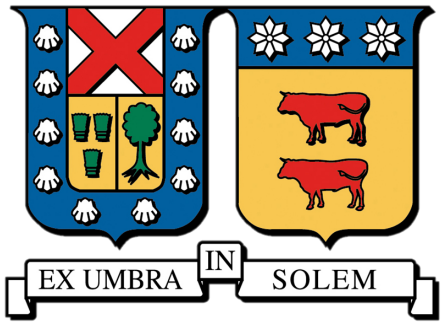
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"Desarrollo de un Sistema Backend para la Automatización de Formación de Equipos en Memorias Multidisciplinarias Usando Aprendizaje por Reforzamiento y Búsqueda de Árbol de Monte Carlo"

JAVIER ALFREDO AHUMADA CALDERÓN

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA: **NICOLÁS ALONSO JARA CARVALLO**

PROFESOR CORREFERENTE: **JOSÉ MANUEL MARTÍNEZ VERDUGO**

OCTUBRE 2025

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Desarrollo de un Sistema Backend para la Automatización de Formación de Equipos en Memorias Multidisciplinarias Usando Aprendizaje por Reforzamiento y Búsqueda de Árbol de Monte Carlo

Nombre del candidato(a): Javier Alfredo Ahumada Calderón

Carrera / Grado: Ingeniería Civil Telemática

Campus: Casa Central Valparaiso **;** **Departamento**: Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Nicolas Alonso Jara C., en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 07/10/2025 ; **Firma:** _____

Estudiante o Candidato(a):

Fecha: 07/10/2025 ; **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Agradezco profundamente a todas las personas que me han acompañado en este camino. A mi madre, Pamela, por inspirarme siempre a dar lo mejor de mí y enseñarme a confiar en mis propias capacidades. A mi padre, Jorge, por transmitirme con su ejemplo los valores de la perseverancia y la determinación. A mis hermanos, María José y Sebastián, por entregarme el cariño y apoyo que todo hermano mayor anhela. A mi pareja, Alexandra, por su apoyo incondicional en los buenos y malos momentos, y por impulsarme a crecer como persona. A mis compañeros y amigos, Santiago López y Cristian Vega, con quienes compartí la aventura de afrontar y superar este desafío juntos. A mis amigos más cercanos, que me animaron en aquellas largas noches de estudio y me recordaron que no estaba solo. Y, finalmente, a mi Profesor Guía, Nicolás Jara y a todos mis profesores, cuyo apoyo, orientación y enseñanzas me dieron las herramientas necesarias para alcanzar este logro.

Resumen

El Programa de Memorias Multidisciplinarias (PMM) de la Universidad Técnica Federico Santa María enfrenta desafíos críticos en la formación de equipos multidisciplinarios, caracterizados por un proceso manual ineficiente que requiere cuatro horas de procesamiento y genera frustración en el 80 % de los participantes. La distribución desigual de preferencias estudiantiles y el crecimiento sostenido del programa intensifican esta problemática.

Para abordar estas limitaciones, se desarrolló y evaluó un sistema backend que integra múltiples enfoques algorítmicos: técnicas de aprendizaje por reforzamiento (PPO y DQN) y búsqueda de árbol Monte Carlo (MCTS). La investigación incluyó el desarrollo de un ambiente de simulación personalizado usando Gymnasium, la implementación de agentes con Stable-Baseline3, y la optimización de hiperparámetros con Optuna. La arquitectura final incorpora un servidor con endpoints REST independientes que facilita la evaluación empírica de cada aproximación.

La implementación reveló diferencias fundamentales en la aplicabilidad práctica de cada método. Mientras que los algoritmos de aprendizaje por reforzamiento requieren reentrenamiento completo para cada nueva configuración del programa (limitando su viabilidad operacional), MCTS demostró adaptabilidad inmediata a configuraciones cambiantes. Los resultados experimentales muestran que MCTS alcanza 71.5 % de satisfacción promedio con tiempos de ejecución de minutos, comparado con 67.83 % de PPO tras múltiples horas de entrenamiento.

El sistema final reduce el tiempo de formación de equipos de cuatro horas a minutos, automatizando efectivamente el proceso mientras mantiene la calidad de las asignaciones. La investigación proporciona criterios claros para la selección de algoritmos en contextos educativos similares, identificando cuándo el aprendizaje por reforzamiento es apropiado versus cuándo métodos de búsqueda directa resultan más efectivos.

Keywords: Formación de equipos multidisciplinarios, aprendizaje por reforzamiento, Monte Carlo Tree Search, automatización de procesos educativos, algorit-

mos de asignación, API REST, optimización combinatoria.

Índice de figuras

1.1. Distribución Postulaciones vs Desafíos 2024. **Fuente:** Elaboración propia basada en datos proporcionados por la coordinación del Programa de Memorias Multidisciplinarias, 2024. 11
1.2. Carreras participantes para el año 2024. **Fuente:** Elaboración propia basada en datos proporcionados por la coordinación del Programa de Memorias Multidisciplinarias, 2024. 11
1.3. Crecimiento del programa. **Fuente:** Elaboración propia basada en datos proporcionados por la coordinación del Programa de Memorias Multidisciplinarias, 2024. 12
1.4. ¿Te sentiste frustrado durante el proceso de formación de equipos? **Fuente:** Elaboración propia basada en una encuesta realizada a estudiantes participantes del Programa de Memorias Multidisciplinarias, 2024. 13
1.5. ¿Sientes que el proceso de formación de los equipos fue eficiente? **Fuente:** Elaboración propia basada en una encuesta realizada a estudiantes participantes del Programa de Memorias Multidisciplinarias, 2024. 13
1.6. ¿Lograste quedar en alguna de tus postulaciones? **Fuente:** Elaboración propia basada en una encuesta realizada a estudiantes participantes del Programa de Memorias Multidisciplinarias, 2024. 14

2.1. Interacción entre agente y ambiente 25

2.2. Taxonomía no exhaustiva pero útil de algoritmos en RL moderno 26
2.3. Agentes disponibles en Stable-Baseline3 30

3.1. Diagrama de Contexto 39
3.2. Diagrama de Arquitectura 40
3.3. Arquitectura del Sistema de Asignación de Equipos 43

4.1. Entrada de todos los métodos del Servidor Selector de Asignador 54
4.2. Estructura de respuesta del componente Servidor Selector de Asignador 59
4.3. Función de Paso 64

5.1. Satisfacción de estudiantes en entrenamiento PPO y DQN con desafíos saturados al último 76
5.2. Satisfacción de estudiantes en entrenamiento PPO y DQN con desafíos saturados primero 76
5.3. Satisfacción de estudiantes en entrenamiento PPO y DQN con carreras saturados al último 77
5.4. Satisfacción de estudiantes en entrenamiento PPO y DQN con carreras saturados primero 77
5.5. Búsqueda de hiperparámetros con Optuna 78
5.6. Satisfacción promedio de los estudiantes ordenados de manera aleatoria para MCTS 82
5.7. Satisfacción promedio de los estudiantes ordenados por desafíos saturados de manera ascendente para MCTS 83
5.8. Satisfacción promedio de los estudiantes ordenados por desafíos saturados de manera descendente para MCTS 84
5.9. Satisfacción promedio de los estudiantes ordenados por tamaño de carreras de manera ascendente para MCTS 85
5.10. Satisfacción promedio de los estudiantes ordenados por tamaño de carreras de manera descendente para MCTS 86

5.11. Entrada y Respuesta de Servidor 87
5.12. Interfaz de Configuración 88
5.13. Interfaz de Estudiantes postulando 89
5.14. Interfaz con la distribución de estudiantes 90
5.15. Tiempos de Respuesta según Método vs. Simulaciones 92
5.16. Tiempo vs. Satisfacción vs. Número de Simulaciones 93

Índice de cuadros

2.1. Agentes de Stable-Baseline3 31
2.2. Agentes de Stable-Baseline3-Contrib 31
2.3. Tabla comparativa de Optimizadores de Hiperparámetros 33

4.1. Estructura de endpoints para algoritmos de aprendizaje automático . . . 57
4.2. Endpoints para algoritmos heurísticos externos 58
4.3. Rango de parámetros de configuración para PPO. 70
4.4. Rango de parámetros de configuración para DQN. 70

5.1. Mejores parámetros de configuración para PPO en este experimento. . . 79
5.2. Resultados de Evaluación de Métodos 91

Índice general

1. Introducción **7**

1.1. La Evolución de la Educación Superior 7

1.2. Memorias Multidisciplinarias en la UTFSM 9

1.3. Definición de la Problemática 10

1.3.1. Hipótesis de la Solución 15

1.3.2. Objetivos 16

1.3.3. Estructura 17

2. Estado del Arte y de la Técnica **19**

2.1. Sistemas de Asignación Automática de Recursos 20

2.2. Algoritmos de Toma de Decisiones 22

2.2.1. Búsqueda de Árbol Monte Carlo 23

2.3. Aprendizaje por Reforzamiento 25

2.3.1. Algoritmos de Aprendizaje por Reforzamiento 25

2.3.2. Herramientas y Frameworks 27

2.3.3. Proveedores de Agentes 28

2.3.4. Optimizadores de Hiperparámetros 31

2.4. Métricas y Evaluación 34

3. Diseño de la solución **35**

3.1. Análisis 35

3.1.1. Alternativas de solución 35

3.1.2. Acercamiento a la solución 36
3.2. Diseño 37
3.2.1. Requisitos del Sistema 37
3.2.2. Diagrama de Contexto 38
3.2.3. Diagrama de Arquitectura 40
3.2.4. Evolución del Módulo Asignador Automático de Desafíos . . . 42
3.2.5. Asignadores de Desafíos 44
3.2.6. Entorno de Soporte y Desarrollo 46

4. Implementación de la solución **49**
4.1. Definición de proceso de asignación 49
4.2. Evaluación de distribuciones 49
4.2.1. Definición de variables y conjuntos 49
4.2.2. Métricas de evaluación 50
4.3. Servidor Selector de Asignador 53
4.3.1. Especificación de Entrada 53
4.3.2. Estructura de Datos de Entrada 53
4.3.3. Procesamiento de Datos de Entrada 54
4.3.4. Especificaciones de Endpoints 56
4.3.5. Endpoints para Asignadores desarrollados 56
4.3.6. Endpoints para Algoritmos Externos 58
4.3.7. Especificación de Respuesta 59
4.4. Algoritmos de Aprendizaje por Reforzamiento 60
4.4.1. Ambiente de Simulación 61
4.4.2. Proximal Policy Optimization 68
4.4.3. Deep Q-Network 68
4.4.4. Optimización de Hiperparámetros 69
4.5. Monte Carlo Tree Search 71

5. Resultados **75**

5.1. Plan de evaluación 75

5.2. Evaluación del Aprendizaje por Reforzamiento 75

5.2.1. Lecciones Aprendidas del Desarrollo de RL 79

5.3. Evaluación de Árbol de Búsqueda de Monte Carlo 81

5.4. Evaluación de conectividad 86

5.4.1. Validación de llamadas APIs 87

5.4.2. Integración con la Interfaz de Usuario 87

5.5. Rendimiento del Sistema 90

5.5.1. Limitaciones del Aprendizaje por Reforzamiento 90

5.5.2. Procesamiento de Datos de Entrada 91

5.5.3. Tiempo de Distribución 91

5.6. Áreas de mejora 93

5.6.1. Escalabilidad 94

5.6.2. Persistencia de Datos 94

5.6.3. Monitoreo 95

6. Conclusiones **97**

6.1. Hallazgos Principales 97

6.1.1. Limitaciones del Aprendizaje por Reforzamiento 97

6.1.2. Efectividad de Monte Carlo Tree Search 98

6.1.3. Factores de Diseño Críticos 98

6.2. Contribuciones y Limitaciones 98

6.3. Recomendaciones 99

6.4. Direcciones Futuras 99

Capítulo 1

Introducción

1.1. La Evolución de la Educación Superior

En medio de la Cuarta Revolución Industrial, donde el avance tecnológico se acelera a un ritmo sin precedentes, los modelos educativos tradicionales enfrentan un punto de inflexión crucial. De manera similar ha cómo la década de 1960 presenció el surgimiento de ARPANET como respuesta a los desafíos de comunicación, con su primera conexión exitosa entre Stanford Research Institute (SRI) y UCLA el 29 de octubre de 1969; las instituciones educativas actuales están respondiendo a las complejas demandas de un mundo interconectado. El enfoque tradicional compartimentado de la educación, donde los estudiantes permanecen confinados dentro de sus disciplinas específicas, está dando paso a un modelo de aprendizaje más dinámico e integrado.

El informe "The Future of Jobs Report 2020"del Foro Económico Mundial, publicado por Saadia Zahidi, Vesselina Ratcheva, Guillaume Hingel y Sophie Brown, destaca que para el 2025, las habilidades críticas incluirán el pensamiento crítico y analítico, la resolución de problemas, las habilidades de autogestión (aprendizaje activo, resistencia, tolerancia al estrés, flexibilidad), el pensamiento analítico, la creatividad y el análisis e interpretación de datos [1]. Esta transformación no es simplemente una tendencia, sino

una evolución necesaria en la educación superior.

Empresas líderes en innovación han demostrado que los avances más significativos surgen en la intersección de múltiples disciplinas. La consultora McKinsey & Company, en su estudio de 2024 ”Making collaboration across functions a reality”que analizó 25 empresas en Europa, Asia y América del Norte, documentó que las organizaciones con colaboración interdepartamental tienen 5 veces más probabilidades de lograr innovaciones revolucionarias [2].

Ejemplos concretos incluyen Spotify, que desarrolló una metodología donde diseñadores e ingenieros trabajan juntos en todas las etapas del proyecto, alternando quién lidera según la fase[3], y Netflix, que ha capitalizado la colaboración entre ingenieros y científicos para desarrollar su motor de recomendaciones, procesando grandes cantidades de datos de espectadores en tiempo real para sugerir contenido[4].

Un estudio citado por Deloitte en colaboración con MIT Sloan Management Review reveló que el 83 % de las empresas digitalmente maduras utilizan equipos multifuncionales en comparación con el 71 % de las empresas en desarrollo y el 55 % de las organizaciones en etapa temprana. El estudio también encontró que el 69 % de las empresas digitalmente maduras reportan que estos equipos tienen una autonomía considerable sobre cómo lograr sus objetivos, comparado con el 53 % de las empresas en desarrollo[5].

Un metaanálisis de 21 estudios publicado en el International Journal of STEM Education por Wu, Yang, Zhou y colaboradores (2024) confirmó una correlación moderadamente positiva ($r = 0.452$) entre la educación STEM y las habilidades de enseñanza interdisciplinaria, proporcionando evidencia empírica del cambio hacia modelos educativos más integrados[6].

Este movimiento global hacia la educación interdisciplinaria no se trata solo de preparar a los estudiantes para el futuro laboral; se trata de formar una nueva generación de solucionadores de problemas capaces de abordar los desafíos complejos de nuestro tiempo.

1.2. Memorias Multidisciplinarias en la UTFSM

La Universidad Técnica Federico Santa María, alineada con las demandas de la industria moderna y los desafíos tecnológicos contemporáneos, implementó el Programa de Memorias Multidisciplinarias (PMM) como una iniciativa innovadora en la formación profesional [7]. Este programa, impulsado por el Proyecto Ingeniería 2030 por The Clover, representa un esquema de titulación revolucionario que trasciende los límites tradicionales de la educación en ingeniería [8].

El PMM se fundamenta en tres pilares estratégicos: la vinculación permanente y significativa con la industria, el impulso al desarrollo tecnológico nacional, y el fortalecimiento de competencias transversales en los futuros profesionales. Durante dos semestres académicos, los estudiantes trabajan en equipos multidisciplinarios para desarrollar soluciones tecnológicas concretas que responden a desafíos reales planteados por la industria y la sociedad.

La metodología del programa se distingue por su enfoque "just-in-time"[9], donde los estudiantes reciben formación contextualizada según las necesidades específicas de cada etapa del proyecto. Este modelo educativo innovador combina el desarrollo de habilidades técnicas con competencias transversales esenciales como la comunicación efectiva, el emprendimiento, la innovación, el liderazgo, y la gestión de proyectos. El programa se estructura a través de cuatro asignaturas y dos talleres de título, diseñados para proporcionar una experiencia de aprendizaje integral.

La UTFSM ha logrado establecer, a través del PMM, un puente efectivo entre la academia y la industria. Los estudiantes no solo aplican sus conocimientos técnicos en proyectos reales, sino que también desarrollan prototipos funcionales o pruebas de concepto que pueden tener un impacto significativo en el mercado. Esta vinculación temprana con el sector productivo ofrece a los estudiantes una ventaja competitiva al momento de iniciar su vida profesional, ya sea como emprendedores tecnológicos o

como profesionales en empresas establecidas.

El programa ha evolucionado desde sus inicios, expandiéndose desde la carrera de Ingeniería Civil Telemática hacia otras disciplinas como Ingeniería en Diseño de Productos, Ingeniería Civil Industrial, e Ingeniería en Construcción Civil, entre otras. Esta diversidad de perspectivas y conocimientos técnicos enriquece el proceso de desarrollo de soluciones y refleja la realidad del mundo profesional actual, donde la innovación surge de la colaboración entre diferentes áreas del conocimiento.

1.3. Definición de la Problemática

El Programa de Memorias Multidisciplinarias (PMM) enfrenta un desafío significativo en su proceso de formación de equipos, el cual se ha vuelto más evidente con el crecimiento del programa. Este desafío quedó particularmente expuesto durante la jornada de formación de equipos del 12 de abril de 2024, donde se manifestaron diversas complejidades operativas y logísticas que merecen un análisis detallado.

Para caracterizar esta situación, se realizó un estudio utilizando los datos disponibles en la plataforma del PMM. La información analizada incluyó datos de los estudiantes participantes, sus carreras de origen, los desafíos a los cuales postularon con sus respectivas preferencias, y los detalles específicos de cada desafío presente en la iteración 2024.

El proceso actual de postulación permite que los estudiantes seleccionen un máximo de tres desafíos de su interés, priorizándolos según sus preferencias personales. En la versión 2024, el programa registró la participación de 64 estudiantes distribuidos entre 38 desafíos disponibles. Sin embargo, el análisis reveló un desequilibrio significativo en la distribución de postulaciones, como se observa en la Figura 1.1, donde 23 estudiantes (aproximadamente un tercio del total) concentraron su interés en el desafío 36, enfocado en la clasificación de imágenes de mamografía mediante Machine Learning.

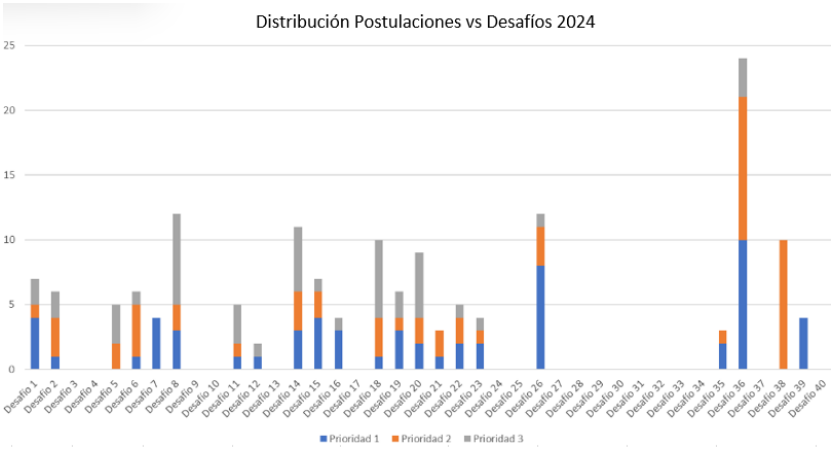


Figura 1.1: Distribución Postulaciones vs Desafíos 2024.
Fuente: Elaboración propia basada en datos proporcionados por la coordinación del Programa de Memorias Multidisciplinarias, 2024.

Esta concentración de preferencias puede atribuirse principalmente al perfil académico y los intereses específicos de los estudiantes de Ingeniería Civil Telemática, quienes representan el 82 % del total de participantes en esta versión (ver Figura 1.2).

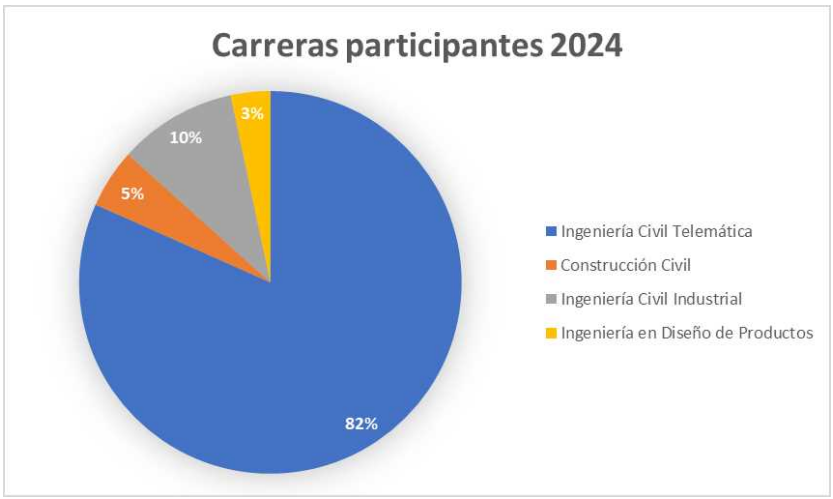


Figura 1.2: Carreras participantes para el año 2024.
Fuente: Elaboración propia basada en datos proporcionados por la coordinación del Programa de Memorias Multidisciplinarias, 2024.

Esta composición académica del programa genera un sesgo hacia desafíos afines al perfil telemático, relegando aquellos que pueden resultar menos atractivos para este

grupo mayoritario. Un ejemplo claro es el desafío 3, relacionado con la simulación y control de velocidades de la solución química al interior de paneles de tubos de calderas para limpieza química, el cual recibió 0 postulaciones.

