Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 7, Juli 2018, hlm. 2644-2648 http://j-ptiik.ub.ac.id

Penerapan *Adaptive* AI pada *Game Turn Based* RPG Dengan Menggunakan Metode *Monte Carlo Tree Search*

Yuka Bimatara Putra[1], Eriq Muh. Adams Jonemaro[2], Muhammad Aminul Akbar[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]hc.art.yuka@gmail.com, [2]eriq.adams@ub.ac.id, [3]muhammad.aminul@ub.ac.id

Abstrak

Banyak metode yang dapat digunakan oleh para *game developer* untuk diterapkan menjadi Artificial Intelligence (AI) dalam agen musuh atau biasa disebut Non Playable Character (NPC) yang bertujuan agar pemain menjadi semakin tertantang dalam menamatkan *game* yang dimainkannya. Salah satu algoritme yang biasa dipakai adalah *Monte Carlo Tree Search* (MCTS). Algoritme MCTS telah terbuksi sukses untuk diterapkan pada permainan *board game turn-based*, yaitu permainan GO. Pada percobaan tersebut, agen AI yang menggunakan MCTS mendapatkan nilai tertinggi dibandingkan dengan penelitian sebelumnya, sehingga saat diuji coba, AI yang menggunakan algoritme MCTS dapat mengalahkan juara pemain GO internasional. Melihat kesuksesan penerapan MCTS tersebut, penelitian ini akan membahas penerapan MCTS pada agen musuh di permainan *turn-based* RPG. Pengujian dilakukan untuk memvalidasi *decision making* yang dipilih agen dan pengaruh loop proses MCTS terhadap *skill* yang akan dipilih. Untuk validasi *decision making*, pengujian akan dilakukan dengan cara melihat tingkat efektifitas dan untuk pengujian pengaruh *loop* proses MCTS terhadap efektifitas dilakukan dengan cara simulasi dengan beberapa skenario yang menggunakan *loop* MCTS yang berbeda-beda. Tingkat efektifitas diuji dengan cara melihat rasio kemenangan (*win ratio*) dari tim agen.

Kata kunci: *Artificial Intelligence, Monte Carlo Tree Search, MCTS, Turn Based RPG.*

Abstract

Many methods can be used by game developers to be implemented into Artificial Intelligence (AI) in enemy agents or commonly called Non Playable Character (NPC) which aims to make players become more challenged in completing the game they play. One of the usual algorithms is Monte Carlo Tree Search (MCTS). The MCTS algorithm has been successfully constructed for turn-based gaming board games, the GO game. In the experiment, AI agents using MCTS scored higher than previous studies, so when tested, AIs using the MCTS algorithm can beat international GO champions. Given the success of the MCTS implementation, this study will discuss the application of MCTS to enemy agents in turn-based RPG games. The test is performed to validate the decision making of the selected agent and the effect of the MCTS process loop on the skill to be selected. For validation of decision making, testing will be done by looking at the effectiveness level and for testing the effect of MCTS process loop on effectiveness is done by simulation with several scenarios using different MCTS loops. The level of effectiveness is tested by looking at the win ratio of the agent team.

Keywords: *Artificial Intelligence, Monte Carlo Tree Search, MCTS, Turn Based RPG.*

1. PENDAHULUAN

Saat ini terdapat banyak sekali macam *gameplay* yang bervariasi. *Game turn-based* RPG merupakan salah satu tipe *game* yang sedang *hype* dalam beberapa tahun terakhir. Pemain akan diajak untuk mengontrol karakter yang sudah ditentukan oleh cerita atau membuat karakter baru dengan skill yang berbeda-beda, yang tetap dalam satu tujuan yaitu untuk mengalahkan musuh. Namun, apabila musuh tidak didukung dengan adanya penerapan artificial intelligence (AI) didalamnya, permainan akan terasa kurang kompetitif. Perilaku yang kurang kompetitif tersebut dikarenakan behavior yang dipunyai oleh musuh tidak mempunyai kombinasi aturan atau musuh hanya diberi aturan if-then dengan urutan aturan yang tetap saja. Tanpa adanya AI yang

kompetitif, *game turn-based* RPG akan terasa mudah ditebak dan terkesan *repetitive*, sehingga dengan adanya hal ini menyebabkan berkurangnya nilai *entertainment* atau hiburan yang didapatkan oleh pemain (Wang & Tan, 2015).

