



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CONFORMACIÓN DE EQUIPOS DE SOFTWARE CON PERSPECTIVA DE GÉNERO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

JOAQUÍN EMILIO OPORTUS FOSTER

PROFESORA GUÍA:
MARÍA CECILIA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:
JOCELYN SIMMONDS WAGEMANN
DANIEL PEROVICH GEROSA

SANTIAGO DE CHILE
2024

Resumen

En el curso de Proyecto de Software de la Universidad de Chile, destinado a simular una experiencia de desarrollo de software real, la conformación de equipos ha sido históricamente un desafío. Anteriormente, este proceso se realizaba manualmente por el docente, lo cual era viable cuando el número de estudiantes era menor. Sin embargo, el crecimiento en la inscripción al curso complicó esta tarea, haciendo necesario un sistema más eficiente. En respuesta, se diseñó una aplicación web para automatizar y facilitar este proceso, basándose inicialmente en la disponibilidad horaria de los estudiantes. La esencia de esta primera solución, aunque innovadora, tenía limitaciones en términos de flexibilidad y no consideraba aspectos como la diversidad de género en los equipos.

Este proyecto se centró en extender la herramienta original para formar equipos, haciéndola más flexible y capaz de adaptarse a diferentes necesidades. Con un nuevo enfoque, esta versión mejorada facilita combinar aspectos importantes como los horarios de los estudiantes y una distribución equilibrada de género en los grupos. Esta actualización no solo conservó la eficacia de la formación de equipos, sino que en muchos casos la mejoró en comparación con la solución anterior.

Para evaluar la eficacia del nuevo sistema, que ahora incorpora el género como una variable, se realizaron pruebas utilizando datos de semestres previos. Estos datos son los mismos que se emplearon para evaluar la herramienta anterior. Se compararon los resultados de evaluación de los equipos formados. Los hallazgos indican que la inclusión de esta restricción adicional no afectó negativamente la calidad de los resultados. Por el contrario, se observó una mejora en las soluciones, reflejada en una conformación de equipos más eficiente.

Este trabajo aspira no solo a satisfacer la necesidad de una formación de equipos más eficiente y equitativa, sino también a establecer un fundamento sólido para nuevas mejoras en la herramienta. Así se podría llegar a adaptar a diversos contextos y, potencialmente, a diferentes cursos en el futuro, ampliando su aplicabilidad e impacto.

Tabla de Contenido

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	2
1.3. Solución Actual	2
1.4. Limitaciones de la Solución Actual	3
1.5. Objetivos	3
1.5.1. Objetivo General	3
1.5.2. Objetivos Específicos	4
2. Estado del Arte	5
2.1. Algoritmos de asignación óptima	5
2.2. Programación con restricciones	5
2.3. Algoritmos genéticos de optimización	6
2.4. Estudios contemporáneos de los efectos de diversidad de género en el trabajo en equipo	6
2.5. Consideraciones éticas a tomar en cuenta al desarrollar un algoritmo de conformación de equipos con perspectiva de género	7
2.6. Herramienta de compatibilidad horaria	7
3. Solución	9
3.1. Modificación del Modelo de Base de Datos	9
3.1.1. Modelo Original	9
3.1.2. Modelo Modificado	9

3.1.3.	Análisis de los Cambios	10
3.2.	Interfaz - Modularización del Formulario del usuario alumno	10
3.2.1.	El formulario ahora incluye género	10
3.3.	Algoritmo original de optimización de equipos diseñado por Javier Lavados[8]	12
3.3.1.	Descripción de funciones que componen al algoritmo de optimización	13
3.3.2.	Limitaciones de este Algoritmo	19
3.4.	Refactorización y Modularización del Sistema de Optimización	20
3.4.1.	Módulo de Restricciones	20
3.4.2.	Diagrama UML - Componente de Restricciones	22
3.5.	Abstracción de Estrategias de Evaluación de Equipos: Componente de Armonía Local	22
3.5.1.	Estrategias de Evaluación de Armonía	22
3.5.2.	Diagrama UML - Componente de Armonía	24
3.5.3.	Módulo de Optimización	25
3.5.4.	Unificación de Módulos Independientes - Clase <code>TeamOptimizer</code>	28
4.	Evaluación	31
4.1.	Resultados y análisis	34
4.1.1.	Herramienta Anterior vs. Algoritmo Genético usando únicamente Restricción de Horario	34
4.1.2.	Herramienta Anterior vs. Algoritmo Genético con Restricciones de Horario y Género	35
4.1.3.	Evaluación del Cumplimiento de Restricciones y Flexibilidad	36
4.2.	Discusión	43
4.2.1.	Mantenimiento de la Armonía en Equipos Conformados	43
4.2.2.	Importancia de la Extensibilidad y la Flexibilidad del Sistema	43
4.2.3.	Análisis de Cumplimiento de Restricciones en la Formación de Equipos	43
4.2.4.	Limitaciones	44
5.	Conclusión	45

Índice de Ilustraciones

3.1. Imagen de nuevo formulario para usuario alumno, incorporando género	11
3.2. Imagen de nueva vista de estado de proceso, ahora se guarda el género de los alumnos	12
3.3. Diagrama UML correspondiente a este módulo, mostrando la relación entre <code>Restriction</code> , sus clases derivadas y el <code>TeamRestrictionManager</code>	22
3.4. Diagrama UML dedicada a la armonía local, mostrando la relación entre <code>HarmonyStrategy</code> , su clase derivada y el <code>HarmonyCalculator</code>	24
3.5. Diagrama UML correspondiente a este módulo, mostrando la relación entre <code>OptimizationAlgorithm</code> , <code>GeneticAlgorithm</code> , y las funciones de evaluación y mutación.	28
3.6. Diagrama UML completo. Este diagrama abarca <code>TeamOptimizer</code> , junto con sus interacciones con los otros módulos y las clases relacionadas con estos. . .	30

Capítulo 1

Introducción

1.1. Contexto

El contexto de este proyecto se sitúa en el ámbito académico de la Ingeniería Civil en Computación en la Universidad de Chile. En este entorno, la formación de equipos de trabajo para proyectos es una actividad recurrente y crucial, particularmente en cursos como Proyecto de Software. Un estudiante del departamento de ciencias de la computación de la Universidad de Chile creó, hace aproximadamente un año, una herramienta de optimización de conformación de equipos que tiene en cuenta la disponibilidad horaria de los alumnos [8]., este software está enfocado en la optimización de equipos basándose principalmente en la disponibilidad horaria de los estudiantes. Si bien esta aproximación ha sido buena y funcional, ha dejado de lado aspectos significativos que pueden influir en la dinámica y eficacia de los equipos.

Uno de estos aspectos significativos a considerar en la conformación de equipos es la distribución de género. Se ha observado que los equipos con distribuciones poco balanceadas en cuanto a género experimentan peor satisfacción y generan peores resultados desde el punto de vista técnico. Algunos casos especialmente perjudiciales son aquellos grupos que no incluyen mujeres o que incluyen solo una [3]. Por lo tanto, es crucial optimizar la conformación de equipos teniendo en cuenta también la distribución de género para mejorar la compatibilidad de los equipos y, en última instancia, el bienestar y el aprendizaje de los estudiantes.

La Universidad de Chile, siendo una institución líder en educación superior, se enfrenta al reto constante de actualizar y mejorar sus métodos y herramientas para mantenerse a la vanguardia en la formación de sus estudiantes. Este proyecto se inserta en ese marco, buscando no solo mejorar un aspecto técnico de un curso en particular, sino también contribuir al desarrollo de prácticas más inclusivas y representativas en el ámbito educativo.

Con este contexto en mente, el proyecto se posiciona como una iniciativa oportuna y necesaria, buscando abordar las limitaciones de la herramienta existente y ampliar sus capacidades para beneficiar a una comunidad académica más amplia.

1.2. Motivación

La motivación para este trabajo surge de las limitaciones inherentes al diseño original de la herramienta desarrollada por Javier Lavados[8]. Aunque eficaz en su contexto, no permitía la integración de género como un criterio en la formación de equipos. Este desafío destacó la necesidad de una arquitectura más flexible, capaz de soportar aún otras modificaciones y extensiones futuras. El rediseño buscó no solo poder agregar más restricciones a la formación de equipos, sino también reestructurar la herramienta para hacerla más adaptable. Dada la complejidad de los problemas de optimización NP-difíciles y la rigidez del diseño inicial, era esencial repensar y reconstruir la herramienta para facilitar mejoras y adaptaciones futuras.

Un aspecto crucial de la relevancia de este problema es la visión del departamento de expandir y mejorar continuamente sus herramientas educativas. Existe un interés creciente por parte de profesores de otros cursos, ajenos a Proyecto de Software, en la utilización de esta herramienta. Además, hay planes para futuros proyectos de título que continuarán extendiendo y mejorando esta herramienta, enfatizando su importancia como un activo en desarrollo continuo. La implementación de esta herramienta en otros cursos requeriría ajustes tanto en los formularios utilizados para recopilar datos de los alumnos como en la heurística de optimización, incluyendo la función objetivo y las restricciones. Es crucial que la arquitectura de aplicación permita estos cambios futuros sin necesidad de una extensa refactorización.

Considerando lo anterior, la expectativa a largo plazo es que esta herramienta se convierta en una solución robusta y versátil, aplicable a una variedad de contextos educativos dentro del departamento y más allá. Por tanto, este proyecto no solo responde a una necesidad inmediata de mejorar la formación de equipos en un curso específico, sino que también contribuye a un objetivo más grande: desarrollar una herramienta que se pueda adaptar y expandir para satisfacer una variedad de necesidades educativas en el futuro.

1.3. Solución Actual

La solución actual desarrollada por Javier Lavados para la conformación de equipos en el curso *Proyecto de Software* es una aplicación web integral que automatiza y mejora significativamente el proceso. Los elementos clave de esta solución son:

Interfaz de Usuario Intuitiva: La aplicación ofrece interfaces claras y fáciles de usar tanto para los alumnos como para los docentes. Los alumnos pueden responder y modificar sus encuestas de disponibilidad horaria, mientras que los docentes tienen un control total sobre el proceso de conformación de equipos.

Automatización del Proceso: La aplicación automatiza el proceso desde la recopilación de las respuestas de disponibilidad horaria de los alumnos hasta la generación automática de equipos basada en esta información. Los docentes tienen la capacidad de realizar ajustes manuales en la conformación generada por el sistema.

Algoritmos de Conformación de Equipos: El sistema utiliza algoritmos probabilísticos para crear equipos balanceados. Inicialmente se conforman equipos válidos utilizando una

heurística que garantiza que se cumplan todas las condiciones del curso. Posteriormente, se optimiza esta conformación utilizando algoritmos de armonía local (AL) y armonía global (AG) para asegurar la equidad entre los equipos. [8]

1.4. Limitaciones de la Solución Actual

A pesar de las mejoras significativas, la solución actual, desarrollada por Javier Lavados, enfrenta importantes limitaciones:

- **Modularidad y Extensibilidad Reducidas:** El diseño del software, enfocado en la disponibilidad horaria, carece de la flexibilidad necesaria para integrar nuevas restricciones, como la inclusión del género. Esto limita severamente la capacidad de adaptar y expandir el sistema para abordar diferentes criterios de formación de equipos.

- **Modelo de Datos y Código de Interfaces Poco Generalizables:** El modelo de datos y el código de las interfaces no están preparados para ajustarse fácilmente a nuevas variables o criterios. La falta de generalidad en estos elementos del software dificulta su uso en contextos variados, más allá de la optimización horaria.

- **Restricciones en la Optimización de Equipos:** Los algoritmos de optimización actuales están altamente especializados para manejar la disponibilidad horaria, pero no ofrecen la flexibilidad necesaria para incorporar otros factores esenciales como el género. Esto es porque la validación no es independiente a la lógica de optimización, lo que hace imposible añadir restricciones sin hacer grandes modificaciones a la heurística existente.