La magnitud de esta problemática aumentará progresivamente con el crecimiento del PMM, como se evidencia en la Figura 1.3, que muestra la tendencia ascendente en el número de participantes a lo largo de los años.

Figura 1.3: Crecimiento del programa.
Fuente: Elaboración propia basada en datos proporcionados por la coordinación del Programa de Memorias Multidisciplinarias, 2024.

El proceso manual de formación de equipos implementado durante la jornada del 12 de abril de 2024 evidenció limitaciones críticas tanto en términos operativos como de satisfacción estudiantil. La asignación manual requirió aproximadamente cuatro horas continuas de trabajo, tiempo que se tradujo en una experiencia problemática para los participantes, según revelan las encuestas realizadas a los estudiantes.

Los resultados de la evaluación estudiantil muestran un panorama preocupante respecto a la percepción del proceso. Como se observa en la Figura 1.4, el 80 % de los encuestados manifestó haber experimentado frustración durante la jornada de formación de equipos. Esta elevada tasa de insatisfacción sugiere que el método actual no solo es ineficiente desde una perspectiva operativa, sino que además genera una experiencia negativa.

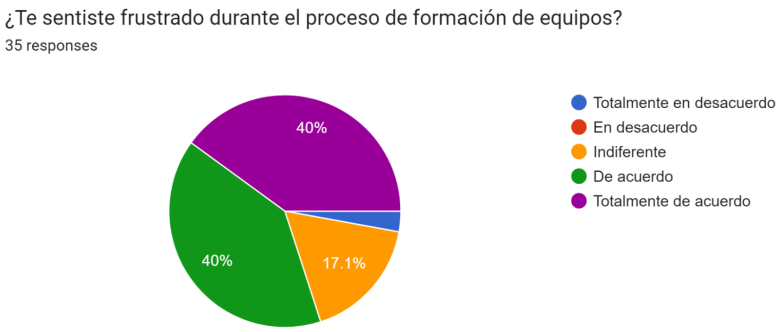


Figura 1.4: ¿Te sentiste frustrado durante el proceso de formación de equipos?
Fuente: Elaboración propia basada en una encuesta realizada a estudiantes participantes del Programa de Memorias Multidisciplinarias, 2024.

La percepción de ineficiencia del proceso resulta aún más contundente, con un 97 % de los estudiantes calificando el método manual como ineficiente (Figura 1.5). Esta opinión evidencia una clara desconexión entre las expectativas estudiantiles y la realidad operativa del programa, lo que plantea la necesidad urgente de implementar mejoras sistémicas en el proceso de asignación.

Figura 1.5: ¿Sientes que el proceso de formación de los equipos fue eficiente?
Fuente: Elaboración propia basada en una encuesta realizada a estudiantes participantes del Programa de Memorias Multidisciplinarias, 2024.

No obstante, es fundamental reconocer que, a pesar de las deficiencias, el método

manual logró resultados satisfactorios en términos de asignación final. El 77 % de los estudiantes consiguió ser asignado a alguno de los desafíos incluidos en sus postulaciones originales (Figura 1.6). Este dato sugiere que, aunque el proceso presenta serias limitaciones operativas y genera insatisfacción, mantiene cierta efectividad en términos de cumplimiento de preferencias estudiantiles.

Figura 1.6: ¿Lograste quedar en alguna de tus postulaciones?
Fuente: Elaboración propia basada en una encuesta realizada a estudiantes participantes del Programa de Memorias Multidisciplinarias, 2024.

El análisis realizado permite identificar tres aspectos críticos que caracterizan la problemática del PMM y que requieren atención para asegurar la sostenibilidad y calidad del programa:

- Proceso Manual: La metodología actual de formación de equipos presenta limitaciones severas en términos de recursos temporales y experiencia estudiantil. El proceso no solo resulta ineficiente desde una perspectiva operativa, sino que genera altos niveles de estrés y frustración tanto para los organizadores como en participantes.
- Balance Multidisciplinario: Existe la necesidad de garantizar la formación de equipos verdaderamente multidisciplinarios, un desafío que se complica por la distribución asimétrica de estudiantes por carrera.

- Escalabilidad: El crecimiento continuo del programa expone la fragilidad del sistema actual resultando progresivamente más impracticable, no solo por los recursos temporales requeridos, sino por la complejidad exponencial que implica gestionar un mayor número de variables, preferencias y restricciones de manera simultánea.

Esta problemática requiere una solución que no solo una automatización del proceso, sino que también considere las múltiples variables y restricciones inherentes al programa, asegurando la formación de equipos efectivos y balanceados que cumplan con los objetivos educativos y profesionales del programa.

1.3.1. Hipótesis de la Solución

La complejidad y los desafíos observados en el proceso de formación de equipos del Programa de Memorias Multidisciplinarias requieren una evaluación sistemática de diferentes técnicas algorítmicas para determinar su viabilidad práctica en este contexto específico. Esta investigación busca determinar la viabilidad de diferentes técnicas algorítmicas para automatizar la formación de equipos multidisciplinarios, explorando tanto las fortalezas como las limitaciones inherentes de cada aproximación en el contexto específico del PMM. La implementación y evaluación empírica de algoritmos de aprendizaje por reforzamiento (Reinforcement Learning) y búsqueda de árbol de Monte Carlo (Monte Carlo Tree Search) permitirá identificar no solo cuál método es más efectivo, sino también establecer criterios claros sobre cuándo cada aproximación es apropiada para problemas similares. Esta investigación busca caracterizar cómo cada técnica algorítmica:

- Se adapta a las características dinámicas del programa (cambios anuales en estudiantes, desafíos y restricciones)

- Balancea las preferencias individuales con los requisitos de diversidad disciplinaria

- Maneja múltiples restricciones simultáneamente bajo diferentes condiciones operativas

- Escala con el crecimiento del programa considerando recursos computacionales disponibles

- Equilibra calidad de asignación con viabilidad operativa en contextos educativos reales

1.3.2. Objetivos

Objetivo General

Desarrollar, implementar y evaluar comparativamente un servidor backend que integre diferentes algoritmos de toma de decisiones para automatizar el proceso de formación de equipos en el Programa de Memorias Multidisciplinarias, determinando cuál técnica resulta más apropiada para las necesidades específicas del programa y proporcionando recomendaciones basadas en evidencia empírica.

Objetivos Específicos

- Diseñar e implementar un servidor backend con una API REST que exponga los endpoints necesarios para la formación automatizada de equipos, garantizando una integración efectiva con la interfaz de usuario desarrollada por el equipo complementario.

- Desarrollar e implementar diferentes algoritmos de toma de decisiones, incluyendo aprendizaje por reforzamiento y búsqueda de árbol de Monte Carlo, adaptándolos específicamente al contexto de formación de equipos multidisciplinarios.

- Crear un marco de evaluación comparativa para analizar el desempeño de los diferentes algoritmos implementados, considerando métricas como tiempo de eje-

cución, calidad de las asignaciones y satisfacción de restricciones multidisciplinarias.

- Determinar y documentar los casos de uso óptimos para cada algoritmo implementado, estableciendo recomendaciones claras sobre qué enfoque utilizar según diferentes escenarios y condiciones del programa.

- Validar la efectividad del sistema mediante pruebas con datos históricos del PMM, comparando los resultados con las asignaciones manuales previas y evaluando la satisfacción de las restricciones del programa.

1.3.3. Estructura

Este documento presenta una estructura metodológica que inicia con una revisión exhaustiva del estado del arte, seguida por el análisis del desarrollo de la solución propuesta. En esta sección se examinan las diferentes alternativas de solución y se fundamenta la aproximación hacia la solución definitiva. Una vez definida la propuesta final, se detalla el proceso completo que abarca el diseño, la implementación y el análisis de resultados, culminando con las conclusiones derivadas de la investigación.

El presente proyecto se desarrolla en el marco de las memorias multidisciplinarias del año 2024, basándose en la problemática propuesta por Marcos Zúñiga, profesor del área de Telemática, director de postgrado y programas del Departamento de Electrónica de la Universidad Técnica Federico Santa María.

Capítulo 2

Estado del Arte y de la Técnica

La automatización de procesos de toma de decisiones en entornos educativos representa uno de los desafíos más significativos en la intersección entre la inteligencia artificial y la gestión académica. La formación de equipos multidisciplinarios, en particular, constituye un problema complejo que requiere considerar múltiples variables, restricciones y objetivos simultáneamente, mientras se mantiene un balance entre la eficiencia operativa y la satisfacción de los participantes. En este capítulo se presenta una revisión exhaustiva del estado actual de las tecnologías y aproximaciones disponibles para abordar esta problemática.

El análisis se estructura en cinco áreas fundamentales que abarcan desde los sistemas existentes de asignación automática hasta las métricas de evaluación empleadas en contextos similares.

El objetivo de esta revisión es identificar las metodologías más adecuadas que permitan:

- Automatizar eficientemente la formación de equipos multidisciplinarios
- Considerar múltiples restricciones y preferencias simultáneamente
- Optimizar la satisfacción global de los participantes
- Garantizar la escalabilidad del sistema para futuras iteraciones del programa

Este análisis no es solo identificar las mejores prácticas y tecnologías disponibles, sino también comprender las limitaciones y desafíos que han enfrentado implementaciones similares. Este entendimiento resulta crucial para el diseño de una solución que no solo automatice el proceso de formación de equipos, sino que también garantice su escalabilidad y adaptabilidad a las necesidades cambiantes del programa.

A continuación, se desarrolla un análisis detallado de cada una de estas áreas, evaluando su potencial aplicación en el contexto específico de nuestro problema y estableciendo las bases para la propuesta de solución que se presentará en capítulos posteriores.

2.1. Sistemas de Asignación Automática de Recursos

En la última década, los sistemas de asignación automática de recursos han experimentado una evolución significativa, particularmente en entornos educativos y organizacionales donde la formación de equipos efectivos es crucial. La literatura académica y la práctica industrial han producido diversas aproximaciones para abordar este desafío.

Uno de los referentes más destacados es el sistema CATME (Comprehensive Assessment of Team Member Effectiveness) [10], que ha sido implementado en más de 2,000 instituciones educativas. Este sistema utiliza algoritmos de optimización para formar equipos basándose en múltiples criterios, incluyendo horarios de los estudiantes, habilidades previas y diversidad de perspectivas. La efectividad de CATME ha sido validada a través de estudios longitudinales que demuestran una mejora significativa en el rendimiento de los equipos formados automáticamente en comparación con la asignación manual.

En la institución de Rose-Hulman de tecnología desarrollo Team-Maker un sistema web que automatiza el proceso de asignación de estudiantes a equipos utilizando criterios definidos por el instructor. Su enfoque innovador incluye la consideración de múltiples atributos estudiantiles como calificaciones previas, GPA, habilidades de escri-

tura y horarios de disponibilidad, además de las competencias técnicas tradicionales. El sistema implementa principios del aprendizaje cooperativo, distribuyendo heterogéneamente las habilidades across los equipos y evitando que las minorías subrepresentadas queden en desventaja numérica. La funcionalidad de Team-Maker fue posteriormente incorporada en el sistema CATME, creando una herramienta integrada que combina formación automatizada de equipos con evaluación de efectividad de miembros del equipo. [11]

En el ámbito latinoamericano, la Pontificia Universidad Católica de Chile implementó un sistema para la formación computacional de equipos diversos y conectados [12]. Su aproximación utiliza el algoritmo genético NSGA-II (Non-dominated Sorting Genetic Algorithm II) para crear equipos que balancean diversidad en atributos de miembros y conexiones previas entre ellos, logrando resultados prometedores en términos de familiaridad y diversidad en equipos multidisciplinarios, con aplicaciones exitosas en proyectos de innovación colaborativa.

Los sistemas mencionados comparten características comunes que han probado ser efectivas:

- La incorporación de múltiples criterios de evaluación
- La capacidad de manejar restricciones duras y blandas
- La consideración de preferencias individuales
- La escalabilidad para manejar grupos grandes de participantes

Sin embargo, también presentan limitaciones comunes, como la dificultad para adaptar los algoritmos a cambios en tiempo real y la necesidad de calibración manual de parámetros importantes.

2.2. Algoritmos de Toma de Decisiones

Los problemas de asignación complejos, como la formación de equipos multidisciplinarios, han sido abordados a través de diversos enfoques algorítmicos, cada uno con sus propias fortalezas y limitaciones. La literatura científica ha documentado la evolución de estos métodos, desde aproximaciones heurísticas simples hasta sofisticados algoritmos de optimización.

El algoritmo Hill Climbing representa una de las aproximaciones más fundamentales. Este método realiza optimizaciones locales, variando una única variable en cada iteración y siguiendo una dirección de ascenso o descenso. Su principal ventaja radica en su simplicidad de implementación y bajo costo computacional. Sin embargo, tiende a quedar atrapado en óptimos locales, lo que puede resultar en soluciones subóptimas para problemas de asignación complejos [13].

Los Algoritmos Genéticos han demostrado ser particularmente efectivos en este dominio. Basados en los principios de evolución natural, estos algoritmos mantienen una población de posibles soluciones que evolucionan a través de operaciones de selección, cruzamiento y mutación. En un contexto universitario similar, se logró una mejora del 40 % en la satisfacción general de los equipos formados en comparación con métodos tradicionales [14].

El Sistema Max-Min, por su parte, ofrece una aproximación robusta para la toma de decisiones en condiciones de incertidumbre. Este método busca maximizar el resultado mínimo posible, proporcionando así una garantía de desempeño en el peor caso. Los estudios de Chen y Liu demuestran su efectividad particular en contextos donde la equidad en la asignación es una prioridad [15].

La búsqueda Tabú ha emergido como una alternativa prometedora. Este algoritmo utiliza una memoria de soluciones previamente exploradas para evitar ciclos y escapar de óptimos locales. Rodriguez y Thompson documentaron su aplicación exitosa en la formación de equipos de investigación, destacando su capacidad para mantener la diversidad en las soluciones generadas [16].

El Método de Monte Carlo, aunque computacionalmente más intensivo, ofrece ventajas significativas en términos de exploración del espacio de soluciones. Su capacidad para manejar la incertidumbre y explorar múltiples escenarios simultáneamente lo hace particularmente valioso en contextos donde las preferencias y restricciones pueden cambiar dinámicamente [17].

La comparación empírica de estos algoritmos revela patrones interesantes:

- Los Algoritmos Genéticos tienden a producir mejores resultados a largo plazo, pero requieren una calibración cuidadosa de sus parámetros.
- La búsqueda Tabú ofrece un mejor balance entre exploración y explotación del espacio de soluciones.
- El Sistema Max-Min resulta especialmente útil cuando la equidad es una prioridad.
- El Método de Monte Carlo demuestra mayor robustez frente a cambios en las condiciones iniciales.

Un aspecto crucial en la implementación de estos algoritmos es la definición de la función objetivo. Las investigaciones recientes sugieren que una combinación ponderada de múltiples criterios, incluyendo preferencias individuales, diversidad disciplinaria y restricciones prácticas, produce los mejores resultados.

La tendencia actual apunta hacia enfoques híbridos que combinan las fortalezas de múltiples algoritmos. Por ejemplo, se propuso para mejorar la precisión del diagnóstico de fallas en redes inteligentes (Smart Grid) una arquitectura que integra elementos de búsqueda Tabú con optimización basada en algoritmos genéticos, logrando resultados superiores a cualquiera de los métodos por separado. [18]

2.2.1. Búsqueda de Árbol Monte Carlo

La Búsqueda de Árbol Monte Carlo (MCTS) representa una evolución significativa en los métodos de toma de decisiones secuenciales. A diferencia de los enfoques

tradicionales, MCTS combina la exploración sistemática del espacio de soluciones con un muestreo aleatorio controlado, lo que lo hace particularmente efectivo en problemas con espacios de búsqueda extensos.

El algoritmo opera a través de cuatro fases fundamentales:

- **Selección:** Partiendo de la raíz, el algoritmo selecciona nodos sucesores hasta alcanzar un nodo hoja, utilizando una política de selección que equilibra exploración y explotación.
- **Expansión:** Se añade uno o más nodos hijos al nodo hoja seleccionado.
- **Simulación:** Se realiza una simulación desde el nuevo nodo hasta un estado terminal o una profundidad predefinida.
- **Retropropagación:** Los resultados de la simulación se propagan hacia atrás actualizando las estadísticas de los nodos visitados.

Una ventaja significativa de MCTS es su capacidad de integración con otros métodos de optimización. Por ejemplo, la combinación de MCTS con aprendizaje por reforzamiento ha demostrado resultados excepcionales en varios dominios. En AlphaGo demostraron la potencia de esta combinación, donde MCTS proporciona una búsqueda guiada mientras que el aprendizaje por reforzamiento mejora la política de selección. [19]

La implementación de MCTS en problemas de asignación requiere considerar varios aspectos clave:

- La definición de una función de evaluación apropiada para las simulaciones
- El balance entre el tiempo de exploración y la calidad de las soluciones
- La adaptación del algoritmo para manejar restricciones específicas del dominio
- La integración efectiva con otros métodos de optimización

2.3. Aprendizaje por Reforzamiento

El aprendizaje por reforzamiento ha emergido como uno de los paradigmas más prometedores en el campo de la inteligencia artificial para abordar problemas complejos de toma de decisiones. Como señala Sutton y Barto [20], este método se distingue por su capacidad de aprender qué acciones tomar a través de la interacción con un ambiente, con el objetivo de maximizar una señal numérica de recompensa (Figura 2.1). La característica distintiva de este enfoque es que el agente debe descubrir qué acciones producen las mayores recompensas mediante un proceso de prueba y error, sin necesidad de una especificación explícita de cómo realizar la tarea.

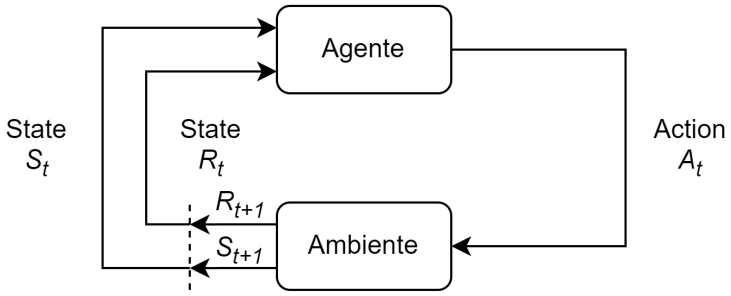


Figura 2.1: Interacción entre agente y ambiente

La estructura fundamental del aprendizaje por reforzamiento se basa en la interacción entre dos componentes principales: el agente y el ambiente. El agente representa la entidad que aprende y toma decisiones, mientras que el ambiente engloba todo lo externo al agente con lo que este interactúa. Esta interacción sigue una secuencia temporal donde en cada paso t, el agente recibe una representación del estado del ambiente $S_t \in S$, selecciona una acción $A_t \in A(S_t)$, y recibe una señal de recompensa R_{t+1} junto con un nuevo estado S_{t+1}.

2.3.1. Algoritmos de Aprendizaje por Reforzamiento

Según estudios de OpenAI [21], los algoritmos de aprendizaje por reforzamiento pueden clasificarse en dos grandes categorías: aquellos libres de modelo (Model-Free

RL) y los basados en modelos (Model-Based RL). Esta taxonomía se puede observar en la Figura 2.2, la cual tiene como objetivo resaltar las opciones de diseño más fundamentales en algoritmos de aprendizaje por reforzamiento.

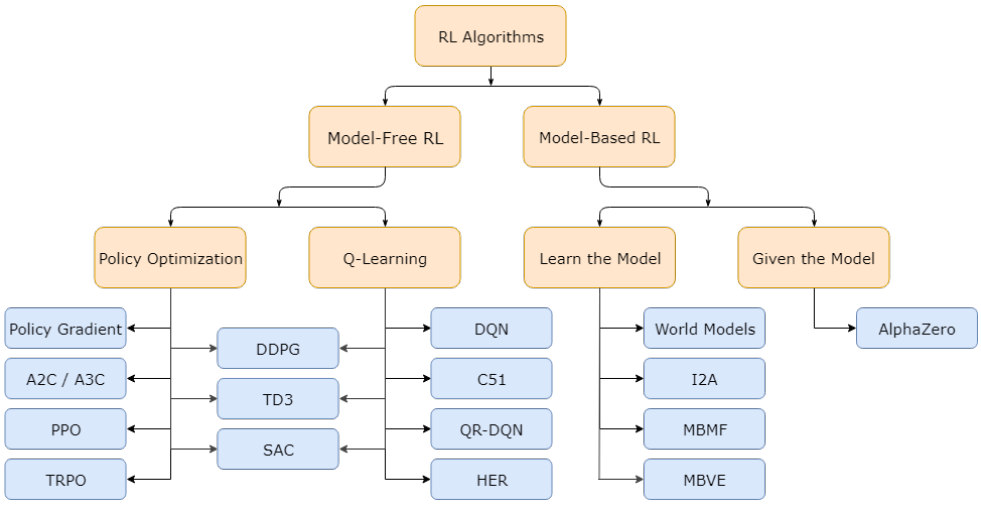


Figura 2.2: Taxonomía no exhaustiva pero útil de algoritmos en RL moderno

El estudio [21], indica que, al momento de el desarrollo de la documentación, septiembre del 2018, los métodos que usan **Model-Free** son más populares y han sido desarrollados y probados más ampliamente que los métodos **Model-Based**. Estos se dividen principalmente en dos enfoques:

Policy Optimization: Esta categoría de algoritmos busca optimizar directamente las políticas (π) que determinan las acciones del agente para maximizar la recompensa total. Existen dos tipos de políticas que mapean el estado s con una acción a:

- Determinística: $\pi(S_t) = A_t$
- Estocástica: $\pi(A_t|S_t) = P_\pi[A = a|S = s]$

Q-Learning: Representa una familia de algoritmos que se centran en aprender una función de valor-calidad (Q-Value). Los componentes principales de este enfoque son:

- Q-Value: Representa la calidad $Q(S_t, A_t)$ de una acción específica en un estado específico

- Q-Table: Una matriz de dimensión $S \times A$ que almacena los Q-Values para cada par estado-acción

- Q-Function: Basada en la ecuación de Bellman [22], selecciona la acción mediante:

$$a(s) = \arg\max_{a} Q_\theta(s, a)$$

Es importante notar que estas aproximaciones no son mutuamente excluyentes. De hecho, existe una variedad de algoritmos que combinan elementos de ambos enfoques, buscando equilibrar sus fortalezas y debilidades. Los métodos de Policy Optimization tienden a ser más estables y confiables en su implementación, mientras que los algoritmos de Q-Learning, cuando están bien implementados, pueden alcanzar un rendimiento superior debido a su mejor utilización de los datos de experiencia.

