SKRIPSI

PERBANDINGAN *GAME-PLAYING AGENT* DENGAN ALGORITMA *MONTE CARLO TREE SEARCH* DAN *TEMPORAL DIFFERENCE TREE SEARCH* PADA PERMAINAN *2048*

Jiang Han

NPM: 6181801034

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022

UNDERGRADUATE THESIS

COMPARISON OF GAME-PLAYING AGENTS WITH MONTE CARLO TREE SEARCH AND TEMPORAL DIFFERENCE TREE SEARCH ALGORITHM ON THE *2048* GAME

Jiang Han

NPM: 6181801034

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2022

LEMBAR PENGESAHAN

PERBANDINGAN *GAME-PLAYING AGENT* DENGAN ALGORITMA *MONTE CARLO TREE SEARCH* DAN *TEMPORAL DIFFERENCE TREE SEARCH* PADA PERMAINAN *2048*

Jiang Han

NPM: 6181801034

Bandung, 5 Juli 2022

Menyetujui,

Pembimbing

Digitally signed by Lionov

Lionov, Ph.D.

Ketua Tim Penguji

Digitally signed by Natalia

Natalia, M.Si.

Anggota Tim Penguji

Digitally signed by Cecilia Esti Nugraheni

Dr.rer.nat. Cecilia Esti Nugraheni

Mengetahui,

Ketua Program Studi

Digitally signed by Mariskha Tri Adithia

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PERBANDINGAN *GAME-PLAYING AGENT*
DENGAN ALGORITMA *MONTE CARLO TREE SEARCH*
DAN *TEMPORAL DIFFERENCE TREE SEARCH*
PADA PERMAINAN *2048*

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 5 Juli 2022

Jiang Han
NPM: 6181801034

ABSTRAK

2048 adalah permainan video *open-source* yang viral pada tahun 2014. Daya tarik permainan tersebut terdapat pada mekanismenya yang sederhana, namun cukup menantang untuk dimainkan oleh pemain manusia. Sudah banyak model *game-playing agent* (GPA) yang dibuat untuk memainkan permainan tersebut secara otomatis. Berbagai GPA tersebut mengimplementasikan algoritma yang berbeda-beda sehingga memiliki tingkat keberhasilan yang bervariasi pula. Salah satu algoritma yang pernah diimplementasikan menjadi GPA untuk *2048* adalah algoritma *Monte Carlo Tree Search* (MCTS). Meskipun begitu, potensi MCTS dalam memainkan permainan tersebut belum sepenuhnya terealisasikan sebab MCTS memiliki banyak varian dan modifikasi yang dapat meningkatkan performa agen. Salah satu modifikasinya adalah dengan menggabungkan MCTS dengan *temporal-difference learning* (TD-Learning) sehingga menghasilkan algoritma *Temporal Difference Tree Search* (TDTS).

Pada penelitian ini, akan dibangun perangkat lunak permainan *2048* dan GPA dengan algoritma MCTS dan TDTS menggunakan bahasa pemrograman Java. Secara spesifik, varian algoritma yang digunakan adalah UCT dan Sarsa-UCT(λ), di mana UCT adalah varian MCTS yang memakai *selection policy* UCB1 dan Sarsa-UCT(λ) adalah varian TDTS yang dihasilkan dengan menggabungkan TD-Learning dengan algoritma UCT. Dilakukan berbagai analisis terhadap kedua algoritma tersebut untuk menerapkannya pada permainan *2048*. Kemudian, sejumlah eksperimen dilakukan untuk mencari konfigurasi parameter optimum untuk kedua algoritma tersebut. Lalu, performa keduanya dibandingkan dan dianalisis.

Hasil eksperimen menunjukkan bahwa, pada konfigurasi optimumnya masing-masing, algoritma Sarsa-UCT(λ) memiliki performa yang setidaknya sama baiknya dengan algoritma UCT pada permainan *2048*. Bahkan, jika anggaran komputasi yang tersedia sangat sedikit, algoritma Sarsa-UCT(λ) cenderung menghasilkan rata-rata skor yang lebih tinggi daripada algoritma UCT. Di sisi lain, skor yang dicapai oleh algoritma Sarsa-UCT(λ) tampak lebih fluktuatif daripada UCT sehingga Sarsa-UCT(λ) lebih sering menghasilkan skor yang lebih tinggi, namun lebih sering juga menghasilkan yang lebih rendah dari UCT. Selain itu, ditemukan bahwa beberapa nilai parameter optimal dari algoritma UCT dan Sarsa-UCT(λ) seperti konstanta eksplorasi, *reward discount rate*, dan *eligibility trace decay rate* memiliki korelasi positif dengan jumlah anggaran komputasi yang tersedia.