- **Necesidad de Mejoras en Algoritmos de Optimización:** Para poder añadir más restricciones y además facilitar la extensión del software en el futuro, es necesario identificar y separar cada uno de las piezas independientes y necesarias dentro del actual proceso de optimización y conformación de equipos

1.5. Objetivos

1.5.1. Objetivo General

El objetivo de este trabajo es extender la herramienta actual de optimización de conformación de equipos del Departamento de Ciencias de la Computación de la Universidad de Chile, con un enfoque en la extensibilidad y adaptabilidad a largo plazo. Esto incluye la refactorización del algoritmo de conformación de equipos para abarcar una mayor diversidad de factores. Hacer esto nos permitirá, entre otras cosas, la posibilidad de incluir restricciones de género al algoritmo.

1.5.2. Objetivos Específicos

Para lograr el objetivo general, se han establecido los siguientes objetivos específicos:

1. **Modificación de la Encuesta para Recolección de Datos de Género y Horario:** Adaptar la encuesta existente para que pueda recopilar información tanto de la disponibilidad horaria como del género de los estudiantes, permitiendo que la herramienta maneje estas dos dimensiones esenciales en la formación de equipos.
2. **Extensiones al Modelo de Datos para Gestión de Nueva Información:** Ampliar y adaptar el modelo de datos de la herramienta para almacenar y utilizar eficazmente la información recopilada sobre género y disponibilidad horaria. Este paso es fundamental para garantizar que la herramienta pueda procesar y utilizar estas nuevas variables en la formación de equipos.
3. **Rediseño y Modularización del Algoritmo de Optimización y Formación de Equipos:** Reconstruir y mejorar el algoritmo de optimización y formación de equipos para que sea capaz de soportar nuevas restricciones, como las de género y disponibilidad. De igual manera, se modulariza el sistema para facilitar futuras extensiones y adaptaciones a otros algoritmos de optimización y funciones objetivo. Esto permitirá que la herramienta se mantenga relevante y útil en diversos contextos y con diferentes requisitos en el futuro.

Capítulo 2

Estado del Arte

2.1. Algoritmos de asignación óptima

Los algoritmos de asignación óptima son métodos computacionales diseñados para resolver problemas de asignación, donde el objetivo es distribuir un conjunto de recursos o tareas entre agentes o destinos de manera que se optimice un criterio específico, como minimizar el costo total o maximizar la eficiencia. Estos algoritmos consideran restricciones como la capacidad de los agentes y la unicidad de la asignación de tareas, asegurando que cada recurso se asigne de la forma más efectiva posible. A través de técnicas como la programación lineal, métodos heurísticos, y algoritmos de ramificación y poda (branch-and-bound), se busca la solución óptima dentro de un espacio de posibilidades, equilibrando la precisión con la eficiencia computacional. Este enfoque encuentra aplicaciones en logística, planificación de producción, y la asignación de personal, entre otros campos. Para un análisis detallado de estos algoritmos y sus aplicaciones, *A survey of algorithms for the generalized assignment problem*[4] ofrece una perspectiva exhaustiva. En las siguientes secciones veremos las técnicas más relevantes para problemas de asignación óptima.

2.2. Programación con restricciones

La programación con restricciones es un paradigma poderoso para resolver problemas de búsqueda combinatoria, utilizando una amplia gama de técnicas de inteligencia artificial, investigación operativa, algoritmos, y teoría de grafos. La idea central es que el usuario especifica las restricciones, y un solucionador de restricciones general se encarga de resolverlas. Estas restricciones son relaciones entre variables de decisión, y el objetivo es encontrar asignaciones a estas variables que satisfagan todas las restricciones dadas. Este enfoque permite modelar y resolver eficientemente problemas complejos en diversos campos, ofreciendo soluciones óptimas bajo criterios de optimización específicos o encontrando todas las soluciones posibles a un problema dado.[11] Muchas veces, encontrar todas las soluciones posibles no es algo factible. En estos casos podemos recurrir al uso de programación con restricciones para encontrar un espacio de soluciones reducido y más abordable.

2.3. Algoritmos genéticos de optimización

Los algoritmos genéticos de optimización son técnicas computacionales inspiradas en el proceso de evolución natural, que se utilizan para resolver problemas complejos de búsqueda y optimización. Estos algoritmos operan mediante la generación y evolución de poblaciones de soluciones candidatas, utilizando operadores genéticos como selección, cruzamiento (o recombinación), y mutación. La eficacia de los algoritmos genéticos radica en su capacidad para explorar eficientemente espacios de búsqueda grandes y complejos, encontrando soluciones óptimas o cercanas a óptimas a través de procesos iterativos basados en principios de selección natural y genética.[12]

2.4. Estudios contemporáneos de los efectos de diversidad de género en el trabajo en equipo

Ciertamente, numerosos estudios han intentado analizar el impacto que tiene la diversidad de un grupo y, en específico, la diversidad de género en el rendimiento de un equipo de trabajo. *Diversity, effort, and cooperation in team-based learning* [5] es una investigación que evalúa el desempeño de grupos de estudiantes universitarios en un contexto de teoría macroeconómica. Encontró que la participación femenina beneficia al equipo. Los grupos altamente balanceados en términos de género (con distribuciones cercanas al 50 por ciento) propician una mejor colaboración, compromiso y éxito individual de sus miembros.

Por su parte, *Impact of Personality and Gender Diversity on Software Development Teams Performance* [6], determinó que la diversidad de género y personalidad influye en el rendimiento de los equipos de desarrollo de software. Encontró que los grupos liderados por hombres son más exitosos cuando incluyen mujeres. Contrariamente, los equipos encabezados por mujeres son más exitosos cuando son exclusivamente femeninos. Los resultados también señalan que existen tipos de personalidad más comunes en hombres o mujeres y que diferentes proporciones de género en un equipo pueden conducir a distintos resultados dependiendo de la personalidad de los integrantes.

El estudio *Joint Effects of Group Efficacy and Gender Diversity on Group Cohesion and Performance* [9], examinó la eficiencia de trabajo en equipo y la diversidad de género. Además, analizó cómo este último factor impacta los resultados del grupo. Se determinó que la diversidad de género contribuye positivamente a la cohesión y calificación final del conjunto.

Finalmente, *Perceptions on Diversity in Brazilian Agile Software Development Teams: A Survey* [7] es un análisis sobre la diversidad de género en equipos de desarrollo de software en Brasil y la percepción de dicha diversidad y su impacto en las actividades del grupo. Para evaluar la percepción sobre esta variable, se realizó una encuesta a una muestra equilibrada en términos de género, que incluyó a 173 participantes. Algunos resultados muestran que el 78,8 por ciento de los encuestados estuvo de acuerdo o muy de acuerdo en que la diversidad es crucial. En este caso, las mujeres tienden a concordar más que los hombres sobre su importancia. El 77,6 por ciento de los participantes calificó la experiencia de trabajar en

un equipo diverso como positiva. En general, la encuesta evidenció que los participantes consideran mayoritariamente que la diversidad es un factor positivo en los trabajos en equipo, especialmente en el contexto de desarrollo de software.

2.5. Consideraciones éticas a tomar en cuenta al desarrollar un algoritmo de conformación de equipos con perspectiva de género

El desarrollo y la implementación de algoritmos de conformación de equipos que consideren aspectos de género y diversidad plantean cuestiones éticas importantes. Es crucial garantizar que el proceso de conformación de equipos no discrimine ni favorezca a ningún individuo o grupo en función de su género. Se deben tener en cuenta los principios éticos de equidad, justicia y respeto a la autonomía de los participantes [1].

Para abordar estas consideraciones éticas, es importante tener en cuenta los siguientes aspectos:

- **Transparencia:** El algoritmo de conformación de equipos y los criterios utilizados para considerar el género deben ser transparentes y comunicados claramente a todos los participantes. Los estudiantes deben comprender cómo se utiliza el género como variable en el proceso de conformación de equipos y cómo esto puede afectar la distribución de los equipos.
- **Consentimiento informado:** Los estudiantes deben estar al tanto del enfoque de género del algoritmo de conformación de equipos. De esta forma los estudiantes tienen la oportunidad de proporcionar su consentimiento y participar en este proceso. Se debe respetar la autonomía de los estudiantes y asegurarse de que entiendan plenamente el funcionamiento del algoritmo.
- **Privacidad y confidencialidad:** La información de género que proporcionen los estudiantes debe ser manejada con el cuidado adecuado y con completa confidencialidad, garantizando la protección de los datos personales de los estudiantes. Para este caso, se deberá solo utilizar la información de género proporcionada por los estudiantes. Ninguna inferencia se derivará de esta basándose en otra información registrada por la universidad.

2.6. Herramienta de compatibilidad horaria

Actualmente, existe una aplicación web desarrollada en Django, que optimiza la formación de equipos de software según la disponibilidad horaria de los miembros [8]. Esta herramienta busca mejorar la eficiencia y la calidad de los resultados en la conformación de equipos.

La disponibilidad horaria de los estudiantes es un factor crucial en la conformación de

equipos, ya que es necesario que los miembros del equipo puedan coordinar sus horarios para trabajar juntos de manera efectiva. La aplicación web de optimización por compatibilidad horaria recopila información de disponibilidad horaria de los estudiantes con el objetivo de generar equipos de trabajo equilibrados. En este contexto, *equilibrados* significa que se busca que ninguno de los equipos del curso tenga una ventaja sobre otro. Esta equidad se mide a través de una buena compatibilidad horaria, es decir, asegurando que los miembros de cada equipo tengan bloques de horario disponibles en común. La optimización se lleva a cabo mediante un enfoque que considera tanto la armonía local como global entre los estudiantes.

La armonía local se refiere a la medida en la que la disponibilidad horaria de los estudiantes dentro de un equipo en particular se superponen. Cuanto mayor sea la coincidencia existente entre la disponibilidad horaria de los alumnos de un equipo, se dice que ese equipo tiene una mayor armonía local. Por otro lado, la armonía global se enfoca en minimizar la variabilidad de la armonía local de todos los equipos. Es importante que todos los alumnos tengan la misma posibilidad de tener éxito.

Aunque la optimización por horarios es un gran avance, se ha planteado que la distribución de género, un factor importante en la conformación de equipos, también puede influir en su probabilidad de éxito. En este sentido, se sugiere que un grupo balanceado con respecto al género, es decir, con una proporción equitativa de géneros y disidencias, tiene mejores resultados que uno no proporcional. Los casos extremos de desbalance, como la presencia de solo una mujer en un grupo de más de tres personas, se consideran particularmente desfavorables. Por lo tanto, es crucial optimizar la conformación de equipos teniendo en cuenta la distribución de género para mejorar la compatibilidad de los equipos y, en última instancia, el bienestar y el aprendizaje de los estudiantes. [3] [5] [6] [7] [9]

Al incorporar la distribución de género en la conformación de equipos mediante la extensión de la herramienta actual de optimización por horarios, se espera lograr una mayor efectividad en la formación de equipos, lo que se traduciría en una mejor experiencia y en la obtención de mejores resultados por parte de los equipos de trabajo.

Si bien la herramienta actual ha demostrado ser eficaz en términos de armonía horaria, su diseño y estructura no ofrecen la flexibilidad necesaria para integrar aspectos adicionales como la equidad de género. Esta carencia subraya la relevancia de desarrollar una solución más versátil. En respuesta a esta limitación, surge la necesidad de rediseñar la herramienta para que sea capaz de manejar múltiples criterios de formación de equipos. La extensión de la herramienta existente para incluir factores como la distribución de género, también preparará el terreno para futuras adaptaciones y mejoras en el ámbito de la optimización de equipos.

Capítulo 3

Solución

3.1. Modificación del Modelo de Base de Datos

El modelo original y el modelo modificado representan dos enfoques distintos para estructurar los datos en una aplicación Django. Vamos a analizar ambos modelos, destacando sus diferencias y los beneficios del nuevo modelo.

3.1.1. Modelo Original

El modelo original consta de varias clases: `Process`, `Team`, `Profile`, y `Survey`. Cada clase representa una entidad diferente en la base de datos. La clase `Survey`, en particular, se utiliza para almacenar las respuestas de la encuesta y tiene múltiples campos booleanos (`m_am`, `tu_am`, etc.) para representar la disponibilidad de los estudiantes. Donde `m_am` se refiere a la mañana del lunes, `tu_pm` a la tarde del martes y así sucesivamente.

