

UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
Departamento de Ingeniería Informática



**Uso de Simulated Annealing para optimizar tiempo en órdenes de
construcción en StarCraft II: experiencia Terran**

Edson Gustavo Reyes Piña

Profesor guía: Manuel Villalobos Cid

Tesis para optar al título de Ingeniero de
Ejecución en Computación e Informática

Santiago – Chile

2021

RESUMEN

Como en cualquier tipo de competición, el objetivo principal es ganar y el videojuego de estrategia en tiempo real, StarCraft II, no es la excepción. Uno de los aspectos más importantes, en el ámbito competitivo de este juego, son los órdenes de construcción, que consisten en patrones de juego para guiar al jugador hasta una unidad o estructura deseada, cuya importancia radica en qué tan buenos son en contraste con los que utilicen los oponentes, dejando al jugador que posea el mejor orden de construcción con una ventaja en su estrategia respecto al contrincante.

Desafortunadamente, en la actualidad no existen muchas opciones que ayuden a los jugadores a conseguir estas secuencias aparte de la experiencia propia o de otros jugadores más experimentados. Para cambiar esta situación, en esta investigación se propone un modelo basado en la metaheurística *Simulated Annealing* que permite obtener órdenes de construcción que buscan optimizar el tiempo de partida que estas secuencias requieren para alcanzar una unidad, un edificio o una mejora de la raza Terran.

Los resultados de este trabajo indican que los órdenes de construcción obtenidos presentan mejoras respecto a órdenes de construcción utilizados por jugadores profesionales en torneos de este videojuego en términos de tiempo.

Palabras Claves: StarCraft II, metaheurísticas, optimización, RCPSP, simulated annealing

*Even if your births and backgrounds are unknown,
and you don't have a master to serve or a sword to fight,
you can each carry your own bushido in your hearts
and become your own samurai.*

AGRADECIMIENTOS

Agradezco inmensamente a mi familia, principalmente a mi madre que siempre está preocupándose por mí y apoyándome en todo lo que puede. También a mi hermana que me escuchaba pacientemente cada vez que algo salía mal, aún si no entendía del todo lo que intentaba explicarle. A mis primos que me ayudaban a despejar la mente jugando diversos videojuegos cuando les era posible. Tampoco puedo olvidar a mi perrito que me hizo compañía todas las noches que me tocaba quedarme hasta la madrugada.

Además, le agradezco a mi profesor guía, Manuel Villalobos Cid, por su constante apoyo durante el avance de este proyecto, pues me enseñó con mucha paciencia muchos conceptos interesantes que desconocía antes de comenzar, además de siempre hacerse un tiempo cuando tenía dudas, sin importar si era horario de almuerzo o un fin de semana en la noche. También a mis compañeros bajo el ala del profesor, principalmente a Brandon y a Andrés que vivieron el proceso de la tesis junto a mí.

Finalmente, quiero agradecer a mis amigos que conocí en la universidad, quienes, durante todos estos años, rieron conmigo en los momentos buenos y sufrieron conmigo en los difíciles. Sin ellos, definitivamente no podría haber llegado hasta acá, así que les agradezco con toda el alma.

TABLA DE CONTENIDO

| | | |
|----------|--------------------------------------------|-----------|
| 1 | Introducción | 1 |
| 1.1 | Antecedentes y motivación | 1 |
| 1.2 | Estado del arte | 2 |
| 1.3 | Descripción del problema | 4 |
| 1.4 | Objetivos del proyecto | 4 |
| 1.4.1 | Objetivo general | 4 |
| 1.4.2 | Objetivos específicos | 5 |
| 1.5 | Metodología | 5 |
| 1.5.1 | Hipótesis | 5 |
| 1.5.2 | Etapa 1: Representación | 6 |
| 1.5.3 | Etapa 2: Implementación | 6 |
| 1.5.4 | Etapa 3: Evaluación | 6 |
| 1.5.5 | Etapa de documentación | 6 |
| 1.6 | Ambiente de desarrollo y herramientas | 7 |
| 1.6.1 | Ambiente de ejecución | 7 |
| 1.6.2 | Herramientas | 7 |
| 1.7 | Alcance y limitaciones | 7 |
| 1.8 | Estructura del Documento | 8 |
| 2 | Marco Teórico | 10 |
| 2.1 | StarCraft II | 10 |
| 2.1.1 | Características generales | 10 |
| 2.1.2 | Árbol de tecnologías | 11 |
| 2.1.3 | Economía | 17 |
| 2.1.4 | Velocidad de juego | 19 |
| 2.1.5 | Órdenes de construcción | 20 |
| 2.2 | StarCraft II como problema de optimización | 21 |
| 2.3 | Metaheurísticas | 22 |
| 2.3.1 | Algoritmos de población | 23 |
| 2.3.2 | Algoritmos de solución única | 23 |
| 2.3.3 | Elección de la metaheurística | 23 |
| 2.3.4 | Simulated Annealing | 24 |
| 2.4 | I-Race | 25 |
| 3 | Algoritmo propuesto y experimentos | 26 |
| 3.1 | Algoritmo propuesto | 26 |
| 3.1.1 | Representación de la solución | 28 |
| 3.1.2 | Árbol de tecnologías | 33 |
| 3.1.3 | Criterios de optimización | 34 |
| 3.1.4 | Factibilidad de las soluciones | 34 |
| 3.1.5 | Operadores | 35 |
| 3.2 | Diseño experimental | 39 |
| 3.2.1 | Parametrización | 39 |
| 3.2.2 | Evaluación | 39 |
| 4 | Resultados | 42 |
| 4.1 | Parametrización | 42 |
| 4.2 | Tiempos de ejecución | 42 |
| 4.3 | Evaluación | 43 |
| 4.3.1 | Convergencia del algoritmo | 43 |

| | | |
|----------|----------------------------------------------------|-----------|
| 4.3.2 | Comparación con algoritmo goloso | 48 |
| 4.3.3 | Órdenes de construcción de profesionales | 51 |
| 4.3.4 | Aplicación dentro del juego | 52 |
| 5 | Conclusiones | 55 |
| 5.1 | Trabajo a futuro | 56 |
| | Glosario | 58 |
| | Referencias bibliográficas | 59 |

ÍNDICE DE TABLAS

| | | |
|------------|---------------------------------------------------------------|----|
| Tabla 2.1 | Entidades Terran en StarCraft II | 13 |
| Tabla 2.2 | Entidades Terran en StarCraft II (cont.) | 14 |
| Tabla 2.3 | Entidades Terran en StarCraft II (cont.) | 15 |
| Tabla 2.4 | Tabla de velocidades de juego | 19 |
| Tabla 3.1 | Ejemplo de acciones | 29 |
| Tabla 3.2 | Ejemplo de edificios | 30 |
| Tabla 3.3 | Ejemplo de trabajadores | 31 |
| Tabla 3.4 | Fragmento de ejemplo de entidades | 32 |
| Tabla 3.5 | Ejemplo de recursos | 33 |
| Tabla 3.6 | Resumen de órdenes de construcción a comparar | 41 |
| Tabla 4.1 | Resultados de i-race | 42 |
| Tabla 4.2 | Resultados de Simulated Annealing | 44 |
| Tabla 4.3 | Resultados de SA: mejor solución para 1 Marine | 46 |
| Tabla 4.4 | Resultados de SA: mejor ejecución para 1 Starport | 46 |
| Tabla 4.5 | Resultados de SA: tercera ejecución para 2 Hellions | 47 |
| Tabla 4.6 | Convergencia de la evaluación de las soluciones | 48 |
| Tabla 4.7 | Resultados del algoritmo goloso | 48 |
| Tabla 4.8 | Resultados de Wilcoxon entre SA y algoritmo goloso | 49 |
| Tabla 4.9 | Resumen de los algoritmos | 51 |
| Tabla 4.10 | Comparación para 1 Marine | 51 |
| Tabla 4.11 | Comparación para 1 Starport | 52 |
| Tabla 4.12 | Comparación para 2 Hellion | 52 |

ÍNDICE DE FIGURAS

| | | |
|------------|-------------------------------------------------------------------------------------------------|----|
| Figura 2.1 | Captura de pantalla de una partida de StarCraft II | 11 |
| Figura 2.2 | Árbol de tecnologías Terran | 16 |
| Figura 2.3 | Parte de un orden de construcción | 21 |
| Figura 2.4 | Carrera para la configuración automática de un algoritmo | 25 |
| Figura 4.1 | Gráfico de generaciones de SA del Experimento 1: novena ejecución para 1 Marine | 44 |
| Figura 4.2 | Gráfico de generaciones de SA del Experimento 1: onceava ejecución para 1 Starport | 45 |
| Figura 4.3 | Gráfico de generaciones de SA del Experimento 1: cuarta ejecución para 2 Hellions | 45 |
| Figura 4.4 | Diagrama de caja Marine | 49 |
| Figura 4.5 | Diagrama de caja Starport | 50 |
| Figura 4.6 | Diagrama de caja Hellion | 50 |
| Figura 4.7 | Orden de construcción para un Marine dentro del juego | 53 |
| Figura 4.8 | Parte de un orden de construcción para dos Hellion dentro del juego | 54 |

ÍNDICE DE ALGORITMOS

| | | |
|---------------|-------------------------------------------------------------------------|----|
| Algoritmo 2.1 | Simulated Annealing | 24 |
| Algoritmo 3.1 | Simulated Annealing aplicado al problema de los órdenes de construcción | 27 |
| Algoritmo 3.2 | Orden de construcción aleatorio | 35 |
| Algoritmo 3.3 | Avanzar segundo | 36 |
| Algoritmo 3.4 | Cortar solución | 38 |

CAPÍTULO 1. INTRODUCCIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

StarCraft II es un videojuego de estrategia en tiempo real (RTS por sus siglas en inglés) desarrollado por Blizzard Entertainment donde cada jugador escoge una de las tres razas (Terran, Protoss y Zerg) por partida, de las cuales, cada una tiene características particulares en términos de funcionamiento, unidades y tecnologías en comparación con las otras dos, lo que resulta en una jugabilidad o experiencia de juego distinta para cada raza. En el caso de los Terran, una especie basada en humanos con tecnologías futuristas, su modo de juego gira en torno a la movilidad de sus unidades para intervenir con las bases enemigas en distintos puntos a la vez, siendo el posicionamiento uno de los puntos clave a la hora de usarles.

Aunque puede parecer un tema banal, los videojuegos de estrategia en tiempo real tienen aplicaciones fuera del ámbito de la entretención. En la medicina se sugiere que los videojuegos RTS ayudan a mejorar las habilidades cognitivas de sus jugadores (Glass, Maddox, & Love, 2013) al necesitar la atención simultánea en múltiples aspectos dentro del juego. Existe también, en el área educacional el caso de StarCraft II siendo utilizado como herramienta seria para enseñar, no sólo teoría de videojuegos, sino que también, matemáticas avanzadas y conceptos economía (Kuo, 2012).

Si bien es sabido que en cualquier juego competitivo el objetivo es intentar ganar al oponente, no siempre resulta sencillo lograrlo. Es lógico buscar otros acercamientos que entreguen nuevas metodologías o estrategias que sean mejores y más eficientes que las ya existentes de manera que se pueda estar, aunque sea solo un paso, por delante del rival con la finalidad de obtener la victoria por sobre este ya sea en un videojuego o, incluso, en un deporte. Dicho esto, en el ámbito de las competencias profesionales de videojuegos (también conocidos como *esports*), StarCraft II posee actualmente dos organizadores oficiales (Favis, 2020), ESL y DreamHack, cuyas competiciones poseen premios de hasta 500.000 dólares.

Al tomar en cuenta los métodos que utilizan los jugadores de StarCraft II para mejorar, se encuentran dos focos principales: aprender de las repeticiones que ofrece el juego y el *theorycrafting* (Kow & Young, 2013). Este último involucra la búsqueda de información acerca del juego a través de medios que son, en gran parte, no oficiales y que van desde videos explicativos hasta foros en línea y *wikis*, conteniendo, en su mayoría, estrategias y debates por parte de los mismos jugadores, siendo de utilidad, sobre todo, para aquellos con pocos conocimientos sobre optimización. Aún si puede parecer sencillo mejorar en el juego siguiendo estos medios, nada asegura que la información obtenida sea de ayuda, pues quienes redactan o participan en la

creación de esta no siempre son jugadores profesionales o competentes en el juego.

Se denominan **órdenes de construcción** a las secuencias de acciones para formar un ejército que los jugadores realizan dentro de una partida, los cuales afectan enormemente el resultado de una partida en conjunto a otros factores del juego como lo son el manejo de los recursos, el posicionamiento del ejército o cualquier decisión que debe realizar el jugador, ya sea inmediata o a futuro. Ante estos elementos presentes en cada partida, se dice que StarCraft II se clasifica como un problema de complejidad *NP-Hard* al igual que algunos de los juegos más populares de Nintendo como Super Mario o Pokémon (Aloupis, Demaine, Guo, & Viglietta, 2012).

Ante esto se tiene que la búsqueda de órdenes de construcción no puede ser abordada para encontrar un óptimo en un tiempo razonable, por lo que de ahora en adelante se referirán como “buenos órdenes de construcción” aquellos órdenes de construcción que sean utilizados por jugadores profesionales o competitivos de alto rango (dígase de aquellos con rango Maestro o Gran Maestro dentro del *ranking* de clasificación del juego), además de aquellas que logren mejorar uno o más de los elementos que estos involucran, sean tiempo de la partida, cantidad de unidades, cantidad de estructuras o cantidad de recursos, cuya importancia varía dependiendo de la estrategia de los jugadores.

1.2 ESTADO DEL ARTE

En la actualidad, existen diversos trabajos relacionados con este género de videojuegos bajo distintos enfoques y con aplicaciones diferentes. Gran parte de estos son enfocados en el desarrollo de inteligencias artificiales que buscan simular las acciones que debe realizar un jugador de StarCraft II (básicamente, algoritmos diseñados para jugar el juego) para utilizarlas en competencias entre estas mismas, junto a las estrategias que los algoritmos de estas inteligencias artificiales requieren para lograr la victoria frente a sus oponentes.

Aunque en el país no existen investigaciones al respecto, internacionalmente se pueden encontrar muchos trabajos en torno a StarCraft, el primer juego de la saga del mismo nombre lanzado en 1998, o a otros juegos de estrategia en tiempo real.

Si bien, ya hace 10 años que StarCraft II fue lanzado al mercado, en la actualidad siguen apareciendo nuevos trabajos enfocados sobre su precuela, StarCraft, ya que ambos son catalogados como *benchmarks* importantes en cuanto a investigación de inteligencias artificiales capaces de “jugar” el videojuego (Wang L. , Zeng, Chen, Pan, & Cao, 2020). Con la finalidad de obtener la victoria, es que surgen investigaciones que atacan distintos puntos del juego en la búsqueda de nuevos algoritmos o acercamientos que mejoren estas inteligencias artificiales y, en algunos casos, ayuden a los jugadores humanos. En los enfoques estudiados se encuentran el

uso de *Machine Learning* en una inteligencia artificial que tras perfeccionarse a través de diversos tipos de aprendizaje logró llegar a altos rangos en la tabla de clasificación de StarCraft II al quedar por sobre el 99,8% de los jugadores (Vinyals, y otros, 2019); el análisis de datos de repeticiones publicadas por Blizzard Entertainment con el fin de encontrar órdenes de construcción óptimas en StarCraft luego de comparar un acercamiento dirigido por datos con un acercamiento tradicional, ambos bajo el enfoque de un algoritmo genético y probados en enfrentamientos entre Terran contra Terran, llegando a la conclusión de que el primer acercamiento obtenía mejores resultados que uno tradicional (Wang P. , Zeng, Chen, & Cao, 2019); o la evaluación de la efectividad de un conjunto de unidades frente a otro utilizando *fuzzy integrals* en un algoritmo genético particular al comparar la precisión con que el algoritmo acierta el grupo de unidades vencedor de miles de enfrentamientos en un simulador, aunque el algoritmo falla al no tomar en cuenta algunos elementos del juego (Wang L. , Zeng, Chen, Pan, & Cao, 2020).

Actualmente, no existe una solución que permita encontrar buenos órdenes de construcción en StarCraft II para llegar a una unidad especificada por un jugador humano, aunque hay investigaciones de hace algunos años que buscan que inteligencias artificiales logren decidir cuál de los órdenes de construcción preestablecidos usar según la información obtenida del enemigo en la partida, esto al trabajar con la inteligencia artificial de libre acceso *UAlbertaBot* e integrarle un algoritmo de selección de órdenes de construcción que tiene en cuenta la información actual de la partida, cuyo resultado fue una mejora en la inteligencia artificial mencionada, logrando derrotar a la inteligencia artificial del juego un 91,7% de las partidas, pero sin ser muy eficaz ante estrategias de ataque temprano o *rush* (Justesen & Risi, 2017).

También existen otras investigaciones que entregan un análisis al usuario acerca del mejor orden de construcción según la cantidad de recursos en base a órdenes de construcción, pero que también son predeterminados, esto fue probado al aplicar el algoritmo (una versión modificada del NSGA II) a las tres razas y simular una partida de 450 segundos, llegando a la conclusión de que esta herramienta resulta útil para que los jugadores mejoren sus estrategias (Köstler & Gmeiner, 2013).

