
División de Ciencias Básicas e Ingeniería

Unidad Iztapalapa

Posgrado en Ciencias y Tecnologías de la Información

# Aplicación de un algoritmo genético multiobjetivo para la replaneación de liberaciones en proyectos ágiles de software

Idónea Comunicación de Resultados para obtener el grado de

Maestro en Ciencias y Tecnologías de la Información

Presenta:

**Victor Hugo Escandon Bailon**

Asesores:

**Dr. Abel García Nájera**

**Dr. Humberto Cervantes Maceda**

Sinodales:

**Dr. Saúl Zapotecas Martínez**

**Mtro. Víctor Hugo Gómez Gómez**

**Ciudad de México, 6 de Diciembre del 2019**

## Agradecimientos

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

A la Universidad Autónoma Metropolitana y en especial a la unidad Iztapalapa, la cual ha sido mi segunda casa durante la licenciatura y ahora en esta etapa de maestría, gracias a esta gran casa de estudios por darme la oportunidad de crecer personal, académica y profesionalmente.

A mis excelentes asesores, el Dr. Humberto Cervantes Maceda y el Dr. Abel García Nájera por todo el esfuerzo y el interés que dedicaron siempre al proyecto, gracias por sus consejos, su guía y sus valiosas observaciones que realizaron a lo largo de este grado.

A mis grandiosos compañeros del PCYTI por todo su apoyo y por compartir sus conocimientos, así como el gran equipo que formamos dentro y fuera de la cancha.

A mis grandes amigos de la UAM-I que tanto hemos compartido, a los que están y a los que tristemente se fueron.

A mis increíbles hermanos Michel Escandon y Erick Escandon que siempre han estado ahí para apoyarme y se que puedo contar con ellos en todo momento.

A mis queridos padres Beatriz Bailon y Victor Escandon por todo su amor, su apoyo, su esfuerzo, su motivación, su ejemplo y sus consejos, en cada uno de los días desde hace casi 30 años.

A mi hermosa esposa Zulema Torres que juntos acabamos de comenzar la hermosa aventura de ser padres, gracias a ella por todo su apoyo, su amor, su dedicación y comprensión, por último, a mi hijo Victor Santiago que llego a llenar completamente mi vida.

# Contenido

**4 Modelo desarrollado**



**5 Algoritmo genético multiobjetivo para resolver el RLPAS**



**6 Experimentos y Resultados**



**7 Conclusiones**

# Capítulo 1

# Introducción

La industria del software en México ha crecido notablemente y de hecho, según “El Economista” con datos de la consultora Gartner [20], en este año 2019 las empresas que operan dentro del territorio invertirán 7,831 millones de dólares en servicios de TI. Lo anterior, en comparación con el gasto en 2018 supone un incremento de 4.5%. Gartner prevé que para el 2020 las empresas crezcan su gasto en 5.4% lo que se traduce en 8,261 millones de dólares. Según los datos de esta consultora, México se encuentra en el segundo lugar en latinoamérica, en cuanto a gastos de servicios de TI se refiere.

Debido a lo anterior los salarios en este ámbito son altos y de acuerdo con la revista Software Guru [19] la media del salario es de 33 mil pesos. Donde los líderes de proyectos ganan de los 39 mil pesos hasta los 44 mil pesos. El salario de los desarrolladores va de los 32 mil a los 35 mil pesos. Con estos datos podemos darnos cuenta que el desarrollo de proyectos de software es caro y que en México esta industria ha crecido importantemente en los últimos años.

Con tan alta demanda en proyectos de software y tanto dinero invertido en ello, se espera que el software desarrollado sea de calidad. Para lograr esta calidad que se espera de un sistema, se utiliza la ingeniería de software (IS), que según la IEEE, es la aplicación de un enfoque sistemático, disciplinado, y cuantificable al *desarrollo*, operación y mantenimiento del software [21].

Aunado a la IS encontramos la administración de proyectos, que es una metodología usada a nivel mundial, por empresas e instituciones para alcanzar objetivos en un tiempo determinado y que de acuerdo con el Project Management Institute (PMI) [24] se compone de 5 fases: Inicio, Planeación, Ejecución, Control y Conclusión. Este trabajo se centra en la fase de ejecución. En esta fase, es en la que se ejecuta el *plan inicial*, resultado de la fase de *planeación*. En la ejecución se desarrollan los sistemas y es a lo largo de esta, donde ocurre la replaneación.

En el desarrollo de proyectos de software, el problema de planificación de proyectos ha sido ampliamente estudiado. Este problema se conoce como Problema de Planeación de Proyectos de Software (o SPSP, por sus siglas en inglés), se ubica en la fase de planeación y con él se busca encontrar una propuesta de planeación que permita terminar lo antes posible y con el mínimo costo un proyecto. Como resultado se obtiene una planeación inicial. Pero, durante la ejecución de este plan, suelen ocurrir *eventos disruptivos*. Les llamaremos eventos disruptivos a estos eventos que por su gravedad nos obligan a modificar el plan inicial,  tales como la introducción de un nuevo requerimiento,

cuando un empleado se va del proyecto, etc. Que por sus consecuencias nos obliguen a realizar adecuaciones al plan inicial, para poder cumplir con los contratos de tiempo y costo establecidos.

## 1.1 Planeación de proyectos de software

El problema de planificación de proyectos de software en las metodologías tradicionales, trata de dar una propuesta de planeación la cual minimice el costo y el tiempo del desarrollo del mismo. Para lograr esto se busca encontrar la mejor asignación de empleados a las tareas del proyecto, tomando en cuenta la experiencia, el salario y capacidad de los empleados, y considerando, además, el tiempo de estimación y requerimientos necesarios en las tareas. Como resultado se produce una (o varias) propuestas de plan de proyecto.

A mediados de los 90's en los tiempos de los métodos muy estructurados y estrictos ocupados en el desarrollo de software, se presentó una reacción contra estos. En esta época se adoptó el uso de las metodologías ágiles, en las cuales se busca que los requerimientos en el desarrollo de proyectos de software se adapten a los cambios y que el desarrollo como su nombre lo indica sea más ágil. De igual manera se busca que los equipos de desarrollo en este tipo de metodologías sean autodirigidos y multidisciplinarios, es decir más ágiles. En este trabajo nos enfocaremos en la metodología ágil Scrum, la cual nació desde 1986 y que lleva ya más de 20 años usándose [12] .

Podemos considerar dos tipos de enfoque en el uso de Scrum, el primero es una versión "muy ágil" que se caracteriza por su alta capacidad de adaptación al cambio. En este enfoque, esencialmente se contrata a un equipo de desarrollo y en cada sprint se decide lo que se van a desarrollar y no forzosamente existe un plan de liberaciones. Este enfoque tiene la ventaja de poder cambiar de ruta muy rápidamente y es ideal por ejemplo para la innovación, sin embargo, no se puede estimar el tiempo y costo del proyecto ya que no se establece un alcance particular.

El segundo enfoque, que es el que consideramos en este trabajo, es un enfoque más "tradicional", ya que en este desde el principio hay una idea de los requerimientos del sistema. En este enfoque si es posible estimar el tiempo y costo del proyecto. En este enfoque, sin embargo, la planeación no se realiza de la misma manera que en una metodología tradicional, ya que en este contexto el software se desarrolla y se entrega al cliente en incrementos llamados *liberaciones* [18]. Cada liberación se compone de uno o varios *sprints*. Las liberaciones duran apenas unas semanas por lo que las planeación no se realiza solo una vez, sino en cada una de ellas. En las liberaciones, se realiza la asignación de historias de usuario a los sprints. Al realizar una planeación en el contexto de Scrum también se obtiene un plan, el cual se conoce como plan de liberaciones.

## 1.2 Replaneación de liberaciones

Una vez que se ha definido la planeación de liberación y durante la ejecución de un proyecto, como se mencionó previamente, durante la ejecución de un plan, pueden ocurrir eventos disruptivos que por su gravedad nos obliguen a realizar cambios al plan inicial. Aún si Scrum busca ser una metodología ágil que se ajusta al cambio, estos eventos pueden afectar gravemente el plan de liberación. Estos cambios deben realizarse rápidamente ya que existen fechas de entrega y costos definidos en el proyecto. Es conveniente que al realizar el ajuste al plan inicial se realice con el menor número de cambios ya que, generalmente cuando se realiza un plan de liberación inicial, éste se realiza considerando las mejores decisiones para el proyecto. A la acción de ajustar o actualizar el plan original después de un evento disruptivo, se le conoce como *replaneación*. Una replaneación se realiza con el fin de generar un nuevo plan que se ajuste a los cambios que introduce el evento disruptivo buscando minimizar las afectaciones que este puede causar e intentar mantener los objetivos originales del plan, que son minimizar el tiempo y costo del desarrollo del proyecto.

Un ejemplo de evento disruptivo es cuando un empleado se va del proyecto, cuando esto sucede perdemos capacidad de desarrollo, por lo que el plan inicial ahora está obsoleto, ya que las tareas a las que estaba asignado este empleado ya no podrán ser completadas de la forma que se había planeado.

En este trabajo nos enfocaremos en el problema de replaneación en las metodologías ágiles, específicamente la replaneación de proyectos que se administran usando Scrum. En la actualidad muchas organizaciones han adoptado el desarrollo de proyectos de software que se administran con Scrum y basta con revisar las vacantes y proyectos de la industria, donde Scrum es la metodología que más aparece.

En las liberaciones se busca desarrollar y entregar funcionalidad del sistema que aporte valor al negocio del cliente de forma rápida, entregando un producto ejecutable en cada una de ellas. Por ejemplo, si se desarrolla una tienda en línea, como primer liberación se podría entregar el catálogo de productos, el carrito de compras y el pago con tarjeta. Al entregar esta funcionalidad, el cliente puede comenzar a vender sus productos y por lo tanto obtener valor de su negocio, aún si otras funcionalidades que desea aún no están disponibles.

Se debe señalar que en los proyectos administrados con Scrum los tiempos de entrega de funcionalidad del proyecto tienden a ser más cortos en comparación con el desarrollo tradicional. En Scrum una liberación dura apenas unas semanas, para realizar una liberación, al inicio del proyecto se realiza una planeación que al igual que en las metodologías tradicionales, se realiza con el fin de minimizar el tiempo y costo del desarrollo, así como decidir qué es lo mejor para la liberación. En Scrum una *historia de usuario* (HU) es la descripción de una funcionalidad del sistema en lenguaje común y es

la forma en que se describen a los requerimientos del sistema. A diferencia de las metodologías tradicionales, durante la planeación de liberaciones no se realiza una asignación de los empleados a las HU y son ellos los que al momento de comenzar un sprint deciden que HU desarrollar. Por lo que al momento de realizar la planeación de una liberación o su replaneación, solo se considera la asignación de HU al sprint que mejor convenga al desarrollo del proyecto.

## 1.3 Problema de replaneación de proyectos de software

Debido a que el desarrollo de software es costoso y que, en general, en un proyecto de software las fechas de entrega son difíciles de cambiar, los administradores de proyectos deben realizar de inmediato una replaneación que se ajuste a las condiciones ante un evento disruptivo, con el fin de minimizar los impactos económicos y operativos, y cumpliendo con las fechas de entrega.

Además de los criterios de tiempo y costo, podemos identificar al menos otros tres criterios importantes que se deben considerar cuando se realiza una replaneación de liberación:

- **Estabilidad**, se desea que al realizar una replaneación ésta no difiera demasiado de la original, ya que se considera que la asignación de las historias de usuario en los distintos sprints de la planeación original, fue pensada cuidadosamente para ser la mejor opción para el desarrollo del proyecto.
- **Valor de liberación**, se busca que las historias de usuario de mayor prioridad para la liberación se asignen a los primeros sprints, y así desarrollar primero las HU que aportan más valor al negocio.
- **Desaprovechamiento de la capacidad de desarrollo**, en el desarrollo de software los equipos son costosos y se debe usar de la mejor manera el tiempo de desarrollo de los empleados para evitar que se desaproveche.

Por lo anterior, al realizar una replaneación de liberación consideramos cinco criterios. Es decir, se consideran cinco objetivos fundamentales: tiempo de duración del proyecto, costo del mismo, estabilidad entre planeaciones, valor de liberación y desaprovechamiento de la capacidad de desarrollo.

## 1.4 Problema de optimización multiobjetivo

Este tipo de problemas, como su nombre lo indica son aquellos problemas en donde se busca optimizar más de un objetivo. Al estar formados por varios objetivos, se busca encontrar los mejores valores para cada uno de los objetivos y combinaciones de valores entre varios objetivos. Estas combinaciones deben presentar valores aceptables para proponer soluciones para el problema que se busca optimizar. Por lo

anterior, en los problemas de optimización multiobjetivo es común que el resultado no sea solo uno, sino un conjunto de propuestas de solución. En este tipo de problemas de optimización es normal que los objetivos estén en conflicto. Si dos objetivos están en conflicto, quiere decir que si se mejora un objetivo, el otro se verá afectado.

Dado que al generar una replaneación de liberación se busca obtener un valor óptimo en cada uno de los objetivos, el problema de replaneación de liberaciones se puede modelar como un problema de optimización multiobjetivo. En este caso, lo anterior se traduce en que ante un evento disruptivo se pueden identificar varias propuestas de solución.

Un ejemplo de conflicto en este trabajo en particular, es el que existe entre los objetivos de tiempo y costo ya que si, por ejemplo, queremos minimizar el tiempo de entrega (realizar la liberación más rápido), seguramente esto requeriría de contratar más personal. Lo anterior se traduciría en que, el costo del proyecto aumentaría, debido a que ahora se debe pagar a más personas. Este es solo un ejemplo pero de hecho, los conflictos entre objetivos pueden involucrar varios de ellos. Siguiendo con el ejemplo del costo, este también es afectado si el desaprovechamiento del equipo de desarrollo es alto. Si se desaprovecha mucho el tiempo, el proyecto tardará más (conflicto con tiempo) y por lo tanto habrá que pagar más tiempo de desarrollo del equipo (conflicto con el costo).

## 1.5 Técnicas heurísticas

Debido a la forma en que en este trabajo se modela el problema de replaneación y al ser este una generalización del problema de asignación [22], el cual se encuentra en la categoría de NP-Difícil [23], no existe un algoritmo que resuelva todos los casos del problema en un tiempo aceptable. Debido a esto se propone un algoritmo heurístico.

La palabra heurística proviene del griego “euriskein” y quiere decir encontrar. Los algoritmos heurísticos buscan encontrar soluciones muy cercanas a las óptimas en problemas difíciles (intratables) [1]. En muchos problemas de optimización no existe una única solución óptima, sino más bien un conjunto de soluciones óptimas. Para resolver un problema de este tipo con un método exacto, por lo regular necesitaría de mucho tiempo e incluso años ejecutándose en una computadora para encontrar la mejor solución. A diferencia de las técnicas exactas, las heurísticas no garantizan encontrar soluciones óptimas, pero sí permiten obtener buenas soluciones en un tiempo razonable para un problema particular [2].

## 1.6 Objetivos del trabajo de investigación

Debido a que el desarrollo de software con Scrum actualmente ha sido adoptado por muchas empresas y que no se encontró ninguna herramienta que sirva de apoyo. En este trabajo se propone una solución, para lo cual se definieron los siguientes objetivos.

### 1.6.1 Objetivo general

El objetivo general de este trabajo de investigación es:

- Desarrollar un algoritmo heurístico para el problema de replaneación de liberaciones en proyectos ágiles de software

### 1.6.2 Objetivos específicos

Para lograr el objetivo general de este trabajo, se consideran los siguientes objetivos específicos:

1. Identificar las características específicas de la planeación de liberaciones usando Scrum.
2. Definir objetivos en el contexto de la planeación de liberaciones en Scrum.
3. Definir un modelo matemático para el problema.
4. Desarrollar un algoritmo heurístico que cubra la necesidad de obtener de manera rápida un conjunto de propuestas de replaneación de liberaciones, después de ocurrir un evento disruptivo.
5. Analizar el desempeño del algoritmo desarrollado mediante la solución de casos de prueba.

## 1.7 Aportaciones

Las aportaciones que se realizaron con este trabajo son las siguientes:

- Se propusieron un modelo matemático y un modelo orientado a objetos del problema de replaneación de liberaciones, que consideran características específicas del desarrollo de proyectos de software con Scrum.
- Se desarrolló un algoritmo genético multiobjetivo que da como solución un conjunto de propuestas de replaneación después de un evento disruptivo.
- Se implementó una herramienta que facilita la tarea de la replaneación de liberaciones. Esta herramienta muestra los resultados del conjunto solución de una manera fácil de interpretar por medio de diagramas.

- Este trabajo se presentó en el XI Congreso Mexicano de Inteligencia Artificial[1].

## 1.8 Estructura del documento

El presente trabajo se estructura de la siguiente manera: En el capítulo 2 se presenta el estado del arte, donde al final se hace un análisis de la revisión de la literatura. En seguida, en el capítulo 3, se detallan las características de la planeación de liberaciones en Scrum. En el capítulo 4 se presenta el modelo desarrollado específicamente para este problema. El capítulo 5 describe nuestra propuesta de solución. Los experimentos que se llevaron a cabo y los resultados obtenidos se especifican y analizan en el capítulo 6. Por último, en el capítulo 7 se presentan las conclusiones del presente trabajo.

[1] V. H. Escandon, H. Cervantes, A. García Nájera, “Aplicación de un algoritmo genético multiobjetivo para la replaneación de liberaciones en proyectos ágiles de software”, XI Congreso Mexicano de Inteligencia Artificial, Junio 2019.

# Capítulo 2

# Estado del arte

Para resolver el problema de replaneación de proyectos de software y en especial la replaneación de liberaciones, es necesario utilizar técnicas heurísticas de optimización ya que al ser una generalización del problema de asignación, se considera un problema NP-Difícil. En la literatura encontrada se mostró que con una heurística y un modelo adecuados se pueden obtener soluciones congruentes y que se asemejan a las decisiones que tomaría un experto en proyectos de software reales.

## 2.1 Revisión bibliográfica

Con el fin de entender el problema de replaneación de proyectos de software desarrollados en el contexto de la etapa de planeación de liberación con Scrum y de identificar aspectos importantes a considerar en las propuestas de solución con heurísticas, se realizó una búsqueda de artículos relacionados que dio como resultado un grupo de 10 artículos que fueron revisados. Cabe resaltar que el problema de replaneación es un problema que actualmente está siendo investigado y que en la revisión de trabajos no se encontró ninguno que tome en cuenta las características de Scrum, ni de ninguna otra metodología ágil. A continuación se presentan el resumen de cada artículo, el análisis de la revisión realizada y una tabla comparativa entre ellos, los cuales ayudarán posteriormente a presentar el estado del arte y las conclusiones.

- **Software Project Rescheduling with Genetic Algorithms [9]**

El objetivo del artículo es modelar el problema de replaneación de proyectos de software que siguen un enfoque tradicional. Los autores mencionan que en el año 2009 que fue escrito el artículo no había investigación de optimización del problema de replaneación. Mencionan que el problema se considera un problema NP-Completo por lo tanto decidieron usar un algoritmo genético para resolverlo, ya que es el que mejor se ajusta al modelo de asignación de empleados a tareas. La solución y modelado del problema se realizaron de la siguiente manera:

- ➢ Las tareas se modelan con un vector y la dependencia entre ellas con un grafo de precedencia.
- ➢ Los empleados tienen una lista de habilidades que poseen, su salario y sus cargas máximas de trabajo.

- ➢ Un empleado puede ser asignado a la tarea si cumple con todas las habilidades necesarias para realizarla.

Se busca minimizar la función objetivo, la cual está compuesta de la eficiencia y la estabilidad.

$$F.O. = Ws * Estabilidad + Wu * Eficiencia$$

La eficiencia se refiere a la duración mínima del proyecto y el costo monetario minimo para realizarlo en base a la asignación del personal y su salario. La eficiencia se compone de:

- ➢ El costo es lo que se debe pagar a los empleados utilizando el salario y las horas que aportan a las tareas.
- ➢ El Tiempo es el tiempo total requerido para finalizar el proyecto.

La estabilidad es la diferencia que existe entre la replaneación y la planeación, se penaliza que exista una gran diferencia. La penalización de estabilidad se obtiene calculando la diferencia de los tiempos de finalización entre la planeación inicial y la planeación nueva. Para ello se toma en cuenta el total del tiempo actual, es decir, contando los retrasos y por último dividiendo la diferencia anterior entre el total de tareas para normalizar el valor.

Para el modelado del cromosoma en el algoritmo genético los autores usaron una matriz de asignación, donde cada fila representa el empleado, la columna representa las tareas y el valor de la celda es el porcentaje de tiempo que dedicará el empleado a la tareas. La Tabla 1 muestra un ejemplo de la matriz de asignación.

Tabla 1. Matriz de asignación

| | Tarea 1 | Tarea 2 | Tarea 3 | Tarea 4 |
|---|---|---|---|---|
| Empleado 1 | 0 | 0.25 | 1 | 0.5 |
| Empleado 2 | 0.5 | 0.5 | 0 | 0 |

Los experimentos que se realizaron como primera parte, tuvieron como objetivo obtener un valor que mejor describiera la penalización de estabilidad, a este coeficiente se le nombró K y se le dio un valor de 1000. Como segunda parte simularon una replaneación con distinto número de tareas nuevas (4,8,12) y variando el parámetro de estabilidad. En el experimento anterior se obtuvo que con un número menor de tareas, la afectación a la estabilidad es mayor. En la tercera parte de los experimentos se mostró como la llegada de nuevos recursos al proyecto no apresura demasiado el proyecto cuando la estabilidad es distinta de cero y que, como es de esperarse, el costo del proyecto aumenta.

Como conclusión los autores sostienen que este trabajo puede servir como apoyo a los administradores de proyectos en la toma de decisiones al realizar una replaneación, después de ocurrir eventos disruptivos en los proyectos de software desarrollados con metodologías tradicionales.

- **Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering [14]**

En este trabajo los autores buscan modelar el impacto de la incertidumbre en la planeación de proyectos de software y desarrollar un sistema que dé como resultado, una planeación con un grado de robustez que permita al proyecto ser tolerante a incertidumbre en los tiempos de estimación de la duración de las tareas, así como para la llegada de nuevas tareas.

El problema al planificar es que el administrador de proyecto no puede ser demasiado conservador ya que al estimar más tiempo, que el de la duración real de las tareas, el proyecto se vuelve más costoso, pero si no se toman en cuenta los riesgos, el proyecto puede desbordarse en tiempo fácilmente al ocurrir algún evento disruptivo. Los autores buscaron desarrollar una herramienta que ayude a explorar el espacio de posibles soluciones, que minimizan el tiempo de entrega mientras que muestran un grado alto de tolerancia a fallos (retrasos en el plan), de modo que los eventos disruptivos no lleven al desbordamiento en la planeación.

Cuando se busca optimizar los dos objetivos anteriores se encuentra claramente que están en conflicto, ya que mientras más robusto sea el proyecto, este será más flexible y no tendrá la fecha de terminación más temprana (tardará más en desarrollarse). Sin embargo se puede buscar una solución con una pareja de valores que sean buenos para ambos objetivos y por eso proponen una función multiobjetivo. Como solución eligieron un algoritmo genético multiobjetivo (MOGA), que encuentre la mejor solución, es decir, el mejor par de valores para la robustez y tiempo de finalización. En el modelo se toman en cuenta dos tipos de eventos disruptivos que pueden presentar incertidumbre:

- La llegada de nuevas tareas
- Demora en la finalización de tareas existentes.

Los autores decidieron construir una función multiobjetivo con tres objetivos a optimizar:

- Minimizar el tiempo total de finalización
- Minimizar la robustez, para ello se simulan tareas al azar, que se agregan al plan y así se minimiza, ya que al tomarse en cuenta tareas que no existen en el plan real se busca que la planeación sea más robusta.
- Minimizar la diferencia en la fecha de terminación de las tareas, para este punto los autores inflan la duración de algunas tareas en el proyecto.

El cromosoma que presentan es un genoma que consta de dos matrices, la primera representa el orden de las tareas, la segunda modela la asignación de equipos a las tareas.

Para el cruce toman un punto único y el operador de mutación simplemente selecciona al azar dos posiciones en el conjunto y las intercambia.

La configuración de los parámetros del algoritmo genético fue la siguiente: La probabilidad de mutación se estableció en 0.1. La probabilidad de cruce se estableció en 0.8. Elitismo de 10, tamaño de la población de 250, 180 generaciones para el proyecto A y B, 250 generaciones.

En los experimentos los autores buscan demostrar el conflicto entre la robustez y el tiempo de finalización, así como cuál de los dos tipos de incertidumbre tiene una afectación mayor en el proyecto. Para realizar los experimentos se tomaron cuatro proyectos reales a los cuales se les aplicó el algoritmo.

En los resultados demuestran el costo tan alto que implica una robustez total y como se comporta de forma inversa al tiempo de finalización, el análisis sirve como guía para los administradores de proyectos, concluyen que con pocos proyectos no pueden deducir cual incertidumbre es más “peligrosa” para el proyecto.

- **Dynamic Resource Scheduling in Disruption-Prone Software Development Environments [11]**

El artículo habla de la importancia de realizar una buena asignación de recursos ya que debe garantizar que los recursos humanos asignados tengan la competencia y capacidad necesaria para realizar la tarea. Como objetivos principales se busca optimizar el costo, que la eficiencia del proyecto se maximice y que la satisfacción del cliente aumente. Mencionan que los eventos disruptivos como los cambios en los requisitos, corregir errores importantes, y la rotación del personal pueden crear incertidumbre que complica la asignación de recursos. El problema es cómo equilibrar la necesidad de responder de manera efectiva a los eventos disruptivos con la necesidad de asegurarse que la respuesta no cree otras perturbaciones a la planeación. El artículo propone 4 preguntas clave para la replaneación:

1. ¿Bajo qué circunstancias debe ejecutarse una replaneación?
2. ¿Qué actividades deberían ser contempladas en la replaneación?
3. ¿Qué tipo de medidas se pueden usar para ayudar a encontrar un equilibrio entre resolver las interrupciones actuales y evitar la creación excesiva de nuevas interrupciones al hacerlo?
4. ¿Qué tipo de algoritmo debe usarse para resolver el problema de la replaneación?

En este artículo los autores proponen una función multiobjetivo, pero al final suman todos los objetivos para obtener un solo valor, para obtenerlo la función pondera que tan bien la replaneación aborda la interrupción (utilidad) y que tan poco crea nuevas interrupciones (estabilidad), como algoritmo para la solución se seleccionó un algoritmo genético. Los eventos disruptivos que se toman en cuenta son:

- ➢ Cambios en los requerimientos: los requerimientos cambian continuamente durante las ejecuciones del proyecto, para abordar estos cambios, se pueden insertar nuevas actividades en los procesos de desarrollo, que requieren asignar recursos a estas actividades.
- ➢ Llegada repentina de actividades urgentes: pueden ser necesarias nuevas actividades para abordar problemas urgentes (por ejemplo, errores graves en el software entregado), los cambios que requieren pueden causar interrupciones en el cronograma o el costo.
- ➢ Desviaciones en la ejecución del proceso: inexactitudes en las estimaciones de costos del proyecto, el desempeño incorrecto por parte del personal del proyecto, la necesidad inesperada de reelaboración o la ocurrencia de excepciones al proceso, pueden causar que el proyecto no avance como se planeó, requiriendo la replaneación de los recursos del proyecto.
- ➢ Rotación de personal: la industria del software experimenta una gran movilidad de personal y rotación de personal que crea interrupciones que generalmente requieren reprogramación

El método de solución propuesto es:

1. Identificar el o los eventos disruptivos y los cambios que deben realizarse para solucionar este evento, la eliminación de tareas y la llegada o supresión de recursos, son eventos que activan la replaneación.
2. Se deben delimitar las actividades y recursos que deben cambiarse y que se incluirán en la replaneación, ya que una interrupción no necesariamente afecta todo el proyecto.
3. Se modelan y construyen las restricciones y objetivos de valor para la replaneación.
4. Se ejecuta el algoritmo genético para optimizar la función objetivo.

Los autores definen un proyecto como:

***Proyecto P = (BasicAttr ,ConSet, PWSet)***, donde:

- ➢ ***BasicAttr*** describe atributos básicos como nombre, generación, descripciones, etc.
- ➢ ***ConSet*** es el conjunto de todas las restricciones de nivel de proyecto (por ejemplo, costo y tiempo). La violación de cada restricción incurrirá en una penalización.
- ➢ ***PWSet*** es un conjunto de peso de prioridad. A cada proyecto en un entorno de proyectos múltiples se le asigna un peso de prioridad relativo a los otros proyectos. Este peso se usa para evaluar la importancia de las necesidades de recursos en cada proyecto

Para que un recurso pueda ser asignado a una actividad debe cumplir con las siguientes características:

- ➢ El recurso cuenta en su conjunto de actividades con el tipo de actividad que se requiere.
- ➢ El recurso cuenta con las habilidades requeridas para la tareas.
- ➢ El recurso tiene un nivel de habilidad superior al requerido por la tarea.

La función objetivo se describe de la siguiente manera:

$$F.O. = Ws * Estabilidad + Wu * Utilidad$$

Donde el valor de la estabilidad se refiere a la importancia de mantener el proceso replanteado similar al proceso inicial. El valor de utilidad muestra la importancia de responder a la interrupción, es decir la utilidad en dinero de lo que se obtendrá al final del proyecto. $W_s$ y $W_u$ son factores de peso para cada uno de ellos. La estabilidad se mide utilizando dos factores:

1. El cambio en la planeación de cada actividad: cambios en la planeación y entrega de las actividades pueden reducir el compromiso del proyecto y la satisfacción del cliente.
2. El cambio en las asignaciones de recursos a cada una de las actividades: cambios en la asignación de recurso pueden requerir más esfuerzo de comunicación, más tiempo de capacitación y desperdicio de preparaciones previas.

La desviación del cronograma se mide por las diferencias entre la planeación inicial y las horas de inicio y finalización del proceso replaneado. La utilidad describe el valor obtenido de un proyecto, osea que describe el beneficio o penalización que tiene terminar antes o después respectivamente un proyecto.

Para representar el cromosoma en el algoritmo genético se usó un vector de asignaciones (matriz aplanada) más una matriz de prioridades, como se muestra:

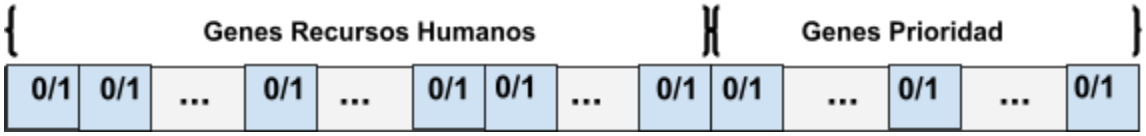



Fig. 2. Estructura del cromosoma

Para los experimentos presentados en el artículo los autores modelaron la llegada de un nuevo requerimiento y de dos correcciones de errores a una planeación inicial, los resultados solo demostraron que su solución se ajusta bien a los factores de estabilidad y utilidad, donde si la estabilidad no se toma en cuenta la utilidad mejora pero la empresa pierde en la capacitación del personal, mencionan que su trabajo puede servir como guía para los administradores de proyectos de software para la toma de decisiones en la asignación de recursos.