Kata-kata kunci: kecerdasan buatan, *game-playing agent*, *2048*, *Monte Carlo Tree Search*, *Temporal Difference Tree Search*

ABSTRACT

2048 is an open-source video game that went viral in 2014. The appeal of the game came from its simple, yet challenging mechanism for human players. There have been many game-playing agents (GPA) models that can play the game automatically. Various GPAs implement different algorithms and thus have varying success rates. One of the algorithms that GPAs have implemented for *2048* is the Monte Carlo Tree Search (MCTS) algorithm. However, MCTS' full potential has not been realized yet as it has many variations and modifications that can enhance an agent's performance. One such modification is by combining MCTS with temporal-difference learning (TD-Learning) which produces the Temporal Difference Tree Search (TDTS) algorithm.

In this study, a software consisting of the *2048* game and GPAs with MCTS and TDTS algorithms is built with the Java programming language. Specifically, the algorithm variants that are used are UCT and Sarsa-UCT(λ), where UCT is an MCTS variant that uses the UCB1 selection policy whereas Sarsa-UCT(λ) is a TDTS variant created by merging TD-Learning with the UCT algorithm. Various analyses have been done to apply those two algorithms to the *2048* game. Then, a number of experiments are performed in order to find the optimal parameter configuration for both algorithms. Afterward, their performances are compared and analyzed.

Experiment results show that Sarsa-UCT(λ) algorithm performs at least as good as the UCT algorithm in the *2048* game. Moreover, when the available computation budget is very low, Sarsa-UCT(λ) algorithm tends to produce a higher average score than the UCT algorithm. On the other hand, the scores achieved by the Sarsa-UCT(λ) algorithm seem to fluctuate more than UCT, causing Sarsa-UCT(λ) to achieve higher scores more frequently, but also to achieve lower scores more frequently than UCT. Furthermore, it is discovered that several optimal parameter values of UCT and Sarsa-UCT(λ) algorithms, such as the exploration constant, reward discount rate, and eligibility trace decay rate, appeared to have a positive correlation with the amount of available computation budget.

Keywords: artificial intelligence, game-playing agent, *2048*, Monte Carlo Tree Search, Temporal Difference Tree Search

Dipersembahkan untuk Tuhan, keluarga, dan teman-teman seperjuangan

KATA PENGANTAR

Puji dan syukur kepada Tuhan Yesus Kristus atas rahmat, anugerah, dan pimpinan-Nya sehingga penulis dapat menyelesaikan skripsi berjudul "Perbandingan *Game-Playing Agent* dengan Algoritma *Monte Carlo Tree Search* dan *Temporal Difference Tree Search* pada Permainan *2048*". Skripsi ini tidak dapat diselesaikan tanpa dukungan dari banyak pihak dalam berbagai bentuk. Oleh sebab itu, penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Ayah dan ibu penulis yang senantiasa mencurahkan kasih sayang mereka ke penulis dan memberi dukungan penuh selama perkuliahan. Juga kepada kakak laki-laki penulis yang memberikan bantuan moral yang sangat membantu saat penulis sedang terpuruk.
2. Bapak Lionov Wiratma selaku pembimbing skripsi yang memberikan banyak ilmu dan arahan agar skripsi ini dapat terselesaikan.
3. Ibu Natalia dan Ibu Cecilia Esti Nugraheni selaku dosen penguji yang telah memberikan berbagai masukan berharga dalam penulisan skripsi ini.
4. Ibu Joanna Helga, Bapak Husnul Hakim, Bapak Irvan Jahja yang telah mendidik, melatih, dan mendukung penulis untuk mengikuti berbagai perlombaan pemrograman kompetitif.
5. Warren Mazmur dan Bryan Heryanto selaku teman satu tim pemrograman kompetitif dan juga sebagai teman baik selama perkuliahan di UNPAR yang mendukung penulis dalam banyak hal.
6. Seluruh dosen Teknik Informatika UNPAR yang telah mengajar dan memberikan ilmunya kepada penulis.
7. Pak Theodorus Riswandi, Pak Andi Yunarto, dan Ibu Sylvia Yazid dari Kantor Internasional dan Kerja Sama UNPAR yang membantu penulis mengurus visa, izin belajar, dan izin tinggal selama berkuliah di UNPAR.
8. Vicky Ricardo Savero, Craven Sachio Saputra, Johan Adi Wiguna, Juan Anthonius Kusjadi, Jason Elington, Michael Michio Hutagaol, Gunawan Christianto, Joshua Lauwrich Nandy, Timothy Lawrence, dan masih banyak pihak-pihak lain menjadi teman dan senior di perkuliahan yang telah banyak bertukar pikiran dan cerita dengan penulis.