Ejemplo del código original:

```
class Survey(models.Model):
    m_am = models.BooleanField(default=False)
    ...
```

3.1.2. Modelo Modificado

El modelo modificado introduce cambios significativos, especialmente en la estructura de la encuesta. Se agregan dos nuevas clases: `ScheduleSurvey` y `GenderSurvey`, y la clase `Survey` se modifica para incluir referencias a estas nuevas clases mediante relaciones uno a uno (`OneToOneField`).

Ejemplo del código modificado:

```

class ScheduleSurvey(models.Model):
    m_am = models.BooleanField(default=False)
    ...

class GenderSurvey(models.Model):
    gender = models.CharField(...)
    ...

class Survey(models.Model):
    schedule_survey = models.OneToOneField(ScheduleSurvey, ...)
    gender_survey = models.OneToOneField(GenderSurvey, ...)

```

3.1.3. Análisis de los Cambios

- **Mejora en el Manejo de Datos:** Con la modularización, la gestión de datos se vuelve más eficiente. Por ejemplo, si se necesita acceder solo a los datos de género para un análisis, se puede consultar directamente la clase `GenderSurvey` sin tener que cargar toda la información de `Survey`.
- **Adaptabilidad a Cambios en los Requisitos:** Si en el futuro los requisitos para las encuestas cambian (por ejemplo, cambiar la forma en que se recopilan las preferencias de horario), estos cambios se pueden gestionar más fácilmente en un módulo específico sin necesidad de reestructurar toda la clase `Survey`.

3.2. Interfaz - Modularización del Formulario del usuario alumno

3.2.1. El formulario ahora incluye género

El rediseño del formulario para los alumnos se centró en la mejora de la extensibilidad y facilitar un poco la creación de nuevos formularios. El cambio principal fue la creación de un componente dedicado para la sección de género, `GenderSurvey.html`, que se sumó a la estructura existente del formulario de disponibilidad horaria, la que por su parte se encapsuló el formulario de disponibilidad en un nuevo `AvailabilitySurvey.html`. Esta modularización permite que el formulario principal, `Survey.html`, ya no contenga todos los elementos fijos, sino que incluya estos componentes de manera más flexible. Esto facilita la adición futura de nuevos atributos al formulario.

Servicios | Conformación de equipos de proyecto de software | Tomas Lucas Ampuero Camacho

Encuesta de disponibilidad horaria

Debe marcar al menos cuatro casillas que corresponden a bloques de tiempo de aproximadamente cuatro horas cada una.

	Lunes	Martes	Miércoles	Jueves	Viernes
Mañana	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tarde	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Género


Hombre

Mujer

Otro

Prefiero no decirlo

Cancelar
Guardar



Área de Desarrollo de Software
 Departamento de Ciencias de la Computación
 Universidad de Chile
 © 2023

Figura 3.1: Imagen de nuevo formulario para usuario alumno, incorporando género

En la encuesta, se observan casillas que los usuarios pueden marcar para indicar su disponibilidad durante las mañanas y tardes de los días laborables, de lunes a viernes. Además, hay una sección dedicada a la identificación de género con varias opciones: Hombre, Mujer, Otro, Prefiero no decirlo y No binario. Esta sección de género es una extensión a la funcionalidad original de la encuesta y es la que se define dentro de `GenderSurvey.html`.

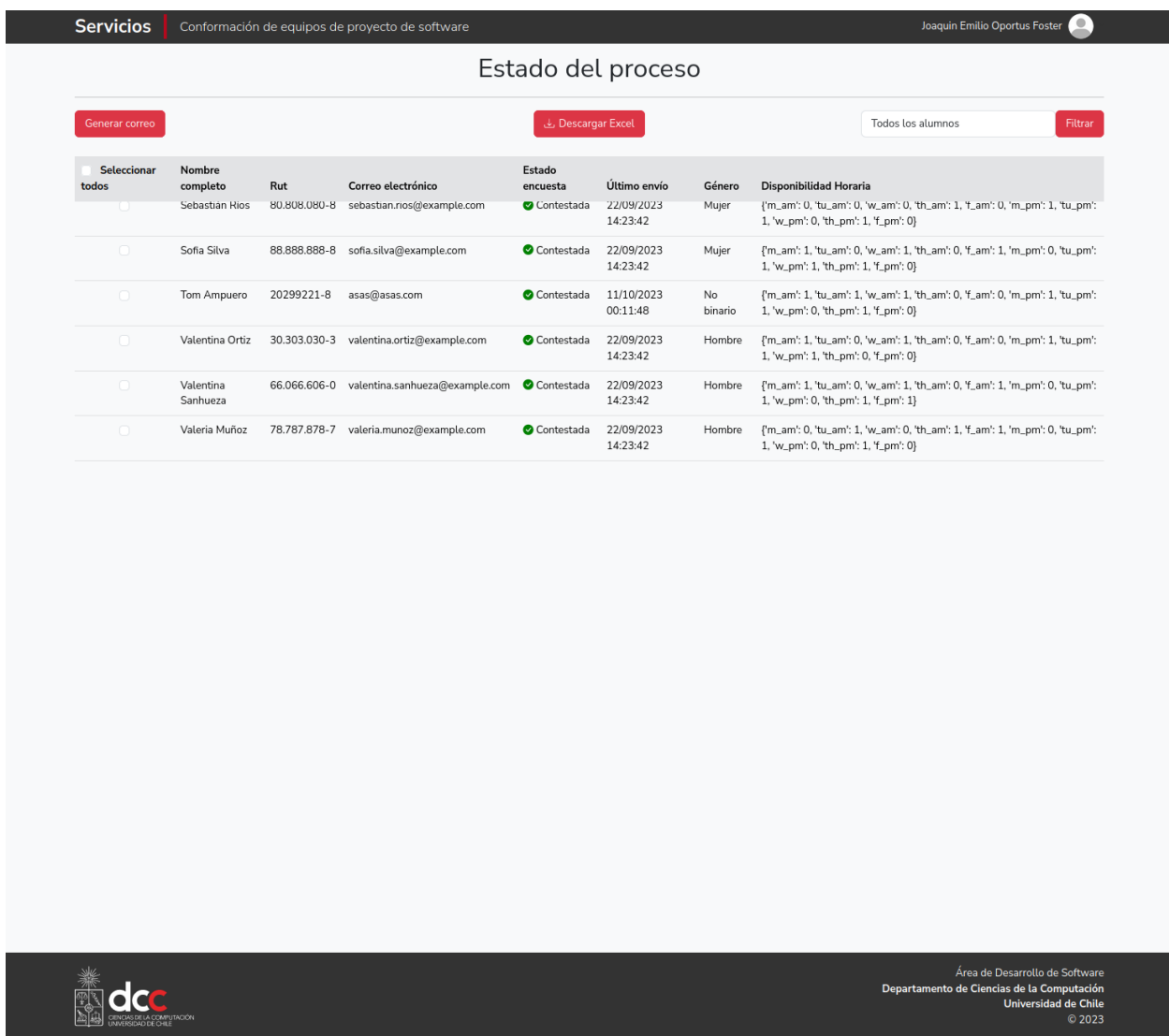


Figura 3.2: Imagen de nueva vista de estado de proceso, ahora se guarda el género de los alumnos

La imagen del *Estado del proceso* de formación de equipos para proyectos de software muestra una tabla. Esta lista a los estudiantes e incluye columnas para *Nombre completo*, *Rut*, *Correo electrónico*, *Estado de la encuesta*, *Último envío*, *Disponibilidad Horaria* y la nueva variable *Género*, demostrando que se registra y visualiza correctamente en la interfaz.

3.3. Algoritmo original de optimización de equipos diseñado por Javier Lavados[8]

Antes de presentar la nueva solución de optimización, es esencial comprender la metodología previamente implementada por Javier Lavados. Al analizar meticulosamente la solución antecedente, podemos identificar sus limitaciones, lo que nos permitirá justificar la necesi-

dad de una nueva estrategia de diseño. La arquitectura del algoritmo de optimización de formación de equipos de Javier se encapsulaba dentro de `algorithms.py`, un módulo que comprende diversas funciones encargadas de la conformación de equipos, la redistribución de alumnos, así como el cálculo de la armonía a niveles local y global, entre otras. Es imperativo revisar cada una de estas funciones detalladamente para comprender su funcionamiento individual. Esta revisión minuciosa del algoritmo previo establecerá una base firme para la posterior introducción de la nueva solución.

3.3.1. Descripción de funciones que componen al algoritmo de optimización

Cálculo de la Armonía Local de un Equipo

Descripción: La función `calculateLocalHarmony` se encarga de evaluar la armonía local de un equipo basándose en la disponibilidad de los cursos para cada miembro del equipo. Se enfoca en dos aspectos principales: la proporción de miembros disponibles en cada bloque de tiempo y la proporción de interacciones potenciales entre los miembros, para estimar qué tan bien un equipo puede trabajar junta dada su disponibilidad.

Parámetros:

- `team`: Lista de miembros del equipo.
- `availabilityCourse`: Diccionario que indica la disponibilidad de los cursos para cada miembro del equipo.

Pseudocódigo:

1. Determinar el número total de miembros en el equipo.
2. Si el equipo tiene uno o ningún miembro, retornar 0 (indicando ninguna armonía).
3. Generar una matriz inicial que refleje el equipo y su disponibilidad para los cursos.
4. Calcular el primer parámetro de armonía como el porcentaje de miembros del equipo que están presentes en cada bloque de disponibilidad.
5. Calcular el segundo parámetro de armonía, representando el porcentaje de interacciones potenciales entre miembros, ajustado por el número total de miembros en el equipo.
6. Combinar ambos parámetros de armonía con igual ponderación para obtener el valor de la armonía local del equipo.
7. Retornar el valor calculado de armonía local, reflejando la capacidad del equipo para trabajar de manera cohesiva basándose en su disponibilidad.

Agrupación de Estudiantes por Disponibilidad

Descripción: La función `inicialMatrix` genera un diccionario que organiza a los estudiantes de un curso según su disponibilidad por días y horas, facilitando así la formación de equipos de trabajo al identificar los momentos en los que grupos de estudiantes están disponibles simultáneamente.

Parámetros:

- `students`: Lista de estudiantes participantes en el curso.
- `availabilityCouse`: Diccionario que mapea cada estudiante a su disponibilidad.

Pseudocódigo:

1. Inicializar un diccionario vacío `availabilityByDay`.
2. Para cada `student` en `students`, realizar lo siguiente:
 - (a) Si `student` está en `availabilityCouse`, obtener su disponibilidad.
 - (b) Para cada día de disponibilidad de `student`, agregar `student` a la lista correspondiente en `availabilityByDay`. Si el día no existe en `availabilityByDay`, crear una nueva entrada con el `student` como primer elemento de la lista.
3. Devolver el diccionario `availabilityByDay` que contiene la disponibilidad de los estudiantes organizada por días.

Verificación de Disponibilidad del Equipo

Descripción: La función `checkTeam` evalúa si los miembros de un equipo tienen la disponibilidad requerida para participar de manera efectiva en el equipo, definida como estar disponible en al menos cuatro bloques de tiempo compartidos. Los miembros que no cumplan con este criterio son identificados para su posible reemplazo.

Parámetros:

- `students`: Lista de estudiantes en el equipo.
- `availibity_team`: Diccionario que mapea los días a los estudiantes disponibles en esos días.

Pseudocódigo:

1. Si el equipo está compuesto por un solo estudiante, retorna una lista vacía (no se requiere verificación).

2. Inicializar una lista `failedAttendance` para almacenar los estudiantes con disponibilidad insuficiente.
3. Para cada `student` en `students`:
 - (a) Contar en cuántos bloques de tiempo `student` está disponible junto con al menos otro miembro del equipo.
 - (b) Si el `student` está disponible en menos de 4 bloques de tiempo, agregarlo a `failedAttendance`.
4. Remover de `students` a aquellos que están en `failedAttendance`.
5. Retornar la lista `failedAttendance` para identificar a los estudiantes que necesitan reemplazo o ajuste en su disponibilidad.

Selección Aleatoria de Estudiantes

Descripción: La función `pickRandomStudents` selecciona aleatoriamente un número especificado de estudiantes de una lista de candidatos disponibles, asegurando que la selección cumpla con el requisito de cantidad deseada, si es posible.

