# IMPLEMENTASI MONTE CARLO TREE SEARCH PADA PERMAINAN KARTU "DAIFUGO"

Eunike Thirza Hanitya Christian[1]
22094703@students.ukdw.ac.id

R. Gunawan Santosa[2]
gunawan@ukdw.ac.id

Erick Purwanto[3]
erickp@ukdw.ac.id

***Abstract***

*Daifugo is climbing card game that is originated from Japan. AI player of Daifugo card game can be implemented using Monte Carlo Tree Search to get optimal result from random simulation. Monte Carlo Tree Search has 4 step, selection, expansion, simulation and backpropagation that is executed until maximal loop is reached. Objective of using Monte Carlo Tree Search on AI player in Daifugo card game is to get move with high winning rate and to observe the effect of number of loop on the method to winning rate*

**Keywords :** *card game, daifugo, AI, Monte Carlo Tree Search*

## 1. Pendahuluan

Permainan kartu merupakan salah satu permainan yang populer dan digemari oleh banyak orang. Ada banyak jenis dari permainan kartu, salah satunya Daifugo(大富豪) atau yang juga sering disebut *Grand Millionaire*. Daifugo adalah permainan kartu yang berasal dari negara Jepang yang menggunakan standard *deck* (52-*pack*). Tujuan dari permainan ini adalah untuk menghabiskan kartu yang di tangan secepat mungkin dengan memainkan kartu yang lebih kuat daripada kartu player lainnya secara terus menerus.

Kendala dari permainan kartu ini adalah permainan ini tidak dapat dilakukan sendiri atau dengan 2 orang. Dibutuhkan minimal 3 orang untuk memainkan permainan ini. Hal ini tentunya menjadi masalah jika tidak ada cukup pemain yang mengerti mengenai permainan kartu ini, tetapi hal itu dapat diatasi dengan pemanfaatan *AI player* pada *desktop game*.

Penerapan *AI* dalam *game programming* telah banyak dipakai untuk *strategy game*. Dengan algoritma yang tepat, sebuah *computer opponent* dapat mempunyai kecerdasan yang sebanding dengan pemain pro. Pada game Chess, Checkers, dan Scrabble, kemampuan *computer opponent* lebih baik daripada *best human player*. Pada Backgammon, kemampuan *computer opponent* setara dengan *top human player*. Pada Go, *computer opponent* dapat mengalahkan seorang professional pada papan 9x9 dengan 9 stone. Pada Bridge (*card game*), *computer opponent* mempunyai kemampuan rata-rata pemain kaliber, kemampuan di bawah *best human player*, tetapi mampu untuk memainkan banyak *bridge hand* yang rumit dengan sempurna.

Metode yang diterapkan pada penelitian ini adalah Monte Carlo Tree Search. Metode ini berfungsi untuk menemukan *option* yang optimal dari *domain* yang ada dengan menguji kombinasi yang dimungkinkan, dan membangun *tree* berdasarkan simulasi yang dilakukan. Metode ini dipilih karena pada permainan kartu Daifugo mempunyai banyak *rule*, kombinasi, dan trik yang menghasilkan beragam result. Dengan menggunakan MCTS, semua kombinasi dapat dites dengan simulasi untuk mendapatkan hasil yang terbaik.

[1] *Mahasiswa Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*
[2] *Dosen Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*
[3] *Dosen Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana*

# 2. Landasan Teori

## 2.1 Daifugo

Daifugo (大富豪) atau Daihinmin (大貧民) adalah permainan kartu yang berasal dari negara Jepang yang dimainkan oleh 3-6 pemain dengan standar *deck* 52 kartu. Tujuan dari game ini adalah menghabiskan semua kartu di tangan secepat mungkin dengan mengeluarkan kartu yang lebih besar daripada kartu milik lawan. Permainan ini mempunyai sedikit kesamaan dengan Big Two (China), Tien Len (Vietnam), dan Great Damulti.

Peraturan *dealing* pada permainan ini adalah dengan mengocok semua 52 kartu, dan membagi semua kartu sesuai dengan jumlah pemain searah jarum jam.

Permainan ini membutuhkan *trick* seperti pada Spade dan Bridge. Tetapi berbeda dari 2 permainan tersebut, tiap *trick* dapat melibatkan lebih dari 1 kartu dari tiap pemain, dan pemain tidak harus mengikuti *trick* tersebut.