- **Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler [5]**

En este trabajo, los autores Chen & Zhang mencionan que para resolver el problema de planeación deben usar un enfoque basado en eventos para generar un plan más estable, en su trabajo también buscan solucionar la asignación de personal, toman en cuenta tres eventos:

- ➢ El momento inicial del proyecto (planeación)
- ➢ Cuando los recursos se liberan al terminar una tarea
- ➢ El momento en que los empleados se unen o abandonan un proyecto

En comparación con otros modelos, ellos presentan su modelo más flexible ya que permite la asignación de un recurso a distintas tareas simultáneamente, esto dota de flexibilidad al proyecto ya que además de resolver la planeación también toman en cuenta la asignación de personal a las tareas.

Como solución proponen un modelo que consta de un esquema para la planificación basada en eventos junto con un esquema de representación que se compone de una lista de tareas que define la prioridad y una matriz de asignación de empleados a las tareas la cual indica la carga de trabajo.

Se menciona que al mantener el esquema simple con una matriz 2D, el espacio de búsqueda es menor y por lo tanto se acelera el proceso, en comparación con algoritmos que usan matrices de tres dimensiones.

La selección de un algoritmo de colonia de hormigas (ACO por sus siglas en ingles) se realizó por que es una metaheurística que va construyendo soluciones paso a paso ya que con una buena heurística se puede ir guiando a las hormigas a las tareas de mayor prioridad tan pronto como sea posible, así como asignar esas tareas a los empleados más adecuados para trabajar en ellas, por lo tanto ACO converge rápidamente y funciona bien en el problema considerado.

En este artículo los autores también toman en cuenta que una tarea no tenga sobrecarga de comunicación, ya que una tarea puede ser desarrollada por más de una persona, los autores estiman un máximo de empleados para resolverla y así mantener ligera la comunicación.

Como función objetivo los autores usan una función compuesta, donde buscan minimizar el costo total del proyecto:

$$min\ F\ :\ \Sigma\ salario\ +\ \Sigma\ penalización$$

donde el salario se refiere a el pago de los empleados dependiendo de las horas de trabajo, hay dos tipos de salario: salario normal y salario para horas extras. La penalización hace referencia a el costo que tiene no terminar una tarea a tiempo (deadline).

Como antes se mencionó, los autores se basan en eventos, de los cuales se supone que se conoce desde el principio lo que va a ocurrir, a diferencia de los sistemas dinámicos ellos realizan el plan inicial tomando en cuenta todos los eventos que sucederán, por lo tanto nunca existe una replaneación, los eventos que consideran son:

- $t = 1$ es el comienzo del proyecto.
- Cualquier empleado se une o abandona el proyecto en *t*.
- Cualquier tarea acabada en el período de tiempo anterior y los recursos correspondientes se liberan y están disponibles en *t*.

Para construir una solución cada hormiga debe construir la lista de tareas y la matriz de asignación, cada hormiga comienza con la lista de tareas para la cual se probaron dos tipos de feromonas:

- La primera es al poner una tarea *t* en la posición *k* de la lista, la feromona se usa para obtener un total de las sumas de las feromonas en esa posición.
- La segunda se refiere a dos tareas consecutivas y revisar si la feromona indica si es una buena solución.

Para garantizar un espacio de búsqueda más amplio las hormigas no se guían completamente por la feromona ya que en el algoritmo se usa una probabilidad para aplicar la feromona o realizar una asignación al azar.

Para la selección de la prioridad de las tareas se mide la holgura que tiene cada una de ellas y se seleccionan las tareas de la ruta crítica, para la selección de empleados a cada una de las tareas se toma en cuenta la competencia del empleado para la tarea y se divide entre su salario, así se selecciona al empleado con más beneficio en competencia/costo, esto ayuda a minimizar el costo total.

Para la matriz de asignación se uso una carga de trabajo en porcentaje, para la construcción de la solución de esta matriz el algoritmo funciona con dos feromonas:

- La primera sirve para seleccionar si el empleado es asignado o no a la tareas
- La segunda sirve para determinar el porcentaje de carga de trabajo que este dedicara a las tareas.

Para mejorar la solución global encontrada por ACO los autores aplican una búsqueda local, la cual consta de dos mutaciones:

- Se selecciona al azar una posición de una tarea y se cambia por otra, que de igual manera se seleccionó al azar, si viola la restricción de precedencia se mueve lo más cerca posible.
- Luego se selecciona una tarea al azar y después un empleado asignado a esa tarea, también al azar y se restablece su asignación de carga de trabajo en 0 (se quita de la tarea), finalmente se

selecciona otro empleado que no esté asignado a esa tarea y se indica la carga de trabajo, estos dos últimos pasos de igual manera se realizan al azar.

Este procedimiento se repite hasta que la solución ya no mejora. En los experimentos los autores usaron tres instancias reales y 80 instancias generadas aleatoriamente. El algoritmo desarrollado se comparó contra otros algoritmos (KGA, KGA-p, 3dGA, TS) desarrollados anteriormente y en todos los casos ACO con la mejora local dio como resultado menores costos, con diferencias de hasta el 20 por ciento.

Analizando los resultados, son mejores ya que al ser basado en eventos se mantiene la estabilidad del plan y por otro lado su algoritmo permite la asignación de empleados a varias tareas en el mismo tiempo lo cual lo hace flexible la asignación de recursos.

- **A Hybrid Approach to Quantitative Software Project Scheduling Within Agile Frameworks [8]**

En este trabajo el autor muestra una forma de modelar la planeación de sistemas con metodologías ágiles usando Scrum . Este artículo no se enfoca en la solución con heurísticas, ya que al presentar instancias pequeñas del problema, buscan resolverlo con el software GAMS (https://www.gams.com/) y modelando el problema con una tecnica de programación entera. Su enfoque es híbrido ya que busca combinar en su modelo las características de Scrum con una planeación tradicional en cada inicio de iteración.

El artículo habla de que la planeación de proyectos de software tradicional y la planeación con metodologías ágiles no son contrarias, ya que los proyectos de software con especificaciones definidas y fechas de entrega acotadas, así como los presupuestos limitados enfrentan objetivos estrictos de igual manera.

En el artículo, el autor busca centrarse en las fechas de entrega y las relaciones entre los costos y el presupuesto para el proyecto. Se presenta un modelo matemático para minimizar el tiempo de cada iteración, ya que al intentar minimizar el costo las iteraciones se hacen muy largas.

En el modelo propuesto por el autor, se toma en cuenta la experiencia del desarrollador, la dificultad de la tarea según al desarrollador a quien sea asignada y también el presupuesto.

La planeación cuantitativa (como le llama a la planeación tradicional) se usa por primera vez en la fase de planificación inicial del proyecto, que incluye la priorización de la historia del usuario. Más allá de esta etapa, la planeación se convierte en una parte recurrente del proceso. Para cada sprint (iteración), se calcula y utiliza un cronograma como punto de referencia e instrumento de control en la siguiente reunión de revisión y retrospectiva.

Lo que se puede adaptar a el enfoque de replaneación, es que el administrador de proyecto en cada iteración incluye la retrospectiva de los desarrolladores en la iteración anterior, como parámetros configurables para una nueva planeación de sprint.

- **Dynamic Staffing and Rescheduling in Software Project Management: A Hybrid Approach [10]**

En este artículo los autores Ge & Xu buscan resolver el problema de planeación y asignación de personal a tareas en el desarrollo de software desde una perspectiva dinámica. Ya que a lo largo del desarrollo del proyecto, en la vida real siempre suceden eventos como la llegada de recursos o cambios en las especificaciones iniciales de las tareas, a los cuales el plan inicial debe adaptarse y es necesaria una replaneación. Para la cual se buscará minimizar el tiempo y costo del proyecto; pero además de esto, un sistema dinámico depende del tiempo y los autores plantean que en la literatura revisada nadie ha tomado en cuenta la productividad del equipo como factor que afecte al desarrollo del proyecto. Por lo cual ellos modelan la productividad del equipo en dos factores, el primero es la comunicación necesaria que el equipo necesita para ponerse de acuerdo en tareas donde varios recursos trabajan juntos y el segundo factor es la productividad individual la cual se compone de tres partes: “La presión del cronograma”, “Las habilidades del recurso” y “el aprendizaje que se obtiene con el tiempo.”

Como método de solución proponen un sistema híbrido que está compuesto de dos métodos heurísticos, el primero es un algoritmo genético el cual optimiza de manera general la planeación inicial y una heurística de Hill Climbing para optimizar de forma local la mejor solución encontrada con el algoritmo genético. La función objetivo busca minimizar la suma de la Eficiencia del proyecto (a lo que ellos toman como el costo total de realizarlo en base a el salario de los recursos) más la estabilidad del proyecto (que es minimizar las diferencias entre una planeación y su antecesor).

$$F.O. = Ws * Estabilidad + We * Eficiencia$$

Las variables We y Ws son configurables e indican el valor de importancia que se le da a la eficiencia y a la estabilidad respectivamente. Para modelar a los individuos de la población para el algoritmo genético, cada uno consta de dos vectores, el primero se obtiene de dos matrices: se toma en cuenta la matriz de habilidades del empleado-tarea y la matriz del porcentaje que dedica cada empleado a cierta tarea, mezclandolas, se obtiene el primer vector. En la figura 3 se muestra cómo se realiza; el segundo vector es una lista de prioridades de las tareas, el cual sólo servirá como desempate en la prioridad de dos tareas pero siempre tomando en cuenta la precedencia, es decir solo servirá en caso de que dos tareas no tengan precedencia (La precedencia se modela con un grafo de precedencia), esto garantiza que todos los individuos formados sean factibles.

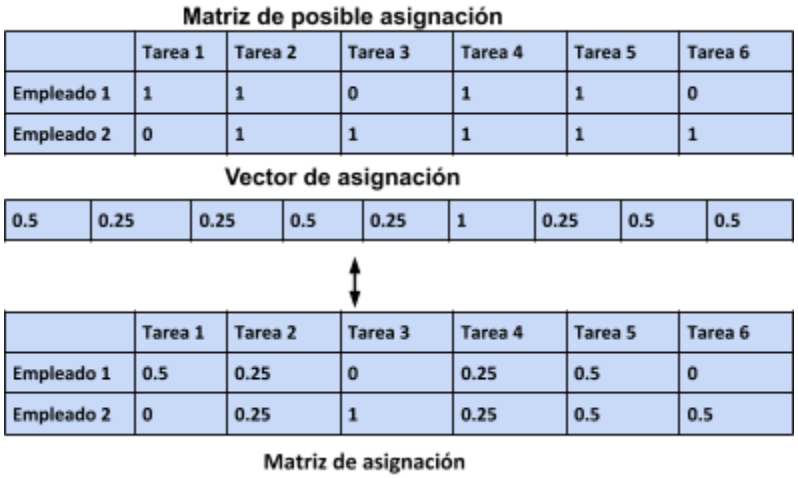



Fig. 3. Primera parte del cromosoma

Para la cruza se obtiene un número al azar y se decide entre hacer una cruza en un punto definido del cromosoma o realizar una "cruza" entre dos números del vector de prioridades. Para la mutación de igual manera se selecciona un número al azar y dependiendo de este se muta una celda del vector de prioridades o del vector de asignación. La función para obtener la aptitud de los individuos de la población es la encargada de tomar en cuenta la producción del equipo y para esto la producción se modela de la siguiente forma:

$$Productividad = (1 - sobrecargaComunicacion / 100) * \Sigma Pindividual$$

La ecuación anterior se puede leer como la suma de las productividades individuales afectada por un factor de la sobrecarga de comunicación (o comunicación necesaria del equipo). Donde la sobrecarga de comunicación se obtiene de una fórmula que depende de la cantidad de personas involucradas en resolver una tarea (donde una sola persona es igual a 0 ya que no necesita comunicación). La productividad individual depende de tres factores:

$$Pindividual = nomP * fhabilidades * faprendizaje * fpresionPlaneacion$$

Donde $nomP$ es un valor que representa la productividad normal de un empleado y es afectada por:

- ➢ Función de habilidades, que consiste en evaluar la aptitud de un empleado para realizar cierta tareas, osea cuantas habilidades tiene el empleado entre el total de las habilidades que requiere la tarea.
- ➢ Función de aprendizaje, que representa la mejora de la comprensión de la tarea y del aprendizaje del empleado (curva de aprendizaje)

- ➢ La función de presión del cronograma (presión del tiempo), se define como una función que depende del tiempo actual y el tiempo planificado, la cual busca representar como un empleado trabaja más “rápido” cuando la fecha de entrega de la tareas está próxima.

Para obtener los resultados del sistema propuesto se realizaron pruebas con empleados y tareas variables, donde se simulaba la llegada de nuevas tareas y empleados, así como con alguno de los factores de productividad desactivados. Lo que más llama la atención es que se realizaron experimentos con dos administradores de proyecto en las cuales se les pidió realizar una planeación y en un segundo experimento una replaneación, de los cuales se obtuvo que el sistema funciona bien y más rápido que los expertos, ya que ellos tardaron 3 horas y 3 y media respectivamente y el sistema tardó entre 19 y 25 minutos. Los resultados entre expertos y sistema fueron muy similares y se muestran en diagramas de Gantt. También realizaron experimentos donde se muestra como afecta los cambios en la eficiencia y estabilidad del proyecto y como se ven reflejados en las soluciones, como ejemplo ponen como si la fecha de entrega es adelantada el sistema asigna más recursos a las tareas para que esta sea realizada de forma más rápida y cómo se asigna a los mejores recursos (que aprenden más rápido).

- **Human Resource Allocation in Agile Software Projects Based on Task Similarities [6]**

En el artículo los autores hablan de la importancia de realizar una buena asignación de los recursos humanos a las tareas, esto quiere decir que la asignación maximice sus habilidades, tomando en cuenta que la productividad puede variar entre desarrolladores. Como particularidad de esta investigación los autores toman en cuenta la similitud entre tareas. Proponen modelar con un enfoque ágil en el cual el software se desarrolla iterativamente y se entrega a los clientes en incrementos llamadas liberaciones. En el artículo le llaman Resource Allocation for Software Release Planning (RASORP), argumentan que la asignación de un grupo de tareas similares adecuadas para las habilidades del empleado puede colaborar para lograr una planificación de lanzamiento eficiente y rentable. Como solución proponen modelar con una función multiobjetivo y se busca minimizar los objetivos de tiempo y costo del proyecto mediante la asignación de tareas adecuadas y similares.

Para realizar una asignación eficiente de tareas a un empleado, se toma en cuenta las habilidades que posee el empleado y su nivel, así como las habilidades que requiere la tarea y el nivel de ellas. Una buena asignación busca que el nivel de las habilidades requeridas por la tarea sea menor o igual que las de el empleado asignado, así los autores buscan modelar el factor de experiencia del desarrollador. El otro aspecto es la similitud entre tareas, ya que tareas similares realizadas por un mismo empleado tienden a realizarse de manera más eficiente. Para el objetivo de tiempo se se usa la siguiente ecuación:

$$Tiempo = \sum N_i = 1 \; TEH_i \times (1 - \alpha \times EfcFactor(i, X_i)) \times (1 - \beta \times SimFactor(i, S))$$

Donde $TEH_i$ es la estimación inicial (o base) del tiempo que tardará en realizarse la tarea, este se multiplica por $EfcFactor$ que es la función que calcula que tan eficiente es una asignación en cuanto a las habilidades de la tarea y empleado, este valor puede ser negativo si la tarea requiere un mayor nivel que el de las habilidades del empleado o positivo si el empleado tiene experiencia mayor o igual a la tarea. Por último se multiplica por $SimFactor$ el cual representa el factor de similitud entre tareas asignadas al empleado. Para obtener el costo se usa la siguiente función:

$$Cost(S) = ValueHours\,(X_i, EstimatedHours(X_i, S))$$

Donde $EstimatedHours$ calcula las horas estimadas que el empleado trabaja en la iteración y $ValueHours$ da como resultado el costo de esas horas.

Para los experimentos, el modelo que los autores desarrollaron se implementó en tres algoritmos, NSGA-II, MOCell y búsqueda aleatoria, en los cuales se utilizaron instancias obtenidas de un proyecto real de una empresa. El que mejor resultados arrojó fue NSGA-II. Los dos parámetros de los algoritmos evolutivos se obtuvieron empíricamente y se configuraron con 256 individuos, 400 generaciones, una tasa de cruce del 90% y una tasa de mutación del 1%. En cuanto a las medidas para evaluar el resultado de los algoritmos utilizados, ocuparon el hipervolumen (HV), distancia generacional (GD) y Spread (SP).

Como conclusión el artículo ajusta una solución multiobjetivo parecida a la planeación de proyectos "tradicional" a las metodologías ágiles, enfocándose en que la planeación se realiza iteración con iteración, en el release planning.

- **Dynamic Software Project Scheduling through a Proactive-Rescheduling Method [3]**

Los autores de este trabajo definen el problema de planeación y replaneación de proyectos desde una perspectiva dinámica y proactiva, es decir, para el modelo de planeación / replaneación que realizaron, tomaron en cuenta los eventos disruptivos que suceden en la fase de ejecución del proyecto: "llegadas de nuevas tareas", "cuando un empleado se va" (por permisos o enfermedades) y cuando un empleado regresa, ya que estos eventos pueden hacer que la planeación inicial quede obsoleta y sea necesario realizar una replaneación. Como aporte singular de esta investigación, está la parte proactiva, con la cual los autores tratan de minimizar el impacto de la incertidumbre a la que está sujeta un proyecto haciendo énfasis en la incertidumbre del cálculo del esfuerzo requerido para que un empleado realice una tarea. Cuando ocurre un evento disruptivo(o varios) y es necesario realizar una replaneación, el sistema ajusta

los tiempos restantes de las tareas que no fueron bien calculados, que ellos ajustan a una distribución normal (mencionan que puede ser fácilmente configurada cualquier distribución).

La solución que presentan es un algoritmo evolutivo multiobjetivo (MOEA), el cual como su nombre lo dice, busca optimizar 4 funciones objetivo:

- $min\ F = [\ f1\ (tl), f2\ (tl), f3\ (tl), f4\ (tl)\ ]$, donde tl es el tiempo actual,
- f1(tl) = Duración del proyecto: Se refiere al tiempo que resta para terminar el proyecto,
- f2(tl) = Costo del proyecto: Se refiere al costo total del proyecto (salario de los empleados),
- f3(tl) = Robustez: Se define como que la replaneación en el tiempo actual (tl) tomando en cuenta el esfuerzo para realizar cada tarea (duración de las tareas y el costo por realizar cada una), no difiera demasiado del plan anterior,
- f4(tl) = Estabilidad: Se define como los cambios que hay en la asignación de los empleados a las tareas, de igual manera se busca minimizar que los empleados cambien demasiado de tareas, al realizar la replaneación.

Los autores ponen un ejemplo simple como guía del modelo de su solución, el cual consta de 4 pasos:

1. En el momento inicial del proyecto, el administrador del proyecto identifica varios atributos del proyecto a desarrollar. Estas son las tareas, dependencias de tareas y esfuerzo, requeridos en ellas. El administrador puede identificar que hay 12 tareas en total, por ejemplo, escribir los diagramas UML, diseñar la base de datos, diseñar plantillas de página web, implementar, probar el software, escribir documentos de diseño de base de datos y un manual de usuario, etc. Además de los atributos del proyecto en sí, el administrador del proyecto también identifica las propiedades de los empleados, como las habilidades adquiridas por cada empleado, la dedicación máxima de cada empleado al proyecto, los salarios de trabajo normales y de tiempo extra de cada empleado.
2. Proporciona la información recopilada en el paso 1 como entrada al algoritmo de planeación proactiva basado en el MOEA. Generará automáticamente planeaciones para minimizar los objetivos de la duración del proyecto, el costo y la sensibilidad del cronograma a las incertidumbres del esfuerzo de la tarea (robustez), satisfaciendo las restricciones de no exceso de trabajo, habilidades necesarias requeridas por las tareas, etc. El resultado es un conjunto de soluciones no dominadas que representan planeaciones con diferentes valores entre los tres objetivos.
3. Una vez que el enfoque genera las soluciones no dominadas, el administrador del software debe elegir una solución para adoptar. El administrador podría elegir el cronograma sugerido por el procedimiento automatizado de toma de decisiones, o seleccionar un cronograma manualmente

en base a la información provista por nuestro enfoque y su propia experiencia y conocimiento sobre el proyecto.

4. Durante la vida del proyecto, pueden ocurrir algunos eventos disruptivos, por ejemplo, alteración de tareas, abandono de empleados, empleado con participación interrumpida, reducción de presupuesto para algunas tareas y cambio de enfoque en otras. Para simplificar, sólo se consideran nuevas llegadas de tareas, permisos de empleados y regreso de empleados. Entre ellos, las llegadas urgentes de tareas, los permisos de los empleados y los retornos se consideran eventos críticos, mientras que las llegadas regulares de tareas se consideran no críticas. Para reducir la frecuencia de replaneación, se emplea un modo impulsado por eventos críticos.

Del ejemplo anterior podemos observar que la replaneación sólo es necesaria si ocurren eventos disruptivos críticos.

El algoritmo que usan es un algoritmo evolutivo normal y representan el cromosoma de la siguiente manera:

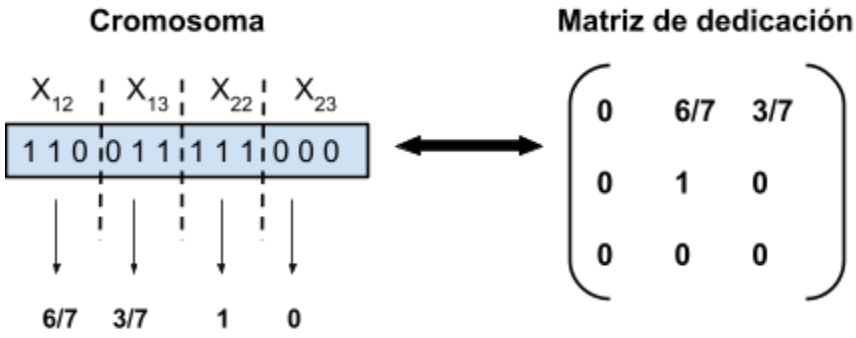



Fig. 4. Cromosoma

El cual se obtiene de la matriz de dedicación, la cual indica que fracción de su tiempo total de cada empleado va a dedicar a cada una de las tareas, el cromosoma se forma de 3 bits para representar en “séptimos” el tiempo que un empleado dedica a una tarea. Y a continuación se muestran las configuraciones finales para el sistema:

Tabla 2. Parámetros algoritmo evolutivo

| | |
|---|---|
| Tamaño población | 100 |
| Cromosoma | Codificación con 3 bits |
| Probabilidad de cruce | 0.9 |
| Probabilidad de mutación | 1 / L , donde L es la longitud del cromosoma |

| | |
|---|---|
| Número máximo de evaluaciones | 10000 |

Los experimentos se realizaron con simulaciones de tareas y empleados variables, comparándose contra otras instancias de MOEA, en el cual su algoritmo se comporta muy bien y da mejores resultados que cualquiera de ellos. Se hace énfasis en que al ser multiobjetivo el resultado es un conjunto de soluciones dependiendo del objetivo al que se le otorgue más importancia, ya que como en este tipo de soluciones multiobjetivo los objetivos pueden afectar entre sí.

- **Model-Based Dynamic Software Project Scheduling [7]**

Este trabajo es el comienzo de una investigación doctoral, el autor propone un modelo matemático para el problema de planeación y replaneación, menciona que los desafíos asociados son la evaluación de que tan buena es una planeación o replaneación, el manejo del comportamiento caótico, la reducción de cargas computacionales y el acomodo de los eventos disruptivos. Menciona que en su doctorado buscará proponer una solución para el problema de planeación de proyectos de software a gran escala para un modelo de ingeniería de software híbrido, con una combinación de RUP y Scrum, la cual llama: “scRUmP”. Que se centra en la agilidad y la calidad, tomando en cuenta el entorno dinámico. Toma en cuenta los eventos disruptivos como que un empleado puede unirse a la empresa después de comenzado el proyecto, si se asigna o reemplaza a un empleado con menos experiencia puede reducir el costo o tiempo del proyecto. Las tareas por su parte se asignan con prioridades, por lo que si llegan tareas con misma prioridad al mismo tiempo, deben ejecutarse y garantizar la eficiencia del proyecto, lo que no es trivial para un administrador de proyecto. Menciona que ninguno de los enfoques existentes toma en cuenta la replaneación continua en las soluciones en entornos dinámicos.

Para modelar la solución, aparte de los modelos estáticos para tareas y empleados (salario, habilidades, esfuerzo, etc), se usa un función multiobjetivo la cual consta de cinco objetivos y para los cuales a diferencia de los demás, no siempre minimiza el valor. A continuación se describen los objetivos y se indicará entre paréntesis con MIN si el objetivo se minimiza y MAX si el objetivo se maximiza,:

- ➢ Duración (MIN). La duración del proyecto es el tiempo máximo requerido para completar el proyecto, el cual se penaliza si no hay experiencia en el equipo(cada empleado tiene un valor de experiencia), para simular el tiempo de entrenamiento (curva de aprendizaje).
- ➢ Costo (MIN). Es el costo de los empleados asignados al proyecto, es decir, es el pago de su salario normal y de horas extras que dedican al proyecto.
- ➢ Fragmentación de tareas (MIN). Este valor sirve para identificar el impacto de que una tarea se retrase ya que también impactaría en los sucesores directos e indirectos de esta. Por lo tanto este valor será menor cuanto menos tareas dependen de otra.:

- ➢ Robustez (MAX). Es la capacidad del cronograma para hacer frente a pequeños incrementos en el tiempo de la duración de las tareas. Se define como el tiempo de inactividad de la tarea por el cual una tarea se puede retrasar sin demorar todo el proyecto.
- ➢ Estabilidad (MAX). Este objetivo mide la desviación entre horarios nuevos y originales. Penaliza que los empleados se muevan demasiado.

- **A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling [4]**

En el artículo los autores mencionan la importancia de los factores humanos en el problema de planeación / replaneación de proyectos, mencionan que hay factores humanos que afectan al proyecto como el aprendizaje del empleado, la motivación y la satisfacción cuando es asignado a cierto tipo de tarea. Proponen una solución con un enfoque dinámico basado en eventos, tomando en cuenta y mejorando su trabajo anterior [3] donde propusieron como solución un algoritmo genético multiobjetivo con cuatro objetivos (costo, duración, estabilidad y robustez). En esta investigación los autores mejoran su modelo tomando en cuenta los siguientes factores:

- ➢ Motivación del empleado para aprender (simula la motivación que tiene el empleado debido a factores en su ambiente que puedan afectar su estado de ánimo, puede obtenerse con un cuestionario)
- ➢ Capacidad de aprendizaje (Capacidad que tiene para aprender, puede obtenerse con la opinión del administrador de proyecto o compañeros)
- ➢ Dificultad para mejorar una habilidad (Este factor se refiere a que tan difícil es mejorar cierta habilidad, ya que no todas se aprenden a la misma velocidad)
- ➢ Por último agregan como un quinto objetivo de optimización la satisfacción del usuario al ser asignado a cierto tipo de habilidad la cual es requerida por cierta tarea (buscan simular la preferencia de los empleados a ser asignados a cierto tipo de tareas)

Los autores como algoritmo de solución proponen un Algoritmo Memético Multiobjetivo de Dos Archivos Basado en Q-Learning (MOTAMAQ), el cual es un algoritmo que combina en este caso, dos tipos de búsqueda global (GA y AMDE) y para cada una de ellas dos tipos de búsqueda local (Mutación), en resumen, el algoritmo selecciona una búsqueda global y una local, dependiendo del estado en el que se encuentre y en base al aprendizaje, respaldado por el conocimiento obtenido del entrenamiento.

En resumen, para el aprendizaje, el algoritmo puede percibir el entorno, seleccionar una acción para ejecutar en el estado actual y luego el entorno retroalimenta una señal de recompensa o penalización de

la acción tomada, en este caso se considera la aptitud del individuo como una recompensa y así los autores ajustan el modelo Q-learning a la solución.

Para abordar las características dinámicas en la planeación de proyectos de software, se tienen en cuenta un tipo de incertidumbre (la incertidumbre en el estimado esfuerzo de las tareas) y tres eventos disruptivos (nuevas llegadas de tareas, salidas de empleados y regresos de empleados). Entre ellos, las llegadas de tareas urgentes, los permisos de los empleados y los retornos se consideran eventos disruptivos, los cuales son detonadores para que se deba realizar una replaneación, las llegadas regulares de tareas son eventos triviales, las cuales se guardan en una cola y se agregan a la replaneación cuando ocurra un evento crítico o cuando llegue la fecha en que alguna de ellas deba realizarse. La función objetivo consta de 5 objetivos de optimización:

- ➢ Duración total del proyecto (tiempo que tardarán en realizarse todas las tareas)
- ➢ Costo total del proyecto (dinero que se debe pagar a los empleados)
- ➢ Robustez (sensibilidad de la planeación a la incertidumbre en el esfuerzo estimado para las tareas)
- ➢ Estabilidad (desviaciones entre la planeación original y la replaneación)
- ➢ Satisfacción (disposición de los empleados para asignarse a las tareas, simula el grado de satisfacción de todos los empleados al ser asignados a las tareas que les corresponden)

Una de las partes que más resaltan los autores en el artículo es el cálculo del aprendizaje, el cual se obtiene con la siguiente fórmula:

$$a_i^k = e_i^{IA} * e_i^{MO} / SD^k$$

donde $e_i^{IA}$ es el factor de aprendizaje del empleado i, $e_i^{MO}$ es el factor de motivación del empleado i , por último $SD^k$ es el nivel de dificultad para aprender la habilidad k. Podemos decir que la tasa de crecimiento de competencia de las habilidades de cada empleado, se obtiene de multiplicar su motivación, por la capacidad de aprendizaje, entre la dificultad para aprender cierta habilidad.

Como aporte al trabajo anterior los autores agregaron el objetivo de satisfacción, el cual se obtiene de la disposición que tiene el empleado *i* para la tarea *k*, es decir, calculan el promedio de la satisfacción de todos los empleados. Los autores mencionan 3 mejoras respecto a su modelo anterior:

1. Para enfatizar los efectos de los factores humanos en el éxito del proyecto, se introducen más propiedades en la iniciativa de los empleados, como la motivación, la capacidad de aprendizaje y el grado de compromiso de cada empleado con cada habilidad (y con cada tarea).
2. Para ser más coherente con la realidad, se puede mejorar la competencia de cada empleado en cada habilidad a lo largo del tiempo.

3. En función del grado de compromiso de cada empleado con cada habilidad (y con cada tarea), se ha considerado la satisfacción de los empleados con el cronograma generado junto con la duración, el costo, la robustez y la estabilidad del proyecto.

