169, n.º 2, págs. 638-653.

DEMEULEMEESTER, Erik Leuven. 2002. *Project scheduling: a research handbook*. 2002.

DORIGO, Marco. 1992. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*. 1992.

ECHEVERRY TOBÓN, Luís Miguel y DELGADO CARMONA, Luz Elena. 2007. Caso práctico de la metodología ágil XP al desarrollo de software. 2007.

FEO, Thomas A y RESENDE, Mauricio GC. 1989. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*. 1989, vol. 8, n.º 2, págs. 67-71.

FRANCO, Gutiérrez. 2013. A Genetic Algorithm for the Resource Constrained Project Scheduling Problem (RCPSP). *School of Industrial Engineering, Universidad de La Sabana*. 2013, vol. 10, n.º 20.

GIL LONDOÑO, Natyhelem. 2006. Algoritmos Genéticos. *Medellin, Colombia*. 2006.

GLOVER, Fred. 1989. Tabu search. *ORSA Journal on computing*. 1989, vol. 1, n.º 3, págs. 190-206.

GLOVER, Fred. 1998. A template for scatter search and path relinking. En. *Artificial evolution*. 1998, págs. 1-51.

GLOVER, Fred y LAGUNA, Manuel. 1999. *Tabu search*. 1999.

HARTMANN. 1997. A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Institut fur Betriebswirtschaftslehre der Universitat Kiel, Germany*. 1997, vol. 10, n.º 20.

HARTMANN y BRISKORN, Dirk. 2008. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*. 2008, vol. 207, n.º 02/2008, págs. 1-14.

HOLLAND, J. 1975. Adaptation in Natural and Artificial Systems. *Unversity of Michigand*. 1975, vol. 1, n.º 3, págs. 208.

KELLEY JR, James E. 1961. Critical-path planning and scheduling: Mathematical basis. *Operations Research*. 1961, vol. 9, n.º 3, págs. 296-320.

KIRKPATRICK, Scott; GELATT, C Daniel; VECCHI, Mario P et al., 1983. Optimization by simmulated annealing. *science*. 1983, vol. 220, n.º 4598, págs. 671-680.

KLEIN, Robert. 2000. Project scheduling with time-varying resource constraints. *International Journal of Production Research*. 2000, vol. 38, n.º 16, págs. 3937-3952.

KOLISCH, Rainer. 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*. 1996, vol. 90, n.º 2, págs. 320-333.

KOLISCH, Rainer y HARTMANN, Sonke. 1999. *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis*. 1999.

KOLISCH, Rainer y PADMAN, Rema. 2001. An integrated survey of deterministic project scheduling. *Omega*. 2001, vol. 29, n.º 3, págs. 249-272.

KOLISCH, Rainer y SPRECHER, Arno. 1997. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European journal of operational research*. 1997, vol. 96, n.º 1, págs. 205-216.

LAGUNA, Manuel; MARTI, Rafael y MARTÍ, Rafael Cunquero. 2003. *Scatter search: methodology and implementations in C*. 2003.

LANCASTER, J. y OZBAYRAK, M. 2007. Evolutionary algorithms applied to project scheduling problems - a survey of the state-of-the-art. *International Journal of Production Research*. 2007, vol. 45, n.º 2, págs. 425-450.

LETELIER, Patricio. 2012. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 2012.

MALCOLM, Donald G; ROSEBOOM, John H; CLARK, Charles E y FAZAR, Willard. 1959. Application of a technique for research and development program evaluation. *Operations research*. 1959, vol. 7, n.º 5, págs. 646-669.

MEDRANO BROCHE, Bolivar E. 2012. *Modelo para la planificación de múltiples proyectos de desarrollo de software*. 2012.

MERKLE, Daniel; MIDDENDORF, Martin y SCHMECK, Hartmut. 2002. Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*. 2002, vol. 6, n.º 4, págs. 333-346.

MODER, Joseph John y PHILLIPS, Cecil R. 1964. *Project Management with CPM and PERT*. 1964.

