Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00059
Matrícula: 2143805414

PLANEACIÓN DE PROYECTOS DE DESARROLLO DE SOFTWARE USANDO TÉCNICAS DE OPTIMIZACIÓN

En la Ciudad de México, se presentaron a las 9:00 horas del día 26 del mes de enero del año 2017 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DR. ANTONIO LOPEZ JAIMES
MTRO. VICTOR HUGO GOMEZ GOMEZ
DR. HUMBERTO GUSTAVO CERVANTES MACEDA

Bajo la Presidencia del primero y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS (CIENCIAS Y TECNOLOGIAS DE LA INFORMACION)

DE: MIGUEL ANGEL VEGA VELAZQUEZ

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

APROBAR

Acto continuo, el presidente del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

MIGUEL ANGEL VEGA VELAZQUEZ
ALUMNO

REVISÓ

LIC. JULIO CESAR DE LARA ISASSI
DIRECTOR DE SISTEMAS ESCOLARES

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JOSE GILBERTO CORDOBA HERRERA

PRESIDENTE

DR. ANTONIO LOPEZ JAIMES

VOCAL

MTRO. VICTOR HUGO GOMEZ GOMEZ

SECRETARIO

DR. HUMBERTO GUSTAVO CERVANTES MACEDA

UNIVERSIDAD AUTÓNOMA METROPOLITANA - IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

PLANEACIÓN DE PROYECTOS DE DESARROLLO DE SOFTWARE USANDO TÉCNICAS DE OPTIMIZACIÓN

Tesis que presenta
Miguel Ángel Vega Velázquez
Para obtener el grado de
Maestro en ciencias y tecnologías de la información

Asesores: Dr. Humberto Gustavo Cervantes Maceda
Dr. Abel García Nájera

Jurado calificador:
Presidente: Dr. Antonio López Jaimes
Secretario: Dr. Humberto Gustavo Cervantes Maceda
Vocal: M.C. Víctor Hugo Gómez

Ciudad de México, Enero 2017

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por haberme otorgado el apoyo financiero para la realización del presente proyecto de investigación.

A mis asesores el Dr. Humberto Cervantes y el Dr. Abel García por la confianza depositada en mí, por el tiempo dedicado al seguimiento del proyecto y por las valiosas observaciones hechas al presente documento.

Al Dr. Antonio López y al M.C. Víctor Hugo Gómez por haber aceptado la invitación para fungir como miembros del jurado evaluador de esta tesis y por las observaciones y comentarios realizados, los cuales enriquecieron el presente documento.

A la Universidad Autónoma Metropolitana, por haberme abierto las puertas una vez más, ahora para desarrollar mis estudios de maestría, permitiéndome crecer profesionalmente.

A mis profesores y compañeros del Posgrado en Ciencias y Tecnologías de la Información por haber compartido sus conocimientos y experiencias conmigo durante este periodo de maestría. A Fabiola Camilo por ser una gran compañera y amiga.

A mis padres Eloina Velázquez y Leonardo Vega por su cariño y por sus consejos, y a mis hermanos por poder contar con ellos en todo momento.

Contenido

Resumen 8

Capítulo 1. Introducción 9

1.1. Contexto 9

1.1.1. La fase de planeación de un proyecto de software 9

1.1.2. Escenarios de planeación 11

1.2. Descripción del problema 12

1.3. Propuesta de solución 12

1.4. Objetivos generales y específicos... 13

1.4.1. Objetivo general 13

1.4.2. Objetivos específicos. 13

1.5. Metodología 14

1.6. Justificación 15

1.7. Hipótesis 15

1.8. Estructura del documento. 16

Capítulo 2. Estado del arte 17

2.1. El problema de planificación de proyectos de software. 17

2.2. Técnicas de optimización para resolver el problema. 19

2.3. Trabajo relacionado 20

2.4. Comparación de trabajos. 30

2.5. Análisis de la revisión bibliográfica. 33

2.6. Conclusiones de la revisión bibliográfica 34

Capítulo 3. Desarrollo de la propuesta 35

3.1. Elementos principales de la herramienta para generar escenarios 35

3.2. El modelo de planeación para la generación de escenarios 36

3.2.1. Requerimientos y decisiones de diseño para el modelo de planeación 36

3.2.2. Entidades del modelo de planeación 37

3.2.3. Representación de una solución. 41

3.2.4. Cálculo de la duración y costo de un proyecto 41

3.2.5. Restricciones del modelo 43

3.3. Algoritmos de optimización para la generación de escenarios 43

3.3.1. NSGA-II 45

3.3.2. SPEA2. 46

3.3.3. SMS-EMOA. 47

3.4. Eliminación de sobrecarga en las soluciones. 47

3.5. Visión de la herramienta GENESPLAN 47

3.5.1. Requerimientos de negocio 49

3.5.2. Enunciado de visión y características principales. 49

3.5.3. Flujo de trabajo para generar escenarios de planeación con GENESPLAN 50

3.5.4. Requerimientos a nivel de usuario: casos de uso . . . . .

3.5.5. Atributos de calidad 51
3.6. El diseño de GENESPLAN 53
3.6.1. Decisiones de diseño. 54
3.6.2. La arquitectura de GENESPLAN.. 54
3.7. Interfaces gráficas del prototipo de GENESPLAN 58
3.7.1. Interfaces gráficas para el caso de uso "Generar escenarios de planeación" . . . 62
3.7.2. Interfaz para el despliegue de asignaciones 62
3.7.3. Interfaz de despliegue de calendario de tareas 63
Capítulo 4. Experimentos y resultados 64
4.1. Diseño de casos de prueba 66
4.2. Variación en el nivel de productividad de los empleados 66
4.3. Variación en el número de empleados 67
4.4. Empleados con características mixtas 70
4.5. Datos aleatorios en los parámetros de un proyecto. 73
4.6. Discusión de resultados. 76
Capítulo 5. Conclusiones. 81
5.1. Síntesis 83
5.2. Conclusiones. 83
5.3. Contribuciones 84
5.4. Trabajo a futuro 85
Anexo A. Posibles escenarios para un plan de proyecto 85
Anexo B. Parámetros de un caso de prueba 86
Anexo C. Casos de prueba con datos aleatorios. 88
Bibliografía 89
101

Índice de Figuras

1.1. Calendario de tareas de un proyecto de desarrollo de software 10
1.2. Propuesta para la generación de escenarios de planeación. 13
2.1. Elementos del problema de planificación de proyectos de software [Alba2007] 17
2.2. Frente de Pareto para dos funciones objetivo 20
2.3. Representación de una solución en el trabajo de Alba y Chicano [Alba2007]. 20
2.4. Hipervolumen dominado por un Frente de Pareto en un problema de optimización con dos objetivos. 22
2.5. División de una tarea en un grafo en [Xiao2013]. 24
2.6. Representación de una solución al problema de planificación de proyectos de software en el trabajo de [García2014] 26
2.7. Modelo de planeación para resolver el problema de planificación de proyectos de software en [García2014]. 27
2.8. Modelo de planeación para resolver el problema de optimización en [Ruhe2009] 27
2.9. Modelo de planeación para resolver el problema de asignación de tareas en [Duggan2004]. 28
2.10. Modelo para resolver el problema de calendarización de proyectos en [Chicano2012] . . 29
3.1. Entidades del modelo de planeación en GENESPLAN 37
3.2. Ejemplo de un perfil analista de requerimientos 38
3.3. Ejemplo de un perfil arquitecto de software. 38
3.4. Ejemplo de un perfil mixto. 38
3.5. Ejemplos de niveles de productividad para un perfil arquitecto de software. 39
3.6. Ejemplo de conformación de equipo para un proyecto de desarrollo de software 40
3.7. Representación de un grafo de precedencia de tareas (GPT). 40
3.8. Representación de una solución para el problema de generación de escenarios. 41
3.9. Cálculo de la duración efectiva de un proyecto por el método de ruta crítica. 42
3.10. Proceso de selección de soluciones candidatas en NSGA-II [Deb2002]. 45
3.11. Calendario de proyecto con presencia de sobrecarga. 48
3.12. Calendario de proyecto con eliminación de sobrecarga 48
3.13. Flujo de trabajo para la generación de escenarios usando la herramienta GENESPLAN. . 51
3.14. Diagrama de casos de uso que soporta la herramienta GENESPLAN. 51
3.15. Estructura general de jMetal [Durillo2011] 57
3.16. Calendario de un proyecto creado con SwiftGantt. 58
3.17. Diagrama de paquetes que representa la vista lógica general de GENESPLAN. 59
3.18. Diagrama de comportamiento que representa el flujo principal del caso de uso primario "Generar escenarios de planeación". 61
3.19. Interfaz gráfica para generar escenarios de planeación.. 62
3.20. Interfaz gráfica de despliegue de escenarios de planeación 63
3.21. Interfaz que despliega las asignaciones entre tareas y empleados para un

determinado escenario de planeación 64
3.22. Interfaz que despliega el calendario de tareas asociado a un escenario de planeación . . 64
4.1. Duración de un proyecto simple con un número variable de empleados 71
4.2. Duración promedio máxima y duración promedio mínima de proyectos con 20 y 30 tareas al formar combinaciones de 8 empleados con niveles de productividad bajo y alto . 74
4.3. Costo promedio máximo y mínimo de proyectos con 20 y 30 tareas al formar combinaciones de 8 empleados con costos bajos y altos. 76
4.4. Frente de Pareto obtenido en una de las ejecuciones utilizando un proyecto de 30 tareas y 8 empleados, utilizando datos aleatorios 80
4.5. Frente de Pareto obtenido en una de las ejecuciones utilizando un proyecto de 30 tareas y 16 empleados, utilizando datos aleatorios 81

Índice de Tablas

1.1. Ejemplos de escenarios de planeación para un proyecto de software 11
2.1. Cuadro comparativo de los trabajos revisados que resuelven el problema de planificación de proyectos de software. 32
3.1. Decisiones de diseño para el modelo de planeación en GENESPLAN. 36
3.2. Notación utilizada en los algoritmos de optimización NSGA-II y SPEA2. 44
3.3. Características consideradas para el prototipo de GENESPLAN 50
3.4. Documentación del caso de uso CU-01 "Generar escenarios de planeación". 52
3.5. Documentación del caso de uso CU-02 "Consultar matriz de asignaciones de un escenario de planeación" 53
3.6. Documentación del caso de uso "Consultar calendario de tareas". 53
3.7. Escenario de atributo de modificabilidad para agregar un algoritmo de optimización 54
3.8. Decisiones de diseño para la implementación de GENESPLAN. 55
3.9. Elementos principales de la arquitectura de GENESPLAN. 61
4.1. Duración promedio mínima y duración promedio máxima en horas de un proyecto con 30 tareas y 8 empleados, al variar el nivel de productividad. 68
4.2. Costo promedio mínimo y costo promedio máximo de un proyecto con 30 tareas y 8 empleados, al variar el nivel de productividad y dejar el costo fijo 69
4.3. Promedio de soluciones al resolver el grupo de casos en el cual hay variación en el nivel de productividad de los empleados. 70
4.4. Duración promedio mínima y duración promedio máxima en horas de un proyecto con 30 tareas, al variar el número de empleados 71
4.5. Costo promedio mínimo y costo promedio máximo de un proyecto con 30 tareas al variar el número de empleados 72
4.6. Promedio de soluciones obtenidas al resolver el grupo de casos con variación en el número de empleados 73
4.7. Promedio de soluciones obtenidas al resolver el grupo de casos en el cual los empleados presentan características mixtas. 75
4.8. Duración promedio máxima y duración promedio mínima al resolver el grupo de casos de prueba con datos aleatorios 77
4.9. Costo promedio mínimo y costo promedio máximo de proyectos al resolver el grupo de casos de prueba con datos aleatorios. 79
4.10. Promedio de soluciones obtenidas al resolver el grupo de casos de prueba con datos aleatorios. 80

Resumen

Cuando se realiza la planeación de un proyecto de software es deseable disponer de alternativas en términos de planes del proyecto con el fin de tomar decisiones basadas en el tiempo y costo asociados a cada alternativa. A estas alternativas las hemos denominado escenarios de planeación. Se ha observado que la generación de escenarios de forma manual regularmente requiere de un tiempo considerable y es propensa a errores, por lo cual es deseable disponer de algún mecanismo que permita realizar esta actividad de forma menos costosa y más precisa. Cabe señalar que la generación de escenarios representa un problema de optimización multiobjetivo, debido a que normalmente la duración y el costo de un proyecto son objetivos que se encuentran en conflicto. En este trabajo se propone el desarrollo de una herramienta para generar escenarios, la cual se basa en un modelo de planeación que toma en cuenta como características las tareas y los recursos humanos asociados a un proyecto. El modelo propuesto permite considerar diversas conformaciones de equipos, como sucede generalmente en las organizaciones de desarrollo de software. A partir de este modelo, es posible obtener escenarios aplicando técnicas de optimización, y en particular, metaheurísticas basadas en la teoría de la evolución de Darwin. Los resultados obtenidos demuestran que la herramienta propuesta permite generar escenarios sensatos para un plan de proyecto.

Palabras clave: escenarios de planeación, modelo de planeación, algoritmos de optimización, GENESPLAN.

Capítulo 1

Introducción

1.1. Contexto

La administración de proyectos es un proceso estratégico que se compone de fases como la concepción, la planeación, la ejecución y monitoreo, y el cierre. En estas fases se planea y se organiza el trabajo mediante la asignación y el control de los recursos necesarios durante un periodo de tiempo limitado, para cumplir con los objetivos de un proyecto [Jurison1999]. Desde otra perspectiva, la administración de proyectos es una actividad en la cual se aplican los conocimientos, las habilidades, las herramientas y las técnicas necesarias para satisfacer los requerimientos de un proyecto [Torres2014]. Uno de los principales roles en este proceso es el administrador de proyectos, quien es el responsable de planear y coordinar las actividades necesarias para que el proyecto llegue a su estado final, con la mayor eficiencia posible en cada una de las fases y cubriendo todas las expectativas.

El trabajo que se describe en el presente documento se enfoca particularmente en la fase de *planeación* de un proyecto de *desarrollo de software*. En esta fase, una de las actividades principales que coordina el administrador del proyecto es la definición de un plan que permita cumplir con los requerimientos y objetivos establecidos para un proyecto. Al elaborar un plan, es deseable disponer de escenarios alternativos en términos de estimación de costo y duración, con el fin de analizar y definir la opción más viable para que se lleve a cabo un proyecto. Sin embargo, generar escenarios es una actividad manual que requiere esfuerzo, principalmente porque se puede invertir un tiempo no razonable incluso para generar pocos escenarios, y además, porque resulta difícil encontrar escenarios cercanos al óptimo. Por estas razones, en el presente trabajo se propone una solución que simplifique la generación de escenarios para planes de proyecto de software.

1.1.1. La fase de planeación de un proyecto de software

En el contexto del desarrollo de software, desafortunadamente una buena administración no garantiza el éxito de un proyecto; sin embargo, una mala administración puede llevar al fracaso. Algunas de las razones podrían ser que el producto o el sistema de software se retrasa y es entregado fuera de tiempo, los costos del proyecto son mayores que los estimados o los requerimientos no se cumplen.

Pese a los problemas que se presenten, la administración de proyectos es una disciplina esencial durante el ciclo de vida de un proyecto, por medio de la cual se busca desarrollar el trabajo asignado de manera eficiente, para cumplir con los requerimientos y los objetivos establecidos.

La administración de un proyecto comienza con las actividades de planeación, asumiendo que se ha definido previamente el alcance del proyecto. Primero, el administrador debe identificar las actividades necesarias para completar el proyecto y debe seleccionar al personal adecuado para desarrollar esas actividades. En ocasiones el presupuesto puede ser limitado y, por ello, se opta por utilizar personal con costo más bajo y con menor experiencia.

Una vez identificadas las actividades y el equipo de trabajo, se establece un plan de proyecto. El plan de proyecto permite guiar el desarrollo de las actividades hacia el cumplimiento de los objetivos y los requerimientos.

En un plan típicamente se especifican:

- Las fases del proyecto.
- El calendario de actividades y tareas, con dependencias y con recursos asignados.
- Las posibles restricciones y/o suposiciones.

En este contexto, una actividad es considerada como un conjunto de *tareas* que permiten completar una parte del proyecto y que tiene una duración definida. Las tareas a su vez, son unidades de trabajo que se asocian a un solo responsable, que pueden desarrollarse en un periodo de tiempo relativamente corto (horas o días) y que pueden tener dependencias con otras tareas. Las tareas pueden ser clasificadas según la fase o disciplina del proyecto en la que se desarrollan, por ejemplo, requerimientos, diseño, implementación, pruebas, etc. [Jacobson2000].

Una vez especificadas las tareas del proyecto, se les asocia un esfuerzo cuantificado. Para estimar el esfuerzo de las tareas, se pueden tomar en cuenta algunos criterios como su tamaño en una unidad cuantificable de software (por ejemplo: puntos de función, líneas de código, número de componentes, etc.) o su nivel de complejidad estimado (simple, medio o difícil). Dicha estimación se puede basar en datos históricos, en algún modelo de estimación de software o simplemente en la "experiencia" [Pressman2005].

Después de cuantificar el esfuerzo, se distribuye el trabajo entre los recursos disponibles, quienes aplican los conocimientos técnicos y las habilidades necesarias para completar las tareas del proyecto, y que por ello, perciben un costo.

Posteriormente se realiza un mapeo entre el esfuerzo estimado de las tareas y el tiempo real de desarrollo de las mismas. Algunos recursos pueden tener características que pueden afectar la duración real de una tarea, por ejemplo, la tasa de productividad que poseen con base en su nivel de experiencia y/o el porcentaje de tiempo que pueden dedicar a las tareas del proyecto (ej. medio tiempo o tiempo completo). Si la tasa de productividad es baja, las tareas pueden tener mayor duración y viceversa. Lo mismo sucede con el porcentaje de disponibilidad de los empleados, cuanto menor sea este, mayor tiempo-calendario les tomará completar sus tareas.

Finalmente, se realiza la calendarización de las tareas con sus respectivas fechas de inicio y de finalización, sus dependencias y los recursos asociados. En la Figura 1.1 se presenta un ejemplo de calendario de tareas para un proyecto de desarrollo de software, que generalmente es representado mediante un cronograma o diagrama de Gantt.

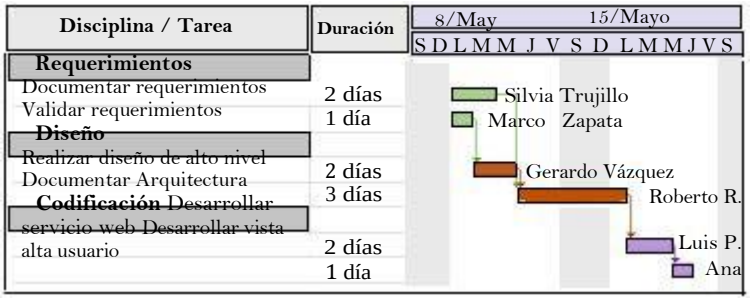


Figura 1.1. Calendario de tareas de un proyecto de desarrollo de software.

A partir de la información del plan, es posible determinar el presupuesto y la fecha de entrega del proyecto, o equivalentemente, el costo y la duración. La duración puede especificarse en alguna unidad de tiempo como días, semanas, meses o años; mientras que el costo generalmente se especifica en alguna unidad monetaria.

1.1.2. Escenarios de planeación

Uno de los problemas que enfrentan muchas organizaciones de desarrollo de software durante la fase de planeación, es la forma de distribuir al personal en las actividades y tareas, de tal manera que el proyecto se desarrolle en el menor tiempo posible y con el menor presupuesto. Desafortunadamente no siempre es posible lograrlo, debido a que la duración y el costo de un proyecto se encuentran en conflicto [Chicano2011]. Esto significa que si se requiere disminuir la duración del proyecto, seguramente el costo del mismo se incrementará, pues se tendrán que asignar recursos con mayor experiencia y/o productividad, siendo más alto el costo que perciben. Análogamente, si a las mismas tareas se asignan recursos con menor experiencia y con un costo un costo más bajo, el costo del proyecto será menor, pero se espera que la duración sea mayor. Por lo tanto, se dice que el costo y la duración de un proyecto presentan una correlación inversa [Chicano2011].

Los miembros de un equipo de proyecto regularmente presentan características diversas como la tasa de productividad, un perfil relativo a las disciplinas del ciclo de desarrollo que dominan, el conocimiento en distintas tecnologías y por supuesto, el costo. Dadas estas características, es posible distribuir los recursos en las tareas de distintas maneras, obteniendo estimaciones de tiempo y costo distintos. A cada una de las posibles formas de asociar recursos a las tareas, se le ha denominado *escenario de planeación*.

Se ha observado que en algunas organizaciones de desarrollo de software los recursos pueden ser clasificados de distintas maneras con base en su nivel de experiencia. Una manera de realizar esta clasificación es como se sugiere en [García2014], donde el nivel de experiencia más bajo para un recurso es "principiante" y por esta razón, percibe el costo más bajo. Los siguientes niveles en orden creciente son junior, senior y experto; donde este último corresponde al costo más alto. En la práctica, esta clasificación puede hacerse tomando en cuenta las características de los recursos. Además, los recursos se pueden especializar en una o más disciplinas del ciclo de desarrollo de software, por ejemplo en requerimientos, en diseño, en implementación, en pruebas, etc. [Jacobson2000]. De esta manera, los recursos podrían tener asociado un perfil, por ejemplo: analista de requerimientos, arquitecto de software, programador junior, programador senior, etc. Desde luego, cada organización puede definir su propia clasificación de recursos, o bien, podrían no tener definida alguna clasificación. En la Tabla 1.1 se presentan ejemplos de escenarios de planeación.

No. de escenario	Recursos asignados	Duración estimada	Costo estimado
1	5 recursos de nivel junior y 2 recursos de nivel senior	6 meses	$90,000
2	3 recursos de nivel junior y 4 recursos de nivel senior	4 meses	$110,000
3	2 recursos de nivel junior y 5 recursos de nivel senior	3 meses	$120,000

Tabla 1.1. Ejemplos de escenarios de planeación para un proyecto de software.

Cada escenario tiene asociado un equipo de recursos, una duración y un costo estimado. Los recursos tienen asociado un nivel de experiencia, en este caso, junior o senior. La característica principal de estos escenarios es la *no dominancia*, esto es, ninguno es superior o inferior a otros, tanto en el costo como en la duración del proyecto. El escenario que presenta la mayor duración, presenta el menor costo y viceversa.

1.2. Descripción del problema

Dado el contexto descrito anteriormente, es deseable disponer de escenarios de planeación alternativos en etapas tempranas de un proyecto, con el fin de analizarlos y determinar la opción más viable o factible para que este se lleve a cabo, sin embargo, se ha observado que en diversas organizaciones de desarrollo de software, la generación de escenarios se realiza de forma "manual". La generación de escenarios de forma manual presenta algunos problemas, pues regularmente se requiere de un tiempo considerable y es propensa a errores, debido a que se utilizan mecanismos como hojas de cálculo o software de administración de proyectos para realizar las estimaciones correspondientes.

Por otra parte, no resulta práctico generar todos los escenarios posibles de un plan de proyecto para obtener los óptimos, ya que el número de estos se puede incrementar de manera exponencial, incluso si el número de tareas y empleados es relativamente bajo.

Debido a los problemas anteriores, se requiere encontrar una manera sencilla de generar escenarios de planeación, que sea menos costosa y más precisa.

1.3. Propuesta de solución

Una manera de simplificar el problema de la generación de escenarios de forma manual, es por medio de una herramienta que automatice o semi-automatice esta actividad. Hasta el momento de redactar la presente idónea comunicación de resultados, se tuvo desconocimiento de herramientas para este propósito, por lo cual se identificó la oportunidad de proponer y desarrollar una.

Para desarrollar una herramienta de generación de escenarios, es necesario considerar al menos los siguientes dos elementos: un *modelo de planeación* para representar las características de las tareas y de los recursos de un plan de proyecto y una *técnica de optimización* que utilice los datos de esas características para generar escenarios alternativos en términos de estimación de tiempo y costo.

De acuerdo con los trabajos revisados sobre el problema de planificación de proyectos, se observó que la mayoría de los modelos de planeación presentan limitaciones que pueden dificultar su adopción en proyectos de software reales. Un ejemplo de estos modelos es el propuesto por Alba y Chicano [Alba2007], cuyas características se presentan en el diagrama UML de la Figura 2.1. Las principales limitaciones encontradas en este y otros modelos, es que no es posible representar perfiles de empleados (por ejemplo: arquitectos de software, programadores, testers, etc.), y además, se asume que los empleados presentan habilidades específicas, lo cual difícilmente se observa en la práctica. Ante estas limitaciones, se propuso un modelo de planeación que intenta aproximarse un poco más a lo que se ha observado en algunas organizaciones de desarrollo.

Por otra parte, los trabajos revisados sugieren el uso de técnicas de optimización como las heurísticas para resolver el problema de planeación de proyectos, debido a que no existen técnicas

exactas que permitan resolverlo. Las heurísticas permiten obtener buenas soluciones en un tiempo razonable, sin embargo, no garantizan obtener las óptimas.

En la Figura 1.2 se presenta un esquema general de la propuesta para apoyar en la generación de escenarios de planeación. La herramienta propuesta toma como parámetros de entrada los datos de las tareas y de los recursos de un plan de proyecto. Estos datos son representados en un modelo de planeación y almacenados en

una estructura de datos. Los datos son utilizados por un algoritmo de optimización, el cual se encarga de calcular y evaluar de manera iterativa, la duración y el costo para un posible conjunto de soluciones, hasta obtener los escenarios finales.

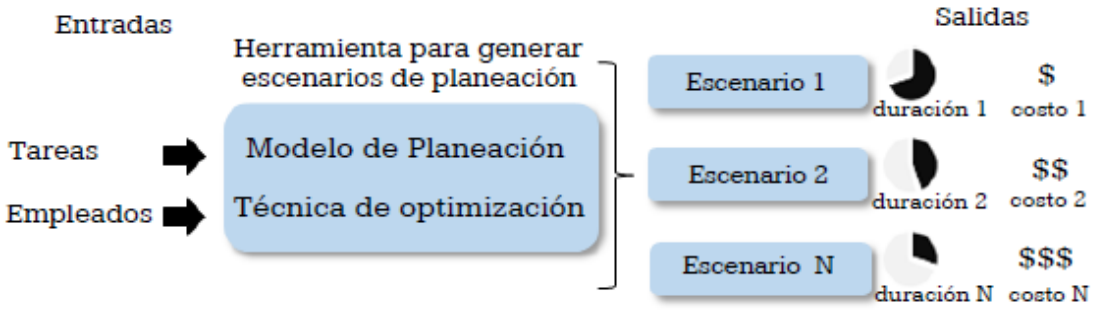


Figura 1.2 Propuesta para la generación de escenarios de planeación.

Uno de los aspectos deseables, es que se puedan modelar distintas organizaciones de desarrollo, en las cuales los empleados pueden ser clasificados de distintas formas, por lo cual, este requerimiento ha sido considerado en el diseño del modelo de planeación.

1.4. Objetivos generales y específicos

A continuación se describen los objetivos del proyecto.

1.4.1. Objetivo general

Con base en el problema descrito en la sección 1.2, se ha planteado el siguiente objetivo general:

- Desarrollar una herramienta que permita obtener escenarios de planeación, con base en la configuración de parámetros asociados a proyectos de desarrollo de software.

1.4.2. Objetivos específicos

Los objetivos específicos que se han definido son los siguientes:

- Revisar el estado del arte sobre el uso de técnicas de optimización en la planeación de proyectos de desarrollo de software.
- Proponer y desarrollar un modelo de planeación flexible para la generación de escenarios de planeación, tomando en cuenta un conjunto de variables.
- Identificar las técnicas de optimización más adecuadas para la generación de escenarios de planeación.

- Implementar en una herramienta el modelo de planeación propuesto y las técnicas de optimización identificadas.

- Evaluar el modelo de planeación y las técnicas de optimización implementados en la herramienta para generar escenarios.

1.5. Metodología

Para cumplir con los objetivos mencionados previamente, se siguió una metodología que consta de las siguientes fases:

1.- *Revisión bibliográfica.* La actividades consideradas en esta fase fueron las siguientes:

- Realizar un estudio sistemático de la literatura para la selección de artículos relacionados con el uso de técnicas de optimización en la planeación de proyectos de desarrollo de software.

- Leer los artículos seleccionados, elaborar los resúmenes correspondientes y sintetizarlos en un cuadro comparativo.

- Analizar los artículos revisados y el cuadro comparativo de trabajos, para obtener conclusiones al respecto e identificar oportunidades de mejora.

- Proponer una solución al problema de la generación de escenarios de planeación.

- Redactar las secciones correspondientes en el documento para la idónea comunicación de resultados.

2.- *Desarrollo de la propuesta.* Las actividades propuestas en esta fase fueron las siguientes:

- Diseñar un modelo de planeación que permita representar de mejor manera las características de los recursos y de las tareas de un proyecto de software.

- Identificar las técnicas de optimización más apropiadas para la herramienta.

- Desarrollar el prototipo de la herramienta para generar escenarios de planeación, incorporando el modelo de planeación y las técnicas de optimización identificadas.

- Incluir documentación y revisión bibliográfica que se vaya generando.

- Redactar el capítulo correspondiente al desarrollo de la propuesta en el documento para la idónea comunicación de resultados.

3.- *Evaluación.* Las actividades correspondientes a esta fase fueron las siguientes:

- Definir un conjunto de casos de prueba para generar escenarios de planeación, considerando las variables del modelo propuesto.

- Ejecutar las pruebas correspondientes y recopilar los resultados.

- Analizar y reportar los resultados obtenidos en el presente documento.

1.6. Justificación

Una herramienta que permita generar escenarios de planeación puede servir de apoyo para las personas que se dedican a realizar estimaciones de costo y duración en etapas tempranas de un proyecto. Un ejemplo podría ser en la etapa de preventa[1], en la cual distintas organizaciones de desarrollo (proveedores) compiten entre sí para realizar una propuesta de solución para un determinado cliente. Las preguntas más comunes que hacen los clientes a los proveedores sobre el desarrollo de un sistema de software son: ¿cuánto costará? y ¿para cuándo estará terminado?. Desde luego no es fácil responder a estas preguntas, pues es necesario hacer un análisis previo y realizar actividades intermedias para obtener una estimación al respecto. Con escenarios alternativos de planeación, existe una mayor posibilidad de que un cliente y un proveedor tengan oportunidad de negociar y aceptar el proyecto.

Otro posible uso de la herramienta puede ser al momento de planear una iteración en un proyecto. Las iteraciones pueden considerarse como "miniproyectos" o proyectos de corta duración, en los cuales se realiza un proceso de trabajo similar para desarrollar partes funcionales de un sistema de software, el cual va evolucionando hasta obtener el sistema final. En cada iteración se identifican tareas que comúnmente son asignadas a recursos que tengan los conocimientos suficientes para su desarrollo, por lo cual, la herramienta propuesta puede generar sugerencias de asignaciones entre tareas y recursos y presentar opciones de calendarización de tareas.

Las principales ventajas que se esperan obtener con el desarrollo de la herramienta propuesta son las siguientes:

- Disponer de escenarios de planeación alternativos de forma inmediata en etapas tempranas de un proyecto de desarrollo de software.
- Simplificar el esfuerzo invertido por las personas que se dedican a realizar estimaciones de costo y duración en etapas tempranas de un proyecto.
- La flexibilidad para modelar distintos tipos de organizaciones de desarrollo de software y conformaciones de equipos para un proyecto.
- Incluso, podría ser de utilidad en la planeación de proyectos que no necesariamente sean de desarrollo de software.

1.7. Hipótesis

La hipótesis en la cual se basa el presente trabajo es la siguiente:

- Por medio de la herramienta propuesta, es posible generar escenarios de planeación alternativos, a partir de las características de las tareas y de los empleados que participan en un proyecto de desarrollo de software. Además, con el modelo de planeación integrado en la herramienta, es posible modelar distintas conformaciones de equipos de desarrollo en organizaciones, en las cuales los empleados pueden ser clasificados por su experiencia y sus habilidades en ciertas fases de un proyecto.

[1] La preventa es la etapa que ocurre cuando se hace una propuesta de solución para desarrollar un sistema de software.

1.8. Estructura del documento

El presente documento está estructurado de la siguiente forma:

- Capítulo 2: Estado del arte. Se presenta un resumen de los artículos revisados que se enfocan en el uso de técnicas de optimización para resolver el problema de planificación de proyectos de desarrollo de software. Además, se agrega un cuadro comparativo de los trabajos revisados y se describen las conclusiones de la revisión bibliográfica.

- Capítulo 3: Desarrollo de la propuesta. Se describen los aspectos más relevantes sobre el desarrollo de la herramienta propuesta, como el modelo de planeación, los algoritmos de optimización para obtener las soluciones y la descripción del diseño y las interfaces gráficas.

- Capítulo 4: Experimentos y resultados. Se describen los resultados obtenidos en la parte experimental utilizando la herramienta propuesta, la cual se llevó a cabo mediante la resolución de grupos de casos de prueba en base a las variables del modelo de planeación.

- Capítulo 5: Conclusiones y trabajo a futuro. Se presenta una síntesis sobre el trabajo realizado, unas conclusiones con base en los objetivos planteados, las principales contribuciones del trabajo y algunas recomendaciones para un trabajo a futuro.

Al final del documento se agregan los anexos correspondientes y la bibliografía empleada.

Capítulo 2

Estado del arte

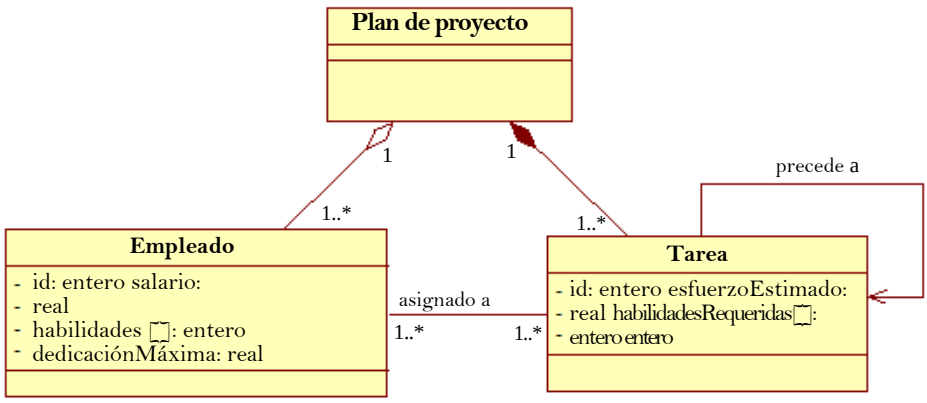
La generación de escenarios de planeación implica obtener soluciones alternativas en términos de estimación de duración y costo para un determinado proyecto de software. Dado que ambos factores están en conflicto [Chicano2011], esta situación representa un problema de optimización que comúnmente es conocido en la literatura como el "Problema de planificación de proyectos de software" [Alba2007]. Al respecto, se han desarrollado múltiples trabajos que resuelven este problema, de los cuales se revisaron algunos de ellos con el fin de comprender el estado del arte actual e identificar y proponer oportunidades de mejora.

En este capítulo se presenta un resumen, un cuadro comparativo y unas conclusiones sobre los trabajos revisados. Estos trabajos fueron seleccionados con base en un protocolo para la revisión sistemática de la literatura, mejor conocido como *estudio de mapeo sistemático* [Espinoza2013]. El protocolo consta de fases como la definición de las preguntas de investigación, la selección de uno o más motores de búsqueda (como Google Scholar, IEEE, etc.) y el filtrado de documentos mediante criterios de inclusión y exclusión. Para ser considerados, los trabajos tuvieron que estar enfocados en proyectos de desarrollo de software y debían utilizar alguna técnica de optimización para encontrar soluciones al problema de planificación de proyectos. Algunos de los aspectos que se describen sobre estos trabajos son: la técnica de optimización utilizada, las variables del modelo de planeación para resolver el problema, la descripción de los casos de prueba y los resultados obtenidos. Al final del capítulo, se describen algunos hallazgos identificados en la revisión bibliográfica y algunos puntos de mejora que fueron considerados en el presente trabajo.