Para el algoritmo MOTAMAQ propuesto como solución las características importantes son:

- El estado actual de la planeación se obtiene con dos factores:
  - Tasa de esfuerzo. Que es el esfuerzo restante para terminar el proyecto.
  - Índice de competencia. Es la suma de todas las competencias de todos los empleados disponibles.
- Se usan dos algoritmos de optimización global y para cada uno de ellos se utilizan dos mas de optimización local, los cuales se describen a continuación:
  - Algoritmo Genético.
    - Mutación de un valor de la matriz al azar.
    - Intercambio de dos filas o columnas al azar.
  - AMDE (Angle Modulated Differential Evolution)
    - Mutación probable en cada uno de los cuatro valores (a,b,c,d)
    - Mutación de un valor al azar

Para los experimentos se usaron 18 proyectos creados para pruebas y 3 con datos de proyectos reales, donde se simulaba la llegada de tareas urgentes, idas y regresos de empleados. Se comparó contra su algoritmo anterior (dε-MOEA) además de otra propuesta actual como es NSGA-II. También se comparó MOTAMAQ contra instancias de él mismo (osea combinaciones de sus búsquedas globales y locales ). Como MOTAMAQ está compuesto por 5 objetivos, se usaron fórmulas para determinar la dispersión, la convergencia y el espacio de búsqueda, en la mayoría de los casos MOTAMAQ mostró la mejor convergencia lo cual indica que la solución propuesta es mejor que las demás, pero en algunas ocasiones el espacio de búsqueda es mayor con los otros algoritmos, debido a que con MOTAMAQ se busca mejorar la solución rápidamente dependiendo del estado en el que se encuentre.

## 2.2 Análisis de la revisión bibliográfica

En la revisión bibliográfica que realizamos, en los trabajos que tratan el problema de replaneación, podemos encontrar varias generalidades. La primera es que no hay una forma general de llamarle al problema de replaneación, además en todos los artículos el desarrollo de un proyecto de software se modela como un sistema dinámico ya que está expuesto a alteraciones externas y cambios con el tiempo. En todos los trabajos se ubica a la replaneación en la fase de ejecución del proyecto y por último

encontramos que la forma general de representar las dependencias entre tareas se realiza con un grafo de precedencia. Para presentar su solución en todos se usaron heurísticas, en ocho de ellos se usan algoritmos genéticos, Shen et al. [4] usa un algoritmo memético y en Chen et al. [5] se usa la heurística de colonia de hormigas. A continuación en la tabla 3 se muestra una comparativa de los trabajos revisados, en ella se muestra un resumen de las características principales de cada uno de ellos, las cuales son: el nombre del artículo, el algoritmo de optimización que usan, la aportación que realizan, cuál es el evento que detona una replaneación, el entorno que representan y por último la forma en que realizaron la evaluación de su trabajo.

Tabla 3. Características de los trabajos revisados

| Referencia | Algoritmo de optimización | Función Objetivo | Aportación | Replaneación (cuando ocurre) | Entorno Estático /Dinámico | Evaluación |
|---|---|---|---|---|---|---|
| Ge Y, Xu B. (2016) "Dynamic Staffing and Rescheduling in Software Project Management: A Hybrid Approach" [10] | Algoritmo Genético<br><br>Cromosoma: compuesto de dos vectores(asignación y prioridad).<br><br>- Cruza: Punto de cruce definido. (prob. 0.65)<br>- Mutación: mutación de un valor del vector al azar (prob. 0.01)<br>- #Población = 1 mil<br>- #Generación = 1mil | Función compuesta:<br><br>Minimizar:<br><br>- Estabilidad (Que no haya un cambio tan radical en el plan inicial)<br>- Eficiencia (Costo total del proyecto) | - Modelado de la productividad de un equipo de desarrollo de software.<br>- Buenos resultados en aplicaciones reales ya que se compararon contra expertos. | Eventos que la disparan:<br>- Llegan o se van empleados<br>- Llegan nuevas tareas | Dinámico:<br><br>- Modela la productividad del equipo de trabajo | - Se usó como ejemplo un proyecto real<br>- Planeación tradicional<br>- 19 tareas<br>- 9 empleados |
| X. Shen, L. L. Minku, R. Bahsoon and X. Yao. (2016) "Dynamic Software Project Scheduling through a Proactive-Rescheduling Method" [3] | Algoritmo Genético<br><br>Cromosoma: vector de asignación<br><br>- Cruza: Punto de cruce definido. (prob. 0.9)<br>- Mutación: mutación de un valor del vector al azar.(prob. 1/L (tam. vect.))<br>- #Población = 100<br>- #Generación = 10mil | Función multiobjetivo:<br><br>MIN F = [f1, f2, f3, f4]:<br><br>- f1 = Duración del proyecto.<br>- f2 = Costo del proyecto.<br>- f3 = Robustez: Se define que la replaneación no difiera demasiado del plan anterior.<br>- f4 = Estabilidad: Se define como las desviaciones entre la planeación original y la replaneación | - Planeación proactiva para minimizar la incertidumbre en la estimación de las tareas.<br>- Función multiobjetivo en la solución de problemas de planeación y planeación | - Llegada de nuevas tareas<br>- Un empleado se va<br>- Un empleado regresa | Dinámico:<br><br>- Se toma en cuenta la incertidumbre en el esfuerzo de las tareas (proactivo) | - Se crearon instancias al azar donde se varió el número de tareas (max 30), nuevas tareas (max 10), el número de empleados (max 10) y el número de habilidades requeridas por tareas (max 7)<br><br>- 3 instancias del mundo real pero no define el número de tareas, ni de empleados. |
| Xiao J., Osterweil L.J., Wang Q., Li M. (2010) "Dynamic Resource Scheduling in Disruption-Prone Software Development Environments" [11] | Algoritmo Genético<br><br>Cromosoma: vector de asignación + prioridades<br><br>- Cruza: Punto de cruce definido. (prob. 0.8)<br>- Mutación: mutación de un valor del vector al azar.(prob. 0.02)<br>- #Población = 60<br>- #Generación = 500 | Función compuesta:<br><br>Minimizar:<br><br>- Estabilidad (desviaciones entre la planeación original y la replaneación)<br>- Utilidad (El valor monetario que se espera ganar del proyecto) | - Cuando un evento disruptivo llega no es necesario cambiar todo el plan, solo hay que validar las actividades afectadas para mantener la estabilidad entre proyectos | - Llegada de nuevas tareas urgentes<br>- Errores graves en la estimación de la duración de las tareas<br>- Rotación de personal (cuando un empleado se va o regresa) | Dinámico | - usa una instancia del mundo real<br>- 12 tareas<br>- 7 empleados<br>- 6 nuevas tareas<br>- 2 nuevos empleados |
| Y. Ge. (2009) "Software Project Rescheduling with Genetic Algorithms" [9] | Algoritmo Genético<br><br>Cromosoma: Matriz de asignación<br><br>- Cruza: Punto de cruce definido. (prob. 0.4)<br>- Mutación: mutación de un valor del vector al azar.(prob. 0.01)<br>- #Población = 100<br>- #Generación = 1000 | Función compuesta:<br><br>Minimizar:<br><br>- Estabilidad (desviaciones entre la planeación original y la replaneación)<br>- Eficiencia (tiempo y costo del proyecto) | - Modelado de replaneación con un algoritmo genético | - Llegada de nuevas tareas<br>- Llegan o se van empleados | Dinámico | - instancias generadas para prueba<br>- hasta 12 tareas<br>- 10 empleados<br>- llegada de hasta 4 tareas<br>- llegada de hasta 4 empleados |

| | | | | | | |
|---|---|---|---|---|---|---|
| Gueorguiev, S., Harman, M., & Antoniol, G.“ Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering” [14] | Algoritmo Genético<br><br>Cromosoma: Doble matriz<br><br>- Cruza: Punto de cruce definido. (prob. 0.8)<br>- Mutación: se seleccionan dos posiciones al azar y se cambian (prob. 0.1).<br>- #Población = 250<br>- #Generación = 180 y 250 | Función multiobjetivo:<br><br>Minimizar:<br><br>- Tiempo(Tiempo total necesario para terminar el proyecto)<br>- Robustez(Tolerante a incertidumbre, o sea nuevas tareas que puedan llegar)<br>- Estabilidad (desviaciones entre la planeación original y la replaneación) | - Frente de pareto de tres objetivos se puede representar por separado en un plano<br>- El costo tan grande que implica la robustez en proyectos | No hay replaneación | Estático | - instancias del mundo real<br>- hasta 20 empleados<br>- hasta 7 habilidades<br>- hasta 120 tareas<br>- extender el tiempo de finalización de hasta el 30% de tareas<br>- agregar hasta un 30% de tareas nuevas |
| W. N. Chen and J. Zhang, (2013) "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler" [5] | Colonia de hormigas<br><br>Feromona: para elección de inicio de tareas, para la asignación de empleados a tareas y para la carga de trabajo de un empleado a una tareas.<br><br>- Hormigas = 10 | Función compuesta:<br><br>Minimizar:<br><br>- Costo de proyecto<br>- Penalización por no terminar tareas a tiempo | - Modela el problema de planificación de tareas y asignación de recursos con un algoritmo de colonia de hormigas<br>- Tiene resultados muy buenos debido a que la estabilidad es minimizada por el enfoque de sus eventos<br>- Para la asignación de empleado busca al mejor empleado en base a competencia/salario | No hay replaneación | Estático: | - instancias creadas para prueba<br>- hasta 92 tareas<br>- hasta 10 habilidades<br>- hasta 20 empleados<br><br>- instancias del mundo real<br>- hasta 15 tareas<br>- hasta 5 habilidades<br>- hasta 10 empleados |
| Yi-Nan Guo, Ying Han (2017)<br>“A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling, Information Sciences” [4] | Algoritmo memético multiobjetivo basado en Q-Learning:<br><br>ALgoritmos de optimización:<br>- Global: Algoritmo Genético.<br>---Local: Mutación de un valor de la matriz al azar.<br>---Local: Intercambio de dos filas o columnas al azar.<br><br>* Cruza: Punto de cruce definido. (prob. 0.9)<br>* Mutación: (prob. 1/L (tam. vect.))<br>* #Población = 100<br>* #Generación = 15mil<br><br>-Global: AMDE (Angle Modulated Differential Evolution)<br>---Local: Mutación probable en cada uno de los cuatro valores (a,b,c,d)<br>---Local: Mutación de un valor al azar<br><br>*Mutaciones: 0.25<br>* #Generación = 15mil | Función multiobjetivo:<br><br>MIN F = [f1, f2, f3, f4, f5]:<br><br>- Duración total del proyecto (tiempo que tardarán en realizarse todas las tareas)<br>- Costo total del proyecto (dinero que se debe pagar a los empleados)<br>- Robustez (sensibilidad de la planeación a la incertidumbre en el esfuerzo estimado para las tareas)<br>- Estabilidad (desviaciones entre la planeación original y la replaneación)<br>- Satisfacción (disposición de los empleados para asignarse a las tareas, simula el grado de satisfacción de todos los empleados al ser asignados a las tareas que les corresponden) | - Ajuste del algoritmo memético basado en Q-learning al problema de planeación y replaneación<br>- Objetivo de satisfacción en base a las preferencia de empleados a ciertas habilidades | - Llegada de nuevas tareas<br>- Un empleado se va<br>- Un empleado regresa | Dinámico:<br><br>- Toma en cuenta el aprendizaje con el tiempo de los empleados.<br>- Toma en cuenta la motivación del empleado para aprender. | - instancias creadas para prueba<br>- hasta 30 tareas<br>- hasta 7 habilidades<br>- hasta 15 empleados<br><br>- instancias del mundo real<br><br>-no explica las dimensiones de los parámetros. |
| Natasha Nigar. 2017. “Model-Based Dynamic Software Project Scheduling” [7] | Aún no tiene un algoritmo desarrollado(propone colonia de hormigas y enjambre de partículas, sin detalles). | Función multiobjetivo:<br><br>MIN F = [f1, f2, f3, f4,f5]<br><br>- (MIN) Duración. La duración del proyecto es el tiempo máximo requerido para completar el proyecto.<br>- (MIN) Costo. Es el costo de los empleados asignados al proyecto, es decir, es el pago de su salario normal y de horas extras que dedican al proyecto. | - Menciona el enfoque ágil “scRUmP”<br>- Las ecuaciones para los valores objetivos tienen iteraciones, haciendo referencia metodologías ágiles. | Aun no esta definido (Propone de la literatura, eventos dinámicos como llegadas de tareas, retorno e ida de empleados) | Dinámico:<br><br>- (Aún no está definido) | No realiza evaluación |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | - (MIN)Fragmentación de tareas. Este valor sirve para identificar el impacto de que una tarea se retrase ya que también impactaría en los sucesores directos e indirectos de esta. Por lo tanto este valor será menor cuanto menos tareas dependen de otra.<br>- (MAX) Robustez. Es la capacidad del cronograma para hacer frente a pequeños incrementos en el tiempo de la duración de las tareas.Se define como el tiempo de inactividad de la tarea por el cual una tarea se puede retrasar sin demorar todo el proyecto.<br>-(MAX) Estabilidad. Este objetivo mide la desviación entre horarios nuevos y originales. Multa que los empleados se muevan demasiado | | | | |
| Lucas Roque, Allysson Allex Araujo, Altino Dantas, Raphael Saraiva, Jerffeson Souza. 2016. “Human Resource Allocation in Agile Software Projects Based on Task Similarities” [6] | Algoritmo Genético<br><br>Cromosoma: Vector<br><br>- Cruza: Punto de cruce definido. (prob. 0.9)<br>- Mutación: se seleccionan dos posiciones al azar y se cambian (prob. 0.01).<br>- #Población = 256<br>- #Generación = 400 | Función multiobjetivo:<br><br>MIN F = [f1, f2]<br><br>- (MIN) Tiempo. La duración del proyecto es el tiempo máximo requerido para completar el proyecto.<br>- (MIN) Costo. Es el costo de los empleados asignados al proyecto, es decir, es el pago de su salario normal y de horas extras que dedican al proyecto. | - Similitud entre tareas (argumentan los autores que si un empleado realiza tareas similares, las realizará más rápido)<br>- Enfocado en el Release Planning | No hay replaneación | Estático:<br><br>- Se realiza una planeación en cada iteración de Scrum | - Equipo Scrum<br>- 5 empleados<br>- 32 historias usuario<br>- iteraciones: no lo menciona<br><br>Evaluación frente de pareto:<br>- HV (Hipervolumen)<br>- GD (Distancia generacional)<br>- SP (Spread/Propagación) |
| Michael Jahr. 2009. “A Hybrid Approach to Quantitative Software Project Scheduling Within Agile Frameworks” [8] | No existe metaheurística, se modela como problema de programación entera. | Función objetivo:<br><br>- (MIN) Tiempo. La duración del proyecto es el tiempo máximo requerido para completar el proyecto | - Se realiza una planeación en cada sprint de Scrum<br>- En cada iteración se modifican los parámetros en base a la retrospectiva del equipo | No hay replaneación | Estático:<br><br>- Se realiza una planeación en cada iteración de Scrum | - Equipo Scrum<br>- 2 desar. senior<br>- 4 desar. junior<br>- 4 historias usuario<br>- iteraciones: 25 días. |

### 2.2.1 Trabajos enfocados en las metodologías ágiles

Con esta revisión bibliográfica, se encontró que hasta el momento sólo Roque et al. [6] y Nigar [7] presentan trabajos para el problema de planeación y replaneación de proyectos de software enfocados en las metodologías ágiles aplicando heurísticas. Estos trabajos solo son introductorios y únicamente mencionan las características del problema, por lo que no presentan una herramienta que sirva de apoyo a los administradores de proyectos de software en el contexto de las metodologías ágiles. El primero [6] propone un modelo estático como propuesta de solución al problema de planeación de proyectos SPSP, en este, los autores proponen un algoritmo genético y sus objetivos solo son dos: el tiempo y el costo del proyecto. Se habla de la fase de liberación pero sin presentar eventos disruptivos. La planeación se realiza cada sprint, por lo que no se realiza ninguna planeación de liberación o su equivalente en las otras metodologías ágiles. En su modelo se busca realizar la asignación de empleados a las tareas y argumentan que si se asigna al mismo empleado a tareas similares este con el tiempo las realizará más rápido. El segundo [7], sólo propone características para el modelo de planeación/replaneación de proyectos en metodologías ágiles ya que no presenta ningún algoritmo ni modelo para el problema, tampoco se puede saber si se refiere a Scrum. Las características que proponen son las mismas que para el SPSP. En este trabajo solo se mencionan los objetivos que se buscarán usar, los cuales son tiempo, costo, robustez, estabilidad y la fragmentación de HU (impacto de que una HU se retrase). Al ser solo una propuesta no tiene más detalle y solo menciona eventos disruptivos generales.

La mayoría de artículos encontrados toman características de los modelos desarrollados para las metodologías tradicionales, tales como la asignación de empleados a las tareas o que la planeación se realiza solo al inicio del proyecto. Por ejemplo, Jahr [8] considera la planeación de proyectos en un enfoque ágil, pero en él encontramos que el autor modela el problema con proyectos muy pequeños de unas 10 historias de usuario y solo a lo más 5 desarrolladores, por lo que propone un método de programación entera sin heurísticas para resolverlo y solo toma como objetivo el tiempo que tarda el desarrollo. Se realiza una planeación de cada sprint y no encontramos replaneación, además presenta asignación de los empleados a las tareas y no considera una estimación adecuada para las HU.
Los artículos en donde se modela el problema de replaneación hacen énfasis en que el desarrollo de software es un sistema dinámico. Los cambios se presentan durante la fase de ejecución debido a que ocurren eventos disruptivos.

### 2.2.2 Trabajos enfocados en las metodologías tradicionales

Sin contar los trabajos de Jahr [8], Roque et al. [6] y Nigar [7], en todos los casos se enfocan a las metodologías tradicionales. Presentan propuestas que buscan resolver el problema de replaneación de proyectos de software, pero se basan en los trabajos donde solo se presenta la planeación, pero ahora

en un entorno dinámico. En Ge [9] podemos encontrar un modelo de replaneación, como propuesta de solución desarrolló un algoritmo genético el cual busca minimizar el tiempo y costo de proyecto, pero además introduce la estabilidad como objetivo del problema, los eventos disruptivos que considera son dos: llegada de nuevas tareas y el movimiento de los empleados (cuando llegan o se van).

Como antes se mencionó el problema de replaneación es actual y en los últimos dos años, podemos resaltar tres documentos dirigidos a las metodologías tradicionales, comenzando con Ge y XU [10] en el cual los autores buscan minimizar el tiempo, costo y estabilidad del proyecto. Los eventos disruptivos que disparan una replaneación son: la llegada o retiro de empleados al proyecto, así como cuando llegan tareas que no se tenían en la planeación original. En este trabajo los autores buscan modelar la productividad del equipo y cómo afecta en el desarrollo de sus habilidades. Como en los casos anteriores la propuesta de solución está basada en un algoritmo genético.

### 2.2.3 Trabajos con funciones compuestas por múltiples objetivos

En el estudio de Shen et al. [3] encontramos por primera vez una función multiobjetivo en el modelo, de esta manera cada objetivo se calcula de manera independiente pero además su algoritmo presenta como solución un conjunto de posibles propuestas. Los objetivos que se buscan optimizar son: duración del proyecto, tiempo de desarrollo, estabilidad y robustez. Ellos consideran la llegada de nuevas tareas, el retiro de un empleado del proyecto y cuando un empleado se incorpora, como eventos disruptivos. Para validar su trabajo utilizaron tres ejemplos de proyectos reales, y los compararon con las propuestas que su algoritmo daba como solución. Shen et al. [4] también presenta una función multiobjetivo la cual se compone de: la duración del proyecto, el costo del desarrollo, la estabilidad entre planeaciones, la robustez para buscar un plan tolerante a eventos e introducen como objetivo la satisfacción de los empleados al ser asignados a tareas de su agrado. Como propuesta de solución usan un algoritmo compuesto de dos heurísticas. Como solución global proponen un algoritmo genético el cual da como resultado una propuesta de planeación, después, tratan de mejorar a esta propuesta aplicando un algoritmo AMDE (Angle Modulated Differential Evolution). Mencionan que para validar su algoritmo se compararon con proyectos reales, además de las creadas especialmente para las pruebas.

### 2.2.4 Trabajos con características de proyectos reales

Para proponer un modelo más cercano a la realidad, en los trabajos de Xiao et al. [11], Chen et al.[5] y Ge et al. [10] los autores modelan la comunicación necesaria entre los empleados para desarrollar una tarea, la cual afecta al proyecto si muchos empleados están asignados a una misma tarea, ya que gastan mucho tiempo en comunicarse. Por último, encontramos que en trabajos más actuales se modela la productividad y curva de aprendizaje de los empleados y la robustez de un proyecto, podemos ver que

esta característica se presenta en Ge et al.[1 y Roque et al.[6], ellos mejoran el modelo presentado en Ge [9].

## 2.3 Conclusiones de la revisión bibliográfica

Después de realizar el análisis podemos concluir que no se encontró ningún trabajo en el cual se presente una herramienta o propuesta de solución enfocada en el problema de replaneación de liberaciones. En ninguno de los trabajos anteriores se modelan las características de Scrum ni de ninguna otra metodología ágil, ya que ninguno presenta ni objetivos, ni características pensadas en el contexto ágil. Los eventos disruptivos que se modelan constantemente son:

- Llegada de nuevas tareas, ya sea por cambios en los requerimientos, correcciones, agregar funcionalidad o que por retraso se tengan que desarrollar en un sprint diferente.
- Cuando un empleado deja el proyecto de manera temporal por permiso o enfermedad y cuando se va de manera definitiva, por una renuncia por ejemplo.
- Cuando un empleado regresa al proyecto o se incorpora por primera vez.

En la literatura se busca proponer una solución al problema de replaneación principalmente con algoritmos evolutivos ya que son los que presentaron mejores resultados, aunque encontramos otras heurísticas como colonia de hormigas.

En todos los artículos encontramos que el costo, el tiempo y la estabilidad como objetivos que siempre aparecen cuando existe el problema de replaneación en entornos dinámicos

Encontramos que cada modelo presenta características específicas y que los investigadores están haciendo un esfuerzo por modelar de una manera más cercana a la realidad, incorporando características como la productividad del equipo y el gasto en comunicación, observamos que en varios casos se toma en cuenta la curva de aprendizaje a lo largo del proyecto.

Los resultados son alentadores pues en los modelos más desarrollados las soluciones obtenidas son similares a las de proyectos en la industria y en algunos casos compitiendo con expertos que tardaron 3 horas en terminar una sola replaneación [10], mientras que los algoritmos heurísticos presentan resultados en a lo más 25 minutos. Lo anterior da una clara ventaja ya que en el mismo tiempo que tarda un experto, al usar una herramienta que automatice la replaneación se obtiene un conjunto de propuestas que presentan el mejor valor en los objetivos y que se pueden usar como guía para que los administradores de proyecto tomen decisiones.

Por todo lo anterior encontramos un nicho de oportunidad ya que en la actualidad las metodologías ágiles y en especial Scrum, han sido adoptadas por grandes empresas de software. Y lo más importante es que no existe un solo modelo que presente características para este contexto ágil y por lo tanto

tampoco ninguna herramienta que sirva de apoyo a los líderes de proyectos al momento de necesitar realizar una replaneación como consecuencia de un evento disruptivo.

# Capítulo 3

# Planeación de liberaciones en Scrum

En la actualidad grandes organizaciones dedicadas al desarrollo de software han adoptado las metodologías ágiles. En las metodologías ágiles el software se desarrolla y se entrega al cliente iterativamente. En este trabajo en particular nos enfocaremos en Scrum, la cual es una metodología ágil para la administración de proyectos que lleva más de 20 años usándose [12]. La Fig. 5 muestra un diagrama de Scrum, en el podemos ver que a las iteraciones de desarrollo se les llama *sprint*. En cada sprint se seleccionan y se desarrollan historias de usuario (HU) para agregar incrementos a la funcionalidad del sistema. Una HU es la descripción de una funcionalidad del sistema en lenguaje común y es la forma en que se describen a los requerimientos del sistema en Scrum.

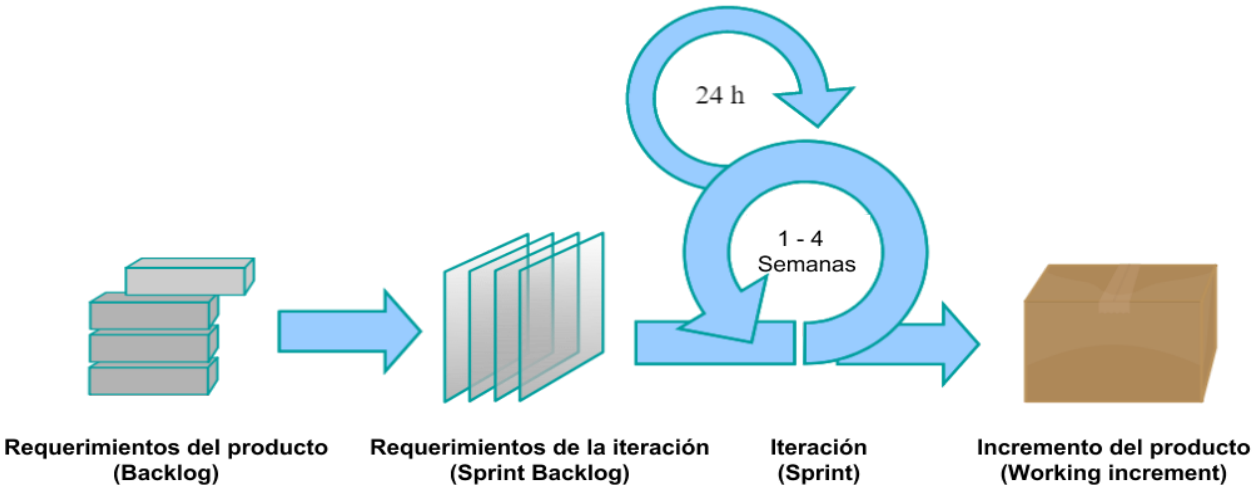



Fig. 5. Diagrama de Scrum

Después de cada sprint, al terminar el desarrollo de las historias de usuario se agrega funcionalidad al sistema, pero esta no siempre es entregada al cliente. Las entregas, como antes se mencionó también se realizan iterativamente y cada una de ellas está formada de uno o varios sprint. A estas entregas en Scrum se les conoce como *liberaciones*. En cada liberación se busca entregar al cliente un conjunto de funcionalidades que aporten valor a su negocio de forma rápida, entregando un producto ejecutable en cada una de ellas. Por ejemplo, si el proyecto es una tienda en línea, para la primer liberación se desea entregar al cliente el carrito de compras, el catálogo de productos y el pago con tarjeta de debito y

credito. Si, se entrega esta funcionalidad al cliente, el podrá comenzar a vender sus productos y comenzar a obtener valor del sistema.  Se debe  señalar que en las metodologías ágiles los tiempos de entrega de funcionalidad del proyecto tienden a ser más cortos en comparación con el desarrollo tradicional, ya que cada liberación dura apenas unas semanas. Para llevar a cabo de la mejor manera una liberación, se realiza una *planeación de liberación*.

En la planeación de liberación se seleccionan las historias de usuario necesarias para cumplir con las metas de la liberación y se asignan a algún sprint para calendarizar el orden en que serán desarrolladas. Al igual que en las metodologías tradicionales, en Scrum al realizar esta planeación lo que se busca es tomar las mejores decisiones para el proyecto, para maximizar el valor del producto que se entrega y además también minimizar el costo y el tiempo de desarrollo de cada liberación. En la Fig. 6 vemos la representación de una planeación de liberación, como se mencionó está formada por varios sprint y las historias de usuario están calendarizadas como mejor conviene al desarrollo del proyecto. Además, se muestran las dependencias entre HU, la prioridad de cada una de ellas que se representa con un color (verde para mayor prioridad, rojo para menor prioridad y amarillo prioridad media) y por último podemos ver dos valores muy importantes en Scrum: puntos de historia (número arriba de cada HU) y velocidad de sprint, que se describen en las secciones 3.1 y 3.2 respectivamente.

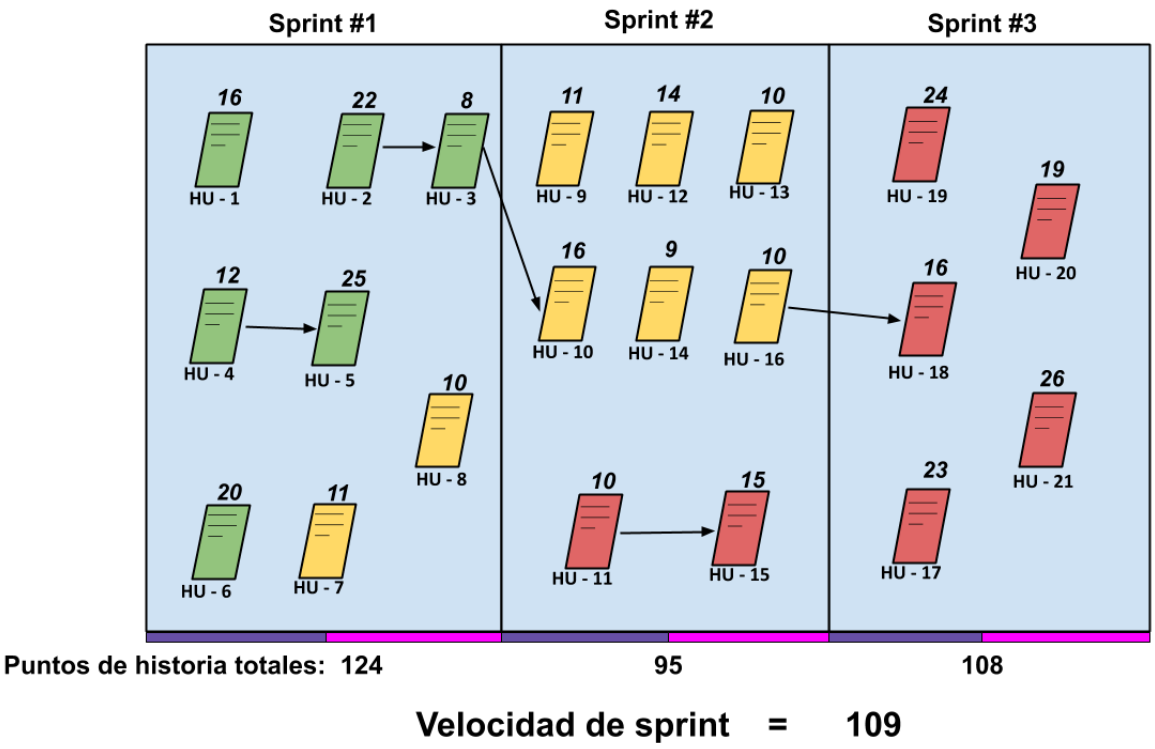



Fig. 6. Representación de una planeación de liberación.

## 3.1 Puntos de historia

Los puntos de historia son la unidad de medida que sirve para expresar una estimación del esfuerzo necesario que se requiere para implementar por completo una HU.

En las metodologías ágiles la estimación del esfuerzo para realizar las historias de usuario recae en el equipo de trabajo, ya que son ellos quienes desarrollan el proyecto, y que mejor conocen el esfuerzo necesario. Cuando se realiza la estimación se debe de tomar en cuenta: la cantidad de trabajo por realizar, la complejidad de lo que se debe hacer en la HU y cualquier riesgo o incertidumbre para realizar el trabajo.