MOQUILLAZA HENRÍQUEZ, Santiago Domingo; VEGA HUERTA, Hugo y GUERRA GRADOS, Luis Angel. 2014. Programación en N capas. *Revista de investigación de Sistemas e Informática*. 2014, vol. 7, n.º 2, págs. 57-67.

MORILLO, Daniel; MORENO, Luis y DÍAZ, Javier. 2014. Metodologías analíticas y heurísticas para la solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 2. *Ingeniería y Ciencia-ing. cienc.* 2014, vol. 10, n.º 20, págs. 203-227.

NARVÁEZ MOLINA Gabriela y Saltos Atiencia, Ramiro. 2010. *Implementación de un Algoritmo Genético para resolver el Problema de Programación de Proyectos con Recursos Limitados*. 2010.

PALOMINO, E. Reynaldo. 2012. *Módulo de técnicas de optimización bioinspiradas para la Biblioteca de Inteligencia Artificial del CEDIN*. 2012.

Patrones del Gang of Four.

PRESSMAN, S. Roger. 2005. *Ingeniería del software.* 6ta. 2005.

PRITSKER, A Alan B; WAITERS, Lawrence J y WOLFE, Philip M. 1969. Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*. 1969, vol. 16, n.º 1, págs. 93-108.

PUJOL MÉNDEZ Naivy y Echevarría Días, Jesús. 2014. *Sistema de Gestión para el Centro Internacional de Postgrado*. 2014.

SALVADOR, Mercado. 2001. Administración aplicada teoría y práctica. *Teoría y Práctica*. 2001.

SANTANA, J Brito; RODRÍGUEZ, C Campos; LÓPEZ, FC García et al., 2004. Metaheurísticas: Una revisión actualizada. *Universidad de La Laguna, España: Departamento de Estadística, Investigación Operativa y Computación*. 2004, vol. 263.

SANTIESTEBAN QUINTANA, Leyanis. 2011. *Metodología para la planificación y seguimiento de proyectos de desarrollo de sistemas informáticos de Planificación de Recursos Empresariales (ERP)*. 2011.

SOMERVILLE, Ian. 2005. *Ingeniería de software Séptima edición*. 2005.

VELASCO, Oscar Germán Duarte. 2002. *UNGenético: Una librería en C++ de Algoritmos Genéticos con Codifcación Híbrida*. 2002.

VISCONTI, Marcelo y ASTUDILLO, Hernán. *Fundamentos de Ingeniería de Software.* Universidad Técnica Federico Snata María.

ZOULFAGHARI, Hossein; NEMATIAN, Javad; MAHMOUDI, Nader y KHODABANDEH, Mehdi. 2013. A New Genetic Algorithm for the RCPSP in Large Scale. *International Journal of Applied Evolutionary Computation*. 2013, vol. 10, n.º 20.

Generado con LaTeX: 9 de junio de 2015: 11:03pm

Apéndices

APÉNDICE A

Artefactos de Ingeniería de Software

A.1. Tareas de ingeniería

Tabla A.1. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 1
Nombre de la tarea: Validación de la base de datos cargada	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 6 de marzo de 2015	**Fecha de fin:** 7 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se realizan los métodos necesarios para validar que el archivo cargado cumpla con el formato definido.	

Tabla A.2. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 1
Nombre de la tarea: Desarrollo de las pruebas de aceptación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 8 de marzo de 2015	**Fecha de fin:** 10 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se especifican los escenarios para probar que la Historia de usuario cargar base de datos ha sido implementada correctamente.	

Tabla A.3. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 2
Nombre de la tarea: Diseño e implementación del SSGS	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.6
Fecha de inicio: 10 de marzo de 2015	**Fecha de fin:** 19 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se define e implementan un SSGS a partir del pseudocódigo obtenido.	

Tabla A.4. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 2
Nombre de la tarea: Validación de las secuencias de planificación aleatorias	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 20 de marzo de 2015	**Fecha de fin:** 21 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se realizan los métodos necesarios para validar que las secuencias obtenidas tienen un formato correcto.	