2.1. El problema de planificación de proyectos de software

El problema de planificación de proyectos de software consiste básicamente en asignar ingenieros de software a cada una de las tareas en que se divide el proyecto, de tal forma que el costo sea mínimo y que sea entregado en el menor tiempo posible [Alba2007]. En la Figura 2.1 se muestra mediante un diagrama de clases UML, un modelo de planeación con los elementos principales para resolver el presente problema.

Figura 2.1. Elementos del problema de planificación de proyectos de software [Alba2007].



Como se puede observar, aparecen tres entidades principales: Plan, Empleado y Tarea. Por una parte, en un "Plan" se especifican y se calendarizan las tareas en las cuales se divide un proyecto, con sus respectivas dependencias y empleados asignados. Un "Empleado" representa una persona que se caracteriza por poseer habilidades, por tener un grado de dedicación máxima en el proyecto y por percibir un salario por el trabajo realizado. En el contexto de este y otros trabajos relacionados, una habilidad representa la capacidad que posee (o no) un empleado para desarrollar tareas de ciertas áreas de un proyecto. Por ejemplo: un empleado puede

tener habilidad para desarrollar tareas de Diseño de interfaces gráficas o tareas de Codificación de componentes, pero puede no tener habilidad para desarrollar tareas de Administración de Bases de Datos.

Por otra parte, una "Tarea" representa una unidad de trabajo que se caracteriza por tener asociado un esfuerzo estimado, el cual corresponde al tiempo (en meses-persona) aproximado para que dicha tarea sea completada. Además, cada tarea tiene asociado un conjunto de habilidades requeridas que deben cubrir entre todos los empleados para que esta pueda concluirse. Las habilidades de los empleados y las habilidades requeridas por una tarea, son valores numéricos. La relación entre las entidades Empleado y Tarea representa una asociación "muchos a muchos", lo cual significa que los empleados pueden participar en el desarrollo de diversas tareas (aunquesolo en una a la vez), y de la misma manera, una tarea puede ser desarrollada por varios empleados. La relación de la entidad Tarea consigo misma, representa las dependencias que puede haber entre las tareas de un proyecto, es decir, el inicio de una puede depender de la terminación de otras.

Como es común en los problemas de optimización, existen restricciones que deben ser consideradas y en este problema son las siguientes:

- Ninguna tarea debe quedar sin asignar (R1).
- Los empleados asignados a una tarea deben cubrir, entre todos, con todas las habilidades requeridas por ésta (R2).
- Un proyecto no debe presentar tiempo extra o sobrecarga de trabajo (R3).

El problema de planeación de proyectos de software puede ser planteado de la siguiente manera:

$$minimizar\ P_{duracion} = max\{t_j^{fin} \mid 1 \leq j \leq T\}$$

$$minimizar\ P_{costo} = \sum_{j=1}^{T} t_j^{costo}$$

$$sujeto\ a:$$

$$\sum_{i=1}^{E} x_{ij} > 0\ \forall j \in \{1, \ldots, T\} \qquad \text{(R1)},$$

$$t_j^{habilidadesReq} \subseteq \bigcup_{\{i|x_{ij}>0\}} e_i^{habilidades}\ \forall j \in \{1, \ldots, T\} \qquad \text{(R2)},$$

$$\sum_{i=1}^{E} e_i^{sobrecarga} = 0 \qquad \text{(R3)},$$

donde T es el número de tareas, E es el número de empleados, t_j^{fin} es el tiempo de finalización de la tarea j , t_j^{costo} es el costo de la tarea j , x_{ij} es la dedicación del empleado i a la tarea j , $t_j^{habilidadesReq}$ es el conjunto de habilidades requeridas por la tarea j , $e_i^{habilidades}$ es el conjunto de habilidades que posee el empleado i y $e_i^{sobrecarga}$ es el tiempo extra del empleado i en el proyecto.

La duración de un proyecto ($P_{duracion}$) está dada por el máximo de los tiempos de finalización de las tareas. Una de las técnicas más utilizadas para estimar la duración de un proyecto es el *método*

de ruta crítica (Critical Path Method) [Kelley1961]. La ruta crítica es la secuencia de actividades que deben ser completadas para que todo el proyecto pueda concluir de acuerdo al cronograma establecido. El costo de un proyecto (P_{costo}) está dado por la suma de los costos de todas las tareas. El costo de una tarea está en función de su esfuerzo estimado y de los salarios de los empleados
asignados a esta.

2.2. Técnicas de optimización para resolver el problema

Una solución al problema de planeación de proyectos de software puede ser representada como una matriz binaria de asignaciones entre tareas y empleados, donde el valor 1 en un elemento de la matriz, indica que una tarea fue asignada a un empleado, mientras que el valor 0 indica que no fue asignada. Al aumentar el número de tareas y de empleados en un proyecto, el número de posibles asignaciones (o escenarios) se puede incrementar exponencialmente, lo cual hace impráctico explorar todas para encontrar aquella solución con el menor costo y la menor duración. Un ejemplo con los escenarios posibles para un proyecto con 4 tareas y 2 empleados, se presenta en el Anexo A. Debido a que no es posible hallar todas las soluciones factibles posibles en un tiempo razonable, el problema de planificación de proyectos de software se clasifica computacionalmente como NP-difícil [Xiao2013] y es necesario recurrir a una clase de técnicas conocidas como *heurísticas*.

A diferencia de las técnicas exactas, las heurísticas permiten obtener buenas soluciones en un tiempo razonable para un problema particular, sin embargo, no garantizan obtener las óptimas [Reeves1996]. En éstas técnicas se realiza una exploración del espacio de búsqueda con el fin de encontrar soluciones que cumplan con ciertos criterios de calidad y con un compromiso entre sus objetivos. Entre las técnicas más utilizadas para resolver problemas, como el de la planificación de proyectos de desarrollo de software, se encuentran las heurísticas bioinspiradas, las cuales se basan en la imitación de procesos biológicos como la evolución natural. Una clase particular de este tipo de heurísticas son los *algoritmos evolutivos*.

Un algoritmo evolutivo es un método que emplea una búsqueda estocástica para encontrar un valor óptimo para un problema determinado y está basado en el proceso de selección natural en una población de individuos. En este proceso, los individuos con las mejores características tienen mayor probabilidad de supervivencia y de reproducción, donde estas características son heredadas por los individuos descendientes, que se convierten en dominantes entre la población [Engelbrecht2007]. Los componentes principales de un algoritmo evolutivo son:

- La codificación de las soluciones al problema, en forma de cromosomas.
- Una función para evaluar la aptitud o capacidad de supervivencia de los individuos.
- La inicialización de la población inicial.
- Los operadores de selección.
- Los operadores de reproducción.

Ejemplos de algoritmos evolutivos (multiobjetivo) son NSGA-II (Non dominated sorting genetic algorithm) [Deb2002], SPEA2 (Strength Pareto Evolutionary Algorithm) [Zitzler2001] y SMS-EMOA (S-Metric Selection Evolutionary Multiobjective Optimization Algorithm) [Beume2007]. Gran parte de los trabajos que a continuación se describen, utilizan algoritmos evolutivos multiobjetivo para hallar posibles soluciones al problema mencionado.

El conjunto de soluciones que se obtiene al ejecutar un algoritmo de optimización multiobjetivo, es conocido comúnmente en la literatura como *Frente de Pareto* [Ehrgott2000]. Estas soluciones tienen la característica de que ninguna es dominada por otra, en todos sus objetivos. En la Figura 2.2 se presenta un ejemplo de un Frente de Pareto para dos objetivos. Nótese que la solución que

presenta el valor más alto en el primer objetivo, presenta el valor más bajo en el segundo objetivo y viceversa. En el problema de planificación de proyectos de software, las funciones objetivo son la duración y el costo.

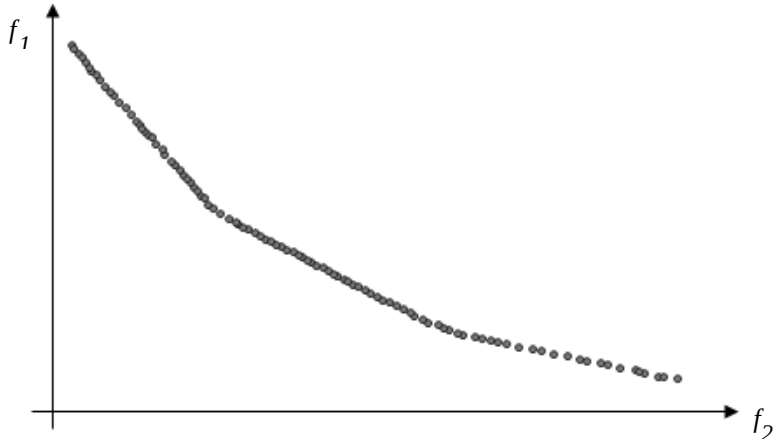


Figura 2.2. Frente de Pareto para dos funciones objetivo.

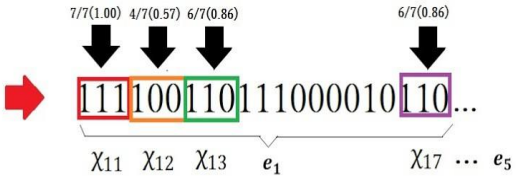
2.3. Trabajo relacionado

En esta sección se presenta un resumen de los trabajos revisados en los que se resuelve el problema de planificación de proyectos de software con diversos algoritmos de optimización. Como se verá más adelante, varios de los trabajos retoman las variables y las restricciones en [Alba2007] para resolver el problema. En estos resúmenes, se describen los aspectos que se consideraron más relevantes, como: el algoritmo de optimización empleado, algunos datos de los casos de prueba y las conclusiones de los autores sobre los resultados obtenidos.

Primero, Alba y Chicano [Alba2007] emplearon algoritmos genéticos para resolver el problema de planificación de proyectos, en el cual se minimizan el costo y la duración de un proyecto. Una solución al problema es representada como una matriz de dedicaciones $X = (x_{ij})$ de tamaño ET, (E es el número de empleados y T es el número de tareas), donde cada entrada x_{ij} representa el grado de dedicación del empleado i a la tarea j. En el contexto de un algoritmo genético, la matriz de dedicaciones X es mapeada a un cromosoma o cadena binaria de longitud $3ET$, ya que se necesitan 3 bits para representar 8 grados de dedicación a tareas (desde "000" hasta "111"). En la Figura 2.3 se presenta un ejemplo con la representación de una solución en forma de matriz de asignaciones y en forma de cromosoma.

tarea / empleado	t_1	t_2	t_3	t_4	t_5	t_6	t_7
e_1	1.00	0.57	0.86	1.00	0.00	0.29	0.86
e_2	1.00	0.86	0.86	0.00	1.00	1.00	0.14
e_3	0.43	0.86	0.00	1.00	1.00	0.57	0.71
e_4	0.00	1.00	1.00	0.14	0.29	0.86	1.00
e_5	1.00	1.00	1.00	0.57	0.14	0.86	0.86

Representación de una solución en forma de matriz



Representación de una solución en forma de cromosoma

Figura 2.3. Representación de una solución en el trabajo de Alba y Chicano [Alba2007].

Una solución x es evaluada mediante la siguiente función de aptitud: $f(x) = 1/q$, si la solución es factible y $1/(q + p)$ en caso contrario, donde q representa una suma ponderada del costo y la duración del proyecto, mientras que p es la penalización obtenida por violar alguna de las

restricciones del problema. Los parámetros del algoritmo genético para realizar las pruebas fueron: una población de 64 individuos, selección por torneo binario, reemplazo elitista y un criterio de paro de 5000 iteraciones.

Para realizar los experimentos, se generaron 5 tipos de pruebas utilizando un programa creado por los autores que permite generar proyectos aleatorios, llamado "generador de casos". El primer tipo de prueba consistió en variar el número de empleados en un proyecto con 10 tareas. Los resultados indicaron que a medida que aumenta el número de empleados, disminuye la duración del proyecto, y además, la tasa de éxito fue mayor con un número menor de empleados. La tasa de éxito es el porcentaje de ejecuciones en las cuales se obtienen soluciones factibles, es decir, soluciones en las cuales no se viola ninguna restricción.

El segundo tipo de prueba, consistió en variar el número de tareas (10, 20 y 30), mientras el salario de los empleados es fijo. En este caso se observó que la duración y el costo son mayores conforme se incrementa el número de tareas, y además, la tasa de éxito fue de 73% en promedio con 10 tareas, de 33% con 20 tareas y de 0% con 30 tareas, pues en este último caso se violan las restricciones del problema.

El tercer tipo de prueba consistió en variar las habilidades de los empleados. En este caso, se utilizó el grafo de precedencia de tareas, el número de empleados y el salario usado en los tipos de prueba 1 y 2, mientras que el número de habilidades de los empleados fue de 2 a 10. Aquí se observó que la tasa de éxito fue más baja con una cantidad menor de habilidades, pues se viola la restricción en la cual los empleados deben cubrir (entre todos), con las habilidades requeridas por una tarea.

El cuarto tipo de prueba consistió en variar todos los parámetros, esto es el número de empleados, de tareas y habilidades por empleado. El número de tareas fue 10, 20 o 30; el número de empleados fue 5, 10 o 15; y el número de habilidades por empleado fue de 4 a 7. Aquí se observó que la tasa de éxito fue de 84% a 100% con 10 tareas, mientras que con 30 tareas no se obtuvieron soluciones factibles.

Finalmente en el quinto tipo de prueba, se crearon configuraciones con 10, 20 o 30 tareas, 5, 10 o 15 empleados y 5 o 10 habilidades. Los resultados indicaron que con 10 tareas, la tasa de éxito fue de 61 a 100%; con 20 tareas de 0 a 12%; y con 30 tareas, 0% en todos los casos. La tasa de éxito fue mejor al incrementar el número de de empleados, pues se dispone de más personal y resulta más fácil hacer la asignación de tareas.

Los autores concluyeron que el costo de los proyectos se incrementó con un aumento en el número de tareas, mientras que la duración regularmente es menor con un mayor número de empleados. Cuando el número de tareas de un proyecto es igual o superior a 30, no fue posible hallar soluciones factibles.

En otro trabajo, Chicano, Luna, Nebro y Alba [Chicano2011], emplearon 5 metaheurísticas para resolver el problema de planificación de proyectos: *NSGA-II, SPEA2, PAES, MOCell y GDE3.*

La parametrización de los algoritmos fue la siguiente: una población de 100 individuos, selección por torneo binario, probabilidad de cruce (p_c) 0.9, probabilidad de mutación (p_m) 1.0/L, (donde L es la longitud de un cromosoma: número de tareas * número de empleados), y parámetros adicionales de cada algoritmo.

Para realizar pruebas, los autores crearon configuraciones variando el número de tareas (10, 20 y 30), habilidades (5 y 10) y empleados (5, 10 y 15). El indicador de calidad que se utilizó para medir el desempeño de los algoritmos fue el hipervolumen. El hipervolumen es una métrica utilizada para comparar la calidad de las soluciones arrojadas por algoritmos de optimización multiobjetivo y corresponde al tamaño del espacio objetivo que es dominado por un Frente de Pareto, es decir, el espacio en el cual las soluciones del Frente superan a otras soluciones en los objetivos de optimización [Beume2007]. En el caso de problemas de optimización con 2 objetivos (por ejemplo: el costo y la duración), el hipervolumen corresponde a un área, como se puede apreciar en la Figura 2.4; mientras que en problemas con 3 objetivos, corresponde a un volumen. Entre mayor sea el hipervolumen dominado por un Frente de Pareto, mayor será su aproximación al Frente óptimo y habrá mejor diversidad en las soluciones arrojadas por un algoritmo de optimización.

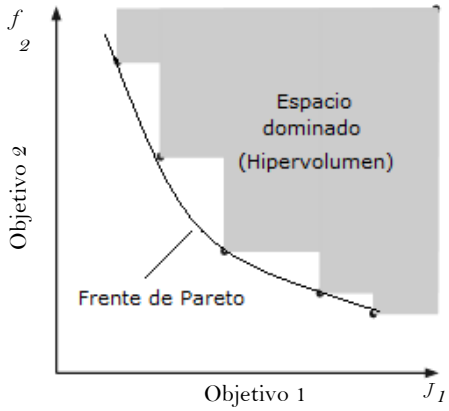


Figura 2.4. Hipervolumen dominado por un Frente de Pareto en un problema de optimización con dos objetivos.

Los resultados del trabajo de [Chicano2011] indicaron que para casos complejos del problema, el algoritmo PAES tuvo mejor desempeño, sin embargo para casos más simples, los algoritmos con mejor desempeño fueron MOCell, GDE3 y NSGA-II.

Los autores también analizaron la participación de los empleados en un proyecto. Observaron que en las soluciones con bajo costo, los algoritmos asignaron mayor carga de trabajo a los empleados que poseen un salario más bajo. Además, cuando se asigna una mayor cantidad de empleados a las tareas, la duración de las mismas disminuye, así como la duración del proyecto, sin embargo, el costo total se incrementa debido a los costos generados por los empleados agregados. Los autores concluyeron que en la mayoría de los casos del problema, se presentó una correlación inversa entre la duración y el costo de un proyecto.

En el trabajo de Luna, González y Chicano [Luna2014], se analizan las capacidades de escalabilidad de 8 algoritmos metaheurísticos multiobjetivo: NSGA-II, SPEA2, PAES, DE-PT, MO-FA, MOABC, MOCell y GDE3, dicho de otra manera, se hace un análisis de las soluciones obtenidas para hallar correlaciones entre sus características y la región en el espacio objetivo en el cual se encuentran esas soluciones.

Para realizar los experimentos, se diseñaron 36 casos de prueba aleatorios con un número variable de empleados y tareas. El número de empleados varía entre 8 y 256 (en potencias de 2), mientras que el número de tareas varía entre 16 y 512 (también en potencias de 2). Para medir la calidad de las soluciones se utiliza el indicador de hipervolumen.

Los resultados indicaron que el algoritmo PAES alcanzó los valores más altos de hipervolumen en 34 de los 36 casos de prueba, con respecto al resto de los algoritmos. También se observó que cuando la participación de los empleados en las tareas es mayor, regularmente se reduce la duración del proyecto, mientras el costo aumenta, sin embargo, hubo casos en los cuales no fue así, lo cual significa que tener más gente trabajando en las tareas, no siempre garantiza la reducción de la duración (y el aumento del costo). Los autores concluyeron que PAES ha demostrado no ser solo uno de los algoritmos con mayor escalabilidad, sino que también, uno de los que mejor calidad ofrece en sus soluciones con respecto a la métrica de hipervolumen.

Por su parte, Jin y Yao [Jin2014], emplean un algoritmo de optimización llamado *PBIL (population based incremental learning)*. Este algoritmo genera y modifica un vector de probabilidades para soluciones candidatas. Los pasos principales de este algoritmo son: la inicialización, el muestreo, la evaluación, la actualización y la mutación de ese vector de probabilidades. El desempeño de las soluciones candidatas es evaluado con la función de aptitud utilizada en el trabajo de Alba y Chicano [Alba2007].

Al problema se le agrega un parámetro δ, el cual representa un factor de eficiencia del equipo de desarrollo, que en la práctica puede representar las experiencias, los procesos, la relación de trabajo, aspectos emocionales y cognitivos del equipo de trabajo; y que tiene un impacto en la duración y el costo del proyecto.

Los parámetros del algoritmo para realizar las pruebas fueron los siguientes: un tamaño de población de 64 individuos, un criterio de selección elitista, 300 iteraciones como máximo y 100 ejecuciones por prueba. Se definieron 5 configuraciones para analizar el impacto del factor eficiencia del equipo (δ) en las soluciones. El valor de δ se interpreta como sigue: δ=1 indica que no se considera el factor, δ=0.5 indica que la duración de las tareas decrece en un 50% (porque la relación entre los miembros del equipo es baja) y δ=1.5 indica que la duración de las tareas se incrementa en un 50% (que significa que la relación de trabajo entre los miembros es adecuada y son más productivos). De las 5 configuraciones, una presenta δ=1, 2 presentan δ=0.5 y las 2 restantes, δ=1.5. Los resultados indicaron que en la configuración donde δ=1, la duración fue de 16.44 meses y un costo de 785,770. En las 2 configuraciones con δ=0.5 la duración del proyecto se incrementó en un 18% y 45%, respectivamente; mientras que el costo se incrementó en un 7.2% y 10%, respectivamente. En tanto, en las configuraciones con δ=1.5 la duración se redujo en un 20% y 33%, respectivamente y el costo se redujo en un 24% y 39%, respectivamente.

Los autores concluyeron que con un número suficiente de iteraciones, PBIL es capaz de encontrar vectores de probabilidad que siempre producen soluciones factibles y deseables con los distintos casos del problema. Además, los resultados demuestran que el factor de eficiencia de un equipo puede influir principalmente en la duración efectiva de un proyecto.

Xiao, Ao y Tang [Xiao2013] emplearon un algoritmo de optimización basado en *colonia de hormigas* (ACO por sus siglas en inglés) para afrontar el problema. Este algoritmo está inspirado en el comportamiento de una colonia de hormigas, las cuales buscan el mejor camino a través de un grafo para llegar a una fuente de comida. Durante su regreso, las hormigas dejan un rastro de feromonas que sirven como guía para otras hormigas que realicen su recorrido, sin embargo, con el paso del tiempo, el rastro de feromonas se evapora y se tienen que buscar otros caminos.

En este caso, ACO se aplicó para encontrar una forma eficiente de determinar el grado de dedicación de los empleados a una tarea. Para esto, cada tarea es dividida en un grafo en forma de matriz (ver Figura 2.5), donde las columnas representan los empleados y las filas los grados de dedicación en el intervalo [0,1] con subintervalos iguales de 0.25. Las hormigas realizan el recorrido

a través de los vértices del grafo y calculan las contribuciones de dedicación de cada empleado a esa tarea.

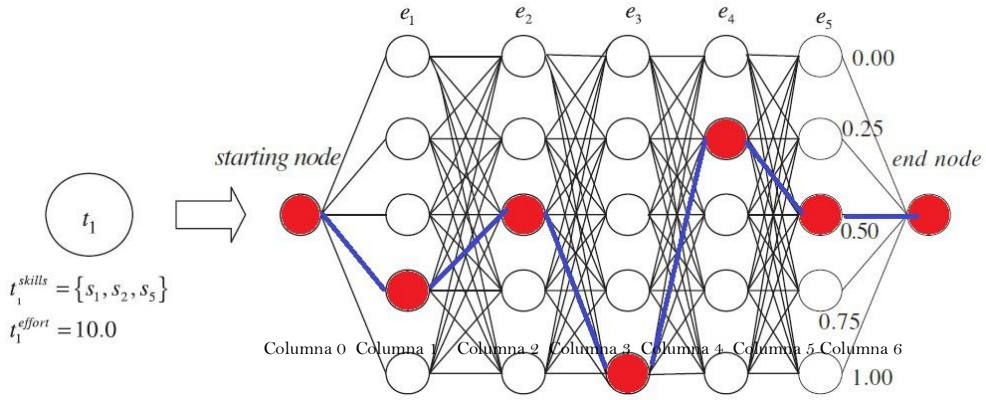


Figura 2.5. División de una tarea en un grafo en [Xiao2013].

Una vez obtenida la matriz de dedicaciones de los empleados para todas las tareas, se calcula la duración y el costo del proyecto mediante un conjunto de ecuaciones descritas en el artículo y se evalúa la calidad de las soluciones mediante la siguiente función de aptitud:
$f(x) = (W_{cost} x P_{cost} + W_{dur} + P_{dur})^{-1}$, donde W_{cost} y W_{dur} corresponden a un valor ponderado para el costo y la duración de proyecto, respectivamente.

Para realizar los experimentos, se crearon casos de prueba con configuraciones de 5, 10 o 15 empleados y 10, 20 o 30 tareas, donde cada caso de prueba fue ejecutado 100 veces. Se consideraron 8 grados de dedicación (0, 1/7, 2/7,...1) de los empleados a las tareas. Los resultados indicaron que conforme se incrementa el número de empleados, se vuelve más complejo encontrar soluciones factibles al problema, principalmente porque se viola la restricción de tiempo extra en el proyecto. Al igual que en otros trabajos, cuando el número de tareas es igual o superior a 30, la tasa de éxito fue de 0%, es decir, no se obtuvieron soluciones factibles al problema.

Por otra parte, al incrementar el número de empleados y dejar fijo el costo de los empleados, el costo del proyecto se mantiene igual en todas las configuraciones y la duración disminuye. El desempeño de ACO fue comparado con el desempeño del algoritmo genético (AG) utilizado en [Alba2007], con los mismos casos de prueba. Los resultados indicaron que la tasa de éxito de ACO fue mayor que la que presentó el AG en la mayoría de las configuraciones. Además, con ACO tanto la duración como el costo, presentaron valores ligeramente menores en las distintas configuraciones. Los autores concluyeron que ACO es una propuesta prometedora para encontrar soluciones al problema de planificación de proyectos de software.

Gonsalves e Itoh, [Gonsalves2010], utilizaron una heurística bioinspirada llamada *optimización multiobjetivo por enjambre de partículas* (MOPSO por sus siglas en inglés). Este algoritmo imita el proceso de intercambio de información en una bandada que está en busca de comida. Los individuos son llamados partículas y la población es llamada enjambre.Las partículas se mueven por un espacio de búsqueda, y cada una tiene asociada una velocidad y una posición. El movimiento de cada partícula se ve influido por su mejor posición local encontrada hasta el momento, así como por las mejores posiciones globales encontradas por otras partículas a medida que recorren el espacio de búsqueda. El objetivo es hacer que el enjambre converja rápidamente hacia las mejores soluciones. MOPSO en este contexto, es utilizado para encontrar el espacio de soluciones que minimice el costo y el tiempo de desarrollo de un proyecto.

En este trabajo, los empleados poseen un nivel de habilidad cuantificado para cada tipo de tarea. Por ejemplo, un recurso puede tener un nivel de habilidad 3 para bases de datos, un nivel 3 para Java, un nivel 2 para diseño orientado a objetos y un nivel 4 para pruebas. Las tareas son agrupadas en disciplinas del ciclo de vida, como requerimientos, diseño y codificación.

Desafortunadamente, los datos de las pruebas realizadas en este trabajo no son reportados, únicamente se desglosan los resultados obtenidos con 6 empleados. Para cada tarea, se reporta la asignación de empleados con base en el nivel de sus habilidades, la duración promedio en días-persona y el costo total. Los resultados indican que con MOPSO se obtuvo una asignación de personal a las tareas, cercana a la óptima. Se observó que cualquier asignación que produzca desajuste en las habilidades de los empleados, incrementa el tiempo de procesamiento estimado para las tareas. Se concluye que con MOPSO, es posible obtener un frente de Pareto bien definido, el cual representa un intercambio entre el tiempo de desarrollo y el costo de un proyecto.

Dupuy, Stark y Salto [Dupuy2013], implementan un algoritmo genético tradicional con una codificación binaria para representar las soluciones. Los cromosomas son de longitud $3ET$ (donde E es el número de empleados y T es el número de tareas) y representan las dedicaciones de los empleados a las tareas. Al manejar 8 posibles valores para esas dedicaciones, se requieren 3 bits para su representación (000, 001, …,111). Para evaluar la aptitud de los cromosomas, se utilizó la expresión del trabajo de Alba y Chicano [Alba2007], la cual consiste en una función inversa de la suma ponderada de la duración y el costo, pues ambos objetivos tienen asociado un peso (w_{dur} y w_{cost}, respectivamente) según su importancia relativa. De esta manera, el objetivo con mayor prioridad tiene un peso más alto. En los experimentos, las ponderaciones fueron $w_{dur} = 10^{-6}$ y $w_{cost} = 10^{-1}$, respectivamente. Las soluciones que son no factibles, sufren una penalización.

En este algoritmo se utilizaron dos operadores de cruce: *1X* y *2DMX*. El operador 1X, es utilizado para recombinar (cruzar) un vector en un determinado punto, mientras que el operador 2DMX es utilizado para recombinar matrices. En este último se selecciona la misma fila y la misma columna en las matrices padres en forma aleatoria, y se intercambian los elementos en el cuadrante superior izquierdo y en el cuadrante inferior derecho de ambos individuos. Se utilizan 4 valores para la probabilidad de cruce (pc): 0.3, 0.5, 0.75 y 0.9, por lo que con 2 tipos de cruce y 4 probabilidades se pueden generar 8 variantes del algoritmo genético.

La parametrización del algoritmo fue la siguiente: el tamaño de la población es de 64 individuos, el criterio de selección es por torneo binario, el operador de mutación es bit-flip con probabilidad de 0.005 y el total de generaciones fue de 20,000. Para realizar los experimentos se crearon 36 casos del problema utilizando 5, 10 o 15 empleados y 10, 20 o 30 tareas, además de un número variable de habilidades de empleados. Por cada configuración, se realizaron 30 ejecuciones.

Los resultados en este trabajo, indicaron que con una probabilidad de cruce relativamente alta (por ejemplo: 0.9), el algoritmo genera mejores soluciones con ambos operadores (1X y 2DMX), respecto a los demás valores de probabilidad de cruce. Con 20 tareas, el algoritmo obtuvo una tasa de éxito mayor al utilizar el operador 1X, pero con 30 tareas no se obtuvieron soluciones factibles con ambos operadores. Los autores concluyeron que el algoritmo propuesto con una variante en el operador de cruce, permite generar asignaciones entre empleados y tareas de forma más eficiente, y por consiguiente, mejores soluciones con respecto a otros algoritmos genéticos.

En el trabajo de García y Gómez [García2014], se emplea un algoritmo genético multiobjetivo (MO-GA) para resolver el problema de planificación de proyectos. Para representar los cromosomas (soluciones), se utiliza una codificación de tipo entero. Los cromosomas tienen una longitud de tamaño $m \times n$, donde los primeros n valores corresponden al tiempo de dedicación del primer

empleado a cada una de las tareas, los siguientes *n* valores corresponden al tiempo de dedicación del segundo empleado, y así sucesivamente, donde los últimos valores corresponden a las dedicaciones del empleado *m*. En la Figura 2.6 se presenta un ejemplo de esta representación con horas de dedicación de los empleados a 5 tareas.

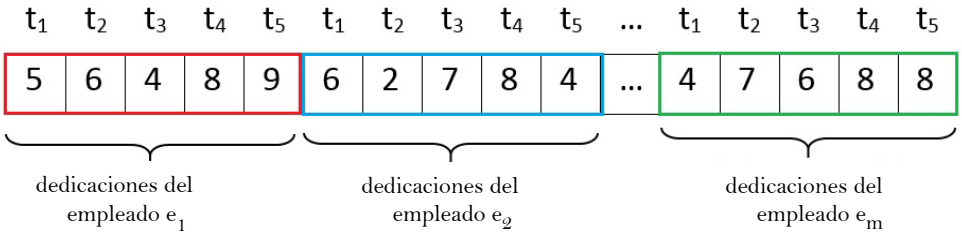


Figura 2.6. Representación de una solución al problema de planificación de proyectos de software en el trabajo de [García2014].

Además del costo y la duración, en este trabajo se intenta minimizar la sobrecarga o tiempo extra en el proyecto. Un proyecto presenta tiempo extra cuando al menos uno de los empleados tiene asignadas tareas que coinciden en algún lapso. Con respecto a la parametrización, MO-GA emplea selección por torneo binario tradicional, en el cual se seleccionan dos individuos aleatoriamente que sirven como padres para la operación de cruce. Los operadores de cruce utilizados para la obtención de un individuo descendiente, fueron la cruza plana y la cruza simple. En la cruza plana, se toman aleatoriamente tiempos de dedicación de empleados a tareas de los padres y se copian en el descendiente, mientras que en la cruza simple, se eligen empleados aleatoriamente y los tiempos de dedicación a tareas del primer padre se copian en el descendiente y los tiempos de dedicación de los empleados restantes se copian del segundo padre [García2014]. El descendiente es sometido a una operación de mutación. Posteriormente se aplica un operador de reparación para las soluciones no factibles y finalmente, se seleccionan los individuos que pasarán a la siguiente generación del algoritmo, sobreviviendo aquellos individuos que tengan mayor aptitud después de ser evaluados.

En este trabajo se hicieron algunas adecuaciones al modelo de planeación de Alba y Chicano [Alba2007] para resolver el problema de planificación de proyectos. Se asume que el ciclo de vida de un proyecto consta de tareas de análisis de requerimientos, diseño, codificación, pruebas, etc.; y se consideran 4 posibles niveles de habilidad para los empleados: nivel *principiante*, nivel *junior*, nivel *senior* y nivel *experto*, donde el salario recibido por cada empleado es de acuerdo a su nivel de habilidad. Las tareas también tienen asociado un nivel de habilidad como los antes mencionados. Utilizando las ecuaciones descritas en el artículo, se calcula la duración, el costo y el tiempo extra del proyecto para evaluar y seleccionar soluciones candidatas. En la Figura 2.7 se presenta un modelo en UML que contiene las características de las tareas y los empleados para resolver el problema.

Para realizar los experimentos correspondientes, se crearon 36 casos del problema y los resultados fueron comparados con los del trabajo de Alba y Chicano [Alba2007]. En configuraciones con 10 tareas, el enfoque de Alb y Chicano presenta una tasa de éxito más baja que en [García2014], sin embargo, en configuraciones con 20 tareas, el enfoque de Alba y Chicano presenta una tasa de éxito ligeramente mayor. Con 30 tareas, en ambos enfoques no es posible encontrar soluciones factibles. Los autores concluyeron que el algoritmo propuesto (MO-GA), es eficiente para encontrar soluciones factibles en gran parte de las configuraciones del problema de planificación de proyectos de software, con respecto a un algoritmo genético tradicional.

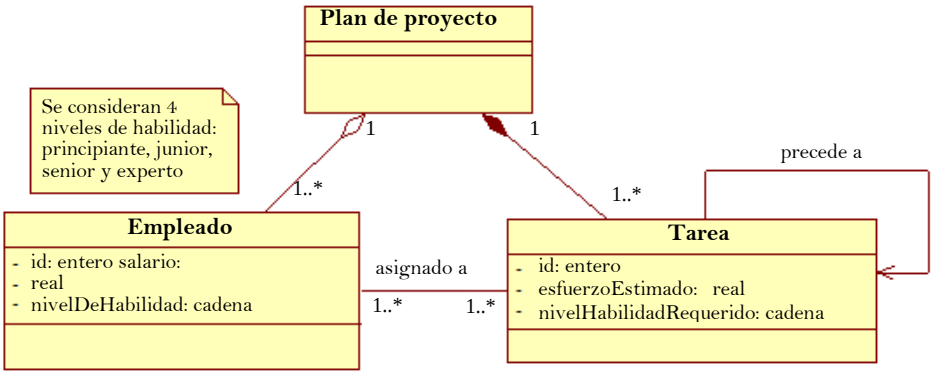
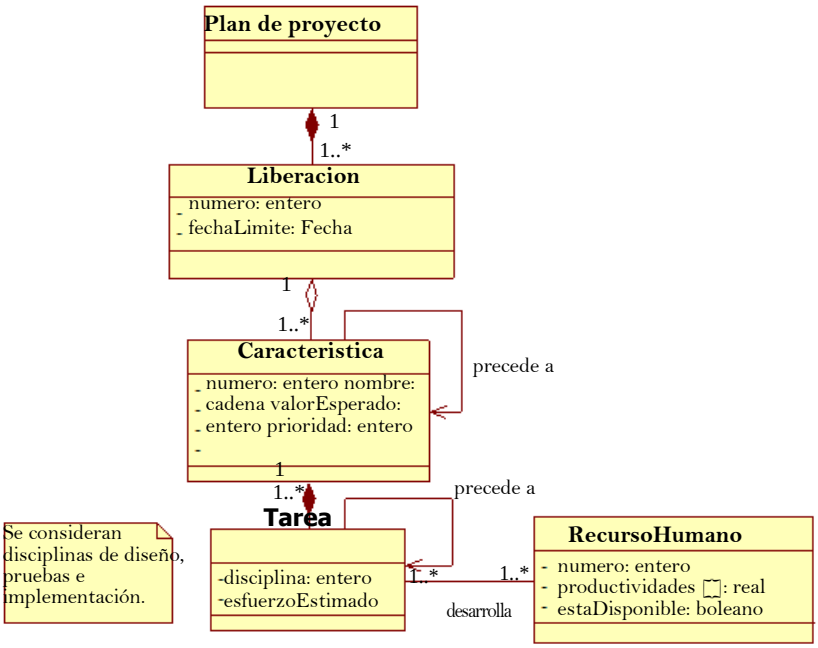


Figura 2.7. Modelo de planeación para resolver el problema de planificación de proyectos de software en [García2014].