Al buscar en trabajos más antiguos se encuentran algoritmos para la predicción de las estrategias de los oponentes al utilizar la información recopilada en tiempo real por una unidad que explora el escenario en busca del enemigo. Esto aplicado con el objetivo de mejorar el algoritmo de una inteligencia artificial (Park, Lee, Cho, & Kim, 2012) e, incluso, la creación automática de escenarios para el juego (Togelius, y otros, 2010).

A través de esta revisión sistemática se puede observar que gran parte de la investigación en torno al juego se aplica en las inteligencias artificiales que compiten entre ellas, dejando a los jugadores humanos en segundo plano y sin muchas soluciones que les ayuden a obtener la victoria o aumenten sus posibilidades de juego. Esto sin contar las escasas

opciones que les entrega el juego mismo, las cuales se pueden reducir a un tutorial muy básico para aprender las mecánicas necesarias y la campaña individual con situaciones de juego preestablecidas. Estos antecedentes dejan a un jugador, con intenciones de mejorar, la única opción de aprender por su cuenta al observar a profesionales o aprender de la experiencia al jugar en línea contra otros oponentes en sus mismas condiciones.

Es importante tener en mente que todos los acercamientos mencionados para StarCraft II en esta sección son anticuados dado que, al igual que muchos videojuegos competitivos en línea, existen parches obligatorios que son lanzados con distinta frecuencia cuyo objetivo principal es modificar el videojuego para equilibrar los aspectos del mismo (debilitando personajes muy fuertes o mejorando los más débiles), como también solucionar errores o agregar nuevas funcionalidades. Con esto, se hace relevante informar que en la actualidad Blizzard ha cesado de lanzar nuevos parches de actualización para StarCraft II.

1.3 DESCRIPCIÓN DEL PROBLEMA

Dada la contextualización anterior y como no existe una metodología o una solución con las que encontrar buenos órdenes de construcción, es que los jugadores no pueden mejorar sus estrategias sin dedicar algo de su tiempo en la búsqueda de órdenes de construcción. Por esto, surge la pregunta: **¿cómo encontrar órdenes de construcción de forma más eficiente que un orden de construcción obtenido por un jugador y en un tiempo computacionalmente razonable, que permitan conseguir una unidad, estructura o tecnología determinada para la raza Terran en StarCraft II?**

1.4 OBJETIVOS DEL PROYECTO

1.4.1 Objetivo general

Desarrollar un modelo, bajo un marco del algoritmo metaheurístico *simulated annealing*, que permita generar buenos órdenes de construcción en el videojuego StarCraft II para la raza Terran, optimizados con el criterio del tiempo que requieren estas secuencias para llegar a la unidad, estructura o tecnología deseada.

1.4.2 Objetivos específicos

1. Modelar el árbol de tecnologías de la raza Terran con sus características y requisitos necesarios para la aplicación del algoritmo.
2. Modelar los elementos del juego StarCraft II necesarios para la aplicación del algoritmo.
3. Implementar SA para la obtención de buenos órdenes de construcción.
4. Evaluar los resultados del algoritmo implementado según el criterio del tiempo.

1.5 METODOLOGÍA

Si bien se contempla un componente de desarrollo que acompañe al algoritmo obtenido, el foco del presente proyecto se encuentra en la investigación. Ante esto es que se propone trabajar bajo el marco de la metodología *Data Science Engineering Process* (Volk, y otros, 2020) que incluye una división de los pasos a seguir en procesos. Revisados los objetivos del proyecto y las características de la solución es que se proponen tres etapas para el desarrollo del modelo. La primera etapa consta de la representación computacional de la raza Terran y la información del juego. La segunda etapa trata de la aplicación de un algoritmo metaheurístico a fin de obtener las soluciones. En la tercera etapa se evalúan los resultados obtenidos de la etapa anterior. Transversalmente a esas tres etapas, se aplicará una etapa de documentación en dónde se redacta el informe correspondiente con la información pertinente al proyecto.

1.5.1 Hipótesis

Con el fin de tener un objetivo al cual llegar según se describe en la metodología, se requiere de una hipótesis que alinee con la pregunta problema descrita en la sección 1.3. Además, acompañada de los conocimientos entregados en el capítulo 2, la hipótesis que se busca comprobar es que **es posible encontrar buenos órdenes de construcción para la raza Terran en StarCraft II utilizando la metaheurística *Simulated Annealing***, tomando como "buen orden de construcción" a la definición dada en la sección 1.1.

1.5.2 Etapa 1: Representación

Engloba los objetivos específicos 1 y 2. En esta etapa se buscan representar computacionalmente los datos del juego necesarios para la aplicación del algoritmo simulated annealing, incluyendo los requisitos y características de las unidades, tecnologías y estructuras como indica el árbol de tecnologías, además de otros elementos, como la cantidad de recursos obtenidos en el tiempo, que influyen fuertemente en los órdenes de construcción.

1.5.3 Etapa 2: Implementación

Alineada con el objetivo específico 3, esta etapa tiene como propósito la aplicación de SA al utilizar la representación del paso anterior, además de los parámetros que deben ser definidos para uso del mismo algoritmo (temperatura inicial, probabilidad de aceptación, etc.). Para lograr esta implementación es que en esta fase se debe programar el algoritmo y ejecutarlo, obteniendo así los órdenes de construcción resultantes.

1.5.4 Etapa 3: Evaluación

Etapa que se asocia con el objetivo específico 4. En esta se evaluarán los resultados obtenidos de la implementación del algoritmo metaheurístico según el tiempo, comparándolo con los órdenes de construcción que sean utilizados por jugadores profesionales y jugadores competitivos (según sus rangos) de StarCraft II.

También se contempla la evaluación del SA al comparar los resultados entre sí, ya sea para las mismas o distintas entradas, para así analizar el comportamiento del algoritmo en cuanto a la calidad de sus soluciones según la función objetivo, indicada más adelante en el capítulo 3, que utiliza el tiempo de juego como uno de los factores para evaluarla.

1.5.5 Etapa de documentación

Esta etapa, que va paralela a las tres anteriores, involucra todos los objetivos específicos, pues es necesaria para evidenciar por completo los procesos y resultados del

proyecto.

1.6 AMBIENTE DE DESARROLLO Y HERRAMIENTAS

1.6.1 Ambiente de ejecución

A lo largo del proyecto se trabajará sobre un Notebook Asus modelo X556U, con un procesador Intel Core i5-7200U (2.5 GHz x4), memoria RAM de 8 GB, disco duro de 1 TB y una tarjeta gráfica integrada Intel HD Graphics 620. Todo esto con un sistema operativo Ubuntu 20.04.1 LTS.

1.6.2 Herramientas

Durante el desarrollo de la solución es necesaria la utilización de las siguientes herramientas:

- R: ambiente de desarrollo especializado para la programación estadística. Será utilizado para implementar la solución propuesta debido a su utilidad en la computación estadística, lo que es útil en el análisis de los resultados obtenidos, además posee la función de generar gráficos de forma nativa.
- RStudio: entorno de desarrollo que utiliza R como base. Será el entorno en el que se codificará debido a su interfaz gráfica que permite un fácil monitoreo de los datos.
- GitLab: herramienta de gestión de código fuente. Será utilizado como control de versiones del código.
- Overleaf: herramienta de procesamiento de textos en el lenguaje LaTeX. Será utilizado para redactar el informe.

1.7 ALCANCE Y LIMITACIONES

A continuación, se indican los alcances y limitaciones a los que está sujeta la solución propuesta.

- El algoritmo metaheurístico encuentra órdenes de construcción que aplican sólo para la raza Terran del juego StarCraft II, por lo que su aplicación a otras razas del mismo juego o a otros videojuegos escapa del alcance propuesto en este proyecto.
- La interfaz que se incluye en el proyecto tiene el único propósito de probar el algoritmo y sus resultados, por lo que no se espera aplicar ningún tipo de métricas de usabilidad ni la inclusión de un manual de usuario que facilite su uso.
- El modelo no tomará en cuenta elementos del juego como posicionamiento, unidades o estructuras perdidas, el tipo de escenario o cualquier elemento que involucre al oponente puesto que estos no son elementos controlables para la creación de un orden de construcción. Además, se asume que el jugador siempre tiene acceso a la extracción de recursos.

Con esto en cuenta, para la evaluación de la solución se compararán el tiempo de la partida en los órdenes de construcción generados para obtener una unidad, estructura o tecnología específica con los utilizados por jugadores profesionales y competitivos según los datos de repeticiones obtenidos de diversos repositorios, como los listados en https://liquipedia.net/starcraft2/Replay_Websites, a modo de analizar si el modelo provee mejores órdenes de construcción que los jugadores competitivos en los distintos rangos del ranking o que un jugador profesional.

También se evaluará la eficiencia del algoritmo SA según el tiempo de ejecución que tarde para encontrar una solución, además de la convergencia de las generaciones. Todo esto para distintas entidades.

1.8 ESTRUCTURA DEL DOCUMENTO

En cuanto al formato de este documento, obviando las secciones de este Capítulo de introducción, se tiene el marco teórico que presenta definiciones del videojuego StarCraft II y de la metaheurística *Simulated Annealing*, pues son utilizadas en el resto del presente documento. Continúa el estado del arte que indica los acercamientos existentes para el problema u otros trabajos relacionados al videojuego de estrategia en tiempo real. Sigue la especificación del algoritmo propuesto, dónde se detallan los elementos que este contiene. El quinto Capítulo de los experimentos con sus resultados explica los pasos realizados con el algoritmo y los efectos que surgieron de ellos. Después se analizan estos resultados en el Capítulo 6, indicando como fue el comportamiento de estos. El último Capítulo del cuerpo del documento trata de las conclusiones, haciendo una retrospectiva con todos los elementos vistos hasta el momento. Luego vienen el

Glosario con algunos de los términos más importantes y las referencias a los trabajos externos revisados en este documento.

CAPÍTULO 2. MARCO TEÓRICO

Antes de avanzar en los capítulos de este documento, es de vital importancia aclarar los conceptos utilizados de modo que exista una base de conocimiento que permita la total comprensión de los temas tratados.

2.1 STARCRAFT II

No todos los videojuegos son iguales y StarCraft II no es una excepción, por lo que detallar algunos de sus términos es de gran utilidad para la comprensión del documento.

2.1.1 Características generales

Similar a otros juegos del género de estrategia en tiempo real, StarCraft II consiste principalmente de construir un **ejército** al gestionar los distintos recursos no renovables que existen en el campo de batalla (o **mapa** de ahora en adelante) con la finalidad de vencer al ejército rival. Todo esto sin turnos establecidos como en otros tipos de juegos de estrategia, pues al tratarse de un juego en tiempo real cada segundo desaprovechado solamente beneficia al oponente y su creciente ejército. Conociendo ya este objetivo de vencer al ejército del oponente, se puede decir que la victoria en una partida se obtiene al derrotar a todas las fuerzas que componen a dicho ejército o cuando el oponente, por voluntad propia, decide rendirse antes de que caiga su ejército.

Antes de iniciar una partida de StarCraft II, cada jugador debe escoger una de las tres **razas** existentes en el videojuego, las cuales poseen elementos muy distintos entre ellas, pero que se rigen bajo las mismas reglas. Como el documento se basa en la raza Terran, las definiciones y los ejemplos presentados en los capítulos que proceden se enfocan en la raza Terran, aún cuando muchos de estos términos aplican también a las razas Protoss y Zerg.

Una vez iniciado el juego, cada jugador recibe los elementos básicos para comenzar la creación de su ejército: doce trabajadores (*SCV* en el caso de la raza Terran) y un edificio principal (*Command Center* para la raza Terran). A partir de aquí, el jugador tiene libertad para construir su ejército siguiendo las reglas de construcción o atacar a las fuerzas enemigas con sus propias unidades.



Figura 2.1: Captura de pantalla de una partida de StarCraft II

Fuente: (Blizzard Entertainment, 2021)

2.1.2 Árbol de tecnologías

Si bien existen distintos tipos de entidades que componen la milicia de un jugador, el videojuego los divide en tres categorías que engloban las características de estos: unidades, edificios y mejoras. Aunque existen fuertes similitudes entre todas estas categorías, como el hecho de que todos poseen **requisitos** de alguna índole, sus funciones principales son muy distintas entre sí.

Las **unidades** son todas aquellas entidades que son producidas por algún edificio, como se puede observar en la figura 2.1 en el recuadro rojo o en la parte central inferior. Su papel consiste, principalmente, en ser la fuerza de ataque de un ejército al movilizarse por el mapa, aunque sus funciones no se limitan sólo a atacar, pues también existen unidades conocidas como **trabajadores** (que en el caso de la raza Terran reciben el nombre de *SCV* o *Space Construction Vehicles*) los cuales se encargan de la recolección de recursos y también de la construcción de edificios. Al tratarse de entidades que pueden participar en combate, existen **estadísticas** (como salud, armadura o ataque) asignadas a cada tipo de unidad.

Los **edificios** son estructuras construidas por los trabajadores o evolucionadas de otros edificios, que se encuentran generalmente fijados a una posición en el mapa (escogida por

el jugador al momento de su creación) como se puede observar en los recuadros azul y verde de la figura 2.1. Su función primordial consiste en la creación de las distintas unidades, aunque su utilidad no se detiene ahí, ya que algunos edificios también poseen características defensivas, otros permiten la investigación de mejoras e incluso uno que aumenta la cantidad de **suministros** (necesarios para la producción de unidades). De igual manera que las unidades, los edificios poseen estadísticas ya que pueden ser destruidos, aunque sólo algunos tienen estadísticas de ataque.

Tal como indica su nombre, las **mejoras** ofrecen distintos tipos de beneficios al resto de entidades, los cuales van desde aumentos en sus estadísticas (como aumentos en el ataque o la defensa) hasta el desbloqueo de **habilidades** que sólo pueden usar unidades o edificios específicos. A diferencia de las unidades o los edificios, las mejoras sólo permiten la creación de una de cada mejora por partida. Como ejemplo se tiene que sólo se puede construir un *Infantry Weapons Level 1* (mejora Terran) a la vez que puedes tener 10 o 15 *Marines* (unidad Terran).

A causa de que a lo largo de este documento se suelen englobar estas tres categorías, es que a partir de ahora se referirá a las unidades, a los edificios y a las mejoras (tanto en conjunto como individualmente) cómo **entidades**, las cuales pueden ser identificadas junto a sus requisitos en la tabla 2.1. Es importante aclarar que aquellas casillas en donde el valor es -1, indica que ninguna entidad aplica para la entidad en la columna respectiva.