Parámetros:

- `studentsOptions`: Lista de estudiantes disponibles para ser elegidos.
- `requiredStudents`: Número de estudiantes requeridos a seleccionar.

Pseudocódigo:

1. Si `requiredStudents` es mayor que la longitud de `studentsOptions`, ajustar `requiredStudents` a la longitud de `studentsOptions` para manejar el caso de no tener suficientes estudiantes disponibles.
2. Inicializar una lista vacía `selectedStudents` para almacenar los estudiantes seleccionados.
3. Mientras la longitud de `selectedStudents` sea menor que `requiredStudents`:
 - (a) Seleccionar aleatoriamente un estudiante de `studentsOptions`.
 - (b) Agregar el estudiante seleccionado a `selectedStudents`.
 - (c) Eliminar el estudiante seleccionado de `studentsOptions` para prevenir selecciones duplicadas.
4. Retornar `selectedStudents`.

Optimización de Equipos mediante Intercambio de Miembros

Descripción: La función `trySwitch` examina la posibilidad de intercambiar un miembro entre dos equipos para optimizar su composición, basándose en la disponibilidad de los miembros. Se busca el intercambio que mejore la disponibilidad global de ambos equipos, considerando la incorporación de un nuevo miembro en un equipo y el traslado del mejor miembro actual a otro equipo.

Parámetros:

- `newTeam`: Equipo que podría recibir un nuevo miembro.
- `oldTeam`: Equipo del cual proviene el nuevo miembro potencial.
- `newmember`: Miembro propuesto para ser añadido al nuevo equipo.
- `availability`: Diccionario que mapea a cada miembro con sus disponibilidades.

Pseudocódigo:

1. Identificar al mejor miembro del `newTeam` en términos de disponibilidad.
2. Crear una versión de prueba del `newTeam` sustituyendo al mejor miembro actual por `newmember`.
3. Evaluar la disponibilidad total del `newTeam` con el cambio propuesto para determinar si mejora la viabilidad del equipo.
4. Repetir el proceso evaluando el impacto de añadir al mejor miembro del `newTeam` al `oldTeam`, para verificar la mejora en su composición.
5. Si el intercambio propuesto mejora la disponibilidad en ambos equipos sin causar la eliminación de miembros, proceder con el intercambio y devolver el nombre del miembro intercambiado; de lo contrario, mantener al `newmember` en el `oldTeam`.

Formación de Equipos Basada en Disponibilidad

Descripción: La función `createTeams` está diseñada para organizar a los estudiantes en equipos, teniendo en cuenta su disponibilidad. Se enfoca en formar equipos del tamaño más equitativo posible, mientras asegura que todos los miembros tengan oportunidades significativas de colaboración. La meta es maximizar la compatibilidad de los horarios de disponibilidad entre los miembros del equipo.

Parámetros:

- `amountStudents`: Número total de estudiantes a ser asignados a equipos.
- `amountTeams`: Número total de equipos a formar.

- **availability**: Diccionario que mapea a cada estudiante con sus tiempos de disponibilidad.

Pseudocódigo:

1. Calcular el tamaño ideal de cada equipo dividiendo la cantidad total de estudiantes por el número de equipos.
2. Crear una lista con todos los estudiantes disponibles para ser asignados a equipos.
3. Para cada equipo a formar:
 - (a) Ajustar el tamaño del equipo según el número de estudiantes restantes para asegurar una distribución equitativa.
 - (b) Seleccionar aleatoriamente los miembros del equipo de la lista de estudiantes disponibles, respetando el número requerido de miembros.
 - (c) Verificar que los miembros seleccionados cumplan con los criterios de disponibilidad mínima para una participación efectiva en el equipo.
 - (d) Realizar ajustes necesarios, como reasignar estudiantes para mejorar la compatibilidad del equipo o cumplir con los requisitos de tamaño.
4. Repetir el proceso hasta que todos los estudiantes hayan sido asignados a equipos, asegurando que cada equipo formado cumpla con las reglas de compatibilidad y tamaño.

Equilibrio de Armonía Local entre Equipos

Descripción: La función `balancear` tiene como objetivo optimizar la distribución de miembros entre diferentes equipos para lograr un balance más equitativo en términos de armonía local. Esto se realiza identificando equipos con la mayor y menor armonía local y tratando de intercambiar miembros entre ellos para mejorar la equidad general. El proceso considera la disponibilidad de cada miembro y cómo los intercambios podrían afectar la desviación estándar de la armonía local entre los equipos.

Parámetros:

- **teams**: Lista de equipos, cada uno representado como un diccionario que incluye datos de armonía local, entre otros.
- **availabilityCourse**: Diccionario que indica la disponibilidad de los cursos para cada miembro de los equipos.
- **actual_dev**: Desviación estándar actual de la armonía local entre los equipos.

Pseudocódigo:

1. Identificar el equipo con la mayor armonía local (mejor equipo) y el equipo con la menor armonía local (peor equipo).

2. Dentro del mejor equipo, localizar al miembro con la mayor disponibilidad de bloques.
3. Dentro del peor equipo, identificar al miembro con la menor disponibilidad de bloques.
4. Evaluar la posibilidad de un intercambio entre estos dos miembros para mejorar la equidad en la armonía local.
5. Realizar el intercambio si resulta en una mejora potencial de la armonía local de ambos equipos.
6. Actualizar la composición de los equipos y recalculando la desviación estándar de la armonía local entre todos los equipos tras el intercambio.
7. Si el intercambio resulta en una desviación estándar mayor o igual a la actual, y se ha realizado el intercambio, revertir la acción y mantener la desviación estándar actual.
8. Retornar la nueva desviación estándar si el intercambio mejora la equidad, o la desviación estándar actual si no se realiza ningún cambio efectivo.

Cálculo de la Armonía Global entre Equipos

Descripción: La función `calculateGlobalHarmony` está diseñada para evaluar la armonía global de todos los equipos, basándose en la armonía local de cada uno y en la disponibilidad de los cursos. Su objetivo es optimizar la distribución de los miembros entre los equipos para lograr una mayor equidad en la armonía local a través de todos los equipos.

Parámetros:

- `teams`: Lista de equipos, cada uno representado por un diccionario que incluye la armonía local, entre otros datos relevantes.
- `availabilityCourse`: Diccionario que indica la disponibilidad de los cursos para los miembros de cada equipo.

Pseudocódigo:

1. Inicializar una lista para almacenar las armonías locales (`Als`) de todos los equipos.
2. Calcular la desviación estándar poblacional de las armonías locales para obtener un indicador inicial de la equidad entre los equipos.
3. Entrar en un ciclo que busca mejorar la equidad entre los equipos mediante el ajuste de su composición, con el objetivo de reducir la desviación estándar de las armonías locales.
4. Dentro del ciclo, invocar a la función `balancear`, pasando los equipos actuales, la disponibilidad de los cursos, y la desviación estándar actual, para intentar obtener una nueva desviación estándar más baja.
5. Calcular la mejora en la equidad comparando la desviación estándar antes y después del intento de balanceo.

6. Actualizar la desviación estándar con el nuevo valor obtenido tras el balanceo.
7. Tras finalizar el proceso de balanceo, recalcular las armonías locales de cada equipo con su composición ajustada.
8. Retornar la media y la desviación estándar poblacional de las armonías locales después de los ajustes, proporcionando una medida de la armonía global alcanzada entre los equipos.

3.3.2. Limitaciones de este Algoritmo

Se ha detallado el funcionamiento del algoritmo creado por Javier. A continuación, se identificarán las limitaciones de este diseño que impiden una mayor extensibilidad de la herramienta, estableciendo así las bases para la introducción de una solución rediseñada en la siguiente sección.

1. **Falta de modularidad:** Esta solución sufre de una notable falta de modularidad, evidenciada por las intersecciones innecesarias entre las funciones que dificultan la extensibilidad y generalización del algoritmo. Las funciones de generación y balance de equipos, junto con las de intercambio de estudiantes, están intrínsecamente conectadas con los componentes clave de la lógica del algoritmo, tales como el manejo de restricciones, el cálculo de la armonía y la estrategia de optimización. Estas funciones comparten intersecciones significativas en términos de armonía, restricciones y estrategias, de tal manera que un cambio en uno de estos aspectos implica la necesidad de realizar ajustes, a menudo no deseados, en los demás.
2. **Alta especificidad en la lógica del algoritmo:** El diseño de este algoritmo está marcado por una alta especificidad en su lógica, donde cada función está estrechamente adaptada a los requisitos inmediatos del problema de formación de equipos, basado en la disponibilidad horaria de los alumnos. Esta especificidad restringe severamente la reutilización de componentes del algoritmo en contextos diferentes o bajo criterios alternativos de optimización de equipos. Por ejemplo, el enfoque de evaluación y formación de equipos no puede adaptarse fácilmente para incorporar consideraciones como la compatibilidad de personalidades o las habilidades complementarias sin una revisión fundamental.
3. **Funciones poco generalizables:** Existen funciones dentro del algoritmo cuya especificidad compromete la generalidad y flexibilidad del sistema. Un claro ejemplo es la función balancear, que implementa una estrategia de intercambio de estudiantes basada en la cantidad de intersecciones de horario. Este método asume que la optimización se centra exclusivamente en la disponibilidad, lo cual no necesariamente se alinea con otros enfoques de formación de equipos, como la armonía basada en compatibilidad de personalidades. La lógica de intercambiar al 'mejor' por el 'peor' en función de un criterio único limita la aplicabilidad del algoritmo a escenarios donde otros factores pueden ser igual o más relevantes.
4. **Duplicidad de responsabilidades:** La solución original también padece de una considerable duplicidad en las responsabilidades asignadas a sus funciones. Diversas partes

del algoritmo realizan de manera redundante el cálculo de intersecciones de disponibilidad de los estudiantes, un proceso que idealmente se centralizaría para evitar ineficiencias y facilitar ajustes. Esta redundancia no solo afecta el rendimiento sino que también complica la mantenibilidad y escalabilidad del código. Por ejemplo, las funciones `inicialMatrix`, `checkTeam`, y `createTeams` llevan a cabo, de distintas maneras, operaciones que evalúan la disponibilidad de los estudiantes, a pesar de que estas tareas podrían ser gestionadas de manera más eficiente y coherente por un componente dedicado exclusivamente a tal fin.

3.4. Refactorización y Modularización del Sistema de Optimización

Para integrar nuevas restricciones en el algoritmo de optimización de manera efectiva, se hizo imprescindible una refactorización exhaustiva de la lógica existente. Este proceso implicó la identificación y separación de todos los componentes que podrían funcionar de manera independiente, adoptando un enfoque basado en la programación orientada a objetos. Este enfoque y diseño surgen a partir de las debilidades identificadas en la solución de Javier Lavados. Como resultado de esto, se establecieron los siguientes módulos clave, que se detallarán en las siguientes subsecciones.

3.4.1. Módulo de Restricciones

El módulo de restricciones se diseñó para garantizar que los equipos de estudiantes cumplan con ciertas normas y criterios específicos, utilizando el paradigma de Programación Orientada a Objetos (OOP). A través de este enfoque, se pueden integrar diversas restricciones de forma eficiente y modular.

Interfaz de Restricciones

La base para la implementación de restricciones es la interfaz abstracta `Restriction`, que define el método `complies_rules`. Este método es responsable de verificar si un equipo dado cumple con la restricción específica implementada por la clase derivada.

```
class Restriction(ABC):
    @abstractmethod
    def complies_rules(self, team, students_dd):
        pass
```

Restricción de Género

Una implementación concreta de `Restriction` es `GenderRestriction`, la cual asegura que no haya ninguna mujer o disidencia de género sola en un grupo.

Cumplimiento de Reglas de Género: La lógica para verificar esta restricción es la siguiente:

1. Contabiliza el número de mujeres o disidencias de género en el equipo.
2. Si el tamaño del equipo es mayor a uno y solo hay una persona de minoría de género, la restricción no se cumple.
3. En cualquier otro caso, la restricción se considera cumplida.

Restricción de Disponibilidad

`AvailabilityRestriction` es otra clase que implementa la interfaz `Restriction`, enfocándose en asegurar que nadie trabaje solo en el equipo. Esta restricción es la usada por el algoritmo de Javier Lavados[8].