Para realizar la estimación es común que el equipo de desarrollo utilice una técnica llamada planning poker [13]. En la Fig. 6 identificamos los puntos de historia de cada HU sobre cada una de ellas.

## 3.2 Velocidad de sprint

La velocidad es una métrica histórica de la capacidad que ha mostrado tener el equipo de desarrollar de forma completa historias de usuario en sprints anteriores. Supongamos que la suma de los puntos de historia que han podido concluir en sprint anteriores son las siguientes:

- Sprint 1: 124
- Sprint 2: 95
- Sprint 3: 108

Podemos decir que como promedio se han desarrollado 109 puntos de historia por sprint. A esta medida se le llama velocidad de sprint.

Esta medida sirve para aproximar la cantidad de trabajo que podemos realizar en el siguiente sprint y así calcular la cantidad de sprint necesarios para terminar la liberación. Si tenemos una liberación que requiere desarrollar historias de usuario cuyos puntos de historia totalizan 425 y sabemos que la velocidad del equipo es de 109 por sprint ,entonces tenemos que:

Número de sprints = 425 / 109 = 3.9

Necesitamos entonces de al menos 4 sprints para desarrollar las HU que conforman la liberación. Aunque existen conceptos adicionales en Scrum, los conceptos que hemos presentado hasta aquí son aquellos que son necesarios para entender la propuesta.

## 3.3 Tiempo extra de un equipo de desarrollo

Una característica adicional que se tomó en cuenta para desarrollar el modelo es el tiempo extra que un equipo de desarrollo puede trabajar. Ya que el trabajo extra es común no solo en el desarrollo de software sino en todos los ámbitos y empresas. Este trabajo extra sirve para aumentar la capacidad de desarrollo del equipo de software y por lo tanto sirve para aumentar la velocidad de sprint del equipo. Debemos considerar que esto tiene como consecuencia un aumento en el costo del proyecto, pues es necesario pagar ese trabajo extra. En el presente trabajo consideramos la posibilidad de que un equipo realice hasta 22.5% de trabajo extra[2], que en una jornada de 8 hrs este porcentaje representa 2 hrs extras. En la Fig. 7 se muestra un diagrama del uso del tiempo extra para aumentar la velocidad de sprint al realizar una replaneación, podemos ver a la izquierda la planeación inicial, después de un evento disruptivo se calcula la capacidad extra que puede trabajar el equipo. En la parte derecha encontramos dos posibilidades: la primera (A) es que, trabajando tiempo extra (sprints más largos y de color gris) la fecha de entrega de la liberación se mantenga, es decir, que la cantidad de sprints al realizar una replaneación no aumente. La segunda posibilidad (B) es que a veces ni trabajando tiempo extra se logrará cumplir con las fechas previstas en la planeación original, por lo que irremediablemente la cantidad de sprints aumenta y la liberación se retrasa.

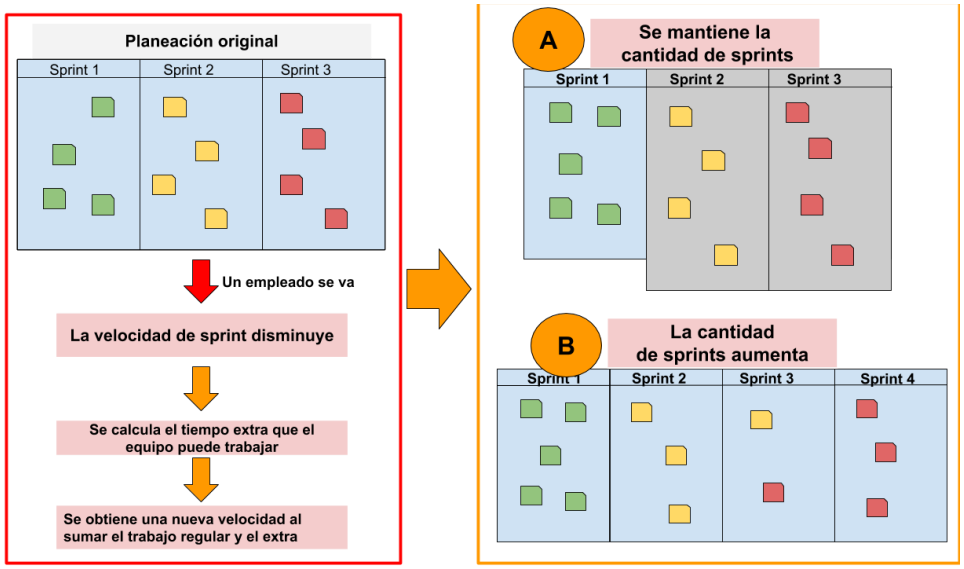



Fig. 7. Aumento de la velocidad de sprint con el tiempo extra.

[2] El artículo 66 de la ley federal del trabajo en México, marca que una jornada extraordinaria de trabajo no puede exceder de 3 horas ni de 3 días a la semana. Lo que equivale aproximadamente a un 22.5% en una semana de trabajo de lunes a viernes.

## 3.4 Eventos disruptivos

En la fase de ejecución de un proyecto de software, es común que ocurran eventos que por su gravedad impactan la planeación original y obliguen a al Scrum Master del proyecto a realizar una replaneación. A estos eventos que por su gravedad obliguen a realizar una planeación, les llamaremos eventos disruptivos.

### 3.4.1 Tipos de eventos disruptivos

Los eventos disruptivos que simularemos se obtuvieron de un análisis de cuáles son los eventos más comunes que ocurren en el desarrollo de software en el contexto de Scrum y que presentan un impacto grave en una liberación.  A continuación se definen los 6 eventos disruptivos que en este trabajo, obligan a realizar una replaneación:

1) **No se termina una HU en el sprint en el que se está ejecutando**. En algún momento del sprint no se termina una o más de las historias de usuario, es decir, se retrasan. Es necesario realizar una replaneación ya que este evento afectará a los sprints siguientes ya que cada uno de ellos ya tiene HU asignadas y ahora además, deberán desarrollarse las HU que se retrasaron.
2) **Se elimina una HU planeada para un sprint posterior al que se está ejecutando**. Este es el caso contrario al evento anterior y es que en las metodologías ágiles los requerimientos son muy cambiantes, por lo que puede ocurrir que en algún sprint de la liberación, se decida que ya no será necesario desarrollar una o más HU. Se necesita una replaneación ya que asi podriamos re distribuir como mejor convenga al plan y tal vez ganar un poco de tiempo para la liberación.
3) **Se agrega una nueva HU**. Como dijimos en las metodologías ágiles nunca se sabe cuándo cambiará o en este caso, llegará funcionalidad nueva. Por lo que es común que en algún momento del desarrollo del plan de liberación lleguen HU que se deben desarrollar como parte de la liberación en curso. Por lo anterior, el Scrum Master del proyecto debe reajustar para poder desarrollar además de la carga de trabajo regular, las nuevas HU que llegaron. Este evento también se presenta cuando tenemos que arreglar un bug, pues comúnmente pertenece a una liberación pasada y se debe entregar lo antes posible.
4) **Cuando se va un empleado permanentemente o temporalmente**. En la industria del software es muy común que los empleados migren, pero además de esto, como cualquier otra persona puede enfermarse, ser despedidos o simplemente salir de vacaciones. Cuando el periodo de ausencia de un empleado es considerable, se debe realizar una replaneación ya que el equipo de desarrollo pierde parte de su capacidad de trabajo.

5) **Se agrega un empleado al proyecto**. En el caso contrario al evento anterior, cuando un empleado regresa de vacaciones, de enfermedad o es recién contratado, el Scrum Master del proyecto ahora cuenta con más capacidad de desarrollo para el proyecto, por lo que es necesario realizar una replaneación que pudiera permitir mejorar la planeación actual.
6) **Se sustituye un empleado**. Cuando un empleado es sustituido por otro en el proyecto, se debe realizar una replaneación ya que los empleados no son iguales en experiencia, ni en conocimiento del proyecto, etc. Por ello, el plan original se debe ajustar a la capacidad del equipo con el desarrollador que acaba de llegar.

Una vez que se definieron los eventos disruptivos, éstos se dividieron en dos grupos, el primer grupo son los que afectan la cantidad total de puntos de historia que se busca entregar como parte de la liberación, este grupo abarca los eventos: 1, 2 y 3.

El segundo grupo de eventos disruptivos contiene a los eventos que afectan a la velocidad de sprint y son: 4, 5 y 6. Los eventos disruptivos que afectan a los miembros del equipo, afectan a la capacidad de desarrollo y por lo tanto repercute directamente en la velocidad del sprint. Una rápida respuesta es necesaria ya que en un calendario ajustado podría significar un retraso significativo en el proyecto.

# Capítulo 4

# Modelo desarrollado

Después del análisis de las características más importantes de Scrum, en el presente trabajo proponemos un modelo para el problema de replaneación de liberaciones ante la ocurrencia de eventos disruptivos. De igual manera presentamos una herramienta que sirva como apoyo al Scrum Master del proyecto para el problema que aquí abordamos y al cual llamaremos: problema de replaneación de liberaciones en proyectos ágiles de software (RLPAS).

## 4.1 Problema de replaneación de liberaciones en proyectos ágiles de software (RLPAS)

El RLPAS lo entenderemos como, que en respuesta a un evento disruptivo se debe ajustar lo antes posible y con el menor número de afectaciones a la planeación original, es decir, se debe realizar una replaneación, para evitar que el costo total, el tiempo de finalización y los demás objetivos importantes no se vean afectados en el proyecto, o en su defecto, que la afectación sea mínima.

Es importante realizar una replaneación para minimizar las consecuencias, ya que todas las afectaciones después del evento disruptivo deben ser absorbidas principalmente por la empresa que lo desarrolla y esto se traduce en pérdidas económicas, además de una mala imagen ante el cliente y posiblemente sanciones si existen retrasos en la entrega. En ocasiones una mala replaneación puede llevar a incrementar los costos por encima del presupuesto y esto puede resultar en la cancelación total del proyecto. Al realizar una planeación o replaneación de la liberación, se considera la asignación de HU al sprint que mejor convenga al proyecto, tomando en cuenta la duración, prioridad y dependencias entre éstas.

## 4.2 Definición de los objetivos de replaneación

En el presente trabajo consideraremos cinco objetivos al realizar una replaneación, los cuales después del análisis realizado se decidió que eran los más importantes en el contexto estudiado. Como se mencionó en las secciones anteriores el tiempo y el costo son los objetivos más comunes en la

planeación y replaneación de proyectos de software, ya que son los que más impacto tienen con el cliente, ya que el costo impacta en las ganancias y el tiempo en el cumplimiento de las fechas de entrega. El tercer objetivo que se tomó en cuenta es el de la estabilidad al realizar una replaneación, ya que al ajustar la planeación original después de un evento disruptivo se busca que se realice con el menor número de cambios respecto al plan original, pues se considera que la planeación original fue propuesta considerando lo que es mejor para el proyecto. Los últimos dos objetivos son parte de las aportaciones que realiza este trabajo, pues están pensados en el contexto ágil del desarrollo de software y los cuales son: el desaprovechamiento de desarrollo del equipo y el valor de liberación. Al optimizar los objetivos anteriores buscamos minimizar el impacto de los eventos disruptivos sobre el proyecto, a continuación definiremos los cinco objetivos:

- **Costo**. Es el costo de la liberación que se calcula sumando el costo de cada uno de los sprints, dependiendo de si se necesita tiempo extra o no en algunos de ellos. Para calcularlo, como se muestra en la Fig. 8, se obtiene el costo de la velocidad de sprint regular en cada sprint y el costo de la velocidad de sprint del trabajo extra del equipo. Por lo que el costo de cada sprint será mayor mientras más tiempo extra se utilice. El costo de todos los sprint que no utilizan tiempo extra es el mismo, ya que el equipo de desarrollo tiene un salario fijo. Al realizar la optimización se busca minimizar este objetivo.

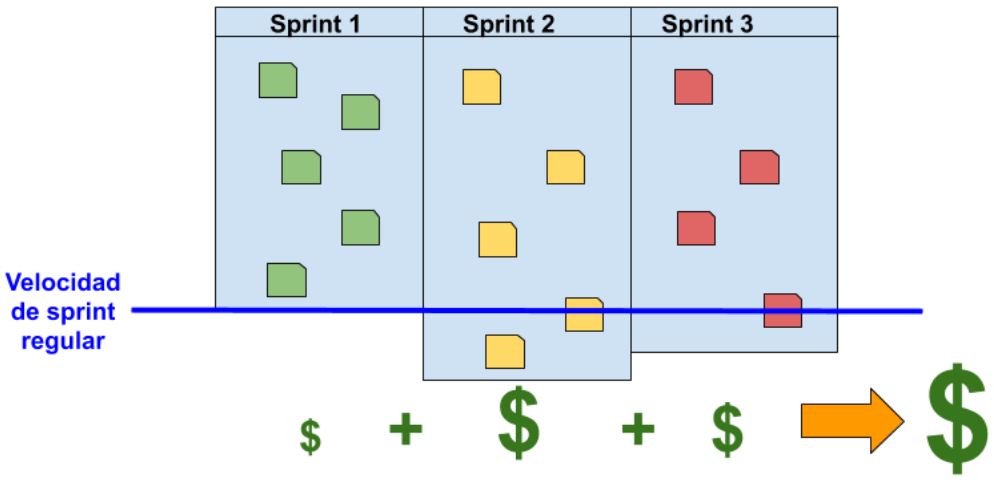



Fig. 8. Costo en una liberación.

- **Tiempo**. Es la cantidad de sprints que son necesarios en la replaneación. Como podemos observar en la Fig. 9, al realizar una replaneación un buen valor de tiempo es cuando el número de sprint no aumenta o tal vez hasta disminuye, ya que el proyecto no se retrasa, en cambio un mal valor de este objetivo es al aumentar sprints al realizar la replaneación ya que la liberación sufrirá un retraso. En la optimización se busca minimizar este objetivo.

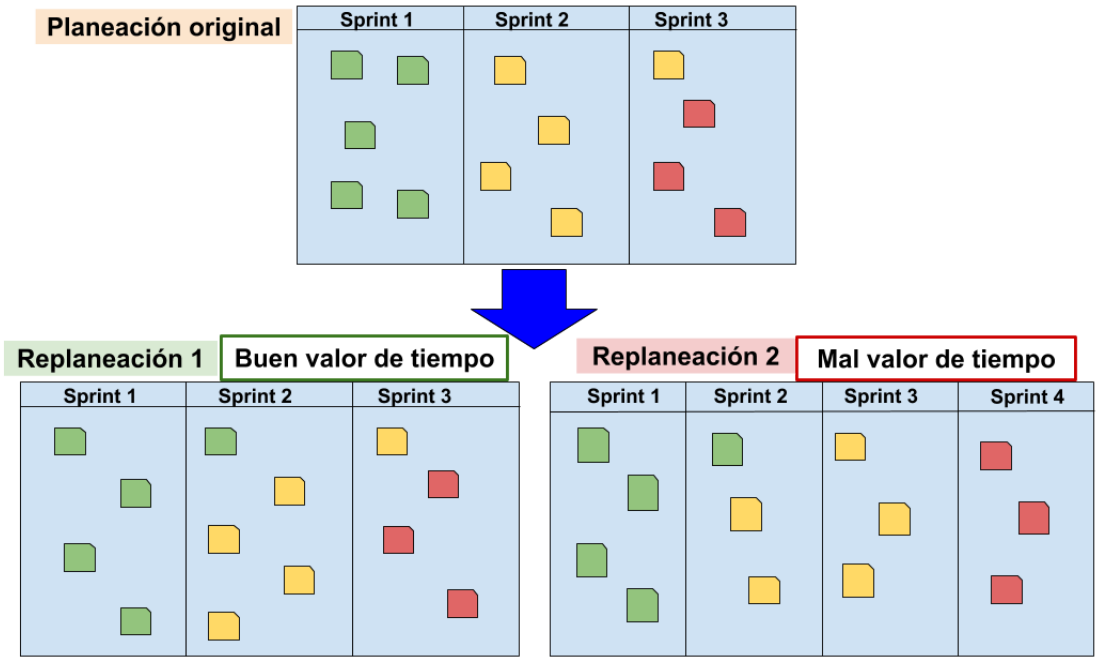



Fig. 9. Tiempo que tarda una liberación.

- **Estabilidad**. Como se comentó anteriormente en esta sección, al realizar una replaneación se busca minimizar las diferencias entre la asignación de historias de usuario a los sprints en la planeación de liberación original, y la que resulta de la replaneación. Ya que la liberación original se considera que fue pensada como lo mejor para el proyecto, además hay algunas HU que requieren de trabajo previo, por lo que sí se mueven de sprints, este trabajo se perdería. La Fig. 10 presenta un buen valor de estabilidad en el lado izquierdo, podemos ver que al realizar la replaneación, el plan de liberación original no cambia y solo se agrega una nueva HU en un sprint. Del lado derecho encontramos un mal valor para este objetivo ya que al realizar la replaneación se realizaron demasiados cambios en la asignación de HU. Como dijimos este objetivo se buscará minimizar.

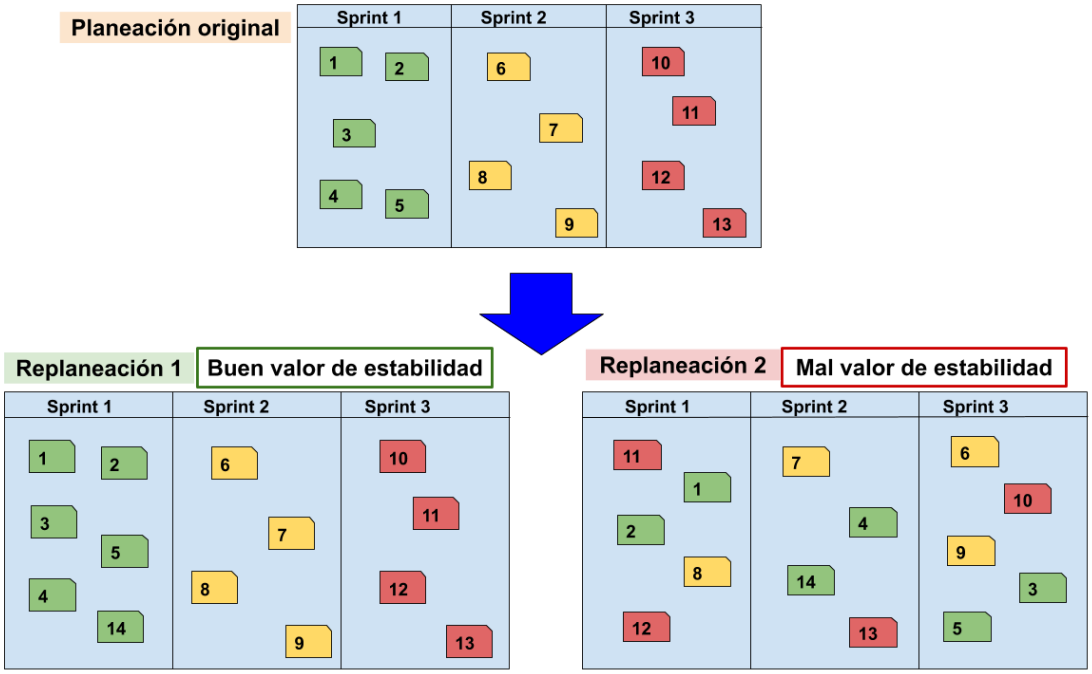



Fig. 10. Estabilidad al realizar la planeación de liberación.

- **Desaprovechamiento de la capacidad de desarrollo**. Este es el primer objetivo que definimos especialmente para el contexto ágil, ya que con él se busca que la capacidad de desarrollo de nuestro equipo se ocupe al máximo. Como bien es sabido los equipos de desarrollo son muy costosos y el tiempo de entrega de una liberación tiende a ser muy corto. Una buena replaneación debe asegurar que en cada sprint se ocupe al máximo la capacidad de desarrollo disponible, lo cual se traduce en que la diferencia de la suma de los puntos de historia de las HU asignadas al sprint y la velocidad de sprint sea mínima. Al ajustar el plan después de un evento disruptivo, la nueva asignación de HU debe buscar la mejor combinación de ellas. Para calcular este objetivo se mide la velocidad de sprint que la replaneación está “desaprovechando”. Es decir medimos la velocidad que se desperdicia en cada sprint, esto es la diferencia entre la velocidad del sprint y el total de puntos de historia de las HU asignadas al Sprint. Si en un sprint se ocupa tiempo extra este también se toma como desaprovechamiento, ya que si aun hay tiempo regular disponible en otro sprint, no debería usarse tiempo extra. Con este objetivo buscamos entonces minimizar el desaprovechamiento de la capacidad de trabajo regular de nuestro equipo de desarrollo. Al realizar una replaneación este objetivo se busca minimizar. En la Fig. 11 vemos la representación de este objetivo, a la izquierda encontramos un buen valor ya que las distribución de las HU en los sprint es similar y se usa aprovecha de buena forma el tiempo de desarrollo. A la derecha se muestra un mal valor ya que podemos darnos cuenta que en los primeros sprint (de color gris) hay una sobrecarga de asignación y por lo tanto se usa

mucho tiempo extra, y en los últimos dos sprint aún existe tiempo regular disponible (marcado con color rojo).

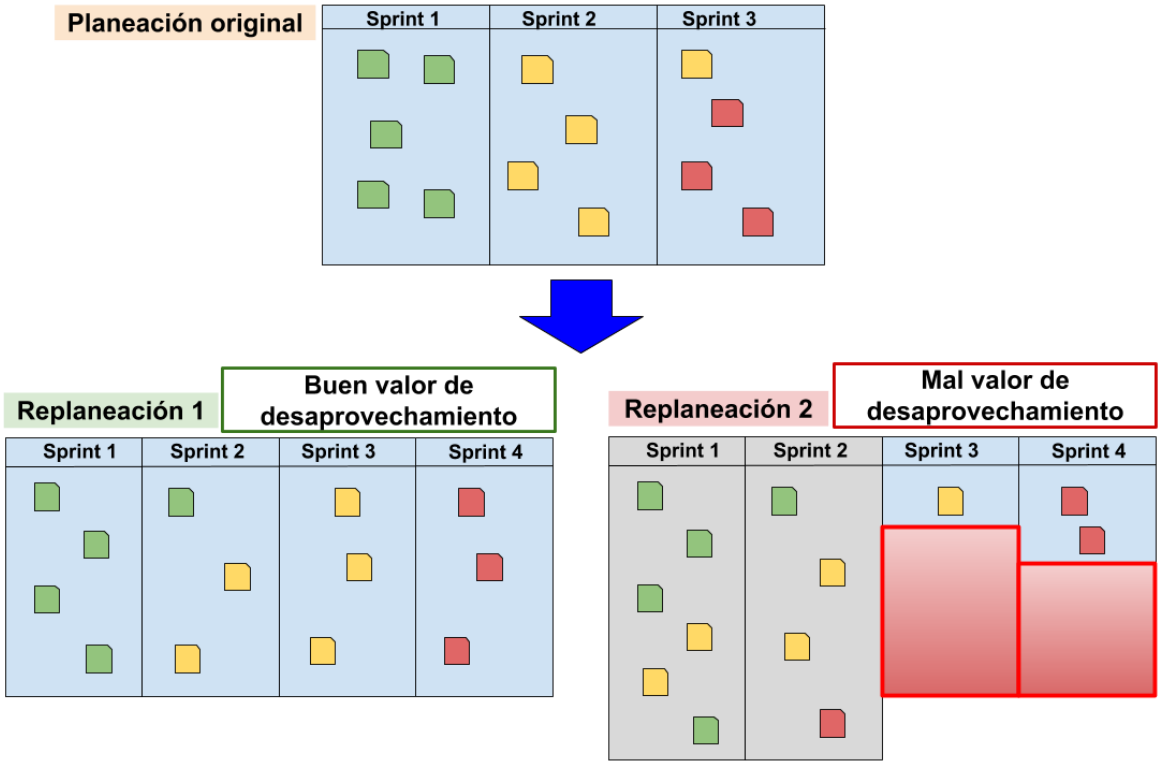



Fig. 11. Desaprovechamiento de la capacidad de desarrollo en una liberación.

- **Valor de liberación**. Para que el modelo se ajuste adecuadamente a la realidad de los proyectos desarrollados con Scrum, sabemos que en la ejecución de una planeación o replaneación se deben desarrollar las HU con mayor prioridad en los primeros sprints. Con ello buscamos entregar en una liberación, el conjunto de funcionalidad que aporte el mayor valor al negocio del cliente. Al tener poco tiempo de desarrollo ya que una liberación dura apenas unas semanas, debemos garantizar que al realizar una replaneación después de un evento disruptivo, se asignen las HU con mayor importancia para el negocio del cliente a los primeros sprints. Con el fin de aumentar la probabilidad de tener el tiempo suficiente para poder desarrollarlas y cumplir con los objetivos de la liberación. Así, si algo pasa y existe un retraso en la liberación, al menos se tiene la seguridad de que la funcionalidad más importante ya está terminada en los primeros sprints. Este es el segundo objetivo que en este modelo está completamente inspirado en las metodologías ágiles y que es parte de nuestras aportaciones. Este objetivo es el único que buscamos maximizar. Por último podemos ver un ejemplo en la Fig. 12, donde tenemos un buen valor de liberación en la parte izquierda ya que está asignado lo más importante para el negocio

del cliente en los primeros sprints y después lo menos importante. Una mala decisión se encuentra a la derecha y es dejar lo más importante al final.

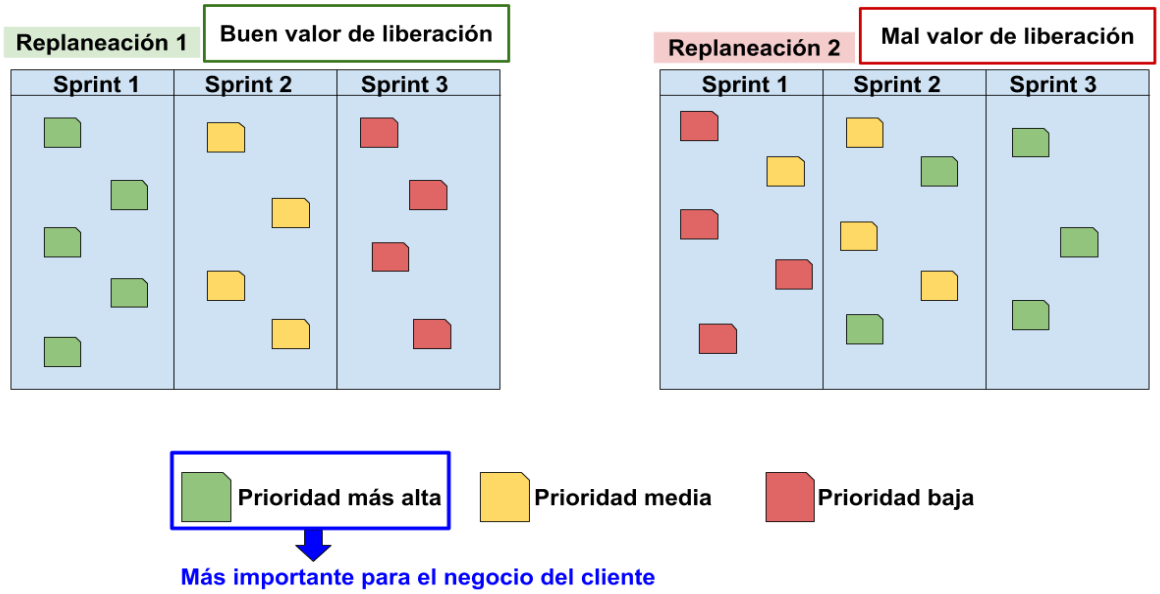



Fig. 12. Dos propuestas de planeación, mejor y peor valor de liberación.

## 4.3 Formulación multiobjetivo del problema RLPAS

En este trabajo, el problema de replaneación de liberaciones de proyectos ágiles de software se modela como un problema de optimización multiobjetivo, ya que al validar independientemente cada objetivo podemos encontrar un conjunto de soluciones que presente los mejores valores para cada uno de ellos. Además, de esta manera, al realizar una replaneación se busca minimizar el impacto de eventos disruptivos en todos los objetivos que se describieron anteriormente. Por lo que como propuesta de solución se busca entregar un conjunto de las mejores soluciones encontradas. A continuación, presentamos formalmente el modelo propuesto.

En un proyecto ágil de software usando Scrum, se tiene un conjunto $H = \{h_1, h_2, \ldots, h_n\}$ de $n$ historias de usuario $h_i$, $\forall\, i \in \{1, \ldots, n\}$, que se tienen que desarrollar. Cada historia de usuario $h_i$ tiene una prioridad $\pi h_i$ y esfuerzo asociado $ph_i$ medido en puntos de historia.

Definiremos un plan de liberación $L$, como el conjunto $L = \{S_1, S_2, \ldots, S_m\}$ de $m$ sprints $S_i$, $\forall\, i \in \{1, \ldots, m\}$. A su vez, cada sprint $S_i$ es el conjunto $S_i = \{h_1^i, h_2^i, \ldots, h_{n_i}^i\} \subseteq H$ de las $n_i$ historias de usuario $h_j^i$, $\forall\, j \in \{1, \ldots, n_i\}$, que se tienen que desarrollar en el sprint $S_i$. Así, los puntos de historia $phS_i$ del sprint $S_i$ se definen como

$$phS_i = \sum_{j=1}^{n_i} ph_j^i \qquad (1)$$

Todos los sprints son conjuntos disjuntos, de tal forma que $S_i \cap S_j = \emptyset,\ \forall\ i, j \in \{1, \ldots, m\},\ i \neq j$ , y $S_1 \cup S_2 \cup \cdots \cup S_m = H = \{h_1, h_2, \ldots, h_n\}$ contiene a las $n$ historias de usuario $h_j$ que se tienen que desarrollar en el plan de liberación $L$.

Para poder desarrollar las historias de usuario asignadas al sprint $S_i,\ \forall\ i \in \{1, \ldots, m\}$, se cuenta con un equipo de trabajo de $k$ empleados y cada empleado $e_j,\ \forall\ j \in \{1, \ldots, k\}$, aporta una velocidad de desarrollo $ve_j$ a la capacidad de desarrollo del equipo. De esta forma, definimos la velocidad $vS_i$ del sprint $S_i$ como

$$vS_i = \sum_{j=1}^{k} ve_j \tag{2}$$

Es decir, sumamos la capacidad que ha mostrado tener cada uno de los empleados en un sprint. Así obtenemos la capacidad total de desarrollo de nuestro equipo por sprint.

Habiendo definido formalmente el problema, ahora podemos plantear las cinco funciones objetivo que se consideran en este trabajo. Primero, el costo del proyecto se define como

$$Costo = \sum_{i=1}^{m} (\ vS_i\ +\ veS_i) \tag{3}$$

en donde $veS_i$ es la velocidad extra del sprint $S_i$: $veS_i = 0$ si $phS_i - vS_i \leq 0$ y $veS_i = phS_i - vS_i$ si $phS_i - vS_i > 0$. En esta ecuación, medimos la cantidad de puntos de historia extra necesarios en el sprint y su costo. El cual se suma al costo regular de un sprint y se obtiene el costo total.