Semoga skripsi ini dapat memberikan manfaat baik bagi para pembaca yang budiman maupun untuk penelitian-penelitian berikutnya. Bila terdapat kesalahan dalam penulisan skripsi ini, penulis mohon maaf sebesar-besarnya

Bandung, Juli 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1 Latar Belakang

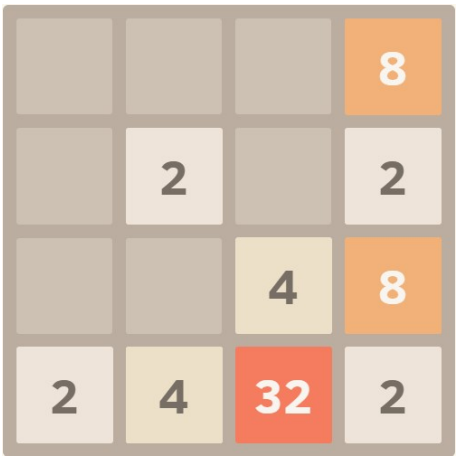
Agen adalah hal apapun yang dapat mengamati lingkungannya melalui sensor dan dapat melakukan suatu aksi terhadap lingkungan tersebut melalui aktuator. Hasil pengamatan agen terhadap lingkungannya disebut *percept*. Agen harus dapat menentukan aksi yang akan dijalankannya sepenuhnya berdasarkan pengetahuan bawaan yang dimilikinya dan sejarah *percept* yang diamatinya, tidak bisa berdasarkan suatu informasi yang belum pernah diamati. Perilaku agen didefinisikan oleh sebuah *agent function*, yang merupakan pemetaan rangkaian *percept* ke aksi yang akan dijalankan. Implementasi dari *agent function* disebut *agent program*.

Menurut Russell dan Norvig [1], agen rasional atau agen cerdas adalah agen yang selalu melakukan aksi yang paling "benar". Ukuran benar/salahnya suatu aksi ditentukan oleh *performance measure* (ukuran kinerja) yang dipakai, umumnya berdasarkan konsekuensi dari aksi tersebut. Agen rasional berbeda dari agen yang sempurna, sebab bisa saja agen rasional menjalankan aksi yang tidak optimal akibat kurangnya *percept* atau akibat sifat lingkungan yang stokastik. Jadi, maksud dari agen rasional adalah agen yang selalu memilih aksi yang memaksimalkan ukuran kinerja harapannya berdasarkan rangkaian *percept* yang sudah diterimanya dan pengetahuan bawaan yang dimilikinya. Untuk menjamin rasionalitasnya, agen juga dapat melakukan *information gathering* dan *learning*. *Information gathering* berarti agen melakukan aksi tertentu untuk mencari lebih banyak informasi mengenai lingkungannya, sedangkan *learning* berarti agen memperbarui pengetahuan bawaannya berdasarkan *percept* yang diterimanya. Agen yang rasional juga harus otonom, artinya agen harus lebih banyak bergantung kepada *percept* yang diterimanya dan pengalamannya daripada kepada pengetahuan bawaan yang diatur oleh perancang agen.

Game-playing agent (GPA) adalah agen yang berinteraksi dengan sebuah lingkungan permainan tertentu. Lingkungan permainan biasanya bersifat kompetitif, di mana ada dua atau lebih agen yang memiliki tujuan yang berlawanan, namun ada juga lingkungan di mana agen-agennya saling bekerja sama untuk mencapai satu tujuan atau yang jumlah agennya hanya satu. Agen untuk permainan lebih mudah diteliti karena lingkungannya yang lebih sederhana, jumlah aksinya yang lebih sedikit, dan aturannya yang lebih presisi. GPA, khususnya pada lingkungan yang kompetitif, biasanya menggunakan algoritma *adversarial search* dalam *agent program*-nya.

