Optimización matemática de la planificación de sprints en equipos Scrum

Denis Ordoñez
Escuela de Matemática y Ciencias de la Computación
Universidad Nacional Autónoma de Honduras
e-mail: daordonezs@unah.hn

ÍNDICE

I. Introducción 2

II. Preliminares y Notación 2
II-A. Metodologías ágiles y Scrum . . . 2
II-B. Optimización matemática . . . 3
II-C. Aplicaciones de optimización en entornos ágiles . . . 3

III. Defnición del Problema 3
III-A. Ejemplo ilustrativo: La planificación tradicional de un sprint . . . 3

IV. Metodología 4
IV-A. Modelado del Problema . . . 4
IV-B. Función Objetivo . . . 4
IV-C. Restricciones del Modelo . . . 4
IV-D. Uso de Julia como entorno de resolución . . . 5
IV-E. Procedimiento metodológico . . . 5

V. Experimentos 6
V-A. Experimento 1: Validación de Balance y Cumplimiento de Cargas Mínimas . . . 6
V-B. Experimento 2: Análisis de Robustez ante Cambios en la Duración del Sprint . . . 6

VI. Resultados 7
VI-A. Resultados obtenidos Experimento 1 . . . 7
VI-B. Resultados obtenidos Experimento 2 . . . 7

VII. Análisis e Interpretación 8

VIII. Conclusiones y Recomendaciones 8

IX. Trabajo Futuro 9

ÍNDICE DE FIGURAS

1. Flujo general del proceso Scrum, mostrando las principales etapas y roles involucrados en la planificación y ejecución de un sprint. . . . 3
2. se representa gráficamente la distribución de las horas asignadas por desarrollador, permitiendo observar la utilización equilibrada de la capacidad de trabajo. . . . 4
3. Carga total de puntos asignados a cada desarrollador en un sprint normal. . . . 7
4. Carga total de puntos asignados a cada desarrollador en un sprint de 2 semanas. . . . 7
5. Distribución de trabajo total por desarrollador. . . . 8
6. Comparación de carga de trabajo por persona en dos escenarios. . . . 8

ÍNDICE DE CUADROS

I. Descripción de las historias de usuario disponibles en el *Product Backlog*, indicando el esfuerzo estimado, el valor de negocio y la habilidad técnica requerida. . . . 4
II. Asignación óptima de historias de usuario a los desarrolladores, considerando las restricciones de capacidad y habilidades. . . . 4
III. Listado de tareas y sus pesos asignados Experimento 1 . . . 6
IV. Desarrolladores y cargas mínimas asignadas Experimento 1 . . . 6
V. Lista de tareas y pesos Experimento 2 . . . 6
VI. Desarrolladores y cargas mínimas asignadas Experimento 2 . . . 6
VII. Resultados de Experimento 1 en Asignación de tareas por desarrollador . . . 7
VIII. Resultados de Experimento 1 en Asignación de tareas por desarrollador . . . 7
IX. Resultados de Experimento 2 en la asignación de tareas por desarrollador en el Caso B . . . 7

Optimización matemática de la planificación de sprints en equipos Scrum

***Resumen*—El presente trabajo aborda la optimización de un problema de asignación de tareas mediante técnicas de programación lineal, aplicado al contexto de equipos que operan bajo el marco ágil Scrum. En muchas organizaciones, la distribución ineficiente de tareas dentro de los sprints genera desequilibrios de carga, reduciendo la productividad y afectando los objetivos de entrega.**

El propósito principal de esta investigación es desarrollar un modelo matemático que asigne de forma equilibrada las tareas a los desarrolladores, considerando su experiencia, disponibilidad y capacidades técnicas. El modelo busca maximizar la eficiencia operativa del equipo y garantizar el cumplimiento de los objetivos de cada sprint.

Como aporte práctico, se propone una herramienta basada en optimización matemática que facilita la planificación de sprint y mejora la toma de decisiones en entornos colaborativos. La implementación se realiza mediante un método de programación lineal, lo que permite obtener soluciones óptimas de manera sistemática y reproducible.

En conjunto, esta propuesta integra los principios de las metodologías ágiles con modelos cuantitativos de optimización, ofreciendo una solución adaptable a distintos equipos y contextos empresariales, orientada a incrementar la productividad y la satisfacción del equipo de desarrollo.

I. Introducción

En los entornos actuales de desarrollo de software, particularmente aquellos que aplican metodologías ágiles como Scrum, uno de los desafíos más frecuentes es la asignación eficiente de tareas entre los miembros del equipo. Esta problemática se torna crítica cuando se busca garantizar una distribución equilibrada de la carga de trabajo, considerando diferencias en experiencia, habilidades técnicas y disponibilidad de cada desarrollador. Una asignación inadecuada no solo disminuye el rendimiento del equipo, sino que también puede comprometer el cumplimiento de los objetivos del sprint y afectar la calidad del producto entregado.

Tradicionalmente, la asignación de tareas en equipos Scrum se realiza de manera manual durante reuniones de planificación, donde el equipo estima y selecciona actividades según la disponibilidad y el criterio de cada miembro. Aunque este enfoque fomenta la auto-organización y la colaboración, no siempre produce resultados óptimos, especialmente en contextos con múltiples restricciones y objetivos de productividad exigentes, como los que se encuentran en entidades financieras.