El tiempo de entrega del proyecto es simplemente la cantidad de sprints considerados, es decir

$$Tiempo\ =\ m \tag{4}$$

La estabilidad de la replaneación se define como

$$Estabilidad = \frac{1}{(m-1)\cdot n} \sum_{i=1}^{n} |\ sprint(i,t) - sprint(i,t-1)\ | \tag{5}$$

en donde t hace referencia a la replaneación y t-1 a la planeación original. La función $sprint(i,t)$ indica el sprint al cual está asignada la historia de usuario $h_i$ al tiempo $t$, por lo que $sprint(i,t) \in \{1, \ldots, m\}$. El factor $1/(m-1)\cdot n$ se incluye para obtener un valor normalizado. Se busca normalizar el valor de este objetivo para que el rango de posibles valores, se encuentre entre 0 y 1. Donde 0 representa el mejor valor y 1 el peor valor de estabilidad. Con esta función medimos los sprint de distancia que tiene cada HU en la replaneación en comparación de su asignación en la planeación inicial. Es decir, medimos

cuántos sprint se movió entre la replaneación (t) y la planeación (t-1) después del evento disruptivo. Si al realizar la replaneación no se realizó ninguna reasignación de las HU, el valor de la estabilidad es cero.

El desaprovechamiento de la capacidad de desarrollo se define como

$$Desaprovechamiento = \sum_{i=1}^{m} | vS_i - phS_i | \qquad (6)$$

Es decir con esta ecuación se mide la diferencia que existe entre la velocidad regular de cada sprint y la cantidad de puntos de historia que tiene asignados.

Finalmente, el valor de liberación se define como

$$Valor\ liberacion = \frac{1}{a} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \sum_{k \in S_i} \sum_{l \in S_j} (\pi h_k - \pi h_l)(j - i) \qquad (7)$$

en donde $a = 2(m-1)n^2$. Esta función objetivo, da como resultado un número entre -1 y 1, en donde 1 significa que la planeación tiene el mejor valor de liberación y -1 el peor valor de liberación. En la función se penaliza cuando una HU de menor prioridad está asignada a un sprint antes que una de mayor prioridad. Las dos primeras sumatorias recorren el sprint actual y el siguiente. Las otras dos sumatorias se encargan de realizar la diferencia entre las prioridades de cada una de las HU en los sprints (el actual y el siguiente).

## 4.4 Modelo de dominio del RLPAS

Como resultado del análisis del RLPAS y del desarrollo del modelo matemático que lo define, en esta sección se presenta el modelo de dominio del problema.

Para comenzar con el desarrollo de esta herramienta y después del análisis del uso de Scrum, de nuestro modelo y de la replaneación, presentamos en la Fig. 13 el modelo de dominio del RLPAS, el cual fue resultado del diseño de la herramienta. El modelo de dominio se presenta usando un diagrama de clases compuesto por las entidades más importantes y sus atributos en el RLPAS. En este modelo encontramos cuatro clases: Empleado, HistoriaUsuario, Sprint y PlanLiberacion. Se muestran, además de los atributos de cada entidad, las relaciones entre las entidades que conforman el dominio del problema, donde un plan de liberación puede estar compuesto por uno o más sprints y un sprint puede tener asignada una o más HU. En el caso de la clase HistoriaUsuario, encontramos una relación con ella misma, esto modela las dependencias que podemos encontrar entre historias de usuario. La clase Empleado se relaciona con el PlanLiberacion ya que el conjunto de empleados, es el equipo que participa en la ejecución del plan. Por último, podemos destacar los atributos puntosHistoria y velocidad en las clases HistoriaUsuario y Sprint respectivamente, los cuales son parte fundamental de Scrum y son además, aportaciones de este trabajo.

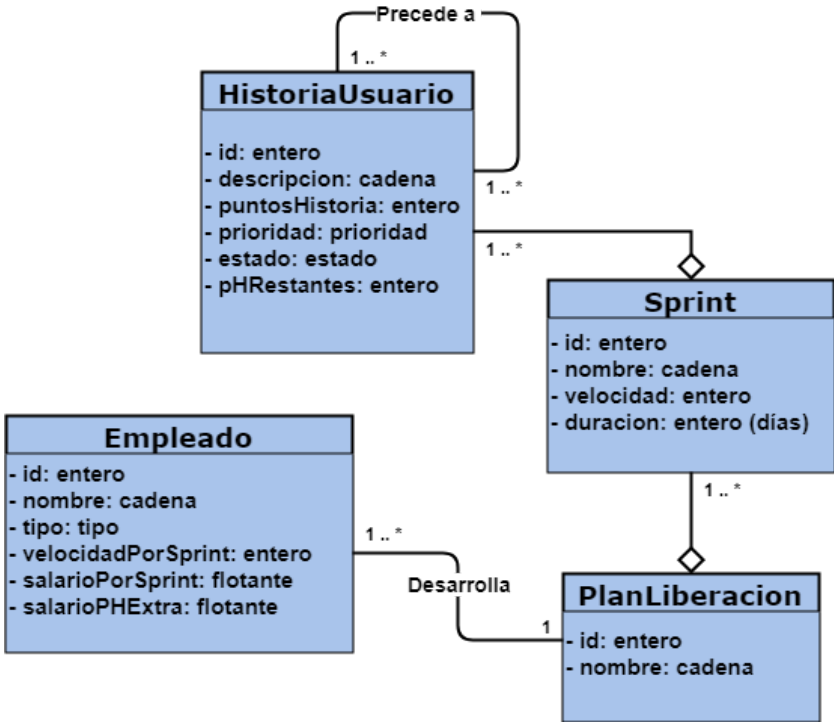



Fig. 13. Modelo de dominio para el RLPAS.

En el modelo de dominio desarrollado, consideramos tres tipos de empleados: junior, senior y master. Los cuales se representan en la clase Empleado en su atributo tipo. Como en los proyectos reales, existen empleados con menor o mayor experiencia. Los de mayor experiencia aportan una mayor velocidad al sprint (ya sea la de base o la que considera el tiempo extra), ya que son más capaces al resolver una HU debido a su experiencia. En la clase Empleado, esta capacidad se representa en el atributo velocidadPorSprint, que es la cantidad de puntos de historia que puede desarrollar el empleado en un sprint.

# Capítulo 5

# Algoritmo genético multiobjetivo para resolver el RLPAS

Después del desarrollo del modelo matemático, de la identificación de los eventos disruptivos que se considerarán en este trabajo y del modelo de dominio, en este capítulo presentaremos la solución que se desarrolló como propuesta para el RLPAS. Al modelar el problema de replaneación como uno multiobjetivo, el resultado es, idealmente, un conjunto de soluciones que presentan los mejores valores encontrados para cada uno de los objetivos y soluciones que presentan una compensación entre ellos.

Existen varios métodos para resolver problemas de optimización multiobjetivo, por las características del problema bajo estudio y debido a que no existe un método que resuelva el RLPAS o cualquier otro problema de asignación en una cantidad de tiempo razonable, hemos empleado un método metaheurístico, en particular, un algoritmo genético (AG). Comenzaremos en la siguiente sección definiendo las características de un AG.

## 5.1 Algoritmos Genéticos

Los algoritmos genéticos son algoritmos basados en población y en los principios de evolución mediante la selección natural. La búsqueda que realiza un algoritmo genético debe tener un balance entre explotar y explorar el espacio de búsqueda [15]. En la Fig. 14 se muestra el diagrama del funcionamiento de un AG, en ella podemos ver que el primer paso es crear una población inicial, después, se califica a cada uno de los individuos de esta población, a esta calificación se le conoce como aptitud. El tercer paso es aplicar los operadores genéticos a la población creada los cuales son: selección, cruzamiento y mutación, esto con el fin de buscar que la población mejore y asi tal vez obtener individuos más aptos. Entonces ahora tenemos a los individuos padre y los individuos hijo que se obtuvieron de aplicar los operadores genéticos, entonces se seleccionan de todos ellos a los individuos que formarán parte de la nueva población, ya que está siempre está acotada a un número máximo de individuos. Esta nueva población se evalúa y si aún no se cumple el criterio de paro del algoritmo, entonces de nuevo se aplican los operadores genéticos para buscar mejorar y obtener una nueva población cada vez, esto se realiza repetidamente hasta alcanzar el criterio de paro. Este criterio puede ser por ejemplo un número de generaciones, es decir, un número de repeticiones del ciclo anteriormente descrito. Si el criterio de paro

se ha cumplido, se muestra la o las mejores solución encontrada hasta ese momento. Un AG al estar basado en población cubre  la necesidad de obtener de manera rápida un conjunto de propuestas de replaneación de liberaciones después de ocurrir un evento disruptivo. Como resultado de la aplicación del AG tendremos no solo una sino una población de propuestas de solución.

Considerando las características presentadas en el párrafo anterior, un AG se compone de cuatro partes importantes:

1. Representación de la solución como un cromosoma.
2. Aptitud de un cromosoma.
3. Generación de una población.
4. Aplicación repetida de los operadores genéticos.

Estos componentes se describen en las siguientes secciones.

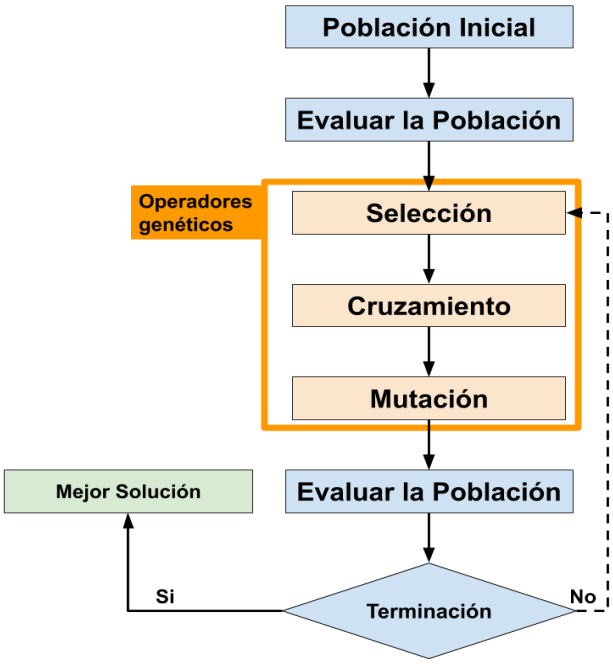



Fig. 14. Diagrama del funcionamiento de un AG.

### 5.1.1 Representación de la solución como cromosoma

La representación de la solución como un cromosoma es lo más importante en un algoritmo genético, ya que una buena representación facilita y hace más eficiente la aplicación de los operadores genéticos. Un cromosoma es una representación en forma de arreglo, es decir, de la forma $a_1 a_2 a_3 \ldots a_n$, en donde cada posición $i$, se le llama gen y a cada $a_i$ se le conoce como alelo.

### 5.1.2 Aptitud de un cromosoma

La aptitud es la capacidad que tienen un individuo de ser mejor frente a otros. Es decir, la aptitud nos muestra que tan bueno es un cromosoma como solución a nuestro problema. En un AG se define una función de aptitud, la cual se calcula con los valores del cromosoma.

### 5.1.3 Generación de una población

Los AG son algoritmos basados en población, en un AG una población es el conjunto de cromosomas sobre los cuales se aplicarán los operadores genéticos con el fin de que esta mejore generación a generación. Una población tiene un límite en la cantidad de individuos que la conforman a este límite se le conoce como tamaño de la población.

### 5.1.4 Operadores genéticos

Los operadores genéticos son el conjunto de operaciones que se aplican a una población para buscar obtener individuos más aptos de una generación a otra. Por lo regular siempre son los siguientes:

- **Selección:** La selección es la forma en que se seleccionan dos cromosomas de la población, para que estos sean los cromosomas padre al realizar el cruzamiento. Existen varios métodos para realizar la selección, pero en la mayoría de ellos los individuos más aptos son los que tienen más probabilidad de ser seleccionados.
- **Cruzamiento:** Una vez seleccionados los cromosomas, estos se combinan para obtener dos cromosomas hijo. En un AG la cruza se realiza con el fin de que los cromosomas hijo puedan tener una mejor aptitud que los padres, al heredar probablemente lo mejor de cada uno de ellos. De igual manera que en la selección existen varios métodos de cruzamiento, el más popular es la cruza en un punto, pero como todo en los AG el tipo de implementación dependerá del problema.
- **Mutación:** Este operador genético sirve para intentar agregar material genético a la población que no se ha podido desarrollar con los otros, es decir, se intentan agregar valores en el cromosoma que no se ha podido obtener con la cruza. Por lo regular la mutación selecciona uno o varios genes del sprint y cambia el valor aleatoriamente.

## 5.2 Algoritmo genético multiobjetivo propuesto

En la sección 5.1 fueron descritas las características generales de un AG mono-objetivo y como se mencionó al principio de este capítulo, el problema de RLPAS se modeló como un problema

multiobjetivo. Por lo que fue necesario extender las características de un AG convencional a varios objetivos. A continuación se presentan las características propias del AG que presentamos como propuesta. Primero comenzaremos con la representación de una propuesta de planeación como un cromosoma y definiremos el concepto de dominación. Esto con el fin de presentar conceptos que nos serán útiles en las subsecciones siguientes.

### 5.2.1 Representación de una replaneación como cromosoma

Para el RLPAS la forma en que se representa una propuesta de replaneación se muestra en la Fig. 15, en ella podemos observar que en cada cromosoma, el primer gen representa a la primer HU, el segundo gen representa a la segunda HU y así sucesivamente. El valor del alelo en cada gen, nos dice en que sprint está asignado la HU a la que representa.

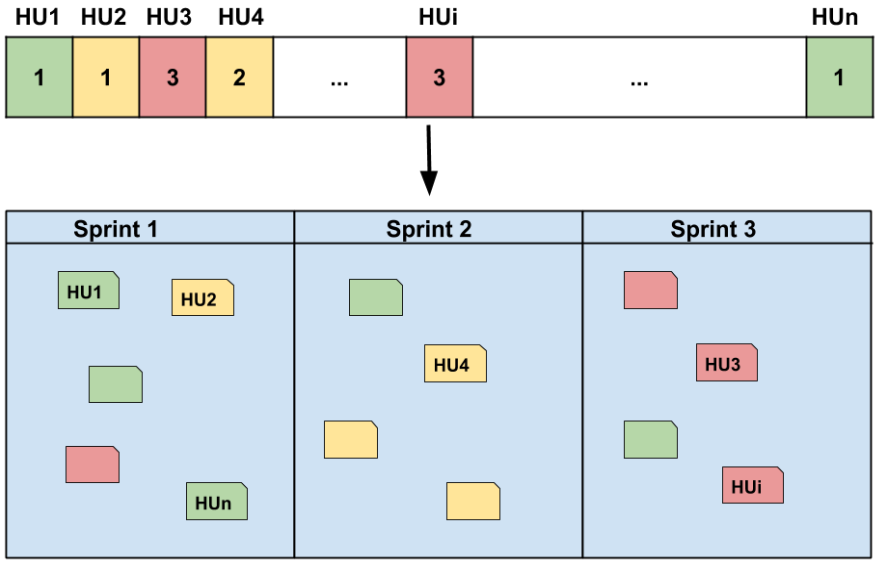



Fig 15. Ejemplo de la representación de una planeación como un cromosoma.

### 5.2.2 Aptitud de un cromosoma aplicando el concepto de dominancia

En el modelo desarrollado para el RLPAS todos los objetivos se consideran igual de importantes, por lo tanto ninguno tiene una ponderación especial. Por lo que en la propuesta de solución que en este trabajo se presenta, el algoritmo calcula la aptitud de cada una de las propuestas de replaneación implementando el concepto de *dominancia*.

Se dice que una solución $X$ *domina* a otra solución $Y$ si $X$ es al menos tan buena como $Y$ para todos los objetivos y es estrictamente mejor en al menos uno de ellos. Si una solución no es dominada por ninguna otra, se utiliza el concepto de *no dominancia*. Por lo anterior, decimos que la aptitud de cada individuo depende de su nivel de no dominancia, donde un nivel de no dominancia de cero representa la

mejor aptitud. Es decir, cuando un individuo presenta un nivel de no dominancia de cero, significa que no es dominado por ningún otro y por lo tanto tiene una mejor aptitud que otros individuos a los cuales domina.

La Fig. 16 muestra un ejemplo donde vemos que la solución $X$ domina a $Y$ ya que al comparar los valores, cuando se buscan minimizar los objetivos, la solución $X$ es mejor en $f_1$ y $f_2$. Se muestra en azul a cada solución del conjunto de soluciones no dominadas y en anaranjado las soluciones dominadas. Este ejemplo es para el caso de dos objetivos, pero se puede extender para más objetivos, en el AG multiobjetivo que presentamos para el RLPAS, se usan cinco objetivos.

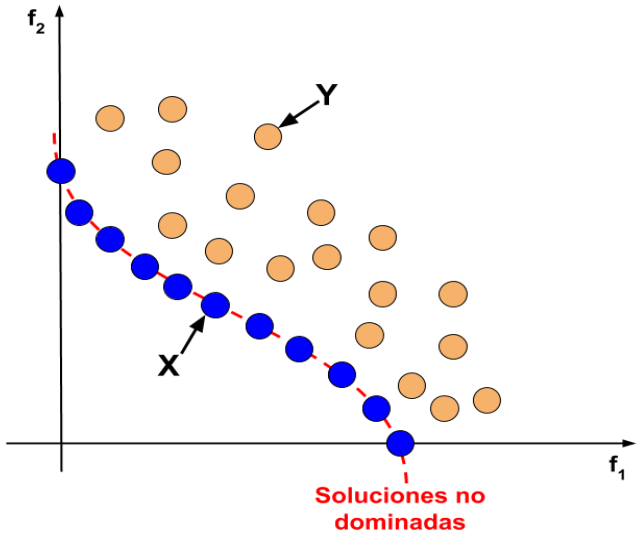



Fig. 16. Dominancia.

### 5.2.3 NSGA-II

Por las características multiobjetivo de nuestro problema, se decidió implementar un AG basado en NSGA-II, ya que NSGA-II es un AG multiobjetivo, que además presenta varias ventajas para el RLPAS, las cuales describiremos a continuación:

1. **Presenta una clasificación rápida basada en la no dominancia**, en esta, se comparan todos los individuos de la población y se les asigna un nivel de no dominancia. Este nivel como antes se describió se usa para asignar la aptitud de cada individuo y clasificar a la población.
2. **Presenta elitismo**, debido a que como solución al RLPAS se busca presentar un conjunto de propuestas de replaneación. NSGA-II nos da ventaja, ya que implementa elitismo, por lo que generación a generación solo sobreviven las individuos con mejor aptitud, es decir, los de menor nivel de no dominancia.

3. **Conserva la diversidad de la población**,a través de la implementación de la distancia de amontonamiento de un individuo. Para calcular esta distancia los individuos se ordenan de forma creciente en cada uno de sus objetivos y se mide la distancia entre ellos. Ésta se obtiene calculando la distancia que tiene el individuo con sus dos vecinos más cercanos. Si el individuo se encuentra en un extremo del espacio objetivo, entonces se le asigna una distancia infinita, ya que a su lado no se encuentran más individuos. Después de calcular el nivel de no dominancia y la distancia de amontonamiento, esta característica es usada en el operador de selección. Si dos individuos tienen el mismo nivel de no dominancia entonces se selecciona el que tenga una mayor distancia. Esto ayuda a nuestra propuesta a mantener la diversidad en los objetivos de los individuos en el conjunto solución.

### 5.2.4 Generación de una población

Para la creación de la población inicial, la primera vez que se ejecuta el algoritmo se crean los individuos de la población inicial a partir de la planeación original. Como se muestra en la Fig. 17 de la planeación inicial se construye un cromosoma padre. Los demás miembros de la población, los cuales podemos encontrar en el recuadro azul, son variaciones del padre, los cuales se obtienen de seleccionar genes al azar, después a cada uno de estos genes se les asigna un valor (sprint) nuevo en su alelo, también al azar. En el ejemplo, vemos en azul los valores que fueron modificados en cada uno de los cromosomas hijo.

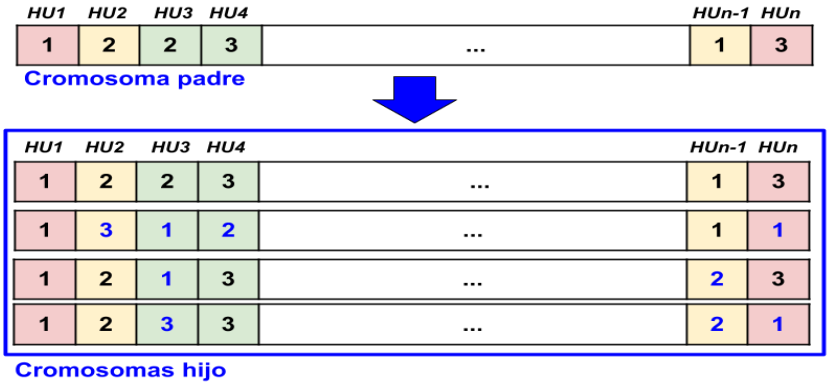



Fig. 17. Creación de la población a partir de un cromosoma inicial.

El procedimiento anterior solo se realiza la primera vez que se ejecuta el algoritmo, ya que al realizar la primera ejecución, el resultado es una población (conjunto) de propuestas de replaneación. Posteriormente si ocurre otro evento disruptivo, se usa el conjunto resultado de la ejecución anterior como población inicial, ya que este contiene todas las propuestas para la planeación de liberación que está en curso.

### 5.2.5 Operadores genéticos en el AG propuesto

Una vez que se definió el cromosoma y la forma en que se obtiene la población inicial, toca el turno a los operadores genéticos. En la Fig. 14 se muestran, dentro del cuadro anaranjado, los tres operadores genéticos, que son selección, cruzamiento y mutación. A continuación describiremos a cada uno de ellos:

- **Selección:** La selección de los cromosoma padre que se usarán en el cruzamiento, en este trabajo, se realiza con un torneo binario. El cual consiste en seleccionar dos individuos al azar de la población y de estos, elegir el que tenga mejor aptitud.
- **Cruzamiento:** Para este trabajo se utilizó el cruzamiento en un punto, el cual consiste en seleccionar un punto de cruce en los cromosomas padre, este se elige al azar y es el mismo para ambos. Como se muestra en la Fig. 18 los padres se combinan en este punto para formar a los cromosoma hijo. Podemos ver que para obtener el “Hijo 1” se asigna primero los genes del “Padre 1” que corresponden a antes del punto de cruce, para los demás genes después del punto de cruce, se asignan los correspondientes del “Padre 2”. Para obtener el “Hijo 2” se realiza al contrario, primero se asignan los genes del “Padre 2” y después los correspondientes del “Padre 1”.

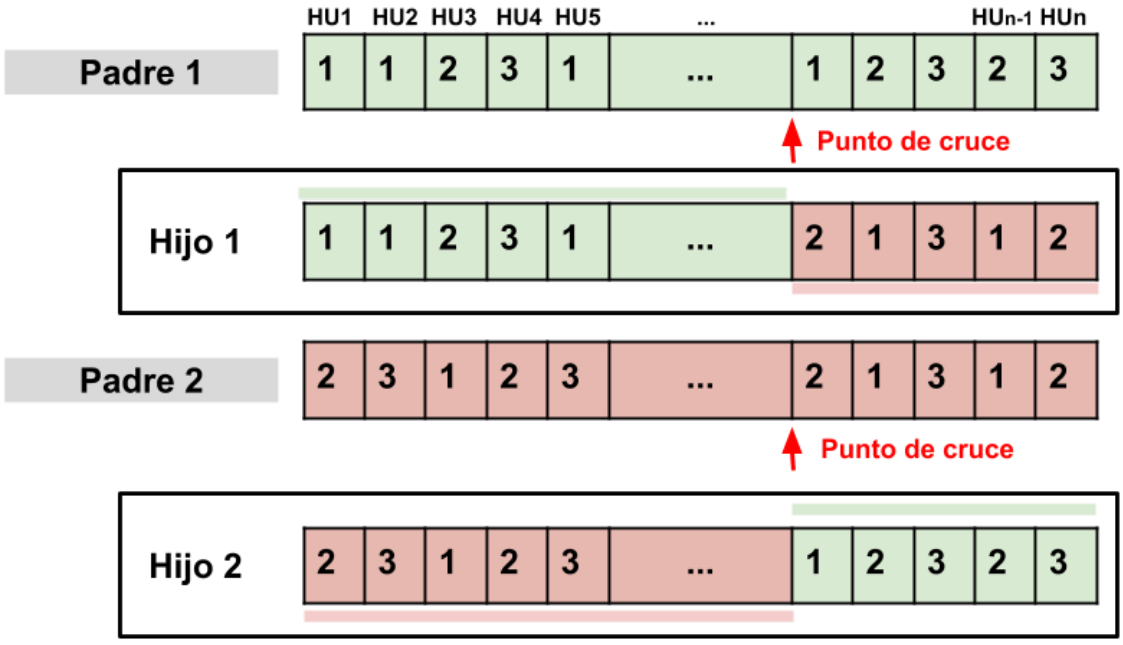



Fig. 18 Cruzamiento de dos cromosomas.

- **Mutación:** La mutación se realiza para intentar agregar material genético a la población, es decir se intentan agregar valores que no se ha podido obtener con la cruza de cromosomas. En este

trabajo se implementó una mutación multigen. Para implementar este operador cada gen del cromosoma tiene una probabilidad de ser mutado. Donde se recorre todo el cromosoma y en cada gen se obtiene un valor al azar, si este valor se encuentra dentro de la probabilidad definida, entonces se le asigna un nuevo valor al alelo. Este valor es correspondiente a un sprint, distinto al que estaba asignado, el cual también obtenemos al azar. En la Fig. 19 encontramos un ejemplo de la mutación de nuestro AG. En la parte superior encontramos al cromosoma antes de mutar y debajo de cada gen encontramos en rojo los resultados mayores a 0.2 y en azul los que fueron menores o iguales y por lo tanto serán mutados, es decir, en este ejemplo asignamos 0.2 como probabilidad de mutación. En la parte inferior se muestra el cromosoma después de la mutación.

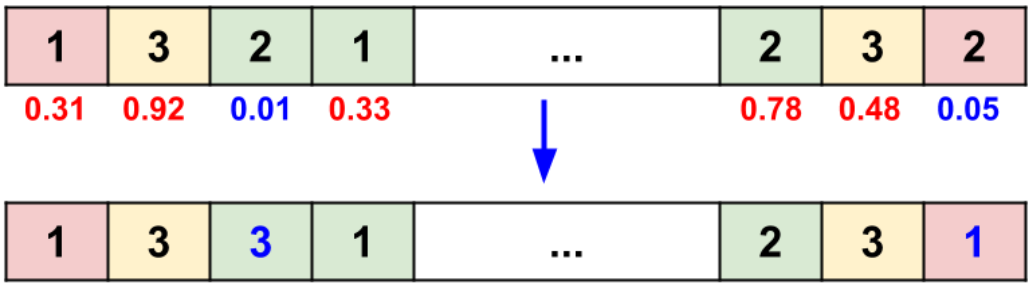



Fig. 19. Mutación gen a gen.

## 5.3 Reparación de cromosoma

En el algoritmo presentado en este trabajo, durante la ejecución todos los individuos son factibles, es decir que cada cromosoma representa una replaneación que es válida. Podemos ver un ejemplo de esto en la Fig. 20, vemos que una planeación o replaneación de liberación está compuesta por HU asignadas a un sprint, entre las HU se muestra con una línea la dependencia que hay entre ellas y su prioridad se distingue dependiendo el color (verde para alta prioridad, rojo para baja y amarilla para prioridad media). Por último se muestra las características de Scrum, las cuales son la velocidad de sprint y los puntos de historia.

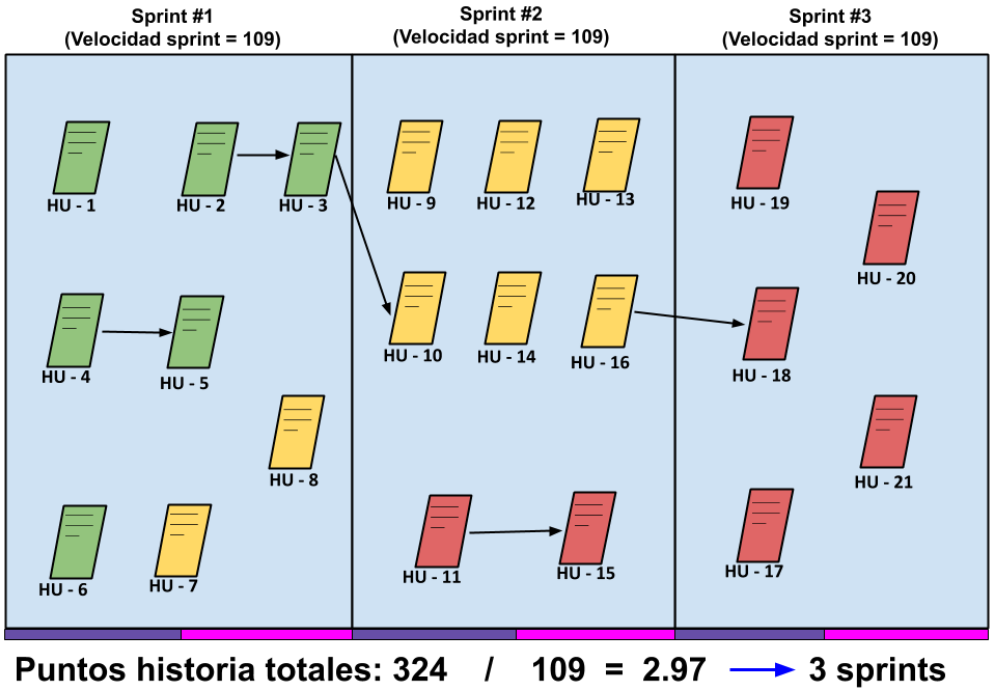



Fig. 20. Replaneación de liberación.

En ocasiones debido a la naturaleza estocástica de los AG suelen ocurrir que los cromosomas a pesar de ser factibles, no tienen sentido desde un punto de vista práctico. El primer caso es cuando en la ejecución del algoritmo al aplicar los operadores genético ocurre que el individuo resultado no respeta las dependencias entre tareas. En el segundo, el cromosoma puede presentar un sprint vacío, lo cual aunque no rompe ninguna regla en la replaneación, en proyectos reales es algo indeseable.

Por lo anterior, se implementó una operación llamada *reparacion de cromosoma* para cada uno de los casos anteriores. En una reparación de cromosoma se busca validar y reparar los genes del cromosoma de tal manera que este sea factible. En este trabajo se nombraron: reparación de dependencias y reparación de sprint vacío, las cuales se describen a continuación.