Tabla A.5. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 2
Nombre de la tarea: Desarrollo de las pruebas de aceptación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 22 de marzo de 2015	**Fecha de fin:** 23 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se especifican los escenarios para probar que la Historia de usuario obtener secuencias con el SSGS ha sido implementada correctamente.	

Tabla A.6. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 3
Nombre de la tarea: Implementación de la funcionalidad cargar secuencias generadas con el SSGS	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 24 de marzo de 2015	**Fecha de fin:** 26 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	

Continúa en la próxima página

Tabla A.6. Continuación de la página anterior

Descripción: Se implementa la funcionalidad de cargar el archivo con las secuencias de planificación, generado por el SSGS.

Tabla A.7. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 3
Nombre de la tarea: Validación de la funcionalidad cargar secuencias de planificación generadas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 27 de marzo de 2015	**Fecha de fin:** 28 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se realizan los métodos necesarios para validar que la opción cargar secuencias obtenidas con el SSGS ha sido implementada correctamente.	

Tabla A.8. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 3
Nombre de la tarea: Desarrollo de las pruebas de aceptación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 28 de marzo de 2015	**Fecha de fin:** 28 de marzo de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se especifican los escenarios para probar que la Historia de usuario cargar secuencias obtenidas con el SSGS ha sido implementada correctamente.	

Tabla A.9. Tarea de ingeniería # 11

Tarea	
Número de tarea: 11	**Número de Historia de usuario:** 4
Nombre de la tarea: Diseño e implementación del GA	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.6
Fecha de inicio: 29 de marzo de 2015	**Fecha de fin:** 13 de abril de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se definen e implementan los operadores del GA.	

Tabla A.10. Tarea de ingeniería # 12

Tarea	
Número de tarea: 12	**Número de Historia de usuario:** 4
Nombre de la tarea: Validación de las secuencias de planificación obtenidas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 13 de abril de 2015	**Fecha de fin:** 15 de abril de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se realizan los métodos necesarios para validar que la opción determinar secuencias con el GA ha sido implementada correctamente.	

Tabla A.11. Tarea de ingeniería # 13

Tarea	
Número de tarea: 13	**Número de Historia de usuario:** 4
Nombre de la tarea: Desarrollo de las pruebas de aceptación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 16 de abril de 2015	**Fecha de fin:** 18 de abril de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se especifican los escenarios para probar que la Historia de usuario determinar secuencias con el GA ha sido implementada correctamente.	

Tabla A.12. Tarea de ingeniería # 14

Tarea	
Número de tarea: 14	**Número de Historia de usuario:** 5
Nombre de la tarea: Implementación del reporte	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 19 de abril de 2015	**Fecha de fin:** 23 de abril de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se exportan en formato PDF los 5 mejores resultados obtenidos por el GA.	

Tabla A.13. Tarea de ingeniería # 15

Tarea	
Número de tarea: 15	**Número de Historia de usuario:** 5
Nombre de la tarea: Validación de Crear reporte	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 24 de abril de 2015	**Fecha de fin:** 26 de abril de 2015
Programador responsable: Esmaykel Vázquez Avila	

Continúa en la próxima página

Tabla A.13. Continuación de la página anterior

Descripción: Se realizan los métodos necesarios para validar que la opción crear reporte ha sido implementada correctamente.

Tabla A.14. Tarea de ingeniería # 16

Tarea	
Número de tarea: 16	**Número de Historia de usuario:** 5
Nombre de la tarea: Desarrollo de las pruebas de aceptación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 26 de abril de 2015	**Fecha de fin:** 27 de abril de 2015
Programador responsable: Esmaykel Vázquez Avila	
Descripción: Se especifican los escenarios para probar que la Historia de usuario crear reporte ha sido implementada correctamente.	