Otro de los trabajos que presenta algunas diferencias con respecto al modelo de planeación de [Alba2007], es el de Ruhe [Ruhe2009], el cual está enfocado en proyectos que siguen un desarrollo incremental. Cada proyecto se descompone recursivamente en liberaciones, características (requerimientos) y tareas. Para cada liberación existe una fecha límite de entrega y para desarrollar las características se asume que hay un pool de recursos disponible. Las características entregadas en una liberación, generan un valor de negocio para el cliente y el objetivo en este problema es precisamente maximizar el valor ganado de todas las características que alcancen a desarrollarse antes de la fecha estipulada.

Para resolver el problema, se utilizó un método de optimización que consta de dos fases. En la primera fase, se utiliza una técnica de programación lineal entera para determinar las características que deben ser entregadas en cada liberación. En la segunda fase, se aplica un algoritmo genético para asignar eficientemente las características de cada liberación a los recursos disponibles. Los elementos del modelo usado para resolver este problema se presentan en el diagrama UML de la Figura 2.8.

Figura 2.8. Modelo de planeación para resolver el problema de optimización en [Ruhe2009].



Uno de los aspectos más importantes que se consideran en este modelo, es la productividad que presentan los ingenieros en distintas disciplinas del ciclo de desarrollo de software como el diseño, la implementación y las pruebas. Se asume que en la práctica estos valores pueden ser juzgados y asignados por el administrador del proyecto.

Los parámetros del algoritmo genético incluyen una población de 100 individuos (cromosomas), un máximo de 500 iteraciones (generaciones), una probabilidad de mutación de 1% y un criterio de parada cuando en 100 generaciones consecutivas no hay mejora en la aptitud de las soluciones, o bien, se ha alcanzado el número máximo de generaciones.

Para realizar los experimentos, se generaron aleatoriamente 600 casos de prueba (proyectos) divididos en 20 grupos de 30. Cada proyecto consta de 20 características, las cuales a su vez, constan de 3 tareas y se dispone de 6 empleados para su desarrollo. Se tienen 2 liberaciones planeadas por proyecto, cada una con duración estimada de 12 semanas. Los resultados indicaron que todos los empleados, salvo uno, se mantienen ocupados durante las 24 semanas del proyecto, siendo muy eficiente la distribución, y además, el 82% de las tareas en promedio, fueron asignadas de forma ideal, es decir, a empleados con mayor productividad.

Los autores concluyeron que la aplicación del método de dos fases permite generar un calendario de proyecto óptimo, respetando la fecha límite de las liberaciones, sin embargo, indican que al problema le falta considerar factores organizacionales y factores humanos, y también el nivel de complejidad de las tareas.

Mientras tanto, Duggan, Byrne y Lyons [Duggan2004] se enfocan en resolver el problema de la asignación de tareas durante la etapa de construcción en el ciclo de vida del software. En este caso se intenta minimizar el tiempo de desarrollo y la cantidad de defectos que se espera inyecten los ingenieros, según su nivel de habilidad. En el modelo para resolver el problema, se consideran características de los ingenieros como el nivel de productividad (medido en unidades de complejidad UOCs), el promedio esperado de defectos inyectados y cinco niveles de habilidad escalados de novato a experto. Por su parte, las tareas tienen asociada una complejidad y un tipo. En la Figura 2.9 se presenta un diagrama en UML con las características de ingenieros y tareas para resolver el problema de asignación de tareas.

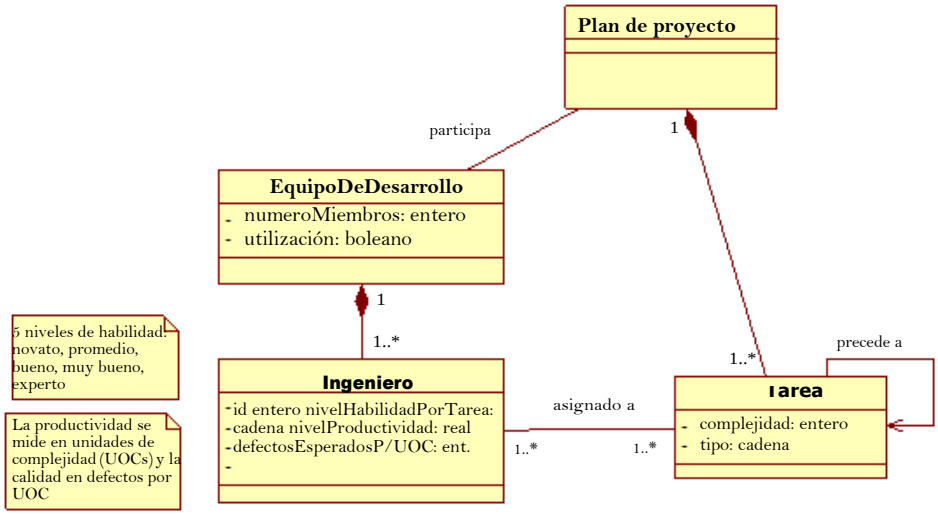


Figura 2.9. Modelo de planeación para resolver el problema de asignación de tareas en [Duggan2004].

Para resolver este problema, los autores emplearon el algoritmo genético NSGA con un criterio elitista, para garantizar que las buenas soluciones no se pierdan en generaciones posteriores. Para realizar los experimentos se utilizó un equipo de 6 ingenieros variando su nivel de habilidad en tareas relacionadas con redes, base de datos, análisis, interfaz gráfica, entre otras.

Se obtuvieron 13 soluciones tiempo-defecto. De estas soluciones, 2 son las extremas, es decir, una presenta la mayor cantidad de defectos estimados y la menor duración, y la segunda presenta la menor cantidad de defectos y la mayor duración. En este caso, las soluciones extremas fueron las siguientes: una solución con 1,500 defectos y un tiempo de desarrollo estimado de 74 días y la otra solución con 820 defectos y un tiempo de desarrollo estimado de 110 días. La asignación de trabajo en las soluciones no fue equitativa, pues tan solo en la solución con menor tiempo de desarrollo, la carga de trabajo fue mayor en los ingenieros que presentan mayor nivel de habilidad en promedio. Los autores concluyeron que el algoritmo utilizado (NSGA), permite optimizar el tiempo de calendarización y la calidad de un proyecto, basándose en una métrica de productividad (unidades de complejidad por día) según el nivel de habilidad de los ingenieros, sin embargo, afirman que aún falta acomodar diversos aspectos como la disponibilidad y aspectos relacionados con los procesos organizacionales.

Finalmente; Chicano, Cervantes, Luna y Recio [Chicano2012] presentan un estudio experimental en el cual se compara el desempeño de 4 algoritmos evolutivos para resolver el problema de planificación de proyectos (reformulado). El objetivo en este caso, es minimizar el costo y el *makespan* del proyecto, donde este último es la cantidad de tiempo que se requiere para completar todas las tareas del proyecto.

En este trabajo se modificaron algunas características con respecto al modelo de [Alba2007]. Las principales diferencias radican en que se considera la productividad de los empleados para realizar diferentes tareas y se eliminan habilidades cuantificadas para tareas como para empleados, tal como se muestra en el diagrama de clases de la Figura 2.10.

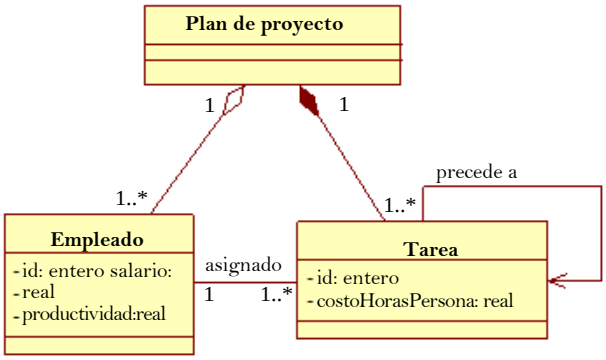


Figura 2.10. Modelo para resolver el problema de calendarización de proyectos en [Chicano2012].

Una solución al problema en este trabajo es representada como $= (d, r, q)$, donde d es un vector de números reales que representa la dedicación de cada empleado a cada tarea, r es un vector de enteros que representa el retardo que cada tarea puede tener al comenzar su desarrollo, y q es una matriz de prioridades que especifica la asignación entre tareas y empleados.

Los algoritmos utilizados fueron NSGA-II, SPEA2, PAES y MOCell con los siguientes parámetros: un tamaño de población de 100 individuos en NSGA-II, SPEA2 y MoCell, probabilidades de cruce y mutación 0.9 y 1/L, respectivamente, donde L es la longitud de un cromosoma; y un criterio de paro

cuando se ha alcanzado un millón de iteraciones. Para evaluar el desempeño de los algoritmos, se utilizó el indicador de hipervolumen y otro indicador llamado "attainment surfaces".

Primero, los resultados fueron analizados con base en la robustez que presentan las soluciones. Una solución robusta es aquella en la cual, tanto el costo como el makespan, no son sensibles a cambios cuando se modifica el costo de tareas individuales debido a imprecisiones en estimaciones iniciales. Los resultados indicaron que las soluciones con alto costo presentaron un bajo makespan y menor robustez, mientras que en soluciones con bajo costo la robustez fue nula, ya sea en el makespan o en el costo. Por otra parte, se observó una correlación negativa entre el makespan y el número de tareas paralelas, así como entre el makespan y el número de empleados. También se observó que tanto NSGA-II y MOCell obtuvieron mejor desempeño con respecto a la métrica de hipervolumen, mientras que PAES resultó ser el menos eficiente.

Los autores concluyeron que los algoritmos MOCell y NSGA-II fueron los más apropiados para la resolución del problema de planificación de proyectos de software reformulado, y aunque es aplicado sobre casos de prueba más reales, afirman que es deseable aplicarlo a un estudio empírico en el cual se utilicen datos proporcionados por empresas de software. Se utilizó un enfoque de robustez debido a que existen estudios que señalan que varios proyectos de software gastan en promedio 30-40% más esfuerzo del que se estima.

2.4. Comparación de trabajos

Con el fin de identificar el aporte y las limitaciones de los trabajos revisados, se sintetiza un cuadro comparativo (Tabla 2.1) sobre la revisión bibliográfica bajo las siguientes dimensiones:

- Algoritmo(s) de optimización utilizado(s)
- Variables del modelo de planeación para resolver el problema
- Objetivos a optimizar
- Datos utilizados en casos de prueba
- Resultados

Artículo	Algoritmo(s) utilizados	Variables del modelo de planeación	Objetivos a optimizar	Datos utilizados en casos de prueba	Resultados
[Alba2007]	Algoritmo Genético (AG)	- Habilidades, salario y grado de dedicación máxima de los empleados. - Habilidades requeridas y esfuerzo estimado de las tareas.	La duración y el costo del proyecto.	48 casos de prueba con: - 10, 20 y 30 tareas; 5, 10 y 15 empleados; 2, 4, 6, 8 y 10 habilidades por empleado. Parámetros del AG: - Tamaño de población: 64 individuos, selección por torneo binario, recombinación: 2-DSPX, mutación: bit-flip (1/L), criterio de paro: 5000 iteraciones	- Con mayor número de empleados, es menor la duración del proyecto y menor es la tasa de éxito. - Con 30 tareas o más, no fue posible encontrar soluciones factibles. - Cuando el número de habilidades de los empleados es bajo, la tasa de éxito es baja.
[Chicano 2011]	Algoritmos evolutivos multiobjetivo: - NSGA-II - SPEA2 - PAES - MOCell	- Habilidades, salario y grado de dedicación máxima de los empleados. - Habilidades requeridas y esfuerzo estimado para las tareas.	La duración y el costo del proyecto.	- Casos de prueba con 5, 10 o 15 empleados, 10, 20 o 30 tareas y 5 o 10 habilidades. Parámetros de los algoritmos: - Población de 100 individuos, mutación polinomial, pm= 1.0/L, donde L es la longitud del individuo, pc = 0.9, selección por torneo binario, recombinación binaria	- PAES ofrece mejores soluciones para casos complejos del problema, mientras que para casos menos complejos los algoritmos más apropiados fueron GDE3 y NSGA-II. - Se presentó una correlación inversa entre el costo y la duración obtenida en las soluciones.
[Luna2013]	Algoritmos multiobjetivo: NSGA-II, SPEA2, PAES, DE-PT, MO-FA, MOABC, MOCell GDE3.	- Habilidades, salario y grado de dedicación máxima de los empleados. - Habilidades requeridas y esfuerzo estimado para las tareas.	La duración y el costo del proyecto.	- 36 casos de prueba - Escala de empleados: 8-256 (potencias de 2). - Escala de tareas: 16-512. - 6-7 habilidades por empleado. - 30 ejecuciones para cada caso de prueba con los 8 algoritmos. - Cada algoritmo posee su propia parametrización, la cual se especifica en el artículo.	- El algoritmo PAES obtuvo mejores soluciones con respecto a la métrica de Hipervolumen en 34 de 36 casos de prueba. -Al aumentar la participación de los empleados regularmente se reduce la duración y se incrementa el costo, sin embargo, hubo casos en los cuales no se presentó esta situación.
[Jin2014]	PBIL (Population-based incremental learning)	- Habilidades, salario y grado de dedicación máxima de los empleados. - Habilidades requeridas y esfuerzo estimado para las tareas. -Factor de eficiencia del equipo de trabajo.	La duración y el costo del proyecto.	- Casos de prueba con factores de eficiencia de 0.5, 1 y 1.5 - Las configuraciones constan de 10 tareas, 5 empleados y 10 habilidades. Parámetros del algoritmo: - Población de 64 individuos, selección elitista, 300 iteraciones como máximo y recombinación N/A.	- En las configuraciones donde el factor de eficiencia es 0.5 se obtuvo una duración y costo mayor que en las configuraciones donde el factor es 1, mientras que en aquellas donde se utilizó un factor de 1.5, se obtuvo un costo y una duración menor.
[Xiao2012]	Ant Colony Optimization (ACO)	- Habilidades, salario y grado de dedicación máxima de los empleados. - Habilidades requeridas y esfuerzo estimado para las tareas.	La duración y el costo del proyecto.	- Configuraciones con 5, 10 y 15 empleados y 10, 20 y 30 tareas. - 8 posibles niveles de dedicación a tareas. - El desempeño de ACO fue comparado con el desempeño del algoritmo genético (AG) propuesto en [Alba2007].	Al incrementar el número de empleados, es más difícil hallar soluciones factibles. Con 30 o más tareas en un proyecto, la tasa de éxito fue 0%. Los resultados indicaron que la tasa de éxito de ACO es mayor que la que presenta el AG de [Alba2007, en la mayoría de las configuraciones.
[Gonsalves 2010]	Multiobjective Particle Swarm Optimization (MPSO)	- Habilidades, salario y grado de dedicación máxima de los empleados. - Habilidades requeridas y esfuerzo estimado para las tareas.	La duración y el costo del proyecto.	- Configuraciones con 6 empleados y tareas de requerimientos, diseño, codificación y pruebas. - No se reportan los parámetros del algoritmo empleado.	Se obtuvo un frente Pareto bien definido que representa un intercambio entre el tiempo y el costo de desarrollo.

Artículo	Algoritmo(s) utilizados	Variables del modelo de planeación	Objetivos a optimizar	Datos utilizados en casos de prueba	Resultados
[Dupuy2013]	Algoritmo genético con operadores de cruce 1X y 2DMX	- Habilidades, salario y grado de dedicación máxima de los empleados. - Habilidades requeridas y esfuerzo estimado para las tareas.	La duración y el costo del proyecto.	- 36 casos de prueba con combinaciones de 5, 10 y 15 empleados; 10, 20 y 30 tareas y 4 a 7 habilidades de empleados. Parámetros del Algoritmo Genético: - Codificación binaria. - 2 operadores de cruce: 1X y 2DMX. - 4 probabilidades de cruce: 0.3, 0.5, 0.75 y 0.9. - Tamaño de población: 64 individuos. - Mutación: bit-flip con probabilidad de 0.005. - 20,000 generaciones como máximo.	- Con el operador de cruce 1X se obtuvieron mejores soluciones que con el operador 2DMX. - Con probabilidades de cruce altas, como 0.9, se obtuvo una mejor calidad en los resultados. - Con un número mayor o igual a 30 tareas, no es posible obtener soluciones factibles, utilizando ambos operadores de cruce.
[García2014]	Algoritmo Genético Multiobjetivo (MO-GA)	- Nivel de habilidad y el salario de los empleados. - Esfuerzo estimado y nivel de habilidad requerido de las tareas.	La duración, el costo y el tiempo extra del proyecto.	- 45 casos de prueba con combinaciones de 5, 10 o 15 empleados y 10, 20 o 30 tareas. - Cada empleado puede realizar 1,2,4 u 8 tareas por día. Parámetros del AG - Codificación entera para soluciones. - Método de selección por torneo binario. - Cruza plana y cruza simple. - Mutación gradiente y mutación aleatoria. - Probabilidad de mutación de 10%. - Operador de reparación para soluciones no factibles.	- Con 30 tareas no es posible encontrar soluciones factibles. - Cuando los empleados realizan 2,4 u 8 tareas por día, el AG encuentra soluciones factibles para todas las configuraciones, excepto una. - En configuraciones con 10 tareas, el enfoque de [Alba2007] presenta una tasa de éxito más baja que en [García2014], sin embargo, en configuraciones con 20 tareas, el enfoque de [Alba2007] presenta una tasa de éxito ligeramente mayor.
[Ruhe2009]	Programación lineal entera y algoritmo genético (AG)	-Productividad por tipo de tarea de los empleados - Tipo y esfuerzo estimado de las tareas.	El valor generado por las características que se alcancen a desarrollar en una fecha establecida.	- Casos de prueba con 20 características, 2 liberaciones de 12 semanas-persona cada una; y 6 desarrolladores. Parámetros del AG: - Población de 100 cromosomas. - Máximo 500 iteraciones. - Probabilidad de mutación de 1%.	- Todos los empleados, salvo uno, se mantienen ocupados en las 24 semanas del proyecto, evitando tiempos de inactividad. - El 82% de las tareas fueron asignadas a empleados con mayor productividad.
[Duggan2004]	Nondominated Sorting Genetic Algorithm (NSGA)	-Productividad de los ingenieros en UOCs. - Nivel de habilidad y cantidad de defectos por UOCs de los ingenieros.	Minimizar la duración y maximizar la calidad del proyecto	Configuraciones con 6 ingenieros y 7 tipos de tarea. Parámetros del algoritmo: - Un criterio de selección elitista. El resto de los parámetros no se reporta.	- Se obtuvieron en promedio 13 soluciones (tiempo vs calidad). - La asignación de trabajo en las soluciones no es equitativa, pues tan solo en la solución con menor tiempo de desarrollo, la carga de trabajo fue mayor en los ingenieros que presentan mayor
[Chicano2012]	- NSGA-II - SPEA2 - PAES - MOCell	-El salario y la productividad de los empleados. -El costo de las tareas (en horas-persona).	El costo y el makespan del proyecto.	- Criterio de parada un millón de evaluaciones de la función objetivo. - Métricas de hipervolumen y "attainment surfaces" para evaluar el desempeño de los algoritmos.	- Mejor desempeño de NSGA-II y MOCell y el más bajo desempeño en PAES, en ambas métricas.

Tabla 2.1. Cuadro comparativo de trabajos que resuelven el problema de planificación de proyectos.

2.5. Análisis de la revisión bibliográfica

Los trabajos revisados presentan similitudes y particularidades con respecto a las dimensiones del cuadro comparativo 2.1, y a continuación se describen.

Primero, se identificó que los trabajos [Chicano2011], [Luna2014], [Jin2014], [Xiao2013], [Gonsalves2010] y [Dupuy2013], se basan en el modelo de planeación de [Alba2007], lo cual significa que se retoman las variables y restricciones para resolver el problema de planificación de proyectos de software. Las variables principales corresponden a características que presentan los empleados y las tareas de un proyecto. Por una parte, los empleados poseen un salario y un conjunto de habilidades (cuantificadas); mientras que las tareas poseen un esfuerzo estimado en meses-persona y un conjunto de habilidades requeridas para su desarrollo. En otros trabajos, se presentan algunas diferencias; por ejemplo en [García2014], se eliminan las habilidades cuantificadas de los empleados y tareas, y a cambio, se agrega un nivel de habilidad para empleados y tareas; mientras que en [Ruhe2009], [Duggan2004] y [Chicano2012], se toma en cuenta la productividad de los empleados en el desarrollo de tareas.

En la mayoría de los trabajos, los objetivos de minimización son la duración y el costo de un proyecto, sin embargo, hubo ciertas particularidades. En [García2014], además se intenta minimizar el tiempo extra en el proyecto; en [Ruge2009] el interés radica en maximizar el valor ganado por las características que alcancen a desarrollarse antes de una fecha de entrega definida; mientras que en [Duggan2004], el objetivo es minimizar la duración y la cantidad de defectos que se espera inyecten los empleados.

Por otra parte, en cada trabajo se recurrió a diversos algoritmos de optimización para resolver el problema de planificación de proyecto. La mayor parte, corresponde a algoritmos metaheurísticos multiobjetivo como NSGA, PAES, SPEA, MoCell. Otras heurísticas bioinspiradas que se utilizaron fueron "Optimización basada en colonia de hormigas" (en [Xiao2013]), "optimización por enjambre de partículas" (en [Gonsalves2010]); mientras que en [Alba2007], [Dupuy2013] y en [García2014], se utilizaron algoritmos genéticos particulares.

En los trabajos basados en [Alba2007], se ejecutaron casos de prueba principalmente con 5, 10 o 15 empleados, 10, 20 o 30 tareas y de 2 a 10 habilidades por empleado. Uno de los aspectos que se evalúan en estos trabajos, es la tasa de éxito, que se refiere al porcentaje de ejecuciones en las cuales el algoritmo de optimización encuentra soluciones factibles al problema. En todos los casos se concluye que con 30 tareas o más, es muy difícil obtener soluciones factibles, pues se viola principalmente la restricción del problema referente a que el proyecto no debe presentar tiempo extra. Además, cuando los empleados presentan pocas habilidades, es muy probable que se viole la restricción referente a que los empleados deben cubrir, entre todos, con todas las habilidades requeridas por una determinada tarea. En algunos trabajos se realizó la comparación con otros algoritmos sobre la tasa de éxito obtenida. En [García2014] por ejemplo, en configuraciones con 10 tareas se presentó una tasa de éxito más alta que en [Alba2007], sin embargo, con 20 tareas, el enfoque de [Alba2007] obtuvo ligeramente una tasa de éxito mayor. En [Xiao2013], la tasa de éxito, la duración, el costo y la aptitud de las soluciones, fue superior respecto al enfoque de [Alba2007] también.

Otro de los aspectos que se evaluaron, fue el desempeño de los algoritmos con respecto a la métrica de hipervolumen, principalmente. Recuérdese que el hipervolumen es una métrica que mide el espacio cubierto por un conjunto de soluciones no dominadas en un espacio objetivo. En [Luna2014], el algoritmo PAES tuvo mejor desempeño, sin embargo, en [Chicano2011] y [Chicano2012], NSGA fue más favorable.

Una situación adicional que se observó, es que pocos trabajos se enfocan en analizar las soluciones obtenidas. Sólo en [Alba2007] y [Chicano2011], se identificó que la duración y el costo de un proyecto, regularmente presentan una correlación inversa, aunque no siempre. Cuando se incrementa el número de empleados, las tareas se pueden completar en menor tiempo y la duración del proyecto disminuye, sin embargo, el costo se incrementa.

2.6. Conclusiones de la revisión bibliográfica

Al revisar estos trabajos, se han observado buenos avances en la resolución del problema de planificación de proyectos de software, un problema que es recurrente en diversas compañías de desarrollo de software. Se observó que estos trabajos definen un modelo de planeación para resolver este problema, sin embargo, se ha identificado que dichos modelos presentan algunas omisiones con respecto a las características de empleados y tareas, por lo que se limita su adopción a proyectos de software reales.

Primero, los empleados pueden tener habilidades solo en ciertos tipos de tareas de un proyecto, por ejemplo, en Bases de Datos, en programación de interfaces gráficas de usuario, etc. Esta situación es difícil de imaginar en la práctica, ya que se espera que un empleado tenga conocimientos en un área de desarrollo de software, también los tenga en otras áreas, por ejemplo en diseño y codificación o en codificación y pruebas. Además, en la mayoría de los trabajos no se toma en cuenta la productividad de los empleados, ya que consideramos que esta característica permite estimar el tiempo real que tomará desarrollar las tareas, también puede estar relacionada con la experiencia de los empleados en el desarrollo de tareas.

Otra situación que se identificó, es que en los modelos de planeación propuestos, las tareas pueden ser desarrolladas por varios empleados a la vez, sin embargo, se ha observado que en algunas metodologías de desarrollo, por ejemplo, iterativo/incremental, las tareas normalmente son asignadas de forma individual, ya que se consideran unidades de trabajo de corta duración (estimadas en horas o en días).

También, se observó que el desempeño de los algoritmos se degrada con proyectos de 30 o más tareas, lo cual significa que en la práctica difícilmente sería viable desarrollar proyectos de mayor tamaño. Este problema se debe principalmente a que el proyecto, en distintas soluciones, presenta tiempo extra, o bien, porque algunas tareas no pueden ser completadas porque los empleados en conjunto, no cuentan con las habilidades requeridas por esas tareas.

Los hallazgos identificados permitieron proponer algunas mejoras, que consisten principalmente en ajustes al modelo de planeación de Alba y Chicano [Alba2007] y un algoritmo para resolver el conflicto del tiempo extra en las soluciones.

Capítulo 3

Desarrollo de la propuesta

Como se mencionó en el capítulo introductorio, es deseable generar escenarios alternativos de planeación en etapas tempranas de un proyecto de desarrollo de software, con el fin de comparar tiempos y costos para tomar una decisión al respecto. Generar estimaciones de tiempo y costo para un proyecto puede llegar a ser una actividad relativamente compleja por la cantidad de ajustes que se tienen que hacer y por el tiempo que se tiene que invertir. Como apoyo a este problema, se propuso desarrollar una herramienta para generar escenarios de planeación, considerando las características de los recursos y de las tareas de un plan de proyecto. A esta herramienta se le ha dado el nombre de "GENESPLAN" (GENerador de EScenarios de PLANeación) y en este capítulo se describen sus elementos principales, así como los aspectos más relevantes de su diseño. Al final, se presentan unas imágenes del prototipo de la herramienta.

3.1. Elementos principales de la herramienta para generar escenarios

De acuerdo con los trabajos que tratan el problema de planeación de proyectos de software [Alba2007], es importante considerar al menos dos elementos en el proceso de generación de soluciones:

1. Un *modelo de planeación* para representar las características de los recursos y de las tareas de un plan de proyecto de software.

2. Una *técnica de optimización* para generar el conjunto de soluciones, a partir de las variables del modelo de planeación.

Para el desarrollo de GENESPLAN, se han retomado también ambos elementos, y en el caso del modelo de planeación se han hecho los ajustes correspondientes.

En el capítulo anterior se identificó que la mayoría de los modelos de planeación de los trabajos revisados acerca del problema de planeación de proyectos de software, presentan una visión relativamente simple de las características de los recursos y de las tareas, lo cual no permite acomodar ciertos aspectos observados en la práctica. Por ejemplo, en la mayoría de las propuestas no se considera el nivel de productividad[2] de los empleados, ya que estos pueden completar tareas en tiempos distintos, ya sea por la experiencia que poseen o por su dominio en las tecnologías requeridas. Tampoco es posible representar perfiles de recursos relativos al contexto organizacional descrito en la sección 1.2, en la cual se mencionó que algunas compañías pueden clasificar a sus empleados según las habilidades y conocimientos poseídos en ciertas disciplinas de un proyecto. Además, en los modelos propuestos se asume que los empleados poseen habilidades solo en ciertas actividades o áreas de un proyecto, por ejemplo en Administración de Bases de Datos, en Codificación de módulos, etc., sin embargo, es común que un empleado pueda tener habilidades en

[2] La productividad de los empleados consiste en la capacidad que tienen para completar una tarea (de cierto tamaño) en una unidad de tiempo, o dicho de otra manera, la rapidez con que completan una tarea.

tareas de múltiples disciplinas o fases de un proyecto, por ejemplo: diseño, codificación y pruebas. Ante estas observaciones, se decidió adaptar un modelo de planeación en el cual se reflejan características un poco más realistas de proyectos de desarrollo de software.

Con respecto al uso de técnicas de optimización, algunas de ellas han demostrado generar resultados muy satisfactorios en distintos problemas, y además, ya existen implementaciones de las mismas en código abierto, mismo que se ha integrado en esta herramienta. Más adelante se describen las técnicas de optimización utilizadas para realizar experimentos con la herramienta.

3.2. El modelo de planeación para la generación de escenarios

Para integrar el modelo de planeación en la herramienta, se tomó como base uno de los modelos en el cual se basan la mayoría de los trabajos que se revisaron, el modelo de Alba y Chicano [Alba2007]. En este modelo se describen las características de empleados y tareas de un proyecto, y se hicieron ajustes tomando en cuenta ciertos requerimientos.

3.2.1. Requerimientos y decisiones de diseño para el modelo de planeación

Uno de los aspectos deseables es poder modelar distintos tipos de organizaciones en las cuales los recursos pueden ser clasificados de distintas maneras según su nivel de habilidad y/o experiencia. Por lo tanto, la *flexibilidad* para representar distintos contextos organizacionales (REQ-01) es el requerimiento principal que se ha buscado cubrir con la reformulación del modelo de planeación. Esto significa que debe ser posible representar distintos perfiles de recursos y clasificar a las tareas según la fase o la disciplina del proyecto en la que se desarrollen: requerimientos, diseño, implementación, etc. (REQ-01a). Adicionalmente, se requiere un indicador que permita medir el nivel de productividad y/o experiencia de los recursos para desarrollar tareas de cierta disciplina (REQ-01b).

Decisión	Requerimientos que satisface	Justificación
Asociar una "disciplina" a las tareas. Con disciplina nos referimos a una fase del ciclo de vida de un proyecto de software, por ejemplo: requerimientos, diseño, implementación, etc.	REQ-01a	Por medio de una disciplina, es posible asignar tareas a empleados, con base en su perfil técnico. Por ejemplo: las tareas de diseño pueden ser asignadas a arquitectos, las tareas de implementación a programadores, etc.
Vector de factores de productividad para los empleados.	REQ-01b	Esta característica permite identificar el grado o el porcentaje de productividad y/o experiencia que presenta un recurso en distintas disciplinas del proyecto. Si el factor de productividad es más alto en cierta disciplina, significa que el recurso tiene la capacidad de completar tareas de esa disciplina en un tiempomenor.

Tabla 3.1. Decisiones de diseño para el modelo de planeación en GENESPLAN.

Para cumplir con los requerimientos de flexibilidad mencionados, se han tomado las decisiones de diseño presentadas en la Tabla 3.1. A partir de estas decisiones, se realizaron ajustes al modelo de planeación de Alba y Chicano [Alba2007].

3.2.2 Entidades del modelo de planeación

Para adaptar el modelo de planeación que se integró en GENESPLAN, se retomaron las dos entidades principales del modelo de Alba y Chicano [Alba2007]: *Empleado* y *Tarea*. Una tarea es una unidad de trabajo que puede ser desarrollada en un periodo de tiempo relativamente corto (horas o días) y un empleado es aquella persona que posee las habilidades y los conocimientos técnicos necesarios para completar una tarea. Las características y las relaciones de estas entidades, se han representado en un diagrama de clases UML (ver Figura 3.1).

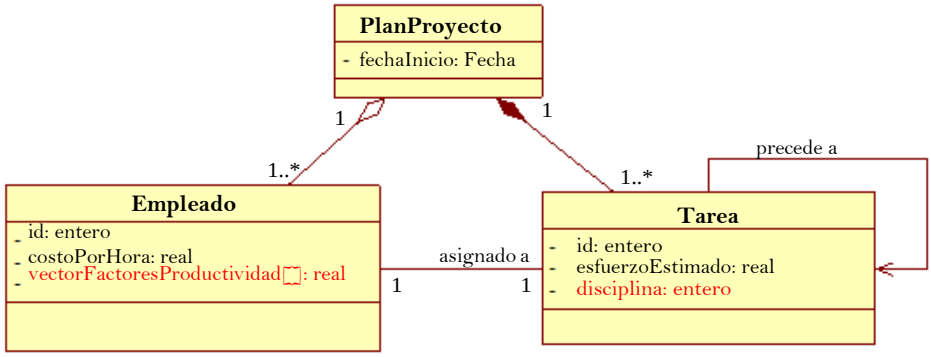


Figura 3.1. Entidades del modelo de planeación en GENESPLAN.

Un plan de proyecto tiene una fecha inicial definida y se compone de un conjunto de tareas y de un equipo de empleados.

Por una parte, los empleados tienen asociadas las siguientes características:

a. Un identificador e^{id}_i (atributo "id" en el modelo de la Figura 3.1).
b. Un costo por hora e^{costo}_i (atributo "costoPorHora" en el modelo de la Figura 3.1).
c. Un vector de factores de productividad $e^{vectorProductividades[]}_i$ en las disciplinas del ciclo de vida del proyecto (atributo "vectorFactoresProductividad" en el modelo de la Figura 3.1).

El costo es la cantidad monetaria percibida por un empleado debido al desarrollo de tareas en un proyecto. El vector de factores de productividad es un conjunto de tuplas de la forma $< disciplina_i, f_i >$, donde f_i representa el factor de productividad de un empleado en la disciplina i. Los factores de productividad son valores que se encuentran en el intervalo [0,1] (que corresponden a un intervalo de [0%-100%]). Cuando un factor de productividad tiene el valor 0 significa que el empleado no cuenta con experiencia para realizar tareas de la disciplina correspondiente y cuando el valor es 1 entonces el empleado posee la productividad máxima en esa disciplina. Un ejemplo de vector de factores de productividad es el siguiente: [*<requerimientos, 0.5>,<diseño, 0.5>,<implementación, 1.0>*], el cual indica que el empleado posee un nivel de productividad del 50% en tareas de requerimientos y diseño, pero posee la productividad máxima en implementación. Por medio del vector de factores de productividad, es posible representar perfiles de empleados relativos al contexto organizacional (REQ-01a), según la disciplina del ciclo de vida en la que se especialicen. Algunos ejemplos de perfiles son los siguientes:

a. Perfil analista de requerimientos: Cuando un empleado tiene mayor conocimiento y posee un factor de productividad más alto en la disciplina de requerimientos. Un ejemplo de este perfil se presenta en el diagrama radial de la Figura 3.2. En este ejemplo, se asume que el empleado tiene también conocimiento en las disciplinas de diseño, implementación, pruebas y administración de proyectos, aunque no son su especialidad.

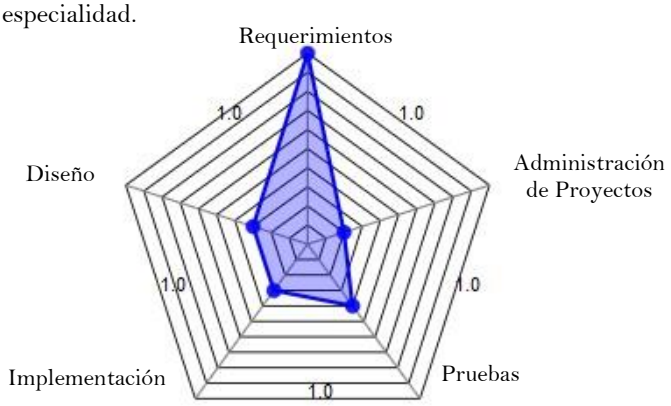


Figura 3.2. Ejemplo de un perfil analista de requerimientos.

b. Perfil arquitecto de software: Cuando un empleado tiene mayor conocimiento y posee un factor de productividad más alto en la disciplina de diseño. En la Figura 3.3 se presenta un diagrama que ilustra este perfil. Al igual que el caso anterior, se asume que el empleado también tiene conocimiento en el resto de las disciplinas, pero tampoco son su especialidad.