Tabla 2.1: Entidades Terran en StarCraft II

| ID | Nombre | Categoría | Tiempo de construcción | Construido de | Desbloqueado por | Mineral | Gas | Suministro | Evoluciona de |
|----|-------------------------|-----------|------------------------|---------------|------------------|---------|-----|------------|---------------|
| 1 | Supply Depot | Edificio | 30 | 20 | -1 | 100 | 0 | -1 | -1 |
| 2 | Refinery | Edificio | 30 | 20 | -1 | 75 | 0 | -1 | -1 |
| 3 | Command Center | Edificio | 100 | 20 | -1 | 400 | 0 | -1 | -1 |
| 4 | Barracks | Edificio | 65 | 20 | 1 | 150 | 0 | -1 | -1 |
| 5 | Engineering Bay | Edificio | 35 | 20 | 3 | 125 | 0 | -1 | -1 |
| 6 | Factory | Edificio | 60 | 20 | 4 | 150 | 100 | -1 | -1 |
| 7 | Bunker | Edificio | 40 | 20 | 4 | 100 | 0 | -1 | -1 |
| 8 | Ghost Academy | Edificio | 40 | 20 | 4 | 150 | 50 | -1 | -1 |
| 9 | Orbital Command | Edificio | 35 | 3 | -1 | 150 | 0 | -1 | 3 |
| 10 | Missile Turret | Edificio | 25 | 20 | 5 | 100 | 0 | -1 | -1 |
| 11 | Sensor Tower | Edificio | 25 | 20 | 5 | 125 | 100 | -1 | -1 |
| 12 | Planetary Fortress | Edificio | 35 | 3 | 5 | 150 | 150 | -1 | 3 |
| 13 | Armory | Edificio | 65 | 20 | 6 | 150 | 100 | -1 | -1 |
| 14 | Starport | Edificio | 50 | 20 | 6 | 150 | 100 | -1 | -1 |
| 15 | Fusion Core | Edificio | 65 | 20 | 14 | 150 | 150 | -1 | -1 |
| 16 | Marine | Unidad | 25 | 4 | -1 | 50 | 0 | 1 | -1 |
| 17 | Marauder | Unidad | 30 | 53 | -1 | 100 | 25 | 2 | -1 |
| 18 | Reaper | Unidad | 45 | 4 | -1 | 50 | 50 | 1 | -1 |
| 19 | Ghost | Unidad | 40 | 53 | 8 | 150 | 125 | 2 | -1 |
| 20 | SCV | Unidad | 17 | 3 | -1 | 50 | 0 | 1 | -1 |
| 21 | Hellion | Unidad | 30 | 6 | -1 | 100 | 0 | 2 | -1 |
| 22 | Widow Mine | Unidad | 30 | 6 | -1 | 75 | 25 | 2 | -1 |
| 23 | Siege Tank | Unidad | 45 | 55 | -1 | 150 | 125 | 3 | -1 |
| 24 | Cyclone | Unidad | 45 | 55 | -1 | 150 | 100 | 3 | -1 |
| 25 | Hellbat | Unidad | 30 | 6 | -1 | 100 | 0 | 2 | -1 |
| 26 | Thor | Unidad | 60 | 55 | -1 | 300 | 200 | 6 | -1 |
| 27 | Viking | Unidad | 42 | 14 | -1 | 150 | 75 | 2 | -1 |
| 28 | Medivac | Unidad | 42 | 14 | -1 | 100 | 100 | 2 | -1 |
| 29 | Raven | Unidad | 60 | 57 | -1 | 100 | 200 | 2 | -1 |
| 30 | Banshee | Unidad | 60 | 57 | -1 | 150 | 100 | 3 | -1 |
| 31 | Liberator | Unidad | 60 | 14 | -1 | 150 | 150 | 3 | -1 |
| 32 | Battlecruiser | Unidad | 90 | 57 | -1 | 400 | 300 | 6 | -1 |
| 33 | Tactical Nuke | Mejora | 60 | 8 | 6 | 100 | 100 | -1 | -1 |
| 34 | Personal Cloaking | Mejora | 120 | 8 | -1 | 150 | 150 | -1 | -1 |
| 35 | Enhanced Shock-waves | Mejora | 110 | 8 | -1 | 150 | 150 | -1 | -1 |
| 36 | Vehicle Weapons Level 1 | Mejora | 160 | 13 | -1 | 100 | 100 | -1 | -1 |
| 37 | Vehicle Weapons Level 2 | Mejora | 190 | 13 | -1 | 175 | 175 | -1 | 36 |

Fuente: Elaboración Propia, 2021

Tabla 2.2: Entidades Terran en StarCraft II (cont.)

| ID | Nombre | Categoría | Tiempo de construcción | Construido de | Desbloqueado por | Mineral | Gas | Suministro | Evoluciona de |
|----|----------------------------------|-----------|------------------------|---------------|------------------|---------|-----|------------|---------------|
| 38 | Vehicle Weapons Level 3 | Mejora | 220 | 13 | -1 | 250 | 250 | -1 | 37 |
| 39 | Vehicle and Ship Plating Level 1 | Mejora | 160 | 13 | -1 | 100 | 100 | -1 | -1 |
| 40 | Vehicle and Ship Plating Level 2 | Mejora | 190 | 13 | -1 | 175 | 175 | -1 | 39 |
| 41 | Vehicle and Ship Plating Level 3 | Mejora | 220 | 13 | -1 | 250 | 250 | -1 | 40 |
| 42 | Weapon Refit | Mejora | 140 | 15 | -1 | 150 | 150 | -1 | -1 |
| 43 | Advanced Ballistics | Mejora | 110 | 15 | -1 | 150 | 150 | -1 | -1 |
| 44 | Rapid Reignition System | Mejora | 80 | 15 | -1 | 100 | 100 | -1 | -1 |
| 45 | Infantry Weapons Level 1 | Mejora | 160 | 5 | -1 | 100 | 100 | -1 | -1 |
| 46 | Infantry Weapons Level 2 | Mejora | 190 | 5 | 13 | 175 | 175 | -1 | 45 |
| 47 | Infantry Weapons Level 3 | Mejora | 157 | 5 | 13 | 250 | 250 | -1 | 46 |
| 48 | Infantry Armor Level 1 | Mejora | 160 | 5 | -1 | 100 | 100 | -1 | -1 |
| 49 | Infantry Armor Level 2 | Mejora | 190 | 5 | 13 | 175 | 175 | -1 | 48 |
| 50 | Infantry Armor Level 3 | Mejora | 157 | 5 | 13 | 250 | 250 | -1 | 49 |
| 51 | Hi-Sec Auto Tracking | Mejora | 80 | 5 | -1 | 100 | 100 | -1 | -1 |
| 52 | Neosteel Armor | Mejora | 140 | 5 | -1 | 150 | 150 | -1 | -1 |
| 53 | Barracks with Tech Lab | Edificio | 25 | 4 | -1 | 50 | 25 | -1 | 4 |
| 54 | Barracks with Reactor | Edificio | 50 | 4 | -1 | 50 | 50 | -1 | 4 |
| 55 | Factory with Tech Lab | Edificio | 25 | 6 | -1 | 50 | 25 | -1 | 6 |
| 56 | Factory with Reactor | Edificio | 50 | 6 | -1 | 50 | 50 | -1 | 6 |
| 57 | Starport with Tech Lab | Edificio | 25 | 14 | -1 | 50 | 25 | -1 | 14 |
| 58 | Starport with Reactor | Edificio | 50 | 14 | -1 | 50 | 50 | -1 | 14 |
| 59 | Ship Weapons Level 1 | Mejora | 160 | 13 | -1 | 100 | 100 | -1 | -1 |
| 60 | Ship Weapons Level 2 | Mejora | 190 | 13 | -1 | 175 | 175 | -1 | 59 |
| 61 | Ship Weapons Level 3 | Mejora | 220 | 13 | -1 | 250 | 250 | -1 | 60 |
| 62 | Tech Lab in Barracks | Edificio | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 63 | Tech Lab in Factory | Edificio | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

Fuente: Elaboración Propia, 2021

Tabla 2.3: Entidades Terran en StarCraft II (cont.)

| ID | Nombre | Categoría | Tiempo de construcción | Construido de | Desbloqueado por | Mineral | Gas | Suministro | Evoluciona de |
|----|-----------------------|-----------|------------------------|---------------|------------------|---------|-----|------------|---------------|
| 64 | Tech Lab in Starport | Edificio | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 65 | Combat Shield | Mejora | 110 | 62 | -1 | 100 | 100 | -1 | -1 |
| 66 | Stimpack | Mejora | 140 | 62 | -1 | 100 | 100 | -1 | -1 |
| 67 | Concussive Shells | Mejora | 60 | 62 | -1 | 50 | 50 | -1 | -1 |
| 68 | Infernal Pre-Igniter | Mejora | 110 | 63 | -1 | 100 | 100 | -1 | -1 |
| 69 | Mag-Field Accelerator | Mejora | 140 | 63 | -1 | 100 | 100 | -1 | -1 |
| 70 | Drilling Claws | Mejora | 110 | 63 | -1 | 75 | 75 | -1 | -1 |
| 71 | Smart Servos | Mejora | 110 | 63 | -1 | 100 | 100 | -1 | -1 |
| 72 | Corvid Reactor | Mejora | 110 | 64 | -1 | 150 | 150 | -1 | -1 |
| 73 | Cloaking Field | Mejora | 110 | 64 | -1 | 100 | 100 | -1 | -1 |
| 74 | Hyperflight Rotors | Mejora | 170 | 64 | -1 | 150 | 150 | -1 | -1 |

Fuente: Elaboración Propia, 2021

Entendiendo las tres categorías de entidades, es que ahora, siguiendo las restricciones de la tabla 2.1, se puede definir al **árbol de tecnologías** como las relaciones que tiene cada entidad con respecto al resto, el cual nos muestra los **requisitos** para desbloquear cada entidad (entidades que se necesitan antes de que la unidad en cuestión pueda ser creada) y su **entidad de creación** (entidad que se encarga de la creación de la entidad en cuestión, que en algunas ocasiones se transforma en esta entidad) tal como se puede observar en la figura 2.2.

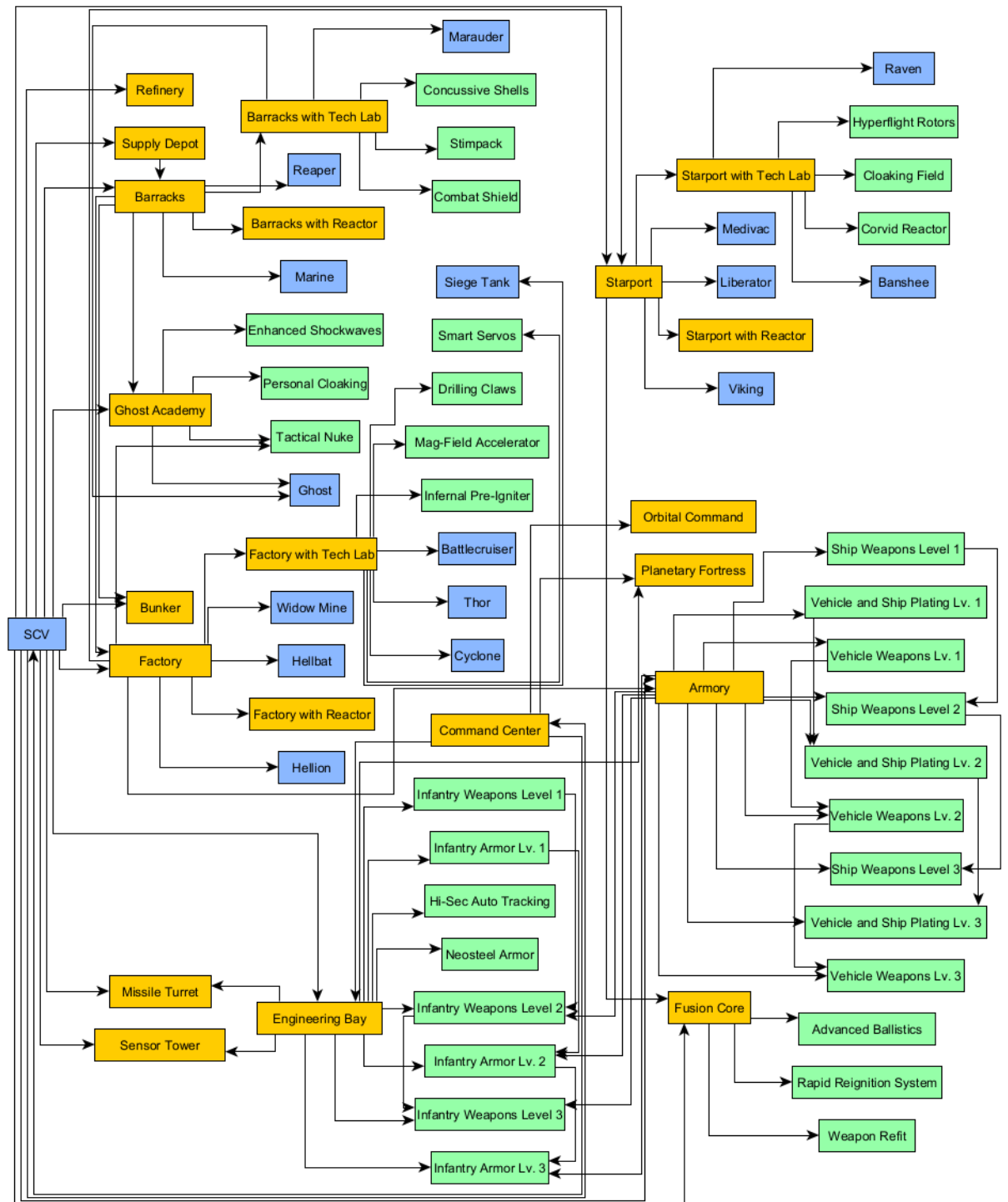


Figura 2.2: Árbol de tecnologías Terran

Fuente: (Elaboración propia, 2021)

2.1.3 Economía

Cómo todo en la vida, cualquiera de las fuerzas previamente mencionadas viene con un costo que varía según la utilidad de cada una. Existen dos tipos de **recursos** no renovables (minerales y gas vespeno), representados en el juego por números enteros como se puede observar en la esquina superior derecha de la figura 2.1, que son obtenidos por medio de los trabajadores y que son sustraídos cada vez que la creación de una entidad lo requiera y en las cantidades que se especifique. Es importante aclarar que estos recursos no pueden ser menores que cero en ningún caso.

Los **minerales** son la fuente principal de recursos en StarCraft II, pues todas las entidades lo requieren para comenzar su proceso de creación. Estos minerales son extraídos directamente de los campos de minerales que se encuentran en el mapa por los *SCV* como se puede observar en el recuadro amarillo de la figura 2.1, para luego ser llevados a un *Command Center* (o cualquier edificio que evolucione de este) en donde pasan a ser contabilizados en la cantidad de minerales obtenidos.

Al comienzo de la partida, la base del jugador inicia en la cercanía de cuatro campos de minerales grandes (que otorgan un máximo de 1800 minerales cada uno) y cuatro pequeños (900 minerales cada uno) resultando en un máximo total de 10800 minerales por base. Si bien estos recursos son limitados y no renovables, el jugador puede encontrar nuevos campos de minerales a lo largo del mapa en las mismas cantidades que los que aparecen en la base inicial, para los cuales es recomendable construir un nuevo *Command Center* que permita depositar el mineral de forma más expedita.

A la hora de extraer los minerales, un *SCV* tarda 1.99 segundos desde que comienza hasta que obtiene los 5 de mineral que puede cargar para, luego, esperar por, aproximadamente, 0.3571 segundos antes de devolverse al *Command Center* para depositarlos y generar un **ingreso** de 5 minerales. Debido a que sólo se admite un trabajador por mineral a la vez, es que ocurre un fenómeno denominado **saturación** en dónde se busca maximizar el ingreso al minimizar la aparición de colas de trabajadores que intentan acceder a un mismo mineral, pues cada segundo a la espera es un segundo perdido. Básicamente, hasta dos *SCV* pueden encargarse de un mismo mineral sin que tengan que esperar a que el otro finalice (ambos tienen una **eficiencia** del 100% en cuanto al ingreso que obtienen), pero, al ser el objetivo principal el de recolectar la mayor cantidad posible de mineral en el tiempo, es que se ha calculado que el óptimo es de 3 trabajadores por mineral (demostrado visualmente en el juego con un total de 24 *SCV* por base), donde el tercer trabajador recibe menos minerales por unidad de tiempo, pero se maximiza el ingreso total por cada mineral. Con esto en mente, los cálculos realizados por un usuario de la comunidad de StarCraft II (PiousFlea, 2010) arrojan un total aproximado de 102

minerales por minuto al contar con 3 trabajadores por mineral.

Un caso particular ocurre con la unidad *MULE* que puede obtener 4 veces los minerales que un *SCV* y no satura los minerales (ignora las colas), pero con la diferencia de que es una unidad temporal que desaparece pasados 65 segundos desde su creación, cuyo único requisito es la utilización de energía desde un *Orbital Command*. La **energía** es un recurso renovable que es característico sólo a algunas entidades como el *Orbital Command*, pero que sólo es utilizado para la creación instantánea de otras entidades por este último en la creación de un *MULE* o la investigación de *Extra Supplies*, mientras que el resto de las entidades que poseen energía, la gastan en el uso de distintas habilidades. Las entidades suelen comenzar con 50 de energía, la cual aumenta en 0.7875 cada segundo hasta un máximo de 200, siendo este recurso independiente para cada instancia de entidad.

Por otro lado, el **gas vespeno** es un recurso más limitado (sólo dos **géiseres** de gas por base) que no necesitan todas las unidades, pero que no deja de ser vital para construir entidades más poderosas que las que requieren solamente del mineral. A diferencia del mineral que puede ser extraído directamente de los campos de mineral por los trabajadores, el gas vespeno necesita de la creación de un *Refinery* para que los *SCV* puedan extraer el recurso, como se presenta en el recuadro verde de la figura 2.1, y llevarlo al *Command Center* para que se agregue a la cantidad de gas total.

Para la extracción de gas, un *SCV* tarda 1.415 segundos en la extracción del recurso desde el *Refinery* y se dirige inmediatamente a la base a depositar un ingreso igual a los 4 de gas vespeno. Al igual que los minerales, el gas vespeno alcanza la saturación a los 3 trabajadores (total de 6 por base), dando un total de 114 de gas por minuto.

Catalogado también como recurso, aunque con un comportamiento muy distinto al de los dos anteriormente vistos, se tiene el suministro, que es un requisito exclusivo de las unidades. El **suministro** es un valor entero que limita la cantidad de unidades que se pueden poseer, pues cada una tiene un valor definido que va desde 1 hasta 6 y que se suma para obtener la cantidad actual de suministro del ejército de cada jugador y que debe ser menor que el **suministro total** que ofrecen algunos edificios. El suministro total es simplemente la suma de 15 veces la cantidad de *Command Centers* o edificios que evolucionen de este más 8 veces la cantidad de *Supply Depots* (edificio) o 16 veces si es que el *Supply Depot* tiene un *Extra Supplies* vinculado. En el caso de agregar una unidad a la cola de creación de cualquier edificio y que el valor de los suministros actuales (contando a la unidad que se desea construir) exceda el de los suministros totales, esta unidad no podrá ser puesta en cola hasta que los suministros totales aumenten.