Cumplimiento de Reglas de Disponibilidad: Esta restricción considera:

1. Se calcula la disponibilidad compartida del equipo y se verifica que cada miembro tenga al menos un compañero en cada franja horaria disponible.
2. Si todos los miembros del equipo tienen compañeros en sus franjas horarias disponibles, la restricción se cumple.

Administrador de Restricciones del Equipo

Para facilitar la aplicación de múltiples restricciones, se utiliza la clase `TeamRestrictionManager`, que mantiene una lista de restricciones e implementa un método para verificar si un equipo cumple con todas ellas.

```
class TeamRestrictionManager:
    def __init__(self):
        self.restrictions = []

    def add_restriction(self, restriction: Restriction):
        self.restrictions.append(restriction)
```

```

def check_restrictions(self, team, students_dd):
    return all(restriction.complies_rules(team, students_dd)
              for restriction in self.restrictions)

```

3.4.2. Diagrama UML - Componente de Restricciones

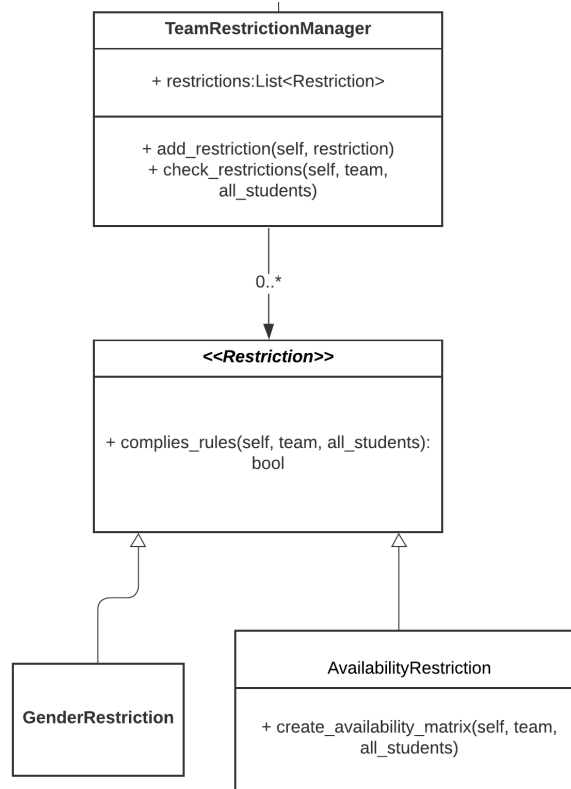


Figura 3.3: Diagrama UML correspondiente a este módulo, mostrando la relación entre **Restriction**, sus clases derivadas y el **TeamRestrictionManager**

3.5. Abstracción de Estrategias de Evaluación de Equipos: Componente de Armonía Local

Este segmento se enfoca en la abstracción de las estrategias para la evaluación de equipos, específicamente en el componente de armonía local.

3.5.1. Estrategias de Evaluación de Armonía

El diseño implementado para la evaluación de la armonía en equipos de estudiantes utiliza el paradigma de Programación Orientada a Objetos (OOP), proporcionando una estructura

extensible que permite integrar múltiples estrategias de evaluación de manera eficiente y modular.

Interfaz de Estrategias de Armonía

La base de este diseño es la interfaz abstracta `HarmonyStrategy`, que declara el método `calculate_local_harmony`. Esta interfaz actúa como un contrato para las clases derivadas, las cuales deben implementar el método proporcionando una lógica específica para calcular la armonía dentro de un equipo basado en criterios definidos.

```
class HarmonyStrategy(ABC):
    @abstractmethod
    def calculate_local_harmony(self, team, students_dd):
        pass
```

Armonía Basada en Tiempo

La clase `TimeBasedHarmony` es una implementación concreta de `HarmonyStrategy`, diseñada para evaluar la armonía de un equipo basándose en su disponibilidad horaria compartida. Este enfoque considera tanto la cobertura de disponibilidad a lo largo de la semana como la interacción potencial entre los miembros del equipo. Esta metodología de cálculo de armonía se mantuvo de la solución anterior.

Cálculo de la Armonía Local: La lógica de cálculo en `TimeBasedHarmony` sigue estos pasos:

1. Inicialmente, verifica si el tamaño del equipo es menor a dos, retornando un puntaje de armonía bajo para equipos demasiado pequeños.
2. Calcula la disponibilidad promedio de todo el equipo considerando los bloques horarios de cada día.
3. Evalúa el porcentaje de interacciones potenciales entre los miembros del equipo, basándose en su disponibilidad coincidente.
4. Combina ambos factores (disponibilidad promedio e interacciones potenciales) para producir un puntaje de armonía.

Calculadora de Armonía

Para integrar diversas estrategias de evaluación, se utiliza la clase `HarmonyCalculator`, que mantiene un diccionario de estrategias y proporciona una interfaz para calcular la armonía local invocando la estrategia deseada por su nombre.

```

class HarmonyCalculator:
    def __init__(self):
        self.strategies = {}

    def add_strategy(self, name, strategy: HarmonyStrategy):
        self.strategies[name] = strategy

    def calculate_local_harmony(self, name, team, students_dd):
        strategy = self.strategies.get(name)
        return strategy.calculate_local_harmony(team, students_dd)

```

Esta arquitectura facilita la incorporación de nuevas estrategias de evaluación de armonía, adaptándose a diferentes criterios y necesidades.

3.5.2. Diagrama UML - Componente de Armonía

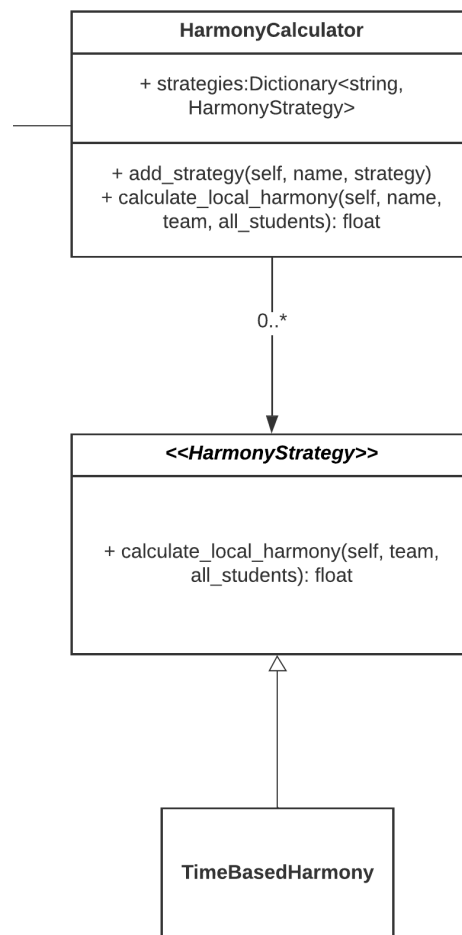


Figura 3.4: Diagrama UML dedicada a la armonía local, mostrando la relación entre HarmonyStrategy, su clase derivada y el HarmonyCalculator

3.5.3. Módulo de Optimización

El módulo de optimización emplea estrategias avanzadas para conformar equipos de estudiantes que maximicen la armonía interna mientras satisfacen ciertas restricciones preestablecidas, usando el paradigma de la Programación Orientada a Objetos (OOP). Este enfoque permite una integración flexible y eficiente de diferentes algoritmos de optimización.

Interfaz del Algoritmo de Optimización

Se define la interfaz `OptimizationAlgorithm`, que establece el método abstracto `optimize_teams` y `create_plausible_solution`. Este último usa constraint programming para generar un conjunto de soluciones iniciales que cumplen con las restricciones definidas.

```
from abc import ABC, abstractmethod

class OptimizationAlgorithm(ABC):
    @abstractmethod
    def optimize_teams(self, students, students_dd,
                      harmony_calculator, restriction_manager):
        pass
```

Generación de Soluciones Plausibles

La interfaz `OptimizationAlgorithm` incluye el método `create_plausible_solution`, diseñado para identificar un subconjunto de soluciones posibles que cumplen con las restricciones del problema. Este enfoque es crucial para manejar la complejidad NP-difícil de la formación de equipos.

Uso de Backtracking: El método `create_plausible_solution` emplea un algoritmo de backtracking para explorar el espacio de soluciones potenciales, retrocediendo cuando una configuración de equipo no cumple con las restricciones. Esto permite generar un conjunto reducido de soluciones plausibles, las cuales sirven como punto de partida para optimizadores como el Algoritmo Genético (GA) u otros, facilitando la búsqueda de una solución óptima.

Integración con el Administrador de Restricciones: Durante el proceso de backtracking, cada vez que se considera añadir un estudiante a un equipo, se utiliza el método `check_restrictions` del `TeamRestrictionManager` para evaluar si la configuración actual del equipo cumple con todas las restricciones establecidas. Esto garantiza que solo se generen soluciones iniciales viables.

Detalles del Proceso de Backtracking: El proceso sigue estos pasos esenciales:

1. Inicializa una solución de trabajo con grupos vacíos y un conjunto de estudiantes sin asignar.
2. Itera a través de los estudiantes, intentando añadir cada uno a un grupo de forma que se cumplan todas las restricciones.
3. Si la adición de un estudiante a cualquier grupo viola una restricción, se retrocede, eliminando al estudiante del grupo actual y probando una nueva configuración.
4. Este proceso se repite hasta que todos los estudiantes han sido asignados a un grupo o hasta que se alcanza el número máximo de intentos permitidos.

```
def create_plausible_solution(self, students, students_dd, number_of_groups,
                             team_restriction_manager, max_attempts=100):
    ...
    for random_student in shuffled_students:
        ...
        for i in sorted_groups:
            ...
            group_test = group_i + [random_student]
            if team_restriction_manager.check_restrictions(group_test, students_dd):
                plausible_solution[i] = group_test
            ...
```

Este fragmento de código ilustra cómo, para cada estudiante considerado, el algoritmo intenta encontrar un grupo donde pueda ser añadido sin violar las restricciones. Utiliza ‘check_restrictions’ para validar cada nueva configuración propuesta.

Esencial para la Extensibilidad: Este paso preliminar es esencial para abordar la naturaleza NP-difícil del problema de formación de equipos, permitiendo que el proceso de optimización se enfoque en un subconjunto manejable de soluciones que ya cumplen con las restricciones básicas. Además, favorece la extensibilidad del sistema, ya que permite integrar fácilmente otros algoritmos de optimización que puedan aprovechar este conjunto inicial de soluciones plausibles para encontrar configuraciones de equipo aún más armoniosas.

Algoritmo Genético para la Optimización de Equipos

La clase `GeneticAlgorithm` implementa la interfaz `OptimizationAlgorithm`, aplicando los principios de evolución y selección natural para optimizar la formación de equipos. Cada individuo dentro de este contexto se define como un equipo único, lo que simplifica la implementación de operaciones genéticas como el crossover, que se conceptualiza como el intercambio de estudiantes entre equipos para mejorar la armonía y cumplir con las restricciones preestablecidas. La población inicial se genera a través del método `create_plausible_solution`, garantizando que todos los equipos iniciales cumplan con las restricciones básicas y proporcionando un punto de partida robusto para la optimización. Este proceso se beneficia

significativamente del uso de la librería DEAP para la optimización genética, dado que facilita la definición de individuos, poblaciones y operaciones genéticas de manera flexible y eficiente.

Optimización con la Librería DEAP: DEAP (Distributed Evolutionary Algorithms in Python) es una librería avanzada para la programación de algoritmos evolutivos. Permite una fácil configuración de los parámetros genéticos y proporciona herramientas robustas para implementar selección, mutación, y crossover, así como la evaluación de fitness de los individuos. En este caso, `GeneticAlgorithm` utiliza DEAP para:

- Definir la estructura de un individuo y la población inicial.
- Aplicar operaciones de crossover diseñadas para intercambiar estudiantes entre equipos, optimizando según la estrategia de armonía definida e incorporando un chequeo del cumplimiento de restricciones.
- Realizar mutaciones que introducen variabilidad en la población, permitiendo explorar nuevas soluciones potenciales.
- Evaluar cada equipo usando `HarmonyCalculator` según la estrategia de armonía elegida, considerando tanto la armonía local como global.