Urutan dari giliran tiap pemain dimulai dari pemain yang mempunyai kartu 3 *of Diamond* (3 ♦ ). Setelah pemain tersebut mengeluarkan kartunya, pemain lain dapat mengeluarkan kartu dengan angka yang lebih tinggi. *Player* juga bisa tidak mengeluarkan kartu pada gilirannya. Tidak mengeluarkan kartu (*pass*) tidak membuat *player* kehilangan gilirannya pada putaran selanjutnya dalam *trick* yang sama. Kartu yang dikeluarkan dapat berupa kombinasi (misal *three-of-a-kind*). Tetapi jika satu *trick* dimulai dari *three-of-a-kind*, maka hanya set *three-of-a-kind* yang bisa dimainkan. *Trick* berakhir saat tidak ada pemain yang bisa mengalahkan set kartu terakhir, atau player mengeluarkan 2. Pemain dengan kartu/set tertinggi dapat memimpin *trick* selanjutnya. Urutan kartu dari nilai terkecil ke terbesar: 3-4-5-6-7-8-9-10-J-Q-K-A-2.

*Game* berakhir jika peringkat 1, 2, 3 dan 4 telah ditentukan. Peringkat didapat dari pemain mana yang tercepat menghabiskan kartu sesuai peraturan yang berlaku.

Strategi dasar dari Daifugo sangatlah sederhana, yaitu dengan menghabiskan kartu-kartu lemah terlebih dahulu sehingga hanya kartu dengan nilai tinggi yang tersisa di tangan pada akhir permainan. Jika *player* terjebak dengan kartu bernilai kecil, akan menjadi sulit untuk memainkan kartu tersebut dan menghabiskan kartu di tangan. Dengan memenangkan *trick*, pemain dapat memimpin *trick* berikutnya dan 1 kartu lemah dapat disimpan untuk dimainkan terakhir. Untuk mencegah pemain dengan kartu yang tersisa sedikit, dapat dilakukan teknik blok dengan memainkan kartu dengan nilai tinggi atau kombinasi sehingga pemain tersebut tidak mampu untuk mengalahkannya. Strategi tambahan dapat dilakukan dengan *skips* atau *clear*, yang memungkinkan *player* lain untuk mencegah *player* tertentu menghabiskan kartu di tangan.

Ada banyak variasi dan optional rule dalam Daifugo, seperti: *Deuce means clear, Match means clear, Jokers are 2s, Jokers are Wilds, Forbidden Last Cards, Skip and Multi-skips, Revolution (Kakumei), Completo, Jack-Back, Kaidan (sequence), Eight Enders (Hachigiri), Deuces Wild Joker High, Direction of Play, People Revolution, Multiple Decks, Despositism, Jokers(without/1/2), Suit Lock (Shibari), Poker Hands.*

Pada Tugas Akhir ini, optional rule yang diimplementasikan adalah:

- without Joker,

Joker tidak digunakan pada permainan.

- Deuce means clear,

Deuce Means Clear adalah peraturan dimana 2 merupakan nilai tertinggi. Sebuah kartu 2 dapat mengalahkan kombinasi dari kartu apapun. Karena tidak ada kartu yang bisa mengalahkan 2, pemegang kartu 2 dapat mencuri *trick* dari pemain lainnya. Strategi ini menjadi penting saat mencegah seorang pemain untuk menghabiskan kartu di tangan.

- Forbidden Last Card,

Forbidden Last Card adalah rule yang melarang pemain untuk mengeluarkan kartu tertentu pada akhir gilirannya. Misal 3, 8, atau 2.

- Skip and Multi-skips,

Skip adalah keadaan dimana seorang pemain dapat mengeskip giliran pemain lainnya jika dia mempunyai kartu dengan angka sama. Misal: memainkan satu kartu 7 dengan *rank* lebih tinggi di atas 7 akan mengeskip 1 pemain. *Multiskip* adalah jika *player* memainkan

lebih dari 1 kartu bernilai sama untuk mengeskip pemain sejumlah kartu tersebut dari giliran. Misal: memainkan 2 kartu 7 di atas 7 akan mengeskip 2 pemain, dst.