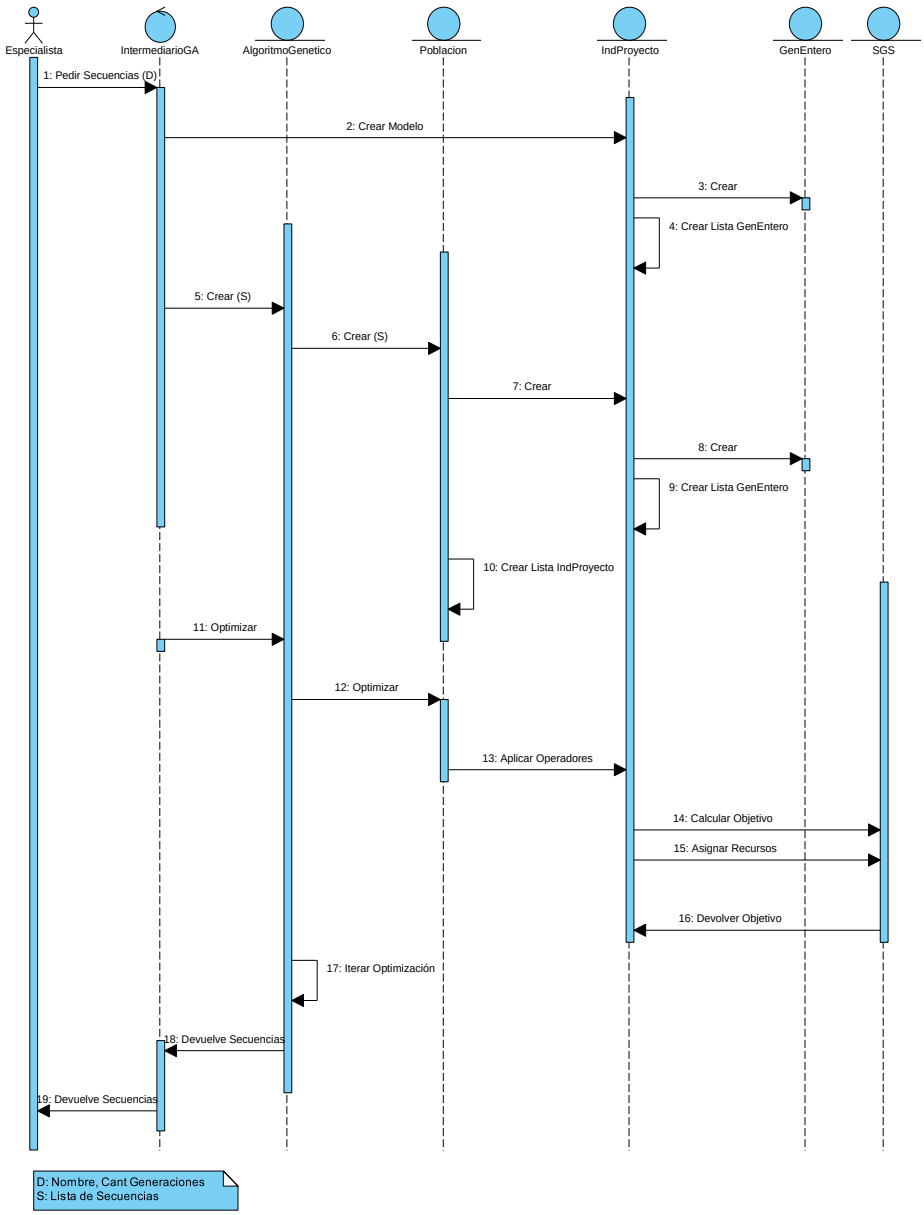

Figura A.1. Diagrama de secuencia del GA

APÉNDICE B

Conjunto de Problemas de PSPLib

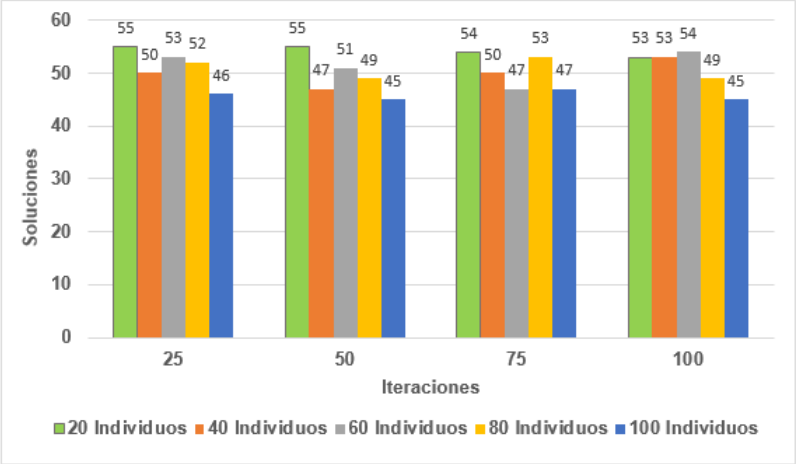

Figura B.1. Resultados para la instancia de prueba *j301-1*

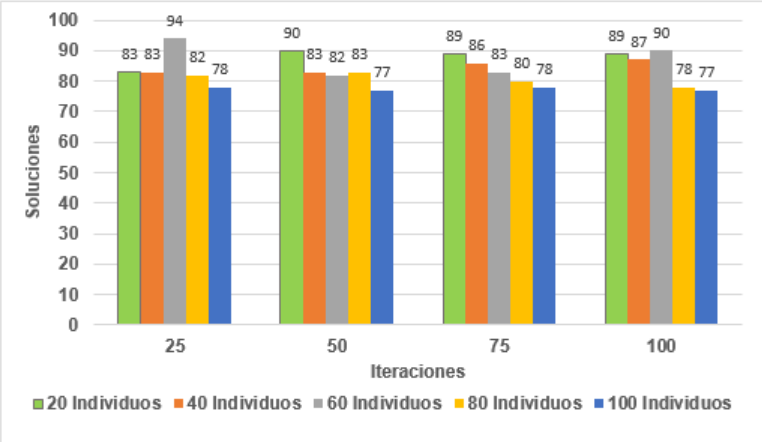

Figura B.2. Resultados para la instancia de prueba *j601-1*

Universidad de las Ciencias Informáticas

Vertex, Entornos Interactivos 3D, Facultad 5

Reporte del Algoritmo Genético para la planificación de tareas de un proyecto de software

Proyecto: Instancia de 30 tareas

Cantidad de Tareas: 32

Cantidad de Soluciones: 80

Recursos:

Tipo Recurso	Cantidad
tipo1	12
tipo2	13
tipo3	4
tipo4	12

Mejores secuencias de planificación de tareas obtenidas:

Solución 1

Tiempo Fin: 47

Núm.	Nombre Actividad	Recursos Asignados
1	Actividad 1	*Actividad Ficticia*
4	Actividad 4	Lorgio, Kit, Fernando
9	Actividad 9	Esmaykel, Miguel, Pepe, Lolo, Pier, Tai
3	Actividad 3	Jorge, Leonardo, Mary, Cristina, Joy, k, Esmaykel, Miguel, Pepe, Lolo
8	Actividad 8	Mario
12	Actividad 12	Enrique, Luis, Pit, Lucas, Walter, Migue, Julio
13	Actividad 13	Jorge, Leonardo, Mary, Cristina
5	Actividad 5	Esmaykel, Miguel, Pepe