Salah satu metode yang sudah dipakai oleh para peneliti beberapa tahun terakhir adalah *Monte Carlo Tree Search* (MCTS). MCTS mulai banyak digunakan oleh para peneliti dikarenakan bertambahnya jumlah publikasi yang membahas tentang MCTS beberapa tahun akhir ini yang dikarenakan sebelumnya telah terbukti sukses membuat juara dunia pemain catur GO dikalahkan dengan AI yang menggunakan MCTS walaupun masih digabungkan dengan AI lain (Galvan-Lopez, et al., 2014). Pada dasarnya MCTS adalah metode yang melakukan *playout* sebanyak mungkin dan akan menyimpan statistik data dari menang / kalah dari simulasi tersebut. Karena seiring dengan bertambahnya *playout*, maka AI akan semakin tahu keputusan yang terbaik yang akan dipilih. Selain permainan catur GO, penerapan juga sudah berhasil pada *game modern board-game* SETTLERS OF CATAN dan juga *game* ber*genre real time strategy* (RTS) SPRING (Chaslot, 2008). Namun, dari penelitian yang sudah dilakukan sebelumnya masih belum ada yang mencoba menerapkan pada *game turn-based* RPG. Berdasarkan hal tersebut, penelitian ini akan membahas penerapan metode MCTS kedalam *game turn-based* RPG. *Game turn-based* RPG memiliki factor bercabang yang banyak sesuai dengan aksi-aksi yang dipunyai setiap karakter yang ada, sehingga metode MCTS yang berbasis *simulasi playout* ini cocok untuk diterapkan *game* sejenis ini.

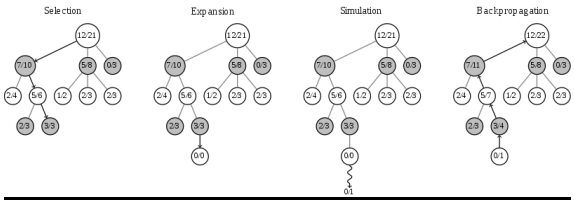
2. TURN BASED ROLE PLAYING GAME

Turn-based role playing game (RPG) adalah jenis *game* yang masih termasuk dalam *genre* RPG. Dalam *turn-based* RPG, pertarungan yang dilakukan berdasarkan giliran dimana player disaat gilirannya dapat melakukan command kepada karakter mereka untuk melakukan suatu aksi dimana setiap giliran yang didapatkan digunakan untuk bertahan hidup dengan tujuan untuk bisa mengalahkan musuh. Aksi ini biasanya berupa mengeluarkan *skill* untuk menyerang yang biasanya setiap pengeluaran skill yang dilakukan membutuhkan *skill point*. Selain aksi berupa mengeluarkan skill ada juga aksi untuk bertahan, ataupun menggunakan suatu benda yang didapatkan dari pertarungan sebelumnya, ataupun aksi lain yang sesuai dari *rule* yang sudah ada dalam *game* tersebut.

3. MONTE CARLO TREE SEARCH

Dalam ilmu komputer, *Monte Carlo Tree Search* (MCTS) merupakan algoritme *heuristic search* untuk beberapa jenis *decision making*, terutama untuk permainan yang membutuhkan kombinasi-kombinasi tertentu. Metode MCTS atau yang sebelumnya hanya diberi nama *Monte Carlo* tersebut ditemukan setelah munculnya metode *minimax* di tahun 1940 yang dengan tujuan untuk menyempurnakan metode *minimax* yaitu saat itu hanya menggunakan *random sampling*. Lalu disempurnakan lagi pada tahun 2006 dan menjadi nama MCTS sampai saat ini.