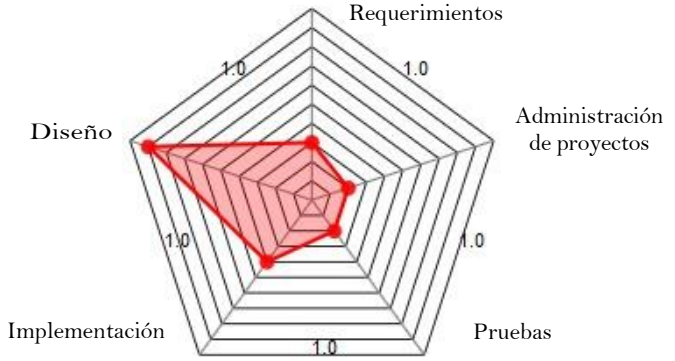


Figura 3.3. Ejemplo de un perfil arquitecto de software.

c. Perfil mixto: Cuando un empleado puede especializarse en múltiples disciplinas. Un ejemplo de este perfil es representado en la Figura 3.4. En este ejemplo, el empleado es más competente en las disciplinas de requerimientos, implementación y administración de proyectos, pero puede tener conocimientos en el resto de las disciplinas, aunque no sean su especialidad.

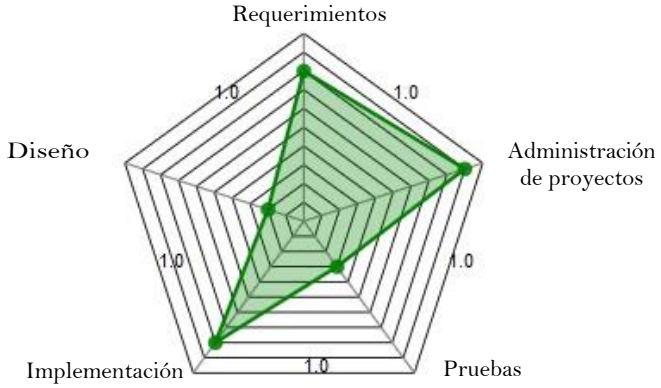


Figura 3.4. Ejemplo de un perfil mixto.

Por simplicidad, en estos ejemplos se consideraron 5 disciplinas, pero es posible modelar proyectos con un número variable de éstas.

Además, para cada perfil es posible modelar un nivel de experiencia y/o productividad. Retomando el ejemplo del perfil arquitecto de software, se pueden asociar niveles de productividad como en el ejemplo de la Figura 3.5. En 3.5 a) el arquitecto posee un factor relativamente bajo en diseño, por lo que puede tener asociado un nivel junior. Por lo contrario, en 3.5 c) el arquitecto posee un factor de productividad máxima en diseño, por lo que tiene asociado un nivel experto.

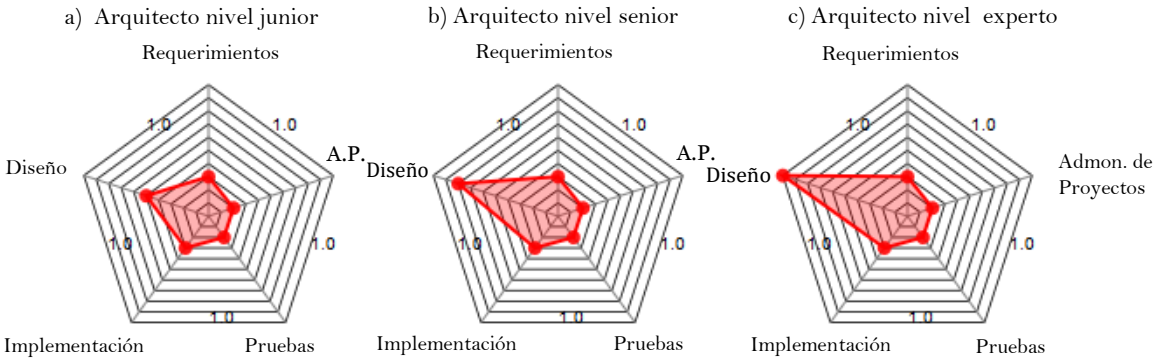


Figura 3.5. Ejemplos de niveles de productividad para un perfil arquitecto de software.

Es importante aclarar que en términos prácticos no existe un significado claro entre factores de productividad con poca diferencia, por ejemplo, no es posible definir la diferencia que hay entre un un empleado con factor de productividad de 10% y un empleado con un factor de 20%, por lo cual al hacer la clasificación de los empleados (como en el ejemplo de la Figura 3.5), es recomendable establecer factores de productividad con mayor diferencia. Un ejemplo puede ser que a empleados de nivel junior se les asigne un factor de 40%, a empleados de nivel senior un factor del 70% y a empleados de nivel experto un factor del 90% o del 100%. Incluso, es posible que empleados con el mismo nivel tengan factores de productividad distintos, por ejemplo: 2 empleados de nivel junior podrían tener factores de productividad de 30% y de 35%, respectivamente.

En la práctica, existen enfoques que permiten medir la productividad de los empleados en cada disciplina, uno de ellos es PSP (Proceso Personal de Software) [Humphrey1997]. PSP permite llevar un registro histórico de las métricas asociadas a cada desarrollador, particularmente, de su productividad promedio por fase o disciplina, la cual se obtiene al medir la correlación entre los tamaños de los productos desarrollados y los tiempos (efectivos) invertidos en cada una de esas fases. Para el modelo propuesto, se requiere que los valores de productividad de cada empleado sean normalizados.

Comúnmente, los equipos de trabajo están conformados por empleados con perfiles y niveles de productividad distintos. Un perfil puede simplificar la asignación de tareas a un empleado, por ejemplo, a un arquitecto de software se le asignan preferentemente tareas de diseño y a un programador tareas de codificación; mientras que el nivel de productividad de los empleados permite estimar la duración real de las tareas.

En la Figura 3.6 se presenta un ejemplo de equipo de empleados con distintos perfiles: un analista de requerimientos y uno de negocio, 2 arquitectos de software, 3 programadores y 2 testers. También es posible tener empleados con un perfil mixto, por ejemplo analista-programador. En este ejemplo, los empleados tienen asociado un nivel de experiencia (junior o senior) y un costo por hora con base en su nivel y su perfil.

Figura 3.6. Ejemplo de conformación de equipo para un proyecto de desarrollo de software.

Por otra parte, las tareas tienen asociadas las siguientes características:

a. Un identificador t_j^{id} (atributo "id" del modelo en UML, Figura 3.1).

b. Un esfuerzo estimado $t_j^{esfuerzo}$ (atributo "esfuerzoEstimado" en el modelo de la Figura 3.1), el cual corresponde al tiempo efectivo en horas que le tomaría a un empleado con productividad máxima desarrollar esa tarea. Se asume que este valor ha sido previamente calculado analizando la complejidad y el tamaño de la tarea.

c. Una disciplina $t_j^{disciplina}$ (atributo "disciplina" del modelo de la Figura 3.1). Cada tarea está asociada con alguna de las disciplinas del ciclo de vida de un proyecto. Con esta característica, es posible clasificar tareas de requerimientos, de diseño, de implementación, de pruebas, etc.

El comienzo de una tarea puede depender de la finalización de otras (relación de asociación en la entidad "Tarea" del modelo de la Figura 3.1). Por ejemplo, el arranque de una tarea de codificación puede depender de que haya concluido la correspondiente tarea de diseño. Las dependencias entre tareas pueden representarse a través de un *grafo de precedencia de tareas* (GPT). El GPT es un grafo dirigido acíclico expresado como $G(V, A)$, donde $V = \{t_1, t_2, \ldots, t_j\}$ es un conjunto de vértices t_j, los cuales corresponden a las tareas del proyecto y $A = \{(t_i, t_j), \ldots, (t_m, t_n)\}$ es el conjunto de aristas, donde (t_m, t_n) representa una dependencia de la tarea t_n con la tarea t_m. En la Figura 3.7 se presenta un ejemplo de un GTP con 6 vértices (tareas) y 7 aristas (dependencias). En este ejemplo t_2 y t_3 pueden comenzar hasta que finalice t_1, mientras que t_5 puede comenzar hasta que finalicen t_2 y t_3.

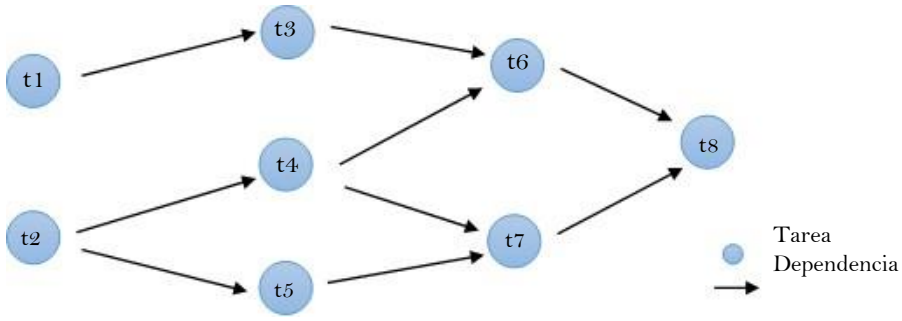


Figura 3.7. Representación de un grafo de precedencia de tareas (GPT).

Al ser una unidad de corta duración, una tarea puede ser desarrollada por un solo empleado (relación de asociación entre la entidad "Tarea" y "Empleado" del modelo de la Figura 3.1), lo cual difiere del modelo de Alba y Chicano [Alba2007], en el cual varios empleados pueden participar a la vez.

3.2.3 Representación de una solución

Una vez conocidas las variables del modelo de planeación, se procede a especificar una representación para las posibles soluciones. En este caso, una solución es representada como una matriz binaria de asignaciones $M = (m_{ij})$ de tamaño $E \; x \; T$, donde E es el número de empleados y T es el número de tareas. El elemento m_{ij} es un valor que puede tomar el valor 0 o 1. Si m_{ij}= 1, significa que la tarea j es asignada al empleado i. La matriz binaria permite establecer una asociación entre las tareas y los empleados, y se utiliza al momento de calcular la duración y el costo de los posibles escenarios. Un ejemplo de matriz de asignaciones es presentado en la Figura 3.8., en el cual se consideran 4 empleados (que corresponde a las filas) y 8 tareas (que corresponden a las columnas). En este caso en m_{21}, m_{32}, m_{13}, m_{24}, m_{45}, m_{36}, m_{17} y m_{48} es donde se presentan las asignaciones. Puede suceder que en algunos casos, uno o más empleados queden sin tareas asignadas, pero no debe suceder que haya tareas sin asignar.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
e_1	0	0	1	0	0	0	1	0
e_2	1	0	0	1	0	0	0	0
e_3	0	1	0	0	0	1	0	0
e_4	0	0	0	0	1	0	0	1

Figura 3.8.Representación de una solución para el problema de generación de escenarios.

3.2.4. Cálculo de la duración y costo de un proyecto

A partir de los valores de las características de los empleados, el grafo de precedencia de tareas (GPT) y la matriz binaria de asignaciones, es posible calcular la duración y el costo de un proyecto mediante un conjunto de ecuaciones. Primero, la duración de una tarea se puede calcular de la siguiente manera:

$$t_j^{duracion} = \frac{t_j^{esfuerzo}}{e_i^{vectorProductividades[t_j^{disciplina}]}} \qquad (1)$$

La ecuación (1) indica que la duración de la tarea j está dada por el cociente entre el esfuerzo estimado y el factor de productividad del empleado asignado a esa tarea. Por ejemplo, si una tarea de diseño tiene un esfuerzo estimado de 20 horas y el empleado asignado tiene un factor de productividad de 0.5 en esa disciplina, entonces la duración real de esa tarea es $\frac{20\,hrs}{0.5} = 40\;horas$. Al obtener la duración, es posible calcular el momento de inicio (2) y el momento de finalización (3) de cada tarea:

$$t_j^{inicio} = \left\{0 \text{ si } t_j \text{ no tiene tareas precedentes y } max\{t_i^{fin}\ \forall i \mid t_i \text{ precede a } t_j\} \text{ en otro caso}\right\}. \qquad (2)$$

$$t_j^{fin} = t_j^{inicio} + t_j^{duracion} \qquad (3)$$

La ecuación (2) indica que el momento de inicio de una tarea t_j es 0, si ésta no tiene tareas precedentes, y en caso contrario, es el máximo de los momentos de finalización de sus tareas precedentes; mientras que (3) indica que el momento de finalización de una tarea t_j está dado por la suma de su momento de inicio y su duración. A partir de (3) es posible estimar la duración total del proyecto $P_{duracion}$ (en tiempo efectivo):

$P_{duracion} = \max\{t_j^{fin} | j = 1, \dots, T\}$, donde $|T|$ es el número de tareas (4)

La ecuación (4) indica que la duración total del proyecto es el valor máximo de los momentos de finalización de las tareas.

En la Figura 3.9 se presenta un ejemplo en el cual se calcula la duración de un proyecto y los momentos de inicio y finalización de las tareas mediante el método de ruta crítica. En este ejemplo el proyecto consta de 8 tareas con sus respectivas dependencias, donde las flechas en rojo indican la ruta crítica. Se puede observar como el tiempo de inicio de tareas que tienen predecesoras se obtiene del valor máximo de los tiempos de finalización de esas predecesoras (ecuaciones (2) y (3)).
En este ejemplo t_7 posee el valor máximo de los tiempos de finalización de todas las tareas, el cual corresponde a la duración efectiva del proyecto. Por convención, la duración del proyecto está en horas.

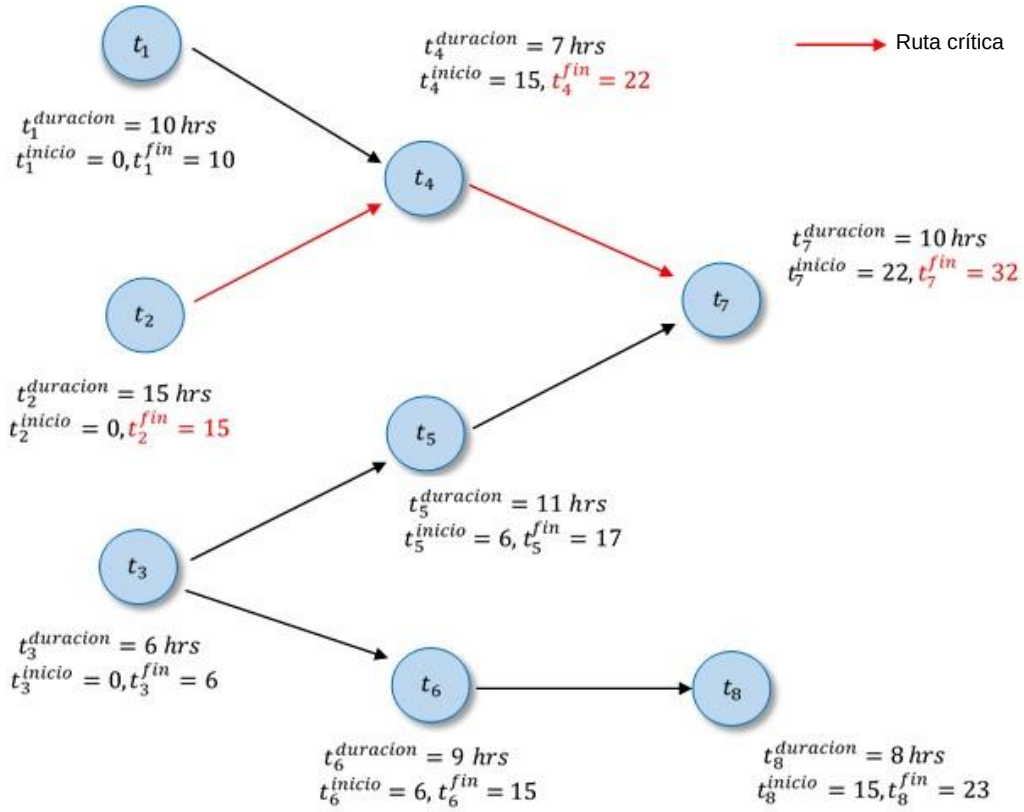


Figura 3.9. Cálculo de la duración efectiva de un proyecto por el método de ruta crítica.

Mientras tanto, el costo de una tarea está dado por (5) y el costo de un proyecto por (6).

$$t_j^{costo} = t_j^{duracion} \times e_i^{costo} \quad (5)$$

$$P_{costo} = \sum_{j=1}^{n} t_j^{costo}, \text{ donde } n \text{ es el número de tareas.} \quad (6)$$

El costo de una tarea j es el producto de su duración y el costo por hora del empleado asignado a esa tarea. El costo total del proyecto es la suma de los costos de todas las tareas. Los valores $P_{duracion}$ y P_{costo} son evaluados por un algoritmo de optimización durante el proceso de generación de soluciones.

3.2.5 Restricciones del modelo

Al igual que en el modelo de Alba y Chicano [Alba2007], el modelo de planeación que se ha ajustado, incluye un conjunto de restricciones, en este caso las siguientes:

1) *Ninguna tarea puede quedar sin asignar.* Para que el proyecto pueda ser completado, se requiere que todas las tareas tengan asociado un empleado.

2) *Una tarea solo puede ser asignada a un empleado con factor de productividad mayor a 0.* Un empleado no puede desarrollar una tarea del ciclo de desarrollo de software en la cual no tenga experiencia.

3) *No se permite la sobrecarga o el tiempo extra en el proyecto.* Los empleados no pueden trabajar en 2 o más tareas al mismo tiempo.

Al igual que el costo y la duración, las restricciones son evaluadas por un algoritmo de optimización. Cuando se viola alguna de estas restricciones se dice que la solución es *no factible* y no puede ser considerada como una opción viable.

3.3. Algoritmos de optimización para la generación de escenarios

Una vez definidas las características del modelo de planeación y las restricciones, se requiere el uso de alguna técnica o algoritmo de optimización para generar un conjunto de soluciones, que en nuestro contexto corresponden a escenarios de planeación para un plan de proyecto. En los capítulos anteriores se mencionó que es necesario recurrir a heurísticas, debido a que es impráctico generar todos los escenarios posibles (incluso computacionalmente) para hallar los óptimos. Una clase particular de heurísticas son los algoritmos evolutivos (descritos brevemente en el capítulo 2), que como su nombre lo indica, se basan en procesos evolutivos para encontrar un conjunto de soluciones cercano al óptimo.

De los trabajos revisados en el estado del arte, se identificó que NSGA-II [Deb2002] y SPEA2 [Zitzler2001], fueron 2 de los algoritmos evolutivos multiobjetivo que presentaron resultados más favorables en la resolución del problema de planificación de proyectos de software. Particularmente, NSGA-II sigue siendo un estándar de comparación con respecto a otros algoritmos. Por su parte, SMS-EMOA [Beume2007] ha sido considerado como uno de los mejores algoritmos para la resolución de problemas de optimización multiobjetivo en general. Por estas razones, los 3 algoritmos mencionados fueron utilizados para realizar los experimentos correspondientes, con el fin de identificar el más apropiado para GENESPLAN.

Por otra parte, estos algoritmos ya cuentan con una implementación en código abierto, lo cual permite simplificar el proceso de generación de escenarios; no obstante, es importante conocer el funcionamiento y las características de estos algoritmos. Debido a que la descripción de los algoritmos utiliza una simbología y algunos conceptos particulares, en la Tabla 3.2 se presenta el significado de la notación utilizada, principalmente en NSGA-II y SPEA2.

Notación	Significado
P_t y Q_t	Representan poblaciones (grupos de individuos) obtenidas por un algoritmo genético en la generación t.
F_i	Representa el i-ésimo frente de Pareto obtenido por un algoritmo de optimización. Un frente de Pareto es un conjunto de soluciones no dominadas entre sí, las cuales tienen la propiedad de que ninguna de ellas es superior o inferior que el resto, con respecto a todos los objetivos de optimización.
$\preccurlyeq_n$	Es el operador "crowded" en NSGA-II. Este operador sirve como criterio de selección de soluciones durante el proceso de optimización, el cual se basa en la comparación de dos propiedades que presentan los individuos: crowding distance ($i_{distance}$) y rango de dominancia (i_{rank}).
$i_{distance}$	Es el "crowding distance" (distancia de apiñamiento) de un individuo . Es un mecanismo que utiliza NSGA-II para calcular la proximidad o distancia de un individuo i con respecto a lo que lo rodean, midiendo el perímetro del área que forma con respecto a sus vecinos más cercanos.
i_{rank}	En NSGA-II, es el rango de dominancia de un individuo i. Es un valor que está asociado directamente con el frente de Pareto al cual pertenece dicho individuo. Un individuo que pertenece a un frente primario F_1 tiene un rango de dominancia mejor que individuo que pertenece a un frente secundario F_2.
$\overline{P}_t$	En SPEA2, es un conjunto externo utilizado para la preservación de individuos de una población P en la generación t, con el fin de evitar que estas se pierdan durante el proceso de optimización.
$S(i) = \lvert\{j \mid j \in P_t \cup \overline{P}_t \wedge i \preccurlyeq j\}\rvert$	En SPEA2, es un valor de fuerza (strength) asociado a un individuo i, el cual representa el número de individuos que éste domina. La expresión del lado derecho de la igualdad representa la cardinalidad ($\lvert\ \rvert$) de los individuos j que están en la poblaciones P y $\overline{P}$, tal que j es dominado por el individuo i, respecto al operador $\preccurlyeq$ (crowded).
$R(i)$	En SPEA2 es el fitness (aptitud) primario de un individuo , el cual se calcula mediante la suma de los valores de fuerza S de sus individuos dominantes. Si $R(i) = 0$, significa que el individuo i no es dominado por otros individuos en la población. Por el contrario si el valor de $R(i)$ es alto significa que el individuo i es dominado por otros individuos.
$D(i)$	En SPEA2, es una función de densidad asociada a un individuo i, la cual es utilizada durante el proceso de selección de soluciones.
fitness (aptitud)	Es un puntaje o valor que se asigna a un individuo en base a ciertos parámetros, el cual es usado como criterio de selección de soluciones en un algoritmo de optimización. En SPEA2 por ejemplo, el fitness de un individuo está dado por $F(i) = R(i) + D(i)$

Tabla 3.2. Notación utilizada en los algoritmos de optimización NSGA-II y SPEA2.

3.3.1. NSGA-II

NSGA-II es un algoritmo de optimización multiobjetivo que fue propuesto como una mejora de su antecesor NSGA (Non dominated sorting genetic algorithm) [Deb2002]. Como lo indica su nombre, este algoritmo utiliza un ordenamiento rápido no dominado, en el cual se obtiene un conjunto de frentes $F_1, F_2, ..., F_n$, a partir de una población inicial P_0, donde F_1 tiene el mejor nivel de no dominancia. Posteriormente, se calcula la distancia de apiñamiento (crowding distance) para las soluciones de cada uno de los frentes obtenidos, así como su rango de dominancia. Las soluciones son ordenadas con base en el operador *crowded* ($\preccurlyeq_n$), tomando en cuenta los dos parámetros anteriores. Si dos soluciones tienen distintos rangos de dominancia, se selecciona aquella que tenga el menor rango, o bien, si tienen el mismo rango, se selecciona aquella con la menor distancia de apiñamiento. A la población resultante de tamaño N se le aplican los operadores de selección, cruce y mutación para generar la población de la siguiente generación. El proceso se repite hasta cumplirse un criterio de paro.

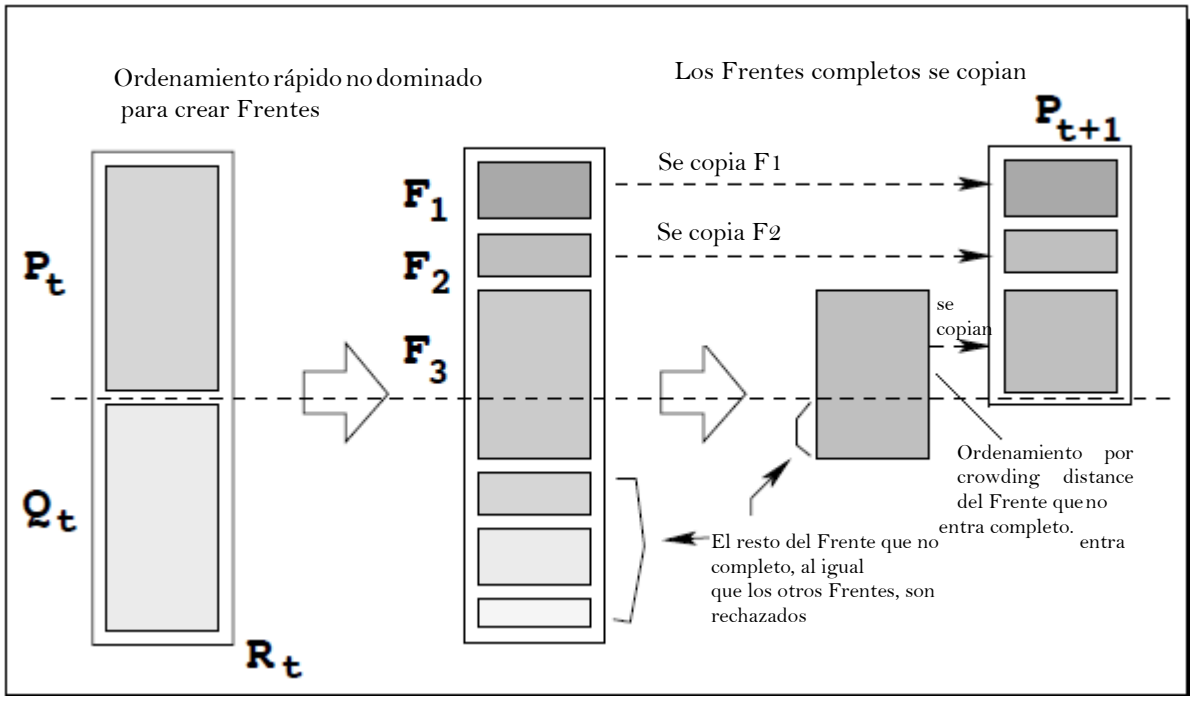


Figura 3.10. Proceso de selección de soluciones candidatas en NSGA-II [Deb2002].

El ciclo de ejecución de NSGA-II es el siguiente:

1. Se crea una población inicial de padres P_o. Al inicio, por medio operadores de selección, recombinación y mutación, se obtiene una población de descendientes Q_0. Tanto P_0 como Q_o son de tamaño N.
2. La población resultante R_t de tamaño $2N$, se ordena de acuerdo a su nivel de no dominancia para obtener los frentes $F_1, ..., F_n$, donde el mejor de ellos es F_1.
3. La nueva población P_{t+1} se forma agregando individuos de los frentes en orden creciente. Sólo se agregan frentes completos, como se puede observar en la Figura 3.10.
4. Si no se han completado N individuos en P_{t+1}, entonces los individuos faltantes se toman como sigue: se ordenan los elementos del último frente visitado en forma descendente mediante el operador crowding y se toman los primeros individuos de ese frente hasta completar N.

5. La población descendiente Q_{t+1} para la siguiente generación se obtiene mediante operadores de selección, recombinación y mutación; y el criterio de selección es por medio del operador $\preccurlyeq_n$. El proceso continúa hasta cumplirse un criterio de paro.

3.3.2. SPEA2

Este algoritmo fue propuesto como una mejora al Strength Pareto Evolutionary Algorithm (SPEA) [Zitzler2001].

Las principales características de este algoritmo son las siguientes:

1. Se utiliza un esquema de asignación de *fitness* (aptitud), en el cual para cada individuo de la población, se toma en cuenta el número de individuos que domina y el número de individuos que lo dominan.

2. Se incorpora una técnica de estimación de densidad del vecino más cercano, la cual permite guiar de manera más precisa el proceso de búsqueda de soluciones.

SPEA2 utiliza una población de tamaño fijo N y un archivo externo de tamaño $\overline{N}$. El algoritmo se ejecuta hasta alcanzar un número máximo de iteraciones T.

Las etapas de este algoritmo son las siguientes:

1. Inicialización. En esta etapa se genera una población inicial P_0, se crea un archivo externo vacío $\overline{P_0}$ y se establece el número de iteraciones $t = 0$.

2. Asignación de fitness (aptitud). Se calculan los valores de fitness de los individuos en P_t y $\overline{P_t}$. Para esto, primero se calcula un valor de fuerza $S(i) = |\{j \mid j \varepsilon P_t \cup \overline{P_t} \wedge i \preccurlyeq j\}|$, el cual denota el número de soluciones que domina la solución i. Después se obtiene el fitness primario de cada individuo $R(i)$ sumando los valores de fuerza S de sus dominadores en la población y en el archivo externo. Posteriormente se calcula una función de densidad $D(i)$ para cada individuo para medir la proximidad con sus vecinos más cercanos. Finalmente, el fitness del individuo está dado por: $(i) = R(i) + D(i)$.

3. Selección ambiental. En esta etapa se copian todos los individuos no dominados de la población P_t y $\overline{P_t}$ en $\overline{P_{t+1}}$. Si el tamaño de $\overline{P_{t+1}}$ excede N, entonces se reduce por medio de un procedimiento de truncado, el cual iterativamente va eliminando individuos de este archivo hasta que su tamaño sea N. En otro caso, si el tamaño de $\overline{P_{t+1}} < N$, entonces este se llena con individuos dominados en P_t y $\overline{P_t}$.

4. Terminación. Si se ha alcanzado el número máximo de iteraciones o se cumple cualquier otro criterio de paro, entonces hacer $A = P_{t+1}$, donde A es el archivo final de soluciones.

5. Selección de padres. Se lleva a cabo un torneo binario de selección con reemplazo sobre el archivo P_{t+1} para la obtención de padres.

6. Variación. Se aplican operadores de recombinación y mutación a los padres obtenidos y se establece $\overline{P_{t+1}}$ como la población resultante. Se actualiza el contador de iteraciones t y se vuelve a la etapa 2 (asignación de fitness).

SPEA2 proporciona una estrategia de asignación de fitness mejorada y una mejor técnica de truncamiento de archivo, comparada con la de su predecesor SPEA [Zitzler1999].

3.3.3 SMS-EMOA

SMS-EMOA (Multiobjective selection based on dominated hypervolume) es un algoritmo que cuenta con un operador de selección que se basa en la métrica de hipervolumen y que utiliza el concepto de ordenamiento no dominado [Beume2007]. La población de individuos evoluciona a un conjunto bien distribuido de soluciones, de este modo el algoritmo se enfoca en regiones interesantes del frente Pareto.

El algoritmo consta de los siguientes pasos:

1. Se inicializa una población de μ individuos y un contador de iteraciones $(t = 0)$.
2. El siguiente proceso se repite hasta cumplir condición de paro:

    a. Se genera un nuevo individuo por medio de operadores de variación aleatoria. Este individuo va a pertenecer a la población de la siguiente generación.
    b. Se seleccionan los μ mejores individuos mediante un algoritmo llamado *reduce*.
    c. Se incrementa $t = t + 1$.

El algoritmo reduce(Q) consta a su vez de los siguientes pasos:

Se obtienen los frentes de la población Q.

a. Si hay más de un frente entonces:
b. Se identifica la solución del peor frente que tenga mayor cantidad de puntos que la dominan.
c. Si hay un sólo frente entonces:

   Se identifica la solución que tenga una menor contribución de hipervolumen para el frente.

d. En cualquiera de los dos casos, se elimina la solución identificada.

3.4 Eliminación de sobrecarga en las soluciones

Una de las situaciones que se puede presentar cuando un algoritmo de optimización genera una solución candidata, es que se viole la restricción relativa al tiempo extra o sobrecarga en el proyecto. Esto sucede cuando al menos uno de los empleados tiene tareas asignadas que coinciden en algún periodo de tiempo (tareas en paralelo). Para ser más ilustrativos con este problema, en la Figura 3.11 se presenta un ejemplo de calendario de proyecto, en el cual un empleado (e_1) tiene asignadas dos tareas ($t_0 y t_4$), y de la misma manera, un empleado (e_0) tiene asignadas dos tareas (t_1 *y* t_2), en ambos casos las tareas se ejecutan en paralelo y requieren de tiempo extra para poder ser completadas.

Una forma de eliminar el tiempo extra que pueden presentar algunas soluciones, es que las tareas sean reprogramadas de tal manera que puedan ser ejecutadas una después de otra, respetando por supuesto las dependencias que tengan asociadas. Para lograrlo, se propuso un algoritmo muy simple de eliminación de sobrecarga.

Primero, para cada posible solución se verifica la presencia de sobrecarga, comenzando desde el tiempo t=0. En el momento en que un empleado tenga 2 tareas paralelas en un instante t de tiempo, se interrumpe el proceso de verificación.

Figura 3.11. Calendario de proyecto con presencia de sobrecarga.

Al identificar sobrecarga, se establece un nuevo orden de ejecución para las tareas sobrecargadas de los empleados. Para establecer un orden de ejecución, se toman en cuenta algunos criterios como las dependencias con otras tareas, la duración de las tareas, el momento de inicio que tenían las tareas.

Una vez establecido el nuevo orden de ejecución de las tareas, se asigna una dependencia secuencial que impide que dos tareas sean realizadas por el mismo empleado. Esta dependencia no forma parte del grafo de precedencia de tareas original (GPT) pero que es necesaria para calcular el nuevo momento de inicio y finalización de esa tarea. En la Figura 3.12 se presenta el ejemplo de calendario de proyecto de la Figura 3.11, al cual se le ha eliminado la sobrecarga o tiempo extra. Para fines ilustrativos, se han puesto en un color diferente las dependencias adicionales al GPT.

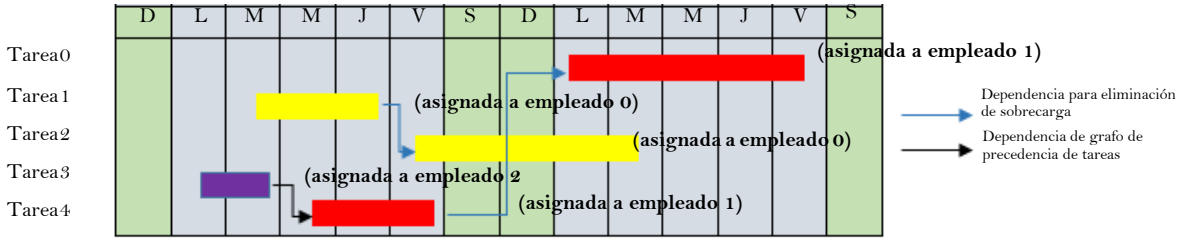


Figura 3.12. Calendario de proyecto con eliminación de sobrecarga.

La resolución de conflictos de sobrecarga es una etapa intermedia durante el proceso de optimización que permite "reparar" soluciones que violan esta restricción, de tal manera que sea posible obtener una mayor cantidad de soluciones factibles. Sin embargo, si en una ejecución todas o la mayoría de las soluciones candidatas requieren el uso del algoritmo de eliminación de sobrecarga, estas podrían resultar en una duración considerablemente mayor, principalmente cuando haya pocos empleados disponibles y la duración de tareas que sean asignadas en paralelo sea relativamente grande, por lo cual pudieran ser soluciones no satisfactorias. El conjunto final puede contener tanto soluciones que requirieron la aplicación del algoritmo de eliminación de sobrecarga, como soluciones que no lo requirieron.

3.5 Visión de la herramienta GENESPLAN

En esta sección se describe un panorama de la herramienta para generar escenarios y los requerimientos a nivel de usuario que se propusieron.

3.5.1. Requerimientos de negocio

En las siguientes secciones se describen los requerimientos de negocio de la herramienta GENESPLAN.

3.5.1.1. Antecedentes y oportunidad de negocio

Como se mencionó anteriormente, se ha observado que en algunas compañías de desarrollo de software las estimaciones de tiempo y costo se realizan de forma manual, limitando la posibilidad de generar múltiples escenarios con el fin de poder elegir una opción adecuada. La automatización de esta actividad permite reducir el tiempo invertido por aquellas personas dedicadas a realizar estimaciones, como un administrador de proyecto por ejemplo. Hasta el momento se desconocen herramientas o mecanismos que simplifiquen esta actividad, por lo que una herramienta podría servir de apoyo para aquellas personas que se encargan de realizar planes de proyecto, teniendo la oportunidad de comparar escenarios alternativos y seleccionar el más favorable.