- Revolution(Kakumei),

Revolution (kakumei) adalah keadaan dimana urutan angka dan *rank* menjadi terbalik. 2 menjadi kartu dengan nilai terendah dan 3 menjadi kartu dengan nilai tertinggi. Counterrevolution (kakumei gaeshi) dapat mengembalikan 2 menjadi kartu tertinggi dan 3 menjadi kartu terendah. Revolution terjadi jika *a set of four* atau *sequence-of-four* dengan rank sama dimainkan secara bersamaan atau secara berurutan oleh beberapa pemain. Pemain yang memainkan set tersebut bisa memilih untuk melakukan revolution atau tidak.

- Sequence (Kaidan),

Sequence (kaidan) adalah peraturan dimana 3 atau lebih kartu yang berurutan dapat dimainkan secara bersamaan (misal: 5-6-7) Kartu tertinggi harus lebih tinggi dari kartu yang dimainkan pada kartu tertinggi dari set sebelumnya. *5-straight* merupakan batas tertinggi dari jumlah kartu yang dapat dimainkan. *Pair – straight* dapat dimainkan pada *single straight* (misal: 5-5-6-6-7-7).

- Eight Enders (Hachigiri),

Eight Enders (hachigiri) adalah peraturan dimana memainkan sebuah kartu 8, *pair* 8, atau *straight* yang dimulai dari 8, atau diakhiri dengan 8 dapat mengakhiri sebuah *trick*, tetapi harus sesuai dengan *pattern* (misal: memainkan *pair* 8 di atas *pair* 5). Pemain dengan kartu 8 dapat memimpin *trick* berikutnya.

- Poker Hands

Poker Hands adalah kartu dengan kombinasi *straight – flush – full house – straight flush* dapat dimainkan pada *trick* 5 kartu. Tiap set harus mengikuti *rule* dari poker.

- Suit Lock (Shibari).

Suit lock (shibari) adalah keadaan dimana jika ada 2 *player* memainkan kartu dengan *rank* sama, maka pemain selanjutnya hanya bisa mengalahkan kartu tersebut dengan kartu dari *rank* tertentu sampai *trick* selesai. Misal: 5 *of Diamond* (5♦) < 7 *of Diamond* (7♦), maka *player* berikutnya hanya boleh memainkan kartu dengan nilai lebih tinggi pada *rank diamond* (♦). Sedangkan *partial lock* (*kata-shibari*) dapat terjadi pada *pair*. Misal 4 *of Heart* (♥) – 4 *of Spade* (♠) < 6 *of Club* (♣) – 6 *of Spade* (♠), maka pemain berikutnya hanya dapat memainkan *pair* yang mempunya kartu dengan nilai *Spade* (♠) di dalamnya. *Partial lock* bisa berubah menjadi *Full lock* jika pemain berikutnya mengeluarkan kartu dengan *rank* sama dengan *rank* yang belum dikunci. Misal pemain mengeluarkan 9 *Club* (♣) – 9 *Spade* (♠), maka *club* yang sebelumnya tidak di *lock* akan menjadi di *lock*. Pada giliran pemain berikutnya, kartu *pair* yang dapat dikeluarkan harus merupakan pasangan *Club* (♣) – *Spade* (♠) dengan nilai yang lebih tinggi.

## 2.2 Monte Carlo

Metode Monte Carlo memberikan perkiraan solusi untuk berbagai persoalan matematika dengan melakukan eksperimen *sampling* statistik. Monte Carlo dapat didefinisikan sebagai *statistical simulation method*, dimana simulasi statistik didefinisikan pada hal umum berdasarkan metode yang memanfaatkan urutan nomer acak untuk melakukan simulasi. Proses pemecahan masalah dengan metode ini membutuhkan banyak simulasi dari angka acak dan probabilitas. Metode Monte Carlo hanya dapat memberikan perkiraan dari jawaban, dengan demikian analisa perkiraan *error* merupakan faktor utama yang perlu diperhatikan dalam menganalisa metode ini. Jenis variasi dari metode Monte Carlo menghasilkan tingkat keakuratan yang berbeda.

## 2.3 Tree

Pada programming, *tree* digunakan untuk menunjukkan sebuah hirarki. *Tree* merupakan struktur data rekursif yang terdiri dari kumpulan *node* dihubungkan oleh *edge*. *Tree* terdiri dari *root* (*node* yang tidak mempunya *parent*) dan kumpulan dari *child node*. *Leaf* adalah *node* yang mempunyai *parent*, tetapi tidak mempunyai *child node*. *Internal node* adalah *node* yang mempunyai *parent* dan *child node*.