Dalam penggunaan algoritme MCTS dalam AI untuk *game* hampir sama layaknya algoritme *minimax*. Namun, dengan MCTS kita tidak perlu untuk mencari semua *node*s yang ada dalam *tree node* untuk mencari solusi yang optimal. Selain itu, dengan MCTS dapat memberikan hasil yang lebih bagus dengan lebih kecilnya *branching factor* dan adaptasi yang relatif sederhana. Proses penerapan metode MCTS dibagi menjadi empat tahapan yaitu *selection*, *expansion*, *simulation*, dan *backpropagation yang masing-masing proses tahapan dapat membuat pertumbuhan tree secara asimetris seperti yang ditunjukkan pada Gambar 1*.



Gambar 1 Cara Kerja Monte Carlo Tree Search

3.1. Selection

Tahap *selection* merupakan proses pemilihan node yang ada di dalam tree hingga mencapai node *leaf*. Proses pememilihan tersebut bisa dilakukan melalui pemilihan acak maupun dengan menggunakan algoritma seleksi lainnya, seperti *Upper Confident Bound for Tree* (UCB). Adapun persamaan UCB ditunjukkan pada Persamaan (1).

$$\mathrm{UCB}_j = X_j + \mathrm{C}\sqrt{\frac{N_p}{N_c}} \qquad (1)$$

Dimana X_j merupakan rata-rata skor yang didapat, C adalah konstanta eksplorasi, N_p adalah jumlah pengunjungan pada node parent. Dan N_C adalah jumlah pengunjungan pada node saat ini.

3.2. Exploration

Tahap *exploration* adalah tahapan untuk memilih node child yang belum pernah dikunjungi. Pada tahap *selection*, node child akan di seleksi apabila semua node telah dikunjungi dengan minimal pengunjungan adalah 1. Namun apabila terdapat node child yang belum dikunjungi, maka node tersebut akan menjadi pilihan dan dilanjutkan pada tahap selanjutnya.

3.3 Simulation

Pada tahap *play-out*, node terpilih yang telah diproses pada tahap sebelumnya, akan dilakukan proses simulasi. Proses simulasi untuk tiap game yang berbeda akan menghasilkan proses yang berbeda-beda juga. Umumnya proses simulasi dilakukan dengan mengambil pergerakan *random* yang dilakukan secara berulang-ulang.

3.4 Backpropagation

Pada tahap ini, akan dilakukan update *score* mulai dari node yang telah dipilih pada tahap *selection* atau *exploration*, menuju ke parent hingga mencapai node *root* kembali.

4. PENERAPAN MCTS

Pada penerapan MCTS akan dilakukan sebagai penentuan pengambilan keputusan. MCTS akan membuat temporary character dimana temporary character itu berguna untuk melakukan simulasi pertarungan yang kemungkinan akan terjadi dalam beberapa turn kedepan, dimana hasil dari simulasi nanti akan didapatkan score dari masing-masing aksi yang dipunyai setiap character. Setelah selesai proses simulasi untuk mencari aksi mana yang mempunyai score yang terbaik, score yang sudah didapatkan akan dimasukkan kedalam character asli. Character asli inilah yang nantinya akan mengeluarkan aksi sesuai perhitungan score yang terbaik yang sudah didapatkan dari proses MCTS yang sudah dilakukan sebelumnya.