7	Actividad 7	Lolo, Joy, k, Pier
18	Actividad 18	Fernanda, Joaquin, Maykel, Gabriel, Gabriele, Susej, Ana
14	Actividad 14	Julia, Yoe, Yane, Frank, Yuly, Mario, Enrique, Luis
10	Actividad 10	Anatol
16	Actividad 16	Angel, Lorgio, Kit, Fernando, Fernanda
21	Actividad 21	Joaquin, Maykel, Gabriel, Gabriele, Susej, Ana
19	Actividad 19	Pit
2	Actividad 2	Tai, Jorge, Leonardo, Mary
11	Actividad 11	Lucas, Walter, Migue, Julio, Mario
15	Actividad 15	Cristina, Esmaykel, Miguel
17	Actividad 17	Anatol, Lorgio, Kit, Fernando, Fernanda, Angel, Joaquin, Maykel
6	Actividad 6	Gabriel, Gabriele, Susej, Ana, Lorgio, Kit, Fernando, Fernanda
27	Actividad 27	Joaquin, Maykel, Anatol, Gabriel, Gabriele, Susej, Ana
29	Actividad 29	Enrique, Luis, Pit, Julia, Yoe, Yane, Frank
20	Actividad 20	Yuly, Enrique, Luis, Pit, Julia, Yoe, Yane, Frank, Mario, Lucas
25	Actividad 25	Pepe, Lolo, Pier, Joy
26	Actividad 26	Josep, Josepe, Martha, Julita
28	Actividad 28	Enrique, Luis, Pit, Julia, Yoe, Yane, Frank, Yuly
22	Actividad 22	k, Tai
23	Actividad 23	Jorge, Leonardo, Mary
24	Actividad 24	Enrique, Luis, Pit, Julia, Yoe, Yane, Frank, Yuly, Walter
31	Actividad 31	Josep, Josepe