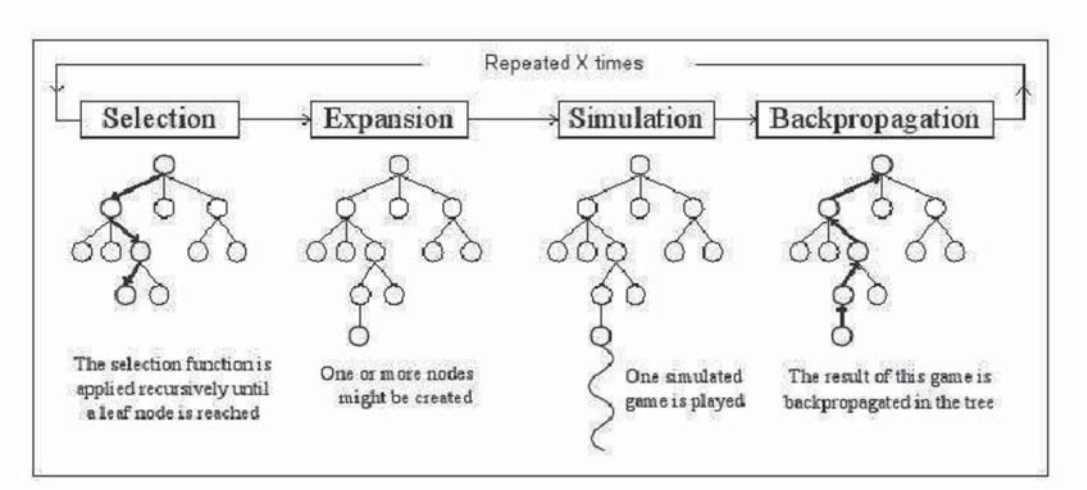
## 2.4 Monte Carlo Tree Search

Monte Carlo Tree Search (MTCS) mempunyai 2 konsep dasar, yaitu: *true value* dari *action* diperkirakan berdasarkan *random simulation*; dan *value* ini digunakan secara efisien untuk mendapatkan sebuah *best first strategy*. Algoritma secara progresif membangun *game tree partial*, dibimbing oleh hasil dari eksplorasi sebelumnya terhadap *tree* tersebut. *Tree* digunakan untuk mengestimasi nilai dari *move*, dengan estimasi ini (terutama untuk *move* yang paling menjanjikan), menjadi semakin akurat.

Algoritma dasar melibatkan pembangunan *search tree* secara *iterative* hingga beberapa *predefined computational budget* (*time, memory*, atau *interation constraint*) tercapai, dimana search diberhentikan dan *best performing root action* dikembalikan. Tiap *node* di *search tree* merepresentasikan *state* dari *domain* dan *directed link* ke *child node* dan merepresentasikan *action* ke *subsequent state*.

4 step yang diaplikasikan tiap iterasi: (sumber: IEEE Transactions on Computational Intelligence and AI Games, Vol. 4, No. 1, March 2012)

1. Selection: dimulai dari sebuah *node* di *root*, *policy chid selection* secara rekursiv diaplikasikan pada turunan *tree* sampai *node* terpenting dari *expandable node* tercapai. Sebuah *node* dikatakan *expandable* jika merepresentasikan *non-terminal state* dan punya *children* yang belum dikunjungi.
2. Expansion: satu atau lebih *node child* yang ditambahkan untuk mengexpand *tree* berdasarkan action yang tersedia.
3. Simulation: sebuah simulasi berjalan dari *new node* sesuai dengan *default policy* untuk menghasilkan sebuah *outcome*.
4. Backpropagation: hasil dari simulasi di "back-up" pada *node* yang dipilih untuk mengupdate statistik.




*Gambar 1*. Monte Carlo Tree Search

Pemilihan *best move* pada giliran dapat dilihat sebagai proses *learning* pada suatu *value* dan penentuan *move* dengan *best value*. 3 elemen dari *learning process*:

- *Online learning* yang merupakan MCTS itu sendiri melakukan pendekatan distribusi dari value dari *move* dari simulasi Monte Carlo pada *tree* asimetris
- *Offline learning* merepresentasikan *domain knowledge* pada *node* atau *playout*.
- *Transient learning correspondent* pada *applying value* dipelajari pada *particular posititon* pada posisi yang mirip