### 5.3.1 Reparación de dependencias

Como se mencionó en la sección anterior, en la fase de ejecución del algoritmo, al aplicar los operadores genéticos de cruzamiento y mutación, y debido a su naturaleza estocástica, el cromosoma puede no respetar las dependencias de precedencia entre HU. Esto significa que al aplicar alguno de los operadores genéticos, una HU que es necesaria para desarrollar otra, queda asignada a un sprint posterior. Por lo que la replaneación no sería factible ya que no se puede desarrollar una HU sin antes haber terminado las HU de las que depende. Si en un sprint son asignadas dos HU, una que depende de la otra, esta planeación es válida sólo si la velocidad de sprint es mayor o igual a la suma del esfuerzo

necesario para terminar toda la cadena de dependencias entre las HU y sus antecesores en el mismo sprint.

En este trabajo las dependencias son modeladas con un grafo de precedencia, si observamos la Fig. 23 de la sección 4.5 encontramos que se muestra una relación de HistoriaUsuario con sí misma. Esta relación es una relación muchos a muchos por lo que una HU puede depender de más de una HU y además puede ser parte de la dependencia de otra. Con esta información se construye un grafo con las HU de las que depende otra HU y así poder usarlo para validarse.

Si se presenta este problema al violar alguna de las dependencias entre HU, el operador de reparación realiza lo siguiente:

1. Se selecciona una HU de la cadena de dependencias que se vio afectada.
2. Del grafo de precedencia se obtienen todas las HU que dependan de ella y se agregan a una cola.
3. Se selecciona la primer HU de la cola y se asigna al sprint más cercano que cuente con una velocidad de sprint suficiente.
4. Si existen más HU en la cola, se selecciona la siguiente y se repite el proceso. Si la cola está vacía, la reparación terminó.

En la Fig. 21, se muestra el ejemplo de la reparación del cromosoma para una cadena de dependencia entre tres HU. En el cromosoma de la parte superior en la imagen, encontramos en rojo los cromosomas que pertenecen a la cadena de dependencias afectada. En el recuadro verde encontramos la cadena de dependencias, como se puede apreciar, HUn depende de HUn-1, que se encuentra en el sprint 1, que a su vez depende de HU3 que se encuentra en el Sprint 2. En la parte inferior a la flecha negra, para aplicar la reparación, vemos en los recuadros naranjas  la cola formada por las HU que se vieron afectadas. Encontramos  en rojo a cada uno de los valores que se modifican en cada paso de la reparación, por último, en azul encontramos los valores de los genes que fueron modificados y que ahora si respetan las dependencias.

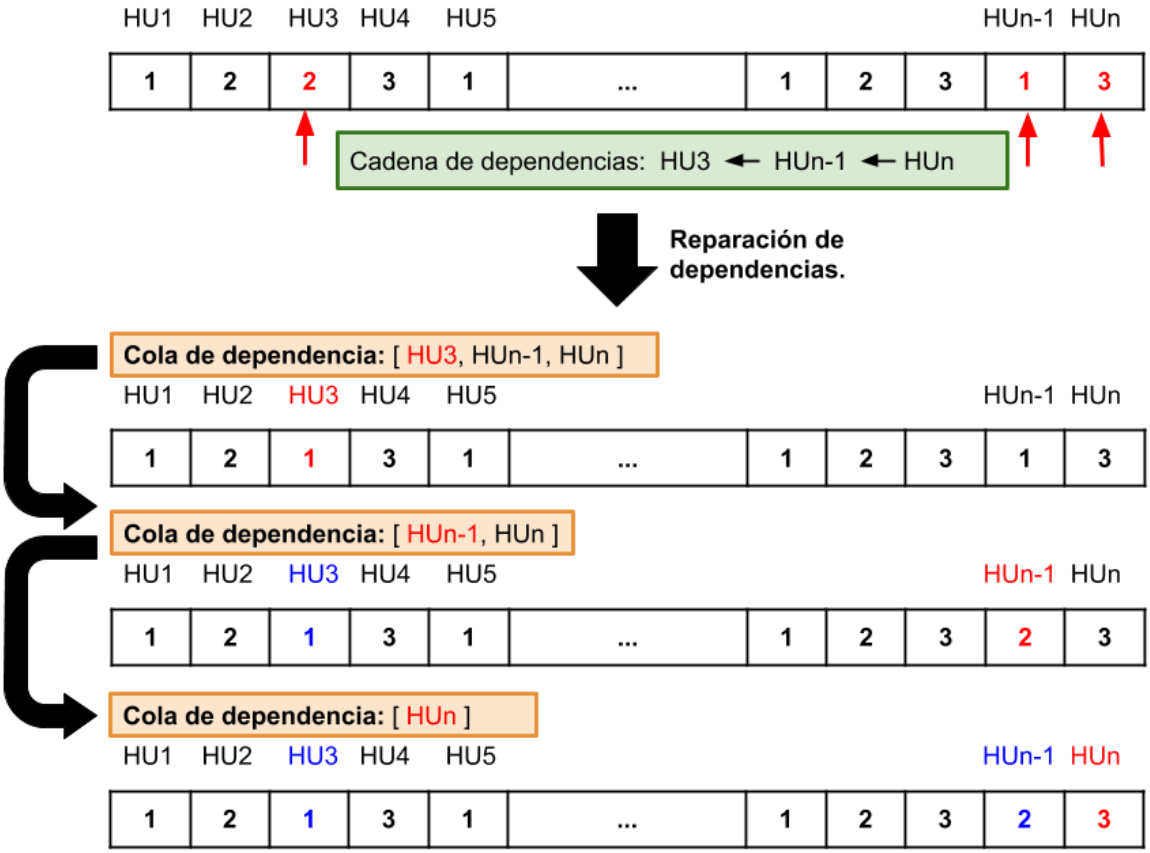



Fig. 21. Reparación de dependencias.

A continuación se muestra un ejemplo de cómo se realiza en una replaneación el procedimiento descrito en la figura anterior. En la Fig. 22 se muestra en el número 1 la planeación que está violando la dependencia entre la HU3 y la HU12. Por el sentido de la flecha muestra que la HU12 depende de la HU3, en este caso se calendarizo primero la HU12 y después la HU3 lo cual no es permitido. Al identificar este error en las dependencias se forma una cola de dependencias, la cual encontramos en color verde debajo de cada ejemplo. En el número 2 se muestra la reasignación de la HU3 la cual debe calendarizar primero que las otros dos. Recordemos que al realizar esta asignación se valida que el sprint tenga la velocidad de sprint suficiente para la HU, que en este caso es la HU3. En el número tres se muestra la recalendarización de la HU12 donde podemos ver que ahora si, la flecha muestra una dirección correcta y no en “sentido contrario” como en el número 1. Vemos que en la lista solo falta la HU13, en el número 4 se realiza su reasignación, de igual manera respetando la dependencia que tiene de la HU12. Este último ejemplo, el número 4, es el resultado de la reparación de dependencias.

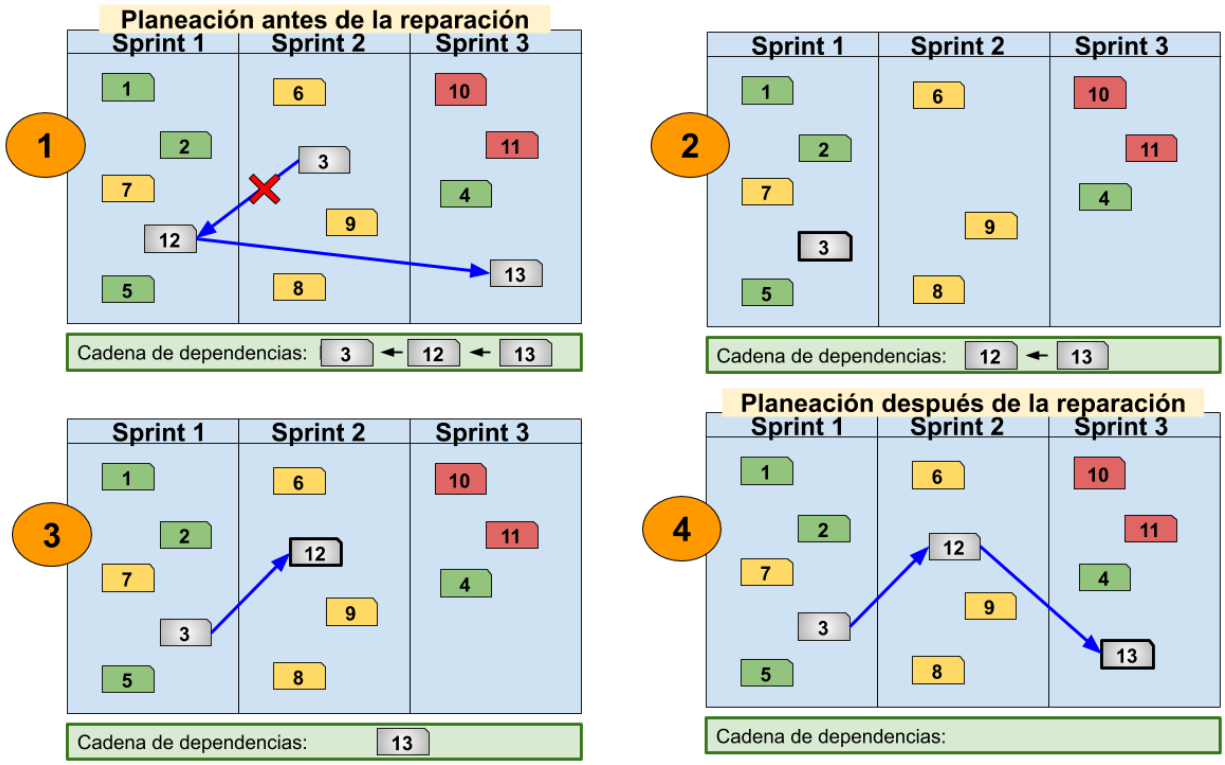



Fig. 22 Ejemplo de la aplicación del reparador de dependencias.

### 5.3.2 Reparación de sprint vacío

En el caso de la reparación de sprint vacío, se tuvo la necesidad de realizar su implementación ya que realizando algunas pruebas preliminares, se encontró que en ciertos casos el cromosoma no presentaba asignaciones de HU en alguno de los sprints. Aunque este sprint es válido ya que no viola ninguna regla, en la práctica no es correcto ya que es indeseable que una liberación presente un sprint vacío. Se decidió implementar este reparador ya que si las soluciones con un sprint vacío fueran descartadas, podríamos eliminar replaneaciones con un buen conjunto de valores, por lo que se decidió repararlas. A continuación en la Fig. 23 se muestra una propuesta de replaneación donde el sprint 3 está vacío, pues no se encuentran HU asignadas a él, esta imagen es el resultado de la ejecución de la herramienta desarrollada, antes de aplicar el operador de reparación de sprint vacío.

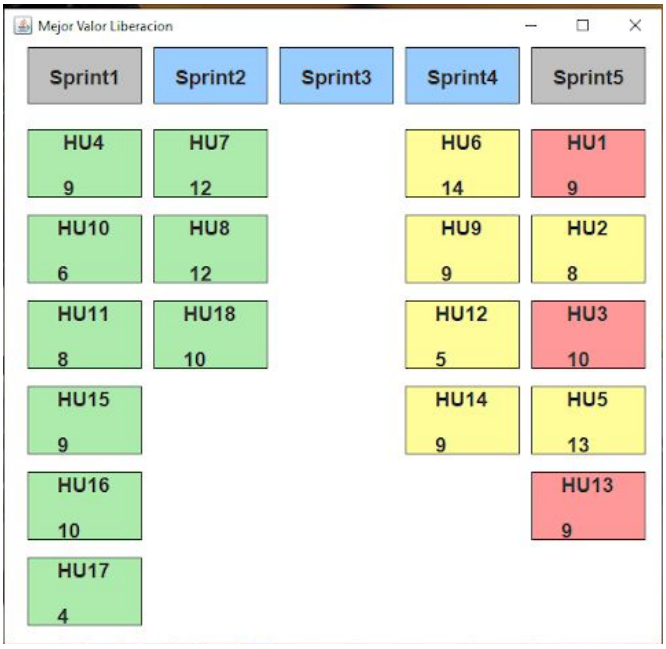



Fig. 23. Propuesta de replaneación con sprint vacío.

La reparación de sprint vacío se realiza de la siguiente manera:

1. Se identifica el sprint vacío en la replaneación.
2. Se reasignan las HU del sprint siguiente al sprint vacío.
3. Si hay más sprints vacíos se siguen recorriendo las HU para no dejar espacios. Es decir se repite el paso número 1 y 2.
4. Si ya no hay sprints vacíos, se elimina el o los sprint vacíos que se recorrieron al final de la replaneación como resultado de la reparación.

En la Fig. 24 se muestra un ejemplo del operador de sprint vacío. Donde en la replaneación superior se identifica el sprint vacío y en la replaneación inferior se realiza la reasignación de HU entre sprints.

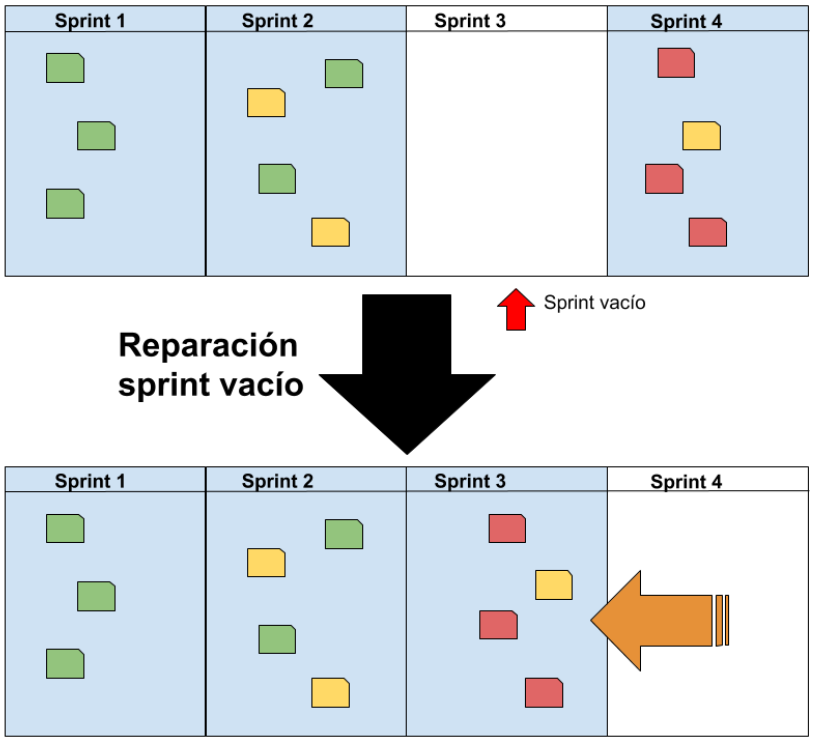



Fig. 24. Reparacion de sprint vacío.

En la sección 4.5 se describió el modelo de dominio del RLPAS y en esta sección se presentó el algoritmo genético multiobjetivo. Al realizar las pruebas preliminares los resultados mostraron que al aplicar la reparación de sprint vacío, el algoritmo mejoraba y presentaba mejores resultados.

## 5.4 El indicador de calidad de hipervolumen

Una vez que se definió el AG multiobjetivo en las secciones anteriores, se tuvo la necesidad de buscar la forma de medir que tan buenas serían las propuestas de solución, para realizar esta medición se usan los indicadores de calidad. Un indicador de calidad es un valor que representa el desempeño de un algoritmo, en este caso se usará el de hipervolumen. El indicador de hipervolumen (HV) [15] mide el espacio que cubren las soluciones no dominadas respecto a un punto de referencia y esto da cierta información de la convergencia y la diversidad de los individuos en la población. Por lo tanto una población es mejor respecto a otra si su HV es mayor. En la Fig. 25 encontramos la representación del cálculo de hipervolumen, el cual está representado con el área en azul claro y encontramos el punto de referencia en color rojo. El cual se establece con las peores soluciones que se obtuvieron para así poder medir las “distancias” que existen entre los peores y los mejores resultados.

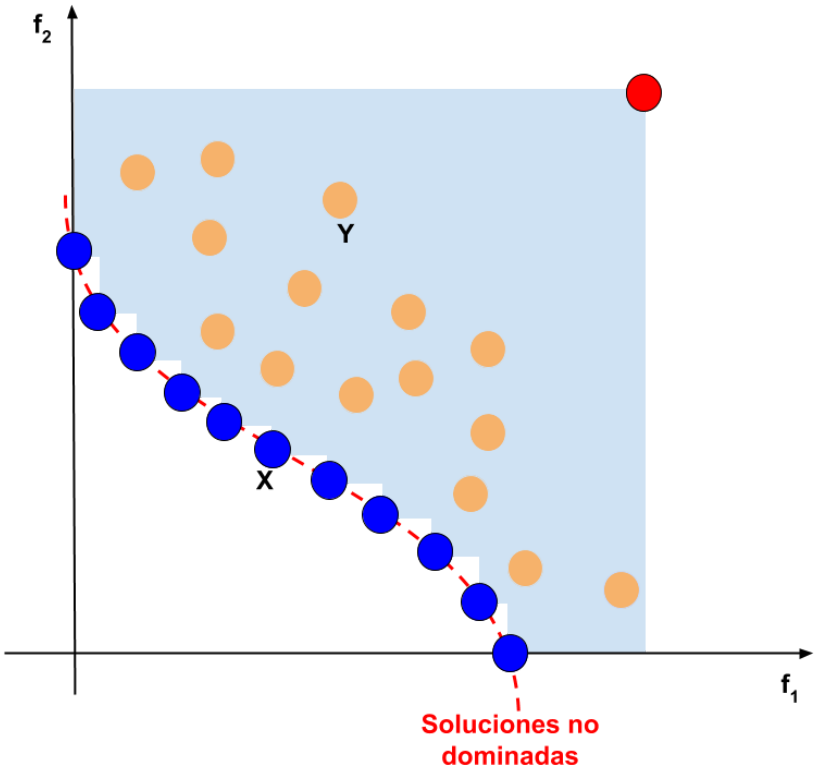



Fig. 25. Cálculo del hipervolumen.

## 5.5 Herramienta para el RLPAS

Después de haber definido las partes necesarias del modelo matemático y del algoritmo genético, se desarrolló una herramienta, con el fin de que sirva como apoyo a los Scrum Master y a los demás responsables de proyectos al realizar una replaneación. Comenzaremos con una breve explicación de las características técnicas de esta: la implementación se realizó en el lenguaje de programación Java, en ella no se usó ningún tipo de framework u herramienta, es decir, toda la codificación del modelo de dominio y de la implementación del AG fue realizada específicamente para este trabajo. Se codificó desde “cero” para así poder garantizar que todo funcionara como se buscaba y en el caso de NSGA-II para entenderlo lo mejor posible.

La herramienta da como resultado un conjunto de propuestas de replaneación, que además presenta de forma visual y que son de fácil interpretación. La Fig. 23 es un ejemplo de una replaneación de liberación resultado de la ejecución de la herramienta. Cabe señalar que esta imagen es solo un ejemplo de propuesta, ya que el conjunto completo está formado de hasta 100 propuestas que la herramienta es capaz de generar. Este es un ejemplo pequeño ya que solo está formado por 18 HU. De forma adicional, la herramienta muestra gráficas que permiten contrastar los valores de dos objetivos que resultaron de

una ejecución. Esto con el fin de poder realizar un análisis de las relaciones y conflictos entre objetivos, además de mostrar el comportamiento de los valores y así poder analizar cada uno de los diferentes casos de prueba.

# Capítulo 6

# Experimentos y Resultados

Al no encontrar trabajos que estudien el problema de replaneación de proyectos con un enfoque orientado a la planeación de liberaciones en las metodologías ágiles, no se cuenta con casos de prueba estandarizados o públicos que permitan comparar la propuesta de solución al RLPAS que se presenta en este trabajo. Por lo tanto, se tuvo la necesidad de crear casos de prueba ad hoc[3]. En las pruebas realizadas no se consideraron las dependencias para una mejor presentación de las propuestas de la replaneación, además, debido a que no aportan un valor extra, ni tienen un impacto en los resultados.

El objetivo de estos experimentos es mostrar que el modelo propuesto para el RLPAS y la herramienta que se generó con el mismo, puede servir como apoyo en la generación de propuestas de replaneación, a los responsables de desarrollar el plan de liberación del proyecto. Para lograr lo anterior, las pruebas se dividieron en dos fases, a continuación se describe el objetivo de cada una:

- **Primera fase**: El objetivo de la primera fase de pruebas es validar la pertinencia de las soluciones, es decir, mediante las gráficas obtenidas con la herramienta, se analizó si el impacto de los objetivos al realizar una replaneación era el esperado, así como la relación y el conflicto entre estos. En esta fase se busca mostrar que las propuestas de replaneación del conjunto resultado son factibles desde un punto de vista práctico, después del análisis de los valores y de las propuestas visuales de replaneación generadas con la herramienta. Para este objetivo se usaron casos de prueba pequeños formados por 12 y 17 HU
- **Segunda fase:** En la segunda fase, el objetivo fue validar la eficiencia de la herramienta, para poder realizarlo esta fase a su vez se compone de dos partes. En una primera parte se midió el impacto de la aplicación de la reparación de sprint vacío, con el fin de verificar que al aplicarlo el conjunto solución es mejor que cuando no se aplica. Esto se realiza midiendo la calidad del conjunto solución con el hipervolumen. Para la segunda parte se mide el tiempo de ejecución del algoritmo y la cantidad de soluciones distintas en el conjunto resultado, con el fin de determinar el desempeño de la herramienta. Estas pruebas se realizan con instancias grandes, formadas por hasta 100 HU y 16 sprints.

[3] Que está hecho especialmente para un fin determinado o pensado para una situación concreta.

En la tabla 4 se muestra la configuración para el AG con la cual se realizaron los experimentos. Estos valores se obtuvieron como resultado de las pruebas preliminares, ya que se identificaron que son con los que mejores soluciones se encontraron, es decir con lo que el algoritmo presenta un mejor desempeño.

Tabla 4. Valores de configuración para realizar los experimentos.

| Parámetro | Valor |
|---|---|
| Tamaño de la población | 100 |
| Probabilidad de cruza | 0.9 |
| Probabilidad de mutación | 0.2 |

Los experimentos se realizaron en una PC con procesador Ryzen 1700X a 3.6 GHz y 16GB de memoria ram a 3000 MHz. Cada una de las fases de pruebas se describe a continuación:

## 6.1 Análisis de pertinencia de la solución

Para esta primer fase, los casos de prueba fueron diseñados para analizar la pertinencia de las propuestas de replaneación obtenidas al ejecutar la herramienta. Por pertinencia nos referimos a que las soluciones a la replaneación tengan sentido desde un un punto de vista práctico, es decir se busca validar cómo afecta cada uno de los objetivos al realizar la replaneación. Esta fase se realiza, analizando las gráficas que muestran los valores en los objetivos del problema y corroborando si los objetivos presentan los resultados esperados. Además, las propuestas visuales de replaneación, permitieron corroborar de forma visual, que las soluciones que están enfocadas a un objetivo, realmente presenten resultados de acuerdo a ese objetivo. Para esta fase se crearon dos casos de prueba pequeños.

El primero, 12 HU y 4 empleados, creado para facilitar el análisis de las soluciones. Esta prueba está formada por 4 HU de prioridad alta, 4 de prioridad media y 4 de prioridad baja; además, cada sprint tiene la velocidad suficiente para que se puedan asignar exactamente 4 HU, por lo que el resultado que se espera es ver ordenadas las HU en cada una de los sprints, según su prioridad.

Los resultados de esta prueba se muestra en la Fig. 25, en esta imagen encontramos en la parte superior la planeación original y en la parte inferior 3 propuestas de replaneación, en todas ellas la velocidad regular de sprint es de 40, con la posibilidad de usar 9 puntos de historia de tiempo extra. Estas propuestas A, B y C son resultado de la ejecución de la herramienta después de que llega una HU que es urgente (HU12). A pesar de que son 5 objetivos, el objetivo de costo está ligado al número de sprints (tiempo) y al tiempo extra (desaprovechamiento) que se usa en cada uno de ellos. Por lo que no

es necesario mostrar una planeación específica de este. El objetivo de tiempo es simplemente el número de sprints en la replaneación por lo que este objetivo encuentra su mejor valor (el mínimo de sprints) en muchas de las propuestas del conjunto solución. A continuación se describen los resultados.

Se eligieron 3 propuestas, ya que son las más representativas para los 5 objetivos y en las cuales visualmente se aprecia la afectación de cada uno de ellos en la replaneación. Observamos que en la planeación original y en la propuesta A, hay títulos de sprint en un recuadro gris, esto quiere decir que en estos sprint se necesita usar tiempo extra. En la solución A observamos el mejor valor de estabilidad ya que al realizar la replaneación no presenta ninguna reasignación de sprint, excepto por la HU que llegó, es igual a la planeación original. La siguiente solución, la que está marcada con la letra B, observamos la replaneación con el mejor valor de liberación, en ella podemos ver cómo a las HU con mayor prioridad se asignan al primer sprint, las de prioridad media en el segundo sprint y en el tercero las de prioridad baja. Este “acomodo” en las HU por prioridad es el resultado esperado, debido a la definición del objetivo de valor de liberación, que si recordamos penaliza que una HU de mayor prioridad esté al frente de una de menor. Para la solución con la letra C, se muestra la replaneación con una combinación de valores cercanos a los máximos globales en los objetivos de estabilidad y desaprovechamiento. Ya que en el caso de la estabilidad, numéricamente esta muy cerca del mejor valor encontrado, ya que presenta pocas reasignaciones. Para el objetivo de desaprovechamiento presenta el mejor valor encontrado.

En la solución marcada con la letra C, a diferencia de la solución de la letra A, esta solución si presenta reasignación, pero son mínimos los movimientos, por lo que es una buena propuesta en el objetivo de estabilidad.

Podemos ver que las soluciones marcadas con la letra B y con la letra C, presentan el mejor valor encontrado para el objetivo de desaprovechamiento, ya que presentan el mismo número de puntos de historia totales en cada uno de los sprints, el cual representa la mejor asignación, ya que se aprovechan de la mejor manera los puntos de historia disponibles en cada sprint. En los tres ejemplos el objetivo de tiempo es el mejor, ya que en las tres es el mismo, presentan 3 sprints. El objetivo de costo para las soluciones B y C presenta el mejor valor ya que no ocupan tiempo extra y tienen el menor número de sprints. Sin embargo, en la solución con la letra A el objetivo de costo aumenta ya que hace uso de tiempo extra y se ve reflejado en este objetivo .

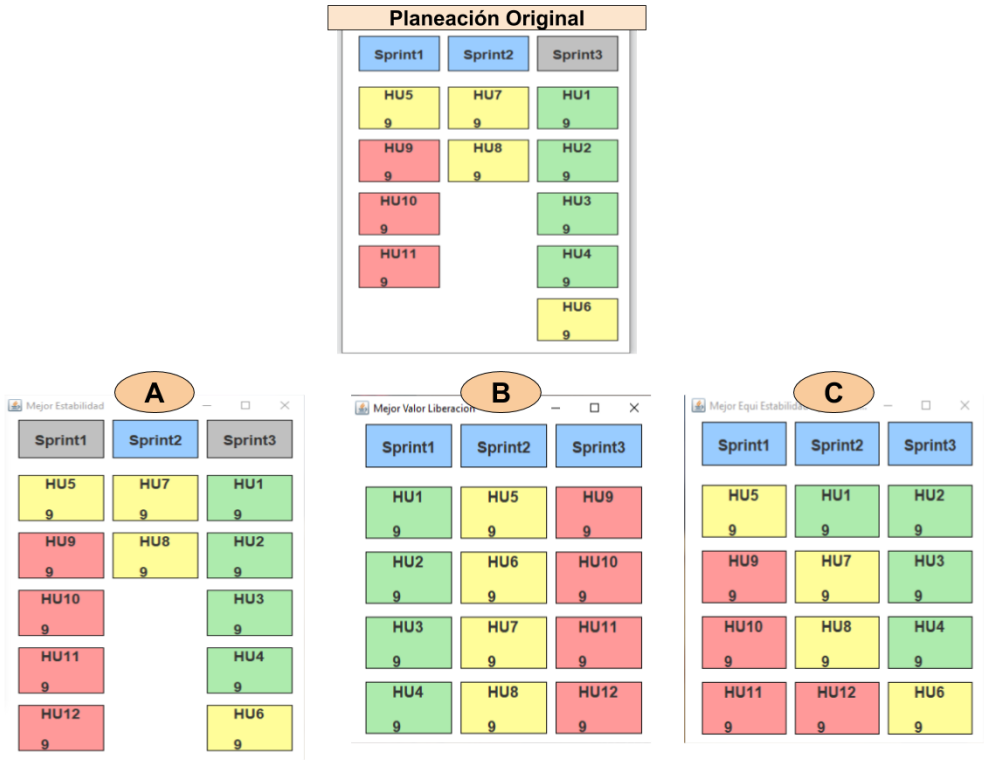



Fig. 25. Tres propuestas con distintos objetivos.

Para el segundo caso de prueba, se construyó una planeación más elaborada. Este caso está formado de 17 HU y 4 empleados, las HU ahora se muestran más dispersas en cuanto a su prioridad y no están proporcionalmente distribuidas. Como antes se mencionó nuestra herramienta da como resultado un conjunto de propuestas de replaneación, las cuales presentan un equilibrio de valores cercanos al óptimo en sus objetivos o presentan el mejor valor encontrado para cada uno de ellos. El mejor valor encontrado para cada objetivo, lo muestra la herramienta, además, es fácilmente identificable en las graficas que esta misma provee.  Entenderemos como equilibrio cuando los valores que presenta la replaneación, son cercanos al mejor valor global en más de un objetivo, ya que al tener cinco objetivos, las soluciones presentan combinaciones para los distintos valores.  Ejemplo de ello se ilustra en la Fig. 26, después de simular un evento disruptivo en el cual se agrega la HU 18. Al ejecutar la herramienta obtenemos el siguiente resultado, podemos ubicar a la izquierda, en el cuadro azul a la planeación inicial. Y dentro del cuadro naranja encontramos cuatro propuestas de replaneación después del evento disruptivo. Como en la figura del caso de prueba anterior, si un sprint necesita de tiempo extra, el título del mismo se muestra en color gris. En la parte superior del cuadro naranja, vemos dos ejemplos de propuestas que presentan un equilibrio en sus valores.En este ejemplo en particular se presenta un equilibrio en los objetivos de valor de liberación y desaprovechamiento para el A, y desaprovechamiento y estabilidad en el B. Mientras que en la parte inferior encontramos propuestas con el valor máximo para uno de los objetivos,

el de valor de liberación en el C y el D para el objetivo de estabilidad. En la solución C como se mencionó, se muestra la replaneación con el mejor valor de liberación, se puede notar que las HU se agrupan según su prioridad y se nota una tendencia de las HU de alta prioridad a la izquierda y las de baja hacia la derecha. En la solución D vemos el mejor valor de estabilidad, es decir el que menos cambios presenta respecto a la planeación original, notamos que la HU 18 se asigna al sprint 4 y es el único ajuste respecto a la planeación inicial. En la solución A observamos una combinación del mejor valor de desaprovechamiento y un buen valor de liberación, se observa claramente la división por prioridad de las HU y que estas, se asignan ocupando de la mejor manera la velocidad regular de cada sprint para evitar desaprovechar tiempo de desarrollo. En la solución B se presenta una replaneación con pocos ajustes pero que busca ocupar el mínimo de tiempo extra, es decir presenta un buen valor para desaprovechamiento y estabilidad, pero no tan bueno como A en el objetivo de valor de liberación. En cambio B si es mejor que A en el valor de estabilidad.