30	Actividad 30	Migue, Julio, Mario, Lucas, Enrique, Luis, Pit
32	Actividad 32	*Actividad Ficticia*

Solución 2

Tiempo Fin: 47

Núm.	Nombre Actividad	Recursos Asignados
1	Actividad 1	*Actividad Ficticia*
4	Actividad 4	Lorgio, Kit, Fernando
9	Actividad 9	Esmaykel, Miguel, Pepe, Lolo, Pier, Tai
3	Actividad 3	Jorge, Leonardo, Mary, Cristina, Joy, k, Esmaykel, Miguel, Pepe, Lolo
8	Actividad 8	Mario
12	Actividad 12	Enrique, Luis, Pit, Lucas, Walter, Migue, Julio
13	Actividad 13	Jorge, Leonardo, Mary, Cristina
5	Actividad 5	Fernanda
7	Actividad 7	Julia, Yoe, Yane, Frank, Yuly
18	Actividad 18	Josep, Josepe, Martha, Julita
14	Actividad 14	Joaquin, Maykel, Gabriel, Gabriele, Susej, Ana, Anatol, Angel
10	Actividad 10	Esmaykel, Miguel, Pepe, Lolo
16	Actividad 16	Joy, k, Pier
21	Actividad 21	Tai, Pier, Joy
19	Actividad 19	Tai, Jorge, Leonardo, Mary
2	Actividad 2	Lorgio, Kit, Fernando, Fernanda, Joaquin, Maykel, Gabriel
11	Actividad 11	Mario
15	Actividad 15	Gabriele, Susej, Ana, Anatol, Angel
17	Actividad 17	Lorgio, Kit, Fernando, Fernanda, Joaquin, Maykel, Gabriel
6	Actividad 6	Gabriele, Susej, Ana, Anatol, Angel, Lorgio
27	Actividad 27	Enrique, Luis, Pit, Lucas, Walter, Migue, Julio
29	Actividad 29	Mario, Julia, Yoe, Yane, Frank, Yuly, Enrique, Luis, Pit, Lucas
20	Actividad 20	Walter, Migue, Julio, Mario, Enrique, Luis, Pit, Lucas
25	Actividad 25	Mario, Enrique, Luis, Pit, Lucas, Walter, Migue, Julio
26	Actividad 26	Kit, Fernando, Fernanda, Joaquin, Maykel, Gabriel, Lorgio, Gabriele
22	Actividad 22	Pier, Joy, Cristina, Esmaykel
28	Actividad 28	Miguel, Pepe
31	Actividad 31	Lolo, Tai, Jorge
23	Actividad 23	Julia, Yoe, Yane, Frank, Yuly, Mario, Enrique, Luis, Pit
24	Actividad 24	Josep, Josepe
30	Actividad 30	Lucas, Walter, Migue, Julio, Mario, Enrique, Luis
32	Actividad 32	*Actividad Ficticia*