2.3.2. Herramientas y Frameworks

La implementación práctica de algoritmos de aprendizaje por reforzamiento requiere la construcción de ambientes de simulación que permitan el entrenamiento y evaluación de los agentes. Para este propósito, la comunidad académica y la industria han desarrollado diversos frameworks y herramientas que facilitan la creación de estos entornos de aprendizaje. A continuación, se analizan las principales herramientas disponibles:

Gymnasium de OpenAI

Gymnasium, una evolución de la biblioteca Gym de OpenAI, representa el estándar de facto para la implementación de ambientes de aprendizaje por reforzamiento. Su arquitectura se basa en cuatro funciones fundamentales: make, reset, step y render. En el núcleo de Gymnasium se encuentra la clase env, que implementa el Proceso de Decisión de Markov (MDP), proporcionando una interfaz estandarizada para la interacción entre el agente y el ambiente [23].

TensorFlow Agents

La biblioteca TF-Agents se destaca por su enfoque modular y su capacidad de integración con el ecosistema de TensorFlow. Esta herramienta facilita no solo la implementación de ambientes, sino también el desarrollo y evaluación de nuevos algoritmos de RL. Ofrece dos vías de implementación: una en Python puro, privilegiando la facilidad de desarrollo, y otra en TensorFlow nativo, optimizada para eficiencia y paralelización [24].

ReAgent de Meta

ReAgent se distingue por su enfoque en escenarios de optimización a gran escala y aprendizaje offline. Esta herramienta está específicamente diseñada para entornos donde el acceso directo al ambiente de entrenamiento es limitado o imposible. Utiliza técnicas de evaluación de políticas contrafácticas (CPE) para validar nuevas políticas sin necesidad de implementación directa [25].

DeepMind's OpenSpiel

OpenSpiel, desarrollado por DeepMind, proporciona una colección comprehensiva de ambientes y algoritmos orientados a la investigación en RL. Su diseño permite la implementación de una amplia gama de escenarios, desde juegos de suma cero hasta entornos multiagente complejos. Además de los ambientes, incluye herramientas para el análisis de la dinámica de aprendizaje y métricas de evaluación [26].

2.3.3. Proveedores de Agentes

La implementación de algoritmos de aprendizaje por reforzamiento requiere un considerable esfuerzo de desarrollo. Por esta razón, diversas organizaciones han creado bibliotecas que proporcionan implementaciones optimizadas y verificadas de los algoritmos más populares. A continuación, se analizan los principales proveedores:

Stable-Baseline3

Stable-Baseline3 constituye una evolución significativa de las implementaciones originales de OpenAI Baselines [27], proporcionando versiones optimizadas y mejoradas de los algoritmos más relevantes en el ámbito del aprendizaje por reforzamiento. Además, tiene una integración nativa con Gymnasium, lo que le permite trabajar con diferentes tipos de espacios de acción:

- **Box:** Espacios de acción continuos de N dimensiones, ideales para problemas de control continuo donde las acciones pueden tomar cualquier valor dentro de un rango específico.

- **Discrete:** Espacios que permiten la selección de una única acción de un conjunto finito predefinido, comúnmente utilizados en problemas de toma de decisiones categóricas.

- **MultiDiscrete:** Espacios que habilitan la selección simultánea de múltiples acciones discretas, útiles cuando se requiere tomar varias decisiones independientes al mismo tiempo.

- **MultiBinary:** Espacios diseñados para manejar combinaciones arbitrarias de acciones binarias, permitiendo activar o desactivar múltiples opciones de manera independiente.

La siguiente tabla (Figura 2.3) presenta una visión completa de los agentes disponibles en Stable-Baseline3 y su compatibilidad con cada tipo de espacio de acción, proporcionando una guía práctica para seleccionar el algoritmo más apropiado según las características específicas del problema a resolver.

Name	Recurrent	Box	Discrete	MultiDiscrete	MultiBinary	Multi Processing
ARS[1]	✗	✓	✓	✗	✗	✓
A2C	✗	✓	✓	✓	✓	✓
DDPG	✗	✓	✗	✗	✗	✓
DQN	✗	✗	✓	✗	✗	✓
HER	✗	✓	✓	✗	✗	✓
PPO	✗	✓	✓	✓	✓	✓
QR-DQN[1]	✗	✗	✓	✗	✗	✓
RecurrentPPO[1]	✓	✓	✓	✓	✓	✓
SAC	✗	✓	✗	✗	✗	✓
TD3	✗	✓	✗	✗	✗	✓
TQC[1]	✗	✓	✗	✗	✗	✓
TRPO[1]	✗	✓	✓	✓	✓	✓
Maskable PPO[1]	✗	✗	✓	✓	✓	✓

Figura 2.3: Agentes disponibles en Stable-Baseline3

TensorFlow Agents

TF-Agents complementa su framework de desarrollo con implementaciones de algoritmos fundamentales como DQN, DDPG, TD3, PPO y SAC. Su integración nativa con TensorFlow facilita el desarrollo de soluciones escalables, aunque su documentación sobre las capacidades específicas de cada agente es más limitada [28]

Stable-Baselines3 - Contrib

Esta extensión de Stable-Baselines3 sirve como plataforma para algoritmos experimentales y las últimas innovaciones en el campo. Mantiene la filosofía de diseño y documentación de su predecesor, pero enfocándose en implementaciones más recientes y menos probadas [29].

Para el contexto específico de sistemas de decisión, es crucial seleccionar agentes que se adapten a las características del problema. Los siguientes cuadros resumen las capacidades de los agentes disponibles:

	Acción			Estado
Nombre	**Discrete**	**MultiDiscrete**	**MultiBinary**	**MultiDiscrete**
A2C	Si	Si	Si	Si
ACER	Si	No	No	Si
ACKTR	Si	No	No	Si
DQN	Si	No	No	Si
Gail	Si	No	No	Si
PPO	Si	Si	Si	Si
TRPO	Si	Si	Si	Si

Cuadro 2.1: Agentes de Stable-Baseline3

	Acción			Estado
Nombre	**Discrete**	**MultiDiscrete**	**MultiBinary**	**MultiDiscrete**
ARS	Si	No	No	Si
MASKABLE PPO	Si	Si	Si	Si
RECURRENT PPO	Si	Si	Si	Si
QR-DQN	Si	No	No	Si

Cuadro 2.2: Agentes de Stable-Baseline3-Contrib

La selección de estos agentes se fundamenta en su compatibilidad con el diseño del ambiente y su capacidad para manejar tanto los estados como las acciones requeridas en el contexto del problema. En particular, todos los agentes seleccionados soportan estados MultiDiscrete y diversos tipos de acciones, lo que los hace adecuados para problemas de asignación complejos.

2.3.4. Optimizadores de Hiperparámetros

El rendimiento de los algoritmos de aprendizaje por reforzamiento depende significativamente de la configuración de sus hiperparámetros. La búsqueda de valores

óptimos para estos parámetros representa un desafío crucial que ha llevado al desarrollo de diversas herramientas especializadas. A continuación, se analizan los principales optimizadores disponibles:

Optuna

Esta herramienta se distingue por su diseño específico para el aprendizaje automático, funcionando como un optimizador de caja negra basado en funciones objetivo. Implementa una variedad de estrategias de optimización, incluyendo GridSearch, búsqueda aleatoria, algoritmos bayesianos y evolutivos. La función objetivo determina los puntos de muestreo en pruebas subsecuentes y proporciona métricas numéricas del rendimiento de los hiperparámetros [30].

Ray Tune

Como parte del ecosistema Ray, Tune proporciona una API unificada para el desarrollo de aplicaciones distribuidas. Su enfoque en la escalabilidad permite la ejecución eficiente de experimentos y ajuste de hiperparámetros a gran escala, aprovechando algoritmos de optimización estado del arte [31].

HyperOpt

HyperOpt se caracteriza por su capacidad de optimización tanto en serie como en paralelo en espacios de búsqueda complejos. Su implementación de algoritmos de optimización bayesiana permite manejar dimensiones continuas, discretas y condicionales, siendo especialmente efectiva en la optimización de modelos con cientos de hiperparámetros [32].

Scikit-Optimize

Desarrollada por el equipo de Scikit-learn, esta biblioteca de código abierto se destaca por su facilidad de uso e implementación de Optimización Bayesiana de Hiper-

parámetros (BHO). Su principal ventaja radica en la capacidad de encontrar configuraciones óptimas con menor número de iteraciones en comparación con métodos de búsqueda aleatoria [33].

El Cuadro 2.3 presenta una comparación detallada de las características de cada optimizador:

	Optuna	Ray Tune	HyperOpt	Sckit-Optimize
Open Source	Si	Si	Si	Si
Algoritmos	GridSearch, Random Search, Algoritmo Bayesiano y Algoritmo Evolutivo	AX/Botorch, HyperOpt, Optimización Bayesiana	Random Search, TPE Adativo, Tree of Parzen Estimators	Optimizaión Bayesiana de Hiperparámetros
Frameworks soportados	PyTorch	PyTorch	PyTorch	ML ofrecidas por la libreria de scikit-learn
Usa GPUs	No	Si	Si	Si
Optimización distribuida	Si	Si	Si	No
Manejo de grandes cantidades de datos	Si	Si	Si	No

Cuadro 2.3: Tabla comparativa de Optimizadores de Hiperparámetros

2.4. Métricas y Evaluación

La evaluación efectiva de sistemas de asignación automática requiere un enfoque multidimensional que considere tanto aspectos cuantitativos como cualitativos. La literatura académica y la experiencia práctica han identificado diversas métricas fundamentales para evaluar estos sistemas.

Según el estudio de Howard et al. [34] estudios, los sistemas de asignación automática deben evaluarse considerando tres dimensiones principales: la satisfacción de los participantes, la eficiencia operativa y la calidad de las asignaciones resultantes. Esta perspectiva multidimensional permite una evaluación más completa y objetiva del rendimiento del sistema.

- **Satisfacción de los participantes:** Evaluaron la satisfacción mediante encuestas a 22 de 82 residentes usando escala Likert de 5 puntos. Los resultados mostraron mejoras significativas: la satisfacción aumentó de 3.3 ($DE = 1{,}2$) a 4.0 ($DE = 1{,}1$) ($p = 0{,}048$) y la percepción de equidad de 3.3 ($DE = 1{,}4$) a 4.2 ($DE = 1{,}0$) ($p = 0{,}020$) tras implementar AIMS.

- **Eficiencia operativa:** La eficiencia operativa se midió principalmente a través de la reducción de conflictos de turnos, definidos como turnos nocturnos seguidos inmediatamente por turnos diurnos en servicios diferentes. Para internos, los conflictos disminuyeron significativamente de 0.7 a 0.3 conflictos por interno ($p < 0{,}01$), reduciendo la carga sobre el sistema de respaldo y mejorando las transiciones de atención.

- **Calidad de las asignaciones:** La calidad se evaluó mediante el cumplimiento de preferencias y la equidad distributiva. Para internos, las asignaciones a primera opción aumentaron de 69.4 % a 96.0 %. Para residentes superiores, las mejoras fueron de 30.5 % a 80.5 % para primera opción y de 46.3 % a 74.4 % para segunda opción (ambos $p < 0{,}001$). El sistema mantuvo la equidad en rotaciones nocturnas e ICU sin diferencias significativas en varianza.

Capítulo 3

Diseño de la solución

3.1. Análisis

La problemática presentada en el capítulo anterior revela la necesidad de desarrollar una solución que automatice el proceso de formación de equipos y optimice la satisfacción global considerando múltiples restricciones y objetivos. La complejidad del problema sugiere la exploración de diferentes aproximaciones algorítmicas para determinar cuál ofrece mejores resultados.

3.1.1. Alternativas de solución

Considerando el estado del arte y las características específicas del problema, se identificaron aproximaciones principales:

- **Optimización mediante Algoritmos Tradicionales:** Esta aproximación involucraría el uso de algoritmos de optimización combinatoria como Hill Climbing o Algoritmos Genéticos. Si bien estos métodos son relativamente simples de implementar y tienen un comportamiento predecible, su capacidad para manejar la naturaleza dinámica del problema y las múltiples restricciones es limitada.

- **Aprendizaje por Reforzamiento:** Esta aproximación permite que el sistema

aprenda políticas de asignación a través de la interacción con un ambiente simulado. Los agentes PPO y DQN, junto con la optimización de hiperparámetros mediante Optuna, ofrecen la capacidad de encontrar estrategias efectivas de asignación considerando múltiples objetivos simultáneamente.

- **Búsqueda de Árbol Monte Carlo:** Esta alternativa proporciona un método sistemático para explorar el espacio de soluciones, evaluando diferentes secuencias de asignación y seleccionando las más prometedoras. Su capacidad para balancear exploración y explotación la hace particularmente adecuada para problemas de decisión secuencial.

3.1.2. Acercamiento a la solución

Tras evaluar las alternativas, se decidió implementar un servidor que ofrezca endpoints independientes para dos aproximaciones algorítmicas distintas: aprendizaje por reforzamiento y búsqueda de árbol Monte Carlo. Esta decisión se fundamenta en varios factores:

- **Comparación Objetiva:** La implementación separada permite evaluar el rendimiento de cada aproximación bajo las mismas condiciones y métricas, facilitando una comparación directa de sus fortalezas y debilidades.

- **Flexibilidad Operativa:** El diseño con endpoints independientes permite a los administradores seleccionar el método más apropiado según sus necesidades específicas o restricciones temporales.

- **Validación Experimental:** La arquitectura facilita la evaluación empírica de cada método utilizando métricas como satisfacción promedio, distribución de preferencias, y balance de equipos.

- **Escalabilidad:** La separación de endpoints permite optimizar y escalar cada método de forma independiente, además de facilitar la futura incorporación de nuevos algoritmos.

El sistema se alojará en un servidor que expone APIs REST separadas para cada método, permitiendo que los administradores elijan entre las soluciones proporcionadas por el aprendizaje por reforzamiento o la búsqueda de árbol Monte Carlo, según sus requerimientos específicos.

3.2. Diseño

3.2.1. Requisitos del Sistema

Requerimientos Funcionales

RF1 El sistema debe automatizar la distribución de desafíos entre los estudiantes basándose en las preferencias y restricciones especificadas.

RF2 El sistema debe generar equipos multidisciplinarios según los parámetros proporcionadas al algoritmo asegurando una distribución igual o mejor al manual.

RF3 El sistema debe permitir a los administradores gestionar y actualizar la lista de desafíos disponibles.

RF4 El sistema debe permitir a los administradores establecer y modificar las restricciones y criterios de formación de equipos.

RF5 El algoritmo debe proporcionar resultados preliminares para evaluar asignaciones previas a su publicación.

Requerimientos No Funcionales

RNF1 El algoritmo debe ser robusto y tolerante a fallos, garantizando la integridad de los datos en caso de interrupciones.

RNF2 El sistema debe ser escalable para soportar un aumento en la cantidad de usuarios y datos sin degradar el rendimiento.

RNF3 El algoritmo debe tener una interfaz de configuración amigable para que los administradores puedan ajustar parámetros sin necesidad de intervención técnica.

RNF4 La interfaz del sistema debe ser intuitiva y fácil de usar para todos los tipos de usuarios.

3.2.2. Diagrama de Contexto

La solución contempla tres tipos de usuarios: **administradores**, **estudiantes** y **organizaciones**, cada uno con formas específicas de interacción con el sistema.

Como se ilustra en la figura 3.1, el diagrama de contexto muestra el flujo de interacciones dentro de la Plataforma del Programa de Memorias Multidisciplinarias:

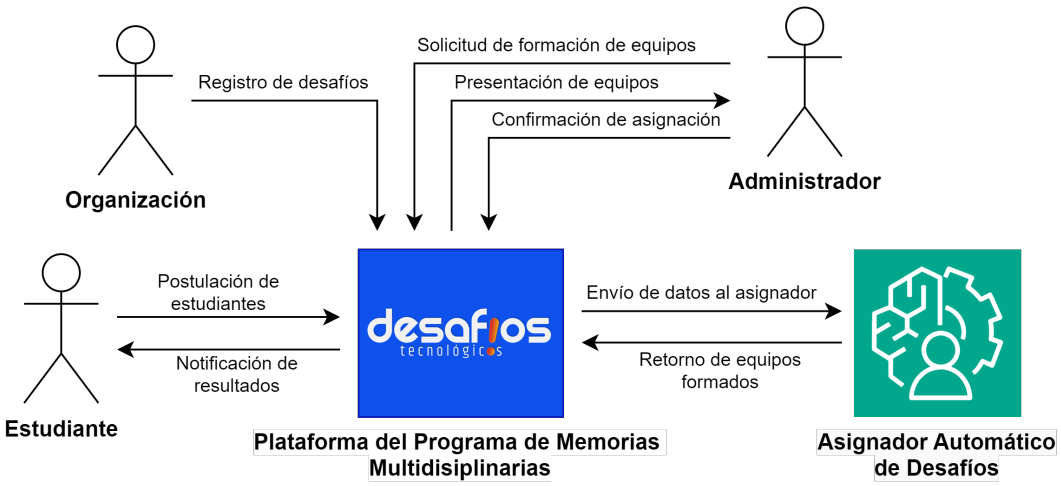


Figura 3.1: Diagrama de Contexto

1. **Registro de desafíos:** Las organizaciones registran sus desafíos tecnológicos directamente en la Plataforma del Programa de Memorias Multidisciplinarias.

2. **Postulación de estudiantes:** Los estudiantes se postulan a los desafíos disponibles a través de la plataforma.

3. **Solicitud de formación de equipos:** El administrador solicita a la plataforma la formación de equipos basándose en parámetros configurables.

4. **Envío de datos al asignador:** La plataforma envía las postulaciones, desafíos y parámetros al **Asignador Automático de Desafíos**.

5. **Retorno de equipos formados:** El **Asignador Autmático de Desafíos** le entrega a la plataforma una propuesta de los equipos multidisciplinarios.

6. **Presentación de equipos:** La plataforma entrega los equipos propuestos al administrador para su revisión.

7. **Confirmación de asignación:** El administrador confirma la asignación de los equipos en la plataforma.

8. **Notificación de resultados:** La plataforma entrega a cada estudiante la información sobre su equipo asignado y el desafío correspondiente.

De esta manera, el sistema conecta las organizaciones con necesidades tecnológicas y equipos multidisciplinarios de estudiantes capacitados para resolverlas.

3.2.3. Diagrama de Arquitectura

El sistema se diseñó con un enfoque modular para abordar los requisitos específicos del PMM: escalabilidad, mantenibilidad, y flexibilidad para incorporar múltiples algoritmos de asignación. La arquitectura resultante se compone de cinco módulos principales que separan claramente las responsabilidades (Figura 3.2).

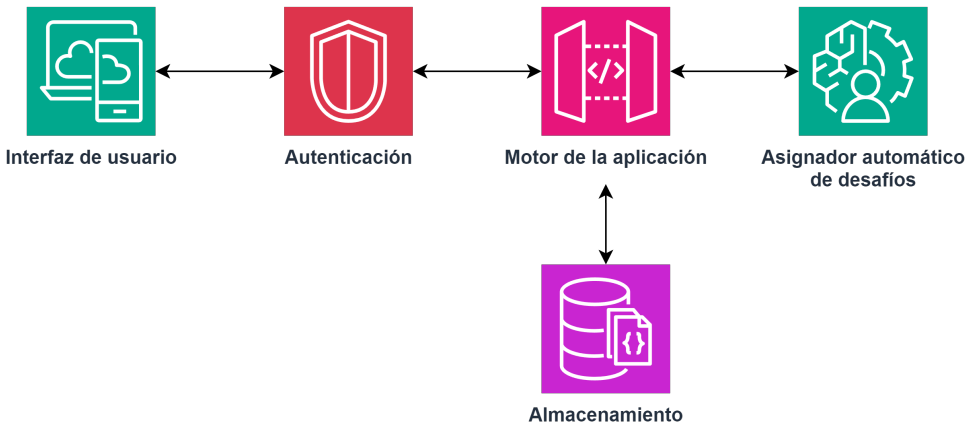


Figura 3.2: Diagrama de Arquitectura

Cada módulo mantiene responsabilidades específicas sin acoplamiento fuerte. El flujo típico inicia en la Interfaz de Usuario, pasa por Autenticación hacia el Motor de la Aplicación, que coordina con Almacenamiento y delega la asignación al módulo Asignador Automático.

Interfaz de Usuario

La interfaz de usuario constituye el punto primario de interacción entre el usuario y la aplicación a través de una interfaz gráfica web. Su propósito fundamental es

facilitar la navegación del usuario a través de las distintas secciones del servicio y permitir la gestión de postulaciones. Este módulo depende directamente del módulo de autenticación, que determina qué funcionalidades se presentan al usuario según sus permisos. El módulo recibe como entrada la información de credenciales del usuario (correo electrónico y contraseña) y produce como salida la información y funcionalidades correspondientes al tipo de usuario autenticado.