2.1.4 Velocidad de juego

Al tratarse de un juego en tiempo real es normal que las acciones sean medidas en segundos o fracciones de estos, pero dado que StarCraft II introduce la función de cambiar la **velocidad de juego** es que se debe explicar en que consiste para poner todo bajo el mismo contexto. El videojuego ofrece cinco velocidades de juego que pueden ser escogidas al crear una partida personalizada o es establecida en la velocidad más alta en caso de que se trate de una partida con emparejamiento automático (partidas clasificatorias). Los **factores de velocidad** de cada velocidad de juego, cuyos valores están indicados en la tabla 2.4, dividen los segundos reales para calcular el tiempo de los elementos del juego que lo requieran.

Tabla 2.4: Tabla de velocidades de juego

| Velocidad de Juego | Factor de Velocidad | Segundos de Tiempo Real en 1 Minuto de Juego <i>Normal</i> |
|--------------------|---------------------|------------------------------------------------------------|
| Slower | 0.6 | 100 |
| Slow | 0.8 | 75 |
| Normal | 1.0 | 60 |
| Fast | 1.2 | 50 |
| Faster | 1.4 | 42.86 |

Fuente: Elaboración Propia, 2021

Para ejemplificar lo anterior, se puede decir que un *Supply Depot*, que tarda 30 segundos reales en crearse a una velocidad de juego *normal*, tardará aproximadamente 21 segundos reales (21.423, pero el juego aproxima estos valores a un número entero) al utilizar la velocidad de juego *faster*.

La lectura de este documento debe ser abordada teniendo en cuenta la velocidad de juego *faster* como la velocidad por defecto, que es la que se utiliza en partidas clasificatorias y en torneos oficiales. Con esto presente, es importante aclarar que los valores en tiempo mencionados en las secciones anteriores, así como en las secciones por venir, son los que se obtienen en el juego al usar esta velocidad de juego *faster*, lo cual afecta principalmente a los tiempos que tardan en crearse las entidades y los tiempos en que los trabajadores recolectan recursos, pero que al tratarse de un factor escalar, es posible la conversión de estos datos a la velocidad de juego deseada. Además, se debe tener presente que la unidad básica a utilizar será 1 segundo en tiempo real, siempre bajo el marco de la velocidad de juego *faster*.

2.1.5 Órdenes de construcción

Si bien ya han sido mencionados a lo largo del documento, aún no se establece una definición concreta sobre los órdenes de construcción.

Cada vez que un jugador comienza la producción de una unidad, la construcción de un edificio o la investigación de una mejora se dice que el jugador está realizando una **acción**, la cual ocurre en un tiempo determinado contado a partir del inicio de la partida. Al reunir estas acciones en una secuencia, ordenada según sus marcas de tiempo respectivas, se tiene lo que se denomina **orden de construcción** (conocido en inglés como *Build Order*), el cual se muestra al final de cada partida. Este término no es usado solamente para referirse a las secuencias que aparecen después de una partida, sino que también se habla de estas series de acciones para referirse a órdenes de construcción "ideales", los cuales tienen las mismas características, pero son secuencias ideales que un jugador debe seguir para alcanzar una o varias entidades deseadas.

Es habitual en la comunidad de StarCraft II que los jugadores compartan estos órdenes de construcción con la finalidad de que sean evaluados por otros jugadores, por lo que no es extraño que se observen distintos formatos a la hora de presentarlos, algunos con elementos extra que facilitan la visualización y comprensión de los mismos. A modo de ejemplo de esto, se puede observar en la figura 2.3 como un orden de construcción puede tener otros valores además de los dos principales (el tiempo y la entidad construida) como la cantidad de suministros máximos y la cantidad de suministros actuales para entregar mayor contexto de la partida, así cómo también existen formatos que utilizan esto último para no incluir la creación de trabajadores con el fin de acortar el tamaño total del orden de construcción.

| | | |
|----|------|-------------------------|
| 14 | 0:17 | Supply Depot |
| 15 | 0:40 | Barracks |
| 16 | 0:43 | Refinery |
| 19 | 1:27 | Reaper, Orbital Command |
| 20 | 1:40 | Command Center |
| 20 | 1:51 | Supply Depot |
| 21 | 1:59 | Marine |
| 22 | 2:08 | Factory |
| 23 | 2:19 | Bunker |
| 23 | 2:20 | Barracks Reactor |
| 24 | 2:24 | Refinery |
| 26 | 2:52 | Orbital Command |
| 26 | 2:53 | Starport |
| 26 | 2:57 | Marine x2 |
| 29 | 2:59 | Widow Mine |
| 32 | 3:14 | Marine x2 |

Figura 2.3: Parte de un orden de construcción

Fuente: (Spawning Tool, 2021)

2.2 STARCRAFT II COMO PROBLEMA DE OPTIMIZACIÓN

El **problema de planificación de proyecto con recursos limitados** (Artigues, 2008), mejor conocido como RCPSPP (*Resource Constrained Project Scheduling Problem*), es un problema de optimización combinatorial cuya solución está definida por un espacio de solución \mathcal{X} que posee un subconjunto de soluciones posibles $\mathcal{Y} \subseteq \mathcal{X}$ asociado a una función objetivo $f : \mathcal{Y} \rightarrow \mathbb{R}$. El objetivo aquí es encontrar una solución $y \in \mathcal{Y}$ tal que la función objetivo $f(y)$ sea optimizada.

Este tipo de problema de optimización, que es representado por una tupla $(V, p, E, \mathcal{R}, B, b)$, involucra el manejo de los distintos requisitos obligatorios asociados a un problema con el fin de obtener una **planificación** de duración mínima, siempre respetando las relaciones de precedencia existentes y la disponibilidad de los recursos.

Un RCPSPP se considera como tal al poseer **actividades** $V = \{A_0, \dots, A_{n+1}\}$ que componen el problema, cuyo inicio se representa como una marca de tiempo denominada **schedule**, y que pueden ser vistas como hitos dentro de esta. Además, cada una de estas actividades tiene una **duración** asignada, representada por un vector p en \mathbb{N}^{n+2} donde p_i es la duración de A_i , y **relaciones de precedencia** E que obligan a que una actividad A_j necesite de

una previa actividad A_i antes de comenzar representado por $(A_i, A_j) \in E$. Estas actividades no solo requieren de otras, sino que, también, se ven afectadas por **recursos** $\mathcal{R} = \{R_1, \dots, R_q\}$ con distinta **disponibilidad** dada por B en \mathbb{N}^q , cuya **demanda** se representan por b con b_{ik} como la cantidad de recurso R_k usado por A_i .

Con esto en mente y luego de contrastar el RCPSP con las características del videojuego, quedan en evidencia las similitudes entre ambos problemas, permitiendo declarar el *problema de los órdenes de construcción* como un problema de planificación de proyecto con recursos limitados. La definición de *actividades* describe fielmente la creación de las distintas entidades que componen un orden de construcción, pues cada una de ellas tiene una *duración* particular desde que inicia su creación y *relaciones de precedencia* en las distintas entidades requisito que cada una posee. Y esto no es todo, pues la creación de entidades también *demanda* suministros, además de los *recursos renovables*, mineral y gas vespeno cuyas *disponibilidades* varían según la cantidad de trabajadores y su ubicación en el mapa. Para ejemplificar esto, se usa a la unidad *Ghost*, cuya creación tarda 29 segundos en un tiempo de juego *faster*, que debe ser construida en el edificio *Barracks*, pero también requiere que el edificio *Ghost Academy* haya sido creado antes de poder iniciar su propia creación, la cual necesita 150 de mineral, 125 de gas vespeno y al menos 2 de suministro disponible.

2.3 METAHEURÍSTICAS

Aún sin garantía de encontrar el óptimo global, las metaheurísticas permiten atacar problemas de optimización para obtener soluciones satisfactorias en un tiempo razonable (Young, 2009). La palabra **heurística** viene del griego *heuriskein* que se refiere al "arte de descubrir nuevas estrategias o reglas para resolver problemas", mientras que el *meta* en metaheurística significa "metodología de nivel superior", por lo que las **metaheurísticas** se definen como metodologías generales de nivel superior que pueden ser usadas como estrategias guía al diseñar heurísticas adyacentes para resolver problemas de optimización específicos.

Existen diversos métodos para clasificar a las metaheurísticas. Un ejemplo de esto es clasificarlas según su inspiración al separar aquellas metaheurísticas inspiradas en la naturaleza de las no inspiradas en la naturaleza, pero en esta oportunidad usará la clasificación según el tipo de solución sobre la que trabaja el algoritmo: algoritmos basados en población y algoritmos basados en solución única (Young, 2009).

2.3.1 Algoritmos de población

Los **algoritmos de población** son un tipo de metaheurísticas, los cuales pueden ser vistos como una evolución de una población (conjunto) de soluciones que, de forma iterativa, se va integrando con la población de soluciones actual por medio del uso de procedimientos de selección, permitiendo una mejor diversificación en el espacio de búsqueda, de forma complementaria a los algoritmos de solución única. Ejemplos de este tipo de metaheurística se pueden ver en *algoritmos evolutivos*, *scatter search* y *algoritmos de estimación de distribución*.

2.3.2 Algoritmos de solución única

Aquellas metaheurísticas que, a diferencia de los algoritmos de población, buscan mejorar sólo una solución son denominadas como **algoritmos de solución única**, pues recorren el espacio de búsqueda saltando de una solución actual a otra de forma iterativa con la intención de acercarse al óptimo global al intensificar la búsqueda en regiones locales. Algunos ejemplos de este tipo de metaheurísticas son *Simulated Annealing*, *Local Search* y *Tabu Search*.

Entre los conceptos comunes para las metaheurísticas con algoritmos de solución única se encuentran los **vecindarios**, estructuras de soluciones asociadas a una solución particular, o las **soluciones iniciales**, que son utilizadas para comenzar cada algoritmo y que puede ser obtenida de manera aleatoria o con un acercamiento *greedy*.

2.3.3 Elección de la metaheurística

Debido a que se trata con un problema del tipo RCPSP es que la utilización de metaheurísticas es habitual para resolverlo. Algunos de estos incluyen *Iterated Local Search* (Ramos, Olivares-Benitez Miranda-Gonzalez, 2021), *Simulated Annealing* (Nai-Hsin, Tsai-Ching Wei-Tong, 2019) y *Variable Neighborhood Search* (Kochetov Stolayar, 2003). De entre las metaheurísticas mencionadas se escoge sólo *Simulated Annealing* debido al contexto de la investigación como examen de titulación de Ingeniería de Ejecución en Computación e Informática, teniendo en cuenta que las otras 2 fueron escogidas en otros trabajos de forma paralela a este, con el fin de realizar una comparación a futuro entre ellos.

2.3.4 Simulated Annealing

La solución propuesta consta del modelo bajo un enfoque metaheurístico, que incluye un algoritmo capaz de encontrar buenos órdenes de construcción para llegar a una unidad, una tecnología o una estructura de la raza Terran en StarCraft II. Esto, según se especifique antes de la ejecución del algoritmo y optimizando el criterio del tiempo de partida para llegar a dicha unidad, estructura o tecnología. La metaheurística a utilizar será **Simulated Annealing** (SA a partir de ahora), la cual es una metaheurística de solución única que busca imitar el proceso de recocido de algunos metales para obtener el material en un estado más puro o, en este caso, acercarse al óptimo (Young, 2009). Si se ve como algoritmo, SA tiene como objetivo escapar del óptimo local de forma en que las soluciones no queden atrapadas en una solución particular. Comenzando con una solución inicial, SA genera un vecino en cada iteración, el cual es seleccionado siempre que mejore la función de costo, pero que en caso de que no mejore esta función, tiene una probabilidad de ser seleccionado según la temperatura (parámetro de control definido acorde al problema) y la diferencia entre los resultados de la función objetivo de las soluciones en comparación.

Algoritmo 2.1: Simulated Annealing

Input: s_0, T

```
1  $s \leftarrow s_0$ ; /* Generación de la solución inicial */
2 while Criterio de detención no satisfecho do
3   while Condición de equilibrio no satisfecha do
4     Se genera un vecino aleatorio  $s'$ ;
5      $\Delta E \leftarrow f(s') - f(s)$ ;
6     if  $\Delta E \leq 0$  then
7        $s \leftarrow s'$ ; /* Acepta la solución vecina */
8     end
9     else
10       $s \leftarrow s'$ ; /* Acepta  $s'$  con una probabilidad  $e^{-\frac{\Delta E}{T}}$  */
11    end
12  end
13   $T \leftarrow g(T)$ ; /* Actualización de temperatura */
14 end
```

Output: Mejor solución encontrada

2.4 I-RACE

Aún antes de ejecutar el algoritmo para optimizar las soluciones, es importante optimizar los parámetros que SA recibe con la finalidad de obtener los mejores resultados posibles; a este proceso recién descrito se le denomina **parametrización**. Ahora, utilizando el paquete de R *i-race* es posible conseguir esta parametrización de forma automática al entregarle conjuntos o rangos de posibles parámetros para que el algoritmo de *i-race* ejecute reiteradamente el algoritmo cuyos parámetros se busca optimizar, que, en este caso, se trata de SA aplicado al problema de los órdenes de construcción, hasta que se cumpla el **presupuesto** o número de ejecuciones previamente ajustado. Finalmente se obtendrá una tabla con los mejores parámetros posibles de entre los rangos entregados, el cual debe ser utilizado en las futuras ejecuciones del algoritmo SA.

El paquete *i-race* (López-Ibañez, y otros, 2016) se basa en el método de carrera iterada, el cual consiste en 3 pasos: (1) el muestreo de nuevas configuraciones para una distribución particular, (2) selección de las mejores de estas nuevas configuraciones al ejecutarlas y (3) actualizar la distribución de las muestras en el orden de las mejores configuraciones. Como se indica en la figura 2.4, una carrera inicia con una cantidad finita de conjuntos de parámetros θ_i donde, en cada paso, son evaluadas en instancias I_j para descartar aquellos θ_i que desempeñan peor que el resto luego de un número T_k de instancias que es mayor para la primera eliminación. Este proceso continúa hasta alcanzar una cantidad mínima de configuraciones "sobrevivientes", un número máximo de instancias que han sido usadas o un presupuesto computacional predeterminado que puede ser tiempo promedio computacional o una cantidad de experimentos.

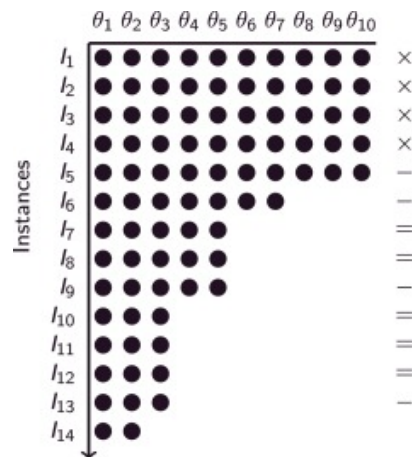


Figura 2.4: Carrera para la configuración automática de un algoritmo

Fuente: López-Ibañez, y otros, 2016

CAPÍTULO 3. ALGORITMO PROPUESTO Y EXPERIMENTOS

En este capítulo se detalla el algoritmo incluido en el modelo propuesto, además de los experimentos a realizar para probar la aplicación de este algoritmo.

3.1 ALGORITMO PROPUESTO

Con la finalidad de cumplir con lo estipulado en el objetivo principal, se requiere una explicación de los pasos, los cuales son detallados en este capítulo para darle forma a la solución computacional que apunta a resolver el problema.

El algoritmo *Simulated Annealing* tiene 3 parámetros de entrada que son necesarios para su ejecución, los cuales son la **temperatura** inicial, T , el **enfriamiento**, dado por el enfriamiento multiplicativo exponencial $T_0 * \alpha^k$, y la **solución inicial**, G_0 . Esto, sin contar con las entradas principal del modelo, los cuales son la **entidad objetivo**, E , para la que se requiere obtener un orden de construcción y la **cantidad** Q_E de la entidad objetivo que se desee. Además se opta porque los **ciclos** estén establecidos desde un inicio en los parámetros N para las iteraciones externas y M para los ciclos internos.

La solución inicial es aleatoria y con una duración en tiempo de juego, t , también aleatoria, pero con un tiempo de juego máximo definido previamente como un **intervalo**, I , que se aplica en cada solución. La cantidad Q_E debe ser un entero igual o mayor a 1. Por otro lado, la temperatura T es un entero al inicio, pero a lo largo del algoritmo puede tomar valores reales al ser multiplicado por α que es un real entre 0 y 1 sin incluir ninguno de estos.