En esta investigación se propone un enfoque basado en programación lineal entera, implementado mediante el Método Simplex, para optimizar la asignación de tareas. Este modelo permite considerar simultáneamente la capacidad, las habilidades y la disponibilidad de cada desarrollador, asegurando un uso eficiente de los recursos humanos. La implementación de esta estrategia busca mejorar la eficiencia del equipo, garantizar el cumplimiento de las métricas del sprint y ofrecer una solución formal y adaptable a diferentes contextos de desarrollo ágil, constituyendo un aporte práctico al manejo de la planificación en entornos colaborativos.

II. Preliminares y Notación

II-A. Metodologías ágiles y Scrum

Las metodologías ágiles surgieron como una respuesta a las limitaciones de los enfoques tradicionales de gestión de proyectos, caracterizados por procesos rígidos y una planificación extensiva a largo plazo. Estas metodologías se basan en la adaptabilidad, la comunicación continua y la entrega incremental de valor al cliente.

Scrum, en particular, es un marco de trabajo ágil que organiza el desarrollo de productos en ciclos cortos denominados *sprint*, los cuales suelen tener una duración de entre una y cuatro semanas. En cada sprint, el equipo selecciona un conjunto de tareas del *Product Backlog* y las desarrolla con el objetivo de entregar un incremento funcional del producto.

Los roles principales en Scrum son los siguientes:

- **Product Owner:** responsable de gestionar el *backlog* del producto, priorizando las historias de usuario de acuerdo con el valor de negocio y las necesidades del cliente.
- **Scrum Máster:** facilitador del proceso y garante de que el equipo siga los principios ágiles, eliminando impedimentos y promoviendo la mejora continua.
- **Equipo de desarrollo:** grupo multidisciplinario encargado de implementar las funcionalidades acordadas en cada sprint.

Los eventos más relevantes dentro del marco de Scrum incluyen:

- **Planificación del sprint (*Sprint Planning*):** reunión en la que se define qué trabajo se realizará durante el sprint.
- **Reuniones diarias (*Daily Scrum*):** encuentros breves que permiten al equipo coordinar esfuerzos y detectar obstáculos.
- **Revisión del sprint (*Sprint Review*):** evaluación de los resultados obtenidos al finalizar el sprint.
- **Retrospectiva (*Sprint Retrospective*):** análisis interno para identificar oportunidades de mejora en el proceso.

Estos eventos garantizan la transparencia, la inspección y la adaptación continua del trabajo, pilares fundamentales del enfoque ágil.

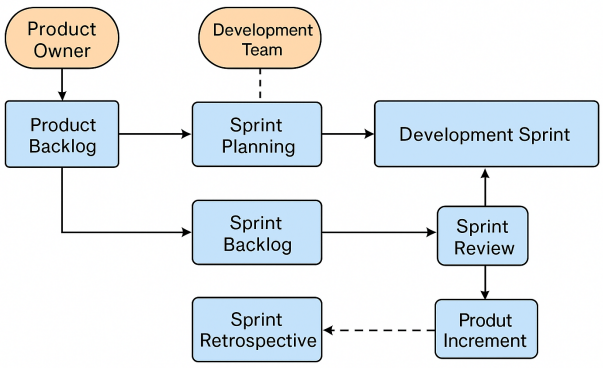

Figura 1. Flujo general del proceso Scrum, mostrando las principales etapas y roles involucrados en la planificación y ejecución de un sprint.

II-B. Optimización matemática

La optimización matemática es una rama de las matemáticas aplicadas que se ocupa de encontrar el mejor resultado posible, de acuerdo con un criterio determinado, dentro de un conjunto de alternativas viables. Dependiendo de la naturaleza del problema, se pueden emplear diferentes enfoques, tales como:

- **Programación lineal (PL):** utilizada cuando tanto la función objetivo como las restricciones son lineales.
- **Programación entera (PE):** empleada cuando las variables de decisión deben tomar valores enteros.
- **Programación lineal entera mixta (PLEM):** combina variables continuas y discretas, siendo una de las más comunes en problemas de asignación y planificación.
- **Métodos heurísticos o metaheurísticos:** tales como algoritmos genéticos, búsqueda tabú o recocido simulado, empleados cuando el espacio de búsqueda es demasiado grande o complejo para técnicas exactas.

En el ámbito de la gestión de proyectos, los modelos de optimización permiten representar y resolver problemas relacionados con la asignación de recursos, la programación de tareas, la minimización de costos o la maximización de beneficios. Estos modelos facilitan la toma de decisiones racionales, basadas en datos y restricciones explícitas.

II-C. Aplicaciones de optimización en entornos ágiles

En los últimos años, diversos estudios han explorado el uso de la optimización matemática en el contexto de la gestión ágil de proyectos. Algunos autores han formulado modelos destinados a la selección de historias de usuario considerando el valor, el esfuerzo y las dependencias existentes; otros se han centrado en la asignación óptima de tareas a los desarrolladores según su nivel de habilidad y disponibilidad.

Sin embargo, la mayoría de estos trabajos abordan aspectos específicos del proceso ágil de manera independiente, sin integrar todas las variables que intervienen en la planificación de un sprint. Por ejemplo, algunos modelos se enfocan únicamente en la priorización del *backlog*, mientras que otros tratan la asignación de recursos sin considerar las interdependencias entre tareas.