Autenticación

El módulo de autenticación gestiona el acceso de usuarios a la plataforma y determina sus niveles de permiso. Este componente permite a los usuarios registrarse, iniciar sesión, y mantiene el control sobre los permisos de acceso a las diferentes funcionalidades. Depende de la interfaz de usuario para obtener los datos de sesión y opera bajo la restricción de permitir solo una sesión activa por usuario. El módulo procesa los datos de registro e inicio de sesión como entrada y produce como salida la información validada del usuario conectado, incluyendo sus permisos y accesos autorizados.

Motor de la Aplicación

El motor de la aplicación actúa como el núcleo coordinador del sistema, gestionando la lógica de negocio y orquestando las interacciones entre los diferentes módulos. Este componente procesa las solicitudes del frontend, coordina con el módulo de almacenamiento para obtener los datos necesarios, y delega las tareas de asignación al Asignador Automático de Desafíos según los parámetros configurados por el administrador.

Almacenamiento

El módulo de almacenamiento actúa como el repositorio central de datos del sistema, gestionando las postulaciones, características de los desafíos y configuraciones del distribuidor. Permite a los estudiantes revisar y gestionar sus postulaciones, mientras

que los administradores pueden gestionar todos los aspectos del sistema. El módulo depende de la interfaz de usuario y la autenticación para funcionar correctamente. Los estudiantes pueden acceder únicamente a sus propias postulaciones, mientras que los administradores tienen acceso completo a todos los datos. El módulo procesa las entradas de postulaciones, desafíos y configuraciones, proporcionando como salida el estado actualizado de estos elementos.

Asignador Automático de Desafíos

Este módulo constituye el componente central para la formación automatizada de equipos multidisciplinarios. Su evolución durante el desarrollo se detalla en la sección 3.2.4, donde se describe la transición desde un diseño centrado en simulación hacia una arquitectura modular y extensible.

3.2.4. Evolución del Módulo Asignador Automático de Desafíos

El diseño inicial del módulo Asignador Automático de Desafíos contemplaba una arquitectura centrada en el paradigma del aprendizaje por reforzamiento. Esta aproximación incluía un componente simulador que interactuaría con diferentes agentes, cuyas funciones principales eran:

- Gestionar los espacios de observación y acción para el entrenamiento de agentes
- Calcular recompensas basadas en las preferencias estudiantiles y restricciones del programa
- Mantener el estado del ambiente durante las asignaciones secuenciales
- Proporcionar retroalimentación para el proceso de aprendizaje iterativo

Durante el desarrollo se identificó que esta arquitectura introducía complejidad innecesaria para algoritmos que no operan bajo el paradigma del aprendizaje por reforzamiento, limitando la integración futura de nuevos tipos de asignadores.

Arquitectura Final

La solución adoptada simplificó el diseño eliminando la abstracción del simulador y creando el **Servidor Selector de Asignador**, que actúa como intermediario entre el Motor de la Aplicación y los diversos algoritmos de asignación (Figura 3.3).

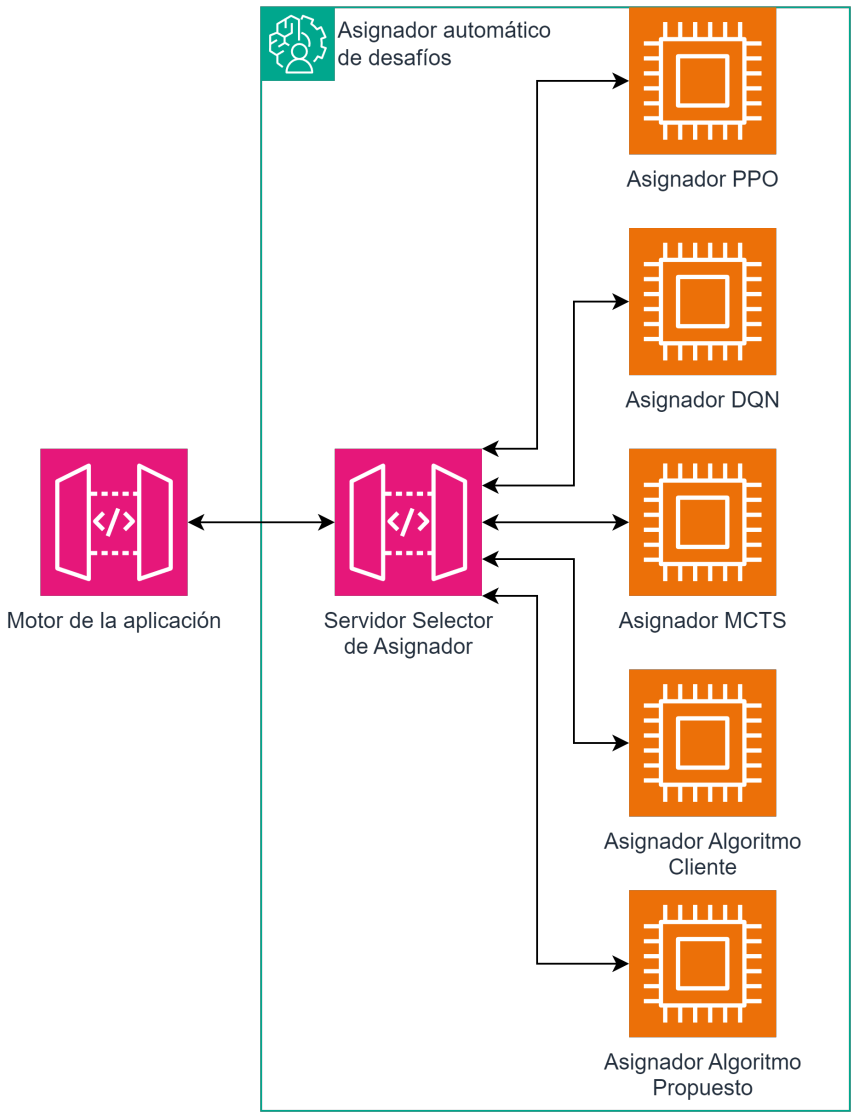


Figura 3.3: Arquitectura del Sistema de Asignación de Equipos

El Servidor Selector de Asignador constituye el núcleo del sistema con las siguientes responsabilidades:

- Implementar APIs REST para la comunicación con el Motor de la Aplicación
- Gestionar la distribución de datos hacia el asignador seleccionado
- Procesar y evaluar los resultados mediante métricas predefinidas
- Mantener una arquitectura extensible para la incorporación modular de nuevos algoritmos

Este componente interactúa con cinco tipos de asignadores especializados: PPO, DQN, MCTS, Algoritmo Cliente y Algoritmo Propuesto, los cuales se detallan en la sección 3.2.5.

Se implementa en Python con el framework Flask garantizando compatibilidad con todos los asignadores y facilita futuras extensiones sin modificaciones estructurales.

3.2.5. Asignadores de Desafíos

Asignador PPO

El Asignador PPO implementa un sistema de formación de equipos multidisciplinarios basado en el algoritmo Proximal Policy Optimization (PPO). Su función principal es optimizar la formación de equipos multidisciplinarios mediante técnicas de aprendizaje por refuerzo, interactuando exclusivamente con el Servidor Selector de Asignador.

Para su implementación se ha utilizado Python como lenguaje base, integrando bibliotecas especializadas como Gymnasium de OpenAI para la creación del entorno de aprendizaje, Stable-Baselines3 para los algoritmos de reinforcement learning, y Optuna para la optimización automatizada de hiperparámetro

Asignador DQN

Este componente implementa un sistema de formación de equipos multidisciplinarios utilizando la arquitectura Deep Q-Network (DQN). Al igual que el asignador PPO, su objetivo principal es optimizar la formación de equipos multidisciplinarios,

pero empleando técnicas de aprendizaje profundo por refuerzo. Mantiene interacción únicamente con el Servidor Selector de Asignador. La implementación también se ha realizado en Python, compartiendo el mismo stack tecnológico que el Asignador PPO: las bibliotecas Gymnasium de OpenAI, Stable-Baselines3 y Optuna para la optimización de hiperparámetros, lo que facilita la consistencia y mantenibilidad del código entre ambos asignadores basados en reinforcement learning.

Asignador MCTS

El Asignador MCTS implementa un sistema de formación de equipos multidisciplinarios basado en el algoritmo de Búsqueda de Árbol Monte Carlo (Monte Carlo Tree Search). Su función principal es optimizar la formación de equipos multidisciplinarios mediante técnicas de búsqueda heurística, ofreciendo un enfoque diferente a los métodos de aprendizaje por refuerzo de PPO y DQN. Este componente interactúa exclusivamente con el Servidor Selector de Asignador y ha sido desarrollado completamente en Python, aprovechando la versatilidad del lenguaje y su facilidad de implementación para algoritmos de búsqueda complejos.

Asignador Algoritmo Cliente

Este componente representa la implementación del algoritmo que fue utilizado en la iteración 2024 del programa de memorias multidisciplinarias. Su función principal es establecer una línea base confiable para la formación de equipos multidisciplinarios, basándose en la metodología existente y probada en producción. El Algoritmo Cliente interactúa únicamente con el Servidor Selector de Asignador y ha sido desarrollado en Python, priorizando la eficiencia en el procesamiento de datos y la facilidad de mantenimiento del código heredado.

Asignador Algoritmo Propuesto

El Asignador Algoritmo Propuesto constituye la implementación de un algoritmo alternativo para la formación de equipos, diseñado específicamente como una propuesta de mejora al sistema actual. Su función principal es proporcionar una solución alternativa optimizada para la formación de equipos multidisciplinarios, incorporando mejoras identificadas durante el análisis del sistema existente. Este componente mantiene interacción exclusiva con el Servidor Selector de Asignador y ha sido desarrollado en Python, maximizando tanto la eficiencia computacional como la mantenibilidad del código, facilitando futuras iteraciones y mejoras sobre la propuesta inicial.

3.2.6. Entorno de Soporte y Desarrollo

La implementación de los componentes especificados requiere un conjunto integral de herramientas de desarrollo y soporte. A continuación, se detallan las herramientas seleccionadas y su justificación técnica:

Entorno de desarrollo integrado (IDE)

Visual Studio Code se ha seleccionado como el entorno de desarrollo principal debido a sus características superiores para este proyecto. Este IDE destaca por su extensa biblioteca de extensiones específicas para las tecnologías del proyecto, su compatibilidad multiplataforma, y su rendimiento optimizado en la gestión de proyectos. La capacidad de previsualización en tiempo real y su interfaz intuitiva proporcionan un entorno de desarrollo eficiente que maximiza la productividad del equipo de desarrollo.

Herramientas de Prueba y Desarrollo

- **Postman**: Esta herramienta se utilizará para la validación y prueba de las APIs REST del Servidor Selector de Asignador, permitiendo un desarrollo y depuración eficientes de los endpoints de la aplicación.

- **Jupyter Notebook**: Seleccionado por su entorno interactivo que facilita el desarrollo y entrenamiento iterativo de modelos de aprendizaje por refuerzo. Su capacidad para combinar código, visualizaciones y documentación en un solo entorno lo hace ideal para la fase de experimentación y desarrollo de modelos.

- **Google Colab**: Esta plataforma se utilizará para el entrenamiento de modelos que requieren recursos computacionales intensivos. Su acceso a GPUs T4 permite acelerar significativamente los procesos de entrenamiento, optimizando el tiempo de desarrollo.

Lenguaje de Programación

Python ha sido seleccionado como el lenguaje de programación principal del proyecto debido a su compatibilidad integral con todas las herramientas y frameworks requeridos. Específicamente, proporciona soporte nativo para:

- Biblioteca Optuna para la optimización de hiperparámetros

- Framework Gymnasium de OpenAI para el desarrollo de entornos de aprendizaje por refuerzo

- Stable-Baseline3 para la implementación de algoritmos de aprendizaje por refuerzo

La combinación de estas herramientas forma un ecosistema de desarrollo robusto y eficiente que facilitará la implementación exitosa de todos los componentes del sistema.

Capítulo 4

Implementación de la solución

4.1. Definición de proceso de asignación

Es importante establecer como se irán asignando los estudiantes para influenciar desde este componente los resultados. Los componentes asignadores a desarrollar, estarán obligados a tener en consideración el ordén de entrada de la lista de los estudiantes, por lo que se define que irán asignando de manera secuencial, que desafíos participaran un estudiante tras otro.

4.2. Evaluación de distribuciones

Para evaluar correctamente la calidad de las distribuciones que presenten las distintas técnicas, es necesario hacer uso de métricas que sean representativas del contexto propio del problema, y que apunten a evaluar la calidad de la distribución.

4.2.1. Definición de variables y conjuntos

- $E = \{e_1, e_2, ..., e_n\}$: conjunto de estudiantes
- $D = \{d_1, d_2, ..., d_m\}$: conjunto de desafíos

- $C = \{c_1, c_2, ..., c_k\}$: conjunto de carreras
- P_i: conjunto de desafíos preferidos por el estudiante e_i
- $\text{pref}_i(d_j)$: posición del desafío d_j en las preferencias del estudiante e_i (1 = primera preferencia, 2 = segunda, etc.)
- $x_{i,j}$: variable binaria que vale 1 si el estudiante e_i es asignado al desafío d_j, 0 en caso contrario
- $carrera(e_i)$: carrera del estudiante e_i
- $T_j = \{e_i \in E : x_{i,j} = 1\}$: conjunto de estudiantes asignados al desafío d_j
- L_c: límite máximo de estudiantes de la carrera c por equipo

4.2.2. Métricas de evaluación

- **Satisfacción Promedio:** Esta métrica cuantifica el grado de alineación entre las asignaciones realizadas y las preferencias de los estudiantes. Un estudiante asignado a su primera opción recibe un puntaje de 1, a su segunda opción recibe 0.5, y así sucesivamente de forma recíproca. Las asignaciones a desafíos no declarados como preferencias reciben puntaje 0. Valores cercanos a 1 indican alta correspondencia con las preferencias estudiantiles.

$$\text{Satisfacción Promedio} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \cdot s_{i,j}$$

donde $s_{i,j}$ es la satisfacción del estudiante e_i por el desafío d_j:

$$s_{i,j} = \begin{cases} \frac{1}{\text{pref}_i(d_j)} & \text{si } d_j \in P_i \\ 0 & \text{si } d_j \notin P_i \end{cases}$$

- **Estudiantes en Primera Prioridad:** Esta métrica contabiliza el número absoluto de estudiantes que fueron asignados a su desafío de mayor preferencia. Representa un indicador directo de la capacidad del algoritmo para satisfacer las preferencias principales de los participantes.

$$\text{Primera Prioridad} = \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \cdot \mathbf{1}[\text{pref}_i(d_j) = 1]$$

- **Estudiantes fuera de Preferencias:** Esta métrica identifica el número de estudiantes asignados a desafíos que no figuran en sus listas de preferencias declaradas. Representa una medida de la calidad mínima aceptable de la asignación, ya que valores altos indican que una proporción significativa de estudiantes fue ubicada en desafíos completamente ajenos a sus intereses manifestados.

$$\text{Fuera de Preferencias} = \sum_{i=1}^{n} \sum_{j=1}^{m} \begin{cases} x_{i,j} & \text{si } d_j \notin P_i \\ 0 & \text{en otro caso} \end{cases}$$

- **Desafíos sin Equipo:** Esta métrica cuantifica el número de desafíos que no recibieron ninguna asignación de estudiantes. Un objetivo desde la perspectiva del PMM sería maximizar la cantidad de desafíos con equipos.

$$\text{Desafíos sin Equipo} = \sum_{j=1}^{m} \begin{cases} 1 & \text{si } |T_j| = 0 \\ 0 & \text{si } |T_j| > 0 \end{cases}$$

- **Desviación Estándar tamaño Equipos:** Esta métrica evalúa la uniformidad en el tamaño de los equipos formados. Considerando que los equipos PMM tienen un rango operativo óptimo (típicamente 3 integrantes, con flexibilidad hacia 2 o 4), una baja desviación estándar indica una distribución más equilibrada.

Sea $\mathcal{T} = T_j : |T_j| > 0$ el conjunto de equipos no vacíos, entonces:

$$\text{Media} = \frac{1}{|\mathcal{T}|} \sum_{T_j \in \mathcal{T}} |T_j|$$

$$\text{Desviación Estándar} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{T_j \in \mathcal{T}} (|T_j| - \text{Media})^2}$$

- **Promedio de carreras por Equipo:** Esta métrica evalúa el grado de diversidad disciplinaria alcanzado en los equipos conformados. El carácter multidisciplinario es un componente fundamental del PMM, por lo que se busca maximizar la variedad de perspectivas académicas dentro de cada equipo. El valor se calcula como el promedio del número de carreras distintas representadas en cada equipo formado.

  $$\text{Promedio Carreras} = \frac{1}{|\mathcal{T}|} \sum_{T_j \in \mathcal{T}} |U_j|$$

  donde $U_j = carrera(e_i) : e_i \in T_j$ es el conjunto de carreras únicas en el equipo T_j.

Se puede ver entonces, que las métricas definidas buscan dar enfoques distintos en lo que es el rendimiento y la calidad de la distribución resultante, permitiendo así, de ser necesario, enfocar las distribuciones a maximizar una métrica en específico, o incluso un grupo de ellas, y de esta forma poder medir aspectos propios del problema, como lo son la satisfacción de los estudiantes, el factor de multidisciplinariedad de los equipos formados, la cantidad de estudiantes no satisfechos en base a la distribución, entre otros.

4.3. Servidor Selector de Asignador

4.3.1. Especificación de Entrada

El componente **Servidor Selector de Asignador** constituye una interfaz que recibe solicitudes del componente **Motor de la aplicación** mediante una arquitectura API REST. Esta interfaz implementa múltiples métodos especializados para el enrutamiento de datos hacia asignadores específicos, según se ilustra en la sección 3.2.4.

4.3.2. Estructura de Datos de Entrada

La estructura de entrada del sistema comprende tres *objetos* fundamentales, conforme a lo establecido en la Figura 4.1:

- **Objeto Estudiantes:** Contiene los registros completos de los participantes del proceso de asignación, incluyendo identificación de su carrera, datos básicos y un conjunto ordenado de postulaciones según orden de preferencia.

- **Objeto Desafíos:** Especifica las características técnicas y académicas de cada proyecto disponible, incluyendo la denominación del desafío, las carreras prioritarias requeridas para la conformación del equipo de trabajo, y la consideración de carreras complementarias para garantizar el enfoque multidisciplinario.

- **Objeto Carreras:** Define el marco regulatorio de participación mediante un listado exhaustivo de carreras habilitadas y los límites máximos de participantes por carrera en cada equipo.

```
{
  "estudiantes": [
    {
      "Nombre": "Santiago Lopez",
      "Carrera": "Ingeniería Civil Telemática",
      "Postulaciones": ["Desafio 1","Desafio 2","Desafio 3","Desafio 4"]
    },{
      "Nombre": "Cristian Vega",
      "Carrera": "Ingeniería Civil Telemática",
      "Postulaciones": ["Desafio 1","Desafio 2","Desafio 3"]
    },{
      "Nombre": "Javier Ahumada",
      "Carrera": "Ingeniería Civil Telemática",
      "Postulaciones": ["Desafio 1","Desafio 2"]
    }
  ],
  "desafios": [
    {
      "Titulo": "Desafio 1",
      "Carreras": ["Ingeniería Civil Eléctrica","Ingeniería Civil Telemática"]
    },{
      "Titulo": "Desafio 2",
      "Carreras": ["Ingeniería Civil Eléctrica","Ingeniería Civil Telemática"]
    },{
      "Titulo": "Desafio 3",
      "Carreras": ["Ingeniería Civil Informática","Ingeniería Civil Telemática"]
    },{
      "Titulo": "Desafio 4",
      "Carreras": ["Ingeniería Civil Telemática"]
    }
  ],
  "carreras": [
    {
      "Nombre": "Ingeniería Civil Telemática",
      "Maximo": 1
    },
    {
      "Nombre": "Ingeniería Civil Informática",
      "Maximo": 4
    },
    {
      "Nombre": "Ingeniería Civil Eléctrica",
      "Maximo": 2
    }
  ]
}
```


Figura 4.1: Entrada de todos los métodos del Servidor Selector de Asignador

4.3.3. Procesamiento de Datos de Entrada

Conforme a lo establecido en la sección 4.1, la implementación del sistema adopta un enfoque de asignación secuencial con el objetivo de optimizar el rendimiento computacional y reducir la complejidad del espacio de soluciones. Esta metodología facilita la evaluación comparativa entre diferentes algoritmos de asignación y permite la explotación estratégica de restricciones del problema para generar oportunidades de

optimización basadas en el ordenamiento de estudiantes.

El sistema implementa cuatro estrategias de ordenamiento diferenciadas, diseñadas para atender diversos objetivos institucionales en el proceso de asignación:

- **Ordenamiento Aleatorio**:

  Esta metodología establece una secuencia aleatoria del listado de estudiantes, independiente de factores como preferencias de desafíos, niveles de saturación o distribución por carreras. El enfoque presenta como principales fortalezas la eliminación de posibles sesgos al momento del envió para el proceso de asignación. Sin embargo, esta puede incluir la ausencia de equidad distributiva generando una posible disminución en la métrica de satisfacción de los estudiantes.

- **Ordenamiento por Saturación de Desafíos**:

  Esta metodología organiza la secuencia del listado de estudiantes en función del nivel de demanda de los desafíos postulados. Tiene dos modalidades, la ascendente, que prioriza estudiantes con desafíos de menor demanda, beneficiando las oportunidades de asignación para desafíos con menor demanda. Y la modalidad descendente, que prioriza los desafíos con mayor demanda.