Penerapan MCTS dalam penelitian ini adalah pada pengambilan keputusan *AI* pada setiap putaran. Pada setiap giliran *AI* untuk bermain, setiap *possible move* akan dikelompokkan menurut kategorinya, dan dari tiap kategori, dicari kemungkinan *move*nya (pembangunan *tree*). Suatu *node* akan dipilih secara *random* dan apabila kartu merupakan kepala (tidak bisa ditandingi oleh *player* lainnya atau merupakan awal dari putaran), *node* tersebut diberi tanda. Untuk memudah proses pembangunan *tree* lebih lanjut pada simulasi berikutnya. Lakukan simulasi berulang kali sampai semua *node* dieksplore, kemudian pilih

*node* dengan hasil (*win*) tertinggi dan *time* (dihitung dari banyaknya putaran sampai *AI* menyelesaikan permainan) terkecil. Pada simulasi untuk mendapatkan *move* terbaik, langkah dari musuh *AI* ditentukan secara berdasarkan *knowledge base rule* dari *game*.

Pada simulasi, kartu player lain ditentukan sebagai berikut: 52 kartu – kartu *player* – jumlah kartu yang sudah keluar. Lalu dibagi secara *random* dan *player* lain melakukan *move* berdasarkan kombinasi terkuat yang memungkinkan berdasarkan peraturan dari *game* Daifugo. Misalnya: *Straight* < *Flush* < *Full House* < *Straight Flush*.

Untuk menentukan move yang akan dilakukan, menggunakan formula :
Q(s) = (rank*100) + ronde
s : nilai dari *state* yang dipilih yang juga digunakan sebagai *label* dalam tabel perhitungan;
Q(s) : hasil yang diperoleh pada label s;
Rank : peringkat yang diperoleh setelah simulasi dilakukan;
ronde : jumlah putaran yang dibutuhkan untuk menghabiskan kartu
*Move* yang dipilih adalah *move* yang mempunyai Q(s) terkecil

## 3. Hasil dan Pembahasan

### 3.1. Implementasi Sistem

AI player yang dibangun pada game ini menggunakan metode Monte Carlo Tree Search, dengan proses *selection, expansion, simulation,* dan *backpropagation*. Berikut merupakan *pseudocode* dari implementasi MCTS pada AI player.

```
copy player_card to mctsCard
if trickLeader = indexPlayer
    generateMCTS_l()
else
    generateMCTS_f()

count possible trick combination
if possible trick combination = 0
    return ""
else
    //---selection---//
    random random1
    random random2
    //---expansion---//
    play mctsTree[random1][random2]
    //---simulation---//
    if totalLoop < loopMCTS
        simulation()
        //---backpropagation---//
        updateHighScore()
    return mctsTree[highestNode1][highestNode2]
```

*Gambar 2*. Pseudocode dari implementasi AI player menggunakan metode MCTS

generateMCTS_l() merupakan fungsi yang dipanggil untuk membangun *tree* berdasarkan kombinasi yang mungkin dimainkan jika *AI player* tersebut merupakan *trick leader*. generateMCTS_f() merupakan fungsi yang dipanggil untuk membangun *tree* berdasarkan kombinasi yang mungkin dimainkan jika *AI player* tersebut merupakan *trick follower*. simulation() merupakan fungsi yang dimainkan untuk menjalankan simulasi *move* player lainnya apabila AI player memainkan kombinasi kartu pada mctsTree[random1][random2] sampai didapatkan nilai dari *move* tersebut. updateHighScore() merupakan fungsi yang dijalankan setelah didapatkan nilai dari suatu *move*.

Simulasi permainan pada proses MCTS mengikuti peraturan dari permainan. Pemilihan *move* pada simulasi dilakukan dengan mencari kombinasi yang mungkin dimainkan dengan nilai seminimal mungkin, serta jenis kombinasi yang mempunyai kemungkinan untuk didapat lebih kecil.

### 3.2. Analisis Sistem

Pengujian pertama merupakan pengetesan kinerja *AI* dalam melawan *human player*. Pengujian ini dilakukan untuk membandingkan performa *AI player* yang diimplementasi dengan metode Monte Carlo Tree Search pada 2 *level* berbeda (*easy dan normal*). *Level easy* mempunyai jumlah perulangan 10 dan *level normal* memiliki jumlah perulangan 100. Berikut merupakan tabel dari hasil permainan *AI vs AI vs AI vs human player* selama 20 kali permainan.