5. PENGUJIAN

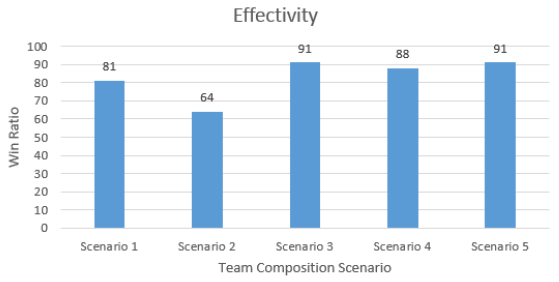
Agen AI diuji dengan 2 jenis pengujian yaitu pengujian efektifitas dan pengujian pengaruh loop terhadap tingkat efektifitas. Pengujian efektifitas dimaksudkan untuk mengetahui apakah agen AI yang menerapkan metode MCTS dalam game turn-based RPG mendapatkan rasio kemenangan yang lebih tinggi dari static AI. Lalu, untuk pengujian pengaruh loop dibagi menjadi 2 pengujian yaitu pengaruh loop MCTS dan pengaruh loop simulation dimana dari pengujian 2 loop yang terdapat dalam proses MCTS ini dimaksudkan untuk mengetahui apakah seiring dengan bertambahnya loop dapat meningkatkan rasio kemenangan yang didapatkan oleh agen AI.

5.1 Pengujian Efektifitas

Pengujian ini dijalankan dengan 5 skenario yang masing-masing skenario menggunakan komposisi team yang berbeda-beda yang ditunjukkan pada Tabel 1 dan Gambar 2.

Tabel 1. Hasil Pengujian Efektifitas

No	Skenario Komposisi Team	Win ratio
1	Healer Healer Warrior vs Wizard Warrior Healer	81
2	Healer Healer Wizard vs Wizard Warrior Healer	64
3	Warrior Warrior Healer vs Wizard Wizard Healer	91
4	Wizard Healer Wizard vs Warrior Warrior Healer	88
5	Warrior Healer Wizard vs Warrior Healer Wizard	91
	Rata-rata	83%



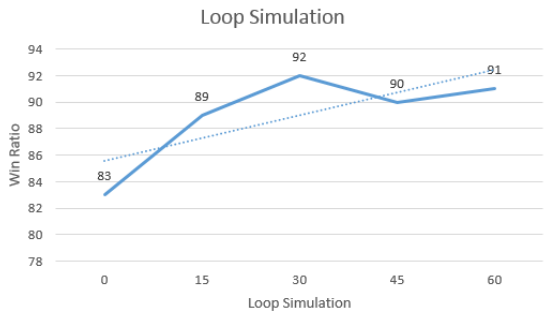
Gambar 2. Hasil Pengujian Efektifitas

Didapatkan hasil bahwa algoritme MCTS terbukti efektif untuk digunakan dalam permainan *turn-based* RPG, dimana dari 5 skenario komposisi team yang berbeda-beda yang telah diuji, didapatkan rasio kemenangan dari masing-masing skenario lebih dari 80% dan

rata-rata rasio kemenangan yang didapatkan adalah 83%, walaupun masih ada 1 komposisi team yang kurang dari 80%, skenario 4. Skenario 4 yang saat itu team MCTS yang menggunakan komposisi team Healer Healer Wizard melawan team static Wizard Warrior Healer didapatkan rasio kemenangan yang paling rendah daripada yang lainnya yaitu 64%. Hal ini terjadi dikarenakan team MCTS yang terdiri dari 2 anggota team Healer dan skill heal mendapatkan score tertinggi sehingga healer lebih banyak mengeluarkan heal daripada mengeluarkan skill untuk menyerang musuh.

5.2 Pengujian Loop Simulation

Pada pengujian *loop simulation*, team MCTS di uji dengan menggunakan *loop simulation* yang berbeda-beda dimana *loop simulation* ini berguna untuk melihat kemungkinan aksi yang akan dikeluarkan dari tiap karakter dalam beberapa turn selanjutnya.