Algoritmo 3.1: Simulated Annealing aplicado al problema de los órdenes de construcción

Input: $T_0, \alpha, G_0, N, M, I, E, Q_E$ **Output:** Mejor solución G_{Best} del orden de construcción

```
/* Inicialización de solución */
1  $P, P_{Best} \leftarrow funcion\_de\_evaluacion(G_0);$ 
2  $G_{SA}, G_{Best} \leftarrow oc\_aleatorio(G_0);$ 
3 for  $n = 1$  hasta  $N$  do
4    $T \leftarrow T_0 * \alpha^n;$ 
5   for  $m = 1$  hasta  $M$  do
6     /* Corte aleatorio de la solución actual */
7      $t_o \leftarrow entero\_aleatorio(0, t_{G_{SA}});$ 
8      $G \leftarrow cortar\_solucion(G_{SA}, t_o);$ 
9     /* Solución aleatoria a partir de la solución actual */
10     $t_f \leftarrow t_o + entero\_aleatorio(0, I);$ 
11     $G' \leftarrow oc\_aleatorio(G, t_o, t_f, E, Q_E);$ 
12    /* Cálculo de la función de evaluación y diferencia entre estos */
13     $P' \leftarrow funcion\_de\_evaluacion(G');$ 
14     $\Delta_P \leftarrow P' - P;$ 
15    /* Comparación con la solución actual */
16    if  $\Delta_P < 0$  then
17       $G_{SA}, P \leftarrow G', P';$ 
18    end
19    else
20       $prob \leftarrow exp(\Delta_P/T);$ 
21       $prob\_acep \leftarrow real\_aleatorio(0, 1);$ 
22      if  $prob\_acep < prob$  then
23         $G_{SA}, P \leftarrow G', P';$ 
24      end
25    end
26    /* Comparación con la mejor solución */
27    if  $P' < P_{Best}$  then
28       $G_{Best}, P_{Best} \leftarrow G', P';$ 
29    end
30  end
31 end
```

Teniendo presentes los operadores detallados en la sección 3.1.5, en el algoritmo 3.1 se distingue la existencia de dos ciclos, uno externo y otro interno, con N y M iteraciones respectivamente.

El ciclo interno (desde la línea 5 a la 25) inicia creando una solución G' vecina a partir de la solución G_{SA} cortado en un tiempo de juego aleatorio t_o dentro de su duración con un largo igual a $t_o + t_f$, con t_f un tiempo de juego aleatorio entre 0 y el intervalo máximo I . Luego se calcula la función de evaluación P' de dicha solución y la diferencia ΔP de este con la función de evaluación actual P con la cual se calcula la probabilidad de aceptar la solución con puntaje P' en caso de que este sea mayor al de P , pues una solución menor siempre se acepta. Luego se reemplaza en el caso de que la solución G' tenga un mejor puntaje que G_{Best} .

El ciclo externo se compone por el ciclo interno y la línea 4 en donde la temperatura se actualiza según el calendario de enfriamiento dado por α , lo cual ocasiona que la probabilidad de aceptación en caso en que ΔP sea menor tras cada iteración externa.

Con lo anterior, y suponiendo que tanto el ciclo interno como el externo son n , la eficiencia algorítmica tendría un orden de $O(n^4)$ debido al orden de $O(n^2)$ que posee la línea 9 con el orden de construcción aleatorio según lo indicado más adelante en la sección 3.1.5.1.

3.1.1 Representación de la solución

Si bien el algoritmo *Simulated Annealing* ya está diseñado, es necesario tomar los elementos del videojuego que se incluyen en la solución e implementarlos para poder aplicar esta metaheurística sobre los órdenes de construcción. Un caso importante de esto, es que las entidades tienen asignado un número entero para cada una, el cual es utilizado en gran parte de las representaciones a continuación.

Lo más importante a representar son las **soluciones** del problema que obtiene internamente el algoritmo hasta entregar la mejor. Cada solución es dividida en cinco tablas con los datos requeridos para poder generar nuevas soluciones y se explican en las secciones a continuación.

3.1.1.1 Acciones

La tabla de **acciones** que es, básicamente, el orden de construcción ya que guarda la entidad a construir y la marca de tiempo (en velocidad *faster* y como entero) de cada entidad que se crea, repitiéndose en cada fila de la tabla. En la tabla 3.1 se puede observar una fracción de las acciones obtenidas por una solución.

Tabla 3.1: Ejemplo de acciones

| Entidad | Marca de Tiempo |
|--------------|-----------------|
| SCV | 2 |
| Refinery | 10 |
| SCV | 17 |
| Supply Depot | 32 |
| Refinery | 42 |
| ... | ... |

Fuente: Elaboración Propia, 2021

3.1.1.2 Edificios

Se tiene también una tabla para los **edificios** que indica el estado de estos en un segundo particular de la solución, la cual almacena los edificios creados y tiene **espacios de creación** disponibles para cada edificio (con la posibilidad de un segundo espacio para aquellos edificios a los que se les acople un *Reactor* que pueden construir dos unidades a la vez) con su **utilización** asociada para medir el tiempo (representado por un entero) restante en la creación de cada unidad, la cual se actualiza en cada segundo de la solución; el último valor es el de la **energía**, que nos muestra la energía actual del edificio en cuestión y es un número flotante.

Para los espacios de creación existe un valor 0 que representa cuando dicho espacio no está siendo utilizado y -1 en caso de que el espacio no esté habilitado. La utilización se reduce en 1 por cada segundo real, excepto cuando es 0, pues significa que la entidad ya fue construida o simplemente no existe una en creación. Finalmente, la energía es un flotante con un tope de 200 y que aumenta 0.7875 por segundo (ver sección 2.1.3), pero que sólo se cuenta para el *Orbital Command* pues posee el único uso de este recurso que afecta a los órdenes de construcción.

Tabla 3.2: Ejemplo de edificios

| Edificio | Espacio 1 | Utilización 1 | Espacio 2 | Utilización 2 | Energía |
|----------------|-----------|---------------|-----------|---------------|---------|
| Supply Depot | 0 | 0 | -1 | 0 | 0 |
| Refinery | 0 | 0 | -1 | 0 | 0 |
| Refinery | 0 | 0 | -1 | 0 | 0 |
| Command Center | 0 | 0 | -1 | 0 | 0 |
| Barracks | Marine | 17 | -1 | 0 | 0 |

Fuente: Elaboración Propia, 2021

3.1.1.3 Trabajadores

Con un acercamiento similar a los edificios, la tabla de **trabajadores** muestra el estado de los *SCV* y los *MULE* en un segundo particular. Existe una columna para los **espacios de creación** que funciona de igual manera que en los edificios, además de la utilización para cada edificio que se esté creando. Lo que separa esta tabla de la anterior, es que no existe la energía, ni una segunda columna para otro espacio de creación o su utilización asociada, pero en su lugar existe una **duración** que indica el tiempo restante (como entero) que le queda al *MULE* antes de desaparecer.

A diferencia de los edificios, los espacios de creación pueden tomar un valor mayor que 0, igual al edificio que representan; 0 en caso de que el trabajador no esté siendo utilizado; -1 en caso de que se encuentre extrayendo minerales; o -2 en caso de que se encuentre recolectando gas vespeno. A continuación se muestra un fragmento de la tabla de trabajadores.

Tabla 3.3: Ejemplo de trabajadores

| Trabajador | Espacio | Utilización | Duración |
|------------|--------------|-------------|-----------|
| SCV | Gas | 0 | No aplica |
| SCV | Gas | 0 | No aplica |
| SCV | Gas | 0 | No aplica |
| SCV | Supply Depot | 5 | No aplica |
| SCV | Minerales | 0 | No aplica |
| SCV | Minerales | 0 | No aplica |
| ... | ... | ... | ... |

Fuente: Elaboración Propia, 2021

3.1.1.4 Entidades

Esta tabla indica las **entidades** que existen en un punto determinado de cada solución al tener una columna con cada **entidad** de la raza Terran, además de una columna con la **cantidad** de cada una de estas. El fin de esta tabla es la de reducir tiempo de procesamiento al no tener que contar las entidades cada vez que se necesite saber la cantidad de alguna de estas.

Tabla 3.4: Fragmento de ejemplo de entidades

| Identificador de entidad | Cantidad |
|--------------------------|----------|
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 1 |
| 5 | 1 |
| ... | ... |
| 14 | 0 |
| ... | ... |
| 17 | 1 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 18 |

Fuente: Elaboración Propia, 2021

3.1.1.5 Recursos

Una tabla que muestra los **recursos** en cada segundo de la solución. Cada fila posee el **tiempo** que avanza de 1 en 1, la cantidad flotante de **minerales**, la cantidad flotante de **gas vespeno** y la cantidad de **suministros máximos** como un entero.

Aunque los minerales y gas vespeno en el juego sólo toman valores enteros, se utilizan decimales debido al aproximado de recolección por segundo. Para el tiempo, se utiliza 0 como el valor al comenzar la partida y mayor que 0 para todos los segundos que le siguen.

La tabla 3.5 muestra un ejemplo de los recursos en el tiempo hasta el segundo 5 de juego.

Tabla 3.5: Ejemplo de recursos

| Tiempo (en segundos) | Minerales | Gas Vespeno | Suministros |
|----------------------|-----------|-------------|-------------|
| 0 | 50.000 | 0.000 | 0 |
| 1 | 57.500 | 0.000 | 15 |
| 2 | 65.000 | 0.000 | 15 |
| 3 | 22.500 | 0.000 | 15 |
| 4 | 30.000 | 0.000 | 15 |
| 5 | 37.500 | 0.000 | 15 |

Fuente: Elaboración Propia, 2021

3.1.2 Árbol de tecnologías

El árbol de tecnologías se ve representado en una tabla cuyas filas representan el **identificador** de cada entidad, las cuales incluyen el **nombre** de cada entidad a fin de presentarlas cuando corresponda, el **tiempo de construcción** como un entero, la **entidad de construcción** como un entero correspondiente a la fila de la entidad, la cantidad (como entero) de los distintos **recursos** que cada entidad necesita para construirse, además de los requisitos como si requieren una entidad previa que les **desbloquee**, si es que es una **mejora** de otra mejora ya existe o si es un edificio que **evolucione** de otro. Esta tabla se encarga de proveer la información particular de cada unidad, tecnología y mejora para los distintos operadores de la solución, volviéndose una tabla de lectura a diferencia de las tablas que componen las generaciones, por lo que, cuando se requiere conocer un camino desde el inicio de la partida hasta una entidad en específico, aquellos datos de esta tabla que muestran las relaciones de precedencia también se representan como un **grafo acíclico dirigido**.

A excepción del nombre, todos los datos toman valores enteros, siendo 0 el mínimo y -1 en caso de que dicha variable no aplique para alguna entidad (por ejemplo, un *Marine* no es desbloqueado por otra entidad, ni se transforma desde alguna entidad particular). Como ya se explicaba, el identificador de cada entidad está dado por el número de fila al que corresponda, iniciando con un edificio que no existe en el juego (*Townhall* en la posición 1) a modo de tener el grafo, formado por este árbol de tecnologías, unido en un mismo nodo inicial.

3.1.3 Criterios de optimización

Existen diversos criterios que pueden ser optimizados en un orden de construcción, siendo el *tiempo de juego* para la entidad a la que se desea llegar el criterio principal que se busca optimizar en este trabajo, dejando a un lado criterios como la cantidad de entidades totales obtenidas, que es inversamente proporcional al tiempo de juego debido a que más entidades requerirán una mayor cantidad de tiempo, o la fuerza del ejército formado, lo cuál requiere algo más de trabajo pues esta *fuerza* no es un valor que el juego entregue ni que se pueda deducir objetivamente o con facilidad.

Teniendo en cuenta lo anterior, otro factor importante es la obtención de la entidad indicada en la cantidad deseada, aunque las soluciones aleatorias no siempre consiguen este objetivo. Es por este motivo que se utiliza el **porcentaje de entidades restantes** del orden de construcción para llegar, el cual indica cuantas entidades distintas restan para alcanzar el objetivo. Para ejemplificar este porcentaje, se tiene el caso del *Marine* que se construye en un *Barracks* que, a su vez, es construido por un *SCV* (construido en un *Command Center*) y requiere que un *Supply Depot* haya sido construido previamente. Si se cuentan la cantidad de entidades requeridas (dejando de lado el *SCV* y el *Command Center* que son entidades con las que se inicia la partida) se tiene un total de 3, incluido el *Marine*. Suponiendo que una solución sólo consigue construir el *Supply Depot*, se dice que el porcentaje restante es del 66.67%, el cual aumentaría a 75% en caso de que se buscara construir 2 *Marines* (4 entidades totales).

3.1.4 Factibilidad de las soluciones

Si bien las soluciones posibles pueden no resultar en el orden de construcción esperado, el algoritmo no entrega soluciones infactibles, pues, tras cumplirse todas restricciones adecuadas, todas las soluciones pueden ser evaluadas según sus tiempos de juego y sus porcentajes de entidades restantes. Con esto, aún si la solución no consigue llegar al 0% de entidades restantes (cumplir con el objetivo esperado en su totalidad) o tarda un tiempo de juego considerablemente elevado para llegar a la entidad deseada, esta puede ser contrastada con otras para decidir cual de ellas es la mejor. Además, se puede replicar en el videojuego cualquier orden de construcción obtenido del algoritmo, pues en su creación se siguen las restricciones del RCPSP en todo momento.

3.1.5 Operadores

La solución posee distintos operadores que apoyan al algoritmo *Simulated Annealing*, que se puede ver en pseudo-código en el algoritmo 3.1, los cuales son detallados en las secciones a continuación.

3.1.5.1 Orden de construcción aleatorio

Una de las funciones principales (utilizada en la línea 8 del algoritmo 3.1) que permite la creación de una solución al entregarnos todas las tablas que las componen. Recibe las tablas de una solución, que pueden ser las iniciales en caso de que sea la primera solución, el tiempo de inicio de la nueva solución (denotado por el tiempo final de la solución que se le entrega) para comenzar a armar la solución 1 segundo después del indicado y el tiempo de fin que indica el tamaño (en tiempo) de la solución.

Este algoritmo consiste en actualizar todas las tablas cada segundo desde el de inicial + 1 hasta el tiempo de fin. Cada ciclo incluye la elección al azar de una de las posibles entidades a crear (que sólo se crea si cumple todos sus requisitos) y la actualización de las tablas de la solución al reducir los segundos de las entidades a la espera de ser creadas y agregar cualquiera que fuera creada durante el ciclo en los lugares correspondientes. Finalmente, el algoritmo entrega las tablas de la solución obtenida.

Algoritmo 3.2: Orden de construcción aleatorio

Input: G, t_0, t_f, E, Q_E

```
1  $G' \leftarrow G$ ;  
2 for  $t = t_0 + 1$  hasta  $t_f$  do  
    /* Actualiza la solución al segundo t con decisión aleatoria */  
3      $G' \leftarrow avanzar\_segundo(t, G', E, Q_E)$ ;  
4 end  
Output:  $G'$ 
```

El pseudocódigo 3.3 muestra a grandes rasgos como las tablas que componen una solución se actualizan 1 segundo, destacando principalmente la elección de la entidad a construir en donde se reduce un poco la lista de posibles candidatos al revisar la disponibilidad de estos, para luego escoger uno entre ellos y luego terminar de revisar si es posible construido con los recursos existentes en el tiempo determinado.

Algoritmo 3.3: Avanzar segundo

Input: G, t, E, Q_E

```
/* Edificios y trabajadores se actualizan 1 segundo */
1  $G' \leftarrow G$ ;
2  $G'_{Edificios}, G'_{Trabajadores}, G'_{Entidades} \leftarrow actualizar\_edificios(G', t)$ ;
3  $G'_{Trabajadores}, G'_{Edificios}, G'_{Entidades} \leftarrow actualizar\_trabajadores(G', t)$ ;
/* Recursos se actualizan 1 segundo */
4  $G'_{Recursos} \leftarrow actualizar\_recursos(G', t)$ ;
/* Se filtran entidades que cumplen sus requisitos */
5  $Entidades \leftarrow entidades\_desbloqueadas(G', Arbol\_Tecnologias)$ ;
/* Se filtran entidades cuyas entidades de construcción están disponibles */
6  $Entidades \leftarrow entidades\_disponibles(G', Entidades, Arbol\_Tecnologias)$ ;
/* Se escoge una entidad al azar de la lista Entidades */
7  $entidad \leftarrow entidad\_aleatoria(Entidades)$ ;
/* Se revisa si existen los recursos necesarios para construir */
8 if  $recursos\_disponibles(entidad, G', Arbol\_Tecnologias)$  then
9 |  $G'_{Edificios}, G'_{Trabajadores}, G'_{Acciones} \leftarrow agregar\_accion(G', entidad, t)$ ;
10 end
```

Output: G'

En cuanto a la eficiencia algorítmica de los 2 algoritmos indicados en esta sección, tenemos que el algoritmo 3.3 tiene un orden $O(n)$ en el peor caso donde las tablas de edificios (en línea 2) o de trabajadores (en línea 3) tiendan a una cantidad muy alta de este tipo de entidades, pero cómo se buscan soluciones con un tiempo cercano a 0, no es común que las tablas tengan un largo mayor que 50.

Por otra parte, el algoritmo 3.2 tiene un orden $O(n^2)$ si consideramos como n al ciclo que inicia en la línea 2, aunque este tiene como máximo un valor de 132.

3.1.5.2 Función de evaluación

Es el operador que entrega un puntaje a cada solución (utilizado en las líneas 1 y 9 del algoritmo 3.1) o, en otras palabras, es la función objetivo que se utiliza en *Simulated Annealing* para comparar las generaciones dentro de los ciclos.