Esta fragmentación plantea la necesidad de desarrollar un enfoque más integral que incorpore simultáneamente los distintos elementos del proceso de planificación: la priorización de historias, la capacidad del equipo, las competencias individuales y las dependencias técnicas. La combinación de estos factores en un modelo de optimización matemática puede aportar una herramienta eficaz para mejorar la toma de decisiones en entornos ágiles y aumentar la eficiencia global de los equipos Scrum.

III. Defnición del Problema

En la planificación de un sprint dentro del marco de trabajo Scrum, el equipo de desarrollo debe seleccionar el conjunto de historias de usuario o tareas que abordará en un período determinado. Esta decisión requiere equilibrar diversos factores, como la capacidad disponible, la complejidad de las tareas, las habilidades requeridas y las prioridades definidas por el *Product Owner*.

Aunque el proceso de planificación busca generar compromisos realistas y sostenibles, en la práctica suele depender en gran medida de la experiencia del equipo o del criterio del *Scrum Máster*. La ausencia de un enfoque cuantitativo puede ocasionar ineficiencias, tales como sobre asignación de tareas, subutilización de recursos o selección de historias que no maximizan el valor entregado al cliente.

El problema se intensifica a medida que aumenta la complejidad del proyecto o el tamaño del equipo, ya que las posibles combinaciones entre tareas y desarrolladores crecen exponencialmente. En estos casos, la toma de decisiones basada únicamente en la intuición resulta insuficiente para garantizar una planificación óptima.

Ante esta situación, se propone formalizar matemáticamente el proceso de planificación de sprints, integrando explícitamente las restricciones y objetivos involucrados. Mediante técnicas de optimización matemática, es posible construir un modelo que determine qué tareas deben seleccionarse y cómo asignarlas a los miembros del equipo, con el propósito de maximizar el valor total del sprint y mantener un equilibrio en la carga de trabajo.

En síntesis, este trabajo plantea el diseño de un modelo de optimización como apoyo a la planificación de sprints en equipos Scrum. El enfoque propuesto busca una asignación más equitativa y eficiente de las tareas, contribuyendo a mejorar la productividad global y la calidad del producto desarrollado.

III-A. Ejemplo ilustrativo: La planificación tradicional de un sprint

Planteamiento del caso: Supóngase un equipo de desarrollo compuesto por tres miembros: Ana, Luis y Carla. Cada uno dispone de una capacidad estimada de **40 horas** para el próximo sprint. El *Product Backlog* contiene cinco historias de usuario, descritas en el Cuadro I.

Los desarrolladores poseen las siguientes habilidades principales:

Cuadro I
DESCRIPCIÓN DE LAS HISTORIAS DE USUARIO DISPONIBLES EN EL *Product Backlog*, INDICANDO EL ESFUERZO ESTIMADO, EL VALOR DE NEGOCIO Y LA HABILIDAD TÉCNICA REQUERIDA.

Historia	Esfuerzo (horas)	Valor (puntos)	Habilidad requerida
H1	30	13	Frontend
H2	20	5	Backend
H3	25	8	Backend
H4	15	5	Testing
H5	10	3	Frontend

- Ana: Frontend y Testing
- Luis: Backend
- Carla: Backend y Testing

Resultados y Visualización: La solución obtenida mediante el uso de asignación empírica, se muestra en el Cuadro II.

Cuadro II
ASIGNACIÓN ÓPTIMA DE HISTORIAS DE USUARIO A LOS DESARROLLADORES, CONSIDERANDO LAS RESTRICCIONES DE CAPACIDAD Y HABILIDADES.

Desarrollador	Historias asignadas	Horas utilizadas
Ana	H1, H5	40
Luis	H2	20
Carla	H3, H4	40

El valor total obtenido en el sprint es:

$$Z^* = 13 + 5 + 8 + 5 + 3 = 34 \text{ puntos.}$$

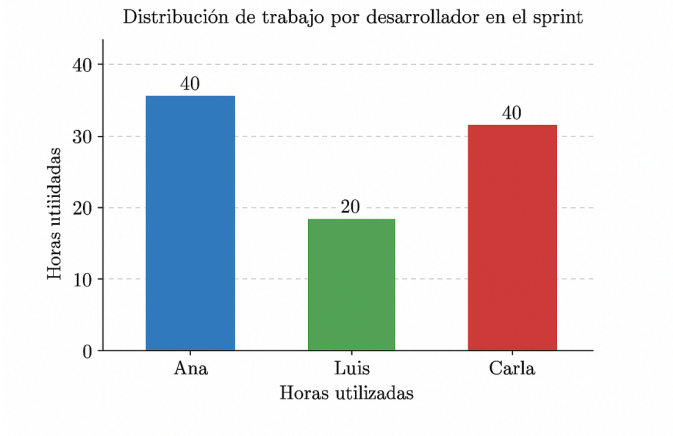

Figura 2. se representa gráficamente la distribución de las horas asignadas por desarrollador, permitiendo observar la utilización equilibrada de la capacidad de trabajo.