Solución 3

Tiempo Fin: 54

Núm.	Nombre Actividad	Recursos Asignados
1	Actividad 1	*Actividad Ficticia*
4	Actividad 4	Lorgio, Kit, Fernando
9	Actividad 9	Esmaykel, Miguel, Pepe, Lolo, Pier, Tai
3	Actividad 3	Jorge, Leonardo, Mary, Cristina, Joy, k, Esmaykel, Miguel, Pepe, Lolo
8	Actividad 8	Mario
12	Actividad 12	Enrique, Luis, Pit, Lucas, Walter, Migue, Julio
2	Actividad 2	Jorge, Leonardo, Mary, Cristina
10	Actividad 10	Fernanda
11	Actividad 11	Julia, Yoe, Yane, Frank, Yuly
26	Actividad 26	Josep, Josepe, Martha, Julita
6	Actividad 6	Joaquin, Maykel, Gabriel, Gabriele, Susej, Ana, Anatol, Angel
7	Actividad 7	Esmaykel, Miguel, Pepe, Lolo
15	Actividad 15	Joy, k, Pier
5	Actividad 5	Tai, Pier, Joy
13	Actividad 13	Tai, Jorge, Leonardo, Mary
16	Actividad 16	Lorgio, Kit, Fernando, Fernanda, Joaquin
21	Actividad 21	Maykel, Gabriel, Gabriele, Susej, Ana, Anatol
18	Actividad 18	Angel, Maykel, Gabriel, Gabriele, Susej, Ana, Anatol

20	Actividad 20	Mario, Enrique, Luis, Pit, Lucas, Walter, Migue, Julio, Julia, Yoe
19	Actividad 19	Mario
14	Actividad 14	Mario, Enrique, Luis, Pit, Lucas, Walter, Migue, Julio
17	Actividad 17	Maykel, Gabriel, Gabriele, Susej, Ana, Anatol, Angel, Lorgio
29	Actividad 29	Mario, Enrique, Luis, Pit, Lucas, Walter, Migue
27	Actividad 27	Kit, Fernando, Fernanda, Joaquin, Lorgio, Maykel, Gabriel
22	Actividad 22	Pier, Joy
28	Actividad 28	Julio, Yane, Frank, Yuly, Julia, Yoe, Mario, Enrique
25	Actividad 25	Cristina, Esmaykel, Miguel, Pepe
23	Actividad 23	Lolo, Tai, Jorge
31	Actividad 31	Josep, Josepe
24	Actividad 24	Luis, Pit, Lucas, Walter, Migue, Mario, Enrique, Julio, Julia
30	Actividad 30	Yoe, Yane, Frank, Yuly, Mario, Enrique, Luis
32	Actividad 32	*Actividad Ficticia*

Solución 4

Tiempo Fin: 54

Núm.	Nombre Actividad	Recursos Asignados
1	Actividad 1	*Actividad Ficticia*
3	Actividad 3	Esmaykel, Miguel, Pepe, Lolo, Pier, Tai, Jorge, Leonardo, Mary, Cristina
2	Actividad 2	Joy, k, Esmaykel, Miguel
6	Actividad 6	Lorgio, Kit, Fernando, Fernanda, Joaquin, Maykel, Gabriel, Gabriele
4	Actividad 4	Susej, Ana, Anatol
9	Actividad 9	Pepe, Lolo, Pier, Tai, Jorge, Leonardo
8	Actividad 8	Mario
7	Actividad 7	Mary, Cristina, Pepe, Lolo
15	Actividad 15	Pepe, Lolo, Mary
5	Actividad 5	Cristina, Esmaykel, Miguel
13	Actividad 13	Pier, Tai, Jorge, Leonardo
11	Actividad 11	Enrique, Luis, Pit, Lucas, Walter
10	Actividad 10	Angel
16	Actividad 16	Susej, Ana, Anatol, Angel, Lorgio
21	Actividad 21	Kit, Fernando, Fernanda, Joaquin, Maykel, Gabriel
12	Actividad 12	Migue, Julio, Julia, Yoe, Yane, Frank, Yuly
26	Actividad 26	Josep, Josepe, Martha, Julita
18	Actividad 18	Gabriele, Kit, Fernando, Fernanda, Joaquin, Maykel, Gabriel
20	Actividad 20	Mario, Migue, Julio, Julia, Yoe, Yane, Frank, Yuly, Enrique, Luis
19	Actividad 19	Mario
14	Actividad 14	Mario, Migue, Julio, Julia, Yoe, Yane, Frank, Yuly
17	Actividad 17	Kit, Fernando, Fernanda, Joaquin, Maykel, Gabriel, Gabriele, Lorgio
29	Actividad 29	Mario, Migue, Julio, Julia, Yoe, Yane, Frank
27	Actividad 27	Susej, Ana, Anatol, Angel, Lorgio, Kit, Fernando
22	Actividad 22	Pier, Tai
28	Actividad 28	Yuly, Pit, Lucas, Walter, Enrique, Luis, Mario, Migue
25	Actividad 25	Jorge, Leonardo, Joy, k
31	Actividad 31	Josep, Josepe
23	Actividad 23	Esmaykel, Miguel, Cristina
24	Actividad 24	Julio, Julia, Yoe, Yane, Frank, Mario, Enrique, Luis, Pit
30	Actividad 30	Lucas, Walter, Migue, Yuly, Mario, Enrique, Luis
32	Actividad 32	*Actividad Ficticia*