El uso de DEAP permite una implementación eficaz del algoritmo genético, facilitando la exploración de un amplio espacio de soluciones y la convergencia hacia equipos óptimos basados en criterios de armonía predefinidos.

Operaciones de Crossover y Evaluación de Soluciones: El crossover se efectúa intercambiando estudiantes de equipos distintos, seleccionados al azar. Antes de realizar intercambios, el `RestrictionManager` verifica que estos respeten las restricciones establecidas. La evaluación de equipos se realiza mediante una fórmula que equilibra la armonía local con la global, utilizando `HarmonyCalculator` para calcular estos valores de forma estratégica. La evaluación de una solución se realiza mediante la siguiente fórmula:

Puntuación = promedio de armonías – (peso de armonía global × armonía global)

1

Criterios de Detención y Flexibilidad de Parámetros: El proceso se detiene al alcanzar un número predefinido de generaciones o cuando la solución muestra poca variación entre generaciones, indicando convergencia. La configurabilidad de parámetros como el número de

¹*Nota: En este contexto, la 'armonía global' se refiere a la desviación estándar de las armonías locales en una solución.

generaciones, la probabilidad de mutación y crossover, y el tamaño de la población inicial, permite adaptar el algoritmo a distintos escenarios y necesidades, desde diferentes secciones de estudiantes hasta otros contextos educativos.

La elección del algoritmo genético, especialmente en conjunto con la librería DEAP, se justifica por su capacidad para manejar la complejidad y la diversidad de las restricciones de formación de equipos, así como por su flexibilidad para ajustar parámetros en busca de la optimización según las necesidades específicas del problema. Este enfoque ofrece una metodología escalable y adaptable para enfrentar los desafíos inherentes a la formación de equipos en ambientes educativos.

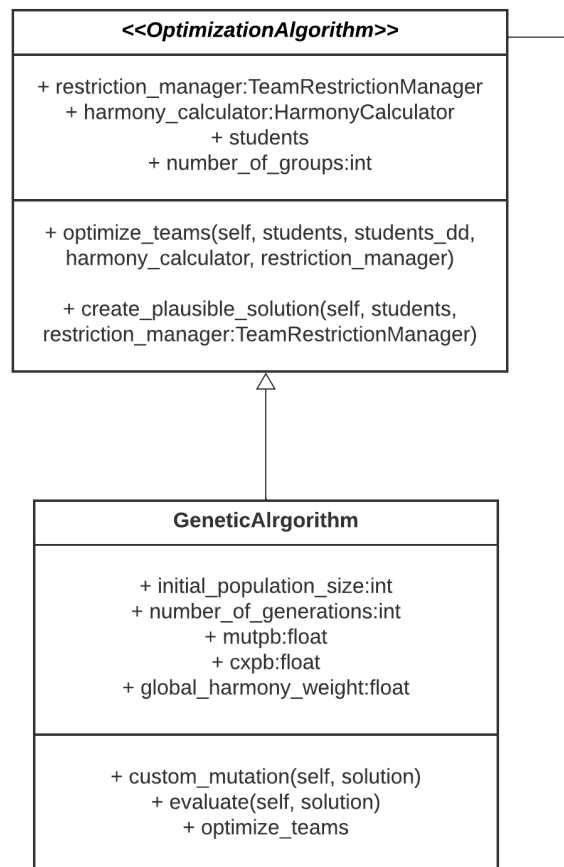


Figura 3.5: Diagrama UML correspondiente a este módulo, mostrando la relación entre `OptimizationAlgorithm`, `GeneticAlgorithm`, y las funciones de evaluación y mutación.

3.5.4. Unificación de Módulos Independientes - Clase `TeamOptimizer`

La clase `TeamOptimizer` representa la culminación de nuestro esfuerzo para integrar varios módulos independientes en un sistema cohesivo de formación de equipos. Esta clase actúa como el punto central de coordinación, interconectando los módulos de restricciones, cálculo

de armonía y algoritmos de optimización.

Funcionalidades de TeamOptimizer

- **Integración de Módulos:** TeamOptimizer combina el OptimizationAlgorithm (como GeneticAlgorithm), HarmonyCalculator, y TeamRestrictionManager para ofrecer una solución integral de formación de equipos.
- **Optimización de Equipos:** Utiliza optimize_teams del OptimizationAlgorithm para formar equipos basándose en parámetros de armonía y restricciones.
- **Flexibilidad y Modularidad:** Permite cambiar fácilmente entre diferentes algoritmos de optimización y estrategias de cálculo de armonía, gracias a su diseño modular.

```
class TeamOptimizer:
    def __init__(self, students, students_dd, number_of_groups,
                 optimization_algorithm, harmony_calculator,
                 team_restriction_manager):
        self.students = students
        self.students_dd = students_dd
        self.number_of_groups = number_of_groups
        self.optimization_algorithm = optimization_algorithm
        self.harmony_calculator = harmony_calculator
        self.team_restriction_manager = team_restriction_manager

    def optimize(self):
        # Use the optimization algorithm to form teams
        ...
```

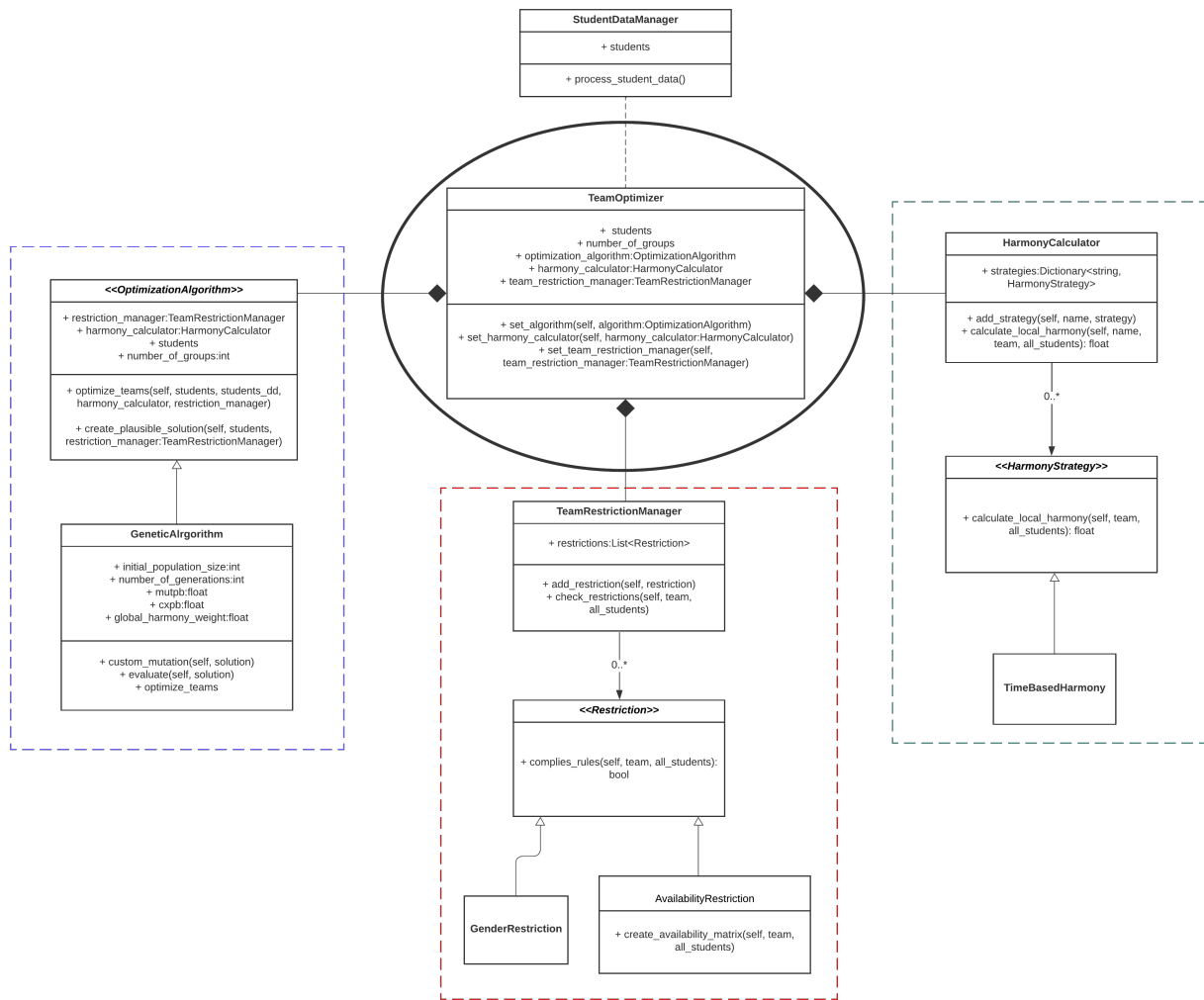


Figura 3.6: Diagrama UML completo. Este diagrama abarca **TeamOptimizer**, junto con sus interacciones con los otros módulos y las clases relacionadas con estos.

Capítulo 4

Evaluación

La evaluación del desempeño de la nueva herramienta de optimización primero abordará una comparación directa con la herramienta original, empleando los conjuntos de datos de estudiantes de los semestres de Primavera 2022 y Otoño 2023, tal como fueron utilizados en el trabajo realizado por Javier Lavados[8].

Para la ejecución del algoritmo genético en esta evaluación, se configuraron los siguientes parámetros:

- Número de generaciones: 65.
- Probabilidad de *crossover*: 1 (100 %).
- Probabilidad de mutación: 0 (0 %).
- Población inicial: 20.

La configuración de 65 generaciones se basa en la observación de convergencia del puntaje evaluador, calculado en función de la armonía del equipo. Esta convergencia sugiere un equilibrio entre la exploración de soluciones potenciales y la eficiencia computacional. Una población inicial de 20 se determinó óptima para proporcionar suficiente variabilidad genética sin comprometer el tiempo de ejecución, que osciló entre 2 y 15 segundos dependiendo de las especificaciones del curso.

La armonía local de los equipos formados se calculará siguiendo el mismo enfoque empleado en la versión original de esta herramienta, mediante la siguiente fórmula:

$$\begin{aligned} \text{Promedio de Disponibilidad por Bloque} &= \frac{1}{10} \sum_{d \in D} \sum_{t \in T} \sum_{e \in E} \text{Disp}(e, d, t) \\ \text{Promedio de Interacciones por Estudiante} &= \frac{1}{|E|} \sum_{e_1 \in E} \left(\frac{\sum_{e_2 \in E, e_2 \neq e_1} I(e_1, e_2)}{|E| - 1} \right) \end{aligned}$$

Luego:

$$\text{Armonía Local} = \text{Promedio}(\text{Promedio de Disponibilidad por Bloque} \\ + \text{Promedio de Interacciones por Estudiante})$$

Donde:

- D representa el conjunto de días {Lunes, Martes, Miércoles, Jueves, Viernes}.
- T representa los bloques de tiempo {AM, PM}.
- E es el conjunto de estudiantes en el equipo.
- $\text{Disp}(e, d, t)$ indica la disponibilidad del estudiante e en el día d y tiempo t .
- $I(e_1, e_2)$ es 1 si los estudiantes e_1 y e_2 tienen disponibilidad coincidente, y 0 de lo contrario.

Además, se evaluará la capacidad de la nueva herramienta para incorporar la restricción de género sin afectar negativamente la armonía del grupo.

Finalmente, examinaremos en mayor profundidad la capacidad de la herramienta para formar equipos que cumplan tanto con las restricciones de género, como las restricciones de horario. Utilizaremos datos reales de estudiantes de Proyecto de Software correspondientes a los semestres de Primavera 2022, Otoño 2023 y Otoño 2024 para llevar a cabo este análisis. Nuestro objetivo es centrarnos en la habilidad de la herramienta para crear grupos que respeten las restricciones establecidas. Para ello, en cada uno de los tres semestres que vamos a analizar, consideraremos cinco diferentes cantidades de grupos. Empezaremos con el número oficial de grupos utilizados en ese semestre, y luego examinaremos la formación de equipos con un incremento de +1 y +2 grupos, así como con una reducción de -1 y -2 grupos. Esta metodología nos permitirá evaluar la flexibilidad de la herramienta y determinar su capacidad para generar equipos viables bajo diversas condiciones.