IV. METODOLOGÍA

IV-A. *Modelado del Problema*

Se definen los siguientes conjuntos:

$$P = \{p_1, p_2, \ldots, p_5\} \quad \text{(desarrolladores)}$$
$$T = \{t_1, t_2, \ldots, t_{12}\} \quad \text{(tareas)}$$
$$W = \{1, 2, 3, 5, 8, 13\} \quad \text{(pesos posibles)}$$

Variables de decisión:

- Asignación binaria de tareas:

$$x_{p,t} = \begin{cases} 1, & \text{si la tarea } t \text{ es asignada a } p \\ 0, & \text{en otro caso} \end{cases}$$

- Selección del peso para cada tarea:

$$y_{t,w} = \begin{cases} 1, & \text{si el peso } w \text{ es asignado a la tarea } t \\ 0, & \text{en otro caso} \end{cases}$$

- Variables auxiliares:

$$pt_t \geq 0 \quad \text{(peso total asignado a la tarea } t)$$
$$z_{p,t} \geq 0 \quad \text{(carga de la tarea } t \text{ asignada a } p)$$
$$carga_p \geq 0 \quad \text{(carga total de } p)$$

IV-B. *Función Objetivo*

Se desea maximizar la distribución equitativa de las tareas:

$$\boxed{\text{máx} \quad obj = \sum_{p \in P} \sum_{t \in T} z_{p,t}}$$

IV-C. *Restricciones del Modelo*

a) Asignación y unicidad:

$$\sum_{p \in P} x_{p,t} = 1, \quad \forall t \in T$$

$$\sum_{w \in W} y_{t,w} = 1, \quad \forall t \in T$$

$$pt_t = \sum_{w \in W} w \cdot y_{t,w}, \quad \forall t \in T$$

b) Linealización del producto: En modelos de programación lineal, las variables y restricciones deben ser lineales para garantizar la aplicabilidad de los métodos de solución, como el Método Simplex [(5)]. Sin embargo, en muchos problemas prácticos, se requiere representar productos entre variables, por ejemplo, entre una variable continua y una binaria, lo cual genera términos no lineales que complican la resolución directa.

El proceso de **linealización** consiste en transformar estas expresiones no lineales en un conjunto equivalente de restricciones lineales, introduciendo variables auxiliares y restricciones adicionales que preservan la relación original. Esto permite mantener la linealidad del modelo y utilizar algoritmos eficientes para obtener soluciones óptimas.

En particular, para el producto de una variable continua pt_t y una variable binaria $x_{p,t}$, se introduce una variable auxiliar $z_{p,t}$ que representa el producto:

$$z_{p,t} = pt_t \cdot x_{p,t}$$

Para garantizar que $z_{p,t}$ tome el valor correcto, se aplican las siguientes restricciones lineales:

$$\begin{cases} z_{p,t} \leq pt_t \\ z_{p,t} \leq M \cdot x_{p,t} \\ z_{p,t} \geq pt_t - M(1 - x_{p,t}) \\ z_{p,t} \geq 0 \end{cases} \quad \forall p \in P,\ t \in T$$

Donde M es una constante suficientemente grande que actúa como un límite superior para pt_t.

La lógica detrás de estas restricciones es la siguiente:
- Si $x_{p,t} = 0$, entonces $z_{p,t}$ debe ser cero, forzado por $z_{p,t} \leq M \cdot 0 = 0$ y la restricción de no negatividad.
- Si $x_{p,t} = 1$, entonces $z_{p,t}$ debe ser igual a pt_t, ya que $z_{p,t} \leq pt_t$ y $z_{p,t} \geq pt_t$.

Este método de linealización es fundamental en programación entera mixta y ha sido ampliamente documentado en la literatura clásica de investigación de operaciones [(9), (8)].

$$M = 20$$

Finalmente, la carga total asignada a cada persona p se define como:

$$carga_p = \sum_{t \in T} z_{p,t}, \quad \forall p \in P$$

c) Carga mínima según experiencia: Se definen las variables C_s y C_j para representar los valores mínimos de carga asignados a desarrolladores senior y junior, respectivamente. Esto permite que el modelo sea más flexible y adaptable a distintos escenarios:

$$C_s \quad : \text{carga mínima para senior}$$
$$C_j \quad : \text{carga mínima para junior}$$

La restricción que garantiza la carga mínima según experiencia queda:

$$carga_p \geq C_s \cdot esSenior_p + C_j \cdot (1 - esSenior_p), \quad \forall p \in P$$

d) Restricción del rango total del sprint: El rango total del esfuerzo esperado para el sprint debe ajustarse acorde a la cantidad de senior y junior, ademas de los valores mínimos definidos:

$$|P_s| \cdot C_s + |P_j| \cdot C_j \leq \sum_{t \in T} pt_t \leq 1{,}2 \times (|P_s| \cdot C_s + |P_j| \cdot C_j)$$

Donde

$$|P_s| = \text{número de desarrolladores senior}$$
$$|P_j| = \text{número de desarrolladores junior}$$

Por ejemplo, si hay 3 senior y 2 junior, con cargas mínimas de 18 y 13 respectivamente:

$$3 \times 18 + 2 \times 13 \leq \sum_{t \in T} pt_t \leq 1{,}2 \times (3 \times 18 + 2 \times 13)$$

$$80 \leq \sum_{t \in T} pt_t \leq 96$$

IV-D. Uso de Julia como entorno de resolución

Para la implementación y resolución del modelo de optimización propuesto se utilizó el lenguaje **Julia**, en conjunto con el paquete **JuMP.jl**, el cual permite formular modelos de optimización matemática de manera declarativa y resolverlos mediante diferentes solvers. Julia se eligió por su alto rendimiento numérico y su facilidad para integrar componentes de análisis, visualización y automatización en un mismo entorno.