Solución 5

Tiempo Fin: 54

Núm.	Nombre Actividad	Recursos Asignados
1	Actividad 1	*Actividad Ficticia*
4	Actividad 4	Lorgio, Kit, Fernando
9	Actividad 9	Esmaykel, Miguel, Pepe, Lolo, Pier, Tai
3	Actividad 3	Jorge, Leonardo, Mary, Cristina, Joy, k, Esmaykel, Miguel, Pepe, Lolo
8	Actividad 8	Mario
2	Actividad 2	Jorge, Leonardo, Mary, Cristina

6	Actividad 6	Fernanda, Joaquin, Maykel, Gabriel, Gabriele, Susej, Ana, Anatol
11	Actividad 11	Enrique, Luis, Pit, Lucas, Walter
15	Actividad 15	Esmaykel, Miguel, Pepe
5	Actividad 5	Lolo, Joy, k
7	Actividad 7	Pier, Tai, Lolo, Joy
10	Actividad 10	Angel
12	Actividad 12	Migue, Julio, Julia, Yoe, Yane, Frank, Yuly
13	Actividad 13	k, Jorge, Leonardo, Mary
18	Actividad 18	Lorgio, Kit, Fernando, Angel, Fernanda, Joaquin, Maykel
14	Actividad 14	Mario, Migue, Julio, Julia, Yoe, Yane, Frank, Yuly
16	Actividad 16	Gabriel, Gabriele, Susej, Ana, Anatol
21	Actividad 21	Lorgio, Kit, Fernando, Fernanda, Joaquin, Maykel
26	Actividad 26	Josep, Josepe, Martha, Julita
20	Actividad 20	Mario, Migue, Julio, Julia, Yoe, Yane, Frank, Yuly, Enrique, Luis
19	Actividad 19	Mario
17	Actividad 17	Angel, Gabriel, Gabriele, Susej, Ana, Anatol, Lorgio, Kit
29	Actividad 29	Pit, Lucas, Walter, Mario, Enrique, Luis, Migue
27	Actividad 27	Fernando, Fernanda, Joaquin, Maykel, Lorgio, Kit, Gabriel
28	Actividad 28	Julio, Julia, Yoe, Yane, Frank, Yuly, Mario, Enrique
25	Actividad 25	Cristina, Lolo, Pier, Tai
31	Actividad 31	Josep, Josepe
22	Actividad 22	Joy, Jorge
23	Actividad 23	Leonardo, Mary, k
24	Actividad 24	Luis, Pit, Lucas, Walter, Migue, Mario, Enrique, Julio, Julia
30	Actividad 30	Yoe, Yane, Frank, Yuly, Mario, Enrique, Luis
32	Actividad 32	*Actividad Ficticia*

Este reporte fue generado en Qt Creator el 02/06/2015