El cálculo de la función de evaluación se realiza en cada solución del algoritmo para chequear si la nueva solución G' es mejor o peor que la solución actual G_{SA} o que la mejor solución G_{Best} y posee dos criterios principales: el porcentaje restante de entidades y el tiempo de juego del orden de construcción.

El porcentaje restante de las entidades se calcula según la entidad objetivo E y la cantidad de la entidad objetivo Q_E que se indiquen al operador. En esencia, el porcentaje restante de las entidades mide cuántas entidades faltan para llegar a la entidad objetivo al usar la cantidad de entidades requisitos Q_R de la entidad objetivo E (con los requisitos de los requisitos siendo parte de este total), teniendo en cuenta que la cantidad Q_R completada puede ser fracción en caso de que alguna entidad aún esté en la cola de creación, junto a la cantidad de la entidad Q_E para obtener el total necesario $Q_R + Q_E$. Con la finalidad de normalizar este valor para que su resultado sea entre 0 y 1 es que se obtiene $P_E = \frac{Q_{Ri} + Q_{Ei}}{Q_R + Q_E}$, con Q_{Ri} y Q_{Ei} como la diferencia entre las entidades actuales en el orden de construcción y las entidades totales requeridas.

De forma más directa que el anterior, el puntaje del tiempo P_t originalmente se pensó normalizar para que el valor estuviera entre 0 y 1 y luego ponderarlo con el puntaje de las entidades, pero dadas las características de SA, en dónde se compara una solución directamente con otra, es que este acercamiento sólo permitía la comparación a nivel localizado entre una solución G y su vecino G' , pero al utilizar el máximo y mínimo de todos los tiempos obtenidos en la ejecución del algoritmo es que, al graficar estos datos, el puntaje de la solución G_{Best} que, en teoría, sólo debe disminuir presentaba aumentos en muchos de los gráficos de los casos de prueba evaluados. Ante esto, se opta por ponderar sólo P_E para multiplicarlo por una constante y utilizar el tiempo de juego en bruto como P_t , que es el tiempo de juego del orden de construcción sumado a 1 entero debido a un desfase de 1 segundo que ocurre en la implementación de la tabla de acciones (la fila 0 son los valores por defecto, mientras que la 1 representa el inicio de la partida).

Con lo anterior, una vez calculados ambos puntajes P_t y P_E es que el puntaje de las entidades restantes P_E pasa por una función $f(P_E) = P_E * k$, siendo k una constante que permite comparar el valor de las entidades restantes con el del tiempo de juego total de la solución. Esta constante k toma el valor 1500, puesto que al compararlo con el tiempo de una partida promedio de aproximadamente 13 minutos (figq, 2012) se tiene que la constante permite que las entidades restantes ponderen un valor mayor al del tiempo de una partida promedio con la finalidad de que el tiempo de juego no se lleve todo el peso de la función objetivo. Finalmente se consigue una ponderación $P = P_t + f(P_E)$.

Usando como ejemplo la evaluación de una solución que busca obtener un orden de construcción en el que se obtengan 2 *Marines* donde se necesitan 4 entidades (contando los dos *Marines*) para cumplir el objetivo, pero que sólo alcanza a construir un *Marine* en un tiempo de juego total de 134 segundos. El puntaje del tiempo P_t será igual a 135, mientras que el de las entidades restantes será de 375 (1 entidad restante dividido en las 4 entidades necesarias y multiplicado por la constante 1500) para, finalmente, conseguir una evaluación de 510. Diremos entonces que esta solución es mejor que cualquiera cuya evaluación resulte en un total mayor a

3.1.5.3 Cortar solución

Operador encargado de recrear las tablas de una solución para un segundo de tiempo de juego menor al tiempo de juego total de esa solución (utilizado en la líneas 6 del algoritmo 3.1) debido a que algunas de las tablas guardan los datos en tiempo real y no se puede acceder con facilidad a sus datos en un momento previo. Este operador recibe el segundo en que se desea recrear la solución y las tablas de la solución.

Este operador inicia "cortando" la tabla de recursos en el tiempo indicado y la tabla de acciones para que incluya las acciones que ocurren durante o previas a ese tiempo especificado. La siguiente es la tabla de entidades pues puede ser recreada a partir de la tabla de acciones junto a las entidades iniciales de cada partida. Ya con estas tablas, sólo resta volver a armar las tablas de trabajadores y edificios al llenarlas con la tabla de entidades y poblar sus columnas de espacio y utilización con la ayuda de la tabla de acciones al obtener la diferencia de tiempo de juego desde su marca de tiempo hasta el tiempo indicado.

Algoritmo 3.4: Cortar solución

Input: G, t

```

/* Se obtienen las tablas de recursos y acciones dentro del tiempo t */
1  $G'_{Recursos} \leftarrow G_{Recursos}[0 : t];$ 
2  $G'_{Acciones} \leftarrow G_{Acciones}[Timestamp = < t];$ 
/* Se crea desde cero la tabla entidades a partir de la nueva tabla acciones
   y sus tiempos de construcción */
3  $G'_{Entidades} \leftarrow recreate\_entities(G'_{Acciones});$ 
/* Se crean desde cero las tablas de trabajadores y edificios a partir de la
   nueva tabla entidades */
4  $G'_{Trabajadores} \leftarrow recreate\_workers(G'_{Entidades});$ 
5  $G'_{Edificios} \leftarrow recreate\_buildings(G'_{Entidades});$ 
/* Se llenan los datos de las tablas de trabajadores y edificios a partir de
   la nueva tabla acciones */
6  $G'_{Trabajadores} \leftarrow fill\_workers(G'_{Entidades});$ 
7  $G'_{Edificios} \leftarrow fill\_buildings(G'_{Entidades});$ 
Output:  $G'$ 

```

La eficiencia algorítmica del algoritmo 3.4 tiene un orden $O(n)$ debido al tamaño de la tabla de acciones, la cual puede aumentar considerablemente dependiendo de t y afectar a las líneas 4 y 5.

3.2 DISEÑO EXPERIMENTAL

Para conseguir los objetivos propuestos es necesario realizar diversos experimentos cuyos resultados permitan llegar a una conclusión respecto al algoritmo SA y su implementación con el problema de los órdenes de construcción, por lo que a continuación se detallan los diversos experimentos llevados a cabo en este documento.

3.2.1 Parametrización

Utilizando el paquete de R *i-race* para la parametrización (López-Ibáñez, y otros, 2016), se le entregan los rangos de los ciclos internos, la temperatura, α usado para el enfriamiento y el intervalo máximo. Se tiene que para el ciclo interno los valores enteros van entre las 25 y 250 iteraciones, el rango de la temperatura oscila entre 500 y 50000, α puede ser un valor entre 0.5 y 0.99 y, finalmente, un intervalo máximo entre 30 y 180 segundos considerando el tiempo de juego *faster*. De estos valores, los que más influyen en el tiempo de ejecución son el ciclo interno y el intervalo máximo. Aunque el ciclo externo también es un valor importante a optimizar, se deja como 50 tanto para la parametrización como para la ejecución del algoritmo, debido a que el tiempo de ejecución necesario para obtener una solución se eleva demasiado al aumentar este valor según lo observado en una etapa previa de pruebas del funcionamiento del algoritmo.

Para el presupuesto de la parametrización se utiliza un valor de 180, el cual define también la cantidad de ejecuciones que realizará *i-race* antes de entregar su respuesta.

3.2.2 Evaluación

Para poder llegar a una conclusión respecto a la hipótesis de la sección 1.5.1 y cumplir los objetivos planteados es que en esta sección se detallan los experimentos a realizar con el algoritmo SA ajustado al problema de los órdenes de construcción. Cada experimento consta de 11 ejecuciones del algoritmo SA para 3 objetivos distintos (1 *Marine*, 2 *Hellion* y 1 *Starport*) dando un total de 33 ejecuciones de la metaheurística.

3.2.2.1 Convergencia del algoritmo

Para probar que las generaciones dentro del algoritmo tienden a optimizar el tiempo de juego para una entidad en específico es que de las ejecuciones indicadas se revisará su convergencia. Esto, con el fin de verificar que las soluciones se acerquen al óptimo (disminuyen el valor entregado por la función de evaluación) en lugar de acercarse. Para este fin, se compararán los puntajes de la solución inicial, la mediana de las soluciones y la mejor solución de cada ejecución.

3.2.2.2 Comparación con algoritmo goloso

Como una metaheurística requiere ser comparada con otros acercamientos es que se utiliza un algoritmo goloso o *greedy* (Young, 2009) para contrastar sus resultados con los de *Simulated Annealing*. Similar a los experimentos con SA, se realizan 11 ejecuciones del algoritmo *greedy* para cada uno de los 3 objetivos, pero con una mayor cantidad de iteraciones.

Se utilizará la prueba no paramétrica de los rangos con signo de Wilcoxon que permite aceptar o rechazar la hipótesis de que "existe diferencia entre ambos algoritmos".

3.2.2.3 Órdenes de construcción de profesionales

Utilizando el sitio web *Spawning Tool* (Leung Paskert, 2013) en donde usuarios pueden publicar sus órdenes de construcción, se buscan órdenes de construcción que incluyan distintas entidades con el objetivo de contrastarlos con las soluciones del algoritmo. Las entidades objetivo de los órdenes de construcción encontrados, siendo gran parte de estos recopilados de partidas de jugadores profesionales en torneos oficiales (obtenidos desde el sitio mencionado), se muestran en la tabla 3.6. Debido a que estas soluciones de profesionales pueden ser consideradas como una especie de *gold standard* es que se comparan con la mejor de las soluciones encontradas para cada objetivo.

Tabla 3.6: Resumen de órdenes de construcción a comparar

| Experimento | Objetivo | Tiempo de juego (minutos) |
|-------------|--------------|---------------------------|
| 1 | 1 x Marine | 1:25 |
| 2 | 1 x Starport | 2:51 |
| 3 | 2 x Hellion | 3:32 |

Fuente: Elaboración Propia, 2021

3.2.2.4 Aplicación dentro del juego

Para verificar que las soluciones entregadas son, efectivamente, posibles de replicar dentro del juego es que es necesario iniciar algunas partidas con la raza Terran y seguir, dentro de lo posible, estos órdenes de construcción con las marcas de tiempo y entidades que indiquen. Gracias a que el videojuego permite guardar las partidas jugadas, además de mostrar diversas estadísticas de cada una, es que el orden de construcción se puede obtener de forma visual.

CAPÍTULO 4. RESULTADOS

Tras haber definido la solución y acoplado los datos conforme a esta, el algoritmo está listo para ser probado y comenzar con la fase de experimentación que se describe en el Capítulo actual.

4.1 PARAMETRIZACIÓN

Luego de la ejecución de *i-race*, cuyo tiempo de ejecución fue aproximadamente de 75 horas, los resultados (mostrados en la tabla 4.1) indican los mejores parámetros a utilizar en el algoritmo SA, de los cuales, tras observar la cantidad de ciclos totales requeridos para cada caso, **se escogen los parámetros de la fila con ID 9** al encontrarse en el primer lugar de la tabla. Con esto en mente, si se deseara reducir el tiempo de ejecución, se recomienda escoger los parámetros con ID 2 en la segunda fila, pues sólo agrega 42 segundos al intervalo máximo a cambio de 150 iteraciones internas menos, aunque para términos de este documento se trabaja con los parámetros de la fila con ID 9.

Tabla 4.1: Resultados de i-race

| ID | Iteraciones internas | Temperatura | Alfa | Intervalo máximo |
|----|----------------------|-------------|------|------------------|
| 9 | 250 | 20583 | 0.56 | 132 |
| 2 | 100 | 4927 | 0.51 | 174 |
| 22 | 100 | 791 | 0.53 | 144 |
| 8 | 250 | 11360 | 0.58 | 152 |

Fuente: Elaboración Propia, 2021

4.2 TIEMPOS DE EJECUCIÓN

Sólo a modo de documentación, pues existen una serie de elementos incontrolables que pueden afectarlo, es que se calcula el tiempo de ejecución de cada una de las ejecuciones del algoritmo SA. Los resultados de las 33 ejecuciones de SA entregaron un promedio de 33,0405 minutos y una mediana de 32,6806 minutos. También se tiene un tiempo de ejecución máximo de 58,0319 minutos y un mínimo de 17,8881. La variación en estos resultados depende de

los recursos que se reparten entre la ejecución del algoritmo y los otros procesos (externos a R) utilizados por el computador, pues los experimentos no fueron realizados en un computador dedicado exclusivamente a estos. Además, otra variación ocurre debido a la aleatoriedad del orden de construcción aleatorio.

4.3 EVALUACIÓN

A continuación se detallan los resultados de los experimentos propuestos en la sección 3.2.2, además de un análisis sobre cada uno de estos.

4.3.1 Convergencia del algoritmo

Tras ejecutar el algoritmo 11 veces para los 3 objetivos (1 *Marine*, 1 *Starport* y 2 *Hellion*) se obtienen para cada uno de estos, además de los órdenes de construcción, los gráficos que muestran como el puntaje de las generaciones de cada iteración oscila al tratarse de *Simulated Annealing*, pues este algoritmo tiene una probabilidad de escoger una solución peor (con un puntaje más alto), pero que siempre escoge una solución con un mejor puntaje. Esta oscilación se detiene conforme avanzan las iteraciones, pues la probabilidad de aceptación es cada vez menor. Los gráficos, escogidos de las ejecuciones al azar, muestran en una tonalidad celeste esta oscilación al indicar el puntaje P_{SA} de G_{SA} en el tiempo, mientras que en azul se observa la mejor solución P_{Best} encontrada de todas las generaciones en el tiempo.

A continuación se muestran los valores de las soluciones y los gráficos resultantes en 1 ejecución aleatoria de cada objetivo del algoritmo:

Tabla 4.2: Resultados de Simulated Annealing

| Ejecución | 1 Marine | 1 Starport | 2 Hellion |
|-----------|----------|------------|-----------|
| 1 | 84 | 180 | 220 |
| 2 | 89 | 170 | 212 |
| 3 | 84 | 195 | 216 |
| 4 | 84 | 205 | 221 |
| 5 | 84 | 210 | 228 |
| 6 | 84 | 185 | 237 |
| 7 | 90 | 191 | 220 |
| 8 | 84 | 182 | 231 |
| 9 | 89 | 177 | 225 |
| 10 | 84 | 192 | 242 |
| 11 | 85 | 207 | 207 |

Fuente: Elaboración Propia, 2021

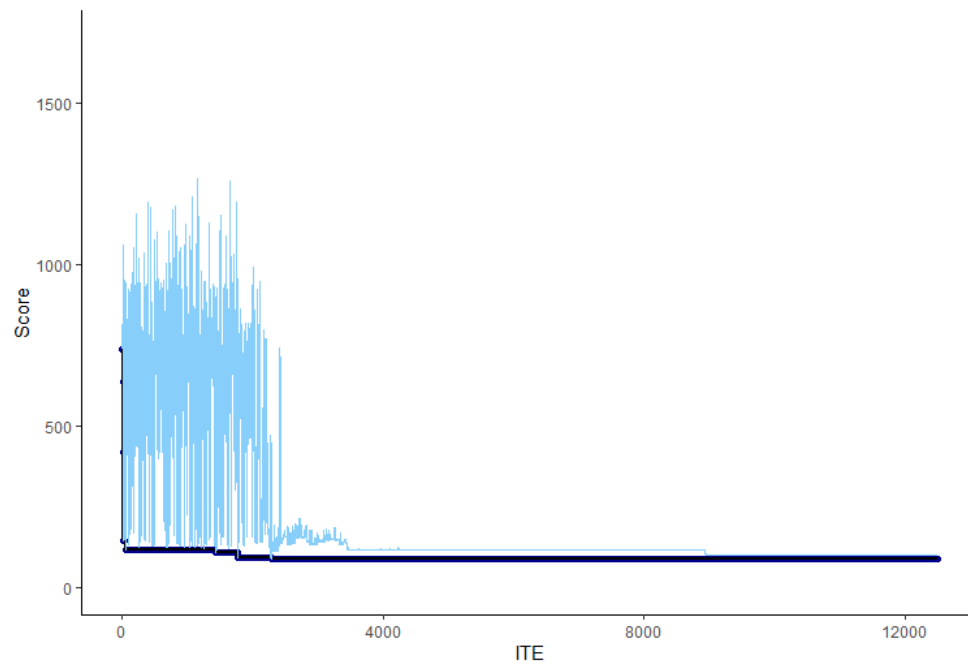


Figura 4.1: Gráfico de generaciones de SA del Experimento 1: novena ejecución para 1 Marine