2048 adalah sebuah permainan video pemain tunggal *open-source* yang dikembangkan oleh Gabriele Cirulli, seorang pemrogram asal Italia, pada tahun 2014. Permainan ini dapat dimainkan pada peramban internet (pada tautan https://play2048.co/) atau perangkat bergerak. Sesaat setelah dirilis, permainan ini menjadi viral dan dimainkan oleh banyak orang. Bahkan dalam kurun waktu kurang dari satu bulan sejak dirilis, halaman web *2048* milik Cirulli telah dikunjungi oleh 10 juta pengunjung unik dan permainan tersebut telah dimainkan lebih dari 100 juta kali[1]. Daya tarik dari permainan ini terletak pada konsep dan cara permainannya yang sederhana, namun cukup menantang untuk diselesaikan bagi pemain manusia. Antarmuka dari permainan ini dapat dilihat pada Gambar 1.1. Kode sumber dari *2048* yang dipublikasikan secara terbuka memungkinkan

[1] Arjun Kharpal, *19-year-old makes viral game hit in a weekend*, diakses dari https://www.cnbc.com/2014/03/26/19-year-old-makes-viral-game-hit-in-a-weekend.html, diakses pada tanggal 2 Januari 2022, pukul 21:06 WIB.



Gambar 1.1: Cuplikan layar dari permainan *2048*.

pengembang-pengembang perangkat lunak untuk membuat banyak modifikasi dari permainan tersebut. Beberapa contoh hasil modifikasinya adalah *2048-3D*[2] dan *Deterministic 2048*[3].

Pada permainan *2048*, diberikan sebuah papan berukuran 4×4 sel yang mengandung beberapa *tile* pada sel-sel tertentu. Pada setiap *tile*, terdapat sebuah bilangan perpangkatan dua yang menyatakan nilai dari *tile* tersebut. Dalam satu langkah, pemain dapat menggeser seluruh *tile* ke atas, bawah, kiri, atau kanan. Jika ada dua *tile* dengan nilai yang sama bertabrakan, maka kedua *tile* tersebut akan bersatu membentuk *tile* baru yang nilainya setara hasil penambahan nilai dari dua *tile* sebelumnya. Skor pemain juga akan bertambah sebanyak nilai dari *tile* baru yang dihasilkan. Suatu langkah dianggap sah hanya jika ada minimal satu *tile* yang berpindah tempat atau bersatu dengan *tile* lain akibat langkah tersebut. Di akhir setiap langkah, sebuah *tile* baru bernilai 2 (dengan peluang 90%) atau 4 (dengan peluang 10%) akan ditambahkan ke dalam papan pada salah satu sel kosong yang dipilih secara acak. Permainan diawali dengan dua *tile* bernilai 2 atau 4 dan berakhir jika sudah tidak ada langkah yang sah (seluruh papan sudah dipenuhi dengan *tile* dan tidak ada *tile* yang bisa digabungkan). Tujuan permainan ini adalah untuk membentuk *tile* yang bernilai 2048 (oleh karena itu permainan ini diberi nama demikian), namun setelah tujuan itu terpenuhi, pemain dapat terus memainkan permainannya untuk mendapatkan skor setinggi-tingginya.

Meskipun memiliki mekanisme dan aturan yang sederhana, ternyata *2048* cukup menantang untuk dikuasai oleh pemain manusia. Menurut Cirulli, selang dua minggu setelah dirilis, permainan ini telah dimainkan lebih dari 100 juta kali, namun hanya 1% yang berhasil mencapai tujuan permainan[4]. Saat ini, strategi permainan *2048* dapat dicari dengan mudah di Internet, namun tetap saja ada tantangan lain bagi pemain manusia. Sekalipun sudah mengikuti strategi yang terbaik, manusia tetap bisa teledor dan melakukan kesalahan sehingga menghasilkan kekalahan. Supaya tidak teledor, bisa saja manusia bermain dengan lebih pelan dan hati-hati, namun hal ini akan membuat progres permainan menjadi lebih lambat sehingga membuat manusia lebih cepat bosan. Selain itu, jika bermain terlalu lama, maka manusia akan kelelahan sehingga meningkatkan peluang terjadi kesalahan.