Estructura general del modelo en Julia: El desarrollo se realizó en el entorno de programación `Julia 1.10`, utilizando las siguientes bibliotecas principales:

- **JuMP.jl**: para la formulación del modelo matemático de optimización.
- **GLPK.jl**: como solver de programación lineal entera mixta (MIP).
- **DataFrames.jl** y **Plots.jl**: para el procesamiento y visualización de resultados.

Ventajas del uso de Julia en la optimización matemática: El uso de Julia presenta varias ventajas frente a entornos tradicionales, entre las que destacan:

- **Flexibilidad y apertura**: Julia es un lenguaje de código abierto que permite combinar optimización con análisis de datos, aprendizaje automático y visualización sin cambiar de entorno.
- **Desempeño**: su motor JIT (*Just-In-Time Compilation*) ofrece un rendimiento comparable al de C o Fortran en problemas numéricos intensivos.
- **Escalabilidad**: el modelo puede adaptarse fácilmente a equipos más grandes o a nuevas restricciones sin modificar la estructura general del código.
- **Integración**: los resultados pueden exportarse a formatos estándar o representarse mediante gráficos automáticos, lo que facilita su interpretación durante las reuniones de planificación del sprint.

IV-E. Procedimiento metodológico

El procedimiento seguido para aplicar el modelo es el siguiente:

1. **Recopilación de datos:** identificación de las tareas del backlog, estimación de esfuerzo y valor de cada tarea, registro de capacidades y habilidades de cada desarrollador.
2. **Formulación del modelo:** definición de variables, función objetivo y restricciones en el entorno de Julia.
3. **Resolución del modelo:** utilización del solver para obtener la asignación óptima de tareas.
4. **Análisis de resultados:** verificación de la distribución de tareas, carga de trabajo y cumplimiento de las métricas de sprint.
5. **Ajustes y validación:** ajustes del modelo para distintos escenarios y validación práctica con equipos de desarrollo.

Esta metodología garantiza que la planificación del sprint sea objetiva, eficiente y reproducible, integrando técnicas de optimización matemática con principios de Scrum para mejorar la productividad y la distribución de tareas dentro del equipo.

V. Experimentos

En esta sección se presentan dos experimentos diseñados para validar la efectividad y lo robusto del modelo de asignación óptima de tareas en equipos Scrum.

V-A. Experimento 1: Validación de Balance y Cumplimiento de Cargas Mínimas

Cuadro III
Listado de tareas y sus pesos asignados Experimento 1

Tarea	Peso (puntos)
t_1	13
t_2	8
t_3	13
t_4	8
t_5	13
t_6	5
t_7	8
t_8	8
t_9	13
t_{10}	3
t_{11}	2
t_{12}	2

Cuadro IV
Desarrolladores y cargas mínimas asignadas Experimento 1

Desarrollador	Experiencia	Carga mínima (puntos)
Carlos	Senior	18
Sofia	Senior	18
Denis	Senior	18
Ana	Junior	13
Luis	Junior	13

Objetivo: Evaluar si el modelo logra asignar las tareas de manera que se cumplan las cargas mínimas establecidas para desarrolladores, evitando sobrecargas o subutilización.

Diseño del Experimento:

- *Datos de entrada:*
  - 5 desarrolladores (3 seniors, 2 juniors), con cargas mínimas según Cuadro IV.
  - 12 tareas con pesos específicos seleccionados de la serie Fibonacci, listadas en la Cuadro III.
- *Condiciones:*
  - Carga mínima para seniors: 18 puntos.
  - Carga mínima para juniors: 13 puntos.
  - Suma total de puntos a asignar entre 80 y 96.
  - Constante grande $M = 20$.

Procedimiento:

1. Ejecutar el modelo con las restricciones y variables definidas.
2. Analizar la asignación final de tareas a cada desarrollador.
3. Verificar que cada desarrollador cumpla o supere su carga mínima requerida.
4. Comprobar que la suma total de puntos asignados esté dentro del rango permitido.

Resultados esperados:

- Asignación total de tareas.
- Ningún desarrollador con carga inferior a la mínima.
- Distribución equilibrada de cargas, con variaciones pequeñas entre seniors y juniors.

V-B. Experimento 2: Análisis de Robustez ante Cambios en la Duración del Sprint

Objetivo: Evaluar cómo varía la asignación de tareas cuando cambia la duración del sprint, manipulando la carga mínima requerida y el total de puntos a asignar.

Diseño del Experimento:

- Modificar la duración del sprint, alterando las cargas mínimas y totales:
  - **Caso A: Experimento 1**
  - **Caso B:** Sprint de 3 semanas (incrementar cargas mínimas proporcionalmente, por ejemplo seniors = 27, juniors = 20, total mínimo y máximo ajustados).