Tabel 1.
Pengujian AI melawan pemain pada level *easy*

| n | *North (AI)* | *East (AI)* | *South (player)* | *West (AI)* |
|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 |
| 2 | 3 | 2 | 1 | 4 |
| 3 | 1 | 2 | 4 | 3 |
| 4 | 3 | 4 | 1 | 2 |
| 5 | 4 | 3 | 1 | 2 |
| 6 | 4 | 3 | 1 | 2 |
| 7 | 2 | 1 | 3 | 4 |
| 8 | 4 | 1 | 3 | 2 |
| 9 | 3 | 4 | 2 | 1 |
| 10 | 2 | 3 | 1 | 4 |
| 11 | 1 | 2 | 4 | 3 |
| 12 | 4 | 1 | 2 | 3 |
| 13 | 4 | 2 | 3 | 1 |
| 14 | 4 | 1 | 3 | 2 |
| 15 | 4 | 1 | 2 | 3 |
| 16 | 3 | 2 | 1 | 4 |
| 17 | 4 | 3 | 2 | 1 |
| 18 | 3 | 1 | 2 | 4 |
| 19 | 4 | 2 | 3 | 1 |
| 20 | 4 | 3 | 1 | 2 |

Tabel 2.
Pengujian AI melawan pemain pada level *easy*

| *n* | *North (AI)* | *East (AI)* | *South (player)* | *West (AI)* |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 |
| 2 | 4 | 2 | 3 | 1 |
| 3 | 1 | 3 | 2 | 4 |
| 4 | 4 | 1 | 2 | 3 |
| 5 | 2 | 1 | 3 | 4 |
| 6 | 1 | 4 | 3 | 2 |
| 7 | 1 | 3 | 4 | 2 |
| 8 | 3 | 1 | 2 | 4 |
| 9 | 4 | 3 | 2 | 1 |
| 10 | 4 | 2 | 1 | 3 |
| 11 | 1 | 2 | 3 | 4 |
| 12 | 3 | 4 | 3 | 2 |
| 13 | 1 | 4 | 3 | 2 |
| 14 | 4 | 1 | 3 | 2 |
| 15 | 3 | 4 | 2 | 1 |
| 16 | 2 | 3 | 1 | 4 |
| 17 | 2 | 3 | 4 | 1 |
| 18 | 3 | 4 | 1 | 2 |
| 19 | 1 | 3 | 4 | 2 |
| 20 | 1 | 2 | 3 | 4 |

Tabel 3.
Persentase rank tiap pemain yang didapat pada akhir permainan level easy

| Rank | North (AI) | East (AI) | South (player) | West (AI) |
|---|---|---|---|---|
| 1 | 15% | 30% | 35% | 20% |
| 2 | 10% | 35% | 25% | 30% |
| 3 | 25% | 25% | 30% | 20% |
| 4 | 50% | 10% | 10% | 30% |

Tabel 4.
Persentase rank tiap pemain yang didapat pada akhir permainan level normal

| Rank | North (AI) | East (AI) | South (player) | West (AI) |
|---|---|---|---|---|
| 1 | 35% | 25% | 20% | 20% |
| 2 | 20% | 20% | 25% | 35% |
| 3 | 20% | 30% | 40% | 10% |
| 4 | 25% | 25% | 15% | 35% |

Tabel 5.
Kesimpulan dari penelitian AI melawan pemain

| Easy | Normal |
|---|---|
| * Persentase total AI player mendapatkan rank 1: 65%<br>* Persentase maksimal mendapatkan rank 1 per AI player: 30%<br>* Persentase minimal mendapatkan rank 1 per AI player: 15% | * Persentase total AI player mendapatkan rank 1: 80%<br>* Persentase maksimal mendapatkan rank 1 per AI player: 35%<br>* Persentase minimal mendapatkan rank 1 per AI player: 20% |
| * Persentase total AI player mendapatkan rank 4: 90%<br>* Persentase maksimal mendapatkan rank 4 per AI player: 50%<br>* Persentase minimal mendapatkan rank 4 per AI player: 15% | * Persentase total AI player mendapatkan rank 4: 85%<br>* Persentase maksimal mendapatkan rank 4 per AI player: 35%<br>* Persentase minimal mendapatkan rank 4 per AI player: 25% |
| *Persentase player mendapatkan rank 1: 35%<br>*Persentase player mendapatkan rank 4: 10% | *Persentase player mendapatkan rank 1: 20%<br>*Persentase player mendapatkan rank 4: 15% |

Tahap pengujian ke-dua merupakan *AI* melawan *AI*. Pengujian ini bertujuan untuk mengetahui pengaruh jumlah perulangan yang dilakukan pada metode Monte Carlo Tree Search terhadap performa *AI player*. Pengujian ini dilakukan dengan menggunakan 4 set kartu yang akan diujikan terhadap 4 *AI player* dengan jumlah perulangan yang berbeda (5, 25, 50, dan 100).