- **Ordenamiento por Tamaño de Cohorte por Carrera**:

  Esta metodología establece la secuencia estudiantil basándose por el número de estudiantes de cada carrera. Tiene dos modalidades, la que prioriza carreras minoritarias garantiza representación equilibrada de carreras con menos estudiantes y puede incrementar la diversidad interdisciplinaria en los equipos, aunque presenta el riesgo de reducción en las métricas de satisfacción de estudiantes pertenecientes a carreras con mayor participación. La modalidad que prioriza carreras mayoritarias optimiza la satisfacción promedio para estudiantes de carreras con mayor representación, pero tiende hacia la marginalización de carreras minoritarias y puede resultar en una potencial reducción de la diversidad grupal.

La arquitectura del sistema permite una configuración flexible con una estrategia de

ordenamiento, delegando la selección metodológica al criterio institucional del cliente. Esta funcionalidad garantiza la adaptabilidad del sistema a los objetivos específicos y la visión estratégica particular de cada programa de memorias multidisciplinarias.

4.3.4. Especificaciones de Endpoints

Considerando las estrategias de preprocesamiento previamente definidas, el componente implementa una arquitectura de endpoints diferenciada según el algoritmo de asignación seleccionado. Cada endpoint aplica una estrategia específica de ordenamiento de datos antes de ejecutar el algoritmo correspondiente.

4.3.5. Endpoints para Asignadores desarrollados

Los asignadores PPO, DQN y MCTS implementan una estructura uniforme de endpoints que aplica diferentes estrategias de preprocesamiento antes de la ejecución del algoritmo correspondiente:

Método	Estructura de Ruta	Descripción
POST	/{asignador}/shuffle_students	Ejecuta el algoritmo seleccionado aplicando ordenamiento aleatorio a la secuencia de estudiantes antes del proceso de asignación.
POST	/{asignador}/sort_students_saturation/asc	Ejecuta el algoritmo seleccionado priorizando estudiantes con postulaciones a desafíos de menor demanda mediante ordenamiento ascendente por saturación.
POST	/{asignador}/sort_students_saturation/desc	Ejecuta el algoritmo seleccionado priorizando estudiantes con postulaciones a desafíos de mayor demanda mediante ordenamiento descendente por saturación.
POST	/{asignador}/sort_students_career/asc	Ejecuta el algoritmo seleccionado aplicando priorización de carreras minoritarias mediante ordenamiento ascendente por número de estudiantes.
POST	/{asignador}/sort_students_career/desc	Ejecuta el algoritmo seleccionado aplicando priorización de carreras mayoritarias mediante ordenamiento descendente por número de estudiantes.

Cuadro 4.1: Estructura de endpoints para algoritmos de aprendizaje automático

Donde `{asignador}` corresponde a los siguientes valores:

- `ppo`: Asignador con algoritmo Proximal Policy Optimization.
- `dqn`: Asignador con algoritmo Deep Q-Network.
- `mcts`: Asignador con algoritmo Monte Carlo Tree Search.

4.3.6. Endpoints para Algoritmos Externos

Los algoritmos externos implementan heurísticas específicamente diseñados para el problema de asignación multidisciplinaria. Dentro de las necesidades de este servidor de selector de asignador, es necesario que pueda añadir otros algoritmos.

Método	**Ruta**	**Descripción**
POST	/algoritmo/propuesta	A partir de la lista de estudiantes, se genera un orden aleatorio de estudiantes para ser asignados a los distintos desafíos, con tal de ir iterando por cada uno de ellos, revisando sus preferencias y siendo asignados a un desafío en particular, siempre que este grupo cumpla con los requisitos necesarios para ingresar al estudiante, y de esta manera conformar los grupos de trabajo.
POST	/algoritmo/cliente	A partir de la lista de desafíos y sus postulaciones por parte de los estudiantes, se realiza un ordenamiento de los desafíos en base a la cantidad de postulaciones que tengan, con tal de conformar desafío a desafío, equipos PMM, modificando las preferencias de los estudiantes durante el proceso.

Cuadro 4.2: Endpoints para algoritmos heurísticos externos

4.3.7. Especificación de Respuesta

El componente **Motor de la aplicación**, al hacer la solicitud correspondiente con los datos necesarios para el **Servidor Selector de Asignador**, recibirá como respuesta información correspondiente a las métricas definidas en la sección 4.2, y genera una lista estructurada que contiene las distribuciones de los equipos.

La estructura de datos de respuesta del sistema se encuentra definida mediante cuatro atributos, como se presenta en la figura 4.2:


```
{
  "status": "success",
  "message": "Procesamiento completado",
  "metrics": {
    "satisfaccion_promedio": 0.7786458333333333,
    "estudiantes_primera_prioridad": 45,
    "estudiantes_fuera_preferencias": 8,
    "estudiantes_sin_desafios": 0,
    "desafios_sin_equipo": 18,
    "std_tamano_equipos": 0.6782329983125269,
    "promedio_carreras_por_equipo": 1.65,
    "total_equipos": 20,
    "tamano_promedio_equipo": 3.2,
    "equipos_tamano_1": 0,
    "equipos_tamano_2": 3,
    "equipos_tamano_3": 10,
    "equipos_tamano_4": 7,
    "equipos_tamano_5_o_mas": 0,
    "equipos_exceden_limite_carrera": 0
  },
  "results": [
    {
      "Nombre": "Cristian Vega",
      "Desafio": "Desafio 1"
    },{
      "Nombre": "Santiago Lopez",
      "Desafio": "Desafio 2"
    },{
      "Nombre": "Javier Ahumada",
      "Desafio": "Desafio 3"
    }
  ]
}
```


Figura 4.2: Estructura de respuesta del componente Servidor Selector de Asignador

El objeto **Estado (status)** constituye un campo de control que especifica el resultado de la operación ejecutada por el sistema. Este campo adopta valores booleanos representados como `success` para indicar la finalización exitosa del procesamiento, o `error` cuando se presentan fallas durante la ejecución del algoritmo de asignación.

El objeto **Mensaje (message)** proporciona retroalimentación textual descriptiva sobre el estado del procesamiento realizado. Para operaciones exitosas, el sistema retorna la cadena de caracteres "`Procesamiento completado`", mientras que en escenarios de error se genera un mensaje con la estructura: "`Error durante el procesamiento: <ERROR>`", donde `<ERROR>` corresponde a la información detallada sobre la naturaleza del error encontrado.

El objeto **Métricas (metrics)** encapsula los valores numéricos resultantes del cálculo de las métricas de evaluación de equipos, tal como fueron establecidas en las especificaciones funcionales del sistema. Esta entidad mantiene la trazabilidad de los indicadores de calidad de la asignación generada.

El objeto **Resultados** representa una colección estructurada que contiene la asignación final de estudiantes a sus respectivos desafíos. Su estructura sigue el esquema de objetos JSON, donde cada elemento se define como:

```
{
    "Nombre": "identificador_estudiante",
    "Desafio": "identificador_desafio"
}
```

Conformando así un arreglo de diccionarios con la distribución óptima calculada por el asignador.

4.4. Algoritmos de Aprendizaje por Reforzamiento

Durante el desarrollo de diversas iteraciones del ambiente para el Algoritmo de Aprendizaje por Reforzamiento, se encontró una problemática propio al desafío: la naturaleza dinámica de los parámetros del programa de memorias multidisciplinarias. La cantidad de estudiantes, desafíos disponibles, preferencias individuales y limitaciones institucionales experimentan variaciones significativas en cada iteración del programa, requiriendo que el ambiente de simulación se adapte continuamente a estas reglas cambiantes.

Esta variabilidad impone restricciones importantes en la implementación del Aprendizaje por Reforzamiento, dado que tanto el espacio de observación como el espacio de acción dependen directamente de estas reglas dinámicas. En consecuencia, un agente entrenado para un conjunto específico de reglas y limitaciones no puede ser utilizado efectivamente para predecir comportamientos óptimos en ambientes con configuraciones diferentes, ya sea con distintos estudiantes, desafíos, o restricciones institucionales.

La solución propuesta contempla el entrenamiento de un agente específico para cada configuración particular del ambiente, permitiendo que el sistema aprenda las reglas específicas de cada iteración. Si bien esta aproximación es técnicamente viable considerando que la formación de equipos ocurre anualmente y los resultados pueden entregarse días después de completarse el proceso de postulaciones, presenta desafíos significativos en términos de escalabilidad y costos operacionales.

La implementación de esta solución requiere recursos computacionales dedicados para cada proceso de entrenamiento, lo que limita considerablemente la escalabilidad del sistema. En el contexto de una potencial expansión del programa PMM hacia otras universidades y contextos institucionales, el modelo actual resultaría prohibitivamente costoso. Cada nueva implementación requeriría infraestructura computacional independiente y procesos de entrenamiento específicos, generando una curva de costos que crece linealmente con el número de instituciones participantes. Esta limitación representa un obstáculo crítico para la adopción masiva de la solución y requiere el desarrollo de alternativas arquitecturales que permitan una mayor eficiencia en el uso de recursos computacionales compartidos.

4.4.1. Ambiente de Simulación

La implementación del prototipo demandó el desarrollo de un ambiente de simulación específicamente diseñado bajo la arquitectura de Gymnasium (Ver sección 2.3.2). Esta elección permite la integración con los agentes proporcionados por Stable-Baseline3, definidos en la sección 2.3.3. El ambiente desarrollado incorpora los si-

guientes componentes:

- **Función de Inicialización:** Su propósito es instanciar el ambiente de simulación, definiendo las características del entorno. El espacio de observación representa el estado S, el conjunto de todos los estados posibles $S_t \in S$. Para este contexto, se decidió definir el estado como un objeto `State` que contiene:

  - *Asignaciones:* Un diccionario que mapea los títulos de los desafíos a una lista de nombres de estudiantes asignados.
    Por ejemplo:

    ```
    {
        "Desafío 1": ["Estudiante 1", "Estudiante 2"],
        "Desafío 2": ["Estudiante 3"],
        ...
    }
    ```

  - *Desafíos Disponibles:* Un diccionario que mapea los nombres de las carreras a una lista de títulos de desafíos disponibles para esa carrera.
    Por ejemplo:

    ```
    {
        "Carrera 1": ["Desafío 1", "Desafío 2"],
        "Carrera 2": ["Desafío 1", "Desafío 3"],
        ...
    }
    ```

  - *Grupos Incompletos:* Un diccionario que mapea los títulos de los desafíos al tamaño actual de los grupos incompletos (grupos con menos estudiantes que el mínimo requerido).
    Por ejemplo:

    ```
    {
        "Desafío 1": 2,
        "Desafío 3": 1,
        ...
    }
    ```

- *Conteo de Carreras:* Un diccionario que mapea los títulos de los desafíos a otro diccionario, que a su vez mapea los nombres de las carreras a la cantidad de estudiantes de esa carrera en el desafío.
  Por ejemplo:

El **espacio de acción** A se define como un espacio discreto con un tamaño igual al número de desafíos. Las acciones representan el índice del desafío a asignar al estudiante actual.

El **espacio de observación** se define como un espacio discreto con un tamaño igual al número de estudiantes. Las observaciones representan el índice del estudiante actual a asignar.

Al definir los rangos de acciones y observaciones de está manera nos permiten disminuir el proceso de calculo para obtener las posibles acciones que puede realizar el agente, la obtención de recompensas y la acción de paso.

- **Función de Paso:** Esta función define lo que sucede en el simulador al recibir una acción A_t. Los pasos se pueden visualizar en la Figura 4.3, y están explicadas a continuación:

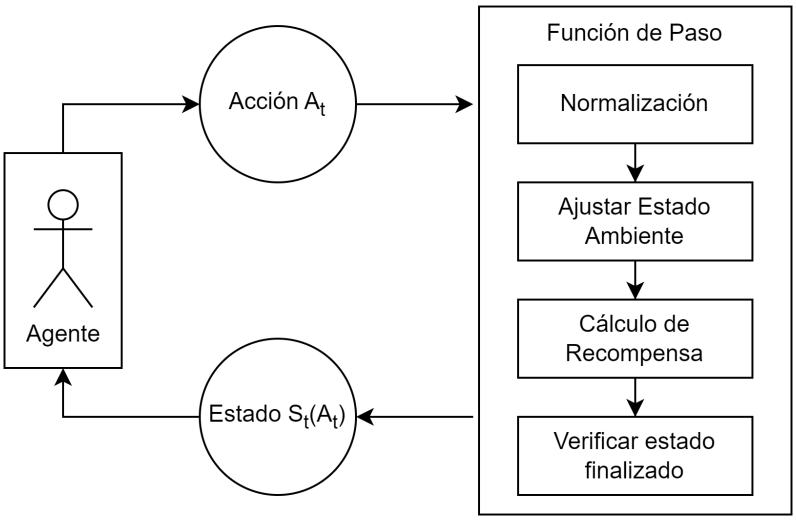


Figura 4.3: Función de Paso

1. Normaliza la acción para obtener el título del desafío correspondiente. Si no hay acciones válidas disponibles, el episodio termina.

   La normalización se realiza de la siguiente manera: primero se obtiene la lista de acciones válidas usando la función `get_valid_actions()`. Luego, se calcula un índice normalizado basado en el valor de la acción y el tamaño del espacio de acciones. Este índice normalizado se utiliza para seleccionar el desafío correspondiente de la lista de acciones válidas.

   Por ejemplo, si el espacio de acción tiene un tamaño de 10 (es decir, hay 10 desafíos en total), pero solo los desafíos en los índices 2, 4 y 7 son válidos para el estudiante actual, y el agente toma la acción 6, el índice normalizado sería:

   $$\text{normalized\_idx} = \lfloor \frac{6 \times 3}{10} \rfloor = 1$$

   Entonces, el desafío seleccionado sería el correspondiente al índice 1 en la lista de acciones válidas, que es el índice 4 en la lista completa de desafíos.

2. Aplica la acción actualizando los diccionarios de estado definidos anteriormente (`assignments`, `career_counts`, `incomplete_groups`, `available_challenges`).

   Por ejemplo, supongamos que el estudiante actual "Estudiante 4" de la carrera "Carrera 2" es asignado al "Desafío 3". Los diccionarios de estado se actualizarían de la siguiente manera:

   a) `assignments`: Se agregaría "Estudiante 4" a la lista de estudiantes asignados a "Desafío 3".

   b) `career_counts`: Se incrementaría en 1 el conteo de estudiantes de "Carrera 2" en "Desafío 3".

   c) `incomplete_groups`: Si el tamaño del grupo de "Desafío 3" aún es menor que el mínimo requerido, se actualizaría el valor correspondiente.

   d) `available_challenges`: Si el grupo de "Desafío 3" alcanza el tamaño máximo, se eliminaría "Desafío 3" de las listas de desafíos disponibles para todas las carreras. Además, si se alcanza el límite de estudiantes de "Carrera 2" en "Desafío 3", se eliminaría "Desafío 3" de la lista de desafíos disponibles para "Carrera 2".

3. Calcula la recompensa basada en las preferencias del estudiante y el desafío asignado.

   La función de recompensa `_calculate_reward()` otorga una recompensa positiva si el desafío asignado está en las preferencias del estudiante (mayor recompensa para las preferencias más altas) y una recompensa negativa si el desafío asignado no está en las preferencias del estudiante.

   Por ejemplo, si las preferencias del estudiante son ["Desafío 1", "Desafío

3", "Desafío 2"], y se le asigna el "Desafío 3", la recompensa sería:

$$\begin{aligned} &\text{preference\_index} = 1 \\ &\text{preference\_reward} = \frac{1{,}0}{(\text{preference\_index} + 1)} \times 2 = 1{,}0 \\ &\text{reward} = 1{,}0 \end{aligned}$$

En cambio, si se le asigna un desafío que no está en sus preferencias, como el "Desafío 4", la recompensa sería:

$$\text{reward} = -0{,}5$$

4. Verifica si el episodio ha terminado (todos los estudiantes han sido asignados).

   Esto se hace simplemente comparando el índice del estudiante actual con el número total de estudiantes. Si el índice actual es igual o mayor que el número total de estudiantes, significa que todos los estudiantes han sido asignados y el episodio ha terminado.

5. Devuelve el nuevo estado (índice del estudiante actual), la recompensa, la bandera `done` y un diccionario de información.

   El diccionario de información incluye detalles como el desafío asignado, el nombre del estudiante actual, el número de acciones válidas disponibles y el número total de asignaciones realizadas hasta el momento.

   Por ejemplo, una posible respuesta de la función de paso podría ser:

```
(
    5,
    0.5,
    False,
    {
      "assigned_challenge": "Desafío 3",
      "current_student": "Estudiante 6",
      "num_valid_actions": 2,
      "total_assignments": 20
    }
 )
```

Esto indica que el nuevo estado es el estudiante con índice 5, la recompensa obtenida es 0.5, el episodio aún no ha terminado, el desafío asignado es el "Desafío 3", el estudiante actual es el "Estudiante 6", hay 2 acciones válidas disponibles y se han realizado 20 asignaciones en total hasta el momento.

- **Función de Reinicio:** Diseñada para preparar el simulador para un nuevo episodio. Limpia las variables de estado, especialmente el diccionario `assignments`, para que se puedan asignar nuevos equipos en el próximo episodio. También incrementa el contador de episodios.

  Esta función se llama automáticamente cuando se detecta que el episodio actual ha terminado, ya sea porque todos los estudiantes han sido asignados o porque se ha alcanzado un estado terminal.

  Por ejemplo, supongamos que el estado actual del ambiente es el siguiente:

```
State(
        assignments={
            "Desafío 1": ["Estudiante 1", "Estudiante 2", "Estudiante 3"],
            "Desafío 2": ["Estudiante 4", "Estudiante 5"],
            ...
        },
        available_challenges={
            "Carrera 1": [],
            "Carrera 2": ["Desafío 3"],
            ...
        },
        incomplete_groups={},
        career_counts={
            "Desafío 1": {
                "Carrera 1": 2,
                "Carrera 2": 1
            },
            "Desafío 2": {
                "Carrera 1": 1,
                "Carrera 3": 1
            },
            ...
        },
        ...
)
```

  Después de llamar a la función `reset()`, el estado se reiniciaría a:

```
State(
  assignments={},
  available_challenges={
      "Carrera 1": ["Desafío 1", "Desafío 2", "Desafío 3", ...],
      "Carrera 2": ["Desafío 1", "Desafío 2", "Desafío 3", ...],
      ...
  },
  incomplete_groups={},
  career_counts={},
  ...
)
```

El diccionario `assignments` se vacía, las listas de desafíos disponibles para cada carrera se reinician a su estado original (conteniendo todos los desafíos), y los diccionarios `incomplete_groups` y `career_counts` también se vacían.

Además, el contador de episodios se incrementaría en 1, indicando que se está comenzando un nuevo episodio de entrenamiento o evaluación.

4.4.2. Proximal Policy Optimization

El algoritmo PPO, al igual que otros métodos de aprendizaje por refuerzo, entrena una red neuronal que actúa como la política del agente. Esta red neuronal mapea las observaciones del ambiente a las acciones que el agente debe tomar para maximizar la recompensa acumulada. El entrenamiento ajusta los pesos de la red neuronal para optimizar la política y mejorar la toma de decisiones del agente en función de las recompensas obtenidas.

Debido a que seleccionamos la biblioteca Stable-Baselines3, solamente necesitamos instanciar el modelo y entregarle el ambiente descrito en la sección anterior.

4.4.3. Deep Q-Network

El algoritmo DQN se diferencia de PPO en su enfoque fundamental de aprendizaje. Mientras PPO utiliza un método basado en política que optimiza directamente la política del agente, DQN emplea un método basado en valor que aprende a estimar el valor Q (calidad) de cada par estado-acción. DQN utiliza una red neuronal profunda para

aproximar la función Q, que predice la recompensa futura esperada para cada acción posible en un estado dado.

Una característica distintiva de DQN es su uso de dos innovaciones clave: el buffer de repetición de experiencia y la red objetivo. El buffer almacena transiciones pasadas (estado, acción, recompensa, siguiente estado) que pueden ser remuestreadas durante el entrenamiento, lo que mejora la estabilidad del aprendizaje. La red objetivo, una copia congelada de la red principal que se actualiza periódicamente, ayuda a reducir la correlación entre las predicciones y los objetivos durante el entrenamiento.

Y al igual que el modelo de PPO, solamente necesitamos instanciar el modelo y entregarle el ambiente, aprovechando la implementación estandarizada proporcionada por Stable-Baselines3.

4.4.4. Optimización de Hiperparámetros

Para optimizar el rendimiento de los algoritmos de aprendizaje por refuerzo, se utilizó la biblioteca Optuna para realizar una búsqueda sistemática de hiperparámetros. Se definieron rangos específicos para cada parámetro basados en las mejores prácticas y la literatura existente.

Configuración de PPO

Para el algoritmo PPO, se realizó la búsqueda de hiperparámetros considerando tanto los aspectos fundamentales del aprendizaje como los parámetros específicos del algoritmo. Los parámetros optimizados se presentan detalladamente en el Cuadro 4.3.

Parámetro	Nombre Técnico	Rango de Valores
Tasa de aprendizaje	learning_rate	1e-5 - 1e-3
Pasos de entrenamiento	n_steps	32 - 2048
Tamaño del lote	batch_size	32 - 256
Épocas de entrenamiento	n_epochs	5 - 20
Factor de descuento	gamma	0.9 - 0.9999
Lambda GAE	gae_lambda	0.9 - 1.0
Rango de recorte	clip_range	0.1 - 0.4
Coeficiente de entropía	ent_coef	0.0 - 0.01
Coeficiente de función de valor	vf_coef	0.1 - 0.9
Norma máxima del gradiente	max_grad_norm	0.3 - 0.9

Cuadro 4.3: Rango de parámetros de configuración para PPO.

Configuración de DQN

Para el algoritmo DQN, se enfocó la optimización en los parámetros críticos que afectan el proceso de aprendizaje y la exploración del agente. La configuración específica de estos hiperparámetros se detalla en el Cuadro 4.4.