Cuadro V
Lista de tareas y pesos Experimento 2

Tarea	Peso (puntos)
t_1	13
t_2	2
t_3	13
t_4	5
t_5	8
t_6	13
t_7	13
t_8	2
t_9	3
t_{10}	5
t_{11}	8
t_{12}	13
t_{13}	13
t_{14}	2
t_{15}	3
t_{16}	5
t_{17}	8
t_{18}	13

Cuadro VI
Desarrolladores y cargas mínimas asignadas Experimento 2

Desarrollador	Experiencia	Carga mínima (puntos)
Carlos	Senior	27
Sofia	Senior	27
Denis	Senior	27
Ana	Junior	20
Luis	Junior	20

- *Datos de entrada:*
  - 5 desarrolladores (3 seniors, 2 juniors), con cargas mínimas según Cuadro VI.
  - 18 tareas con pesos específicos seleccionados de la serie Fibonacci, listadas en la Cuadro V.
- *Condiciones:*
  - Carga mínima para seniors: 27 puntos.
  - Carga mínima para juniors: 20 puntos.

- ◇ Suma total de puntos a asignar entre 121 y 146.
- ◇ Constante grande $M = 30$.

Procedimiento:

1. Ejecutar el modelo con parámetros para el sprint de 2 semanas (Caso A).
2. Ejecutar el modelo con parámetros ajustados para el sprint de 3 semanas (Caso B).
3. Comparar resultados de asignación, cargas individuales y totales.
4. Evaluar si el modelo se adapta adecuadamente a las nuevas restricciones.

Resultados esperados:

- Incremento proporcional en cargas asignadas según duración del sprint.
- Mantenimiento del balance entre desarrolladores.
- Comprobación de que el modelo puede adaptarse a distintos escenarios sin perder validez.

VI. Resultados

VI-A. Resultados obtenidos Experimento 1

Cuadro VII
Resultados de Experimento 1 en Asignación de tareas por desarrollador

Tarea	Carlos	Ana	Luis	Sofía	Denis
t_1	1	0	0	0	0
t_2	0	0	0	1	0
t_3	0	0	0	0	1
t_4	0	1	0	0	0
t_5	0	0	0	0	1
t_6	0	0	1	0	0
t_7	0	1	0	0	0
t_8	0	0	0	1	0
t_9	0	0	1	0	0
t_{10}	1	0	0	0	0
t_{11}	0	0	0	1	0
t_{12}	1	0	0	0	0

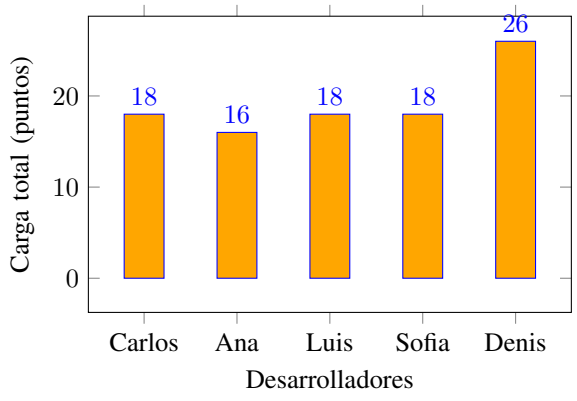

Figura 3. Carga total de puntos asignados a cada desarrollador en un sprint normal.

VI-B. Resultados obtenidos Experimento 2

Caso A

Cuadro VIII
Resultados de Experimento 1 en Asignación de tareas por desarrollador

Tarea	Carlos	Ana	Luis	Sofía	Denis
t_1	1	0	0	0	0
t_2	0	0	0	1	0
t_3	0	0	0	0	1
t_4	0	1	0	0	0
t_5	0	0	0	0	1
t_6	0	0	1	0	0
t_7	0	1	0	0	0
t_8	0	0	0	1	0
t_9	0	0	1	0	0
t_{10}	1	0	0	0	0
t_{11}	0	0	0	1	0
t_{12}	1	0	0	0	0

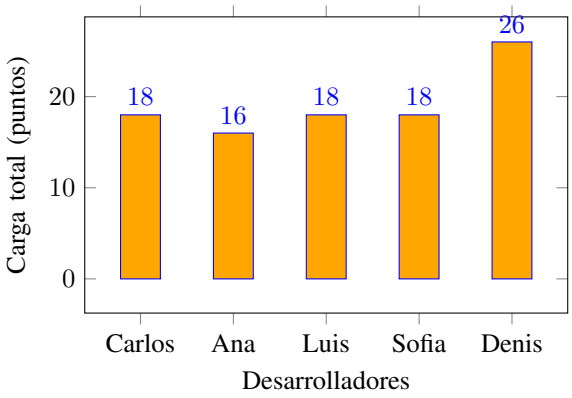

Figura 4. Carga total de puntos asignados a cada desarrollador en un sprint de 2 semanas.