3.5.1.2. Objetivos y necesidades de negocio

Con base en los antecedentes, se ha identificado el siguiente objetivo de negocio:

ON-1 Poder obtener el "mejor" escenario de planeación en el tiempo disponible (con el fin de obtener las mejores ganancias y satisfacción para un cliente).

Las necesidades derivadas del objetivo de negocio son las siguientes:

- *NEC-01*. Disponer de escenarios de planeación alternativos, en términos de estimación de tiempo y costo para un proyecto de software.

- *NEC-02*. Conocer el detalle de cada escenario de planeación, como la distribución de recursos y el calendario de tareas obtenido.

3.5.2. Enunciado de visión y características principales

La herramienta GENESPLAN consiste en una aplicación que permite generar escenarios de planeación con su respectiva estimación de tiempo y costo, a partir de las características de las tareas y de los empleados de un proyecto de desarrollo de software. Además, para cada escenario es posible consultar la asignación entre empleados y tareas, así como el calendario del proyecto. Una de las principales ventajas de la herramienta, es que las personas que se dedican a realizar planes de proyecto pueden disponer de escenarios alternativos de forma inmediata para tomar una decisión al respecto.

Las características[3] que debe satisfacer la herramienta propuesta y que están alineadas con las necesidades de negocio mencionadas anteriormente, se describen en la Tabla 3.3

Identificador	Descripción	Necesidad asociada
CAR-01	La herramienta debe permitir agregar los datos del modelo de planeación (tareas y empleados).	NEC-01
CAR-02	La herramienta debe permitir seleccionar la fecha de inicio del proyecto y los posibles días festivos o no laborales.	NEC-01
CAR-03	La herramienta debe generar y desplegar una lista de escenarios de planeación.	NEC-01
CAR-04	La herramienta debe permitir consultar el detalle de un escenario: a. La matriz de asignaciones, especificando el tiempo que van a dedicar los empleados a sus respectivas tareas. b. Un diagrama de Gantt con la duración de las tareas y sus dependencias.	NEC-02

Tabla 3.3. Características consideradas para el prototipo de GENESPLAN.

3.5.3. Flujo de trabajo para generar escenarios de planeación con la herramienta GENESPLAN

El proceso de generación de escenarios de planeación consta de distintas etapas, las cuales se ilustran en el diagrama de actividades de la Figura 3.13 y que a continuación se describen.

1. El usuario configura los parámetros del modelo de planeación como:
    a. El número de tareas y empleados, y sus respectivas características.
    b. El grafo de precedencia de tareas (GPT).
    c. La fecha de inicio del proyecto.

2. La herramienta verifica que los parámetros del modelo tengan la sintaxis correcta. Si algún parámetro es incorrecto, el usuario lo corrige.

3. Se ejecuta alguno de los algoritmos de optimización para generar y transformar soluciones candidatas hasta obtener la soluciones finales, las cuales corresponden a los escenarios de planeación.

4. Se presenta un conjunto de escenarios y para cada uno de ellos se muestra la duración del proyecto, el costo y un par de opciones para consultar la matriz de asignaciones entre recursos y tareas resultante y el calendario de tareas correspondiente, respectivamente.

[3] Las características son requerimientos de alto nivel que describen aspectos funcionales o no del sistema.

5. El usuario tiene la posibilidad de consultar la matriz de asignaciones y el calendario de tareas para un determinado escenario.

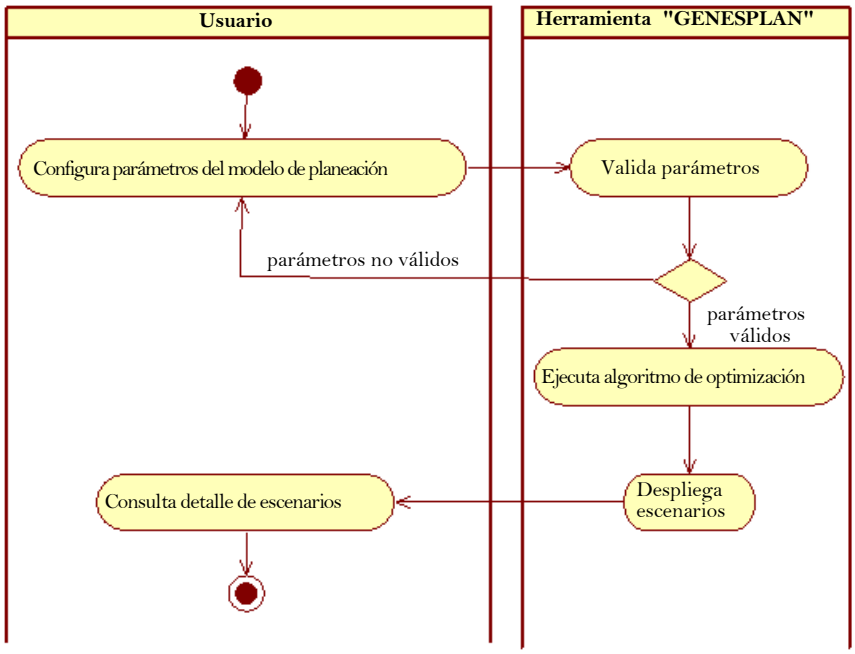


Figura 3.13. Flujo de trabajo para la generación de escenarios usando la herramienta GENESPLAN.

3.5.4. Requerimientos a nivel de usuario: casos de uso

A partir del flujo de trabajo y de las características, se han derivado 3 casos de uso[4], los cuales se representan en el diagrama[5] de la Figura 3.14.

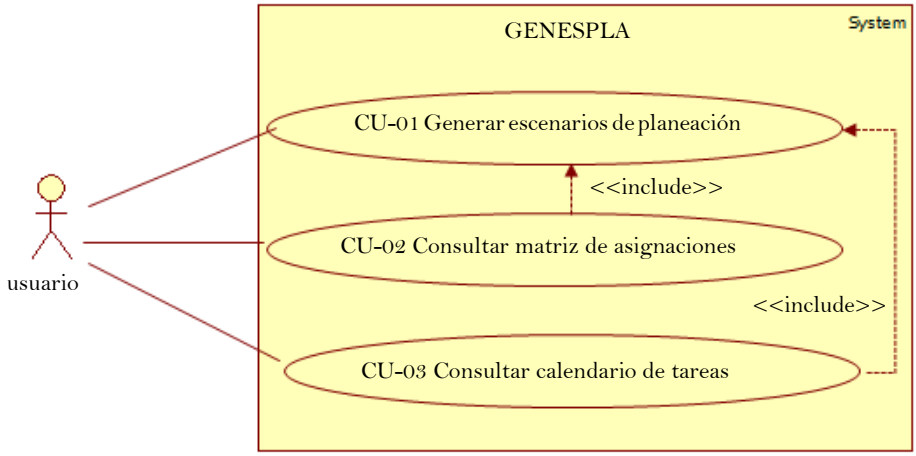


Figura 3.14. Diagrama de casos de uso que soporta la herramienta GENESPLAN.

El caso de uso CU-01(Generar escenarios de planeación) es el primario. Los casos CU-02 y CU-03 presentan una relación de inclusión con CU-01, lo cual significa que este forma parte del flujo de CU-02 y CU-03.

[4] ~~Los casos de uso describen comportamientos~~ funcionales de un sistema como intercambios entre un usuario y ese sistema.
[5] Un diagrama de casos de uso es un diagrama que muestra las relaciones entre actores y casos de uso dentro de un sistema.

Con respecto a los elementos del diagrama de la Figura 3.14, el área rectangular representa la frontera del sistema y el actor es el usuario de la herramienta, quien normalmente puede ser un administrador de proyecto, un líder de proyecto o cualquier otra persona que se dedique a realizar planes de proyecto.

Para conocer con mayor detalle el flujo de los casos de uso soportados por GENESPLAN, en las Tablas 3.4, 3.5 y 3.6 se describen los detalles de CU-01, CU-02 y CU-03, respectivamente.

Identificador y nombre	CU-01 Generar escenarios de planeación
Descripción	Este caso de uso permite generar una lista de escenarios de planeación, a partir de los datos del plan de proyecto.
Actor	Usuario
Precondiciones	1. El sistema muestra la interfaz para generar escenarios de planeación.
Postcondiciones	1. El sistema habrá desplegado una lista de escenarios de planeación.
Flujo principal	1.- El usuario agrega un archivo con los datos de las tareas y de los empleados de un plan de proyecto, selecciona la fecha de inicio del mismo, los posibles días festivos. Luego, presiona el botón para generar los escenarios. 2.- a). El sistema verifica que los datos del archivo de texto tengan la sintaxis correcta. b). Con estos datos genera los escenarios y los despliega en una nueva interfaz (interfaz de despliegue de escenarios). Para cada escenario se muestra: ▪ la duración, ▪ el costo, ▪ el periodo del proyecto, ▪ una opción para consultar la respectiva matriz de asignaciones y, ▪ una opción para consultar el calendario de tareas.
Flujos alternativos	En 2 si hay un error de sintaxis en algunos datos del archivo adjunto: 1. El sistema notifica al usuario con un mensaje. 2. El usuario hace las correcciones necesarias y el flujo retoma en 1.

Tabla 3.4 Documentación del caso de uso CU-01 "Generar escenarios de planeación".

Identificador y nombre	CU-02 Consultar matriz de asignaciones de un escenario de planeación
Descripción	Este caso de uso permite consultar la matriz de asignaciones entre tareas y empleados para un determinado escenario de planeación.
Actor	Usuario
Precondiciones	1. El sistema muestra la interfaz de despliegue de escenarios.
Postcondiciones	1. El sistema habrá desplegado la matriz de asignaciones del escenario seleccionado.

Flujo principal	1. El usuario selecciona la opción "Ver asignaciones" del escenario seleccionado. 2. El sistema despliega una interfaz que contiene las asignaciones entre tareas y empleados para el escenario seleccionado. En dicha matriz se muestra el número de horas que dedican los empleados a cada una de las tareas que les fueron asignadas por el sistema.

Tabla 3.5 Documentación del caso de uso CU-02 "Consultar matriz de asignaciones de un escenario de planeación".

Identificador y nombre	CU-03 Consultar calendario de tareas para un escenario de planeación
Descripción	Este caso de uso permite consultar el calendario de tareas en forma de diagrama de Gantt, para un determinado escenario de planeación.
Actor	Usuario
Precondiciones	1. El sistema despliega la lista de escenarios de planeación.
Postcondiciones	1. El sistema habrá desplegado el calendario de tareas del escenario de planeación seleccionado.
Flujo principal	1. El usuario selecciona la opción "Ver calendario de tareas" del escenario de su preferencia. 2. El sistema despliega una interfaz con el calendario de tareas en forma de diagrama de Gantt para el escenario elegido. En dicho calendario se muestra de manera gráfica la duración de las tareas con sus respectivas dependencias. El usuario puede consultar el calendario en una escala diaria o semanal.

Tabla 3.6. Documentación del caso de uso "Consultar calendario de tareas".

3.5.5. Atributos de calidad

Los atributos de calidad son características medibles que permiten establecer el grado de calidad de un sistema. Ejemplos de atributos de calidad son el desempeño, la seguridad, la confiabilidad, la modificabilidad, la usabilidad, entre otros [Cervantes2016]. En este caso, la modificabilidad (AC-01) se consideró como el principal atributo de calidad de la herramienta GENESPLAN.

La modificabilidad tiene que ver principalmente con la facilidad para introducir cambios en un sistema, cuyos criterios de medición pueden ser: el tiempo que toma hacer un cambio, el número de módulos afectados, entre otros.

En ocasiones los sistemas pueden sufrir cambios y este caso no es la excepción, pues como se menciona en el Capítulo 5 (Conclusiones) en la Sección "Trabajo a futuro", es posible que se agreguen funcionalidades en GENESPLAN. Por ejemplo, se podrían agregar interfaces para capturar directamente los datos de las tareas y de los empleados para un plan de proyecto, se podrían hacer cambios al modelo de planeación, o incluso, agregar un nuevo algoritmo de optimización. Así que el diseño, del cual se hablará más adelante, debe soportar la realización de cambios en la herramienta.

Los atributos de calidad se pueden especificar a través de escenarios [Bass2012]. Un escenario expresa una respuesta medible de un sistema ante un estímulo, en un contexto particular. En la Tabla 3.7. se describe un ejemplo de escenario asociado a la modificabilidad. Los escenarios generalmente se expresan en frases que incluyen distintos elementos: una fuente, un estímulo, un artefacto, entorno y una respuesta [Cervantes2016].

Los atributos calidad forman parte de los drivers (guías) de la arquitectura de un sistema.

Elemento	Descripción
Fuente	Un desarrollador
Estímulo	Se requiere agregar una interfaz gráfica a la herramienta, por ejemplo, una interfaz para configurar los parámetros de un algoritmo de optimización.
Artefacto	Código
Entorno	Durante el desarrollo de una nueva versión de la herramienta.
Respuesta	La interfaz gráfica es agregada, sin modificar el contenido de otras interfaces.

Tabla 3.7. Escenario de atributo de modificabilidad para agregar un algoritmo de optimización.

3.6 El diseño de GENESPLAN

En esta sección se describen los elementos principales del diseño de GENESPLAN, considerando los casos de uso mencionados y el atributo de calidad modificabilidad. Primero se describen las decisiones de diseño que se tomaron y posteriormente se documenta el diseño a través de diagramas UML.

3.6.1. Decisiones de diseño

Para definir el diseño de GENESPLAN, se han tomado las decisiones presentadas en la Tabla 3.8. Las decisiones de diseño permiten satisfacer la mayor parte de los drivers y los requerimientos establecidos. Cada una de las decisiones tiene asociado un concepto de diseño [Cervantes2016], como un patrón de diseño o un framework (marco de trabajo). Los patrones de diseño son soluciones conceptuales a problemas recurrentes de diseño, mientras que los frameworks son soluciones a nivel de código que se enfocan en resolver problemas particulares como la persistencia de datos, aspectos de presentación de un sistema o transacciones.

Decisión de diseño	Concepto de diseño	Drivers que satisface	Justificación
Usar jMetal [Durillo2011] en vez de codificar los algoritmos "desde cero". Se describe en la sección 3.6.1.1.	Framework	CU-01 y AC-01	Este framework cuenta con una implementación de algoritmos de optimización que han sido utilizados en otros trabajos. Además, la flexibilidad en su arquitectura es una de sus principales ventajas, ya que permite integrar algún componente que se necesite en el proceso de optimización, como un algoritmo, un operador genético, o una nueva

			especificación del problema de planeación de proyectos de software.
Utilizar Swing (Java)	Librería gráfica / framework	CU-01, CU-02 y CU-03	Se requieren interfaces gráficas de usuario para ejecutar los casos de uso mencionados anteriormente, y la librería Swing proporciona los componentes necesarios para tal propósito. Además es sencilla de usar ya que permite agregar componentes de manera rápida, simple y dinámica.
Utilizar "SwiftGantt"	Librería gráfica / framework	CU-03	Es una librería que permite crear diagramas de gantt, necesaria para desarrollar parte del caso de uso CU-03 (Consultar el calendario de tareas de un escenario de planeación).
Utilizar el patrón "Capas"	Patrón de diseño	AC-01	Este patrón permite agrupar funcionalidades específicas del sistema y facilita el desarrollo y la modificabilidad, por ejemplo, sería posible agregar nuevas interfaces de usuario, si se requieren.

Tabla 3.8. Decisiones de diseño para la implementación de GENESPLAN.

3.6.1.1 El framework jMetal

Una parte fundamental en el proceso de generación de escenarios de planeación, es el uso de un algoritmo de optimización para generar estimaciones de tiempo y costo. Particularmente, la optimización es un proceso complejo que engloba un conjunto de módulos y componentes relacionados entre sí, tales como operadores genéticos (cruce, mutación y selección), componentes para las posibles representaciones de una solución, componentes para indicadores de calidad, interfaces para agregar nuevos algoritmos, interfaces para especificar problemas de optimización, entre otros componentes. Al respecto, existe un framework llamado jMetal [Durillo2011], el cual cuenta con una arquitectura sofisticada para llevar a cabo el proceso de optimización y que permite adaptar fácilmente componentes asociados con el problema que se desea resolver.

jMetal es un framework de código abierto que está basado en el lenguaje de programación Java y que surgió por la necesidad de resolver problemas de optimización con uno o más objetivos y por las limitaciones de diseño que presentaban otros frameworks, por ejemplo; el código no era reutilizable. A raíz de estos problemas, los autores propusieron el desarrollo de un software "desde cero", el cual tuvo los siguientes objetivos de diseño:

a. Simplicidad y facilidad de uso. Cada componente está centrado en una función específica y sus operaciones tenían que ser entendibles y fáciles de usar.

b. Flexibilidad. Es posible incorporar mecanismos sencillos para ejecutar los algoritmos bajo diferentes configuraciones de parámetros, tanto del propio algoritmo, como del problema a resolver. Además los cambios en los operadores y la representación de las soluciones son mínimos.

c. Portabilidad. La ejecución de los algoritmos es independiente de las arquitecturas de las máquinas y del sistema operativo.

d. Extensibilidad. Es posible agregar fácilmente nuevos algoritmos, operadores y problemas.

Mientras tanto, las principales características que tiene jMetal son las siguientes:

1. Implementación de clásicos y modernos algoritmos de optimización multiobjetivos como NSGA-II, SPEA2, PAES, PESA-II, MOCell, GDE3, IBEA, SMS-EMOA, SMPSO, entre otros.

2. Versiones de ejecución en paralelo de algunos de los algoritmos, entre ellos NSGA-II.

3. Una gran variedad de problemas de prueba, con y sin restricciones.

4. Implementación de una variedad de indicadores de calidad[6] como hipervolumen, spread, distancia generacional, epsilon, etc.

5. Diferentes formas de representación de variables: binaria, real, entera, permutación.

En cuanto a su arquitectura, jMetal se compone de 4 paquetes principales:

1. Paquete *core*. Contiene las clases donde se especifican los operadores genéticos (cruce, mutación y selección) utilizados por un algoritmo de optimización, las posibles representaciones de una solución, los indicadores de calidad para las soluciones, etc.

2. Paquete *problem*. Contiene las clases donde se especifican los problemas de optimización que se desean resolver. Los problemas pueden ser multiobjetivo o de un sólo objetivo. Estas clases implementan una interfaz llamada "Problem".

3. Paquete *algorithm*. Contiene las clases donde se especifican los algoritmos de optimización, con su respectivo flujo. Estas clases implementan una interfaz llamada "Algorithm".

4. Paquete *exec*. Contiene las clases para ejecutar los algoritmos de optimización. Estas clases implementan una interfaz llamada "AlgorithmRunner".

En la Figura 3.15 se presenta un diagrama de paquetes UML que resume la arquitectura de jMetal, en el cual se identifican las relaciones entre los componentes principales de los paquetes mencionados.

Un algoritmo tiene asociada una clase principal donde se lleva a cabo su ejecución (por ejemplo NSGAII se ejecuta en NSGAIIRunner, SPEA2 se ejecuta en SPEA2Runner, etc). Para el algoritmo seleccionado, se especifica el problema a resolver en una clase que implementa la interfaz "Problem". En esta clase se especifican los objetivos de optimización y las restricciones que son evaluados para cada posible solución. Las soluciones deben tener alguna forma de representación, por ejemplo, binaria (que se especifica en la clase "BinarySolution"), entera (se especifica en la clase "IntegerSolution"), o alguna otra representación. Además, para el algoritmo que se va a ejecutar también se deben especificar los parámetros de los operadores genéticos: cruce, mutación y selección.

[6] Los indicadores de calidad permiten evaluar el desempeño de los algoritmos de optimización.

La estructura de jMetal por paquetes permite añadir fácilmente componentes asociados con algoritmos o problemas de optimización, utilizando las interfaces correspondientes.

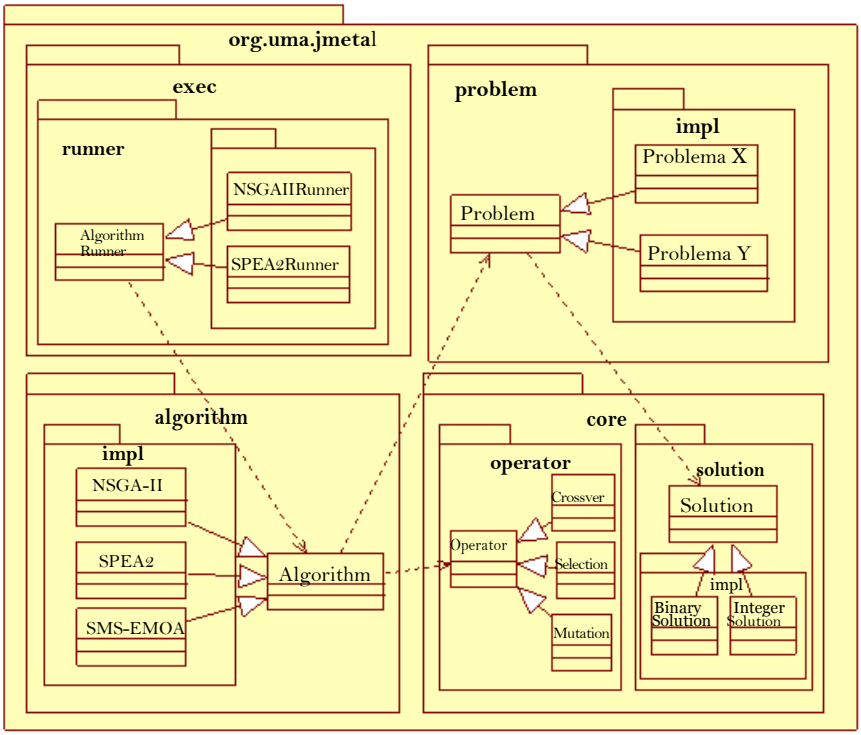


Figura 3.15. Estructura general de jMetal [Durillo2011].

3.6.1.2. SwiftGantt

Uno de los requerimientos que se propusieron para el desarrollo de GENESPLAN es el despliegue del calendario de tareas para un determinado escenario de planeación, y por simplicidad, se ha optado por utilizar un framework o biblioteca para cumplir con este requerimiento. Actualmente existe una variedad de frameworks de Java que permiten cubrir esta necesidad, uno de ellos SwiftGantt[7]. SwiftGantt es una biblioteca de Java de código abierto que permite crear diagramas de Gantt para una aplicación de escritorio. Los principales objetivos de diseño de esta librería son la flexibilidad, la compatibilidad y la facilidad de uso.

Las principales características de SwiftGantt son las siguientes:

- El soporte de distintas unidades de tiempo (horas, días, semanas, meses, años).
- La exportación de un diagrama como un archivo de imagen.
- El soporte de múltiples lenguajes (inglés, chino, japonés).

Para crear un diagrama, es necesario definir un modelo (GanttModel) en el cual se especifiquen los datos de las tareas de un proyecto como su duración, su descripción y sus dependencias. Una de las ventajas que ofrece esta librería es que permite especificar los detalles de un modelo de una

[7] http://swiftgantt.sourceforge.net/en/index.html.

forma muy simple y sencilla, con respecto a otras librerías. Además, permite personalizar los elementos de un diagrama como colores y tamaños de barras, distintas escalas de tiempo, entre otros aspectos. En la Figura 3.16 se presenta un ejemplo de diagrama de Gantt creado con esta librería.

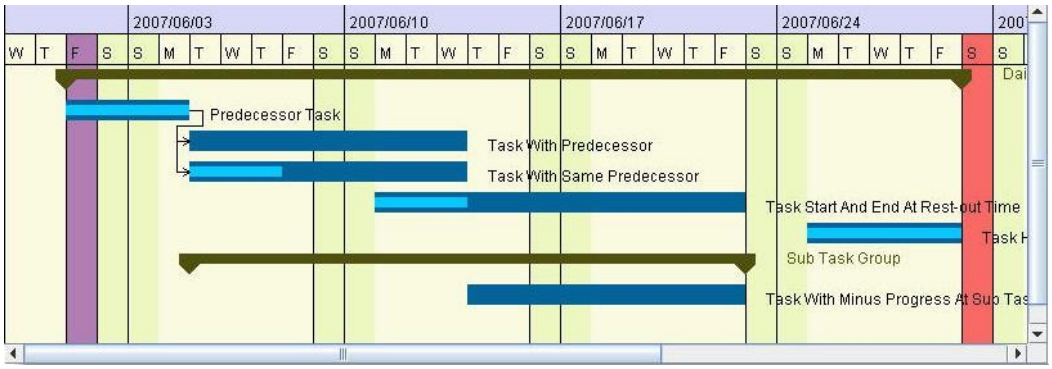


Figura 3.16. Calendario de un proyecto creado con SwiftGantt.

3.6.2 La arquitectura de GENESPLAN

Desde el punto de vista de diseño de software, documentar la arquitectura de un sistema consiste básicamente en describir los subsistemas y las relaciones que existen entre sus componentes [Cervantes2016]. Anteriormente se describió un resumen de la arquitectura de jMetal, donde el sistema se divide en 4 paquetes que contienen las clases e interfaces y estas tienen asociada una función específica durante el proceso de optimización. Ahora se describe la arquitectura de la herramienta para generar escenarios de planeación en su conjunto, la cual incluye el modelo de planeación y el framework jMetal.

Una forma de documentar la estructura de un sistema de software es a través de vistas. Una vista permite modelar gráficamente una parte o la totalidad de un sistema desde alguna perspectiva y normalmente es representada a través de un diagrama [Cervantes2016]. Existen distintos enfoques para hacer la documentación de una vista, entre ellos, el enfoque "4+1 view" [Kruchten1995]. Este enfoque sugiere 4 vistas principales: una vista lógica, física, de proceso, de desarrollo y una adicional que es la vista de casos de uso (de ahí el "+1").

Para documentar la arquitectura de GENESPLAN se ha utilizado exclusivamente la vista lógica que forma parte del enfoque "4+1 view". Primero se presenta la vista general de la herramienta desde una perspectiva estática y posteriormente la vista que soporta el caso de uso primario CU-01, desde una perspectiva dinámica. La vista general, ha sido representada mediante un diagrama de paquetes, mientras que la vista del caso de uso primario ha sido representada mediante un diagrama de secuencia.

3.6.2.1. Vista general de GENESPLAN

GENESPLAN se compone de 2 capas lógicas principales:

1. Capa de presentación
2. Capa de lógica de aplicación

En la Figura 3.17 se presenta el contenido de ambas capas y las relaciones entre sus elementos.

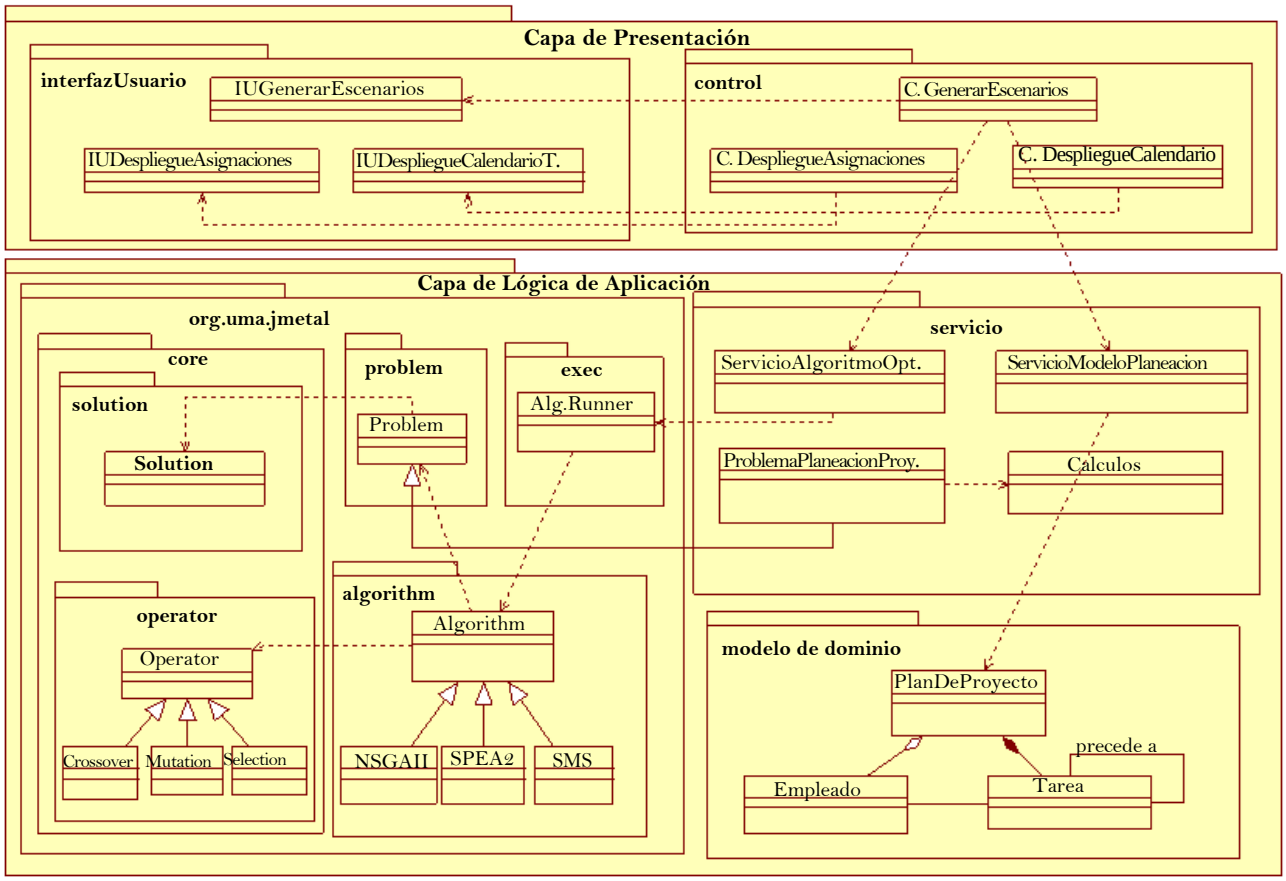


Figura 3.17. Diagrama de paquetes que representa la vista lógica general de GENESPLAN.

En la capa de presentación se agrupan los componentes que permiten al usuario capturar los datos para la generación de escenarios de planeación y consultar información de estos. Se compone de 2 grupos de módulos: "interfazUsuario" y "control". En la subcapa "interfazUsuario" se agrupan los módulos asociados con las interfaces a través de las cuales interactúa el usuario con la herramienta. El grupo "control" está formado por módulos en los que se describe el flujo asociado con los casos de uso descritos anteriormente.

Mientras tanto, en la capa de lógica de aplicación se agrupan los componentes que se encargan del proceso de generación de escenarios. A su vez, esta se compone de una subcapa de servicios y una subcapa asociada con las entidades del modelo de dominio del problema (Tarea y Empleado). La capa de servicios contiene los componentes que se encargan de tareas específicas como la generación del modelo de planeación, la comunicación con los elementos de jMetal para obtener las soluciones, la especificación del problema y los cálculos del costo y la duración de proyectos para las posibles soluciones.

La arquitectura fue diseñada con el objetivo de agregar y/o reemplazar componentes en versiones futuras de la herramienta, sin afectar a otros; por ejemplo, se puede configurar otro algoritmo de optimización si así se requiere, se puede agregar una nueva interfaz de usuario, o bien, modificar la especificación del problema de planeación de proyectos de software.

Los elementos principales de las capas de GENESPLAN se describen en la Tabla 3.9. Para cada uno de estos se describe la responsabilidad asignada.

Componente	Capa a la que pertenece	Responsabilidad o función
IUGenerarEscenarios	Presentación	Por medio de esta interfaz es posible agregar los parámetros necesarios para generar los escenarios de planeación. Estos parámetros son los datos de las tareas y los empleados, la fecha de inicio del proyecto y posibles días festivos.
IUDespliegueEscenarios	Presentación	Esta interfaz se encarga de desplegar una lista de escenarios de planeación. Para cada escenario se presenta la duración, el costo, el periodo de ejecución del proyecto, una opción para consultar la matriz entre tareas y empleados y otra opción para consultar el calendario de tareas.
IUDespliegueAsignaciones	Presentación	Se encarga de presentar la cantidad de horas que dedicarán los empleados a las tareas que les fueron asignadas por uno de los algoritmos de optimización.
IUDespliegueCalendarioTareas	Presentación	Se encarga de presentar un diagrama de Gantt generado con SwiftGantt, en el cual se muestra de manera gráfica la duración de las tareas y las dependencias entre ellas.
ControlGenerarEscenarios, ControlDespliegueAsignaciones y ControlDespliegueCalendario Tareas	Presentación	Se encargan de llevar el flujo de control de los casos de uso y preparar la información que se presentarán en las interfaces de usuario.
ServicioModeloPlaneación	Lógica de aplicación	Se encarga de efectuar las operaciones para construir las entidades del modelo de planeación, cuyos datos son utilizados en el proceso de generación de escenarios. Estas operaciones incluyen la lectura, el parseo y validación de los datos.
ServicioAlgoritmoOptimizacion	Lógica de aplicación	Se encarga de la comunicación con los componentes principales de jMetal (Algoritmo, operadores, problema) para llevar a cabo el proceso de optimización.
PlaneaciónProyectosSoftware	Lógica de aplicación	Contiene la especificación del problema a optimizar, y se encarga de evaluar el costo y la duración y las restricciones para las posibles soluciones.
Calculos	Lógica de aplicación	En esta clase se especifican las operaciones aritméticas para calcular el costo y la duración de las posibles soluciones durante el proceso de optimización.
PlanProyecto, Tarea y Empleado	Lógica de	Estas son las clases relativas al modelo de

	aplicación	dominio, en las cuales se especifican los datos necesarios para generar los cálculos. Un plan de proyecto está formado por un conjunto de tareas y un conjunto de empleados, tal como en el modelo de planeación de la Figura 3.1 (Sección 3.2.2).
Algorithm, Operator, Problem, Solution	Lógica de Aplicación	Son las interfaces principales de jMetal mediante las cuales se lleva a cabo el proceso de optimización. Un algoritmo necesita operadores genéticos de cruce, mutación y selección; y necesita la especificación de un problema. A su vez un problema requiere de una representación para las soluciones.

Tabla 3.9. Elementos principales de la arquitectura de GENESPLAN.

3.6.2.2 Vista del caso de uso "Generar escenarios de planeación"

Ahora se presenta un diagrama de secuencia en el cual interactúan en tiempo de ejecución los componentes principales que soportan el flujo del caso de uso primario "Generar escenarios de planeación" (ver Figura 3.18).

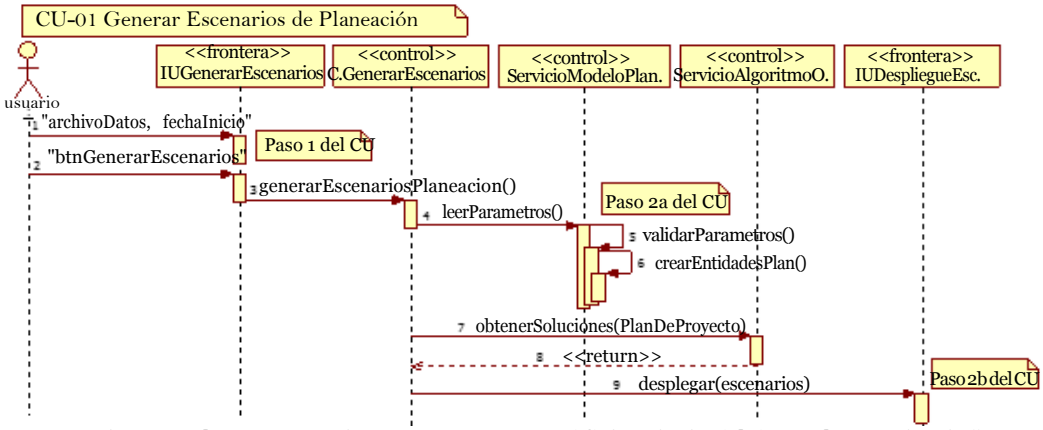


Figura 3.18. Diagrama de comportamiento que representa el flujo principal del caso de uso primario "Generar escenarios de planeación".