Tabel 6.
Set kartu yang akan diujikan pada 4 AI player

| Set | Kartu |
|---|---|
| 1 | 3♦, 3♠, 4♠, 5♣, 6♦, 7♦, 8♣, 10♦, 10♣, J♠, Q♦, 2♣, 2♥ |
| 2 | 4♣, 5♦, 6♠, 8♦, 9♣, 9♥, 10♠, Q♠, K♦, K♠, A♦, A♣, 2♦ |
| 3 | 4♦, 5♠, 6♣, 7♠, 8♠, 9♣, 9♠, J♦, J♣, Q♣, A♥, A♠, 2♠ |
| 4 | 3♣, 3♥, 4♥, 5♥, 6♥, 7♣, 7♥, 8♥, 10♥, J♥, Q♥, K♣, K♥ |

Tabel 7.
Rencana pengujian

| Test | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | Set-1 | Set-2 | Set-3 | Set-4 |
| 2 | Set-2 | Set-3 | Set-4 | Set-1 |
| 3 | Set-3 | Set-4 | Set-1 | Set-2 |
| 4 | Set-4 | Set-1 | Set-2 | Set-3 |

Tabel 8.
Persentase rank dari pengujian Tes-1

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 11% | 33% | 15,3% | 40,7% |
| 2 | 23,6% | 21,3% | 35,7% | 19,4% |
| 3 | 32,8% | 23,8% | 26,4% | 17% |
| 4 | 32,6% | 21,9% | 22,6% | 22,9% |

Tabel 9.
Persentase rank dari pengujian Tes-2

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 10,9% | 10,2 | 45,5% | 33,4% |
| 2 | 21% | 29,9 | 18,4% | 30,7% |
| 3 | 31,5% | 35,3% | 14,4% | 18,8% |
| 4 | 36,6% | 24,6% | 21,7% | 17,1% |

Tabel 10.
Persentase rank dari pengujian Tes-3

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 5,8% | 41,8% | 28,9% | 23,5% |
| 2 | 21,7% | 17,4% | 33,7% | 27,2% |
| 3 | 36,3% | 16,3% | 17,8% | 29,6% |
| 4 | 36,2% | 24,5% | 19,6% | 19,7% |

Tabel 11.
Persentase rank dari pengujian Tes-4

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 24,5% | 24,5% | 31,1% | 19,9% |
| 2 | 21,3% | 26,7% | 23,2% | 28,8% |
| 3 | 23,4% | 24,5% | 24,7% | 27,4% |
| 4 | 30,8% | 24,3% | 21% | 23,9% |

Tabel 12.
Persentase rank berdasarkan set kartu

| Rank | Set-1 | Set-2 | Set-3 | Set-4 |
|---|---|---|---|---|
| 1 | 24,45% | 24,625% | 12,8% | 38,125% |
| 2 | 28,675% | 23,175% | 29,025% | 19,125% |
| 3 | 23,475% | 27,4% | 31,35% | 17,775% |
| 4 | 22,95% | 24,8% | 26,825% | 24,975% |

Tabel 13.
Persentase rank AI pada set kartu-1

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 11% | 24,5% | 28,9% | 33,4% |
| 2 | 23,6% | 26,7% | 33,7% | 30,7% |
| 3 | 32,8% | 24,5% | 17,8% | 18,8% |
| 4 | 32,6% | 24,3% | 17,8% | 17,1% |

Tabel 14.
Persentase rank AI pada set kartu-2

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 10,9% | 33% | 31,1% | 23,5% |
| 2 | 21% | 21,3% | 23,2% | 27,2% |
| 3 | 31,5% | 23,8% | 24,7% | 29,6% |
| 4 | 36,6% | 21,9% | 21% | 19,7% |