Fuente: Elaboración propia, 2021

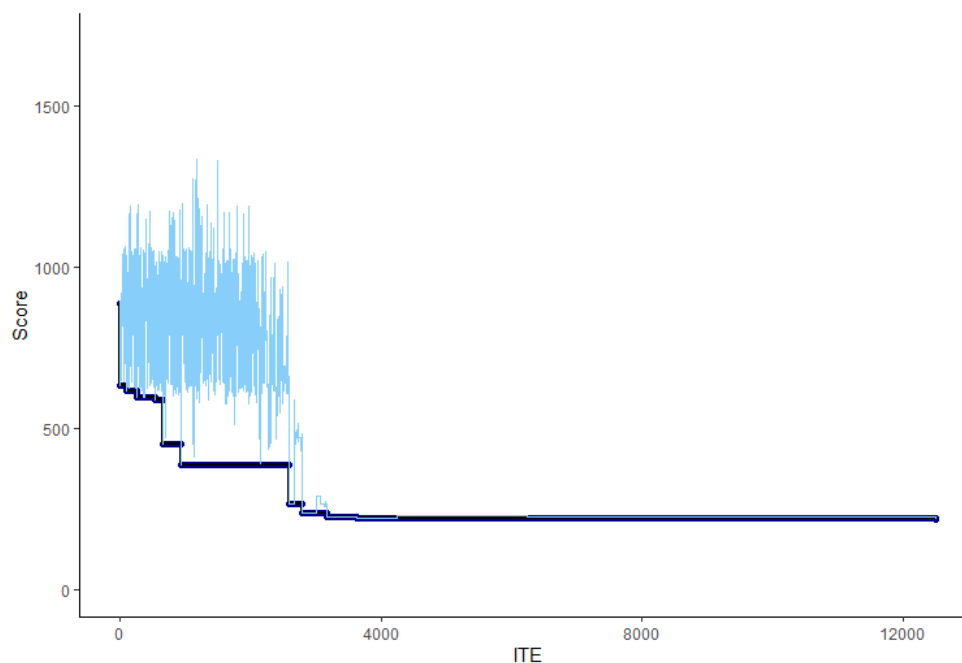


Figura 4.2: Gráfico de generaciones de SA del Experimento 1: onceava ejecución para 1 Starport

Fuente: Elaboración propia, 2021

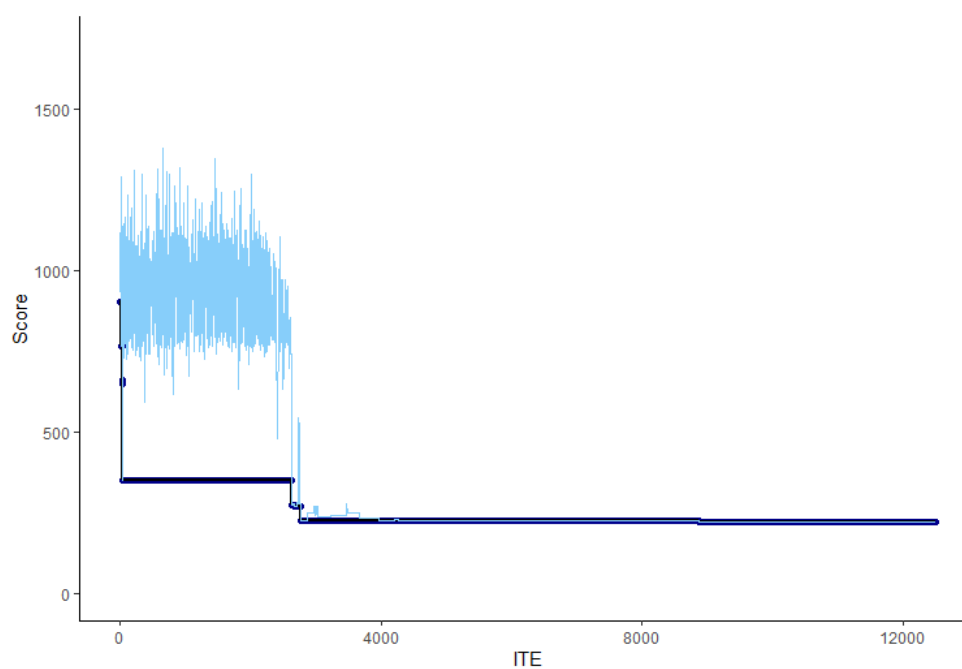


Figura 4.3: Gráfico de generaciones de SA del Experimento 1: cuarta ejecución para 2 Hellions

Fuente: Elaboración propia, 2021

En los gráficos se puede observar claramente como la mejor solución (marcada en

azul oscuro) es cada vez menor conforme las iteraciones aumentan. También es común entre las soluciones el hecho de que la mejor solución rápidamente se reduce en las primeras iteraciones, mejorando lentamente a partir de aquí hasta que alcanza el equilibrio en cada ejecución.

Tabla 4.3: Resultados de SA: mejor solución para 1 Marine

| Tiempo (en minutos) | Acción |
|---------------------|--------------|
| 0:07 | Supply Depot |
| 0:17 | SCV |
| 0:37 | Barracks |
| 0:49 | SCV |
| 0:57 | Refinery |
| 1:07 | Refinery |
| 1:15 | SCV |
| 1:23 | Marine |

Fuente: Elaboración Propia, 2021

Tabla 4.4: Resultados de SA: mejor ejecución para 1 Starport

| Tiempo (en minutos) | Acción |
|---------------------|--------------|
| 0:07 | Supply Depot |
| 0:19 | SCV |
| 0:27 | Refinery |
| 0:36 | SCV |
| 0:55 | SCV |
| 1:03 | Barracks |
| 1:14 | Refinery |
| 1:29 | SCV |
| 1:41 | SCV |
| 1:53 | Marine |
| 2:06 | Factory |
| 2:16 | Reaper |
| 2:29 | SCV |
| 2:49 | Starport |

Fuente: Elaboración Propia, 2021

Tabla 4.5: Resultados de SA: tercera ejecución para 2 Hellions

| Tiempo (en minutos) | Acción |
|---------------------|--------------------|
| 0:01 | SCV |
| 0:14 | Supply Depot |
| 0:21 | SCV |
| 0:33 | SCV |
| 0:40 | Refinery |
| 0:48 | SCV |
| 1:03 | SCV |
| 1:10 | Barracks |
| 1:21 | SCV |
| 1:30 | Refinery |
| 1:39 | SCV |
| 1:54 | SCV |
| 1:55 | Supply Depot |
| 2:02 | Reactor (Barracks) |
| 2:22 | Factory |
| 2:35 | SCV |
| 2:38 | Reaper |
| 2:45 | Marine |
| 2:52 | SCV |
| 3:05 | Hellion |
| 3:12 | Marine |
| 3:15 | Reaper |
| 3:26 | Hellion |

Fuente: Elaboración Propia, 2021

En las tablas 4.3, 4.4 y 4.5 se observan las mejores soluciones encontradas en las ejecuciones de SA.

Para mostrar de mejor manera como el algoritmo converge a 0 es que se muestran a continuación algunas de las soluciones de 9 de las 33 ejecuciones de SA de forma aleatoria, específicamente la solución inicial, la solución de la iteración 1000 la mediana de las mejores soluciones y las mejores soluciones de cada ejecución.

Tabla 4.6: Convergencia de la evaluación de las soluciones

| Objetivo | Ejecución | Solución Inicial | Iteración 1000 | Mediana | Mejor Solución |
|------------|-----------|------------------|----------------|---------|----------------|
| 1 Marine | 1 | 638 | 84 | 84 | 84 |
| 1 Marine | 6 | 724 | 91 | 84 | 84 |
| 1 Marine | 7 | 658 | 90 | 90 | 90 |
| 1 Starport | 2 | 937.05 | 410 | 192 | 170 |
| 1 Starport | 5 | 858 | 437 | 217 | 205 |
| 1 Starport | 9 | 820 | 413 | 227 | 227 |
| 2 Hellion | 2 | 768.86 | 589.57 | 249 | 249 |
| 2 Hellion | 4 | 901.14 | 349 | 227 | 221 |
| 2 Hellion | 9 | 789.86 | 336 | 226 | 225 |

Fuente: Elaboración Propia, 2021

Como se puede observar en la tabla 4.6, de las 3 ejecuciones escogidas aleatoriamente para cada experimento, todas disminuyen o mantienen su valor, lo cual reafirma lo visto en los gráficos de las figuras 4.1, 4.2 y 4.3 en donde se representa visualmente la convergencia de la mejor solución.

4.3.2 Comparación con algoritmo goloso

Tras realizar las 33 ejecuciones con el algoritmo goloso se obtuvieron los siguientes valores para los órdenes de construcción obtenidos, separados por objetivo.

Tabla 4.7: Resultados del algoritmo goloso

| Ejecución | 1 Marine | 1 Starport | 2 Hellion |
|-----------|----------|------------|-----------|
| 1 | 85 | 189 | 238 |
| 2 | 107 | 189 | 221 |
| 3 | 121 | 214 | 229 |
| 4 | 89 | 402 | 254 |
| 5 | 90 | 232 | 271 |
| 6 | 90 | 198 | 215 |
| 7 | 84 | 188 | 215 |
| 8 | 90 | 192 | 232 |
| 9 | 89 | 221 | 731 |
| 10 | 89 | 204 | 257 |
| 11 | 89 | 198 | 263 |

Fuente: Elaboración Propia, 2021

Con estos resultados, acompañados de los resultados de SA en la tabla 4.2 se puede

realizar la prueba de Wilcoxon de forma sencilla en el lenguaje R, con el cual se obtienen los siguientes resultados:

Tabla 4.8: Resultados de Wilcoxon entre SA y algoritmo goloso

| Experimento | P-value |
|-------------|----------|
| Marine | 0.007048 |
| Starport | 0.04508 |
| Hellion | 0.04502 |

Fuente: Elaboración Propia, 2021

De esto se desprende que, debido a que el p -value es menor que 0.05, es posible declarar que existe una diferencia entre los algoritmos SA y goloso al rechazar la hipótesis nula indicada en la sección 3.2.2.2. Sumado a esto, también se puede saber información de los resultados en cuanto a su distribución, la cual puede visualizarse en los siguientes diagramas de caja:



Figura 4.4: Diagrama de caja Marine

Fuente: Elaboración propia, 2021

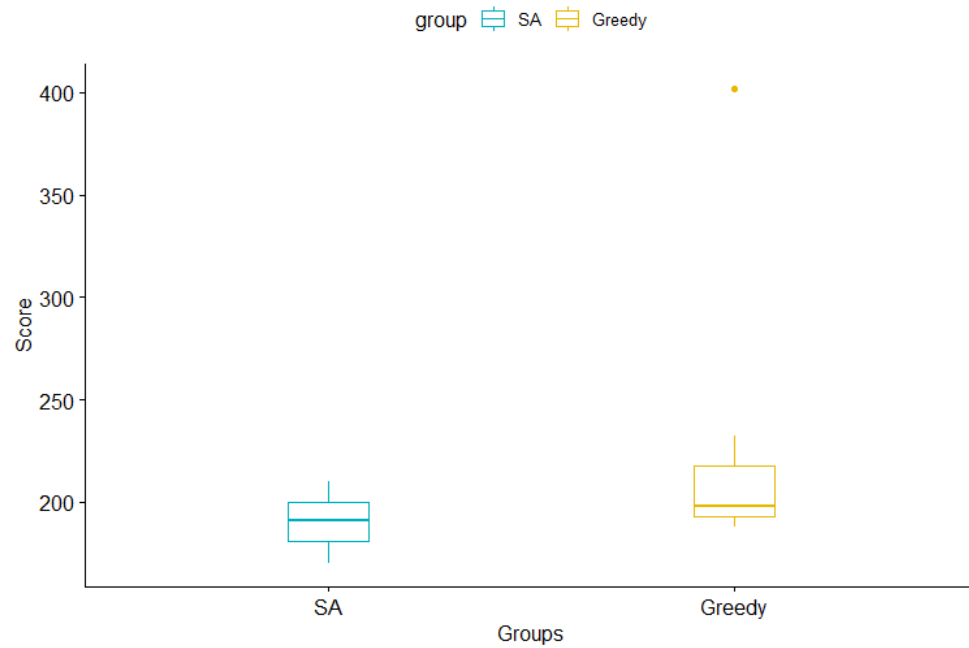


Figura 4.5: Diagrama de caja Starport

Fuente: Elaboración propia, 2021

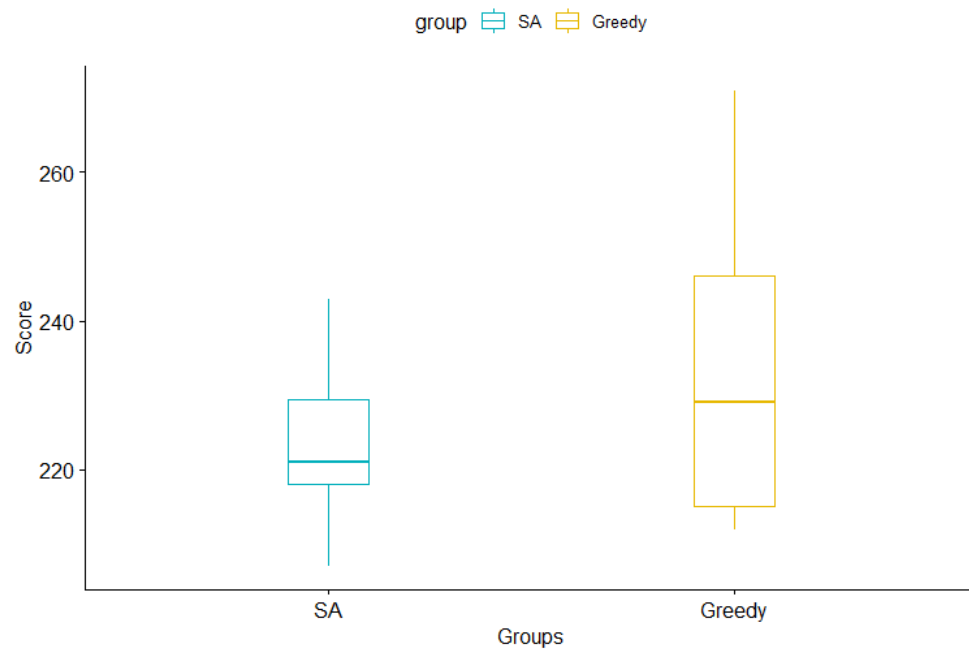


Figura 4.6: Diagrama de caja Hellion

Fuente: Elaboración propia, 2021

Tabla 4.9: Resumen de los algoritmos

| Experimento | Mínimo | 1 ^{er} Cuartil | Mediana | Promedio | 3 ^{er} Cuartil | Máximo |
|-----------------|--------|-------------------------|---------|----------|-------------------------|--------|
| Marine SA | 84 | 84 | 85 | 85.55 | 87 | 90 |
| Marine Greedy | 84 | 89 | 89 | 93 | 90 | 121 |
| Starport SA | 170 | 181 | 191 | 190.4 | 200 | 210 |
| Starport Greedy | 188 | 193 | 198 | 221.1 | 217.5 | 402 |
| Hellion SA | 207 | 218 | 221 | 223.5 | 229.5 | 242 |
| Hellion Greedy | 215 | 225 | 238 | 284 | 260 | 731 |

Fuente: Elaboración Propia, 2021

La tabla 4.9 resume los valores de las figuras 4.4, 4.5 y 4.6, los cuales dejan ver que en todos los aspectos las soluciones obtenidas por SA son mejores que aquellas con el algoritmo goloso, pues los valores del puntaje son más cercanos a cero en la metaheurística.

4.3.3 Órdenes de construcción de profesionales

Como se indica en la sección 3.2.2.3, se compararon los mejores órdenes de construcción obtenidos con algunos órdenes de construcción de profesionales en torneos oficiales, recopilados por la comunidad de StarCraft II. A continuación se pueden observar las tablas que contrastan los órdenes de construcción entregados por SA con los órdenes de construcción de profesionales, los cuales muestran los datos más relevantes a la hora de compararlos.

Tabla 4.10: Comparación para 1 Marine

| Orden de construcción | Tiempo (en minutos) | Entidades | SCV |
|-----------------------|---------------------|-----------|-----|
| Profesional | 1:25 | 4 | 7 |
| SA | 1:23 | 5 | 3 |

Fuente: Spawning Tool, 2021

Si se confronta el mejor orden de construcción obtenido (aquel más con menor puntaje y con más unidades creadas en caso de un empate) con el orden de construcción encontrado más rápido para llegar a un *Marine*, se observa que tanto la cantidad de entidades como el tiempo es mejor en el orden de construcción obtenido por el algoritmo, pues acorta 2 segundos de la solución que propone el jugador y crea un *Refinery* extra que ayuda a mejorar la cantidad de gas vespeno a recolectar, aunque la cantidad de trabajadores (SCV) es mayor en el orden de construcción profesional.

Tabla 4.11: Comparación para 1 Starport

| Orden de construcción | Tiempo (en minutos) | Entidades | SCV |
|-----------------------|---------------------|-----------|-----|
| Profesional | 2:51 | 12 | 10 |
| SA | 1:49 | 8 | 6 |

Fuente: Spawning Tool, 2021

La tabla 4.12 muestra la comparación entre el orden de construcción obtenido de SA con el orden de construcción de un jugador de alto rendimiento. En esta ocasión el orden de construcción de SA presenta una diferencia de 2 segundos a favor para llegar al *Starport*, pero se obtienen menos entidades que en el orden de construcción de un jugador en un torneo oficial. Con esto en mente, el orden de construcción de SA logra mejorar el tiempo del orden de construcción profesional, pero a cambio de un mejor ejército.