El flujo inicia cuando el usuario de la herramienta (actor) agrega los parámetros del plan de proyecto a través de una instancia de la interfaz "IUGenerarEscenarios" (1 y 2). Este objeto a su vez se comunica con una instancia del componente "ControlGenerarEscenarios" para solicitar que se ejecute el proceso de generación de escenarios, el cual incluye pasos intermedios (3). El objeto de control se apoya de un servicio ("ServicioPlaneacionModelo") para solicitar la lectura (4), el parseo y la verificación de los datos de las tareas y de los empleados especificados en un archivo de texto (5); si los datos tienen la sintaxis correcta, se crean las estructuras donde se almacenan los valores de las características de las entidades del modelo (6). Posteriormente el objeto de control se comunica con una instancia de "ServicioAlgoritmoOptimización" a la cual le solicita obtener las soluciones usando los datos del modelo (7). Aunque ya no se ilustra en el diagrama, el paso intermedio es la comunicación con las instancias de los componentes de jMetal, quienes se encargan de llevar la secuencia del proceso de optimización. Una vez obtenidas las soluciones (8), estas se despliegan a

través de una instancia la interfaz de usuario "IUDespliegueEscenarios" (9), finalizando de esta manera el flujo principal del caso de uso.

3.7. Interfaces gráficas del prototipo de GENESPLAN

El flujo de trabajo para la generación de escenarios de planeación requiere de interfaces gráficas para capturar datos por parte del usuario y para presentar información sobre los escenarios obtenidos. Al respecto, se desarrollaron las siguientes interfaces las cuales están asociadas con alguno de los casos de uso:

1. 2 Interfaces para el caso de uso "Generar escenarios de planeación", una es para capturar datos y la otra para desplegar los escenarios de planeación obtenidos.

2. Interfaz de despliegue de asignaciones (para el caso de uso "Consultar matriz de asignaciones de un escenario de planeación").

3. Interfaz de despliegue de calendario de tareas (para el caso de uso "Consultar calendario de tareas de un escenario de planeación").

3.7.1. Interfaces gráficas para el caso de uso "Generar escenarios de planeación"

La primera de estas interfaces permite capturar los parámetros necesarios para generar escenarios de planeación y se divide en 3 secciones principales (ver Figura 3.19).

Figura 3.19. Interfaz gráfica para generar escenarios de planeación.

En la parte superior izquierda es posible adjuntar un archivo de texto con los datos de las tareas y los empleados del proyecto. En el Capítulo 4 (Sección 4.1), se describe la sintaxis y el significado de los datos de un proyecto. En la parte superior derecha se dispone de un calendario para seleccionar la fecha de inicio del proyecto. En la parte inferior se dispone de un calendario para

seleccionar días festivos que puedan ocurrir durante el proyecto. Una vez configurados estos datos, el usuario puede generar los escenarios de planeación. La configuración de estos datos corresponde al paso inicial del caso de uso CU-01 (Generar escenarios de planeación).

La segunda interfaz permite desplegar la lista de escenarios obtenidos (ver Figura 3.20) y corresponde al segundo paso del caso de uso CU-01.

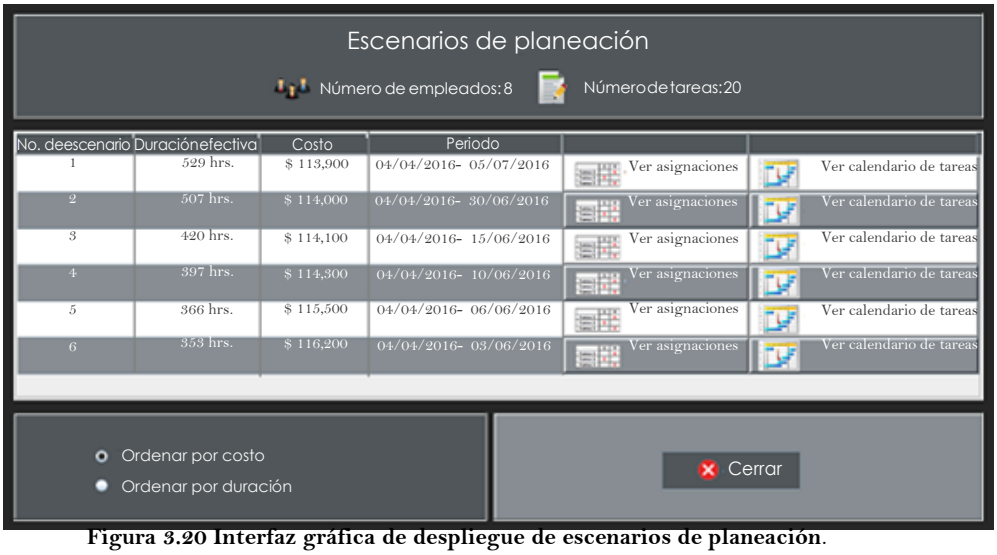


Figura 3.20 Interfaz gráfica de despliegue de escenarios de planeación.

Los datos presentados para cada escenario son los siguientes:

1. La duración del proyecto.
2. El costo del proyecto.
3. El periodo de tiempo estimado para el proyecto.
4. Una opción para consultar la relación de asignaciones entre tareas y empleados.
5. Una opción para consultar el calendario de tareas del proyecto.

Cabe señalar que el escenario con la duración más alta presenta el costo más bajo y viceversa. Esto se debe a la característica de no dominancia que presentan las soluciones generadas por un algoritmo de optimización, es decir, ninguna solución es superior o inferior al resto con respecto a la duración y al costo de un proyecto (aún después de haber eliminado el conflicto de sobrecarga en soluciones que la presentan).

Con el despliegue de esta interfaz concluye el flujo del caso de uso primario "Generar escenarios de planeación".

3.7.2. Interfaz para el despliegue de asignaciones

En esta interfaz presenta la relación entre tareas y empleados para un determinado escenario de planeación (ver Figura 3.21). Esta relación corresponde al número de horas planeadas que los empleados dedicarán a las tareas asociadas. También, para cada empleado se muestra el total de horas que dedicará al proyecto. Debido a que los empleados pueden poseer distintos factores de productividad, el total de horas entre los empleados puede ser variable. El despliegue de esta interfaz permite concluir el flujo del caso de uso CU-02 ("Consultar matriz de asignaciones de un escenario de planeación").

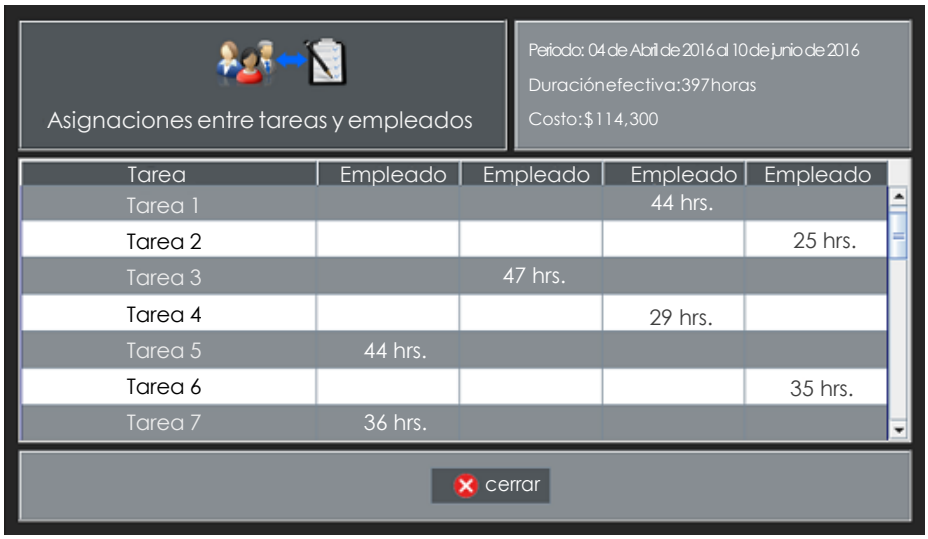


Figura 3.21. Interfaz que despliega las asignaciones entre tareas y empleados para un determinado escenario de planeación.

3.7.3. Interfaz de despliegue de calendario de tareas

Esta interfaz permite visualizar gráficamente la duración planeada de las tareas, así como las dependencias que hay entre ellas (ver Figura 3.22). Las barras de las tareas se presentan con un color diferente, de tal manera que se puedan identificar fácilmente las que fueron asignadas a cada empleado. El usuario tiene la oportunidad de visualizar el calendario de tareas en una escala temporal, ya sea diaria o semanal.

Figura 3.22. Interfaz que despliega el calendario de tareas asociado a un escenario de planeación.

Para dibujar el diagrama por medio de SwiftGantt en la interfaz gráfica, fue necesario definir un modelo de tareas con sus respectivos datos, tales como su nombre, el empleado asignado y sus dependencias. En el diagrama es posible identificar fácilmente la fecha de inicio y de finalización del proyecto y por convención, los días sábado y domingo no son laborales.

Con el despliegue de esta interfaz gráfica, se concluye el flujo del caso de uso CU-03 "Consultar calendario de tareas de un escenario de planeación").

Las interfaces gráficas y los componentes utilizados en el proceso de generación de escenarios se desarrollaron utilizando el lenguaje de programación Java y por el momento, la herramienta opera en un entorno local. Es posible realizar modificaciones en un futuro con respecto al modelo de planeación, el diseño y el prototipo de la herramienta, de lo cual se hablará en el capítulo 5 del presente documento.

Capítulo 4

Experimentos y resultados

En este capítulo se presentan algunos de los resultados obtenidos de los experimentos que se llevaron a cabo utilizando la herramienta propuesta para generar escenarios de planeación (GENESPLAN). Para realizar los experimentos se definieron casos de prueba en los cuales se varían los parámetros del modelo de planeación (descrito en el capítulo anterior), tales como el número de tareas y empleados, y sus respectivas características. El objetivo de estos experimentos fue corroborar que la herramienta funciona adecuadamente, que la propuesta es sensata y que por medio de ésta es posible obtener soluciones en términos de estimación de costo y duración para proyectos de desarrollo de software.

4.1 Diseño de casos de prueba

Los casos de prueba para realizar los experimentos fueron diseñados con el *generador de casos*, un programa desarrollado por Alba y Chicano [Alba2007] que permite simular proyectos de software, cuyos parámetros son generados aleatoriamente. Los elementos principales de un caso de prueba son: empleados, tareas y el grafo de precedencia de tareas (GPT). Estos elementos, a su vez, tienen asociados diversos parámetros, los cuales corresponden a las características que presentan en el modelo de planeación. Los valores de los parámetros son generados aleatoriamente a partir de una muestra de una distribución de probabilidad; una distribución normal para el caso del grafo de precedencia de tareas, los costos de los empleados y los esfuerzos estimados de las tareas; y una distribución uniforme entera para el caso del número de tareas, el número de empleados y los factores de productividad. Los datos generados siguen una sintaxis particular (ver Anexo B).

Para realizar los experimentos se definieron 4 grupos de casos de prueba:

1. *Grupo 1: variación en el nivel de productividad de los empleados.* Los empleados pasan por distintos niveles de productividad (que van desde muy bajo hasta alto), los cuales se determinaron con base en el valor de sus factores de productividad en las disciplinas correspondientes.

2. *Grupo 2: variación en la cantidad de empleados.* Se formaron equipos con 4, 8, 12 y 16 empleados, permaneciendo fijos el nivel de productividad y el costo.

3. *Grupo 3: los empleados presentan características mixtas.* Se formaron combinaciones de empleados con 2 niveles de productividad, alto y bajo, en donde el costo es proporcional al nivel de productividad que poseen. Además, se tiene la misma cantidad de perfiles por disciplina en el ciclo de vida de un proyecto.

4. *Grupo 4: datos aleatorios.* En estos casos de prueba, los factores de productividad y el costo de los empleados son valores aleatorios. Además, los empleados tienen perfiles mixtos, es decir, poseen factores de productividad mayores a cero en múltiples disciplinas.

Para ejecutar las pruebas se generaron aleatoriamente 3 grafos de precedencia de tareas: el primero consta de 20 tareas, el segundo de 30 tareas y el tercero de 40 tareas. Se decidió que cada

proyecto tuviera asociadas 4 disciplinas del ciclo de vida, y por esta razón, que el número de empleados fuera un múltiplo de 4, con el fin de tener la misma cantidad de perfiles por disciplina, por ejemplo, con 8 empleados se tendrían 2 analistas de requerimientos, 2 arquitectos de software, 2 programadores y 2 testers. El número de empleados en estos grupos de casos de prueba fueron 4, 8, 12 y 16, los cuales se formaron de la siguiente manera: el equipo de 4 empleados es un subconjunto del equipo de 8 empleados, el cual a su vez es un subconjunto del equipo de 12 y finalmente, el equipo de 12 empleados forma parte del equipo de 16 empleados. En los primeros 3 grupos de casos de prueba, los empleados se especializan en alguna de las disciplinas, mientras que en el último caso son especialistas en múltiples disciplinas.

Con el fin de obtener resultados estadísticos más confiables, cada algoritmo de optimización se ejecutó 30 veces para resolver cada caso de prueba. En cada ejecución se obtienen 2 soluciones extremas, de las cuales una presenta la mayor duración y el menor costo, y la segunda presenta la menor duración y el mayor costo. En cada caso de prueba, los valores máximos y mínimos tanto del costo como de la duración se promedian, de tal manera que se obtiene un costo promedio mínimo y un costo promedio máximo, y del mismo modo, una duración promedio mínima y una duración promedio máxima.

Los resultados de la resolución de los casos de prueba se reportan en 2 tablas, de las cuales una contiene la duración (efectiva) promedio mínima y máxima, mientras que la segunda tabla contiene los resultados relativos al costo promedio mínimo y máximo. Los resultados se presentan con los algoritmos de optimización NSGA-II [Deb2002], SPEA2 [Zitzler2001] y SMS-EMOA [Beume2007], con el fin de saber si el comportamiento de la duración y el costo es congruente en cada grupo de casos de prueba, es decir, si la duración y el costo obtenidos se aproximan a los esperados, de acuerdo a las características de las tareas y de los empleados. Es importante recordar que uno de los pasos del proceso de optimización, es la ejecución del algoritmo propuesto para eliminar el conflicto de sobrecarga en el proyecto en las soluciones que lo presentan. Al resolverse este conflicto, el proyecto adquiere una nueva duración, pero el costo se mantiene igual.

Finalmente, se presenta una tabla con el promedio de soluciones factibles (escenarios de planeación) obtenidas, tomando en cuenta las 30 ejecuciones de los algoritmos.

4.2 Variación en el nivel de productividad de los empleados

El presente grupo de casos de prueba tuvo como objetivo corroborar si las soluciones son sensatas con respecto a la duración y el costo de un proyecto, al variar el nivel de productividad de los empleados. El resultado esperado era que la duración de un proyecto tuviera una disminución al incrementar el nivel de productividad de los empleados y viceversa.

En este grupo de casos de prueba, los algoritmos de optimización fueron ejecutados bajo las siguientes condiciones:

- El costo por hora es el mismo para todos los empleados ($100), sin importar su nivel de productividad.

- Se generaron aleatoriamente factores de productividad y se clasificaron de acuerdo a los siguientes niveles:

    1. Nivel muy bajo, cuando los factores de productividad están entre 0% y 24%.
    2. Nivel bajo, cuando los factores de productividad están entre 25% y 49%.
    3. Nivel medio, cuando los factores de productividad están entre 50% y 74%.

4. Nivel alto, cuando los factores de productividad están entre 75% y 99%[8].

En la práctica, los factores de productividad pueden estar asociados con el nivel de habilidad o experiencia que pueden presentar los empleados en diversas disciplinas en el ciclo de vida de un proyecto. De hecho, los niveles de productividad mencionados previamente se pusieron como ejemplo con el fin de asociarlos con los 4 niveles de habilidad para empleados sugeridos en [García2014] (principiante, junior, senior y experto), aunque es posible establecer una clasificación distinta de niveles de productividad. Bajo la clasificación anterior, un empleado que posee un nivel de productividad muy bajo puede ser considerado como un empleado de nivel principiante, un empleado con nivel de productividad bajo puede ser considerado como un empleado de nivel junior, un empleado con nivel de productividad medio puede corresponder a un empleado de nivel senior y un empleado con nivel de productividad alto puede representar un nivel experto. También, por simplicidad, el límite inferior y superior de los niveles de productividad se asignaron de manera uniforme, pero pueden establecerse distintos intervalos. El número de niveles de productividad puede depender de la clasificación de empleados en diversas organizaciones.

En la Tabla 4.1 se presenta la duración promedio mínima y la duración promedio máxima (en horas) obtenidas para los casos de prueba con 30 tareas y 8 empleados al variar el nivel de productividad.

	NSGA-II		**SPEA2**		**SMS-EMOA**	
Nivel de productividad de los empleados	**Duración promedio mínima**	**Duración promedio máxima**	**Duración promedio mínima**	**Duración promedio máxima**	**Duración promedio mínima**	**Duración promedio máxima**
Muy bajo	1,355	1,644	1,367	1,658	1,354	1,653
Bajo	684	823	675	835	670	827
Medio	388	466	387	452	386	448
Alto	281	325	281	336	280	323

Tabla 4.1. Duración promedio mínima y duración promedio máxima en horas de un proyecto con 30 tareas y 8 empleados, al variar el nivel de productividad.

Los resultados de la tabla anterior indican que al incrementar el nivel de productividad de los empleados, la duración promedio mínima y la duración promedio máxima disminuyeron. Cuando los empleados poseen un nivel de productividad relativamente bajo, les toma un tiempo mayor desarrollar las tareas y, por consecuencia, se incrementa la duración del proyecto. Por el contrario, cuando los empleados poseen un nivel de productividad relativamente alto, dada su experiencia, les toma un tiempo menor desarrollar las mismas tareas del proyecto. Por lo tanto, se presenta una correlación inversa entre la productividad de los empleados y la duración de las tareas del proyecto. Un comportamiento similar se obtuvo en los casos con 20 y 40 tareas.

De la tabla anterior se puede observar que cuando los empleados presentaron un nivel de productividad muy bajo, la duración promedio mínima fue 0.87% menor utilizando el algoritmo NSGA-II y 0.95% menor con SMS-EMOA, con respecto a SPEA2; mientras que la duración promedio máxima fue menor en un 0.84% con NSGA-II y 0.3% con SMS-EMOA, también con respecto a SPEA2. Cuando los empleados presentaron un nivel bajo, la duración promedio mínima fue menor en 1.31%

[8] El valor máximo posible en el generador de casos de Alba y Chicano es 99, aunque es posible agregar manualmente el valor 100.

utilizando SPEA2 y 2.04% utilizando SMS-EMOA, con respecto a NSGA-II; mientras que la duración promedio máxima fue 1.43% menor con NSGA-II y 0.95% menor con SMS-EMOA, con respecto a SPEA2. Con un nivel de productividad medio, la duración promedio mínima fue 0.25% menor con SPEA2 y 0.51% menor con SMS-EMOA, con respecto a la obtenida con NSGA-II; mientras que la duración promedio máxima fue 3% menor con SPEA2 y 3.86% menor con SMS-EMOA, con respecto NSGA-II. Finalmente, cuando los empleados presentaron un nivel de productividad alto, la duración promedio mínima fue 0.35% menor con SMS-EMOA con respecto a NSGA-II y SPEA2; mientras que la duración promedio máxima fue 3.27% menor con NSGA-II y 3.87% menor con SMS-EMOA, con respecto a SPEA2.

Como se puede notar, en este grupo de casos de prueba SMS-EMOA encontró soluciones que en promedio presentan una duración menor con respecto a las encontradas por NSGA-II y SPEA2, aunque las diferencias en porcentaje fueron relativamente pequeñas.

Por otra parte, el costo promedio mínimo y el costo promedio máximo del proyecto presentaron una disminución cuando se incrementó el costo de los empleados, según su nivel de productividad (ver Tabla 4.2).

	NSGA-II		**SPEA2**		**SMS-EMOA**	
costo de los empleados	**Costo promedio mínimo**	**Costo promedio máximo**	**Costo promedio mínimo**	**Costo promedio máximo**	**Costo promedio mínimo**	**Costo promedio máximo**
Muy bajo	580,063	596,397	580,303	595,293	579,953	594,770
Bajo	257,317	258,760	257,387	258,827	257,360	258,880
Medio	145,353	147,070	145,327	147,160	145,417	147,150
Alto	103,827	104,340	103,840	104,353	103,847	104,390

Tabla 4.2. Costo promedio mínimo y costo promedio máximo de un proyecto con 30 tareas y 8 empleados, al variar el nivel de productividad y dejar el costo fijo.

Como se especificó en el modelo de planeación, el costo de cada tarea del proyecto es el producto entre su duración (en horas) y el costo por hora del empleado asignado, así que cuando la duración de las tareas disminuye (al aumentar el nivel de productividad) y el costo por hora es el mismo para todos los empleados, el costo de las tareas también disminuye y, por consecuencia, el costo del proyecto también disminuye.

Como se puede observar en la tabla 4.2, cuando los empleados presentaron un nivel de productividad muy bajo, el costo promedio mínimo fue 0.04% menor utilizando NSGA-II y 0.06% menor con SMS-EMOA, con respecto al obtenido con SPEA2; mientras que el costo promedio máximo fue 0.18% menor con SPEA2 y 0.27% menor con SMS-EMOA, con respecto al obtenido con NSGA-II. Cuando los empleados presentaron un nivel de productividad bajo, el costo promedio mínimo fue 0.02% menor con NSGA-II y 0.01% menor con SMS-EMOA, con respecto al obtenido con SPEA2; y el costo promedio máximo fue 0.04% menor con NSGA-II y 0.02% menor con SPEA2, con respecto al obtenido con SMS-EMOA. Con un nivel de productividad medio, el costo promedio mínimo fue 0.04% menor con NSGA-II y 0.06% menor con SPEA2, con respecto al obtenido con SMS-EMOA; mientras que el costo promedio máximo fue 0.06% menor con NSGA-II y 0.006% menor con SMS-EMOA, con respecto al obtenido con SPEA2. Finalmente, cuando los empleados presentaron un nivel de productividad alto, el costo promedio mínimo fue 0.01% menor con NSGA-II

y 0.006% menor con SPEA2, con respecto al obtenido con SMS-EMOA; mientras que el costo promedio máximo fue 0.04% menor con NSGA-II y 0.03% menor con SPEA2, con respecto al obtenido con SMS-EMOA. En estos casos particulares, NSGA-II encontró soluciones que en promedio presentaron un menor costo, con respecto a las soluciones arrojadas por SPEA2 y por SMS-EMOA. En la Tabla 4.3 se presenta el promedio de soluciones obtenido con cada algoritmo.

Algoritmo	Nivel de productividad promedio de los empleados			
	Muy Bajo	Bajo	Medio	Alto
NSGA-II	7	8	4	5
SPEA2	6	8	5	6
SMS-EMOA	7	8	5	5

Tabla 4.3. Promedio de soluciones al resolver el grupo de casos en el cual hay variación en el nivel de productividad de los empleados.

En este grupo de casos de prueba es más favorable disponer de empleados con mayor nivel de productividad, pues ayudaría a disminuir la duración efectiva de un proyecto y el costo, sin embargo, en la práctica difícilmente el costo es el mismo para todos los empleados en un proyecto.

4.3 Variación en el número de empleados

El objetivo principal de este grupo de casos de prueba fue observar la influencia del número de empleados en la duración y el costo de un proyecto. En algunos trabajos como [Chicano2011] o [Chicano2012], el incremento de empleados regularmente permitió disminuir la duración de un proyecto (aunque el costo se incrementó), por lo cual en estas pruebas la intención fue verificar si sucedía algo similar. Para ello, se hizo variación en la cantidad de empleados (4, 8, 12 y 16), mientras el costo de ellos fue el mismo para todos (como en el anterior grupo de casos), al igual que el nivel de productividad (aunque hubo ligeras variaciones en los factores de productividad). Al tener 4 disciplinas en el ciclo de vida de cada proyecto, los perfiles se distribuyeron de manera uniforme. De esta manera, con 4 empleados se tiene un perfil por disciplina (por ejemplo: un analista de requerimientos, un arquitecto, un programador y un tester), mientras que con 8 empleados se disponen de 2 empleados por disciplina y así sucesivamente con 12 y 16 empleados.

En la Tabla 4.4 se presenta la duración promedio mínima y la duración promedio máxima que se obtuvo con equipos de 4, 8, 12 y 16 empleados en un proyecto con 30 tareas. Aunque se experimentó con los 4 niveles de productividad, se presentan únicamente los resultados con un nivel de productividad medio (en el cual los factores de productividad están en el intervalo [50%-74%]).

En los casos de prueba con 4 empleados, las duraciones promedio mínima y máxima fueron considerablemente mayores con respecto a los casos con 8, 12 y 16 empleados. Esta situación se debe principalmente a que gran parte de las soluciones presentaron sobrecarga en el caso de 4 empleados y, cuando ésta fue eliminada, las duraciones promedio mínima y máxima se extendieron considerablemente, mientras que en los casos restantes la presencia de sobrecarga fue muy baja, por lo que la duración no se extendió demasiado.De hecho, en el caso con 4 empleados, el incremento promedio en la duración al haber resuelto el conflicto de sobrecarga fue entre 40% y 50%. En cambio, con 8 empleados este incremento fue menor a 5%, mientras que con 16 empleados fue menor a 1%. Una posible razón de esta situación, es que cuando se presenta sobrecarga en el

proyecto y se agregan más empleados, las tareas se redistribuyen de manera más eficiente, lo cual produce que la duración resultante al eliminar este conflicto se extienda menos, con respecto a casos en los que se dispone de una menor cantidad de empleados. Por otra parte, cuando ya no se presenta el conflicto de sobrecarga, llega un momento en el cual el incremento en el número de empleados ya no permite reducir la duración de un proyecto (por ejemplo, en la duración promedio mínima con 12 y 16 empleados).

	NSGA-II		SPEA2		SMS-EMOA	
Número de empleados	**Duración promedio mínima**	**Duración promedio máxima**	**Duración promedio mínima**	**Duración promedio máxima**	**Duración promedio mínima**	**Duración promedio máxima**
4	646	646	646	646	646	646
8	387	441	386	453	390	446
12	372	441	371	443	374	438
16	372	395	371	401	372	399

Tabla 4.4. Duración promedio mínima y duración promedio máxima en horas de un proyecto con 30 tareas, al variar el número de empleados.

Para explicar la situación en la cual el incremento de empleados ya no permite reducir la duración de un proyecto, se presenta un ejemplo muy simple (ver Figura 4.1) en el cual hay un calendario de tareas previamente definido (parte izquierda de la figura) y que es posible completar utilizando diversos empleados, como es posible apreciar en la parte derecha de la figura. Obsérvese que si hay un empleado disponible, el proyecto se completará en 10 horas. En cambio, si hay 2 empleados disponibles, la carga de trabajo se distribuye de mejor manera y el proyecto se completará en 6 horas. Sin embargo, al agregar un tercer empleado ya no es posible reducir más la duración del proyecto, por lo cual, en términos prácticos, el nuevo empleado es innecesario. Una situación similar se presentó en los casos con 12 y 16 empleados en este grupo de pruebas.

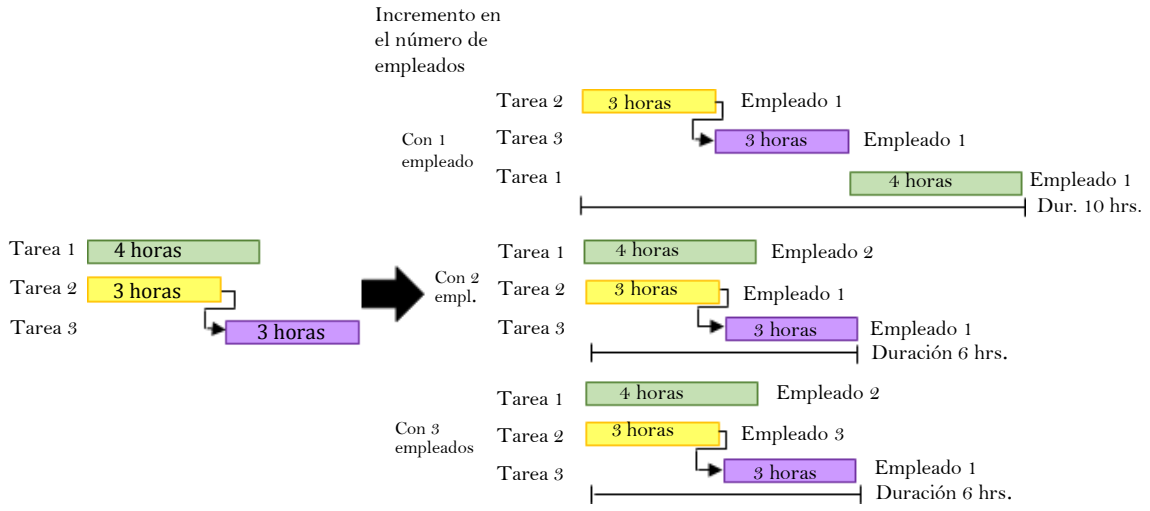


Figura 4.1. Duración de un proyecto simple con un número variable de empleados.

En la tabla 4.4 es posible observar que con 4 empleados la duración promedio mínima y la duración promedio máxima es equivalente con los 3 algoritmos, debido a que solo se encontró una solución. Mientras tanto, con 8 empleados la duración promedio mínima fue 0.76% menor con

NSGA-II y 1.02% menor con SPEA2, con respecto a la obtenida con SMS-EMOA; mientras que la duración promedio máxima fue 2.65% menor con NSGA-II y 1.54% menor con SMS-EMOA, con respecto a la obtenida con SPEA2. Con 12 empleados, la duración promedio mínima fue 0.53% con NSGA-II y 0.80% menor con SPEA2, con respecto a la obtenida con SMS-EMOA; mientras que la duración promedio máxima fue 0.45% menor con NSGA-II y 1.12% menor con SMS-EMOA, con respecto a la obtenida con SPEA2. Finalmente, con 16 empleados la duración promedio mínima fue 0.27% menor con SPEA2 con respecto a NSGA-II y SMS-EMOA; mientras que la duración promedio máxima fue 1.5% menor con NSGA-II y 0.5% menor con SMS-EMOA, con respecto a la obtenida con SPEA2.

Con base en la diferencia en porcentajes, NSGA-II encontró soluciones que en promedio presentan una menor duración en estos casos de prueba, con respecto a las soluciones encontradas por SPEA2 y SMS-EMOA.

Por otra parte, en la Tabla 4.5, se presenta el costo promedio mínimo y el costo promedio máximo obtenido con un número variable de empleados, utilizando los 3 algoritmos de optimización.

	NSGA-II		**SPEA2**		**SMS-EMOA**	
Número de empleados	**Costo promedio mínimo**	**Costo promedio máximo**	**Costo promedio mínimo**	**Costo promedio máximo**	**Costo promedio mínimo**	**Costo promedio máximo**
4	144,100	144,100	144,100	144,100	144,100	144,100
8	145,323	146,990	145,347	147,147	145,377	147,260
12	145,780	147,680	145,867	147,700	145,917	147,817
16	147,140	147,763	147,087	147,870	147,157	147,860

Tabla 4.5. Costo promedio mínimo y costo promedio máximo de un proyecto con 30 tareas al variar el número de empleados.

Es importante recordar que en el modelo de planeación propuesto, el costo total de un proyecto se estima a partir del costo efectivo de todas las tareas (tal como en modelos de planeación de otros trabajos o como se calcula en programas de cómputo como Project, Gantter, etc.). El costo efectivo de una tarea es el producto de su duración en horas y el costo por hora del empleado asignado a esta.

En la tabla 4.5 se puede observar que el costo se incrementa a medida que aumenta el número de empleados, aunque las diferencias en los 4 equipos son menores al 3%. Estas diferencias se deben a las variaciones que hay en los factores de productividad de los empleados, a pesar de que éstos se encuentran en el mismo nivel de productividad, por ejemplo, cuando dos empleados tienen un nivel de productividad medio (cuyo intervalo de factores es [50%-74%]) pero tienen factores de 55% y 60% respectivamente.

Además, es posible observar que en el proyecto de 30 tareas, tanto el costo promedio mínimo como el costo promedio máximo, son equivalentes con los 3 algoritmos de optimización en el caso con 4 empleados, debido a que en este caso hay solo una solución al problema. Con 8 empleados, el costo promedio mínimo es 0.03% menor con NSGA-II y 0.02% menor con SPEA2, con respecto al obtenido con SMS-EMOA; mientras que el costo promedio máximo fue 0.18% menor con NSGA-II y 0.07% menor con SPEA2, con respecto al obtenido con SMS-EMOA. En tanto, con 12 empleados el

costo promedio mínimo fue 0.09% menor con NSGA-II y 0.03% menor con SPEA2, con respecto al obtenido con SMS-EMOA; mientras que el costo promedio máximo fue 0.09% menor con NSGA-II y 0.07% menor con SPEA2, con respecto al obtenido con SMS-EMOA. Finalmente, con 16 empleados el costo promedio mínimo fue 0.01% menor con NSGA-II y 0.04% menor con SPEA2, con respecto al obtenido con SMS-EMOA; mientras que el costo promedio máximo fue 0.07% menor con NSGA-II y 0.006% menor con SMS-EMOA, con respecto al obtenido con SPEA2.

Por lo tanto, en estos casos particulares, NSGA-II y SPEA2 encontraron soluciones que en promedio presentan un costo promedio mínimo y máximo menor, con respecto a las encontradas por SMS-EMOA.

El promedio de soluciones (escenarios) obtenidas con los 4 equipos de empleados y los 3 algoritmos, se presenta en la tabla 4.6. En el caso con 4 empleados sólo hay un empleado disponible por cada disciplina, por lo cual solamente hay una posible forma de hacer la asignación de tareas. Mientras tanto, con 8 empleados se obtuvieron entre 5 y 6 soluciones en promedio, con 12 empleados se obtuvieron entre 6 y 7 soluciones en promedio y con 16 empleados se obtuvieron entre 5 y 6 soluciones en promedio.

Algoritmo	**Número de empleados**			
	4	**8**	**12**	**16**
NSGA-II	1	5	6	5
SPEA2	1	6	7	6
SMS-EMOA	1	5	6	5

Tabla 4.6. Promedio de soluciones obtenidas al resolver el grupo de casos con variación en el número de empleados.

4.4 Empleados con características mixtas

En la práctica es poco común que todos los empleados en un equipo posean un mismo nivel de productividad y un mismo costo, por esta razón, se ejecutaron los algoritmos de optimización para resolver casos de prueba en los cuales los empleados presentan variación en tales características y también en su perfil técnico. Por simplicidad, se consideraron 2 niveles de productividad: bajo y alto, mientras que el costo de los empleados es proporcional a su nivel de productividad. Aunque en la práctica, el costo de los empleados puede depender de diversos factores como la cantidad de años de experiencia del empleado, el perfil o el tiempo de disponibilidad (tiempo completo o medio tiempo), en este caso se decidió que el costo por hora de cada empleado fuera el valor de su productividad. De esta manera un empleado que posee un factor de productividad de 30%, también tiene un costo bajo por hora de $30.