Caso B

Cuadro IX
Resultados de Experimento 2 en la asignación de tareas por desarrollador en el Caso B

Tarea	Carlos	Ana	Luis	Sofía	Denis
t_1	0	0	1	0	0
t_2	1	0	0	0	0
t_3	0	0	1	0	0
t_4	0	0	0	1	0
t_5	1	0	0	0	0
t_6	0	1	0	0	0
t_7	0	0	0	1	0
t_8	1	0	0	0	0
t_9	0	0	0	1	0
t_{10}	0	1	0	0	0
t_{11}	0	0	0	1	0
t_{12}	0	0	0	0	1
t_{13}	0	0	0	0	1
t_{14}	0	0	0	0	1
t_{15}	1	0	0	0	0
t_{16}	1	0	0	0	0
t_{17}	0	0	0	1	0
t_{18}	1	0	0	0	0

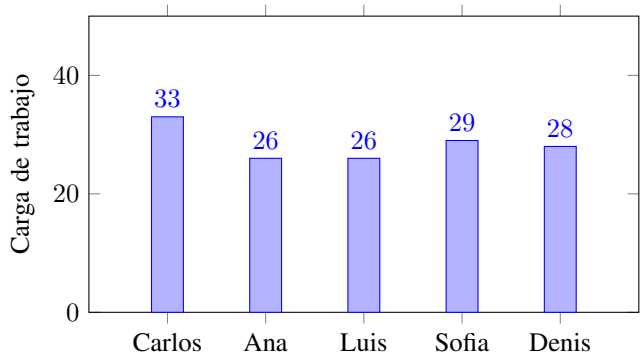

Figura 5. Distribución de trabajo total por desarrollador.

Comparación Caso A con Caso B

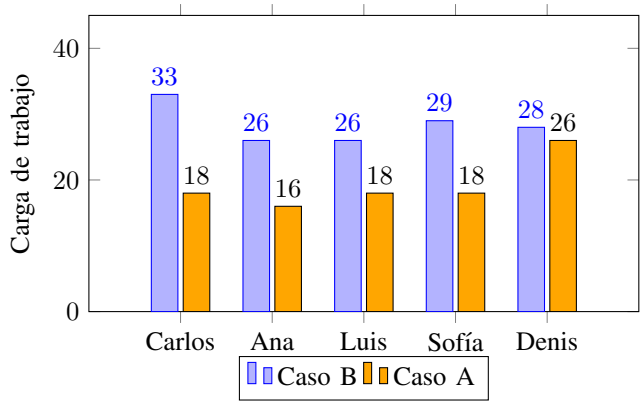

Figura 6. Comparación de carga de trabajo por persona en dos escenarios.

VII. Análisis e Interpretación

Los resultados muestran una distribución adecuada de las tareas, donde cada miembro asume una carga de trabajo coherente con su nivel de experiencia. Por ejemplo, los desarrolladores con mayor experiencia, como Denis, recibieron una cantidad ligeramente mayor de trabajo, lo que es razonable considerando su capacidad para manejar tareas más complejas. En contraste, integrantes con menos experiencia, como Ana, tuvieron una carga menor, permitiendo mantener el equilibrio general del equipo.

En términos de productividad global, el sprint logró alcanzar un valor total de **96 puntos**, lo cual indica que el equipo puede completar las tareas planificadas sin sobrecargarse. Este resultado se encuentra dentro del rango considerado factible, demostrando que el modelo de optimización logra un balance entre la eficiencia y la capacidad real del grupo.

De forma cualitativa, el modelo muestra tres ventajas principales:

- Asigna las tareas de forma objetiva, evitando decisiones basadas solo en percepciones o intuición.
- Aprovecha de manera eficiente las habilidades específicas de cada miembro del equipo.
- Permite alcanzar un nivel de carga de trabajo equilibrado que mejora la planificación y el bienestar del equipo.

En conclusión, el modelo de optimización demuestra ser una herramienta valiosa para apoyar la toma de decisiones en la planificación de sprints. Su aplicación permite distribuir el trabajo de manera justa y eficiente, reduciendo el riesgo de sobreasignación y aumentando la probabilidad de que el equipo cumpla con los objetivos del sprint sin comprometer la calidad del trabajo ni el bienestar de los desarrolladores.

VIII. Conclusiones y Recomendaciones

- La asignación de las 12 tareas en el Experimento 1 entre los cinco participantes se realizó de forma equilibrada, cubriendo la totalidad de la demanda. Carlos, Luis y Sofía recibieron cargas idénticas de 18 unidades, Denis 26 y Ana 16, como lo muestra la Figura 3 evidenciando una distribución homogénea salvo por la menor carga de Ana. La estrategia combinó tareas de diferentes pesos para equilibrar las cargas: algunos participantes recibieron menos tareas de mayor peso, mientras que otros tuvieron más tareas de menor peso. El valor objetivo alcanzado (96) confirma la asignación eficiente y completa, asegurando que cada tarea fue asignada a una sola persona sin solapamientos como se ve en el Cuadro VII y manteniendo un reparto cercano a la equidad.

- En el Experimento 2 en el Caso B el modelo logró una asignación factible y completa de las 18 tareas, garantizando que cada tarea se asignara a un único participante como lo denota el Cuadro IX. Sin embargo, la distribución resultante presenta disparidades significativas en las cargas acumuladas, con Carlos y Denis concentrando gran parte de los pesos más altos, mientras que Ana y Luis quedaron con cargas sensiblemente menores,lo mostrado en la Figura 5. Este patrón sugiere que el algoritmo priorizó la factibilidad y la cobertura total por encima de un equilibrio estricto en la carga de trabajo. En contextos reales, este tipo de resultado podría ser aceptable si las diferencias responden a criterios de capacidad, especialización o disponibilidad, pero si el objetivo fuera la equidad absoluta, sería necesario incorporar restricciones adicionales que limiten la variación máxima permitida entre cargas.