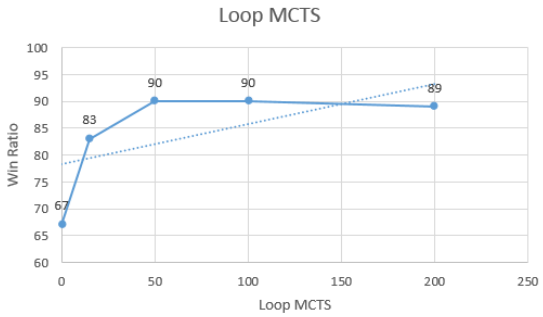


Gambar 3. Hasil Pengujian Loop Simulation

Didapatkan hasil bahwa semakin bertambahnya jumlah *loop simulation* yang digunakan dalam proses MCTS, semakin bertambah jumlah *loop* maka akan meningkat juga jumlah rasio kemenangan yang akan didapatkan oleh agen AI yang menggunakan MCTS. Namun, MCTS juga memiliki batas *loop simulation* yang menyebabkan rasio kemenangan akan tetap stabil dan jumlah *loop* itu tergantung *game* yang diuji dengan menerapkan algoritme MCTS ini.

5.3 Pengujian Loop MCTS

Pada pengujian *loop* MCTS, team MCTS di uji dengan menggunakan *loop* MCTS yang berbeda-beda dimana *loop* MCTS ini berguna untuk melihat kemungkinan semua aksi yang dimiliki dari agen AI.



Gambar 4. Hasil Pengujian Loop MCTS

Sama seperti pengaruh *loop simulation*, hal ini membuktikan bahwa semakin bertambahnya jumlah *loop* MCTS yang digunakan dalam proses MCTS, semakin bertambah jumlah *loop* maka akan meningkat juga jumlah rasio kemenangan yang akan didapatkan oleh agen AI yang menggunakan MCTS. Namun, MCTS juga memiliki batas *loop* MCTS yang menyebabkan rasio kemenangan akan tetap stabil dan jumlah *loop* itu tergantung *game* yang diuji dengan menerapkan algoritme MCTS ini.

6. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan terhadap penerapan MCTS pada *game turn-based* RPG, maka dapat disimpulkan beberapa poin sebagai berikut:

1. Penerapan MCTS dapat digunakan untuk pemilihan skill dalam permainan *turn-based* RPG.
2. Inti dari proses MCTS adalah pada proses *simulation*. Pada proses *simulation* digunakan untuk memprediksi kejadian pada beberapa giliran selanjutnya.
3. Penggunaan *loop* mcts dan *loop simulation* sangat berpengaruh terhadap rasio kemenangan. Semakin banyak *loop* yang digunakan, maka semakin meningkat juga rasio kemenangan yang didapatkan. Namun, MCTS juga memiliki batas *loop* yang menyebabkan rasio kemenangan akan tetap stabil ataupun fluktuatif dan jumlah *loop* itu tergantung kondisi *game* yang diuji dengan menerapkan algoritme MCTS.
4. Metode MCTS terbukti efektif jika diterapkan pada permainan *turn-based* RPG yang ditunjukkan dengan rata-rata rasio kemenangan yang didapatkan dengan menggunakan 5 skenario

komposisi team yang berbeda yaitu 83%.

DAFTAR PUSTAKA

Tobias Graf, Marco Platzner, "Monte-Carlo simulation balancing revisited", *Computational Intelligence and Games (CIG) 2016 IEEE Conference on*, pp. 1-7, 2016, ISSN 2325-4289. Chaslot, Guillaume., Bakkes, Sander., Szita, Istvan., Spronck, Pieter. (2008). Monte-Carlo *Tree* Search: A new Framework for *Game* AI.

Whitehouse, Daniel., Cowling, Peter I., Powley. Edward J. (2013). Integrating *Monte Carlo Tree Search* with Knowledge-Based Methods to Create Engaging Play in a Commercial Mobile *Game*

Rabin, S. (2014). *Game AI Pro.* CRC Press.

Galvan-Lopez, E., Li, R., Patsakis, C., Clarke, S. dan Cahill, V., 2014. Heuristic-Based Multi-Agent Monte Carlo Tree Search. In Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference, Greece, 7 - 9 Juli 2014 on (pp. 177-182).

Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. dan Colton, S., 2012. A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games, 4(1), pp.1-43.

Buckland M., 2002. *AI techniques for game programming.* 1st ed. Premier press.