Mempertimbangkan keterbatasan manusia, menjadi menarik untuk mencoba membuat perangkat lunak berupa agen permainan untuk memainkan permainan *2048*. Tidak seperti manusia, perangkat lunak tidak bisa menjadi lelah atau melakukan keteledoran sehingga ada peluang untuk mendapatkan skor yang lebih tinggi daripada pemain manusia. Saat ini, sudah ada banyak agen permainan yang diciptakan dengan berbagai macam algoritma untuk mencapai skor setinggi-tingginya. Secara informal, beberapa agen permainan yang telah dibuat dibagikan melalui situs tanya jawab Stack

[2]Michal Opler, *2048-3D*, https://joppi.github.io/2048-3D/, diakses pada 04 Juni 2022 pukul 07:33 WIB.
[3]Pengguna Github dengan nama akun *jmfork*, *Deterministic 2048*, https://jmfork.github.io/2048/, diakses pada 23 Juli 2022 pukul 05:46 WIB.
[4]Lihat catatan kaki 1.

Overflow[5]. Secara formal pun, sudah ada banyak publikasi ilmiah mengenai performa berbagai agen permainan dalam memainkan permainan ini.

Ada banyak macam algoritma yang telah digunakan untuk membuat agen permainan *2048* dengan performa yang beragam, mulai dari algoritma berbasis pencarian pada pohon hingga algoritma pembelajaran mesin. Mayoritas solusi di situs tanya jawab *Stack Overflow* menggunakan algoritma berbasis pencarian pada pohon permainan, seperti *minimax*, *expectimax*, atau *Monte Carlo Tree Search*. Salah satu agen dengan performa terbaik adalah agen yang dibuat oleh Robert Xiao dengan nama akun *nneonneo*[6]. Xiao mengimplementasikan agennya dengan algoritma *expectimax*, pengkodean *game state* menjadi bilangan biner (yang disebut *bitboard*), dan beberapa fungsi heuristik hingga berhasil mencapai skor tertinggi sebesar 794076 dengan median skor 387222.

Di sisi lain, telah dibuat juga agen-agen yang menggunakan algoritma berbasis pembelajaran mesin, seperti *temporal-difference learning*, *n-tuple network*, dan *convolutional neural network*, terutama di kalangan peneliti. Salah satu agen berbasis pembelajaran mesin dengan performa paling baik adalah agen yang dibuat dan diteliti oleh Jaśkowski [6]. Algoritma yang digunakan oleh Jaśkowski adalah *temporal-difference learning* dengan *n-tuple network* dan beberapa modifikasi untuk meningkatkan performanya. Hasilnya, agen yang diteliti memiliki rata-rata skor sebesar 609104. Meskipun begitu, tidak semua agen berbasis pembelajaran mesin memiliki performa yang baik. Dari hasil perbandingan performa antara agen-agen berbasis *deep neural network* dengan agen yang dibuat dan diteliti oleh Matsuzaki [7], terlihat bahwa agen yang memakai salah satu algoritma *reinforcement learning* yang disebut *Q-learning* cenderung memiliki performa yang buruk karena rata-rata skor yang dicapainya tidak sampai 20000. Agen permainan yang dibuat oleh Dedieu dan Amar dengan algoritma *deep reinforcement learning* juga tidak memberikan hasil yang memuaskan sehingga skornya tidak dicantumkan dalam hasil penelitian mereka.

Monte Carlo Tree Search (MCTS) adalah salah satu algoritma yang cukup sering dipakai untuk membuat agen permainan. Browne, dkk. [8] menyebutkan bahwa kelebihan utama dari MCTS adalah algoritmanya yang bersifat umum (dapat bermain dengan cukup baik hanya dengan mengetahui mekanisme permainan tanpa perlu diajarkan strategi permainannya) dan bersifat *anytime* (lamanya algoritma berjalan dapat disesuaikan dengan jumlah waktu yang tersedia) di mana peningkatan daya komputasi umumnya juga menghasilkan peningkatan performa permainan. Sudah ada beberapa agen permainan *2048* yang dibuat—secara formal maupun informal—dengan algoritma MCTS dengan performa yang cukup baik, namun potensi dari algoritma ini belum sepenuhnya terealisasikan. MCTS sendiri sebetulnya bukan sebuah algoritma spesifik, melainkan sebuah kerangka (*framework*) algoritma yang dapat dikonfigurasi untuk menghasilkan sekelompok algoritma konkret yang berbeda-beda. Salah satu algoritma konkret MCTS yang paling populer adalah algoritma UCT (*upper confidence bounds applied to trees*). Masih banyak variasi dari MCTS yang belum diuji performanya pada permainan *2048*.