- **Restricción de Horario:** Cada equipo debe asegurar que todos sus miembros compartan al menos 4 bloques horarios en los cuales puedan trabajar juntos con al menos otra persona. Esta restricción se puede expresar como:

$$\forall \text{ equipo } E, \forall \text{ miembro } m \in E : \sum_{\text{bloque } b} \text{horas}(m, b) \geq 4$$

donde $\text{horas}(m, b)$ indica si el miembro m trabaja en el bloque horario b con al menos otro miembro del equipo.

- **Restricción de Género:** Basándonos en los hallazgos de la investigación citada [3], los equipos deben conformarse de manera que no aislen a una mujer o persona de género diverso. En términos simples, un equipo debe:
 - No tener mujeres o personas de género diverso, o

- Tener más de una mujer o persona de género diverso.

Esta restricción se puede formalizar como:

$$\forall \text{ equipo } E : \text{MoD}(E) = 0 \vee \text{MoD}(E) > 1$$

donde $\text{MoD}(E)$ cuenta el número de mujeres y personas de género diverso en el equipo E .

Para la asignación de género, se emplea un método de estimación basado en los nombres de los alumnos. Este enfoque tiene limitaciones: no permite la inclusión de géneros no binarios y puede generar asignaciones erróneas. Sin embargo, se adoptó debido a que el género de los estudiantes no fue recopilado por la herramienta durante esos semestres.

El objetivo de esta evaluación es doble:

1. Demostrar que el rendimiento de la nueva herramienta de optimización es, al menos, comparable al de la solución original en términos de armonía de los grupos formados bajo las mismas restricciones de horario.
2. Mostrar que añadir nuevas restricciones, como el género, a la herramienta de optimización es factible y no parece perjudicar la cohesión de los equipos formados, preservando la armonía de los equipos a niveles similares a los obtenidos sin estas nuevas restricciones.

4.1. Resultados y análisis

4.1.1. Herramienta Anterior vs. Algoritmo Genético usando únicamente Restricción de Horario

Tabla 4.1: Herr. v1 Horario S2-2022

Equipo	Integrantes	Armonía
1	8	75.62 %
2	8	72.67 %
3	7	74.28 %
4	7	76.42 %
5	7	73.09 %
6	7	73.57 %
7	7	77.85 %
8	7	80.71 %
9	7	71.42 %
10	7	73.57 %
11	7	76.42 %
12	7	55.71 %
promedio		73.44 %
std		5.87 %

Tabla 4.2: Herr. v2 Horario S2-2022

Equipo	Integrantes	Armonía
1	8	73.75 %
2	8	71.25 %
3	7	75.71 %
4	7	73.81 %
5	7	75.71 %
6	7	74.29 %
7	7	75.71 %
8	7	75.71 %
9	7	77.86 %
10	7	74.29 %
11	7	75.00 %
12	7	71.43 %
promedio		74.54 %
std		1.79 %

Tabla 4.3: Herr. v1 Horario S1-2023

Equipo	Integrantes	Armonía
1	7	75.71 %
2	6	75.00 %
3	6	72.50 %
4	6	74.16 %
5	6	75.00 %
6	6	72.50 %
7	6	77.50 %
promedio		74.62 %
std		1.64 %

Tabla 4.4: Herr. v2 Horario S1-2023

Equipo	Integrantes	Armonía
1	7	75.00 %
2	6	75.00 %
3	6	75.00 %
4	6	75.00 %
5	6	76.67 %
6	6	76.67 %
7	6	75.83 %
promedio		75.60 %
std		0.73 %

La comparativa directa entre la herramienta anterior y el algoritmo genético, con enfoque en la restricción de horario para el semestre S2-2022, revela diferencias en los promedios de armonía y las desviaciones estándar. La herramienta anterior registró un promedio de armonía del 73.44 % y una desviación estándar de 5.87 %, mientras que la versión con el algoritmo genético presentó un promedio de 74.54 % y una desviación estándar de 1.79 %. En el semestre S1-2023, se observan resultados similares, con el algoritmo genético mostrando un promedio de armonía de 75.60 % y una desviación estándar de 0.73 %, comparado con la herramienta anterior. Los resultados son suficientemente similares para sugerir que la nueva herramienta no compromete los resultados de la conformación de equipos.

4.1.2. Herramienta Anterior vs. Algoritmo Genético con Restricciones de Horario y Género

Tabla 4.5: Herr. v1 Hor. & Gén. S2-2022

Equipo	Integrantes	Armonía
1	8	75.62 %
2	8	72.67 %
3	7	74.28 %
4	7	76.42 %
5	7	73.09 %
6	7	73.57 %
7	7	77.85 %
8	7	80.71 %
9	7	71.42 %
10	7	73.57 %
11	7	76.42 %
12	7	55.71 %
promedio		73.44 %
std		5.87 %

Tabla 4.6: Herr. v2 Hor. & Gén. S2-2022

Equipo	Integrantes	Armonía
1	8	73.75 %
2	8	71.25 %
3	7	75.71 %
4	7	73.81 %
5	7	75.71 %
6	7	74.29 %
7	7	75.71 %
8	7	75.71 %
9	7	77.86 %
10	7	74.29 %
11	7	75.00 %
12	7	71.43 %
promedio		74.54 %
std		1.79 %

Tabla 4.7: Herr. v1 Hor. & Gén. S1-2023

Equipo	Integrantes	Armonía
1	7	75.71 %
2	6	75.00 %
3	6	72.50 %
4	6	74.16 %
5	6	75.00 %
6	6	72.50 %
7	6	77.50 %
promedio		74.62 %
std		1.64 %

Tabla 4.8: Herr. v1 Hor. & Gén. S1-2023

Equipo	Integrantes	Armonía
1	6	75.00 %
2	6	76.67 %
3	6	75.00 %
4	7	73.57 %
5	6	75.83 %
6	6	77.50 %
7	6	75.83 %
promedio		75.63 %
std		1.18 %

Al aplicar las restricciones de horario y género en el semestre S2-2022, los resultados entre la herramienta anterior y la nueva versión del algoritmo genético muestran que ambos enfoques producen resultados comparables en términos de promedio de armonía y desviación estándar. La herramienta anterior muestra un promedio de 73.44 % con una desviación estándar de 5.87 %, y la nueva versión alcanza un promedio de 74.54 % con una desviación estándar de 1.79 %. Para el semestre S1-2023, los resultados siguen una tendencia similar, con el algoritmo genético obteniendo un promedio de armonía de 75.63 % y una desviación estándar de 1.18 %, en comparación con la herramienta anterior. Nuevamente estos resultados sugieren que el nuevo algoritmo de optimización y los cambios de diseño de la herramienta no comprometen la armonía de los equipos conformados.

4.1.3. Evaluación del Cumplimiento de Restricciones y Flexibilidad

Esta sección se dedica a explorar en profundidad cómo la herramienta aborda el desafío de conformar equipos que respeten las restricciones de género y horario, utilizando para ello datos reales de estudiantes de Proyecto de Software de los semestres de Primavera 2022, Otoño 2023, y Otoño 2024. Analizaremos variaciones en la cantidad de grupos para evaluar la flexibilidad de la herramienta y su capacidad para adaptarse a diferentes escenarios.

Semestre Primavera 2022

Tabla 4.9: Herr. v2, S2-2022 14 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	7	4	3	75.00 %	Cumple	Cumple
2	7	7	0	73.57 %	Cumple	Cumple
3	6	6	0	74.17 %	Cumple	Cumple
4	6	6	0	75.83 %	Cumple	Cumple
5	6	6	0	74.17 %	Cumple	Cumple
6	6	6	0	76.67 %	Cumple	Cumple
7	6	1	5	77.50 %	Cumple	Cumple
8	6	2	4	75.00 %	Cumple	Cumple
9	6	4	2	74.17 %	Cumple	Cumple
10	6	6	0	74.17 %	Cumple	Cumple
11	6	6	0	73.33 %	Cumple	Cumple
12	6	6	0	76.67 %	Cumple	Cumple
13	6	2	4	77.50 %	Cumple	Cumple
14	6	6	0	75.00 %	Cumple	Cumple

Tabla 4.10: Herr. v2, S2-2022 13 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	7	7	0	74.29 %	Cumple	Cumple
2	7	7	0	77.86 %	Cumple	Cumple
3	7	7	0	73.57 %	Cumple	Cumple
4	7	7	0	72.86 %	Cumple	Cumple
5	7	4	3	75.71 %	Cumple	Cumple
6	6	6	0	75.83 %	Cumple	Cumple
7	6	6	0	75.00 %	Cumple	Cumple
8	7	1	6	74.29 %	Cumple	Cumple
9	6	6	0	75.00 %	Cumple	Cumple
10	6	6	0	75.83 %	Cumple	Cumple
11	7	3	4	76.43 %	Cumple	Cumple
12	7	2	5	75.71 %	Cumple	Cumple
13	6	6	0	75.00 %	Cumple	Cumple

Tabla 4.11: Herr. v2, S2-2022 12 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	8	3	5	75.63 %	Cumple	Cumple
2	7	7	0	75.71 %	Cumple	Cumple
3	7	7	0	74.29 %	Cumple	Cumple
4	7	7	0	72.86 %	Cumple	Cumple
5	8	2	6	74.46 %	Cumple	Cumple
6	7	7	0	75.00 %	Cumple	Cumple
7	7	4	3	75.71 %	Cumple	Cumple
8	7	7	0	74.29 %	Cumple	Cumple
9	7	7	0	75.71 %	Cumple	Cumple
10	7	7	0	75.71 %	Cumple	Cumple
11	7	7	0	75.71 %	Cumple	Cumple
12	7	3	4	75.00 %	Cumple	Cumple

Tabla 4.12: Herr. v2, S2-2022 11 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	8	8	0	77.50 %	Cumple	Cumple
2	8	8	0	75.00 %	Cumple	Cumple
3	8	8	0	75.00 %	Cumple	Cumple
4	8	8	0	74.38 %	Cumple	Cumple
5	8	8	0	76.25 %	Cumple	Cumple
6	8	8	0	75.00 %	Cumple	Cumple
7	8	2	6	76.25 %	Cumple	Cumple
8	8	8	0	72.50 %	Cumple	Cumple
9	8	2	6	76.25 %	Cumple	Cumple
10	7	1	6	73.57 %	Cumple	Cumple
11	7	7	0	75.00 %	Cumple	Cumple

Tabla 4.13: Herr. v2, S2-2022 10 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	9	4	5	76.11 %	Cumple	Cumple
2	9	9	0	73.06 %	Cumple	Cumple
3	9	3	6	75.00 %	Cumple	Cumple
4	9	9	0	74.72 %	Cumple	Cumple
5	9	9	0	75.00 %	Cumple	Cumple
6	9	9	0	73.33 %	Cumple	Cumple
7	8	8	0	76.25 %	Cumple	Cumple
8	8	8	0	75.00 %	Cumple	Cumple
9	8	8	0	75.00 %	Cumple	Cumple
10	8	1	7	75.63 %	Cumple	Cumple

Semestre Otoño 2023

Tabla 4.14: Herr. v2, S1-2023 9 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	5	5	0	75.00 %	Cumple	Cumple
2	5	5	0	74.00 %	Cumple	Cumple
3	5	5	0	76.00 %	Cumple	Cumple
4	4	4	0	76.25 %	Cumple	Cumple
5	5	5	0	75.00 %	Cumple	Cumple
6	5	5	0	75.00 %	Cumple	Cumple
7	5	2	3	76.00 %	Cumple	Cumple
8	5	2	3	77.00 %	Cumple	Cumple
9	4	4	0	76.25 %	Cumple	Cumple

Tabla 4.15: Herr. v2, S1-2023 8 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	6	6	0	75.83 %	Cumple	Cumple
2	5	5	0	75.00 %	Cumple	Cumple
3	5	5	0	76.00 %	Cumple	Cumple
4	6	6	0	75.83 %	Cumple	Cumple
5	5	5	0	77.00 %	Cumple	Cumple
6	6	3	3	75.00 %	Cumple	Cumple
7	5	2	3	74.00 %	Cumple	Cumple
8	5	5	0	76.00 %	Cumple	Cumple