Parámetro	Nombre Técnico	Rango de Valores
Tasa de aprendizaje	learning_rate	1e-5 - 1e-2
Tamaño del buffer de experiencia	buffer_size	5,000 - 100,000
Tamaño del lote	batch_size	32, 64, 128
Factor de descuento	gamma	0.8 - 0.99
Tasa de decaimiento épsilon	epsilon_decay	0.9 - 0.9999

Cuadro 4.4: Rango de parámetros de configuración para DQN.

La configuración de búsqueda de hiperparámetros permite una exploración exhaustiva del espacio de configuraciones posibles, buscando optimizar el rendimiento de am-

bos algoritmos para la tarea específica de formación de equipos multidisciplinarios.

4.5. Monte Carlo Tree Search

La asignación de estudiantes a desafíos en nuestro contexto, como se mencionó anteriormente, presenta varios desafíos que pueden limitar el aprendizaje por refuerzo. Uno de los principales desafíos es la naturaleza dinámica del problema, donde la cantidad de estudiantes, desafíos, preferencias y limitaciones pueden variar en cada iteración del programa. Esto requiere que el ambiente de simulación se adapte constantemente a estas reglas cambiantes, lo que dificulta la aplicación directa de RL. En RL, el agente aprende a través de la interacción continua con un ambiente estático, donde el espacio de observación y el espacio de acción permanecen constantes. Sin embargo, en este caso, estos espacios son dependientes de las reglas cambiantes, lo que significa que un agente entrenado para un conjunto específico de reglas puede no ser efectivo cuando se enfrenta a un ambiente diferente con otras reglas, limitaciones, desafíos y estudiantes.

Considerando estas limitaciones, el algoritmo de Monte Carlo Tree Search (MCTS) es una alternativa adecuada para abordar el problema de asignación de estudiantes a desafíos. A diferencia de RL, MCTS no requiere un entrenamiento extenso y puede adaptarse rápidamente a cambios en el ambiente. MCTS [17] construye un árbol de búsqueda de manera incremental, explorando y evaluando diferentes acciones en cada estado, lo que le permite encontrar soluciones efectivas con un número limitado de simulaciones.

Para el desarrollo del algoritmo de MCTS se necesario desarrollar las 4 etapas principales:

- **Selección:** Comenzando desde el nodo raíz, se selecciona el nodo hijo más prometedor utilizando una política de selección, en nuestro caso se utilizó el UCT (Upper Confidence Bound for Trees) [35], que balancea la explotación de nodos con alto valor y la exploración de nodos menos visitados. La fórmula es la

siguiente:

$$\text{UCT} = \frac{v_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}}$$

donde:

- v_i es el valor total acumulado del nodo i (la suma de las recompensas obtenidas en las simulaciones que pasaron por este nodo).
- n_i es el número de visitas al nodo i.
- C es una constante de exploración que controla el equilibrio entre la explotación y la exploración. En el código, se utiliza $C = \sqrt{2}$ por defecto.
- N es el número total de visitas al nodo padre.

- **Expansión:** Si el nodo seleccionado no es terminal, o sea que no tenga hayan más estudiantes a asignar o que es imposible asignar el estudiante a algún desafío, se expande creando uno o más nodos hijos que representen los posibles desafíos que puede ser asignado el estudiante en ese nodo.

- **Simulación:** Desde el nodo expandido, se realiza una serie de simulaciones hasta alcanzar un estado terminal. Las acciones durante la simulación se toman de acuerdo a una política de default, la cual esta basado en la aleatoriedad o bien la preferencia más alta disponible del estudiante dentro de la lista de desafíos viables. Esto esta en un ratio de que el 70 % del tiempo utiliza la preferencia más alta disponible, en otros casos se escoge uno aleatorio.

- **Retropropagación:** El resultado de la simulación se propaga hacia atrás a través de los nodos visitados, actualizando sus estadísticas (valor y número de visitas).

Además de las 4 etapas principales se presenta algunas características especiales que lo diferencian de una implementación estándar de MCTS:

- **Manejo de grupos incompletos:** El algoritmo realiza un seguimiento de los grupos incompletos (aquellos con menos estudiantes que el mínimo requerido) y

prioriza la asignación de estudiantes a estos grupos. Esto se logra mediante una función que verifica si una acción es viable considerando la disponibilidad de estudiantes para completar los grupos incompletos antes de permitir la asignación a grupos nuevos o completos.

- **Actualización dinámica de los desafíos disponibles:** A medida que se asignan estudiantes a los desafíos, el algoritmo actualiza dinámicamente la lista de desafíos disponibles para cada carrera. Esto se realiza en la función `apply_action`, donde se verifica si un desafío ha alcanzado su capacidad máxima o si se han alcanzado los límites de estudiantes por carrera. Los desafíos que ya no están disponibles se eliminan de la lista de desafíos disponibles para las carreras correspondientes.

- **La función de evaluación:** La función de evaluación del estado terminal considera múltiples factores relevantes para el problema, como la satisfacción de las preferencias de los estudiantes, el número de estudiantes asignados y la completitud de los grupos.

La efectividad del algoritmo implementado de MTCS depende principalmente del número de simulaciones, lo permitirá tomar cada vez mejores decisiones sobre que nodo seleccionar al evaluar la función de UCT (Upper Confidence Bound for Trees) mencionada anteriormente.

Capítulo 5

Resultados

5.1. Plan de evaluación

Para evaluar como se adaptan los agentes al desarrollo actual y como explorará el árbol de búsqueda de monte carlo, se utilizarán los datos obtenidos de la fuente oficial del Programa de Memorias Multidisciplinarias, de los cuales son las postulaciones de los estudiantes y desafíos del año 2024, que fue el caso que demostró la necesidad de la creación de un sistema de distribución de desafíos automatizado.

5.2. Evaluación del Aprendizaje por Reforzamiento

Desafíos saturados

A continuación se muestran los resultados de entrenar los dos agentes en el ambiente donde primero se ordenan los estudiantes en base a la cantidad de postulaciones a desafíos de manera ascendente o descendente.

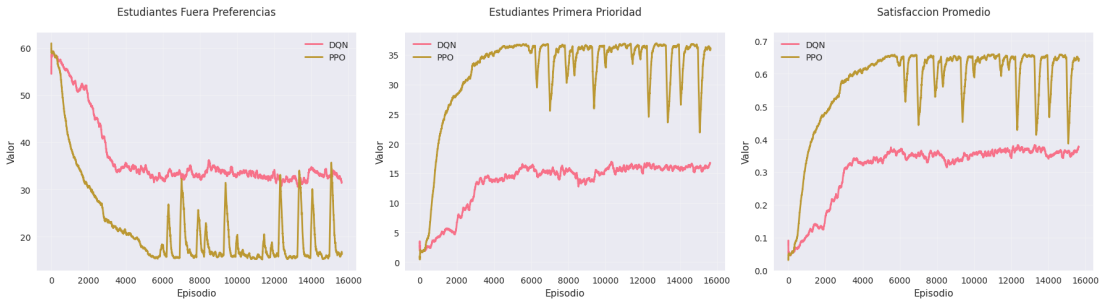


Figura 5.1: Satisfacción de estudiantes en entrenamiento PPO y DQN con desafíos saturados al último

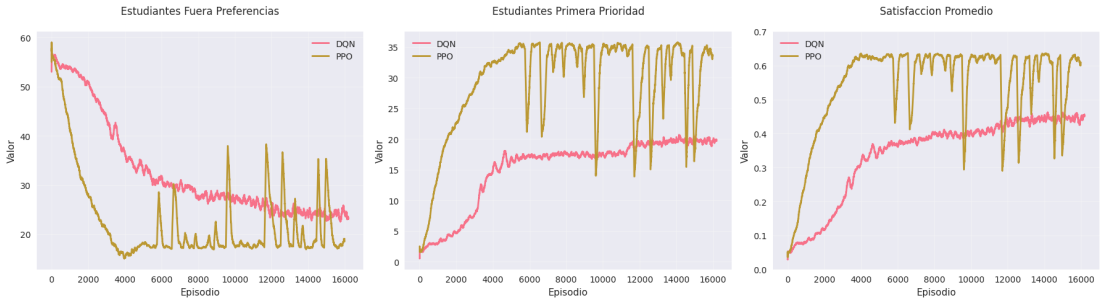


Figura 5.2: Satisfacción de estudiantes en entrenamiento PPO y DQN con desafíos saturados primero

En primera instancia podemos confirmar que PPO llega a un punto de crecimiento estacionario primero que el agente de DQN, pero los resultados muestran que si continuamos entrenando el modelo podría llegar tal vez a un resultado mejor. Podemos ver también que al ordenar por los desafíos saturados primero, el agente llega a entregar una buena asignación antes que al hacerlo de manera contraria, o sea ordenándolos de menos saturados a más saturados.

Tamaño de carreras

A continuación se muestran los resultados de entrenar los dos agentes en el ambiente donde primero se ordenan los estudiantes en según el tamaño de las carreras manera ascendente o descendente.

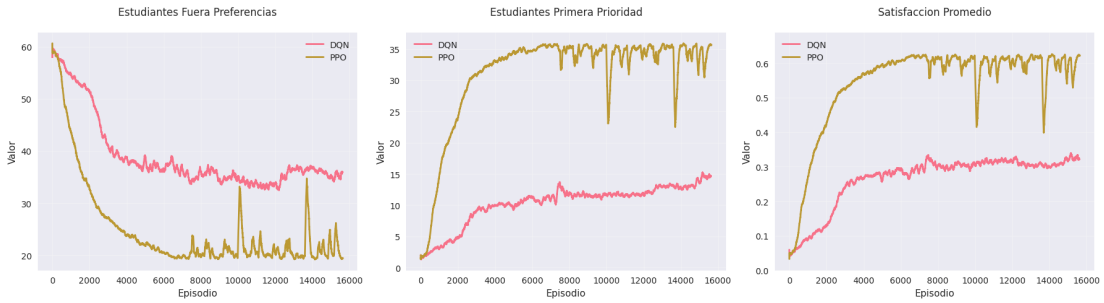


Figura 5.3: Satisfacción de estudiantes en entrenamiento PPO y DQN con carreras saturados al último

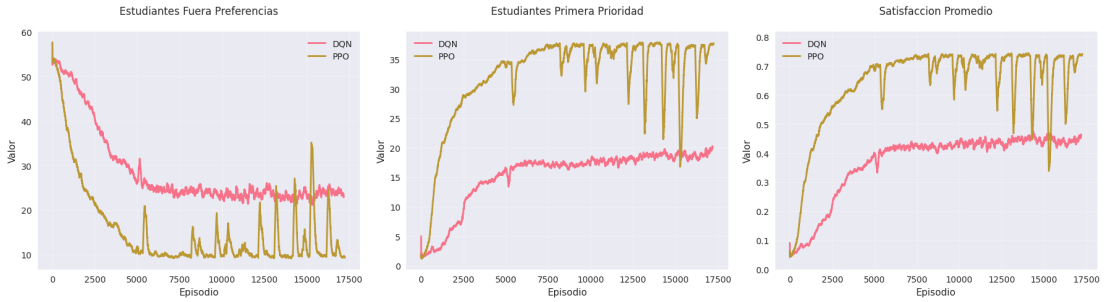


Figura 5.4: Satisfacción de estudiantes en entrenamiento PPO y DQN con carreras saturados primero

De los resultados podemos ver que al dejar las carreras saturadas al último la satisfacción promedio de los estudiantes es menor que si hubiéramos realizado al revés, atacando el problema con las carreras más saturadas primero. También se puede ver que los estudiantes en su primera prioridad son menores y que el número de estudiantes que quedaron en desafíos fuera de sus preferencias son mayores.

Búsqueda de hiperparámetros

Finalmente se buscaron los mejores hiperparámetros para el agente PPO para encontrar su mejor configuración para este contexto en especifico, para ello se utilizó **optuna**, la cual estaba encargada de encontrar los mejores valores vistos en el Cuadro 4.3. Por temas de limitaciones computacionales y tiempo de entrenamiento, se decidió darle 50 intentos con $120{,}000$ pasos cada uno.

En la Figura 5.5 podemos apreciar las 50 iteraciones representadas por un color único, se analizó cuales hiperparámetros fueron los mejores dentro de este espació reducido de intentos al evaluar la métrica de satisfacción promedio de los participantes.

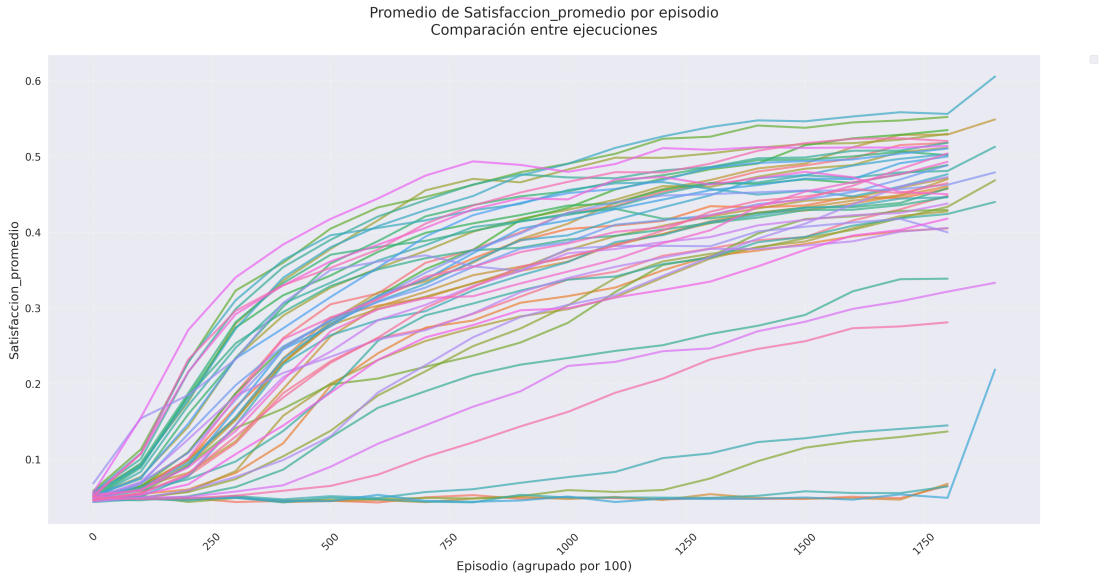


Figura 5.5: Búsqueda de hiperparámetros con Optuna

Al terminar las 50 iteraciones se termino con un modelo con las configuraciones de parámetros entregados en el Cuadro 5.1, dejando un modelo que consigue una satisfacción acumulada de $67{,}83$.

Parámetro	Nombre Técnico	Mejor Valor encontrado
Tasa de aprendizaje	learning_rate	0,00093
Pasos de entrenamiento	n_steps	1301
Tamaño del lote	batch_size	158
Épocas de entrenamiento	n_epochs	7
Factor de descuento	gamma	0,9456
Lambda GAE	gae_lambda	0,90024
Rango de recorte	clip_range	0,2172
Coeficiente de entropía	ent_coef	0,00621
Coeficiente de función de valor	vf_coef	0,4049
Norma máxima del gradiente	max_grad_norm	0,5357

Cuadro 5.1: Mejores parámetros de configuración para PPO en este experimento.

5.2.1. Lecciones Aprendidas del Desarrollo de RL

La implementación completa del ambiente de simulación y el entrenamiento de agentes PPO y DQN proporcionó una comprensión valiosa sobre las limitaciones inherentes del aprendizaje por reforzamiento para este tipo de problemas. Esta experiencia de desarrollo reveló criterios fundamentales para evaluar cuándo el RL es o no la aproximación apropiada.

Limitaciones de Configuración Dinámica

El principal obstáculo identificado radica en la naturaleza dinámica del PMM. Cada iteración anual presenta:

- Diferentes números de estudiantes y distribuciones por carrera
- Conjuntos distintos de desafíos disponibles
- Variaciones en las preferencias estudiantiles

- Modificaciones en las restricciones institucionales

Estos cambios requieren redefinir completamente los espacios de observación y acción del ambiente, invalidando cualquier política previamente aprendida. Específicamente, se identificó que problemas con configuraciones que cambian anualmente requieren soluciones que no dependan de entrenamiento extenso previo.

Costos Computacionales Prohibitivos

El entrenamiento de cada agente demandó recursos significativos:

- PPO: Múltiples horas para convergencia básica
- DQN: Tiempos similares con menor estabilidad
- Optimización de hiperparámetros: Más de 24 horas para 50 iteraciones limitadas

Considerando que estos costos se repetirían para cada nueva configuración del programa, la viabilidad operacional resulta cuestionable, especialmente en contextos donde las decisiones deben tomarse en plazos cortos tras el cierre de postulaciones.

Criterios para Aplicabilidad de RL

Esta experiencia sugiere que el RL es apropiado cuando:

- Las configuraciones del problema permanecen estables durante múltiples iteraciones
- Los recursos computacionales para reentrenamiento están garantizados
- El costo temporal del entrenamiento es compatible con los plazos operativos
- La mejora esperada justifica la inversión computacional

Valor Metodológico de la Implementación

A pesar de sus limitaciones operacionales, el desarrollo del ambiente de RL contribuyó metodológicamente al:

- Establecer un framework de evaluación sistemática
- Identificar métricas relevantes para el dominio educativo
- Proporcionar una línea base comparativa para métodos alternativos
- Generar criterios de selección algorítmica para contextos similares

Esta experiencia valida la importancia de la implementación empírica para identificar limitaciones que no son evidentes en el análisis teórico inicial, proporcionando lecciones valiosas para futuras investigaciones en automatización de procesos educativos.

5.3. Evaluación de Árbol de Búsqueda de Monte Carlo

La efectividad del algoritmo implementado de MTCS depende principalmente del número de simulaciones, lo permitirá tomar cada vez mejores decisiones sobre que nodo seleccionar al evaluar la función de UCT (Upper Confidence Bound for Trees) mencionada anteriormente. Por eso mismo se decidió evaluar como aumenta en 3 casos distintos, donde se evidenciará como se comporta a medida que se cambia el orden de los estudiantes.

Aleatorio

Los resultados del algoritmo MCTS con orden aleatorio de estudiantes muestran una tendencia de mejora constante a medida que aumenta el número de simulaciones. Con una única simulación, el algoritmo alcanza una satisfacción promedio de 0.617 ± 0.047, mientras que al incrementar a 1000 simulaciones, este valor mejora hasta 0.668

± 0.029. Es notable que la variabilidad en los resultados, representada por la desviación estándar, disminuye conforme aumenta el número de simulaciones, lo que indica una mayor consistencia en las soluciones generadas.

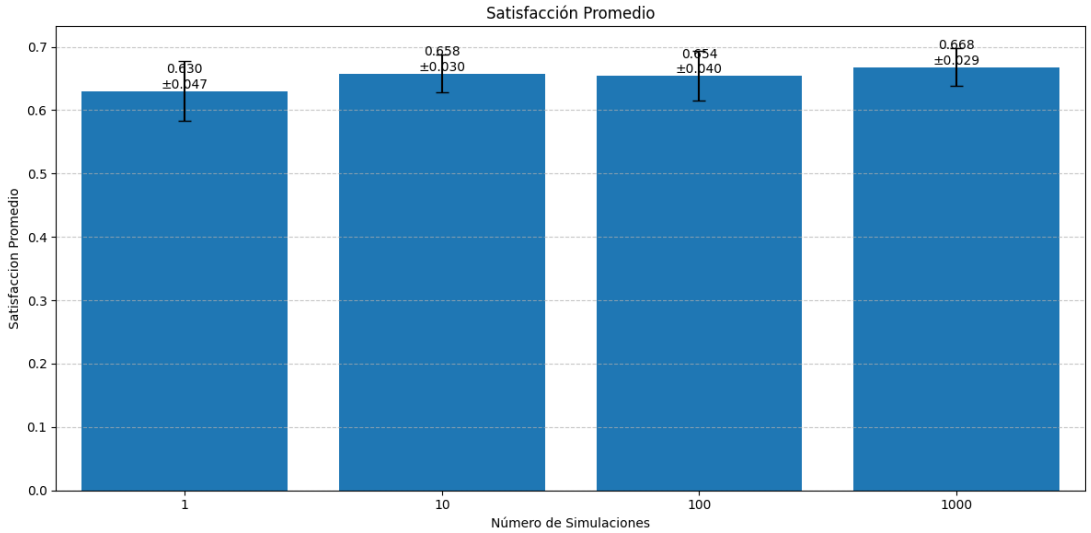


Figura 5.6: Satisfacción promedio de los estudiantes ordenados de manera aleatoria para MCTS

Desafíos saturados

Al ordenar los estudiantes según la saturación de los desafíos, se observan dos comportamientos distintos:

Saturación ascendente (menos a más saturados):

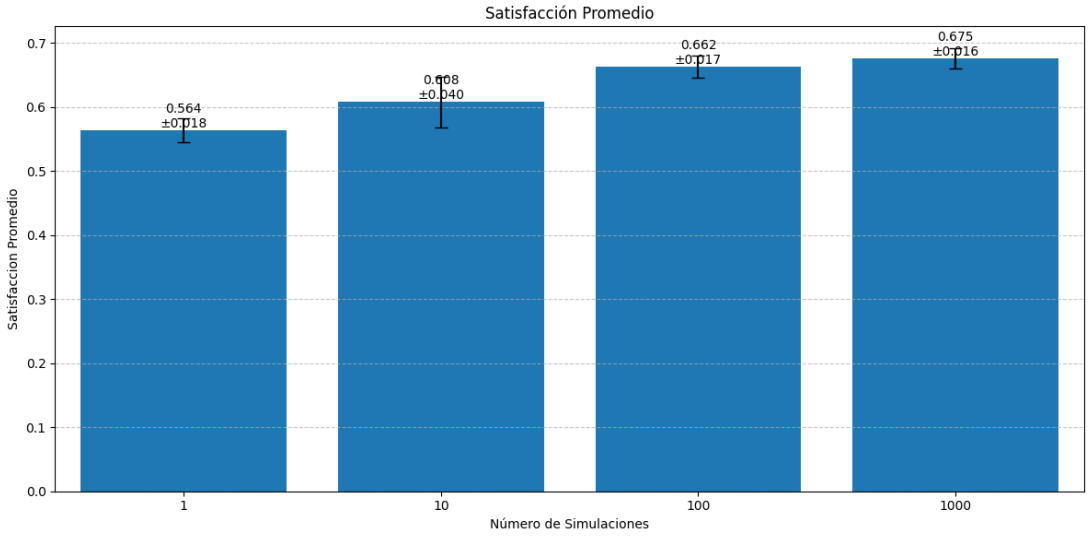


Figura 5.7: Satisfacción promedio de los estudiantes ordenados por desafíos saturados de manera ascendente para MCTS

El algoritmo muestra un rendimiento inicial más bajo, con una satisfacción promedio de 0.564 ± 0.018, pero logra una mejora significativa al aumentar el número de simulaciones, alcanzando 0.675 ± 0.016 con 1000 simulaciones. Este comportamiento sugiere que el algoritmo requiere más exploraciones para encontrar buenas soluciones cuando comienza con los desafíos menos demandados.