Dalam disertasinya pada tahun 2018, Vodopivec mengusulkan untuk menginkorporasikan metode *temporal-difference learning* (TD-Learning) dengan MCTS menjadi kerangka algoritma baru yang dinamakan *Temporal Difference Tree Search* (TDTS) [4]. TD-Learning sendiri adalah salah satu algoritma *reinforcement learning* yang dipakai dalam pembuatan beberapa agen permainan *2048* yang performanya paling baik. Dari metode TDTS, Vodopivec merancang sebuah algoritma yang disebut Sarsa-UCT(λ). Setelah dilakukan pengujian performa agen dengan algoritma UCT dan Sarsa-UCT(λ) pada beberapa permainan klasik, ditemukan bahwa performa agen Sarsa-UCT(λ) setidaknya sama baiknya dengan agen UCT. Bahkan dalam kasus tertentu, agen Sarsa-UCT(λ) bisa memiliki performa yang lebih baik dari agen UCT. Oleh karena itu, metode TDTS ini tampak menarik dan menjanjikan untuk dipakai sebagai algoritma untuk agen permainan *2048*.

Pada skripsi ini, perangkat lunak permainan *2048* akan dibuat ulang sebagai lingkungan permainan dan pengujian GPA. Selain dipakai untuk melakukan pengujian, permainan *2048* ini

[5] *What is the optimal algorithm for the game 2048?*, https://stackoverflow.com/questions/22342854, diakses pada 12 Oktober 2021, pukul 15:09 WIB

[6] Jawaban pengguna dengan nama akun *nneoneo* pada pertanyaan *What is the optimal algorithm for the game 2048?*, https://stackoverflow.com/a/22498940, diakses pada 12 Oktober 2021, pukul 15:27 WIB.

juga dapat dimainkan oleh pengguna perangkat lunak untuk mengenali permainan tersebut. Setelah itu, akan dibuat dua agen permainan yang masing-masing menggunakan algoritma MCTS dan TDTS dengan algoritma spesifik yang digunakan secara berturutan adalah UCT dan Sarsa-UCT(λ). Sebagai tambahan, akan dibuat juga agen yang bermain secara acak sebagai tolak ukur rasionalitas dari agen-agen lainnya. Perangkat lunak akan memiliki dua antarmuka pengguna, yaitu dalam bentuk *command-line interface* (CLI) dan *graphical user interface* (GUI). CLI akan dipakai untuk mengkonfigurasi pengujian agen permainan dan mengeluarkan hasil akhir pengujian, baik melalui *terminal* maupun melalui berkas teks. GUI akan dipakai untuk memainkan permainan *2048* untuk pemain manusia. Keseluruhan perangkat lunak akan diimplementasikan dengan bahasa Java.

Setelah perangkat lunak dibangun, akan dilakukan serangkaian eksperimen terhadap agen-agen permainan yang tersedia. Mula-mula, akan dilakukan eksperimen terhadap agen yang bermain secara acak untuk mendapatkan tolak ukur dari performa agen permainan. Lalu, akan dilakukan beberapa eksperimen untuk mencari nilai parameter yang optimal, terutama untuk agen UCT dan Sarsa-UCT(λ). Terakhir, performa dari kedua agen UCT dan Sarsa-UCT(λ) akan dibandingkan dan dianalisis. Sifat *anytime* dari algoritma MCTS juga dimiliki oleh UCT dan Sarsa-UCT(λ), yang berarti kedua algoritma tersebut dapat berjalan selama mungkin sesuai keinginan penguji. Namun untuk alasan kepraktisan, durasi permainan agen akan dibatasi selama pengujian dilakukan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dipaparkan, berikut adalah rumusan masalah yang hendak diselesaikan oleh skripsi ini:

1. Bagaimana cara membangun perangkat lunak permainan *2048*—menggunakan bahasa Java—yang dapat diintegrasikan dengan agen permainan?
2. Bagaimana cara merancang agen permainan *2048* yang menggunakan algoritma *Monte Carlo Tree Search* (MCTS) dan *Temporal Difference Tree Search* (TDTS), dan kemudian mengimplementasikannya dengan bahasa Java?
3. Bagaimana performa agen permainan dengan algoritma TDTS bila dibandingkan dengan agen permainan yang menggunakan algoritma MCTS yang standar?