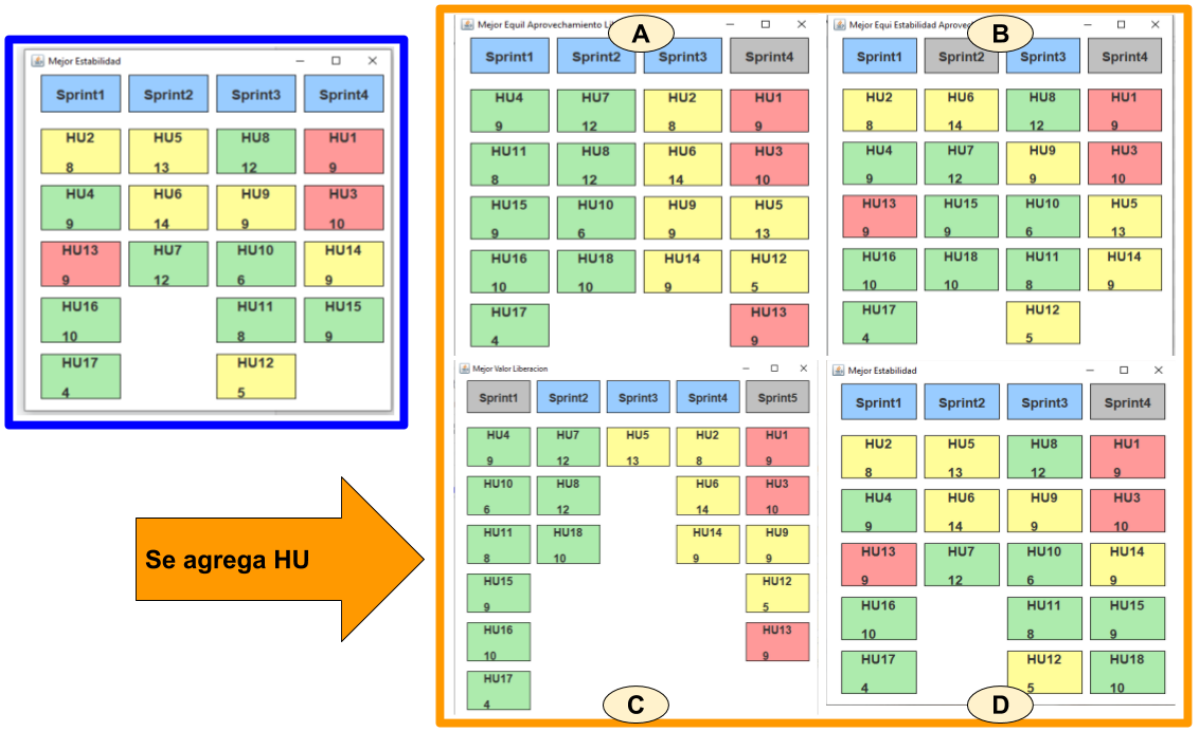



Fig. 26. Resultado de la ejecución de la herramienta al simular que llega una nueva historia de usuario (HU18).

Los anteriores casos de prueba sirvieron también para validar la implementación del reparador de sprint vacío, en la Fig. 27 se muestra en el lado izquierdo la propuesta de solución al ejecutar la herramienta sin el reparador y a la derecha cuando si se aplica. Cabe mencionar que al aplicar la replaneación la solución puede mejorar y este ejemplo lo demuestra, ya que en la replaneación sin reparador son tres los sprint que necesitan tiempo extra (el cuadro de título es de color gris). Al aplicar la reparación esto disminuye y ahora solo dos sprint necesitan de tiempo extra, es decir, se aprovecha mejor el tiempo de desarrollo. Al ser estos dos casos el mejor valor para el objetivo de liberación con o sin reparación, toma

esta forma debido a que por la función que se usa en el cálculo del objetivo obliga a que las HU de mayor y menor prioridad se ubiquen en los extremos.

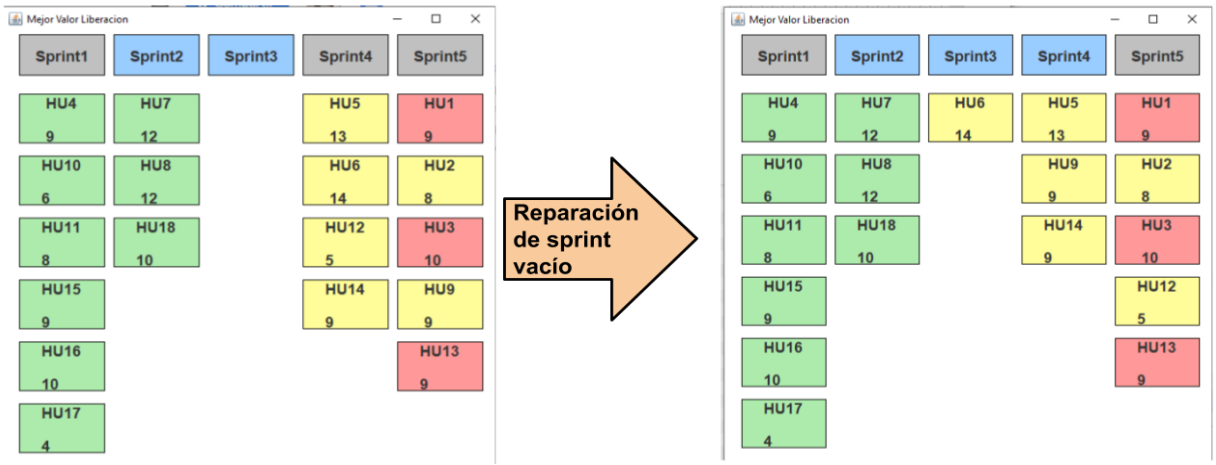



Fig. 27. Replaneación de liberación después de implementar la reparación de sprint vacío.

De los experimentos anteriores además de los resultados esperados al realizar las pruebas de replaneación, observamos que los objetivos de costo y tiempo están relacionados con el de desaprovechamiento del equipo de desarrollo, ya que cuando el desaprovechamiento aumenta, el tiempo también lo hace. Al desperdiciar mucho tiempo de desarrollo, la liberación tardará más en terminarse. En el caso del costo es lo mismo, ya que al tardar más en realizar la liberación, se deberá gastar en más sprints para terminar el proyecto. Como se mencionó en la sección 5.5, la herramienta genera gráficas para su posterior análisis, las cuales sirvieron también para validar esta fase. A continuación en la Fig. 28 se ilustra un ejemplo de ello. En estas gráficas se muestran los valores para dos de los objetivos de cada uno de los individuos (recordemos que cada individuo presenta cinco objetivos). En el lado izquierdo de la imagen, encontramos la gráfica A con los objetivos de desaprovechamiento en X y para el eje Y el costo. Para el lado derecho encontramos la B que en el eje X presenta el objetivo de desaprovechamiento y en el eje Y el de tiempo. Los datos de la gráfica A demuestran que el desaprovechamiento impacta directamente al costo, ya que si el desaprovechamiento crece (valores más a la derecha de la gráfica), el costo también se incremente ya que al desaprovechar demasiado, necesitaremos más tiempo para terminar la liberación y esto se traduce en pagar más tiempo al equipo de desarrollo. De la gráfica B podemos deducir que el desaprovechamiento, si impacta en el tiempo, pero solo cuando este crece mucho. Podemos ver como el valor en el tiempo se mantiene aunque el desaprovechamiento crezca, y después da un salto que corresponde a la introducción de un nuevo sprint. Esto es fácil de explicar, ya que si el tiempo se desaprovecha aún se puede mantener la cantidad de sprint, si se trabaja tiempo extra. Como el tiempo extra tiene un límite, si el desaprovechamiento crece

demasiado, irremediablemente se deberá retrasar el proyecto al aumentar el número de sprints en la liberación. Cabe señalar que el análisis de estas gráficas, se realizó para todos los pares de objetivos, aquí se presentan sólo dos pares, ya que más adelante en este capítulo se mostrarán más resultados, en los cuales este tipo de gráficas servirán como apoyo.

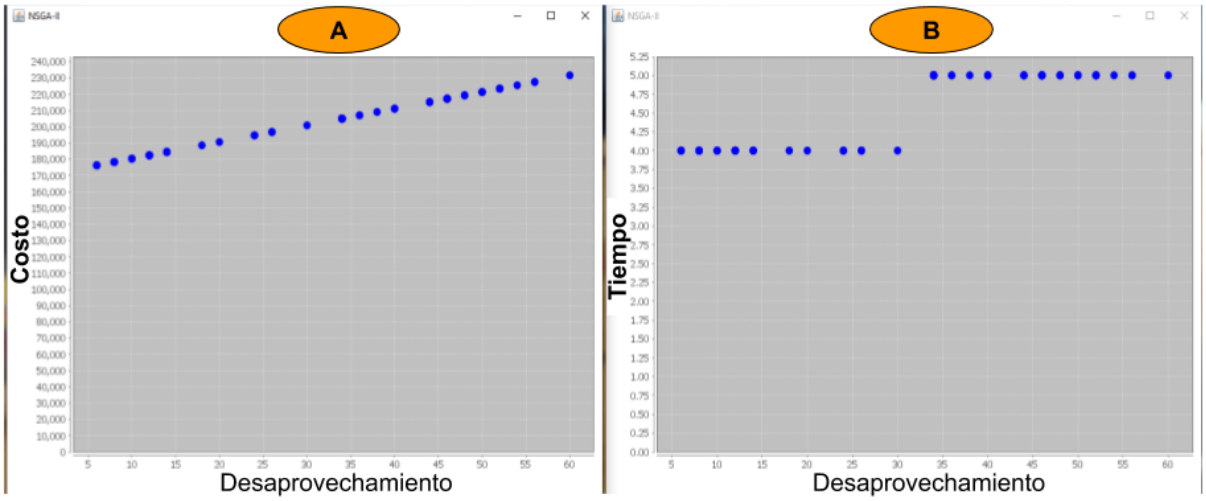



Fig. 28. Objetivos proporcionales al desaprovechamiento.

## 6.2 Análisis de la eficiencia del algoritmo propuesto

La segunda fase de pruebas tiene como objetivo evaluar la eficiencia de la solución que en este trabajo se propone. Entenderemos que medir la eficiencia se refiere a medir los tiempos de ejecución del algoritmo en casos de prueba más complejos y que los resultados sean pertinentes para que la herramienta pueda ser usada como apoyo en la replaneación de proyectos en Scrum. Además como muestra de la capacidad de nuestra herramienta al presentar propuestas de solución, se midió la cantidad de distintas propuestas de solución en el conjunto resultado que presenta el AG.

Como se comentó al inicio de esta sección, no encontramos datos de casos de prueba públicos con los cuales poder comparar nuestro algoritmo y, por lo tanto, tampoco la herramienta desarrollada. Por lo que, al tener la necesidad de evaluar nuestro algoritmos con casos más complejos, se utilizó el generador de casos de prueba desarrollado por Alba y Chicano [17], el cual genera casos de prueba para el problema de planeación de proyectos de software. Este generador es público y se puede descargar desde el siguiente enlace: http://tracer.lcc.uma.es/problems/psp/. Al estar originalmente diseñada para SPSP se tuvo que adecuar esta herramienta, para la generación de casos de prueba para el RLPAS, pero sin muchas complicaciones se pudieron simular todas las características necesarias para realizar las pruebas. Todos los datos que se obtienen al ejecutar esta herramienta se crean

aleatoriamente y son: estimación de las HU (en puntos de historia), dependencias entre HU, prioridad de las HU, capacidad de desarrollo de cada empleado (se asigna un valor según su experiencia) y salario de cada empleado por sprint.

Para realizar las pruebas de esta fase, la herramienta se configuró para que creara aleatoriamente planeaciones con 40, 70 y 100 HU y para el número de empleados se consideraron 4, 5 y 6, respectivamente. Se concluyó que estas son pruebas suficientemente grandes, ya que están compuestas por mínimo 7, 11 y 15 sprint respectivamente, y si cada sprint dura 15 días (lo cual en la práctica es muy común), cada prueba representa varios meses de trabajo. Al ejecutar el generador de casos de prueba se obtuvieron instancias, que al traducirlas a planeaciones, estuvieron compuestas de hasta 16 sprints. En cada una de las pruebas se simularon dos eventos disruptivos: partida de un empleado e introducción de una nueva HU, ya que éstas tienen un mayor impacto negativo en la planeación.

### 6.2.1 Evaluación del impacto del uso del reparador de sprint vacío

Al analizar los resultados en la primera fase de pruebas, se concluyó que el reparador de sprint vacío, además de cumplir con la función de garantizar que todas las soluciones sean factibles, genera poblaciones que presentan mejores valores en los objetivos de costo, desaprovechamiento y tiempo. Para demostrarlo se usó el hipervolumen como indicador de calidad.

Para comenzar con estas pruebas, se eligió un caso de prueba de los anteriormente descritos. Después se simula un evento disruptivo y se ejecuta 10 veces el AG con cada configuración. En cada ejecución tenemos la misma cantidad de HU, el mismo número de empleados, el mismo evento disruptivo y se aplica la reparación de sprint vacío. De las 10 repeticiones se obtiene un promedio del tiempo de ejecución del algoritmo y también de los valores máximos (los peores) de todas las ejecuciones para cada uno de los objetivos, estos valores sirven para obtener los puntos para formar el vector de referencia al calcular el hipervolumen. Se realiza así el cálculo del vector de referencia para garantizar que todos los puntos de cada una de las pruebas sean tomados en cuenta al realizar la medición del hipervolumen. Al construir el vector con los valores más alejados al mínimo (los peores), el hipervolumen siempre se comienza a calcular desde el mismo conjunto de puntos de referencia y así siempre contiene a todos los de cualquier conjunto solución que se utilizó para realizar las pruebas.

El procedimiento anterior se volvió a realizar pero ahora sin la reparación de sprint. Ahora entonces tenemos 10 repeticiones con reparación de sprint vacío y 10 sin ella, y además dos conjuntos de valores máximos para cada tipo de ejecuciones. De los dos conjuntos obtenemos los máximos globales de las 20 ejecuciones. Los máximos globales como ya se mencionó antes, son los puntos de referencia con los cuales se calcula el hipervolumen de cada población resultado. Por último, se calculó el hipervolumen de cada conjunto solución correspondiente a cada ejecución del algoritmo. Al final se obtuvo el promedio del

hipervolumen de las 10 poblaciones con reparación y el promedio de las 10 sin reparación. El resumen de los resultados de los experimentos se muestran en la Tabla 5. Con esta tabla se busca mostrar el impacto de la reparación de sprint vacío, el cual es claro en la mejora del hipervolumen al aplicar la reparación. En la primera columna encontramos el número de HUs, la segunda, muestra el número de empleados y en la tercera se muestra el evento disruptivo: **EMP-** indica que se va un empleado y **HU+** representa que se agrega una nueva HU. Las siguientes cuatro columnas representan, respectivamente, el promedio en el tiempo de ejecución y el promedio del hipervolumen para el algoritmo sin reparación y con reparación. En la penúltima columna se muestra la diferencia entre los tiempos de ejecución del algoritmo cuando se aplica y cuando no se aplica el operador de reparación. Por último, en la columna final se encuentra el resultado de la diferencia entre los valores del hipervolumen al aplicar el reparador de sprint vacío y cuando este no se aplica.

Tabla 5. Datos de los casos de prueba, el evento disruptivo y resumen de los resultados obtenidos con el algoritmo genético multiobjetivo, sin y con operador de reparación.

| No. HU | No. Emp. | Evento | Sin reparación | | Con reparación | | Δ Tiempo (s) | Δ Hv |
|---|---|---|---|---|---|---|---|---|
| | | | t(s) | HV | t (s) | HV | | |
| 40 | 3 | EMP- | 111.158 | 166716.55 | 109.904 | 184277.84 | -1.254 | 17561.29 |
| 40 | 4 | HU+ | 102.09 | 146908.69 | 104.49 | 177666.49 | 2.4 | 30757.8 |
| 70 | 4 | EMP- | 235.19 | 240939.09 | 243.59 | 301195.96 | 8.4 | 60256.87 |
| 70 | 5 | HU+ | 214.69 | 217033.16 | 212.79 | 254914.15 | -1.9 | 37880.99 |
| 100 | 5 | EMP- | 317.42 | 310596.38 | 339.34 | 315241.21 | 21.92 | 4644.83 |
| 100 | 6 | HU+ | 307.22 | 369386.78 | 313.48 | 463991.84 | 6.26 | 94605.06 |

Como se esperaba los resultados muestran que la reparación de sprint vacío ayudó a que el algoritmo alcance un mayor hipervolumen. Podemos interpretar entonces que, cuando se aplica la reparación de sprint vacío, las replaneaciones de la población resultado presentan mejores valores en sus objetivos. Es decir, encuentran mejores replaneaciones, ya que, en general, la población converge a mejores resultados para los objetivos de tiempo, costo, estabilidad, desaprovechamiento y valor de liberación. Podemos ver que en la mayoría de los casos, el promedio del tiempo de ejecución aumenta al aplicarse la reparación, pero este incremento no es considerable ya que en el peor de los caso solo tardó 21 segundos más en ejecutarse. De hecho, para el primero y cuarto experimentos mostrados en la tabla, el promedio del tiempo de ejecución con la reparación es menor que cuando no se aplica. Esto se explica debido a que no siempre es necesario aplicar la reparación de sprint debido a la aleatoriedad de la generación de casos de prueba, por lo que aunque se aplique el reparador, puede que al ejecutar el AG este nunca sea utilizado. Los resultados muestran que aplicar la reparación en el algoritmo da como

resultado una población con un HV mayor, por lo tanto el algoritmo tiene un mejor desempeño y no afecta significativamente el tiempo de ejecución.

### 6.2.2 Evaluación del desempeño de la herramienta para el RLPAS

Esta fase final de pruebas, busca evaluar el desempeño de la herramienta en los experimentos anteriormente realizados, pero en este caso solo nos enfocaremos en aquellos en los que se aplicó la reparación de sprint vacío ya que se demostró que con ella, el algoritmo muestra un mejor desempeño. Por desempeño entenderemos que se analiza el tiempo de ejecución, la cantidad de propuestas de replaneación diferentes y la calidad de las replaneaciones obtenidas con la herramienta. Al final de la sección se presenta un análisis de los datos mostrados en las gráficas de comparación entre objetivos que nos provee la herramienta que se desarrolló, ya que nos ayudarán a mostrar el comportamiento del AG.

En general el proceso para realizar los experimentos fue el siguiente: Se simula un evento disruptivo y se ejecuta el algoritmo 10 veces. En cada ejecución tenemos la misma cantidad de HU, el mismo número de empleados, el mismo nivel de experiencia para cada uno de ellos y el mismo evento disruptivo. De las 10 repeticiones se obtiene un promedio del tiempo de ejecución del algoritmo y un promedio de la cantidad de diferentes replaneaciones del conjunto resultado. El resumen de los resultados de los experimentos se muestran en la Tabla 6. En la primer columna encontramos el número de HU, la segunda muestra el número de empleados y en la tercera se muestra el evento disruptivo. La cuarta columna muestra el promedio del tiempo de ejecución en segundos (t). Y la quinta columna, nos muestra el número promedio de distintas propuestas de replaneación que nos da como resultado la herramienta, en los resultados encontramos hasta 100 propuestas distintas (en promedio) ya que este es el número máximo de individuos en la población.

Tabla 6. Datos de los casos de prueba, el evento disruptivo, tiempo de ejecución del AG y número de distintas propuestas obtenidas.

| No. HU | No. Emp. | Evento | t | # promedio propuestas |
|---|---|---|---|---|
| 40 | 3 | EMP- | 109.90 | 97.9 |
| 40 | 4 | HU+ | 104.49 | 98.9 |
| 70 | 4 | EMP- | 243.59 | 100 |
| 70 | 5 | HU+ | 212.79 | 99.5 |
| 100 | 5 | EMP- | 339.34 | 100 |
| 100 | 6 | HU+ | 313.48 | 100 |

A continuación se muestran ejemplos de los resultados de ejecutar la herramienta en los experimentos descritos anteriormente. Cabe señalar que en cada ejecución el conjunto resultado está compuesto por al menos 97 distintas propuestas de replaneación, aquí se muestra solo una de ese conjunto. Estas propuestas de replaneación que se muestran como ejemplo, fueron elegidas del conjunto solución, ya que muestran valores interesantes y representativos de los objetivos.

Para comenzar, en la Fig. 29 podemos ver los resultados de la ejecución de la herramienta para una instancia de 40 HU, la cual muestra el máximo valor para el objetivo de valor de liberación. En esta propuesta de solución se encuentran claramente las HU divididas por prioridades, donde las de mayor prioridad se muestran en verde y las de prioridad más baja en rojo, las de prioridad media son amarillas. Podemos observar que al tratarse del máximo valor de liberación, las HU con prioridad alta se repliegan a la izquierda y las de prioridad baja a la derecha, al centro encontramos las de prioridad media. Como esperábamos, las HU de alta prioridad se calendarizan al inicio y las de baja al final de la replaneación. La replaneación está formada por 9 sprints, con una velocidad de 45 puntos de historia en cada uno. Encontramos en 3 sprints el recuadro de título en color gris, lo que indica que en estos se necesita tiempo extra.  El evento disruptivo simulado es que un empleado abandone el proyecto, con el cual se pierde velocidad en cada uno de los sprint.

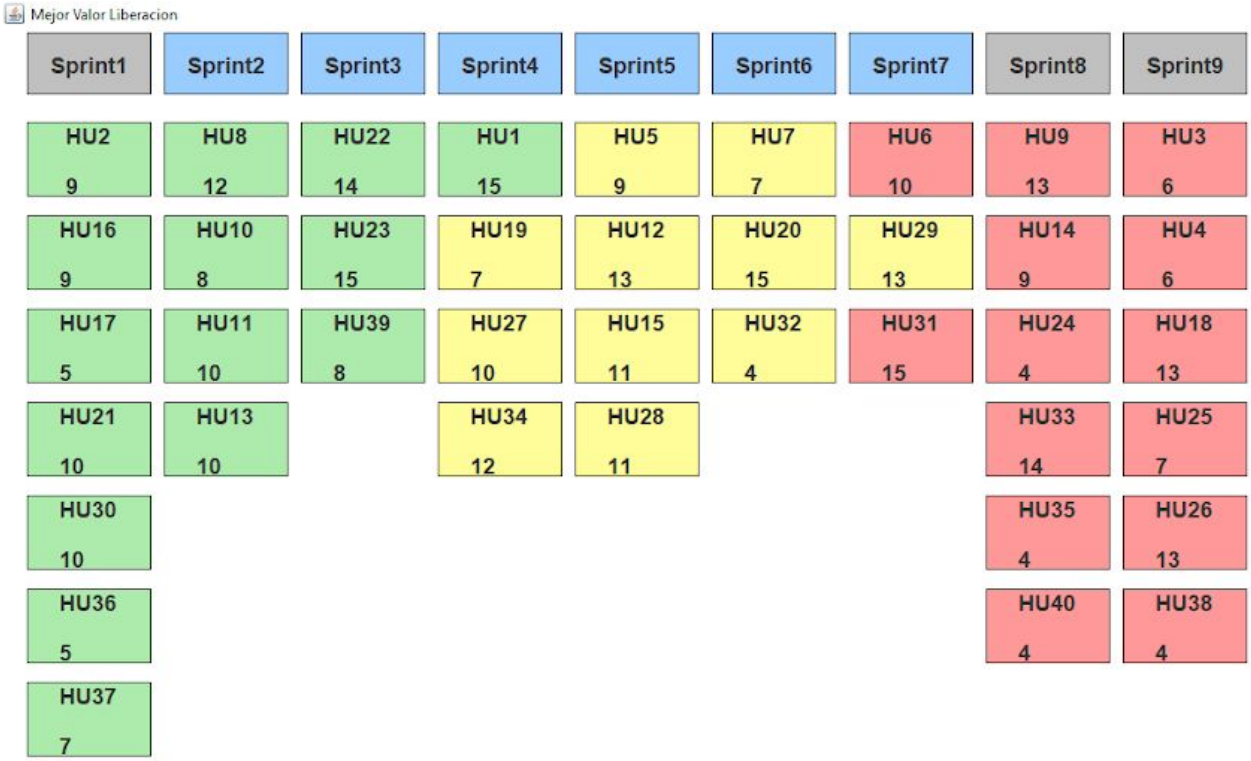



Fig. 29. Mejor valor de liberación para una planeación de 40 HU.

En la Fig. 30 se ilustra el resultado de un equilibrio en los objetivos de valor de liberación y desaprovechamiento, la replaneación también consta de 40 HU como el ejemplo anterior. Se realiza simulando el evento de que un empleado se va. En ella podemos ver cómo la cantidad de HU y de puntos de historia asignado a cada sprint es similar, debido a que presenta un valor mínimo de desaprovechamiento (el mejor valor), es decir, se aprovecha muy bien la capacidad del equipo de desarrollo. A su vez podemos observar que la mayoría de las HU se colocan según su prioridad, ya que el valor de liberación que esta replaneación presenta, está muy cerca del mejor encontrado. Esta propuesta abarca 9 sprints, donde cada uno tiene una velocidad de 45 puntos de historia y en ninguno es necesario utilizar tiempo extra.

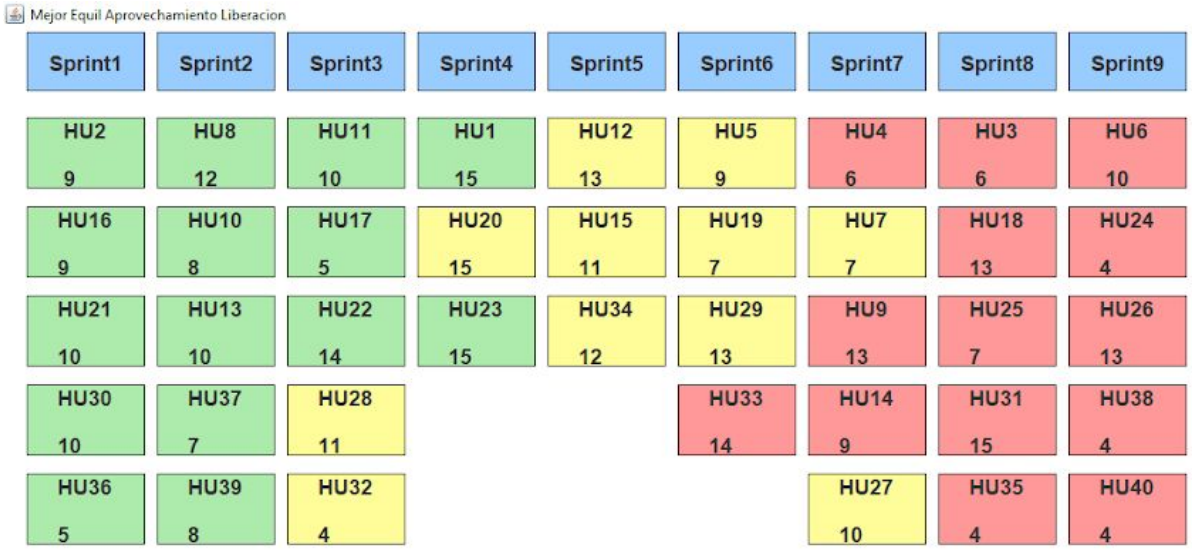



Fig 30. Replaneación con equilibrio de los objetivos compuesta de 40 HU.

En el caso de las pruebas con 70 HU y después de que un empleado abandona el proyecto, se muestra un ejemplo de replaneación en la Fig. 31. Esta consta de 13 sprints. Como en los casos anteriores vemos que la mayoría de HU se asignan según su prioridad y que la cantidad de HU así como los puntos de historia es parecida en todos los sprint. Podemos notar que hay una HU de baja prioridad calendarizada en el primer Sprint y las HU no están acomodadas exactamente por prioridad, esto es porque la replaneación presenta una combinación de valores interesantes en sus objetivos. En específico, no presenta el mejor valor para el objetivo de valor de liberación, pero sí uno muy cercano al mejor encontrado y si presenta el mejor valor encontrado para los objetivos de desaprovechamiento, tiempo y costo.