Saturación descendente (más a menos saturados):

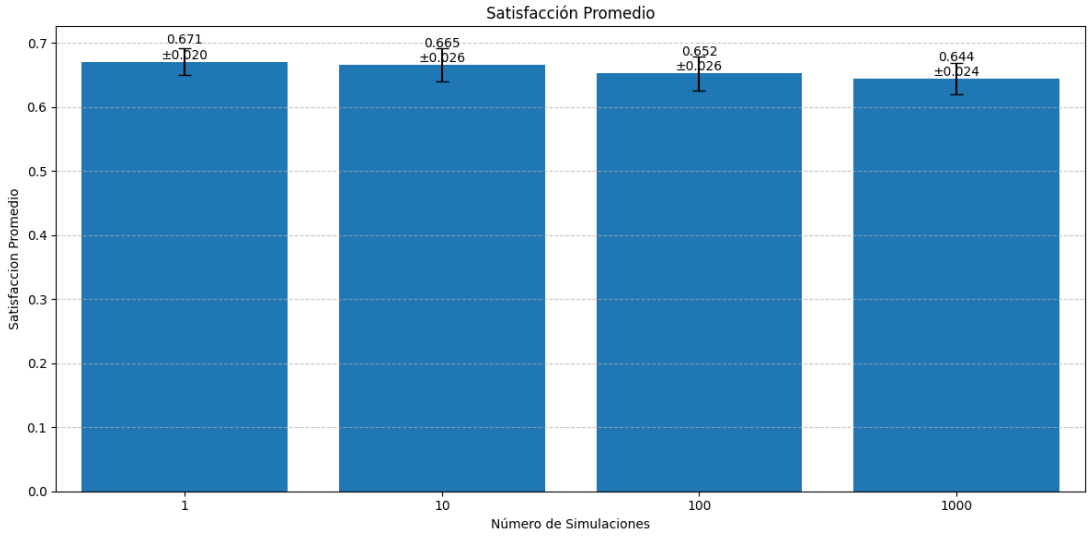


Figura 5.8: Satisfacción promedio de los estudiantes ordenados por desafíos saturados de manera descendente para MCTS

Partiendo de los desafíos más saturados, el algoritmo alcanza una satisfacción inicial más alta de 0.671 ± 0.025, manteniéndose relativamente estable hasta las 1000 simulaciones con 0.641 ± 0.024. Esta estabilidad sugiere que abordar primero los casos más competitivos permite al algoritmo encontrar soluciones satisfactorias con menos simulaciones.

Tamaño de carreras

Similar al caso anterior, se observaron diferencias significativas según el orden de procesamiento:

Tamaño ascendente (carreras pequeñas primero):

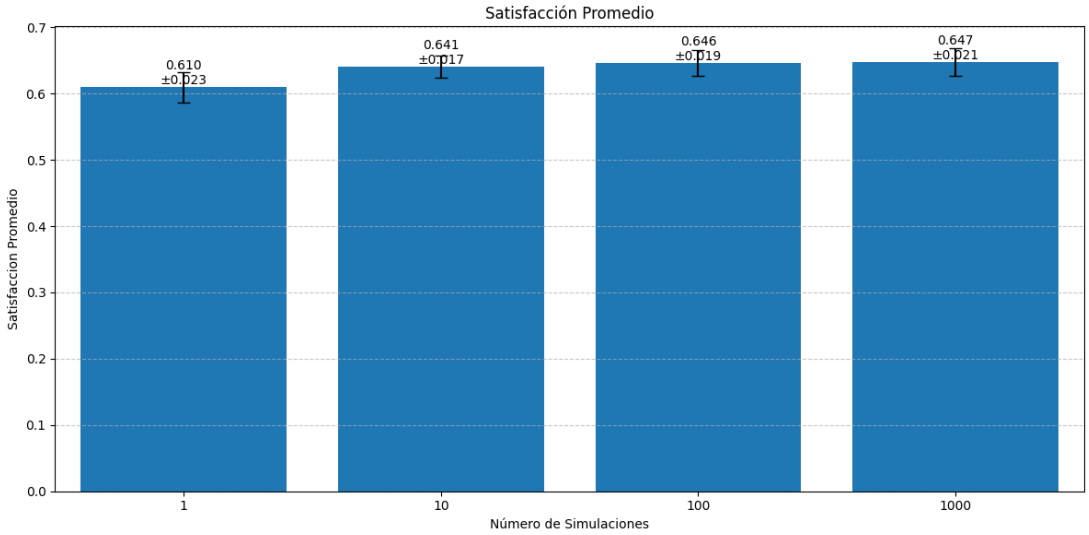


Figura 5.9: Satisfacción promedio de los estudiantes ordenados por tamaño de carreras de manera ascendente para MCTS

El algoritmo comienza con una satisfacción promedio de 0.610 ± 0.023 y mejora gradualmente hasta 0.647 ± 0.021 con 1000 simulaciones. La mejora es modesta pero consistente, sugiriendo que el algoritmo puede manejar eficientemente la asignación comenzando con las carreras más pequeñas.

Tamaño descendente (carreras grandes primero):

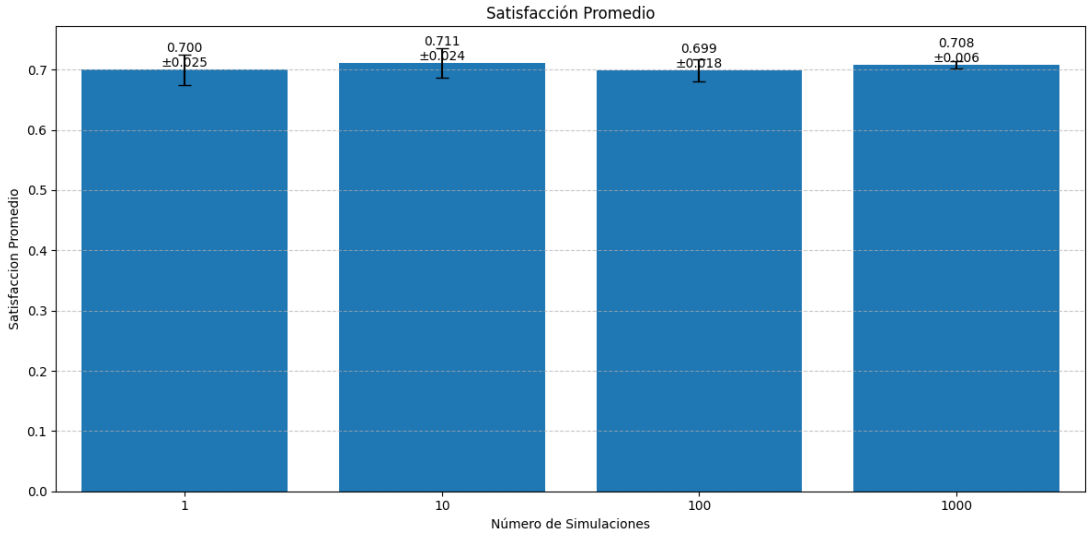


Figura 5.10: Satisfacción promedio de los estudiantes ordenados por tamaño de carreras de manera descendente para MCTS

Comenzando con las carreras más grandes, se obtiene una satisfacción inicial más alta de 0.705 ± 0.025, aunque se observa una ligera disminución a 0.708 ± 0.020 con 1000 simulaciones. Este comportamiento indica que priorizar las carreras más grandes permite al algoritmo encontrar soluciones más eficientes desde el inicio.

En todos los casos, se observa que el incremento en el número de simulaciones tiende a estabilizar los resultados, reduciendo la variabilidad en las soluciones encontradas. Esto sugiere que el algoritmo MCTS es capaz de encontrar soluciones consistentes y de buena calidad una vez que se le permite realizar suficientes exploraciones del espacio de búsqueda.

5.4. Evaluación de conectividad

Para validar la integración completa del sistema, se realizaron pruebas exhaustivas de la conectividad entre el frontend, el servidor de asignación y los algoritmos implementados. Las pruebas se enfocaron en tres aspectos principales: la correcta transmisión de datos, el manejo de estados de carga y la visualización de resultados.

5.4.1. Validación de llamadas APIs

La primera fase de pruebas se realizó utilizando la herramienta Postman, una plataforma diseñada para el desarrollo, prueba, documentación y monitorización de API (Interfaces de Programación de Aplicaciones) para verificar el funcionamiento de los Endpoints especificados en la sección 4.3.4. Se comprobó que el servidor procesa correctamente las peticiones POST con los datos de los participantes y devuelve las asignaciones en el formato JSON especificado (Ver Figura 5.11).

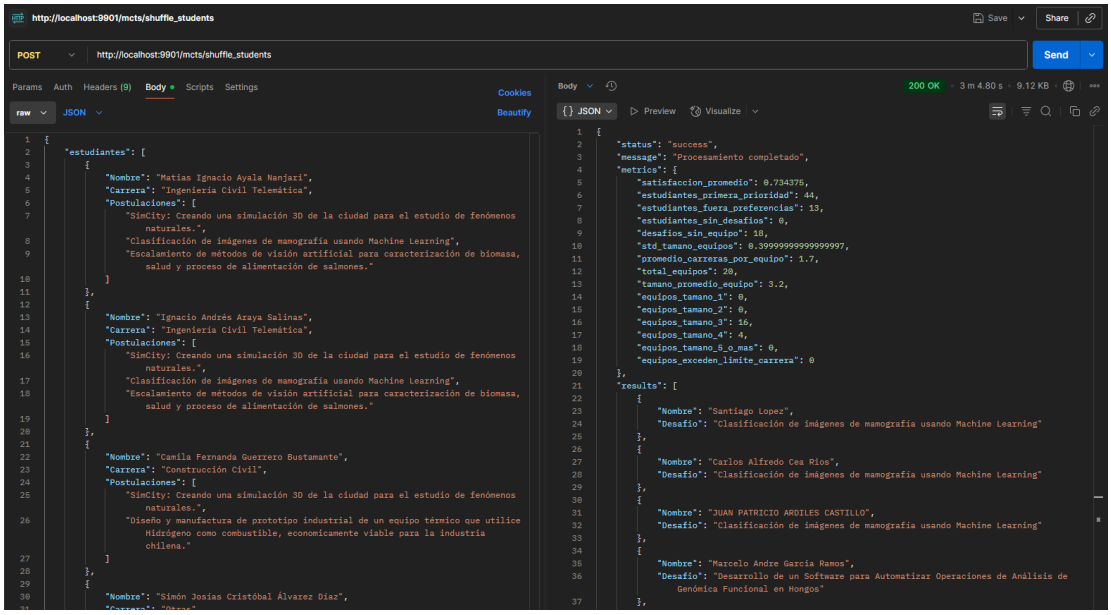


Figura 5.11: Entrada y Respuesta de Servidor

Las pruebas confirmaron que la estructura de la respuesta mantiene la consistencia definida en la sección 4.3.7, incluyendo las métricas de evaluación y los detalles de las asignaciones realizadas.

5.4.2. Integración con la Interfaz de Usuario

La integración con el Frontend se validó a través de tres estados principales de interacción:

- **Configuración de los Asignadores**: Para el correcto funcionamiento del Ser-

vidor Selector de Asignador, es necesario que dentro del requerimiento tenga detallado la cantidad máxima de estudiantes de la misma carrera por grupo. En la Figura 5.12 se muestra una interfaz interactiva con esta configuración.

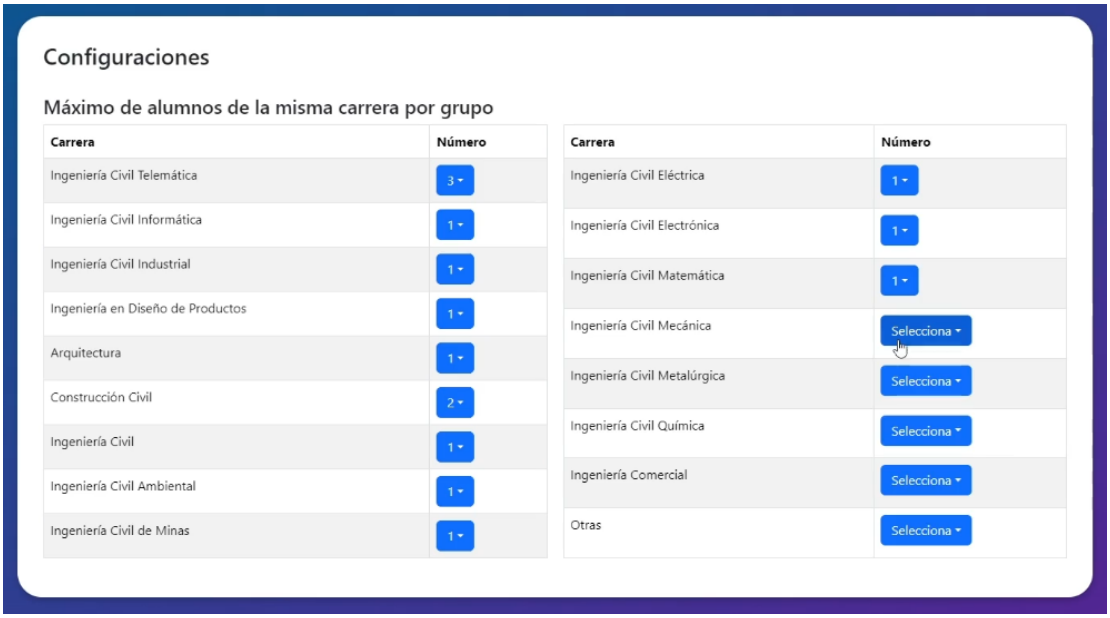
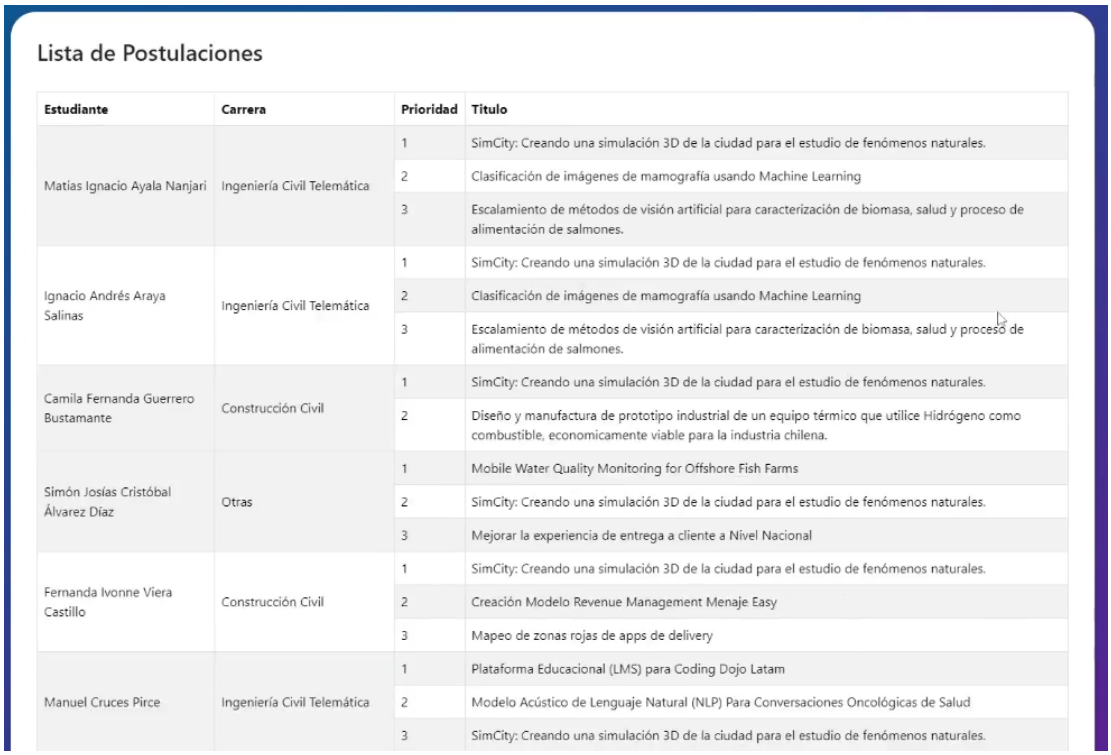


Figura 5.12: Interfaz de Configuración

- **Envió de Estudiantes**: La interfaz (Ver Figura 5.13) permite al organizador a ver un resumen de los estudiantes con sus postulaciones, al confirmar que este todo en orden, se podrá enviar la solicitud al Servidor Asignador a que forme los distintos equipos.



Lista de Postulaciones

Estudiante	Carrera	Prioridad	Titulo
Matias Ignacio Ayala Nanjari	Ingeniería Civil Telemática	1	SimCity: Creando una simulación 3D de la ciudad para el estudio de fenómenos naturales.
		2	Clasificación de imágenes de mamografía usando Machine Learning
		3	Escalamiento de métodos de visión artificial para caracterización de biomasa, salud y proceso de alimentación de salmones.
Ignacio Andrés Araya Salinas	Ingeniería Civil Telemática	1	SimCity: Creando una simulación 3D de la ciudad para el estudio de fenómenos naturales.
		2	Clasificación de imágenes de mamografía usando Machine Learning
		3	Escalamiento de métodos de visión artificial para caracterización de biomasa, salud y proceso de alimentación de salmones.
Camila Fernanda Guerrero Bustamante	Construcción Civil	1	SimCity: Creando una simulación 3D de la ciudad para el estudio de fenómenos naturales.
		2	Diseño y manufactura de prototipo industrial de un equipo térmico que utilice Hidrógeno como combustible, economicamente viable para la industria chilena.
Simón Josías Cristóbal Álvarez Díaz	Otras	1	Mobile Water Quality Monitoring for Offshore Fish Farms
		2	SimCity: Creando una simulación 3D de la ciudad para el estudio de fenómenos naturales.
		3	Mejorar la experiencia de entrega a cliente a Nivel Nacional
Fernanda Ivonne Viera Castillo	Construcción Civil	1	SimCity: Creando una simulación 3D de la ciudad para el estudio de fenómenos naturales.
		2	Creación Modelo Revenue Management Menaje Easy
		3	Mapeo de zonas rojas de apps de delivery
Manuel Cruces Pirce	Ingeniería Civil Telemática	1	Plataforma Educacional (LMS) para Coding Dojo Latam
		2	Modelo Acústico de Lenguaje Natural (NLP) Para Conversaciones Oncológicas de Salud
		3	SimCity: Creando una simulación 3D de la ciudad para el estudio de fenómenos naturales.