1.3 Tujuan

Berdasarkan masalah yang telah dirumuskan sebelumnya, maka skripsi ini bertujuan untuk:

1. Membangun perangkat lunak permainan *2048*—menggunakan bahasa Java—yang dapat diintegrasikan dengan agen permainan.
2. Merancang agen permainan *2048* yang menggunakan algoritma *Monte Carlo Tree Search* (MCTS) dan *Temporal Difference Tree Search* (TDTS), lalu mengimplementasikannya dengan bahasa Java.
3. Membandingkan performa agen permainan dengan algoritma TDTS dengan agen permainan yang menggunakan algoritma MCTS yang standar.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Versi permainan 2048 yang digunakan adalah versi asli yang dibuat oleh Gabriele Cirulli yang terdapat pada tautan https://github.com/gabrielecirulli/2048 dengan peraturan yang sesuai dengan penjelasan pada Bagian 1.1.
2. Waktu pengujian performa agen-agen permainan *2048* dibatasi sedemikian rupa sehingga satu permainan tidak menghabiskan waktu lebih dari dua menit.

1.5 Metodologi

Metodologi penelitian yang akan digunakan antara lain:

1. Memainkan permainan *2048* yang sudah ada untuk mendapatkan pemahaman dasar mengenai mekanisme permainannya serta mengenali strategi-strategi dasar dari permainan tersebut.
2. Melakukan studi literatur mengenai MCTS, *temporal-difference learning*, dan TDTS.
3. Melakukan analisis terhadap algoritma yang akan dipakai.
4. Melakukan analisis dan desain perangkat lunak yang akan dibangun.
5. Membangun perangkat lunak permainan *2048*, agen permainan, dan alat pengujian dengan bahasa Java berdasarkan desain yang sudah dibuat.
6. Melakukan eksperimen untuk mencari nilai-nilai parameter yang paling optimal untuk masing-masing agen berbasis MCTS dan TDTS.
7. Melakukan eksperimen dan membandingkan performa dari agen MCTS dan TDTS.
8. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Pembahasan penelitian dalam skripsi ini akan dibagi ke dalam tujuh bab, yang terdiri dari:

1. **Bab 1 Pendahuluan**
   Bab 1 akan membahas latar belakang mengenai pengertian *game-playing agent*, permainan *2048*, serta algoritma-algoritma yang dapat dipakai untuk memainkan permainan *2048*, terutama MCTS dan TDTS. Bab ini juga akan membahas mengenai rumusan masalah, tujuan yang ingin dicapai, batasan masalah, metodologi penelitian, dan sistematika pembahasan dari skripsi ini.
2. **Bab 2 Landasan Teori**
   Bab 2 akan membahas mengenai dasar teori yang dibutuhkan untuk melakukan penelitian ini, terutama mengenai *game-playing agent*, permainan *2048*, dan algoritma MCTS serta TDTS.
3. **Bab 3 Analisis Masalah dan Perangkat Lunak**
   Bab 3 akan membahas mengenai berbagai macam analisis yang dilakukan untuk memecahkan masalah penelitian, mulai dari pemodelan permainan *2048*, analisis penerapan algoritma MCTS dan TDTS pada *2048*, dan juga analisis perangkat lunak yang akan dibangun.
4. **Bab 4 Perancangan Perangkat Lunak**
   Bab 4 akan membahas mengenai rancangan perangkat lunak, mulai dari rancangan antarmuka, rancangan proses, hingga rancangan kelasnya.
5. **Bab 5 Implementasi dan Pengujian**
   Bab 5 akan membahas detail implementasi dari perangkat lunak pengujian GPA pada permainan *2048*. Bab ini juga membahas pengujian-pengujian yang dilakukan untuk memastikan perangkat lunak sudah berjalan dengan benar.
6. **Bab 6 Eksperimen *Game-Playing Agent***
   Bab 6 akan membahas mengenai eksperimen-eksperimen yang dilakukan dengan perangkat lunak yang sudah dibangun untuk membandingkan performa dari agen MCTS dan TDTS pada permainan *2048*.
7. **Bab 7 Kesimpulan dan Saran**
   Bab 7 akan membahas mengenai kesimpulan dari penelitan yang sudah dilakukan dan saran-saran untuk penelitian lanjutan pada topik yang sama.