Tabla 4.12: Comparación para 2 Hellion

| Orden de construcción | Tiempo (en minutos) | Entidades | SCV |
|-----------------------|---------------------|-----------|-----|
| Profesional | 3:32 | 16 | 13 |
| SA | 3:26 | 13 | 10 |

Fuente: Spawning Tool, 2021

En este caso se tiene una solución algo más compleja debido a los requisitos totales de un Hellion, lo que queda en evidencia al observar ambas soluciones, pues la cantidad de entidades y el tiempo total aumenta en comparación a los experimentos anteriores. En este caso, aunque la cantidad de entidades es mayor en el orden de construcción del profesional, se puede observar como el tiempo es mejor por 5 segundos en la solución de SA y, además, incluye un *Reaper* extra, lo cual significa un mejor ejército en término de poder de ataque.

Con las comparaciones anteriores queda en evidencia que el algoritmo está a la par con los órdenes de construcción usados por profesionales y jugadores de alto rango, consiguiendo incluso acortar algunos segundos para algunas de estas soluciones y, en algunos casos, obtener un mejor ejército si lo único que se busca es atacar al oponente.

4.3.4 Aplicación dentro del juego

Tras probar en el videojuego los órdenes de construcción descritos en las secciones anteriores, se deduce que si es posible llevarlos a cabo, aunque se requiere de un buen manejo de los controles de StarCraft II, pues no es sencillo, sin antes practicar o realizar repetidos

intentos, seguir las secuencias teniendo menos de un segundo para realizar la acción en el tiempo específico.

A continuación se muestran capturas de pantalla del juego con los órdenes de construcción resultantes para 1 *Marine* y 2 *Hellions*, los cuales se acercan lo más posible a los órdenes de construcción, pues igualarlos a la perfección puede resultar en varias horas de intentos, aunque queda en evidencia que los tiempos se cumplen y son posibles en el juego.




| Time | Action | Supply |
|------|-------------------------------------------------------------------------------------------------------|---------|
| 0:09 |  Supply Depot | 12 / 15 |
| 0:15 |  SCV | 13 / 15 |
| 0:37 |  Barracks | 13 / 23 |
| 0:49 |  SCV | 14 / 23 |
| 1:04 |  SCV | 15 / 23 |
| 1:08 |  Refinery | 15 / 23 |
| 1:18 |  Refinery | 15 / 23 |
| 1:24 |  Marine | 16 / 23 |

Figura 4.7: Orden de construcción para un Marine dentro del juego

Fuente: Elaboración propia, 2021

| Time | Action | Supply |
|------|--------------------------------------------------------------------------------------------------|---------|
| 0:01 |  SCV | 13 / 15 |
| 0:14 |  Supply Depot | 13 / 15 |
| 0:23 |  SCV | 14 / 15 |
| 0:34 |  Refinery | 14 / 15 |
| 0:40 |  SCV | 15 / 23 |
| 0:52 |  SCV | 16 / 23 |
| 0:59 |  Refinery | 16 / 23 |
| 1:06 |  SCV | 17 / 23 |
| 1:20 |  SCV | 18 / 23 |
| 1:32 |  Barracks | 18 / 23 |
| 1:39 |  SCV | 19 / 23 |
| 1:53 |  SCV | 20 / 23 |
| 2:03 |  Supply Depot | 20 / 23 |
| 2:15 |  SCV | 21 / 23 |
| 2:19 |  Reaper | 22 / 23 |
| 2:39 |  Factory | 22 / 31 |
| 2:45 |  SCV | 23 / 31 |

Figura 4.8: Parte de un orden de construcción para dos Hellion dentro del juego

Fuente: Elaboración propia, 2021

CAPÍTULO 5. CONCLUSIONES

En el presente documento se explica cómo el videojuego StarCraft II tiene diversos trabajos científicos, con la mayoría apuntando a mejorar inteligencias artificiales cuyos propósitos son principalmente jugarlo para ganar, pero dejando a un lado a los jugadores reales sin ofrecer soluciones para ayudarles a mejorar. Por esto es que se plantea un modelo que busca facilitar la búsqueda de los órdenes de construcción al utilizar un acercamiento metaheurístico que, tras representar los elementos necesarios del juego, logra ofrecer soluciones a la par con las que usan los jugadores profesionales de más alto nivel, demostrado por medio de experimentos.

En cuanto a los resultados obtenidos de los experimentos realizados en la sección 4, se logran optimizar los parámetros de entrada utilizados en el algoritmo SA. También se demuestra que las soluciones entregadas son realizables dentro del videojuego, lo cual refleja que las soluciones siguen las restricciones impuestas en este RCPSP. También se desprende que las soluciones entregadas están al nivel de las soluciones de profesionales, pues las diferencias entre los tiempos de juego de los órdenes de construcción ofrecidos benefician a los entregados por el algoritmo SA. Con esto en cuenta, el modelo aún tiene espacio para cambios que permitan mejoras en las soluciones como la creación de múltiples entidades en un mismo segundo.

La pregunta de hipótesis (declarada en la sección 1.5.1) habla de conseguir buenos órdenes de construcción, que fueron definidos como aquellos a la par o que superen los de jugadores profesionales o de alto rango clasificatorio, para la raza Terran utilizando SA. Aquí, los experimentos realizados junto a las comparaciones con órdenes de construcción de jugadores profesionales y de alto rango sugieren que los obtenidos a través del algoritmo implementado si pueden ser catalogados como buenos órdenes de construcción puesto que, en algunos casos, las soluciones entregadas por SA presentan claras ventajas, aunque sea en un par de segundos, sobre los órdenes de construcción con los que fueron contrastados. Además se tiene que al contrastar los resultados de *Simulated Annealing* con los de un acercamiento goloso, los de SA obtienen mejores resultados en general según lo detallado en la sección 3.2.2.2.

Por otro lado, las soluciones que entrega el algoritmo son ejecutadas en un tiempo razonable si se contrastan con las horas que un jugador puede tardar para llegar a la misma respuesta o si, computacionalmente, se intentase obtener todas las posibles soluciones. Si bien el tiempo de ejecución para llegar a una solución no es útil para usarlo en tiempo real, debido a que el orden de construcción es necesario al inicio de la partida y la solución tarda en promedio 33,0405 minutos, este puede ser modificado para conseguir tiempos de ejecución menores al costo de disminuir la calidad de las soluciones. Además, si bien los resultados no son inmediatos, el objetivo de estos es ayudar al jugador a preparar su para un futuro encuentro en lugar de indicarle como debe jugar en una partida actual.

También se puede decir que las soluciones consiguen llegar a la unidad requerida y en las cantidades establecidas, lo cual demuestra el último punto de la pregunta problema. Con esto, se permite afirmar que el modelo propuesto si puede encontrar órdenes de construcción de forma más eficiente que un orden de construcción aleatorio y en un tiempo razonable, el cual permite conseguir una unidad y una cantidad de esta determinadas, siempre que sea de la raza Terran en StarCraft II.

Los objetivos mencionados en la sección 1.4 fueron completados del todo, pues los órdenes de construcción obtenidos del modelo desarrollado si cumple con las características que se buscan de este en cuanto a la optimización del tiempo requerido en los órdenes de construcción, además de que el algoritmo se hace bajo un enfoque metaheurístico con el algoritmo *Simulated Annealing*.

En cuanto a los 4 objetivos específicos, se puede observar a lo largo del documento que estos fueron completados, pues tanto el objetivo 1 como el 2 y el 3 se presentan en el capítulo 3 que detalla el algoritmo propuesto y los experimentos a realizar sobre este. El objetivo 3 también se prueba junto al objetivo 4 en el capítulo 4 con los resultados de los experimentos planificados.

5.1 TRABAJO A FUTURO

En vista de que las soluciones que ofrece el modelo tienen paso a cambios que permitan aumentar la eficiencia de las soluciones, existen algunos cambios que pueden mejorar la calidad de estas, como modificar el algoritmo para permitir más acciones por segundo (actualmente sólo permite una acción por segundo) o mejorar el cálculo de los recursos obtenidos por los trabajadores, ya que actualmente se utiliza un valor constante sin considerar la saturación que varía con la cantidad de trabajadores y tampoco considera el límite que tiene cada mineral o géiser de gas para ofrecer. Además de esto, se tiene la posibilidad de asignar un puntaje a cada entidad para que estas tengan un peso en la solución final, con el fin de escoger no sólo la solución con el tiempo más corto, sino que la solución que entregue un mayor beneficio al jugador en el menor tiempo posible, lo cual podría ayudar a acercarse o superar las soluciones usadas por profesionales que no sólo buscan construir una entidad en específico, sino que intentan conseguir un ejército capaz de superar al de su oponente.

También queda pendiente para el modelo la estimación del impacto que tendría reducir la cantidad de iteraciones del algoritmo, el cual afecta directamente el tiempo de ejecución. Esto, con la finalidad de conseguir soluciones en menos tiempo, intentando no sacrificar la calidad de estas soluciones.

Otro trabajo pendiente yace en las comparaciones, que si es necesario contrastarlas

con otros órdenes de construcción ya existentes, no es necesario que estos sean de otros jugadores, pues también se pueden contrastar con soluciones entregadas con otras metaheurísticas para así analizar cual de ellas entrega mejores soluciones para este problema.

Si bien este trabajo se enfoca en la raza Terran de StarCraft II, el modelo puede ser expandido a las otras razas tras modelar sus características particulares que las diferencian de los Terran. Además, teniendo en cuenta el tipo de problema, el modelo puede aplicar no sólo a StarCraft II, sino que a cualquier videojuego de estrategia en tiempo real si se tienen en cuenta los elementos en común con este modelo enfocado en el RCPSP.

Para finalizar, se espera que tras incluir los puntos mencionados en esta sección, se pueda publicar ese nuevo documento a modo de aportar a la comunidad científica y a la comunidad de StarCraft.

GLOSARIO

- **Árbol de tecnologías:** elemento del juego StarCraft II que muestra las unidades y estructuras necesarias para crear cada unidad y estructura de una raza. Se le atribuyen también las características de coste de tiempo de cada unidad y estructura.
- **Benchmark:** literalmente punto de referencia. Llámese de algo que es utilizado como punto comparativo.
- **Esports:** competiciones de videojuegos que conllevan eventos de gran escala.
- **Estructura:** uno de los elementos del juego StarCraft II que consiste de edificaciones o construcciones, cuyas principales funciones son las de creación de unidades e investigación de tecnologías.
- **Heurística:** se califica de heurístico a un procedimiento para el que se tiene un alto grado de confianza y que encuentra soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación.
- **Metaheurística:** estrategias para diseñar y/o mejorar los procedimientos heurísticos orientados a obtener un alto rendimiento.
- **NP-Hard:** de las categorías de complejidad computacional, son aquellos problemas cuya resolución resulta más difícil que el problema más difícil de la categoría NP o Nondeterministic polynomial time.
- **Orden de construcción:** secuencia de acciones que indica que unidad o estructura debe ser creada en un punto de tiempo determinado, comenzando desde que inicia la partida, para lograr conseguir una unidad o estructura específica. Va de la mano con la estrategia que se desee utilizar.
- **Protoss:** una de las tres razas jugables del juego StarCraft II que posee unidades, estructuras y características propias en comparación al resto. Puede describirse como una raza robótica avanzada.
- **Ranking:** en el contexto de StarCraft II, una especie de tabla de posiciones que asigna rangos a los jugadores según su habilidad en partidas en línea clasificatorias.
- **Recursos:** uno de los elementos del juego StarCraft II que se representa por números enteros. Existen dos tipos de recursos: gas vespeno y minerales, los cuales deben ser extraídos desde el escenario para que vayan aumentando.
- **Theorycrafting:** análisis matemático de las mecánicas de un juego con el fin de encontrar estrategias o tácticas.
- **Terran:** una de las tres razas jugables del juego StarCraft II que posee unidades, estructuras y características propias en comparación al resto. Puede describirse como una raza humana con tecnologías futuristas.
- **Unidad:** uno de los elementos del juego StarCraft II que consiste de un miembro del "ejército" de cada jugador y que puede movilizarse por el escenario. Aunque existen varias funciones dentro de las distintas unidades, su función principal es la de atacar.
- **Zerg:** una de las tres razas jugables del juego StarCraft II que posee unidades, estructuras y características propias en comparación al resto. Puede describirse como una raza de criaturas con forma de insectos.

REFERENCIAS BIBLIOGRÁFICAS

- Aloupis, G., Demaine, E. D., Guo, A., & Viglietta, G. (2012). Classic nintendo games are (computationally) hard. (p. 36).
URL <https://arxiv.org/abs/1203.1895>
- Artigues, C., Demasse, S., & Néron, E. (2008). Resource-constrained project scheduling: Models, algorithms, extensions and applications. (pp. 21–25).
- Entertainment, B. (????).
- Favis, E. (2020). Blizzard cuts esports deal with esl, dreamhack around starcraft ii, warcraft iii: Reforged.
URL <https://www.washingtonpost.com/video-games/2020/01/07/esl-dreamhack-announce-esports-deal-with-blizzard-around-starcraft-ii-world-warcraft-iii-reforged/>
- figq (2012). [sc2] avg game length by race and matchup.
URL <https://t1.net/blogs/304522-sc2-avg-game-length-by-race-and-matchup>
- Glass, B. D., Maddox, W. T., & Love, B. C. (2013). Real-time strategy game training: Emergence of a cognitive flexibility trait.
URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3737212/>
- Justesen, N., & Risi, S. (2017). Continual online evolutionary planning for in-game build order adaptation in starcraft.
URL <https://www-scopus-com.ezproxy.usach.cl/record/display.uri?eid=2-s2.0-85026384177&origin=resultslist&sort=plf-f&src=s&st1=Continual+online+evolutionary+planning+for+in-game+build+order+adaptation+in+starcraft&st2=&sid=8c847032da5b9cd1a2c5e3b88e168410&sot=>
- Kochetov, Y., & Stolyar, A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem.
URL <http://ns.math.nsc.ru/LBRT/k5/Kochetov/Papers/Kochetov-Ufa-03.pdf>
- Kow, Y., & Young, T. (2013). Media technologies and learning in the starcraft esports community.
URL <https://dl.acm.org/doi/10.1145/2441776.2441821>
- Kuo, I. (2012). Schools are using starcraft 2 as serious education tools.
URL <https://www.gamification.co/2012/12/04/schools-using-starcraft-2-as-education-tools/>
- Köstler, H., & Gmeiner, B. (2013). A multi-objective genetic algorithm for build order optimization in starcraft ii.
URL <https://link-springer-com.ezproxy.usach.cl/article/10.1007/s13218-013-0263-2>
- Leung, K., & Paskert, D. (2013). Organizing starcraft 2 build orders, guides, and replays.
URL <https://lotv.spawningtool.com/>
- López-Ibáñez, M., Dubois-Lacoste, J., Pérez, L., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration.
URL <https://www.sciencedirect.com/science/article/pii/S2214716015300270>
- Nai-Hsin, P., Tsai-Ching, H., & Wei-Tong, C. (2019). Using hybrid simulated annealing algorithm in resource constrained project scheduling problem.
URL <https://www.ijoi-online.org/index.php/current-issue/191-using-hybrid-simulated-annealing-algorithm-in-resource-constrained-project-scheduling-problem>

- Park, H., Lee, K., Cho, H.-C., & Kim, K.-J. (2012). Prediction of early stage opponents strategy for starcraft ai using scouting and machine learning.
URL <https://dl.acm.org/doi/10.1145/2425296.2425298>
- PiousFlea (2010). Scientifically measuring mining speed.
URL <https://tl.net/forum/sc2-strategy/140055-scientifically-measuring-mining-speed>
- Ramos, A. S., Olivares-Benitez, E., & Miranda-Gonzalez, P. A. (2021). Multi-start iterated local search metaheuristic for the multi-mode resource-constrained project scheduling problem.
URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12830>
- Suarez, O. (2014). Una aproximación a la heurística y metaheurísticas.
URL <https://core.ac.uk/display/236383515?recSetID=>
- Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelbäck, J., & Yannakakis, G. (2010). Multiobjective exploration of the starcraft map space.
URL <https://ieeexplore-ieee-org.ezproxy.usach.cl/document/5593346>
- Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., & Agapiou, J. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning.
URL <https://www-nature-com.ezproxy.usach.cl/articles/s41586-019-1724-z>
- Volk, M., Staegemann, D., Bosse, S., & Häusler, R. (2020). Approaching the (big) data science engineering process.
URL <https://www.scitepress.org/Papers/2020/95698/pdf/index.html>
- Wang, L., Zeng, Y., Chen, B., Pan, Y., & Cao, L. (2020). Team recommendation using order-based fuzzy integral and nsga-ii in starcraft.
URL <https://ieeexplore-ieee-org.ezproxy.usach.cl/document/9044841>
- Wang, P., Zeng, Y., Chen, B., & Cao, L. (2019). A data-driven approach to solve a production constrained build-order optimization problem.
URL <https://ieeexplore.ieee.org/document/8866045>
- Young, T. (2009). Metaheuristics: From design to implementation.
URL <https://onlinelibrary-wiley-com.ezproxy.usach.cl/doi/book/10.1002/9780470496916>