Las pruebas se realizaron con las cantidades de empleados que se han venido utilizando (4, 8, 12 y 16), aunque en este apartado solo se reportan los resultados con 8 empleados, ya que en el resto de configuraciones se observó un comportamiento similar. En cada configuración se tiene la misma cantidad de perfiles por disciplina del ciclo de vida del proyecto, de esta manera, en el equipo de 8 empleados 2 son especialistas en tareas de la disciplina 1 (por ejemplo, en requerimientos), 2 empleados son especialistas en tareas de la disciplina 2 (por ejemplo, en diseño), 2 empleados son especialistas en tareas de la disciplina 3 (por ejemplo, en codificación) y 2 empleados son especialistas en tareas de la disciplina 4 (por ejemplo, en pruebas). Se establecieron combinaciones

variando los 2 niveles de productividad mencionados, y de la misma manera, 2 niveles de costo. Con 8 empleados, las combinaciones fueron las siguientes:

- todos los empleados con productividad baja (y costo por hora bajo), (8B-0A)
- 7 empleados con productividad baja (y costo por hora bajo) y 1 empleado con productividad alta (y costo por hora alto) (7B-1A),
- 6 empleados con productividad baja (y costo por hora bajo) y 2 empleados con productividad alta (y costo por hora alto) (6B-2A),
- 5 empleados con productividad baja (y costo por hora bajo) y 3 empleados con productividad alta (y costo por hora alto) (5B-3A),
- 4 empleados con productividad baja (y costo por hora bajo) y 4 empleados con productividad alta (y costo por hora alto) (4B-4A),
- 3 empleados con productividad baja (y costo por hora bajo) y 5 empleados con productividad alta (y costo por hora alto) (3B-5A),
- 2 empleados con productividad baja (y costo por hora bajo) y 6 empleados con productividad alta (y costo por hora alto) (2B-6A),
- 1 empleado con productividad baja (y costo por hora bajo) y 7 empleados con productividad alta (y costo por hora alto) (1B-7A),
- todos los empleados con productividad alta (y costo por hora alto) (0B-8A)

Retomando la clasificación de empleados por nivel de habilidad sugerida en [García2014], estas combinaciones pueden representar equipos cuyos empleados tienen nivel de habilidad junior o bien, nivel experto. En la Figura 4.2 se presenta un gráfico con el comportamiento de la duración en proyectos con 20 y 30 tareas. Como los resultados son similares con los 3 algoritmos, se presentan los obtenidos con NSGA-II. Cada barra o rectángulo en el gráfico, representa la diferencia entre la duración promedio máxima y la duración promedio mínima en la respectiva combinación.

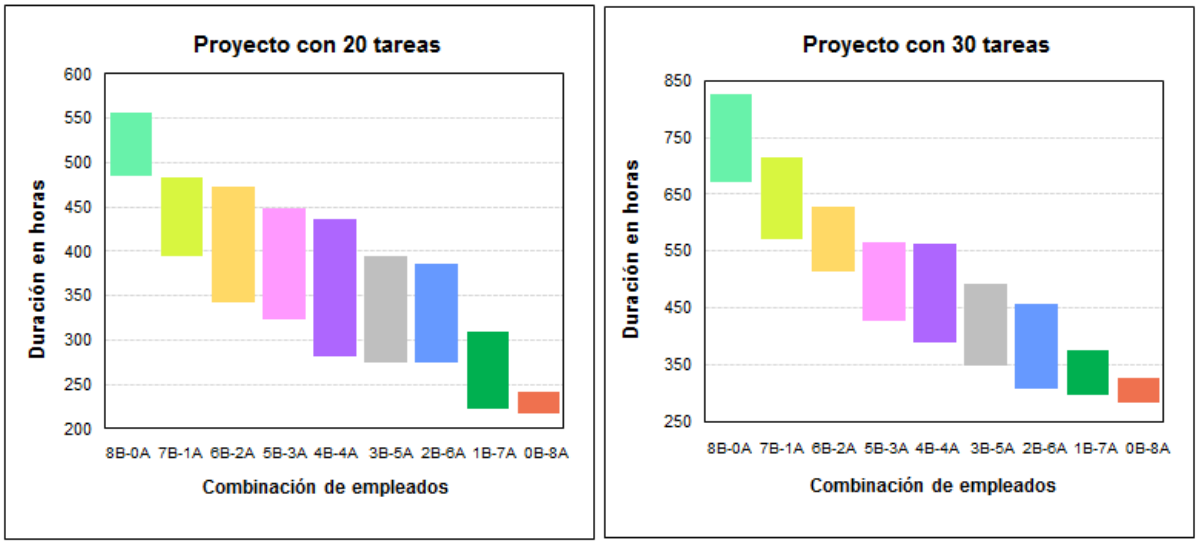


Figura 4.2. Duración promedio máxima y duración promedio mínima de proyectos con 20 y 30 tareas al formar combinaciones de 8 empleados con niveles de productividad bajo y alto.

En los gráficos anteriores se puede apreciar cómo la duración promedio mínima y la duración promedio máxima disminuyen poco a poco conforme se van sustituyendo empleados con menor nivel de productividad, por empleados con mayor productividad. Sin embargo, la sustitución de un empleado con menor productividad por un empleado con mayor productividad, no siempre garantiza reducir la duración de un proyecto. De hecho, para todas las ejecuciones del mismo caso,

se obtuvieron algunas soluciones en las cuales la duración fue mayor al utilizar una mayor cantidad de empleados con "productividad alta", con respecto a combinaciones en las cuales se disponía de una menor cantidad de empleados con este mismo nivel de productividad. Estas situaciones particulares se pueden presentar debido a las posibles formas de asignar las tareas a los empleados. Aunque las tareas asignadas al nuevo empleado se desarrollen en un tiempo menor, puede suceder que al resto de los empleados les tome un mayor tiempo completar sus tareas y por lo tanto la duración del proyecto sea mayor.

Por otra parte, el costo promedio mínimo y el costo promedio máximo de los proyectos con 20 y 30 tareas presentan un comportamiento creciente (ver Figura 4.3). Este comportamiento se debe a que en cada combinación se tiene un mayor número de empleados con alta productividad y con alto costo. Es importante recordar que en los grupos de casos anteriores (variación en el nivel de productividad y variación en el número de empleados), el sueldo de los empleados fue el mismo ($100 por hora) para todos los niveles de productividad, pero en este grupo, el costo por hora de un empleado fue igual a su factor de productividad, es decir, los costos de los empleados fueron menores a $100. Estas características produjeron que los costos de proyectos en este grupo de casos, fueran menores que los obtenidos en los grupos de casos anteriores.

En el proyecto con 20 tareas, la diferencia entre la mayor duración y la menor duración, considerando todas las configuraciones, fue apenas de 0.82%, mientras que en el proyecto con 30 tareas, esta diferencia fue de 0.83%. En casos particulares también se observaron soluciones en las cuales el costo de un proyecto fue mayor al formar combinaciones con una mayor cantidad de empleados con "costo bajo", con respecto a combinaciones con menor cantidad de empleados con este mismo nivel de costo. Esto se debe a que en ocasiones al utilizar una mayor cantidad de empleados con un nivel de productividad bajo y costo bajo, el proyecto se desarrolle en un periodo mayor y a largo plazo pueda resultar más costoso que si se utilizaran empleados con más experiencia y con mayor costo. El número de soluciones que se obtuvieron en las combinaciones anteriores con los proyectos de 20 y 30 tareas se presenta en la Tabla 4.7.

Combinación de empleados	**Promedio de soluciones por proyecto**	
	20 tareas	**30 tareas**
8B-0A	7	7
7B-1A	6	8
6B-2A	5	6
5B-3A	4	6
4B-4A	9	10
3B-5A	6	5
2B-6A	6	6
1B-7A	8	7
0B-8A	5	8

Tabla 4.7. Promedio de soluciones obtenidas al resolver el grupo de casos en el cual los empleados presentan características mixtas.

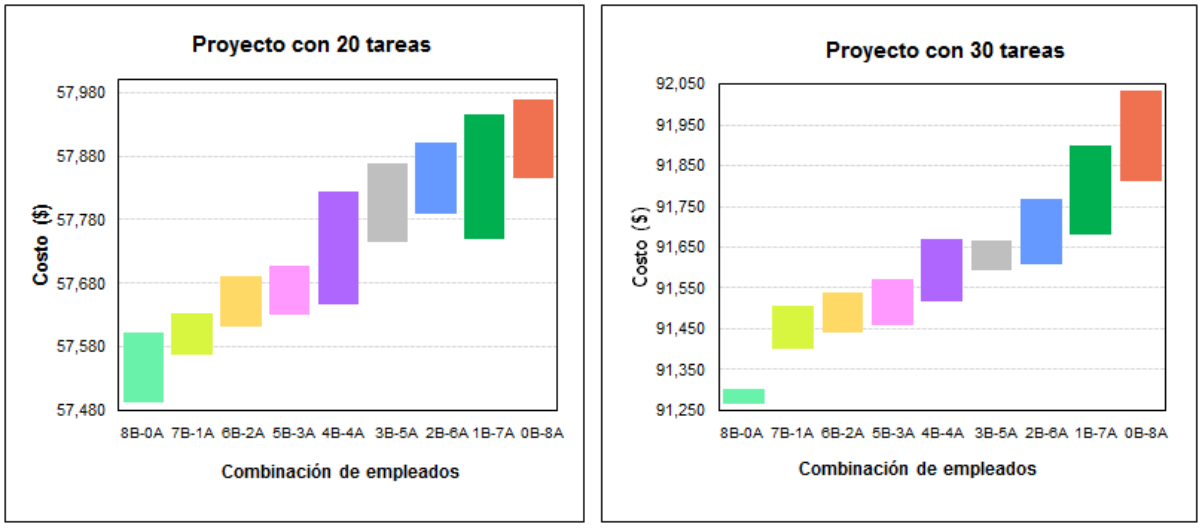


Figura 4.3. Costo promedio máximo y mínimo de proyectos con 20 y 30 tareas al formar combinaciones de 8 empleados con costos bajos y altos.

4.5 Datos aleatorios en los parámetros de un proyecto

En este grupo de casos de prueba, todos los valores de las características de las tareas y los empleados fueron generados de forma aleatoria, y además, los empleados se especializan en múltiples disciplinas, es decir, presentan un perfil mixto. Al utilizar valores aleatorios es posible que se presenten situaciones que probablemente en la práctica no tengan mucho sentido, por ejemplo, que el costo de los empleados sea relativamente alto, mientras que sus factores de productividad sean relativamente bajos y viceversa. También puede suceder que los factores de productividad de los empleados y los esfuerzos estimados de las tareas presenten mucha disparidad, es decir, algunos sean valores muy altos y otros muy bajos. Pese a estas posibles situaciones, el propósito de estas pruebas fue corroborar que incluso usando datos aleatorios, es posible obtener soluciones congruentes en términos de la estimación de duración y costo para diversos proyectos.

Para realizar los experimentos de este grupo de casos de prueba, se generaron 3 grafos de precedencia de tareas (proyectos), uno con 20 tareas, uno con 30 tareas y uno con 40 tareas, y esas tareas pertenecen a 4 disciplinas del ciclo de vida. Al igual que en los anteriores grupos de casos de prueba, se formaron equipos con 4, 8, 12 y 16 empleados, donde los 4 primeros empleados forman parte del equipo de 8 empleados, los cuales a su vez se utilizan para formar parte del equipo de 12 empleados, y finalmente el equipo de 12 empleados forma parte del equipo de 16 empleados. Todos los empleados participan en los 3 proyectos.

Los datos generales de los empleados en este grupo de casos de prueba fueron los siguientes:

- El equipo de 4 empleados presentó un costo promedio de 48.75 por hora con una desviación estándar de 16.54 y factores de productividad promedio de 48.56% con una desviación estándar de 20.7.

- El equipo de 8 empleados presentó un costo promedio de 51.75 por hora con una desviación estándar de 13.57 y factores de productividad promedio de 49.91% con una desviación estándar de 25.14.

- El equipo de 12 empleados presentó un costo promedio de 56.08 por hora con una desviación estándar de 15.27 y factores de productividad promedio de 51.51% con una desviación estándar de25.9.

- El equipo de 16 empleados presentó un costo promedio de 57.43 por hora con una desviación estándar de 13.57 y factores de productividad promedio de 50.44% con una desviación estándar de25.5.

El esfuerzo estimado promedio de las tareas en el proyecto de 20, fue de 24.7 horas con una desviación estándar de 12.95, mientras que en el proyecto de 30 tareas el esfuerzo estimado promedio fue de 22.7 horas con una desviación estándar de 11.09 y en el proyecto de 40 tareas el esfuerzo estimado promedio fue de 24.15 horas con una desviación estándar de 9.67. Los esfuerzos estimados de las tareas son con respecto a un empleado de productividad máxima (100%).

Los valores de las características de los empleados y las tareas de los 3 proyectos de prueba, se presentan en el Anexo C del presente documento.

Los resultados de la duración promedio mínima y la duración promedio máxima utilizando los 3 algoritmos de optimización , se presentan en la Tabla 4.8.

		NSGA-II		SPEA2		SMS-EMOA	
Número de tareas	**Número de empleados**	**Duración promedio mínima**	**Duración promedio máxima**	**Duración promedio mínima**	**Duración promedio máxima**	**Duración promedio mínima**	**Duración promedio máxima**
20	**4**	277	404	280	412	280	389
	8	241	302	245	298	247	299
	12	238	294	240	297	242	298
	16	234	266	234	272	234	270
30	**4**	365	493	366	483	373	495
	8	235	310	235	316	239	309
	12	224	256	222	259	225	258
	16	222	254	221	268	226	264
40	**4**	555	721	563	765	566	751
	8	343	439	341	467	345	462
	12	314	382	316	414	320	402
	16	285	322	278	342	283	319

Tabla 4.8. Duración promedio máxima y duración promedio mínima al resolver el grupo de casos de prueba con datos aleatorios.

En los resultados de la tabla 4.8 se observa que en el proyecto de 20 tareas, al utilizar 4 empleados se obtuvo una duración promedio mínima y máxima más alta que al utilizar 8 empleados. Las diferencias en el caso de la duración promedio mínima fueron de 13% con NSGA-II, de 12.5% con SPEA2 y de 11.8% con SMS-EMOA, mientras que en el caso de la duración promedio máxima,

las diferencias fueron de 27.2% con NSGA-II, de 27.7% con SPEA2 y de 23.1% con SMS-EMOA. Estas diferencias se deben a que al utilizar 4 empleados se presentó una mayor sobrecarga en el proyecto, y al resolverse este conflicto, la duración se extendió 36% en promedio, mientras que con 8 empleados se extendió 11.5% en promedio. Posiblemente si hubiese poca o nula sobrecarga en el proyecto, la diferencia en la duración promedio sería más baja, pues la diferencia promedio en productividad fue menor a 2% como se especificó en los datos generales.

Por otra parte, las diferencias en la duración mínima promedio fueron más bajas entre los casos con 8 y 12 empleados y entre los casos con 12 y 16 empleados debido a que la presencia de sobrecarga fue más baja también y la diferencia promedio en productividad fue también menor a 2%. La diferencia en la duración mínima promedio entre los casos con 8 y 12 empleados fue 1.24% con NSGA-II y de 2% con SPEA2 y SMS-EMOA, mientras que las diferencias en la duración promedio máxima fueron de 2.6% con NSGA-II y de 0.3% con SPEA2 y SMS-EMOA. Las diferencias en la duración promedio mínima entre los casos con 12 y 16 empleados fueron de 1.7% con NSGA-II, de 2.5% con SPEA2 y de 3.3% con SMS-EMOA y las diferencias en la duración promedio máxima fueron de 9.5% con NSGA-II, de 8.4% con SPEA2 y de 9.3% con SMS-EMOA. La diferencia en productividad promedio también fue menor a 2% entre ambos equipos de empleados. En los proyectos con 30 y 40 tareas, la duración promedio mínima y la duración promedio máxima disminuyen al variar el número de empleados. Al igual que en el proyecto de 20 tareas, la diferencia en la duración promedio fue mayor entre los casos con 4 y 8 empleados, pues al resolverse el conflicto de sobrecarga la duración resultante es mayor que en el caso de 8 empleados.

Con respecto a las diferencias en los resultados arrojados por los algoritmos de optimización, en el proyecto de 20 tareas tales diferencias fueron menores a 2.5% con los 4 equipos de empleados en lo referente a la duración promedio mínima, mientras que las diferencias en la duración promedio máxima fueron menores a 5.5%. Con el proyecto de 20 tareas, NSGA-II arrojó soluciones que en promedio presentaron una duración promedio menor con respecto a las arrojadas por SPEA2 y SMS-EMOA. En el proyecto con 30 tareas las diferencias en la duración mínima promedio entre los algoritmos fueron menores a 2.2%, mientras que las diferencias en la duración promedio máxima fueron menores a 5.2%. En este proyecto NSGA-II y SPEA2 arrojaron soluciones que en promedio presentaron una menor duración con respecto a las arrojadas por SMS-EMOA. En el caso del proyecto de 40 tareas, las diferencias en la duración promedio mínima fueron menores a 2.4% y las diferencias en la duración promedio máxima fueron menores a 6%. En este proyecto, NSGA-II obtuvo una duración promedio mínima más baja en los casos con 4 y 12 empleados y SPEA2 en los casos con 8 y 16 empleados, mientras NSGA-II obtuvo una duración promedio máxima más baja en los casos con 4, 8 y 12 empleados y SMS-EMOA en el caso con 16 empleados.

Mientras tanto, en la Tabla 4.9 se presentan los resultados obtenidos con respecto al costo promedio mínimo y al costo promedio máximo, con los 3 algoritmos. En los 3 proyectos se puede observar cómo el costo promedio mínimo y el costo promedio máximo presentan un incremento al variar el número de empleados. Este comportamiento se debe a que al agregar más empleados a un proyecto, la duración promedio disminuye pero el proyecto se vuelve más costoso, y además, porque el costo promedio por hora de los equipos tuvo un incremento (48.75, 51.75, 56.08 y 57.43, respectivamente).

Al hacer la comparación de resultados por algoritmo, se observa que en el proyecto de 20 tareas las diferencias en el costo promedio mínimo entre los 3 algoritmos de optimización, fueron menores al 1.8%, mientras que las diferencias en el costo promedio máximo fueron menores al 3.34%. En este proyecto, NSGA-II obtuvo un costo promedio mínimo más bajo en los casos con 4, 12 y 16 empleados y SPEA2 en el caso con 8 empleados, mientras que el costo promedio máximo fue menor con NSGA-II en los casos con 4, 12 y 16 empleados y SMS-EMOA en el caso con 8 empleados.

		NSGA-II		SPEA2		SMS-EMOA	
Número de tareas	**Número de empleados**	**Costo promedio mínimo**	**Costo promedio máximo**	**Costo promedio mínimo**	**Costo promedio máximo**	**Costo promedio mínimo**	**Costo promedio máximo**
20	**4**	36,058	42,524	36,290	43,516	36,697	43,611
	8	39,744	43,138	39,210	43,237	39,583	42,845
	12	44,202	47,795	44,269	48,355	44,859	47,926
	16	44,419	48,511	44,457	49,027	44,464	50,190
30	**4**	54,360	57,819	54,847	59,252	54,966	59,202
	8	55,335	59,444	55,363	61,527	55,605	61,425
	12	57,367	60,198	57,272	61,855	57,842	61,676
	16	58,170	64,281	57,830	65,406	58,347	64,162
40	**4**	95,728	98,505	95,212	100,867	94,943	99,075
	8	96,254	101,931	96,733	104,088	96,613	103,207
	12	99,401	103,578	100,373	107,462	100,183	106,278
	16	108,299	111,180	108,535	111,718	108,686	111,881

Tabla 4.9. Costo promedio mínimo y costo promedio máximo de proyectos al resolver el grupo de casos de prueba con datos aleatorios.

En tanto, en el proyecto con 30 tareas NSGA-II, las diferencias en el costo promedio mínimo fueron menores al 1.1% entre los 3 algoritmos, mientras que las diferencias en el costo promedio máximo fueron menores al 3.4%. En este proyecto, NSGA-II obtuvo un costo promedio mínimo más bajo en los casos con 4 y 8 empleados y SPEA2 en los casos con 12 y 16 empleados, mientras que NSGA-II obtuvo un costo promedio más bajo en los casos con 4, 8 y 12 empleados y SMS-EMOA en el caso con 16 empleados. Finalmente, en el proyecto de 40 tareas, las diferencias en el costo promedio mínimo fueron menores al 1% con los algoritmos, mientras que las diferencias en el costo promedio máximo fueron menores al 3.61%. En este proyecto con NSGA-II se obtuvo un costo promedio mínimo más bajo en los casos con 8, 12 y 16 empleados y con SMS-EMOA en el caso con 4 empleados, mientras que con NSGA-II se obtuvo un costo promedio máximo más bajo con los 4 equipos.

El promedio de soluciones que se obtuvo en los 3 proyectos, utilizando los 3 algoritmos de optimización, se presenta en la Tabla 4.10.

		Algoritmo de optimización		
Número de tareas	**Número de empleados**	**NSGA-II**	**SPEA2**	**SMS-EMOA**
	4	10	10	8
	8	5	5	5

20	**12**	5	5	5
	16	4	5	4
30	**4**	18	15	13
	8	7	9	8
	12	4	7	5
	16	4	7	6
40	**4**	11	10	11
	8	9	8	8
	12	6	6	8
	16	5	6	9

Tabla 4.10. Promedio de soluciones obtenidas al resolver el grupo de casos de prueba con datos aleatorios.

Los resultados obtenidos en este grupo de pruebas indicaron que a medida que disminuye la duración de un proyecto, regularmente el costo se incrementa y viceversa.

A continuación se presentan 2 figuras en las cuales se observa el frente de Pareto (duración vs costo) obtenido en una de las ejecuciones de los algoritmos de optimización al resolver 2 casos de prueba de este grupo. En la Figura 4.4 se ilustra el frente de Pareto utilizando el proyecto con 30 tareas y 8 empleados, mientras que en la Figura 4.5 se presenta el frente de Pareto utilizando el mismo proyecto con 30 tareas pero con 16 empleados. En el caso de prueba con 8 empleados, NSGA-II encontró soluciones que presentaron en promedio duración y costo menores con respecto a las soluciones encontradas por SPEA2 y SMS-EMOA, mientras que en el caso con 16 empleados, no se obtuvo un frente de Pareto en el cual todas las soluciones tuvieran una menor duración y un menor costo, con respecto a los otros frentes. Como es característico de un frente de Pareto, la solución que presenta la duración más baja presenta el costo más alto y viceversa.

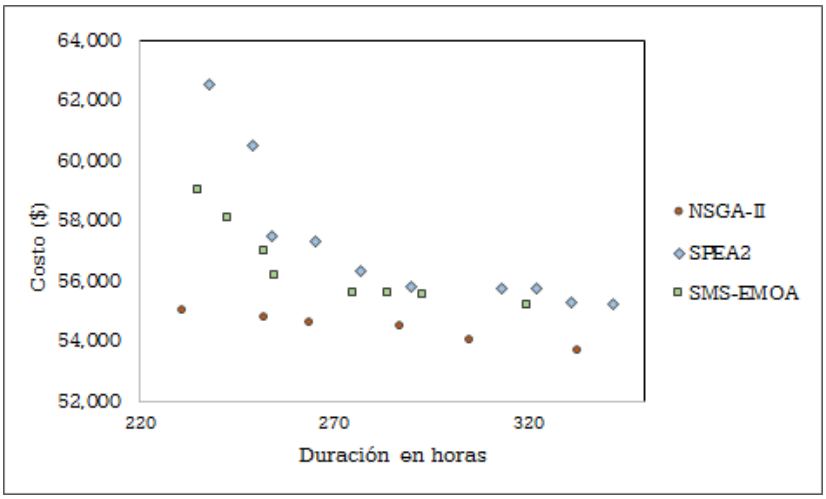


Figura 4.4. Frente de Pareto obtenido en una de las ejecuciones utilizando un proyecto de 30 tareas y 8 empleados, utilizando datos aleatorios.

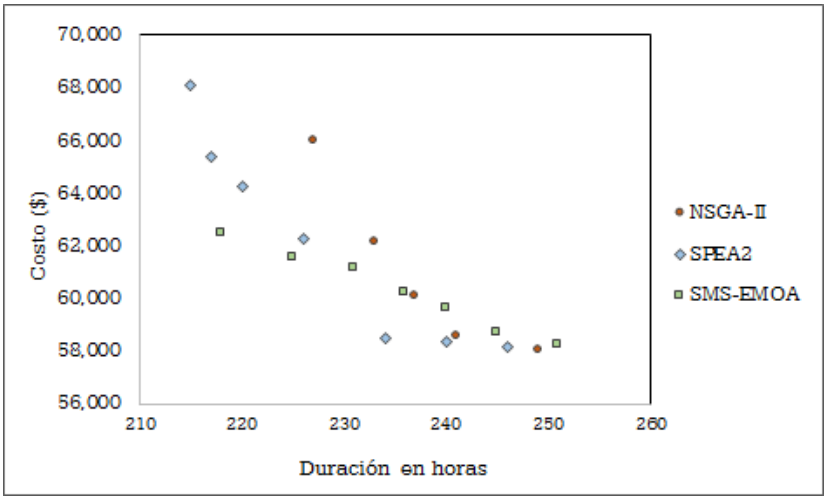


Figura 4.5. Frente de Pareto obtenido en una de las ejecuciones utilizando un proyecto de 30 tareas y 16 empleados, utilizando datos aleatorios.

4.6 Discusión de resultados

Para corroborar la sensatez de GENESPLAN con respecto a la generación de escenarios de planeación, se definieron 4 grupos de experimentos o casos de prueba donde cada uno de ellos presenta una variación en los parámetros que puede tener un proyecto, como el número de tareas y empleados y sus respectivas características. Para resolver los casos de prueba en cada grupo, se ejecutaron los algoritmos de optimización NSGA-II [Deb2002], SPEA2 [Zitzler2001] y SMS-EMOA [Beume2007], con el objetivo de verificar si los resultados arrojados son congruentes de acuerdo a las características de los empleados. Para hacer el análisis correspondiente, se registraron los resultados asociados con la duración promedio mínima, la duración promedio máxima, el costo promedio mínimo y el costo promedio máximo, al resolver los casos de pruebas. A continuación se describe una síntesis de lo observado en cada grupo de casos de prueba.

En el primer grupo (variación del nivel de productividad de los empleados) se corroboró que en efecto cuando los empleados presentan un nivel de productividad mayor, la duración del proyecto disminuye y viceversa. Como el costo permanece fijo en este grupo, el costo disminuye a medida que la duración disminuye. Al resolver este grupo de casos de prueba, SMS-EMOA encontró soluciones que en promedio tienen una menor duración con respecto a las encontradas por NSGA-II y SPEA2, mientras que NSGA-II encontró soluciones que en promedio presentan un costo menor con respecto a las soluciones que arrojaron SPEA2 y SMS-EMOA.

En el segundo grupo de casos de prueba (variación en el número de empleados), se observó que cuando un proyecto presenta sobrecarga, el incremento en el número de empleados ayuda a que la duración resultante al eliminar este conflicto, sea menor que la duración obtenida si se utiliza una menor cantidad de empleados, ya que al haber pocos empleados disponibles, la pila de tareas es mayor para ellos y les toma un mayor tiempo completarlas, en cambio al haber más empleados disponibles, la carga de trabajo se redistribuye de manera más eficiente, ocasionando que la duración de ese proyecto sea más baja. Mientras tanto, cuando un proyecto ya no presenta el conflicto de sobrecarga, llega un momento en el cual el incremento de empleados (con un mismo nivel de productividad) ya no permite reducir más la duración de un proyecto, aunque sí incrementa el costo. En la resolución de este grupo de pruebas, NSGA-II encontró soluciones que presentaron en promedio una menor duración con respecto a soluciones arrojadas por SPEA2 y SMS-EMOA,

mientras que NSGA-II y SPEA2 hallaron soluciones que presentaron en promedio, un costo menor que las soluciones arrojadas por SMS-EMOA.

En el tercer grupo (características mixtas de los empleados), se observó que la duración promedio mínima y la duración promedio máxima en distintos proyectos, regularmente presentaron un comportamiento decreciente cuando se sustituyó un empleado con un nivel de productividad bajo por un empleado con nivel de productividad alto; mientras que el costo promedio mínimo y el costo promedio máximo, presentaron un comportamiento regular creciente al sustituir empleados con menor productividad (y menor costo) por empleados con mayor productividad (y mayor costo), presentándose una correlación inversa entre ambos objetivos. A pesar de que se obtuvo el comportamiento esperado, se encontraron soluciones donde no sucedió de esta manera, por lo cual el hecho de disponer de más empleados con alta productividad, no siempre garantiza disminuir la duración del proyecto y de la misma manera, el hecho de disponer de una mayor cantidad de empleados con costo bajo, no siempre garantiza reducir el costo de un proyecto. En este grupo de casos de prueba, NSGA-II también arrojó soluciones que en promedio presentaron una menor duración y un menor costo, con respecto a las soluciones arrojadas por SPEA2 y SMS-EMOA.

Finalmente, en el grupo de casos de prueba con datos aleatorios, se observó que la presencia de sobrecarga en un proyecto es más alta cuando se dispone de una baja cantidad de empleados, lo cual produce que la duración resultante al eliminar este conflicto sea mayor que la obtenida si se utiliza una mayor cantidad de empleados. En este caso, la duración promedio mínima y la duración promedio máxima presentaron una disminución al variar el número de empleados, considerando que los empleados de un equipo forman parte del siguiente. Mientras tanto, el costo presentó un comportamiento regular creciente, debido a que con más empleados los proyectos suelen ser más costosos, y además, el costo promedio por hora en cada equipo fue creciente. Los resultados arrojados por los algoritmos en la resolución de este grupo de casos de prueba, nuevamente permitieron corroborar que a medida que la duración de un proyecto disminuye, regularmente el costo se incrementa. Como ejemplos de esta correlación, se presentó un par de gráficos (Figuras 4.5 y 4.6) en el cual se observa el frente de Pareto obtenido al ejecutar los 3 algoritmos.

Con base en los resultados arrojados por los 3 algoritmos de optimización, se corroboró que la duración y el costo presentaron un comportamiento congruente con respecto al esperado, de acuerdo a los datos de distintos casos de prueba. Como resultado de una ejecución de los algoritmos de optimización, se obtiene un frente de Pareto, el cual corresponde a un conjunto escenarios de planeación para un determinado proyecto. Una de las características del grupo de escenarios resultante es que el escenario que presenta la duración más baja, tiene el costo más alto y análogamente, el escenario que presenta la duración más alta, tiene el costo más bajo, por lo cual se corrobora que la duración y el costo de un proyecto regularmente presentan una correlación inversa. Además se observó que las diferencias en porcentaje entre los algoritmos de optimización al resolver los casos de prueba, fueron menores a 6% con respecto a la duración promedio mínima y a la duración promedio máxima, mientras las diferencias con respecto al costo promedio mínimo y al costo promedio máximo fueron menores al 2%. Por otra parte, hubo casos de prueba particulares donde cada algoritmo encontró soluciones que en promedio presentaron una menor duración y un menor costo. En el grupo de casos donde se hizo variación en el nivel de productividad de los empleados, SMS-EMOA encontró soluciones que en promedio presentaron menor duración y menor costo con respecto a las encontradas por NSGA-II y a SPEA2, mientras que en el resto de los grupos de casos de prueba, NSGA-II arrojó soluciones que en promedio presentaron menor duración y menor costo que las arrojadas por SPEA2 y SMS-EMOA. Los resultados obtenidos en los distintos experimentos, permiten comprobar que la herramienta propuesta (GENESPLAN) puede generar escenarios de planeación sensatos para diversos proyectos de desarrollo de software.

Capítulo 5

Conclusiones

5.1 Síntesis

El presente proyecto de investigación consistió en desarrollar una herramienta para generar escenarios de planeación, los cuales corresponden a posibles asociaciones entre tareas y recursos humanos, que de acuerdo a sus características, resultan en estimaciones distintas de duración y costo para un mismo plan de proyecto de software. A esta herramienta se le dió el nombre de GENESPLAN y está basada en un modelo de planeación que es utilizado por un algoritmo de optimización para encontrar soluciones óptimas (escenarios).

De acuerdo con la revisión bibliográfica, se observó que la mayoría de los modelos de planeación presentan una versión relativamente simple de las características de los empleados, por ejemplo las habilidades de los empleados están cuantificadas, no se consideran aspectos de productividad, ni tampoco es posible representar perfiles técnicos de empleados; situaciones que limitan su adopción a proyectos de software reales. Por esta razón se adaptó un modelo (a partir del modelo de Alba y Chicano [Alba2007]) con el fin de apegarse un poco más a lo que se ha observado en algunas organizaciones de software, en las cuales los empleados pueden ser clasificados de distintas maneras según sus conocimientos en una o más fases de un proyecto y según su nivel de habilidad.

El modelo de planeación fue integrado en el prototipo de GENESPLAN, el cual está formado por un conjunto de interfaces gráficas para la captura y el despliegue de información relacionada con los escenarios de planeación. Para cada escenario se muestra la duración, el costo, el periodo de tiempo real y además es posible observar las asignaciones entre tareas y empleados y su correspondiente calendario de tareas en forma de diagrama de Gantt. Desde una perspectiva lógica, GENESPLAN se compone de 2 capas principales: una capa de presentación, en la cual se especifican las interfaces y el flujo de los casos de uso, y una capa llamada "lógica de aplicación", compuesta por módulos que se encargan de realizar cálculos y otros servicios relacionados con el proceso de optimización para la generación de soluciones. Dado que el objetivo de este trabajo no fue implementar un nuevo algoritmo de optimización, una de las decisiones de diseño fue incorporar una librería (framework) llamada jMetal [Durillo2011], la cual cuenta con una implementación de diversos algoritmos, entre ellos NSGA-II [Deb2002], SPEA2 [Zitzler2001] y SMS-EMOA [Beume2007].

Para realizar los experimentos se crearon proyectos artificiales mediante un *generador de casos* que fue desarrollado por Alba y Chicano [Alba2007]. Se definieron y se resolvieron grupos de casos de prueba con el objetivo de corroborar si la herramienta puede generar resultados sensatos con respecto a la duración y al costo de un proyecto. Los algoritmos de optimización utilizados en estas pruebas fueron NSGA-II, SPEA2 y SMS-EMOA, ya que el primero sigue siendo un estándar de comparación con respecto a otros algoritmos, mientras que SPEA y SMS-EMOA son considerados como dos de los mejores algoritmos en la actualidad. Los resultados obtenidos fueron congruentes con respecto al comportamiento esperado y a las características de los empleados, donde NSGA-II arrojó soluciones que en promedio tienen una menor duración y costo que las obtenidas por SPEA2 y SMS-EMOA, en la mayor parte de los casos de prueba.

5.2 Conclusiones

El objetivo general del presente proyecto de investigación fue desarrollar una herramienta para obtener escenarios de planeación en base a parámetros asociados a un plan de proyecto de software, como el conjunto de tareas y el conjunto de recursos humanos disponibles. Para cumplir con este objetivo, se cubrieron los objetivos específicos descritos en el Capítulo 1 Sección 1.4.

El primer objetivo específico fue *revisar el estado del arte sobre el uso de técnicas de optimización en la planeación de proyectos de desarrollo de software.* Este objetivo se cubrió mediante una revisión bibliográfica, en la cual se hizo la selección sistemática, la lectura, el resumen y el análisis de trabajos relacionados con la resolución del problema de planeación de proyectos de software. De esta revisión se observó que gran parte de estos trabajos se basan en el modelo de planeación de Alba y Chicano [Alba2007] y que se centran más en evaluar el desempeño de los algoritmos de optimización empleados, que en evaluar la congruencia y la calidad de las soluciones. Además, se identificaron oportunidades de mejora con respecto al modelo de planeación que fue integrado en la herramienta propuesta (GENESPLAN).

El segundo objetivo específico fue *proponer y desarrollar un modelo de planeación flexible que tomara en cuenta un conjunto de variables para la generación de escenarios de planeación.* A partir de los hallazgos identificados en los modelos de planeación propuestos, se ajustó un modelo para GENESPLAN tomado como base el modelo de Alba y Chicano [Alba2007]. Este modelo incluye características de empleados y tareas que permiten representar distintos contextos organizacionales, en los cuales los empleados pueden ser clasificados de diversas maneras, según su perfil técnico y su nivel de habilidad y/o productividad.

El tercer objetivo fue *identificar las técnicas de optimización más adecuadas para la generación de escenarios de planeación.* En esta fase se identificaron algoritmos que en esos momentos destacaron por su buen desempeño en la resolución del problema de planeación de proyectos de software, por ejemplo: NSGA, SPEA, PAES, etc., sin embargo, en la actualidad SPEA2 [Zitzler2001] y de SMS-EMOA [Beume2007], son considerados los algoritmos de con mejor desempeño en la resolución de diversos problemas de optimización, por lo cual se decidió utilizarlos junto con NSGA-II [Deb2002] para realizar los experimentos correspondientes.

El siguiente objetivo fue *implementar en una herramienta el modelo de planeación propuesto y las técnicas de optimización identificadas.* En esta fase se desarrolló el prototipo de GENESPLAN, el cual se compone de un conjunto de interfaces gráficas que permiten generar los escenarios de planeación y consultar su detalle. La herramienta incluye el modelo de planeación ajustado, un framework que contiene la implementación de diversos algoritmos de optimización y un algoritmo que fue propuesto para resolver conflictos de sobrecarga en un proyecto.