- En ambos escenarios se cumplió con la asignación completa de tareas. Aunque las cargas de trabajo presentaron cierta desigualdad como lo podemos ver en la Figura 6, esto responde a una estrategia orientada a priorizar la experiencia y habilidades específicas de cada participante por encima del equilibrio de distribución. El incremento del valor objetivo de 96 a 142 refleja que, en entornos con mayor complejidad y diversidad de tareas, es más eficiente asignar responsabilidades clave a quienes poseen mayor capacidad técnica, incluso si esto implica cargas superiores para ciertos miembros. Esta decisión fortalece la calidad y eficacia, aún sacrificando cierta equidad en el reparto.

- De manera general, el modelo aplicado demostró ser capaz de garantizar la asignación completa y factible de todas las tareas en los distintos escenarios planteados. En cada experimento se observó que la estrategia no solo aseguró que cada tarea fuese cubierta por un único participante sin solapamientos, sino que además permitió balancear en gran medida las cargas de trabajo, aunque con diferencias inevitables según el caso. Estos resultados muestran que el modelo es flexible y eficiente, pues puede priorizar tanto la equidad como la factibilidad dependiendo de la complejidad y características de las tareas. Si bien en algunos escenarios la distribución no fue perfectamente homogénea, esta aparente desigualdad se justifica al considerar criterios de capacidad, experiencia y especialización, lo que finalmente contribuye a mejorar la calidad y eficacia global del proceso de asignación.

IX. Trabajo Futuro

El modelo propuesto constituye un punto de partida para integrar la optimización matemática en la planificación de sprints bajo el marco de trabajo Scrum. Los resultados obtenidos muestran su potencial para mejorar la distribución de tareas y la eficiencia del equipo, sin embargo; existen diversas oportunidades de ampliación y mejora.

En primer lugar, se propone incorporar nuevos criterios en el modelo, como la prioridad de las historias de usuario, la interdependencia entre tareas y el nivel de urgencia de los requerimientos. Esto permitiría reflejar con mayor precisión las condiciones reales de un proyecto de desarrollo de software.

Asimismo, se sugiere comparar el enfoque de programación lineal con otros métodos de optimización, tales como algoritmos genéticos, heurísticas de búsqueda local o técnicas basadas en enjambres de partículas. Estas alternativas podrían generar soluciones eficientes en menor tiempo y adaptarse mejor a escenarios con mayor cantidad de tareas o equipos más grandes.

Otra línea de avance consiste en integrar el modelo dentro de plataformas de gestión ágil, como Jira o Trello, automatizando la asignación de tareas directamente en las herramientas utilizadas por los equipos de desarrollo. Esto facilitaría su adopción y permitiría validar su utilidad en contextos reales.

Finalmente, se plantea realizar estudios que evalúen el impacto del modelo en la productividad, la satisfacción de los desarrolladores y el cumplimiento de los objetivos del sprint. Este análisis permitiría valorar el beneficio práctico de la optimización, no solo desde un punto de vista técnico, sino también humano y organizacional.

En conjunto, estas propuestas buscan consolidar un sistema de apoyo a la decisión más completo, flexible y aplicable a distintos tipos de equipos, fortaleciendo la conexión entre la optimización matemática y las metodologías ágiles de desarrollo de software.

Agradecimientos

A mis padres, por su apoyo incondicional, su ejemplo constante y por motivarme a superar cada desafío con determinación y esfuerzo. A mis profesores, por compartir su conocimiento y guía durante todo este proceso formativo. Y a mis compañeros y conocidos dentro de la carrera, por su colaboración, amistad y por haber hecho de esta experiencia un camino de aprendizaje y crecimiento compartido.

Disponibilidad de Datos

El código fuente se puede revisar en el repositorio: [(4)]

Referencias

[(1)] Michael Becker et al. «Multi-sprint planning and smooth replanning: An optimization model». En: *Journal of Systems and Software* 86.6 (2013), págs. 1619-1630. DOI: 10.1016/j.jss.2013.02.010. URL: https://www.sciencedirect.com/science/article/abs/pii/S0164121213001039.

[(2)] Rory Burke. *Project Management: Planning and Control Techniques*. Chichester, UK: John Wiley & Sons, 2013.

[(3)] Mike Cohn. *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice Hall, 2004.

[(4)] Deniso700. *Optimización Matemática de la Planificación de Sprints en Equipos Scrum*. https://github.com/DENISO700/Optimizacion-matematica-de-la-planificacion-de-sprints-en-equipos-Scrum.git. Repositorio de GitHub. 2025.

[(5)] Frederick S. Hillier y Gerald J. Lieberman. *Introduction to Operations Research*. 11th. New York, NY: McGraw-Hill Education, 2021.

[(6)] David Martínez, José Pardo y Carlos Roldán. «Improvements for the Planning Process in the Scrum Method». En: *Applied Sciences* 15.1 (2025), pág. 202. DOI: 10.3390/app15010202. URL: https://www.mdpi.com/2076-3417/15/1/202.

[(7)] Ken Schwaber y Jeff Sutherland. *The Scrum Guide*. Scrum.org, 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.

[(8)] Hamdy A. Taha. *Operations Research: An Introduction*. 10th. Boston, MA: Pearson, 2017.

[(9)] Wayne L. Winston. *Operations Research: Applications and Algorithms*. 5th. Boston, MA: Cengage Learning, 2022.