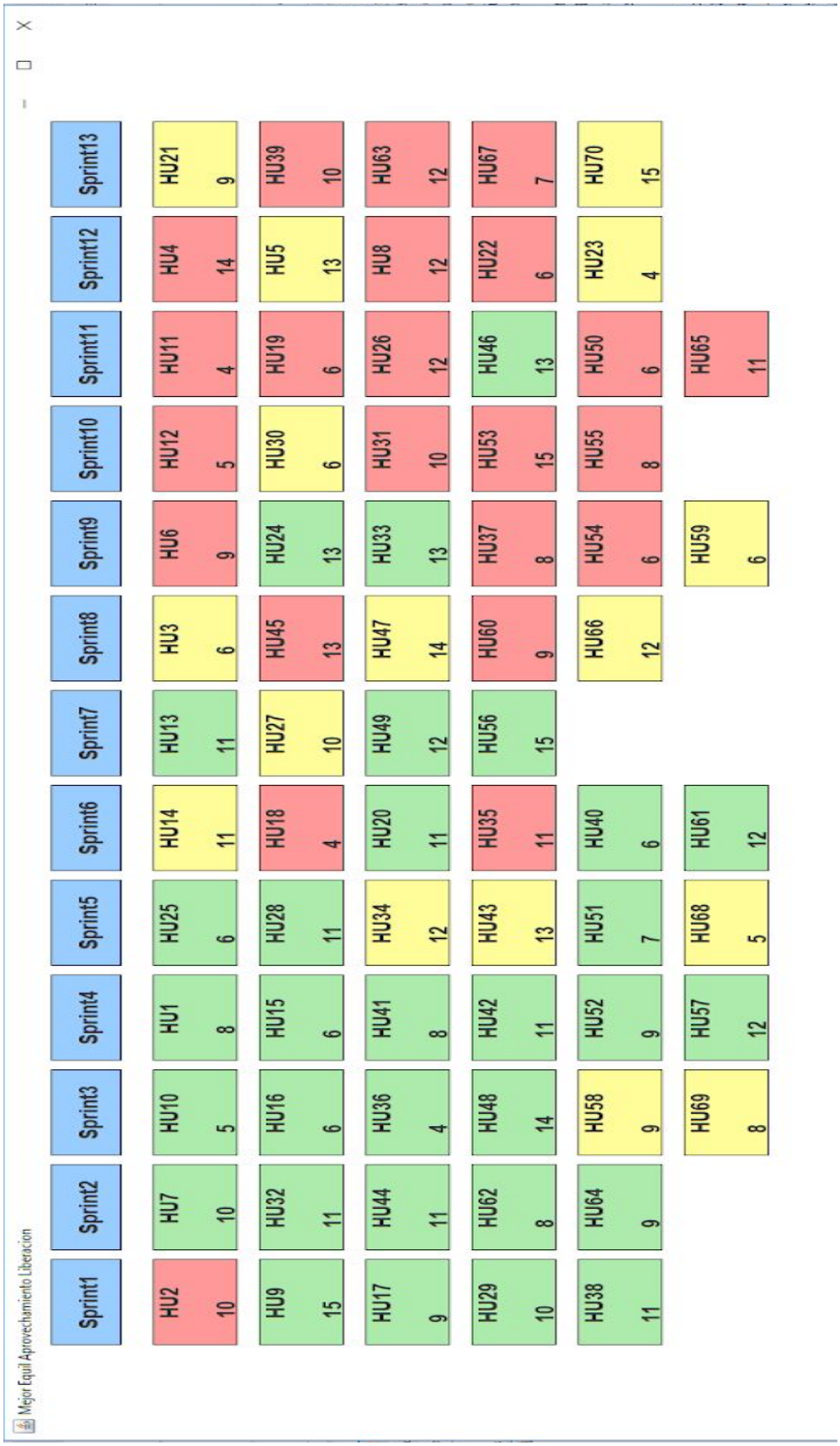
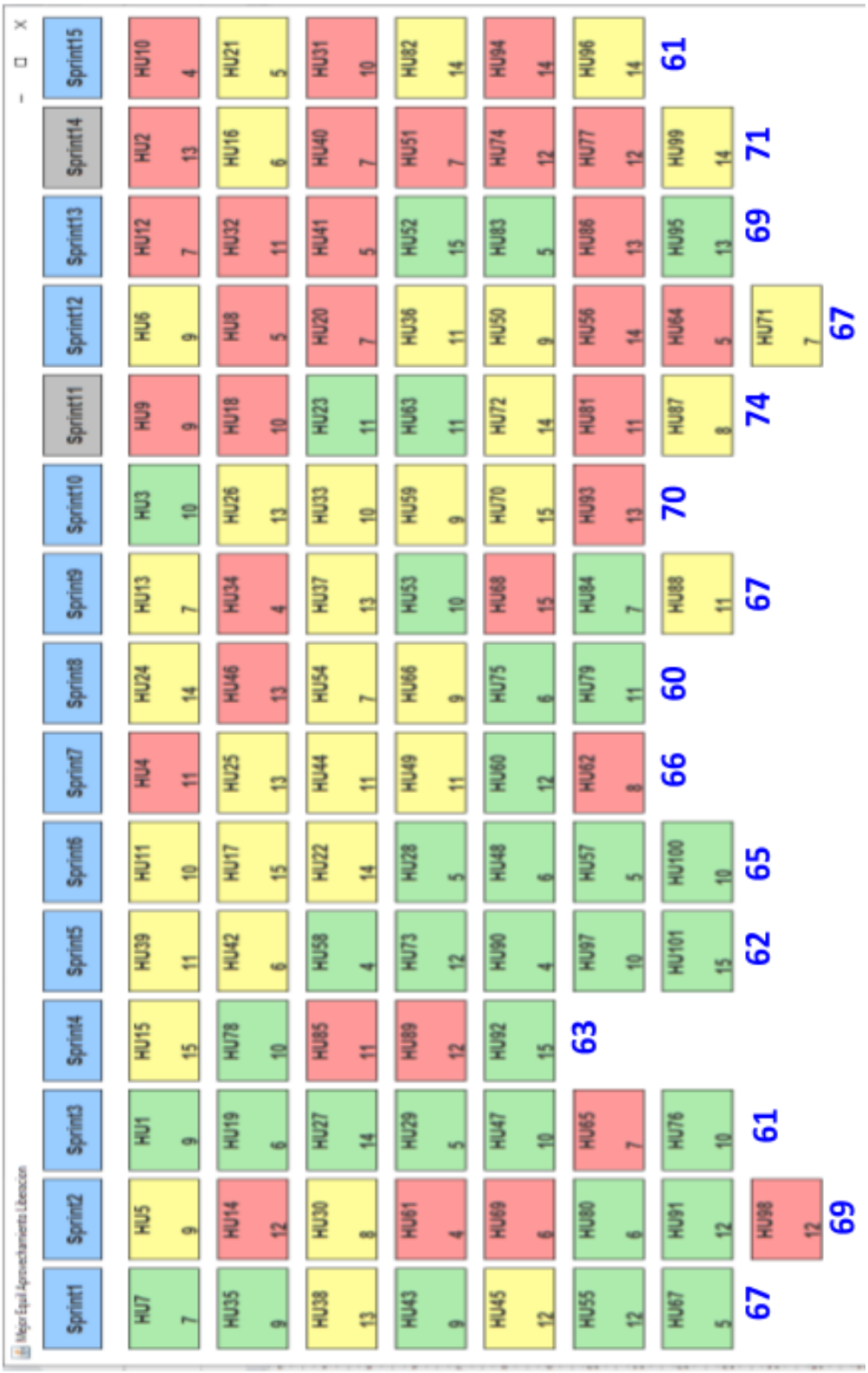



Fig. 31. Equilibrio de objetivos para una planeación de 70 HU.

Los resultados de la segunda fase de pruebas, muestran que en los experimentos, el tiempo de ejecución del algoritmo va de 1.8 minutos en los casos de prueba con 40 HU, hasta un poco menos de 6 minutos en los de 100 HU. En pocos minutos la herramienta dependiendo de las características del caso de prueba (planeación inicial) obtiene 97.9 propuestas de replaneación en promedio, para los casos formados por 40, 70 y 100 HU. En el mejor de los casos da como resultado un máximo de 100 distintas replaneaciones. La cantidad de propuestas de replaneación varía debido a la naturaleza aleatoria de los AG.

Aun en el caso que el algoritmo presente un conjunto de 100 diferentes propuestas de replaneación, este tarda como máximo 6 minutos para las instancias más grandes, en comparación con los 180 minutos en que tarda un experto realiza una sola replaneación [10]. En los experimentos, las pruebas con 100 HU presentan replaneaciones formadas de 16 sprints, si cada sprint dura 15 días (lo cual es común), la replaneación realizada equivale a un plan de 240 días de trabajo. En resumen, la herramienta desarrollada puede realizar no una, sino un conjunto de 100 propuestas de replaneación de un proyecto de 8 meses en unos pocos minutos. Además, la ventaja es que conforme se ejecute una nueva planeación (replaneación) y ocurran nuevos eventos disruptivos, se podrán hacer nuevas replaneaciones de manera rápida.

La Fig. 32 muestra un ejemplo del conjunto solución, una replaneación formada por 100 HU y en la cual se considera un equipo de trabajo de 6 empleados. Este ejemplo, muestra cómo en los anteriores un resultado interesante entre desaprovechamiento y valor de liberación. Sin perder de vista que como todas las propuestas del conjunto solución, presenta buenos resultados en todos los objetivos, es decir, en todos los objetivos el valor es cercano al máximo global encontrado. En este ejemplo se nota de nuevo la tendencia de agruparse las HU de mayor prioridad a la izquierda y después más a la derecha las de menor prioridad. Por el color gris de los cuadros de título de sprint, sabemos que solo en dos sprints se utiliza tiempo extra y si sumamos la cantidad de puntos de historia asignados a cada sprint nos damos cuenta que es similar, este valor se muestra con un número de color azul al final de cada sprint. Por lo que podemos decir que en esta replaneación se aprovecha de excelente manera al equipo de desarrollo.


Mejor Equil Aprovechamiento Liberacion
Sprint1 Sprint2 Sprint3 Sprint4 Sprint5 Sprint6 Sprint7 Sprint8 Sprint9 Sprint10 Sprint11 Sprint12 Sprint13 Sprint14 Sprint15
HU7 7 HU5 9 HU1 9 HU15 15 HU39 11 HU11 10 HU4 11 HU24 14 HU13 7 HU3 10 HU9 9 HU6 9 HU12 7 HU2 13 HU10 4
HU35 9 HU14 12 HU19 6 HU78 10 HU42 6 HU17 15 HU25 13 HU46 13 HU34 4 HU26 13 HU18 10 HU8 5 HU32 11 HU16 6 HU21 5
HU38 13 HU30 8 HU27 14 HU85 11 HU58 4 HU22 14 HU44 11 HU54 7 HU37 13 HU33 10 HU23 11 HU20 7 HU41 5 HU40 7 HU31 10
HU43 9 HU61 4 HU29 5 HU89 12 HU73 12 HU28 5 HU49 11 HU66 9 HU53 10 HU59 9 HU63 11 HU36 11 HU52 15 HU51 7 HU82 14
HU45 12 HU69 6 HU47 10 HU92 15 HU90 4 HU48 6 HU60 12 HU75 6 HU68 15 HU70 15 HU72 14 HU50 9 HU83 5 HU74 12 HU94 14
HU55 12 HU80 6 HU65 7 HU97 10 HU57 5 HU62 8 HU79 11 HU84 7 HU93 13 HU81 11 HU56 14 HU86 13 HU77 12 HU96 14
HU67 5 HU91 12 HU76 10 HU101 15 HU100 10 HU88 11 HU87 8 HU64 5 HU71 7 HU95 13 HU99 14
HU98 12
67 69 61 63 62 65 66 60 67 70 74 67 69 71 61
Fig. 32. Replaneación de 100 HU.

Para entender un poco mejor la dificultad de este problema, es importante mencionar que el tamaño del espacio de búsqueda del RLPAS en general es del orden de $m^n$ donde $m$ es igual al número de sprints y $n$ la cantidad de HU.

Por último y como parte de los resultados, a continuación se muestran algunas gráficas para poder analizar el comportamiento de los objetivos y en especial el conflicto que existe entre algunos de ellos. En la Fig. 33 se muestran tres gráficas de 3 ejecuciones distintas en experimentos distintos de instancias formadas por 70 HU y 5 empleados. En cada una de ellas encontramos los valores de dos objetivos de los individuos del conjunto solución, recordemos que el conjunto solución está formado por las soluciones no dominadas con respecto a los cinco objetivos. En esta imagen podemos ver cómo los objetivos de costo y desaprovechamiento muestran una clara tendencia lineal creciente. Lo que quiere decir que como en los casos más pequeños de la fase uno de pruebas, estos valores son proporcionales ya que si el desaprovechamiento crece (se desaprovecha al equipo de desarrollo), el proyecto será más costoso. Esto en proyectos reales es cierto ya que si el equipo no se aprovecha de manera adecuada se deberá de ocupar mucho tiempo extra y esto eleva el precio del proyecto.

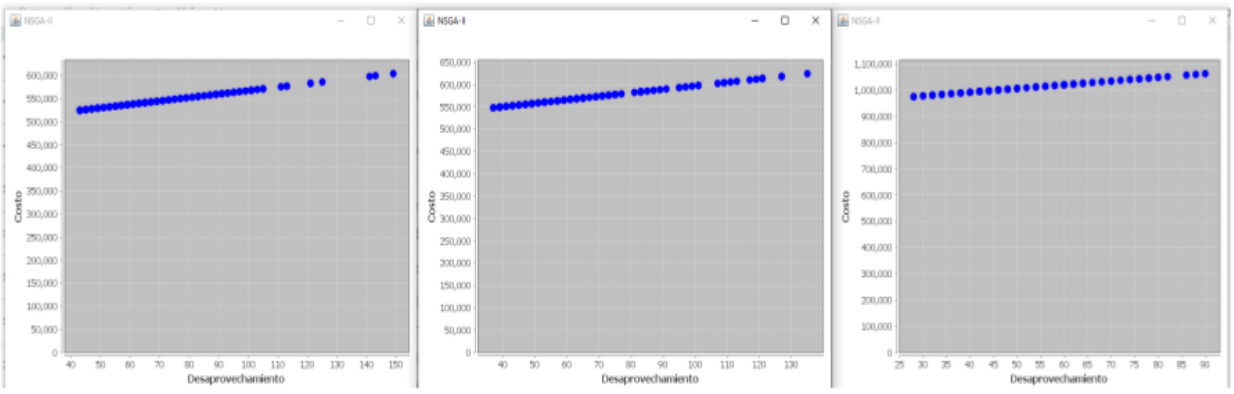



Fig. 33. Costo y desaprovechamiento valores proporcionales.

En otro caso, en la Fig. 34 también se muestran las soluciones no dominadas, no se encuentra una relación clara entre los valores de los objetivos mostrados, que en este caso son el desaprovechamiento y el valor de liberación. Estos objetivos están en conflicto ya que un pequeño cambio en el valor de alguno de ellos puede mejorar o empeorar el otro. En la gráfica se muestra que ambos valores se buscan minimizar, si observamos, en los tres casos, los puntos tienden a la izquierda y hacia abajo. Lo anterior muestra que el algoritmo funciona como se esperaba, ya que sin importar que las gráficas sean de ejecuciones distintas, los valores en los objetivos muestran una tendencia similar (tienden a valores pequeños).

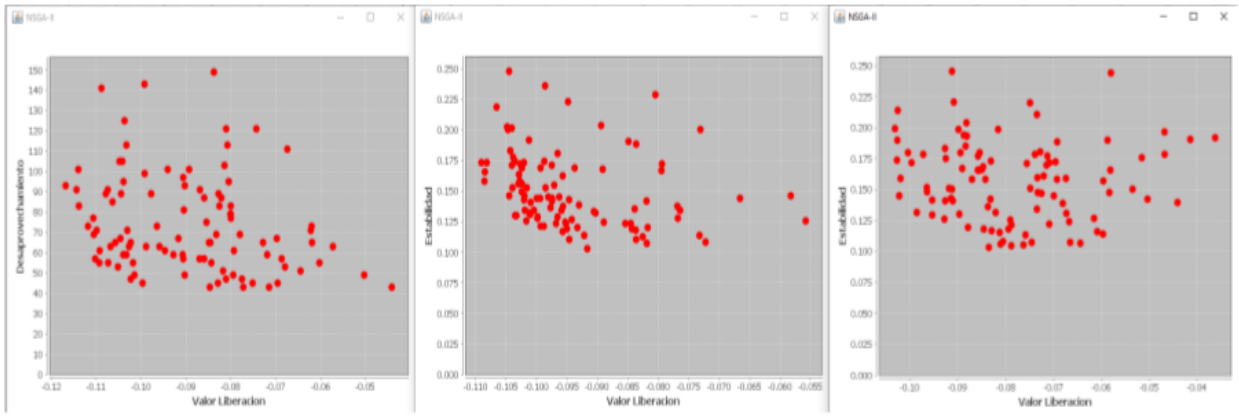



Fig. 34. Desaprovechamiento y valor de liberación, objetivos en conflicto.

Por último, encontramos la Fig. 35, que muestra los valores de las soluciones no dominadas, en esta figura se grafican los objetivos de estabilidad y desaprovechamiento los cuales presentan un comportamiento similar al caso anterior. Igualmente vemos que los puntos tienden a la izquierda y abajo lo cual indica que el algoritmo converge de manera adecuada. El objetivo de tiempo no se muestra ya que al presentarse las mejores soluciones este objetivo presenta un valor constante en casi todo el conjunto solución. Es decir, el algoritmo siempre encuentra la forma de calendarizar las HU de tal manera que esta sea la que menor tiempo tarda, por lo que en todas las propuestas de replaneación del conjunto solución el tiempo crece.

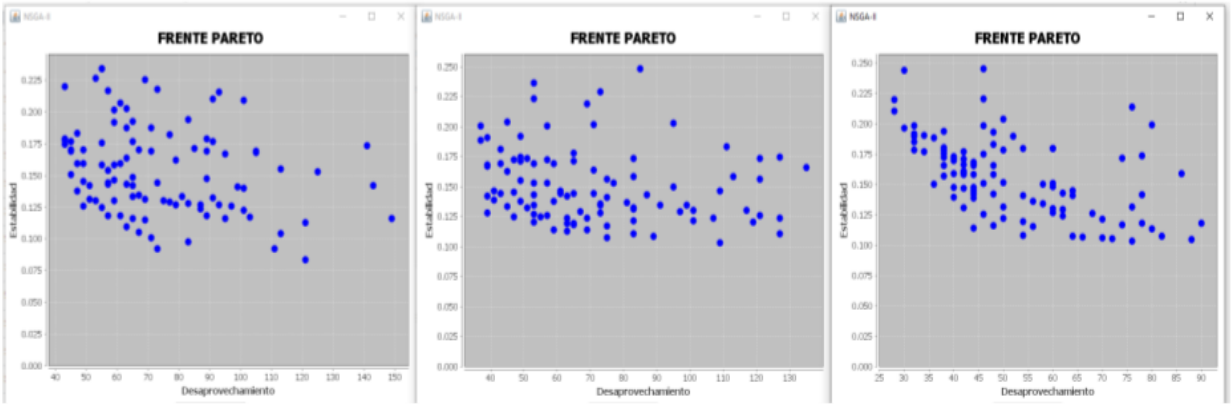



Fig. 35. Estabilidad y desaprovechamiento muestran que el algoritmo converge de manera satisfactoria.

## 6.3 Limitaciones del estudio

Debido a que no hay trabajos publicados que consideren el problema que aquí se analiza, es inevitable que existan ciertas limitaciones del estudio. A continuación se mencionan algunas.

No hay datos comparables, debido a que no hay datos estandarizados o públicos, sin embargo creemos que esto no es un problema, ya que los casos de prueba fueron generados con una herramienta especializada. Esta herramienta fue diseñada para crear instancias válidas de proyectos de desarrollo de software.

Otra limitante es que no teníamos datos de proyectos reales, sin embargo consideramos que esto no es un problema, ya que las pruebas muestran que las planeaciones son factibles y pertinentes, ya que se mostró con graficas que el algoritmo funciona como se esperaba al realizar una replaneación. Los objetivos tienen el impacto esperado cuando una planeación presenta su mejor valor o un equilibrio entre objetivos.

Adicionalmente, una limitante podría ser que las pruebas no se realizaron considerando dependencias, sin embargo éstas no afectan de forma significativa las métricas que se utilizaron para medir el desempeño y la calidad del algoritmo.

# Capítulo 7

# Conclusiones

A continuación se presentará en las siguientes secciones: la síntesis, las conclusiones y las contribuciones de este trabajo.

## 7.1 Síntesis

El problema de replaneación de liberaciones se puede resumir como que al ocurrir un evento disruptivo, se debe ajustar de manera rápida y con el menor número de cambios la planeación de liberación que se ha establecido. De modo que el costo total, el tiempo de desarrollo, la estabilidad del plan, el valor de liberación y desaprovechamiento del equipo, no se vean afectados o que en su defecto la afectación sea mínima. Esto es importante ya que el desarrollo de software es costoso y tiene fechas de entrega definidas, en las metodologías ágiles los tiempo de entrega de cada liberación son de unas cuantas semanas, por lo que una replaneación rápida y eficiente es aún más importante.

Un experto tarda en promedio 180 minutos en realizar una sola planeación sin mostrar con resultados, que esta sea lo mejor para el proyecto. Por lo anterior se tiene la necesidad de una herramienta para los Scrum Master (lider de proyecto en Scrum ) y demás responsables de realizar la planeación de liberación, que sirva como apoyo al realizar una replaneación.

La revisión bibliográfica indicó que existen pocos trabajos sobre replaneación en las metodologías ágiles, de hecho no se encontró uno solo que considere las características principales de estas. Todos los trabajos que existen presentan objetivos y características de las metodologías tradicionales, por lo anterior, nuestro trabajo es una contribución, ya que identifica y modela las características en el contexto ágil.

Se analizaron las características principales de las metodologías ágiles y en especial de Scrum, para presentar en este trabajo un modelo para el problema de replaneación de liberaciones en proyectos que se desarrollan en el contexto ágil con Scrum. También se definieron los eventos disruptivos que obligan a realizar una replaneación.

El problema de replaneación de liberaciones podemos considerarlo como una generalización del problema de asignación, el cual es un problema de la clase NP-difícil. Por lo tanto, para resolverlo, se propuso un algoritmo genético multiobjetivo que implemen ta NSGA-II.

Para definir el RLPAS se desarrolló un modelo matemático compuesto por cinco objetivos, el cual sirvió para proponer una solución, la cual en este trabajo se presenta como un algoritmo genético multiobjetivo basado en NSGA-II. Este algoritmo se implementa en una herramienta la cual muestra como resultado, un conjunto de propuestas de replaneación después de un evento disruptivo, las cuales son fáciles de interpretar.

Los resultados de las pruebas muestran que los resultados son coherentes y las propuestas de replaneación, resultado de la ejecución de la herramienta son correctas.

## 7.2 Conclusiones

A diferencia de los trabajos encontrados en la revisión bibliográfica, la propuesta de solución descrita en este trabajo implementa las características específicas del desarrollo ágil enfocado en el uso de Scrum. Estas características son los conceptos de puntos de historia y velocidad de sprint. Además, a diferencia de considerar la planeación completa de un proyecto, como lo hacen la mayoría de las propuestas encontradas en la literatura, aquí nos enfocamos en la planeación de liberaciones, que es un concepto similar pero en un contexto de metodologías ágiles.

El objetivo principal de este trabajo el cual es “Desarrollar un algoritmo heurístico para el problema de replaneación de liberaciones en proyectos ágiles de software”, fue alcanzado de manera satisfactoria, ya que se desarrolló un algoritmo heurístico basado en NSGA-II para el problema de replaneación de liberaciones en proyectos ágiles de software. Este algoritmo da como resultado un conjunto de propuestas de replaneación. Además, se presentó una herramienta como apoyo al RLPAS, la cual presenta propuestas de replaneación que son fáciles de interpretar por el Scrum master y los encargados del desarrollo de la planeación de las liberaciones del proyecto. Para lograr lo anterior, fue necesario completar cada uno de los objetivos específicos propuestos.

Se logró identificar las características del desarrollo de proyectos en un contexto ágil e integrar los conceptos de planeación de liberación, sprint, puntos de historia y velocidad de sprint a nuestro trabajo, con esto se alcanzó el primer objetivo específico, que se definió como “Identificar las características específicas de la planeación de liberaciones usando Scrum”.

Después, se desarrolló un modelo matemático para el problema de RLPAS, con lo cual, cumplimos el segundo y tercer objetivo específico, estos objetivos específicos son “Definir objetivos en el contexto de la planeación de liberaciones en Scrum” y “Definir un modelo matemático para el problema”, respectivamente. Este modelo es el primero en definir objetivos especialmente identificados en un contexto ágil como lo es Scrum. Para este modelo se definieron con funciones matemáticas cada una de los criterios que consideramos como los más importantes al realizar una replaneación después de un

evento disruptivo, los cuales son los objetivos del AG multiobjetivo: tiempo, costo, estabilidad, desaprovechamiento y valor de liberación.

Por último, se alcanzó ampliamente el cuarto objetivo específico, definido como “Desarrollar un algoritmo heurístico que cubra la necesidad de obtener de manera rápida un conjunto de propuestas de replaneación de liberaciones, después de ocurrir un evento disruptivo”, al desarrollar e implementar en una herramienta un algoritmo heurístico para el RLPAS, el cual nos permite obtener un conjunto de propuestas de planeación de liberaciones.

## 7.3 Contribuciones

Las contribuciones que se realizan con este trabajo son las siguientes:

1. Se presenta un modelo matemático para el problema de replaneación de liberaciones en proyectos ágiles de software (RLPAS).
2. Este modelo es el primero en presentar características específicas de la planeación y planeación en el desarrollo ágil con Scrum, las cuales son: planeación de liberación, sprint, puntos de historia y velocidad de sprint. Además de considerar las características comunes en todos los proyectos, como son: dependencia entre HU, experiencia de los empleados, entre otras; las cuales encontramos en el modelo de dominio de la sección 4.5.
3. Se introducen los objetivos de desaprovechamiento del equipo de desarrollo y valor de liberación, los cuales son características fundamentales al realizar una planeación de liberación.
4. El AG multiobjetivo que se desarrolló, cubre la necesidad de obtener de manera rápida un conjunto de propuestas de replaneación de liberación después de un evento disruptivo.
5. Se desarrolló una herramienta que puede ser usada como apoyo por los Scrum Master y otros encargados de hacer la planeación de liberaciones, la cual presenta propuestas de replaneación que son fácil de interpretar visualmente.

## 7.4 Trabajo futuro

A lo largo del desarrollo de este trabajo se estudiaron distintas técnicas y surgieron algunas otras ideas con las que tal vez, el modelo podría ser nutrido para buscar modelar de mejor manera algunas características del desarrollo de proyectos en la industria. De modo que como trabajo futuro se propone obtener datos de proyectos realizados en la industria del software. Con estos datos simular un evento disruptivo y ejecutar el algoritmo, para comparar los resultados de la herramienta, con los resultado de las decisiones que tomaron los Scrum Master en los proyectos. Otra propuesta para el trabajo futuro es tomar en cuenta la curva de aprendizaje de cada empleado y el gasto que se realiza en la comunicación

entre desarrolladores al realizar una HU, y cómo esto afecta al desarrollo. La comunicación debe ser limitada a un número máximo de empleados que pueden trabajar juntos y así evitar una sobrecarga de comunicación en el proyecto. Por último, como propuesta de trabajo a futuro, se podrían asignar perfiles a los empleados, por ejemplo: “desarrollador”, “tester”, etc., con el fin de acercar un poco más nuestro modelo a los equipos que desarrollan los proyectos en la industria.

De igual manera se podrían implementar otro tipo de heurísticas, como MOEA/D y SMS-EMOA. Se buscará implementar una segunda heurística para mejora de la población resultado (optimización local). Para obtener más datos de los resultados es recomendable estudiar otros indicadores de calidad.

Referencias

[1] De los Cobos S., Goddard J., Gutiérrez M. A., Martínez A. E., “Búsqueda y Exploración Estocástica”, UAM Iztapalapa, 2010

[2] Reeves C. R. (1995). “Modern heuristic techniques for combinatorial problems. Advanced topics in computer science”. Mc Graw-Hill.

[3] X. Shen, L. L. Minku, R. Bahsoon, & X. Yao, “Dynamic software project scheduling through a proactive-rescheduling method”, IEEE Transactions on Software Engineering, vol. 42(7), p. 658-686, 2016.

[4] X. Shen, L. L. Minku, N. Marturi, Y. N. Guo, & Y. Han,“A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling”, Information Sciences,vol. 428,p. 1-29, 2018.

[5] W. N. Chen, & J. Zhang, “Ant colony optimization for software project scheduling and staffing with an event-based scheduler”, IEEE Transactions on Software Engineering, 2012.

[6] L. Roque , A. Araújo , A. Dantas , R Saraiva, J. Souza , “Human Resource Allocation in Agile Software Projects Based on Task Similarities”, Lecture Notes in Computer Science, vol 9962. Springer, Cham, 2016.

[7] N. Nigar, “Model-based dynamic software project scheduling”, In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering , p. 1042-1045, ACM, August 2017.

[8] M. Jahr , “A hybrid approach to quantitative software project scheduling within agile frameworks”. Project Management Journal, vol. 45, p. 35–45, 2014.

[9] Y. Ge, “Software project rescheduling with genetic algorithms”, In Artificial Intelligence and Computational Intelligence, AICI'09, International Conference on Vol. 1, p. 439-443,IEEE, 2009.

[10] Y. Ge , & B. Xu, “Dynamic Staffing and Rescheduling in Software Project Management: A Hybrid Approach”, PloS one, 11(6), e0157104, 2016.

[11] J. Xiao, L. J. Osterweil, Q. Wang, & M. Li,“Dynamic resource scheduling in disruption-prone software development environments”, In International Conference on Fundamental Approaches to Software Engineering, p. 107-122. Springer, Berlin, Heidelberg, March 2010.

[12] J. Sutherland, & K. Schwaber, “The scrum guide. The definitive guide to scrum: The rules of the game”, Scrum org., vol. 268, 2013.

[13] Cohn, M., “Agile estimating and planning. Pearson Education”, 2005.

[14] Gueorguiev, S., Harman, M., & Antoniol, G., “Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering.”, In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp. 1673-1680, ACM., July 2019

[15] Beume, N., Fonseca, C. M., López-Ibáñez, M., Paquete, L., & Vahrenhold, J., “On the complexity of computing the hypervolume indicator.” Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 1075-1082. IEEE, 2009.

[16] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T., “A fast and elitist multiobjective genetic algorithm: NSGA-II. transactions on evolutionary computation”, vol. 6, no 2, p. 182-197. IEEE, 2002.

[17] E. Alba, J. Francisco Chicano, “Software project management with GAs”, Information Sciences, vol. 177(11), p. 2380-2401, 2007.

[18] K. S. Rubin, “Essential Scrum: a practical guide to the most popular agile process, Addison-Wesley, 2012.

[19] Software Guru, https://sg.com.mx/revista/58/estudio-de-salarios-sg-2019, 25 del Julio de 2019.

[20] El Economista, https://www. eleconomista .com.mx /tecnologia /Empresas-gastaran-7831 -millones-de-dolares-en-servicios-de-TI-en-Mexico-20190304-0057.html, 25 de Julio del 2019.

[21] IEEE Standard Glossary of Software Engineering Terminology, IEEE std 610., p. 12-1990, 1990.

[22] Kuhn, H. W. , “The Hungarian method for the assignment problem. Naval research logistics quarterly”, vol. 2(1-2),p. 83-97, 1955.

[23] Fisher, M. L., Jaikumar, R., & Van Wassenhove, L. N., “A multiplier adjustment method for the generalized assignment problem. Management science, vol. 32(9), p. 1095-1103, 1986.

[24] PMI, https://www.pmi.org/ , 6 de Agosto del 2019.

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

# ACTA DE EXAMEN DE GRADO

No. 00082
**Matrícula: 2173802421**

Aplicación de un algoritmo genético multiobjetivo para la replaneación de liberaciones en proyectos ágiles de software

VICTOR HUGO ESCANDON BAILON
ALUMNO

REVISÓ

MTRA. ROSALIA SERRANO DE LA PAZ
DIRECTORA DE SISTEMAS ESCOLARES

En la Ciudad de México, se presentaron a las 11:00 horas del dia 6 del mes de diciembre del año 2019 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DR. SAUL ZAPOTECAS MARTINEZ
MTRO. VICTOR HUGO GOMEZ GOMEZ
DR. ABEL GARCIA NAJERA

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: VICTOR HUGO ESCANDON BAILON

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.

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JESUS ALBERTO OCHOA TAPIA

PRESIDENTE

DR. SAUL ZAPOTECAS MARTINEZ

VOCAL

MTRO. VICTOR HUGO GOMEZ GOMEZ

SECRETARIO

DR. ABEL GARCIA NAJERA

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Casa abierta al tiempo

División de Ciencias Básicas e Ingeniería

Unidad Iztapalapa

Posgrado en Ciencias y Tecnologías de la Información

# Aplicación de un algoritmo genético multiobjetivo para la replaneación de liberaciones en proyectos ágiles de software

Idónea Comunicación de Resultados para obtener el grado de

Maestro en Ciencias

Presenta:

**Victor Hugo Escandon Bailon**

Asesores:

**Dr. Abel García Nájera**

**Dr. Humberto Cervantes Maceda**

Sinodales:

**Dr. Saúl Zapotecas Martínez**

**Mtro. Victor Hugo Gómez**

**Ciudad de México, 6 de Diciembre del 2019**