El último objetivo fue *evaluar el modelo de planeación y las técnicas de optimización implementados en la herramienta para generar escenarios*, el cual se cubrió con el diseño y la resolución de 4 grupos de casos de prueba, de acuerdo a las características de empleados y tareas. Se observó que los resultados arrojados por los 3 algoritmos de optimización presentaron ligeras diferencias y que fueron congruentes con base en las características de los proyectos de prueba y en el comportamiento esperado, con respecto a la duración y el costo.

Por medio del prototipo que se desarrolló es posible generar escenarios de planeación para proyectos de desarrollo de software a partir de las características de empleados y tareas, sin embargo, es posible aún mejorar y proponer ciertos aspectos, los cuales se retomarán en un trabajo a futuro. Una de las ventajas de GENESPLAN es la flexibilidad del modelo de planeación que permite

representar diversas conformaciones de equipos, ya que en algunas organizaciones de desarrollo los empleados pueden ser clasificados por sus conocimientos en alguna disciplina del ciclo de vida de software o por su nivel de habilidad y/o experiencia. Es deseable que esta herramienta pueda servir de apoyo para aquellas personas que se dedican a realizar planes de proyectos.

5.3 Contribuciones

Una de las principales contribuciones de este trabajo fue la adaptación de un modelo flexible que permite representar distintos contextos organizacionales, en los cuales los empleados pueden estar clasificados de diversas maneras, ya sea por su nivel de habilidad o por las fases de un proyecto en las cuales poseen mayores conocimientos. De este modo, es posible modelar empleados con distintos perfiles técnicos, por ejemplo: arquitecto de software, programador, ingeniero de pruebas, administrador de proyectos, entre otros; y para cada uno de estos perfiles, se puede representar su nivel de habilidad o experiencia utilizando factores de productividad en esas disciplinas (bajo el supuesto de que un empleado con mayor experiencia tiene la capacidad de desarrollar sus tareas en menor tiempo).

Además, se propuso un algoritmo simple para resolver el conflicto de sobrecarga en un proyecto, el cual se presenta cuando uno o más empleados tienen tareas asignadas que se desarrollan de forma paralela. Este problema es muy recurrente, principalmente cuando los proyectos tienen 30 o más tareas.

Cabe mencionar que el presente trabajo fue presentado en dos eventos académicos. La primera participación fue en la "8a. Semana de Computación y Matemáticas Aplicadas" (SCMA 2015), que se llevó a cabo en la Universidad Autónoma Metropolitana Unidad Cuajimalpa en la Ciudad de México, bajo el título *Planeación de proyectos de desarrollo de software usando técnicas de optimización*. La segunda participación fue en "4th International Conference on Software Engineering Research and Innovation" (CONISOFT 2016, pp. 85-94), que tuvo lugar en la Universidad Popular Autónoma del Estado de Puebla, bajo el título *Desarrollo de una herramienta para generar escenarios de planeación de proyectos.*

5.4 Trabajo a futuro

Diversos aspectos de GENESPLAN pueden ser mejorados. Primero, es posible realizar ajustes al modelo de planeación para la generación de escenarios, ya que se asumen ciertas cuestiones, por ejemplo, que los empleados están disponibles durante todo un proyecto, lo cual no siempre sucede en la práctica, ya que es posible que uno o más empleados abandonen el mismo, o bien, que algunos tengan periodos de inactividad como permisos o vacaciones, lo cual sin duda afecta al desarrollo de tareas. En este caso sería deseable encontrar una nueva asignación, de tal forma que el costo y la duración del proyecto en la asignación actual no se vean afectados considerablemente.

Además, sería deseable realizar experimentos con datos proporcionados por algunas organizaciones y podrían realizarse ajustes al prototipo de GENESPLAN de tal manera que sea una herramienta más interactiva y presentable para el usuario. Hasta el momento, los parámetros de un proyecto tales como las características de tareas y empleados y el grafo de precedencia de tareas, son especificados a través de un archivo de texto, ya que esto facilitó la parte experimental. Para un uso más práctico, una posibilidad es desarrollar interfaces gráficas para la captura de datos, o bien, otra alternativa es obtenerlos de alguna base de datos o documento (en caso de ser posible). Una vez hechos los ajustes correspondientes, se pretende que la herramienta esté disponible públicamente.

Anexo A

Posibles escenarios para un plan de proyecto

Sean T_1, T_2, T_3 y T_4 tareas de cierta disciplina de un proyecto de software (por ejemplo: tareas de diseño), las cuales necesitan ser asignadas a 2 empleados especialistas (E_1 y E_2) en este tipo de tareas. Asumiendo que el número de tareas asignadas a cada empleado puede ser variable, es posible formar combinaciones[9] ($C(n,r)$) para asignar r tareas a E_1 y $(4 - r)$ tareas a E_2, donde $0 \leq r \leq 4$. En este caso, se tienen las siguientes posibilidades:

- $C(4, 0) = 1$ posibilidad para asignar 0 tareas a E_1 (y 4 tareas a E_2),
- $C(4, 1) = 4$ posibilidades para asignar 1 tarea a E_1 (y 3 tareas a E_2),
- $C(4, 2) = 6$ posibilidades para asignar 2 tareas a E_1 (y 2 tareas a E_2),
- $C(4, 3) = 4$ posibilidades para asignar 3 tareas a E_1 (y 1 tarea a E_2) y
- $C(4, 4) = 1$ posibilidad para asignar 4 tareas a E_1 (y 0 tareas a E_2),

por lo tanto, para este ejemplo, hay 16 posibilidades para establecer asignaciones entre tareas y empleados. Estas posibilidades se desglosan en la Tabla A.1.

Número de escenario (o asignación)	**Número de tareas asignadas a cada empleado**	**Asignación**			
		T_1	T_2	T_3	T_4
1	0 tareas asignadas a E_1 y 4 tareas asignadas a E_2	E_2	E_2	E_2	E_2
2	1 tarea asignada a E_1 y 3 tareas asignadas a E_2	E_1	E_2	E_2	E_2
3		E_2	E_1	E_2	E_2
4		E_2	E_2	E_1	E_2
5		E_2	E_2	E_2	E_1
6	2 tareas asignadas a E_1 y 2 tareas asignadas a E_2	E_1	E_1	E_2	E_2
7		E_1	E_2	E_1	E_2
8		E_1	E_2	E_2	E_1
9		E_2	E_1	E_1	E_2
10		E_2	E_1	E_2	E_1
11		E_2	E_2	E_1	E_1
12		E_1	E_1	E_1	E_2

[9] El número de combinaciones para seleccionar r elementos de un total de n, se expresa como C(n,r) = n! / r!(n-r)!

13	3 tareas asignadas a E_1 y 1 tarea asignada a E_2	E_1	E_1	E_2	E_1
14		E_1	E_2	E_1	E_1
15		E_2	E_1	E_1	E_1
16	4 tareas asignadas a E_1 y 0 tareas asignadas a E_2	E_1	E_1	E_1	E_1

Tabla A.1. Posibles escenarios para un plan con 4 tareas, 2 empleados y una disciplina.

Por lo tanto, habría 16 posibilidades para asignar 4 nuevas tareas de una segunda disciplina a 2 nuevos empleados (especialistas en esa disciplina). Esto significa que al considerar 8 tareas (4 por disciplina) y 4 empleados (2 especialistas por disciplina), el total de asignaciones es: 16 * 16 = 216, pues a cada asignación de tareas y empleados de la disciplina 1, se le asocia cada una de las asignaciones del caso de la segunda disciplina. Siguiendo esta analogía, al considerar 3 disciplinas, y al aumentar proporcionalmente el número de tareas y empleados, el total de posibles asignaciones se incrementa a: 16 * 16 * 16 = 4,096. Con una cuarta disciplina, este número se incrementa a 65,536. El número de escenarios posibles crece exponencialmente (para este ejemplo) conforme se incrementa el número de tareas por disciplina y el número de empleados especialistas, por lo cual resulta impráctico explorar todas las posibilidades para encontrar el escenario con menor costo y menor duración.

Anexo B

Parámetros de un caso de prueba

Parámetro	Sintaxis	Ejemplo y significado en el modelo de planeación propuesto
Número de tareas	task.number=<value>	"task.number=10", significa que el proyecto consta de 10 tareas.
Esfuerzo estimado de las tareas	task.<taskNumber>.cost =<value>	"task.1.cost=20.0", significa que la tarea 1 tiene un esfuerzo estimado de 20 horas (con respecto a un empleado con productividad máxima).
* Disciplina de una tarea	task.<taskNumber>.skill =<value>	"task.1.skill=2", significa que la tarea 1 tiene asociada la disciplina 2.
Número de aristas en el grafo de precedencia de tareas (GPT)	graph.arc.number=<value>	"graph.arc.number=20", significa que el grafo de tareas tiene 20 aristas (o dependencias).
Arista o dependencia en el GPT	graph.arc.<arcNumber> =<value1 value2>	"graph.arc.1=2 4" significa que la tarea 2 precede a la tarea 4, en el arista 1.
Número de empleados	employee.number =<value>	"employee.number=30", significa que en el proyecto participan 30 empleados.
costo de un empleado	employee.<employee Number>.salary=<value>	"employee.1.salary=50.0", significa que el empleado 1 posee un costo de $50 (por hora).
* Factor de productividad de un empleado en alguna disciplina	employee.<employeeNumber>. skill.<skillNumber>=<value>	"employee.1.skill.2=70", significa que el empleado 1 posee un factor de productividad de 0.7 (o bien, 70%) en la disciplina 2.
* Estos parámetros tienen un significado distinto en el modelo de planeación.		

Tabla B.1 Parámetros de un caso de prueba para la generación de escenarios.

Anexo C

Casos de prueba con datos aleatorios

C.1. Caso de prueba con un proyecto de 20 tareas

Datos de los empleados **Nota: El costo de los empleados (salary) es redondeado a dos decimales.**	employee.0.salary=32.06642704068693 employee.0.skill.0=50 employee.0.skill.1=56 employee.0.skill.2=63 employee.0.skill.3=69 employee.0.skill.number=4 employee.1.salary=60.820241881344 92 employee.1.skill.0=27 employee.1.skill.1=33 employee.1.skill.2=37 employee.1.skill.3=69 employee.1.skill.number=4 employee.2.salary=65.59761397385385 employee.2.skill.0=7 employee.2.skill.1=36 employee.2.skill.2=41 employee.2.skill.3=43 employee.2.skill.number=4 employee.3.salary=38.569577252936185 employee.3.skill.0=44 employee.3.skill.1=45 employee.3.skill.2=60 employee.3.skill.3=97 employee.3.skill.number=4 employee.4.salary=55.06583897892821 employee.4.skill.0=13 employee.4.skill.1=29 employee.4.skill.2=59 employee.4.skill.3=69 employee.4.skill.number=4 employee.5.salary=40.174615632010216 employee.5.skill.0=17 employee.5.skill.1=54 employee.5.skill.2=59 employee.5.skill.3=97 employee.5.skill.number=4 employee.6.salary=57.684080718570684 employee.6.skill.0=16 employee.6.skill.1=26 employee.6.skill.2=73 employee.6.skill.3=82 employee.6.skill.number=4 employee.7.salary=67.80577378037502 employee.7.skill.0=45 employee.7.skill.1=79 employee.7.skill.2=91 employee.7.skill.3=93 employee.7.skill.number=4 employee.8.salary=87.96311124894521

	employee.8.skill.0=19 employee.8.skill.1=37 employee.8.skill.2=64 employee.8.skill.3=99 employee.8.skill.number=4 employee.9.salary=57.527520805554474 employee.9.skill.0=23 employee.9.skill.1=30 employee.9.skill.2=78 employee.9.skill.3=88 employee.9.skill.number=4 employee.10.salary=48.51317146115117 employee.10.skill.0=10 employee.10.skill.1=66 employee.10.skill.2=16 employee.10.skill.3=69 employee.10.skill.number=4 employee.11.salary=66.23545194352506 employee.11.skill.0=20 employee.11.skill.1=32 employee.11.skill.2=44 employee.11.skill.3=54 employee.11.skill.number=3 employee.12.salary=55.047825234625954 employee.12.skill.0=51 employee.12.skill.1=72 employee.12.skill.2=33 employee.12.skill.3=86 employee.12.skill.number=4 employee.13.salary=64.91412449146308 employee.13.skill.0=70 employee.13.skill.1=48 employee.13.skill.2=63 employee.13.skill.3=93 employee.13.skill.number=4 employee.14.salary=60.57615876837832 employee.14.skill.0=19 employee.14.skill.1=64 employee.14.skill.2=19 employee.14.skill.3=64 employee.14.skill.number=4 employee.15.salary=67.57975726703596 employee.15.skill.0=16 employee.15.skill.1=28 employee.15.skill.2=31 employee.15.skill.3=75 employee.15.skill.number=4 employee.number=16
Datos del grafo de precedencia de tareas	graph.arc.0=2 4 graph.arc.1=0 7 graph.arc.10=2 14 graph.arc.11=13 14 graph.arc.12=4 15 graph.arc.13=5 15 graph.arc.14=12 15 graph.arc.15=7 16 graph.arc.16=3 17 graph.arc.17=5 17 graph.arc.18=11 17 graph.arc.19=16 17 graph.arc.2=4 7 graph.arc.20=5 18 graph.arc.21=0 19

	graph.arc.22=6 19 graph.arc.23=16 19 graph.arc.3=3 10 graph.arc.4=7 10 graph.arc.5=9 10 graph.arc.6=1 11 graph.arc.7=9 11 graph.arc.8=3 12 graph.arc.9=3 13 graph.arc.number=24
Datos de las tareas	task.0.cost=56.0 task.0.skill.0=7 task.0.skill.number=1 task.1.cost=32.0 task.1.skill.0=24 task.1.skill.number=1 task.10.cost=40.0 task.10.skill.0=24 task.10.skill.number=1 task.11.cost=20.0 task.11.skill.0=26 task.11.skill.number=1 task.12.cost=23.0 task.12.skill.0=79 task.12.skill.number=1 task.13.cost=17.0 task.13.skill.0=5 task.13.skill.number=1 task.14.cost=51.0 task.14.skill.0=48 task.14.skill.number=1 task.15.cost=30.0 task.15.skill.0=24 task.15.skill.number=1 task.16.cost=22.0 task.16.skill.0=42 task.16.skill.number=1 task.17.cost=35.0 task.17.skill.0=83 task.17.skill.number=1 task.18.cost=16.0 task.18.skill.0=11 task.18.skill.number=1 task.19.cost=19.0 task.19.skill.0=62 task.19.skill.number=1 task.2.cost=14.0 task.2.skill.0=12 task.2.skill.number=1 task.3.cost=10.0 task.3.skill.0=75 task.3.skill.number=1 task.4.cost=21.0 task.4.skill.0=96 task.4.skill.number=1 task.5.cost=19.0 task.5.skill.0=84 task.5.skill.number=1 task.6.cost=12.0 task.6.skill.0=98 task.6.skill.number=1 task.7.cost=32.0 task.7.skill.0=44 task.7.skill.number=1

	task.8.cost=14.0 task.8.skill.0=85 task.8.skill.number=1 task.9.cost=11.0 task.9.skill.0=13 task.9.skill.number=1 task.number=20

Tabla C.1. Datos del caso de prueba aleatorio utilizando un proyecto de 20 tareas.

C.2. Caso de prueba con un proyecto de 30 tareas

Datos de los empleados **Nota: El costo de los empleados (salary) es redondeado a dos decimales.**	employee.0.salary=32.06642704068693 employee.0.skill.0=50 employee.0.skill.1=56 employee.0.skill.2=63 employee.0.skill.3=69 employee.0.skill.number=4 employee.1.salary=60.820241881344 92 employee.1.skill.0=27 employee.1.skill.1=33 employee.1.skill.2=37 employee.1.skill.3=69 employee.1.skill.number=4 employee.2.salary=65.59761397385385 employee.2.skill.0=7 employee.2.skill.1=36 employee.2.skill.2=41 employee.2.skill.3=43 employee.2.skill.number=4 employee.3.salary=38.569577252936185 employee.3.skill.0=44 employee.3.skill.1=45 employee.3.skill.2=60 employee.3.skill.3=97 employee.3.skill.number=4 employee.4.salary=55.06583897892821 employee.4.skill.0=3 employee.4.skill.1=9 employee.4.skill.2=59 employee.4.skill.3=69 employee.4.skill.number=4 employee.5.salary=40.174615632010216 employee.5.skill.0=17 employee.5.skill.1=54 employee.5.skill.2=59 employee.5.skill.3=97 employee.5.skill.number=4 employee.6.salary=57.684080718570684 employee.6.skill.0=16 employee.6.skill.1=26 employee.6.skill.2=73 employee.6.skill.3=82 employee.6.skill.number=4 employee.7.salary=67.80577378037502 employee.7.skill.0=45 employee.7.skill.1=79 employee.7.skill.2=91 employee.7.skill.3=93 employee.7.skill.number=4 employee.8.salary=87.96311124894521 employee.8.skill.0=19

	employee.8.skill.1=37 employee.8.skill.2=64 employee.8.skill.3=99 employee.8.skill.number=4 employee.9.salary=57.527520805554474 employee.9.skill.0=23 employee.9.skill.1=30 employee.9.skill.2=78 employee.9.skill.3=88 employee.9.skill.number=4 employee.10.salary=48.51317146115117 employee.10.skill.0=10 employee.10.skill.1=66 employee.10.skill.2=16 employee.10.skill.3=69 employee.10.skill.number=4 employee.11.salary=66.23545194352506 employee.11.skill.0=20 employee.11.skill.1=32 employee.11.skill.2=44 employee.11.skill.3=54 employee.11.skill.number=3 employee.12.salary=55.047825234625954 employee.12.skill.0=51 employee.12.skill.1=72 employee.12.skill.2=33 employee.12.skill.3=86 employee.12.skill.number=4 employee.13.salary=64.91412449146308 employee.13.skill.0=70 employee.13.skill.1=48 employee.13.skill.2=63 employee.13.skill.3=93 employee.13.skill.number=4 employee.14.salary=60.57615876837832 employee.14.skill.0=19 employee.14.skill.1=64 employee.14.skill.2=19 employee.14.skill.3=64 employee.14.skill.number=4 employee.15.salary=67.57975726703596 employee.15.skill.0=16 employee.15.skill.1=28 employee.15.skill.2=31 employee.15.skill.3=75 employee.15.skill.number=4 employee.number=16
Datos del grafo de precedencia de tareas	graph.arc.0=0 4 graph.arc.1=4 5 graph.arc.10=13 14 graph.arc.11=15 16 graph.arc.12=2 17 graph.arc.13=2 18 graph.arc.14=6 18 graph.arc.15=8 18 graph.arc.16=8 19 graph.arc.17=18 19 graph.arc.18=1 20 graph.arc.19=7 20 graph.arc.2=8 9 graph.arc.20=14 20 graph.arc.21=10 21 graph.arc.22=17 21

	graph.arc.23=6 22 graph.arc.24=12 22 graph.arc.25=0 23 graph.arc.26=1 24 graph.arc.27=11 24 graph.arc.28=2 25 graph.arc.29=4 25 graph.arc.3=2 10 graph.arc.30=18 25 graph.arc.31=18 26 graph.arc.32=21 26 graph.arc.33=24 26 graph.arc.34=3 27 graph.arc.35=14 27 graph.arc.36=20 27 graph.arc.37=5 28 graph.arc.38=15 29 graph.arc.4=6 10 graph.arc.5=8 11 graph.arc.6=4 12 graph.arc.7=6 12 graph.arc.8=8 13 graph.arc.9=9 13 graph.arc.number=39
Datos de las tareas	task.0.cost=30.0 task.0.skill.0=58 task.0.skill.number=1 task.1.cost=9.0 task.1.skill.0=21 task.1.skill.number=1 task.10.cost=11.0 task.10.skill.0=85 task.10.skill.number=1 task.11.cost=32.0 task.11.skill.0=62 task.11.skill.number=1 task.12.cost=12.0 task.12.skill.0=27 task.12.skill.number=1 task.13.cost=27.0 task.13.skill.0=77 task.13.skill.number=1 task.14.cost=27.0 task.14.skill.0=17 task.14.skill.number=1 task.15.cost=26.0 task.15.skill.0=39 task.15.skill.number=1 task.16.cost=33.0 task.16.skill.0=97 task.16.skill.number=1 task.17.cost=12.0 task.17.skill.0=39 task.17.skill.number=1 task.18.cost=13.0 task.18.skill.0=25 task.18.skill.number=1 task.19.cost=51.0 task.19.skill.0=34 task.19.skill.number=1 task.2.cost=6.0 task.2.skill.0=33 task.2.skill.number=1 task.20.cost=34.0

	task.20.skill.0=34 task.20.skill.number=1 task.21.cost=21.0 task.21.skill.0=16 task.21.skill.number=1 task.22.cost=10.0 task.22.skill.0=79 task.22.skill.number=1 task.23.cost=10.0 task.23.skill.0=44 task.23.skill.number=1 task.24.cost=19.0 task.24.skill.0=30 task.24.skill.number=1 task.25.cost=32.0 task.25.skill.0=17 task.25.skill.number=1 task.26.cost=32.0 task.26.skill.0=44 task.26.skill.number=1 task.27.cost=17.0 task.27.skill.0=38 task.27.skill.number=1 task.28.cost=3.0 task.28.skill.0=47 task.28.skill.number=1 task.29.cost=16.0 task.29.skill.0=40 task.29.skill.number=1 task.3.cost=33.0 task.3.skill.0=83 task.3.skill.number=1 task.4.cost=29.0 task.4.skill.0=62 task.4.skill.number=1 task.5.cost=30.0 task.5.skill.0=51 task.5.skill.number=1 task.6.cost=27.0 task.6.skill.0=18 task.6.skill.number=1 task.7.cost=34.0 task.7.skill.0=32 task.7.skill.number=1 task.8.cost=30.0 task.8.skill.0=65 task.8.skill.number=1 task.9.cost=16.0 task.9.skill.0=50 task.9.skill.number=1 task.number=30

Tabla C.2. Datos del caso de prueba aleatorio utilizando un proyecto de 30 tareas.

C.3. Caso de prueba con un proyecto de 40 tareas

Datos de los empleados **Nota: El costo de los empleados (salary) es redondeado a dos decimales.**	employee.0.salary=32.06642704068693 employee.0.skill.0=50 employee.0.skill.1=56 employee.0.skill.2=63 employee.0.skill.3=69 employee.0.skill.number=4 employee.1.salary=60.82024188134492 employee.1.skill.0=27

	employee.1.skill.1=33 employee.1.skill.2=37 employee.1.skill.3=69 employee.1.skill.number=4 employee.2.salary=65.59761397385385 employee.2.skill.0=7 employee.2.skill.1=36 employee.2.skill.2=41 employee.2.skill.3=43 employee.2.skill.number=4 employee.3.salary=38.569577252936185 employee.3.skill.0=44 employee.3.skill.1=45 employee.3.skill.2=60 employee.3.skill.3=97 employee.3.skill.number=4 employee.4.salary=55.06583897892821 employee.4.skill.0=13 employee.4.skill.1=29 employee.4.skill.2=59 employee.4.skill.3=69 employee.4.skill.number=4 employee.5.salary=40.174615632010216 employee.5.skill.0=17 employee.5.skill.1=54 employee.5.skill.2=59 employee.5.skill.3=97 employee.5.skill.number=4 employee.6.salary=57.684080718570684 employee.6.skill.0=16 employee.6.skill.1=26 employee.6.skill.2=73 employee.6.skill.3=82 employee.6.skill.number=4 employee.7.salary=67.80577378037502 employee.7.skill.0=45 employee.7.skill.1=79 employee.7.skill.2=91 employee.7.skill.3=93 employee.7.skill.number=4 employee.8.salary=87.96311124894521 employee.8.skill.0=19 employee.8.skill.1=37 employee.8.skill.2=64 employee.8.skill.3=99 employee.8.skill.number=4 employee.9.salary=57.527520805554474 employee.9.skill.0=23 employee.9.skill.1=30 employee.9.skill.2=78 employee.9.skill.3=88 employee.9.skill.number=4 employee.10.salary=48.51317146115117 employee.10.skill.0=10 employee.10.skill.1=66 employee.10.skill.2=16 employee.10.skill.3=69 employee.10.skill.number=4 employee.11.salary=66.23545194352506 employee.11.skill.0=20 employee.11.skill.1=32 employee.11.skill.2=44 employee.11.skill.3=54

	employee.11.skill.number=3 employee.12.salary=55.047825234625954 employee.12.skill.0=51 employee.12.skill.1=72 employee.12.skill.2=33 employee.12.skill.3=86 employee.12.skill.number=4 employee.13.salary=64.91412449146308 employee.13.skill.0=70 employee.13.skill.1=48 employee.13.skill.2=63 employee.13.skill.3=93 employee.13.skill.number=4 employee.14.salary=60.57615876837832 employee.14.skill.0=19 employee.14.skill.1=64 employee.14.skill.2=19 employee.14.skill.3=64 employee.14.skill.number=4 employee.15.salary=67.57975726703596 employee.15.skill.0=16 employee.15.skill.1=28 employee.15.skill.2=31 employee.15.skill.3=75 employee.15.skill.number=4 employee.number=16
Datos del grafo de precedencia de tareas	graph.arc.0=1 5 graph.arc.1=2 6 graph.arc.10=9 14 graph.arc.11=10 14 graph.arc.12=0 15 graph.arc.13=1 16 graph.arc.14=3 16 graph.arc.15=4 16 graph.arc.16=8 16 graph.arc.17=13 16 graph.arc.18=3 17 graph.arc.19=5 17 graph.arc.2=3 7 graph.arc.20=3 18 graph.arc.21=14 18 graph.arc.22=1 19 graph.arc.23=12 19 graph.arc.24=3 20 graph.arc.25=10 20 graph.arc.26=13 20 graph.arc.27=14 20 graph.arc.28=2 21 graph.arc.29=15 21 graph.arc.3=5 7 graph.arc.30=18 22 graph.arc.31=15 23 graph.arc.32=11 24 graph.arc.33=18 24 graph.arc.34=21 24 graph.arc.35=11 25 graph.arc.36=12 25 graph.arc.37=16 25 graph.arc.38=21 26 graph.arc.39=25 26 graph.arc.4=2 9 graph.arc.40=2 27 graph.arc.41=15 27

	graph.arc.42=23 27 graph.arc.43=7 28 graph.arc.44=8 28 graph.arc.45=9 28 graph.arc.46=17 28 graph.arc.47=9 29 graph.arc.48=12 29 graph.arc.49=16 29 graph.arc.5=0 12 graph.arc.50=17 29 graph.arc.51=2 30 graph.arc.52=10 30 graph.arc.53=15 30 graph.arc.54=29 30 graph.arc.55=10 31 graph.arc.56=12 32 graph.arc.57=16 32 graph.arc.58=25 32 graph.arc.59=9 33 graph.arc.6=1 12 graph.arc.60=14 33 graph.arc.61=30 33 graph.arc.62=30 35 graph.arc.63=34 35 graph.arc.64=3 36 graph.arc.65=22 36 graph.arc.66=31 36 graph.arc.67=13 37 graph.arc.68=19 37 graph.arc.69=27 37 graph.arc.7=6 12 graph.arc.70=34 37 graph.arc.71=0 38 graph.arc.72=5 38 graph.arc.73=4 39 graph.arc.8=9 13 graph.arc.9=12 13 graph.arc.number=74
Datos de las tareas	task.0.cost=15.0 task.0.skill.0=42 task.0.skill.number=1 task.1.cost=17.0 task.1.skill.0=16 task.1.skill.number=1 task.10.cost=17.0 task.10.skill.0=24 task.10.skill.number=1 task.11.cost=29.0 task.11.skill.0=52 task.11.skill.number=1 task.12.cost=21.0 task.12.skill.0=0 task.12.skill.number=1 task.13.cost=15.0 task.13.skill.0=3 task.13.skill.number=1 task.14.cost=32.0 task.14.skill.0=98 task.14.skill.number=1 task.15.cost=48.0 task.15.skill.0=9 task.15.skill.number=1 task.16.cost=26.0 task.16.skill.0=71

	task.16.skill.number=1 task.17.cost=13.0 task.17.skill.0=11 task.17.skill.number=1 task.18.cost=24.0 task.18.skill.0=22 task.18.skill.number=1 task.19.cost=32.0 task.19.skill.0=83 task.19.skill.number=1 task.2.cost=12.0 task.2.skill.0=94 task.2.skill.number=1 task.20.cost=19.0 task.20.skill.0=33 task.20.skill.number=1 task.21.cost=32.0 task.21.skill.0=99 task.21.skill.number=1 task.22.cost=40.0 task.22.skill.0=42 task.22.skill.number=1 task.23.cost=15.0 task.23.skill.0=66 task.23.skill.number=1 task.24.cost=23.0 task.24.skill.0=22 task.24.skill.number=1 task.25.cost=13.0 task.25.skill.0=35 task.25.skill.number=1 task.26.cost=38.0 task.26.skill.0=43 task.26.skill.number=1 task.27.cost=14.0 task.27.skill.0=62 task.27.skill.number=1 task.28.cost=22.0 task.28.skill.0=15 task.28.skill.number=1 task.29.cost=23.0 task.29.skill.0=13 task.29.skill.number=1 task.3.cost=17.0 task.3.skill.0=68 task.3.skill.number=1 task.30.cost=17.0 task.30.skill.0=67 task.30.skill.number=1 task.31.cost=37.0 task.31.skill.0=16 task.31.skill.number=1 task.32.cost=30.0 task.32.skill.0=36 task.32.skill.number=1 task.33.cost=16.0 task.33.skill.0=17 task.33.skill.number=1 task.34.cost=36.0 task.34.skill.0=97 task.34.skill.number=1 task.35.cost=32.0 task.35.skill.0=50 task.35.skill.number=1 task.36.cost=23.0

	task.36.skill.0=93 task.36.skill.number=1 task.37.cost=31.0 task.37.skill.0=5 task.37.skill.number=1 task.38.cost=47.0 task.38.skill.0=96 task.38.skill.number=1 task.39.cost=14.0 task.39.skill.0=98 task.39.skill.number=1 task.4.cost=12.0 task.4.skill.0=79 task.4.skill.number=1 task.5.cost=22.0 task.5.skill.0=55 task.5.skill.number=1 task.6.cost=30.0 task.6.skill.0=60 task.6.skill.number=1 task.7.cost=19.0 task.7.skill.0=66 task.7.skill.number=1 task.8.cost=27.0 task.8.skill.0=65 task.8.skill.number=1 task.9.cost=16.0 task.9.skill.0=75 task.9.skill.number=1 task.number=40

Tabla C.3. Datos del caso de prueba aleatorio utilizando un proyecto de 40 tareas.

Bibliografía

[Alba2007] E. Alba and J. F. Chicano. Software project management with GAs. *Information Sciences*, 177 (11): 2380-2401, 2007.

[Chicano2011] F. Chicano, F. Luna, A. J. Nebro and E. Alba. Using multi-objective metaheuristics to solve the software project scheduling problem. In *Genetic and Evolutionary Computation Conference* 2011, pp. 1915-1922. ACM, 2011.

[Luna2014] F. Luna, D. González, F. Chicano and M. Vega. The software project scheduling problem: A scalability analysis of multi-objective metaheuristics. *Applied Soft Computing*, 15: 136-148, 2014.

[Jin2014] N. Jin and X. Yao. Heuristic optimization for software project management with impacts of team efficiency. In *2014 IEEE Congress on Evolutionary Computation*, pp. 3016-3023. IEEE, 2014.

[Xiao2013] J. Xiao, X. Ao and Y. Tang. Solving software project scheduling problems with ant colony optimization. *Computers & Operations Research*, 40 (1): 33-46, 2013.

[Gonsalves2010] T. Gonsalves and K. Itoh. Multi-Objective Optimization for Software Development Projects. In *International multiconference of engineers and computer scientists*, 1: 1-6, 2010.

[Dupuy2013] G. Dupuy, N. Stark y C. Salto. Algoritmo evolutivo para el problema de planificación en proyectos de desarrollo de software. In *XVIII Congreso Argentino de Ciencias de la Computación*, pp. 110-119, 2013.

[García2014] A. García-Nájera and M. Gómez-Fuentes. A Multi-objective Genetic Algorithm for the Software Project Scheduling Problem. In *Mexican International Conference on Artificial Intelligence*, pp. 13-24. Springer International Publishing, 2014.

[Ruhe2009] G. Ruhe and A. Ngo-The. Optimized Resource Allocation for Software Release Planning. *IEEE Transactions on Software Engineering*, 35(1): 109-123, 2009.

[Duggan2004] J. Duggan, J. Byrne and G. Lyons. A task allocation optimizer for software construction. *IEEE Software*, 21(3): 76-82,2004.

[Chicano2012] F. Chicano, A. Cervantes, F. Luna and G. Recio. A Novel Multiobjective Formulation of the Robust Software Project Scheduling Problem. In *European Conference on Applications of Evolutionary Computation*, pp. 497-507. Springer Berlin Heidelberg, 2012.

[Espinoza2013] A. Espinoza, R. Garay, A. Martínez, L. Castro. Estudio de Mapeo Sistematizado sobre la Estimación de Valor del Producto de Software. In *Congreso Internacional de Investigación e Innovación en Ingeniería de Software*, pp. 138-145, 2013.

[Jurison1999] J. Jurison. Software project management: the manager's view. *Communications of the Association for Information Systems*, 2(3), 1999.

[Reeves1996] C. Reeves. *Modern heuristic techniques*. John Wiley & Sons, 1996.

[Kelley1961] J. Kelley. Critical-path planning and scheduling: Mathematical basis. *Operations Research*, 9(3): 296-320, 1961.

[Jacobson2000] I. Jacobson, G. Booch and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 2000.

[Deb2002] K. Deb, A. Pratap, S. Agarwal and T. Merarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182-197, 2002.

[Zitzler1999] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4): 257-271, 1999.

[Zitzler2001] E. Zitzler, M. Laumanns and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. *Eurogen*, 3242(103): 95-100, 2001.

[Beume2007] N. Beume, B. Naujoks and M. Emmerich. "SMS-EMOA: Multiobjective selection based on dominated hypervolume". *European Journal of Operational Research*, 181(3): 1653-1669, 2007.

[Durillo2011] J. J. Durillo y A. J. Nebro. jMetal: a Java Framework for Multi-Objective Optimization. *Advances in Engineering Software*, 42(10): 760-771, 2011.

[Sommerville2005] I. Sommerville. *Software Engineering 9*. Addison Wesley, 2006.

[Torres2014] Z. Torres y H. Torres. *Administración de proyectos*. Grupo Editorial Patria, 2014.

[Pressman2005] R. Pressman. *Software Engineering: A practitioner's approach*. Mc Graw Hill, 2005.

[Bass2012] L. Bass, P. Clements and R. Kazman: *Software Architecture in Practice*. Addison Wesley, 2012.

[Kruchten1995] P. Kruchten. Architectural Blueprints—The "4+1" View Model of Software Architecture". In *Tri-Ada Tutorials* 95: 540-555, 1995.

[Cervantes2016] H. Cervantes y R. Kazman. *Designing Software Architectures: A Practical Approach*. Addison Wesley, 2016.

[Ehrgott2000] M. Ehrgott. *Multicriteria Optimization.* Springer, 2000.

[Humphrey1997] W. Humphrey. *Introduction to the Personal Process Software.* Addison Wesley, 1997.

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

PC y TI

UNIVERSIDAD AUTÓNOMA METROPOLITANA - IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

PLANEACIÓN DE PROYECTOS DE DESARROLLO DE SOFTWARE USANDO TÉCNICAS DE OPTIMIZACIÓN

Tesis que presenta
Miguel Ángel Vega Velázquez
Para obtener el grado de
Maestro en ciencias y tecnologías de la información

Asesores:	Dr. Humberto Gustavo Cervantes Maceda
	Dr. Abel García Nájera
Jurado calificador:	
Presidente:	Dr. Antonio López Jaimes
Secretario:	Dr. Humberto Gustavo Cervantes Maceda
Vocal:	M.C. Víctor Hugo Gómez Gómez

Ciudad de México, Enero 2017