Tabla 4.16: Herr. v2, S1-2023 7 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	6	6	0	75.00 %	Cumple	Cumple
2	7	5	2	75.71 %	Cumple	Cumple
3	6	6	0	75.83 %	Cumple	Cumple
4	6	2	4	76.67 %	Cumple	Cumple
5	6	6	0	75.00 %	Cumple	Cumple
6	6	6	0	74.17 %	Cumple	Cumple
7	6	6	0	76.67 %	Cumple	Cumple

Tabla 4.17: Herr. v2, S1-2023 6 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	8	8	0	74.38 %	Cumple	Cumple
2	7	7	0	77.14 %	Cumple	Cumple
3	7	7	0	74.29 %	Cumple	Cumple
4	7	4	3	75.00 %	Cumple	Cumple
5	7	4	3	77.14 %	Cumple	Cumple
6	7	7	0	75.71 %	Cumple	Cumple

Tabla 4.18: Herr. v2, S1-2023 5 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	8	8	0	75.63 %	Cumple	Cumple
2	9	3	6	73.89 %	Cumple	Cumple
3	9	9	0	75.56 %	Cumple	Cumple
4	9	9	0	75.00 %	Cumple	Cumple
5	8	8	0	75.00 %	Cumple	Cumple

Semestre Otoño 2024

Tabla 4.19: Herr. v2, S1-2024 11 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	5	5	0	78.00 %	Cumple	Cumple
2	5	2	3	77.00 %	Cumple	Cumple
3	5	5	0	75.00 %	Cumple	Cumple
4	4	4	0	75.00 %	Cumple	Cumple
5	5	5	0	75.00 %	Cumple	Cumple
6	5	5	0	76.00 %	Cumple	Cumple
7	5	5	0	76.00 %	Cumple	Cumple
8	5	5	0	76.00 %	Cumple	Cumple
9	5	5	0	75.00 %	Cumple	Cumple
10	5	5	0	77.00 %	Cumple	Cumple
11	5	0	5	77.00 %	Cumple	Cumple

Tabla 4.20: Herr. v2, S1-2024 10 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	6	6	0	74.17 %	Cumple	Cumple
2	6	6	0	75.83 %	Cumple	Cumple
3	6	6	0	74.17 %	Cumple	Cumple
4	5	5	0	77.00 %	Cumple	Cumple
5	6	2	4	77.50 %	Cumple	Cumple
6	5	5	0	77.00 %	Cumple	Cumple
7	5	5	0	77.00 %	Cumple	Cumple
8	5	5	0	76.00 %	Cumple	Cumple
9	5	5	0	77.00 %	Cumple	Cumple
10	5	1	4	76.00 %	Cumple	Cumple

Tabla 4.21: Herr. v2, S1-2024 9 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	6	2	4	75.83 %	Cumple	Cumple
2	6	6	0	76.67 %	Cumple	Cumple
3	6	2	4	75.83 %	Cumple	Cumple
4	6	6	0	75.00 %	Cumple	Cumple
5	6	6	0	76.67 %	Cumple	Cumple
6	6	6	0	78.33 %	Cumple	Cumple
7	6	6	0	75.83 %	Cumple	Cumple
8	6	6	0	75.00 %	Cumple	Cumple
9	6	6	0	75.83 %	Cumple	Cumple

Tabla 4.22: Herr. v2, S1-2024 8 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	7	7	0	77.14 %	Cumple	Cumple
2	7	2	5	76.43 %	Cumple	Cumple
3	7	7	0	75.00 %	Cumple	Cumple
4	7	7	0	76.43 %	Cumple	Cumple
5	7	7	0	76.43 %	Cumple	Cumple
6	7	4	3	77.14 %	Cumple	Cumple
7	6	6	0	74.17 %	Cumple	Cumple
8	6	6	0	75.83 %	Cumple	Cumple

Tabla 4.23: Herr. v2, S1-2024 7 grupos

Equipo	Integrantes	Hombres	Mujeres/Disid.	Armonía	Restr. de Gén.	Restr. de Hor.
1	8	8	0	76.96 %	Cumple	Cumple
2	7	3	4	75.71 %	Cumple	Cumple
3	8	8	0	75.00 %	Cumple	Cumple
4	8	8	0	75.63 %	Cumple	Cumple
5	8	8	0	76.25 %	Cumple	Cumple
6	7	3	4	75.71 %	Cumple	Cumple
7	8	8	0	75.63 %	Cumple	Cumple

Tras el análisis de diversos escenarios y configuraciones de grupo a lo largo de tres semestres indica que la herramienta forma equipos que cumplen con todas las restricciones de género y horario. Además, la armonía de los equipos se mantiene entre un 72 y un 80 % durante todos los escenarios explorados.

4.2. Discusión

La revisión y análisis de los resultados obtenidos tras la implementación de la nueva herramienta de formación de equipos revelan aspectos importantes sobre el impacto de las mejoras realizadas. A continuación, se discuten estos aspectos en detalle.

4.2.1. Mantenimiento de la Armonía en Equipos Conformados

Una observación fundamental de los resultados obtenidos para los semestres analizados es que la armonía de los equipos conformados no se vio comprometida a pesar de las modificaciones sustanciales realizadas en el diseño y funcionalidad de la herramienta. Este hallazgo es relevante porque sugiere que las mejoras en diseño y extensibilidad, orientadas a hacer la herramienta más adaptable y fácil de modificar, no afectaron adversamente la eficacia de la formación de equipos. La capacidad de mantener e incluso mejorar ligeramente la armonía promedio de los equipos, mientras se reduce la variabilidad de estos resultados (como se indica por la disminución en la desviación estándar), sugiere que los cambios implementados han sido beneficiosos desde el punto de vista de la funcionalidad sin sacrificar la calidad de los equipos formados.

4.2.2. Importancia de la Extensibilidad y la Flexibilidad del Sistema

La nueva arquitectura de la herramienta, caracterizada por su diseño modular y flexibilidad, abre la puerta a futuras mejoras y exploraciones que anteriormente no eran factibles. Esta extensibilidad es crucial no solo para permitir la implementación de nuevos algoritmos de optimización y la modificación de parámetros existentes de manera más sencilla, sino también para adaptar la herramienta a necesidades cambiantes o a nuevos conocimientos obtenidos de la investigación y la práctica en el ámbito de la formación de equipos. La capacidad de integrar fácilmente nuevos parámetros o ajustar los algoritmos sin necesidad de reestructuraciones profundas garantiza que la herramienta pueda evolucionar y mejorar continuamente, manteniéndose relevante y efectiva.

4.2.3. Análisis de Cumplimiento de Restricciones en la Formación de Equipos

En el análisis de distintos escenarios y configuraciones de grupos de trabajo, observamos que la herramienta consistentemente generó equipos que respetaron todas las restricciones establecidas, incluidas las de género y horario. También se observó que independientemente de las variaciones en el número de equipos, se mantuvieron las distribuciones de género acordes a las restricciones específicas. Este hallazgo sugiere que la herramienta posee la capacidad para ajustarse a la formación de equipos con distribuciones de género específicas.

Este resultado muestra que la herramienta tiene una flexibilidad funcional para cumplir con requisitos particulares de formación de equipos.

4.2.4. Limitaciones

A pesar de los resultados positivos, es crucial reconocer las limitaciones de los resultados, principalmente relacionadas con la cantidad de datos disponibles para la comparación. La falta de una gama más amplia de semestres para el análisis limita la capacidad de generalizar los hallazgos a un contexto más extenso. Además, la ausencia de simulaciones de datos para compensar esta carencia de información real restringe la profundidad del análisis de la efectividad de la herramienta en diversos escenarios. Para futuras investigaciones, sería beneficioso ampliar el conjunto de datos analizados, ya sea a través de la inclusión de más semestres reales o mediante la generación de datos simulados que permitan una evaluación más robusta de la herramienta.

Capítulo 5

Conclusión

Este trabajo se centró en la extensión y mejora de una herramienta de aplicación web existente para la formación de equipos, ampliando su capacidad para manejar una variedad más amplia de restricciones, incluyendo disponibilidad horaria y género. Como parte de este proceso, se desarrolló y evaluó un algoritmo genético como una propuesta innovadora para reemplazar el algoritmo de optimización anterior. Esta reestructuración no solo respondió a la necesidad de una mayor modularidad y flexibilidad en el diseño del sistema, sino que también se orientó hacia facilitar el camino para que la aplicación siga creciendo en el futuro.

Los objetivos planteados al inicio del estudio se han alcanzado satisfactoriamente. Se desarrolló y evaluó un algoritmo que no solo cumple con los requisitos de formación de equipos sino que obtuvo métricas de armonía comparables en relación con el sistema previo. No se identificaron objetivos significativos que no se hayan alcanzado.

Los resultados positivos pueden atribuirse a la robustez del nuevo algoritmo genético y a la cuidadosa consideración de las restricciones de género. Este enfoque no comprometió la eficacia del algoritmo y logró generar soluciones deseables en cuanto a distribución de género, para todos los escenarios evaluados.

La relevancia del trabajo se manifiesta en su contribución a la creación de equipos más armónicos y equitativos. La investigación subraya la importancia de considerar la diversidad y la inclusión en las herramientas de optimización y cómo estas pueden ser incorporadas sin sacrificar el rendimiento.

Entre las lecciones aprendidas, destacamos la capacidad de los algoritmos genéticos para adaptarse a nuevas restricciones y mejorar los resultados. Para trabajos futuros, se propone explorar la inclusión de otras variables como habilidades y preferencias personales, y evaluar el impacto del algoritmo en diferentes contextos y con distintas poblaciones de estudiantes.

Estas conclusiones sellan el estudio, ofreciendo una visión integral del trabajo y abriendo la puerta a futuras investigaciones que puedan expandir aún más la utilidad y eficacia de la herramienta desarrollada.

Bibliografía

- [1] Belmont report. Inside NKU Research, Grants and Contracts Research Compliance IRB Resources Ethical Principles, 2013. Recuperado de: [<https://inside.nku.edu/rgc/research-compliance/irb/resources/ethical-principles.html>].
- [2] Sebastián Ignacio Aguilera Valenzuela. *Herramienta para el análisis del trabajo en equipo en cursos de ingeniería de software en el DCC*. PhD thesis, Departamento de Ciencias de la Computación, Universidad de Chile, 2022.
- [3] María Cecilia Bastarrica and Jocelyn Simmonds. The effects of team gender composition in capstone software development projects. In *SCCC*, pages 1–6, 2022.
- [4] Dirk G. Cattrysse and Luk N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. 1992.
- [5] Molly Espey. Diversity, effort, and cooperation in team-based learning. *The Journal of Economic Education*, 2017.
- [6] Abdul Rehman Gilal, Jafreezal Jaafar, Mazni Omar, and Muhammad Zahid Tunio. Impact of personality and gender diversity on software development teams’ performance. *IEEE International Conference on Computer, Communication, and Control Technology*, 2014.
- [7] Karina Kohl and Rafael Prikladnicki. Perceptions on diversity in brazilian agile software development teams: A survey. In *GE’18: IEEE/ACM 1st International Workshop on Gender Equality in Software Engineering*, page 4, 2018.
- [8] Javier Lavados. Herramienta para la conformación de equipos para proyecto de software. Memoria en curso.
- [9] Cynthia Lee and Jiing-Lih Farh. Joint effects of group efficacy and gender diversity on group cohesion and performance. *Applied Psychology: An International Review*, 53(1):136–154, 2004.
- [10] Maria Noonan. The ethical considerations associated with group work assessments. *Nurse Education Today*, 33:1422–1427, 2013.
- [11] Francesca Rossi, Peter van Beek, and Toby Walsh. Handbook of constraint programming. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Foundations of Artificial Intelligence*, volume 2, pages 181–309. Elsevier, 2006.

- [12] M. Srinivas and Lalit M. Patnaik. Genetic algorithms: A survey. *Computer*, 27(6):17–26, 1994.
- [13] Gunter K Stahl and Martha L Maznevski. Unraveling the effects of cultural diversity in teams: A retrospective of research on multicultural work groups and an agenda for future research. *Journal of International Business Studies*, 52:4–22, 2021.