Tabel 15.
Persentase rank AI pada set kartu-3

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 5,8% | 10,2% | 15,3% | 19,9% |
| 2 | 21,7% | 29,9% | 35,7% | 28,8% |
| 3 | 36,3% | 35,3% | 26,4% | 27,4% |
| 4 | 36,2% | 24,6% | 22,6% | 23,9% |

Tabel 16.
Persentase rank AI pada set kartu-4

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 24,5% | 41,8% | 45,5% | 40,7% |
| 2 | 21,3% | 17,4% | 18,4% | 19,4% |
| 3 | 23,4% | 16,3% | 14,4% | 17% |
| 4 | 30,8% | 24,5% | 21,7% | 22,9% |

Tabel 17.
Persentase rank AI

| Rank | AI(loop=5) | AI(loop=25) | AI(loop=50) | AI(loop=100) |
|---|---|---|---|---|
| 1 | 13,05% | 27,375% | 30,2% | 29,375% |
| 2 | 21,9% | 23,835% | 27,75% | 26,525% |
| 3 | 31% | 24,975% | 20,825% | 23,2% |
| 4 | 34,05% | 23,825% | 21,225% | 20,9% |

## 4. Kesimpulan

Berdasarkan hasil analisis dan implementasi sistem, maka diperoleh kesimpulan sebagai berikut :

1. Dari analisa hasil dari pengujian *AI player* melawan *AI player*, maka dapat disimpulkan semakin besar jumlah perulangan / *loop*, semakin optimal performa *AI*. *Loop optimal* untuk *AI* daifugo adalah antara 50 sampai 100. Faktor lain yang mempengaruhi persentase kemenangan *AI* adalah *luck* (kartu *player*) dan *move player* yang dapat mengubah strategi *AI* (*skip* dan *multi-skip*, *revolution* dan *counter revolution*, *lock*).
2. Dari analisa hasil dari pengujian *AI player* melawan *human player*, maka dapat disimpulkan pada *level easy* persentase kemenangan *AI* melawan *human player* mencapai 65%, sedangkan pada *level normal* persentase kemenangan *AI* melawan *human player* mencapai 80%. Persentase *human player* mendapatkan *rank* terendah pada *level easy* adalah sebesar 10%, dan persentase *human player* mendapatkan *rank* terendah pada *level normal* adalah sebesar 15%.
3. Kelemahan dari *AI player* yang dibangun terletak pada bagian simulasi dimana tidak adanya strategi untuk mengatasi kemungkinan *forbidden last card*, yaitu memainkan kartu yang dilarang untuk dimainkan sebagai kartu terakhir dari *player* yang menyebabkan *player* mendapatkan rank terendah, tidak adanya strategi untuk tidak memainkan kartu pada saat tertentu (menunggu kartu tertinggi dimainkan, menyimpan kartu untuk kombinasi yang lebih menguntungkan), tidak adanya strategi untuk mencegah terjadinya revolusi/*counter*-revolusi yang merugikan player tersebut

**Daftar Pustaka**

Adams, Ernest. 2010. *Fundamentals of Game Design*. United States of America: Pearson.

Bounzy, Bruno. *MCTS Experiments on the Voronoi Games.* Lipade Universite Paris Descartes, FRANCE

Browne, Cameron. *A Survey of Monte Carlo Tree Search Methods.* IEEE:2012

Chaslot, Guillaume. Bakkes, Sanders. Szita, Istvan. Spronck, Pieter. *Monte Carlo Tree Search: A New Framework For Game AI.* Universiteit Maastricht.

Gerrison,G.E.H. 2010. *Combining Monte Carlo Tree Search and Opponent Modeling in Poker*. Maastricht University

Kozelek, Tomas. 2009. *Method of MCTS and the Game Arimaa*. Charles University in Prague Faculty of Mathematics and Physics: 2009

Kleij, A. A. J. van der. 2010. *Monte Carlo Tree Search and Opponent Modeling Through Player Clustering in No-Limit Texas Hold'em Poker*. Netherlands: University of Groningen

Whitehouse, Daniel. Powley, Edward J. Cowling, Peter I. *Determinization and Information Set Monte Carlo Tree Search for the Card Game Dou Di Zhu*.

須藤, 郁弥. 篠原, 歩. モンテカルロ法を用いたコンピュータ大貧民の思考ルーチン設計. 東北大学大学院 情報科学研究科

須藤, 郁弥. 成澤, 和志. 篠原, 歩. UECコンピュータ大貧民大会向けクライアント「snowl」の開発. 東北大学 大学院情報科学研究科