Figura 5.13: Interfaz de Estudiantes postulando

- **Resultado de Distribución**: El sitio logra capturar de manera correcta la respuesta definida en la sección 4.3.7, como se ve en la figura 5.14

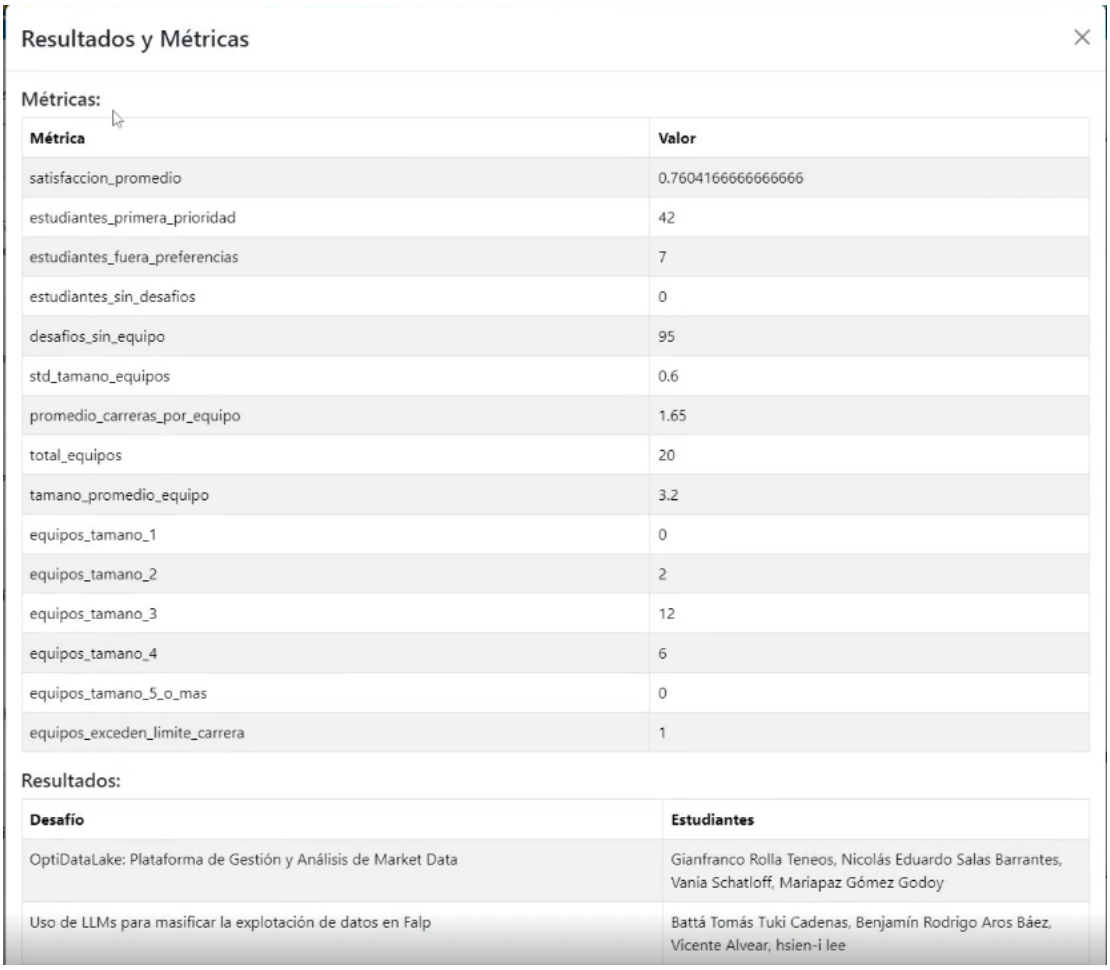


Figura 5.14: Interfaz con la distribución de estudiantes

5.5. Rendimiento del Sistema

5.5.1. Limitaciones del Aprendizaje por Reforzamiento

Durante la evaluación del prototipo del asignador que usa el Aprendizaje por Reforzamiento con los agentes PPO y DQN se encontró que el tiempo que toma en hacer un entrenamiento completo por la problemática definida en la Sección 4.4 es muy alto para obtener un resultado similar o peor que el usar el algoritmo del árbol de búsqueda de monte carlo. Cada evaluación obtenida en la sección 5.2 tomaba múltiples horas, e incluso un día al momento de hacer la búsqueda de hiperparámetros con Optuna, donde

la satisfacción acumulada llego solamente a un 67,83 %. Además esta solución se ve limitada al momento de necesitar escalarlo o montarlo en un ambiente en la nube.

5.5.2. Procesamiento de Datos de Entrada

Se evaluaron los distintos métodos de distribución de estudiantes definidos en la sección 4.3.3 con 40 entradas distintas para evaluar el comportamiento de cuatro métricas relevantes.

Método	**Satisfacción Promedio**	**Primera Prioridad**	**Fuera Preferencias**
Aleatorio	0.688 ± 0.032	38.4 ± 3.1	12.5 ± 1.1
Saturación Asc.	0.666 ± 0.001	40.0 ± 0.0	17.0 ± 0.0
Saturación Desc.	0.715 ± 0.019	36.5 ± 1.1	8.8 ± 1.7
Carrera Asc.	0.677 ± 0.013	37.6 ± 1.6	13.4 ± 1.2
Carrera Desc.	0.700 ± 0.013	38.0 ± 0.9	10.8 ± 1.0

Cuadro 5.2: Resultados de Evaluación de Métodos

Como se aprecia en el Cuadro 5.2, se analizaron las métricas de Satisfacción Promedio, número de estudiantes en su primera prioridad, número de estudiantes que no quedaron en ninguna de sus preferencias y la desviación estándar del tamaño de los equipos formados. Dejando que para este caso, si se requiere maximizar la satisfacción general y disminuir la cantidad de estudiantes que quedan fuera de sus postulaciones se recomienda ordenar a los estudiantes priorizando a los con postulaciones a desafíos de mayor demanda mediante ordenamiento descendente por saturación.

5.5.3. Tiempo de Distribución

Los tiempos de distribución presentados en la Figura 5.15 evidencian que todos los métodos implementados para el Asignador, basado en el algoritmo de árbol de búsqueda de Monte Carlo, completan el proceso de asignación en tiempos significativamente menores en comparación con el procedimiento manual.

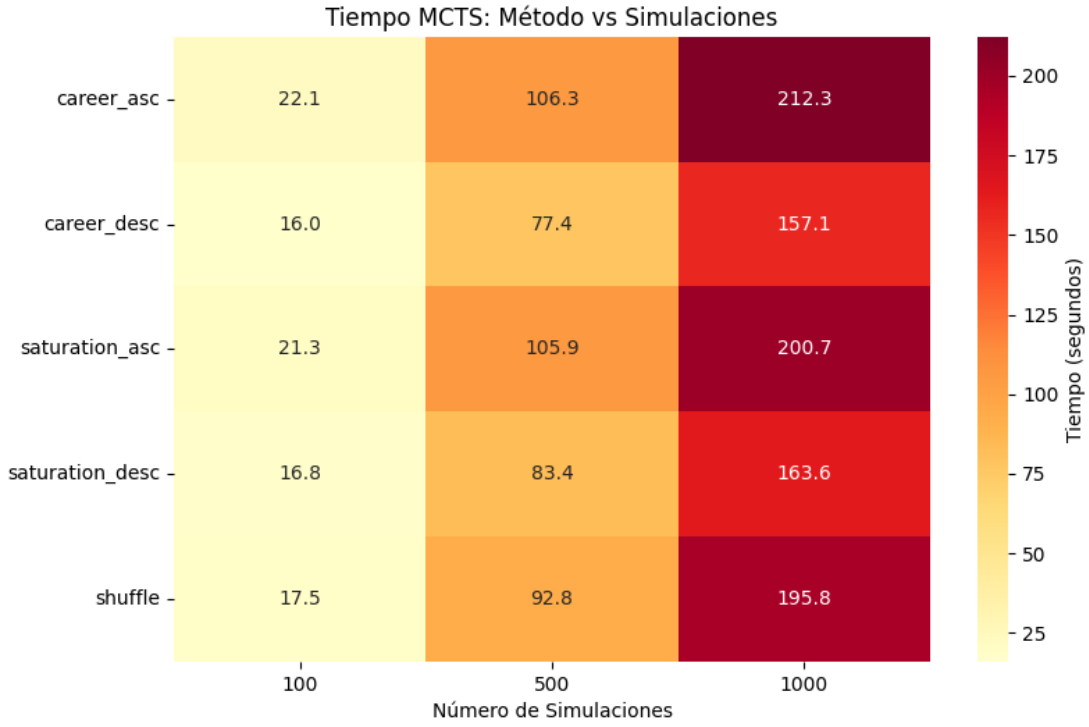


Figura 5.15: Tiempos de Respuesta según Método vs. Simulaciones

De igual modo, se analizó el comportamiento de la satisfacción promedio de los estudiantes en función del incremento del número de simulaciones y del tiempo de respuesta, con enfoque en el método recomendado según los resultados representados en el Cuadro 5.2, el cual corresponde al Método de Ordenamiento de Desafíos más demandados en primer lugar.

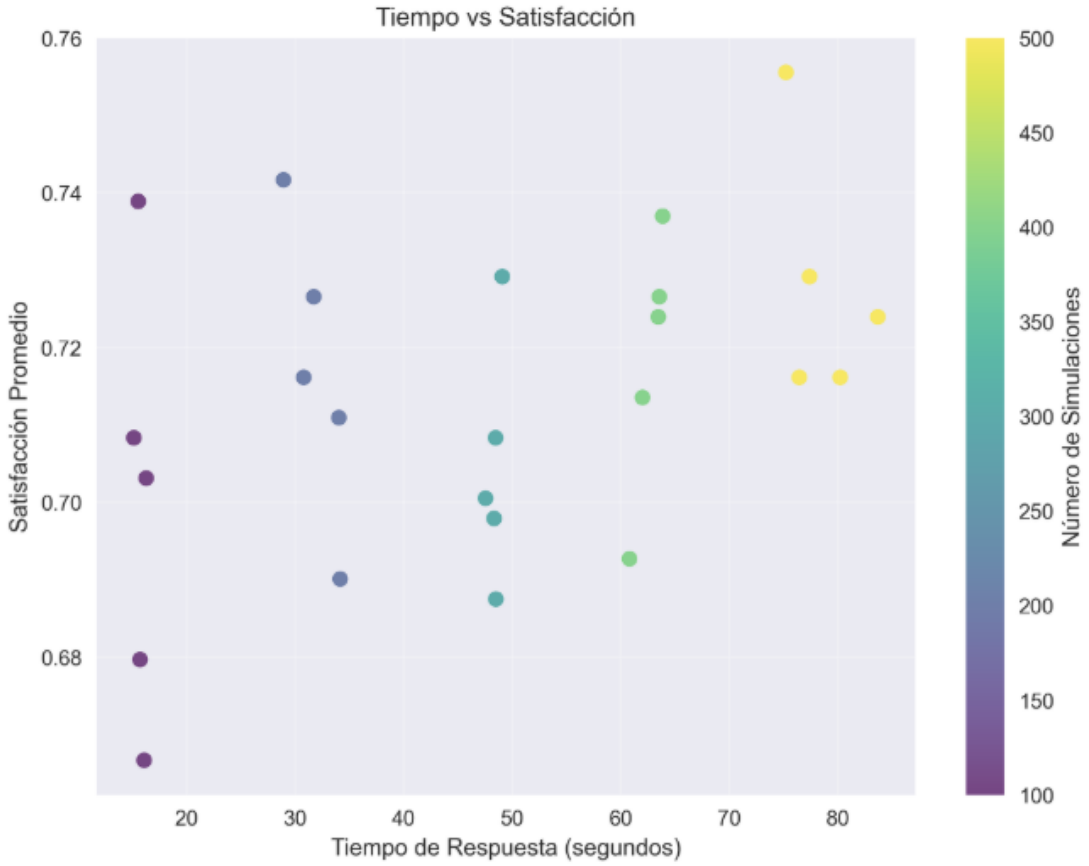


Figura 5.16: Tiempo vs. Satisfacción vs. Número de Simulaciones

Tal como se aprecia en la Figura 5.16, la calidad de las asignaciones tiende a ser más consistente y elevada a medida que se incrementa la cantidad de simulaciones. No obstante, el tiempo de respuesta presenta un crecimiento prácticamente lineal. Pese a ello, incluso considerando aumentos sustantivos en el número de simulaciones, la eficiencia operativa se mantiene significativamente superior, alcanzando una reducción temporal del 98,5 % en comparación con el proceso manual, el cual demandaba aproximadamente 4 horas.

5.6. Áreas de mejora

El análisis de los resultados obtenidos y de las limitaciones identificadas durante la implementación del sistema revela oportunidades significativas de mejora, particular-

mente en aspectos relacionados con la escalabilidad y la eficiencia operativa del servidor. Estas mejoras resultan fundamentales para garantizar la viabilidad del sistema en escenarios de crecimiento del programa y su potencial adopción en otras instituciones.

5.6.1. Escalabilidad

La implementación actual utiliza una arquitectura rígida donde todos los asignadores se ejecutan en un único servidor. Esta aproximación presenta limitaciones significativas para el escalamiento horizontal. El sistema podría beneficiarse de una arquitectura distribuida basada en múltiples instancias, complementada con un balanceador de carga que habilite un escalamiento dinámico según la demanda.

La incorporación de un sistema de encolamiento y procesamiento asíncrono permitiría gestionar múltiples solicitudes de formación de equipos de manera concurrente, optimizando la distribución de cargas computacionales. Este enfoque habilitaría el manejo simultáneo de solicitudes de diferentes instituciones sin comprometer el rendimiento del sistema.

El servidor podría implementar un sistema de identificadores únicos asociados a las credenciales del cliente, permitiendo consultas sobre el estado de las asignaciones en progreso. Adicionalmente, la incorporación de un sistema de selección dinámica de algoritmos, basado en características del problema como número de participantes y restricciones temporales, optimizaría automáticamente el balance entre calidad de solución y tiempo de procesamiento.

5.6.2. Persistencia de Datos

La implementación actual carece de un sistema robusto de persistencia de datos. La incorporación de un sistema de trabajos (jobs) permitiría almacenar las distribuciones generadas para cada cliente, habilitando su recuperación posterior mediante consultas a la API. Para este propósito, una base de datos NoSQL resultaría apropiada, facilitando la gestión de grandes volúmenes de datos y permitiendo la implementación de

esquemas de replicación que garanticen alta disponibilidad.

Este enfoque habilitaría el almacenamiento de configuraciones personalizadas por cliente, historial de asignaciones previas, y métricas de rendimiento para análisis posterior. La persistencia de datos también facilitaría la implementación de funcionalidades de auditoría y trazabilidad del proceso de asignación.

5.6.3. Monitoreo

La adopción de una infraestructura basada en la nube facilitaría la implementación de un sistema integral de monitoreo. Este incluiría métricas de rendimiento del sistema (utilización de CPU, memoria y tiempo de respuesta), registro de eventos críticos, configuración de alertas automatizadas para condiciones anómalas, y dashboards de administración que posibiliten el monitoreo en tiempo real.

La implementación de logging estructurado y métricas de negocio (como satisfacción promedio de asignaciones y tiempo de procesamiento por algoritmo) proporcionaría información valiosa para la optimización continua del sistema y la toma de decisiones operativas.

Capítulo 6

Conclusiones

Esta investigación abordó el desafío crítico de la formación automatizada de equipos multidisciplinarios en el Programa de Memorias Multidisciplinarias (PMM) de la UTFSM, evaluando comparativamente diferentes aproximaciones algorítmicas para determinar la más efectiva en este contexto específico.

A través del desarrollo e implementación de un servidor backend que integra algoritmos de aprendizaje por reforzamiento y búsqueda de árbol de Monte Carlo, se logró crear una plataforma de evaluación que permitió identificar las fortalezas, limitaciones y casos de uso óptimos para cada técnica. Los resultados experimentales revelaron hallazgos significativos que divergen parcialmente de las expectativas iniciales.

6.1. Hallazgos Principales

6.1.1. Limitaciones del Aprendizaje por Reforzamiento

Aunque los algoritmos PPO y DQN demostraron capacidad de optimización (PPO alcanzó 67.83 % de satisfacción tras optimización de hiperparámetros), la investigación reveló limitaciones críticas no anticipadas en su aplicabilidad práctica. El requerimiento de reentrenamiento completo para cada nueva configuración del programa (cambios en estudiantes, desafíos o restricciones) resulta operacionalmente inviable, requiriendo

múltiples horas o días de procesamiento por iteración. Esta limitación fundamental compromete la escalabilidad prometida inicialmente y restringe su uso a escenarios donde las reglas del programa permanezcan estables por múltiples iteraciones.

6.1.2. Efectividad de Monte Carlo Tree Search

MCTS emergió como la solución más viable para las características específicas del PMM, alcanzando 71.5 % de satisfacción promedio con tiempos de ejecución de minutos. Su capacidad de adaptación inmediata a nuevas configuraciones sin reentrenamiento lo convierte en la aproximación óptima para programas con reglas dinámicas anuales. La investigación confirmó que su efectividad aumenta con el número de simulaciones, identificando un punto óptimo de balance entre calidad de solución y tiempo computacional.

6.1.3. Factores de Diseño Críticos

Un descubrimiento metodológico importante fue que el orden de procesamiento de estudiantes afecta significativamente los resultados. Procesar primero los casos más complejos (desafíos saturados o carreras mayoritarias) produce consistentemente mejores resultados globales, reduciendo estudiantes asignados fuera de sus preferencias del 17.0 % al 8.8 %.

6.2. Contribuciones y Limitaciones

La implementación exitosa de una API REST modular demuestra la viabilidad técnica de la solución y facilita la integración con sistemas externos. Sin embargo, este estudio presenta limitaciones importantes:

- Validación limitada a datos de una sola iteración del programa (2024)
- Ausencia de validación estadística rigurosa entre algoritmos

- Comparación de paradigmas computacionales fundamentalmente diferentes
- Métricas de multidisciplinariedad requieren validación con expertos del dominio

6.3. Recomendaciones

Para implementaciones en producción del PMM, se recomienda:

1. Uso de MCTS como algoritmo principal, con 500-1000 simulaciones para balance óptimo tiempo-calidad.
2. Ordenamiento por saturación descendente de estudiantes para maximizar satisfacción global.
3. Evaluación de RL únicamente en contextos donde las reglas permanezcan estables.

Para proyectos futuros de automatización educativa, se recomienda evaluar primero:

1. La frecuencia de cambios en las reglas del problema.
2. Los recursos computacionales disponibles para entrenamiento.
3. Los requisitos de tiempo de respuesta del sistema.

Estos criterios, derivados de la experiencia de implementación, pueden prevenir la inversión en aproximaciones inadecuadas.

6.4. Direcciones Futuras

Este trabajo establece una metodología sólida para la evaluación de algoritmos de asignación en contextos educativos. Investigaciones futuras deberían:

- Validar los hallazgos con múltiples datasets y configuraciones del programa.

- Desarrollar métricas más sofisticadas de multidisciplinariedad.
- Explorar algoritmos híbridos que combinen las fortalezas de ambas aproximaciones.
- Evaluar la aplicabilidad en otros contextos de formación de equipos universitarios.

La honestidad sobre las limitaciones identificadas no disminuye el valor de esta investigación, sino que proporciona una base sólida para decisiones informadas y desarrollos futuros en el campo de la automatización de procesos educativos complejos.

Bibliografía

[1] S. Zahidi, V. Ratcheva, G. Hingel, and S. Brown. The future of jobs report 2020. https://www.weforum.org/publications/the-future-of-jobs-report-2020/, 2020.

[2] McKinsey & Company. Making collaboration across functions a reality. https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/making-collaboration-across-functions-a-reality, 2024.

[3] E. Lind, F. Silva, S. Child, and F. McAliister. Making collaboration across functions a reality. https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/making-collaboration-across-functions-a-reality, 2024.

[4] Venkata Rahul Sarabu. How collaboration between data engineers and data scientists unlocks actionable insights. https://www.dataversity.net/how-collaboration-between-data-engineers-and-data-scientists-unlo
2024.

[5] Gerald C., Anh Nguyen P., Nanda R., and J. Copulsky. Teaming your way through disruption. https://www.deloitte.com/us/en/insights/topics/business-strategy-growth/cross-functional-collaboration.html, 2021.

[6] X. Wu, Y. Yang, X. Zhou, et al. A meta-analysis of interdisciplinary teaching abilities among elementary and secondary school stem teachers. 2024.

[7] Desafios Tecnológicos USM. ¿qué es el programa memorias multidisciplinarias? https://www.desafiostecnologicos.usm.cl/memorias-multidisciplinarias, 2025.

[8] Vera G. Proyecto ingeniería 2030 potencia el ecosistema de innovación de la usm. https://usm.cl/noticias/proyecto-ingenieria-2030-potencia-el-ecosistema-de-innovacion-de-la-u 2021.

[9] Desafios Tecnológicos USM. Preguntas frecuentes. https://www.desafiostecnologicos.usm.cl/preguntas-frecuentes, 2025.

[10] M. W. Ohland, M. L. Loughry, D. J. Woehr, L. G. Bullard, R. M. Felder, C. J. Finelli, R. A. Layton, H. R. Pomeranz, and D. G. Schmucker. The comprehensive assessment of team member effectiveness: Development of a behaviorally anchored rating scale for self and peer evaluation. 2012.

[11] R. Layton, M. Ohland, H. Pomeranz, and D. Woehr. Software for student team formation and peer evaluation: Catme incorporates team-maker. 2007.

[12] D. Gómez-Zará, A. Das, B. Pawlow, and N. Contractor. In search of diverse and connected teams: A computational approach to assemble diverse teams based on members' social networks. 2022.

[13] Kevin Alexander Lagos Lavín. Interrogation scheduling problem: Una estrategia basada en preferencias de profesores y alumnos. https://repositorio.usm.cl/handle/11673/52587, 2021.

[14] Víctor F. Suarez, Álvaro Guerrero, and Omar D. Castrillon. Programación de horarios escolares basados en ritmos cognitivos usando un algoritmo genético de clasificación no-dominada, nsga-ii. 2022.

[15] Andrés Saldaña Crovo, Cristian Oliva San Martín, and Lorena Pradenas Rojas. Modelos de programación entera para un problema de programación de horarios para universidades. 2007.

[16] Sánchez P. Estudio del problema de school timetabling utilizando técnicas de ia, 2009.

[17] Alberto Daudén Figueras. Desarrollo de un algoritmo para la distribución de exámenes finales en el itba, 2019.

[18] Q. Sun, C. Wang, Z. Wang, et al. A fault diagnosis method of smart grid based on rough sets combined with genetic algorithm and tabu search. 2013.

[19] David Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. 2018.

[20] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction (2nd ed.), 2015.

[21] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Introduction to rl, 2018.

[22] Richard Bellman. On the theory of dynamic programming. 1952.

[23] The Farama Foundation. Gymnasium documentation (s.f), 2025.

[24] A; Barham P; Brevdo E Abadi, M; Agarwal. Tensorflow: Large-scale machine learning on heterogeneous systems., 2015.

[25] Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Zhengxing Chen, Yuchen He, Zachary Kaden, Vivek Narayanan, and Xiaohui Ye. Horizon: Facebook's open source applied reinforcement learning platform. *arXiv preprint arXiv:1811.00260*, 2018.

[26] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. Openspiel: A framework for reinforcement learning in games, 2020.

[27] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[28] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`, 2018.

[29] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[30] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[31] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[32] James Bergstra, Daniel Yamins, and David D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pages 115–123, June 2013.

[33] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. scikit-optimize (version v0.8.1). `https://zenodo.org/record/4014775`, 2020.

[34] Frederick M. Howard, Catherine A. Gao, and C. Sankey. Implementation of an automated scheduling tool improves schedule quality and resident satisfaction. *PLOS One*, 15(8):e0236952, 2020.

[35] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pages 282–293. Springer, 2006.