PENDEKATAN ALGORITMA *GREEDY* DALAM PERMAINAN *POKEMON TRADING CARD GAME*

Ramon Rusli – NIM : 13506025

Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha no 10 Bandung
e-mail: if16025@students.if.itb.ac.id , mon_nie@yahoo.com , monmon_justice@yahoo.co.id

ABSTRAK

Makalah ini membahas mengenai studi dan pendekatan algoritma *Greedy* dalam meningkatkan performansi permainan seorang *player* dalam *attack phase* permainan *Pokemon Trading Card Game (Pokemon TCG)*. Dalam makalah ini, *Cardlist* yang digunakan adalah *Trainer Base Set* yang memiliki kartu-kartu dengan fungsi dasar dalam permainan *Pokemon TCG,* sehingga diharapkan dapat mempesempit penjelasan mengenai implementasi algoritma *Greedy* di dalam permainan *Pokemon TCG*. Selain itu, penyempitan ruang lingkup juga dilakukan untuk menghindari hilangnya arah tujuan utama pembuatan makalah ini. Dalam hal ini, ada banyak sekali hal yang dapat mempengaruhi performansi permainan untuk meraih kemenangan *player*, seperti *Pokemon Card* mana yang sebaiknya dijadikan *active Pokemon*, di mana sebaiknya *energy Card* diletakkan, kapan saat yang tepat untuk mengevolusi *Pokemon*. serangan apa yang sebaiknya dilakukan pada suatu *attack phase*, dan lainnya. Hal-hal tersebut dipengaruhi oleh pola pikir dan strategi *player* untuk mencapai kemenangan. Pada makalah ini, permasalahan menentukan *attack* yang dipilih dalam *attack phase* akan dibahas dalam makalah ini, dengan pendekatan algoritma *Greedy*.

Kata kunci: *Greedy*, *Pokemon Trading Card Game* (*Pokemon TCG*), *Trainer Base Set*, *attack phase*, *active Pokemon*, *Pokemon Card*, *Trainer Card*, *energy Card*, *player*.

1. PENDAHULUAN

Permainan *Pokemon Trading Card Game* (*Pokemon TCG*) merupakan salah satu permainan kartu yang dipusatkan pada konsep *Pokemon* Battle. Semua kartu *Pokemon* memiliki Hit Points dan *Attack* yang dapat menerima tambahan *energy Card* dan *Trainer Card*. Secara garis besar, permainan ini berlangsung secara bergantian dari *player* yang satu dengan yang lainnya. Penyerangan dilakukan tiap giliran *player* (*turn*) satu kali saja, dan dilakukan oleh *active Pokemon player* terhadap *Pokemon* lawan. *Player* dianggap mengalahkan *active Pokemon* lawan jika *Hit Point* (HP) dari *active Pokemon* lawan sudah habis dan mengirimkan *Pokemon* tersebut ke dalam *discard* pile. Jika itu terjadi, lawan mengganti *Pokemon*nya dengan *Pokemon* lainnya (*benched Pokemon*) dan kembali bertarung lagi. Kemenangan bisa didapat *player* dengan 3 cara, jika ia dapat mendapatkan *prize* sebanyak 6 kali, jika *player* lawan tidak dapat mengambil kartu lagi dari *library*, atau jika *player* lawan tidak memiliki *benched Pokemon* saat *active Pokemon* miliknya kalah.

Pendekatan yang digunakan di dalam algoritma *Greedy* adalah membuat pilihan yang memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (*local optimum*) pada *Set*iap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global (*global optimum*). Secara umum algoritma *Greedy* disusun oleh elemen-elemen berikut :

- Himpunan kandidat.
- Himpunan solusi
- Fungsi seleksi (*selection function*)
- Fungsi kelayakan (*feasible*)

Algoritma *Greedy* dapat menyelesaikan beberapa masalah dalam kehidupan nyata, dan persoalan yang akan dibahas dalam makalah ini adalah pendekatan algoritma *Greedy* dalam *attack phase* permainan *Pokemon TCG*

2. METODE

2.1 Definisi Algoritma *Greedy*

Algoritma *Greedy* adalah strategi yang memecahkan masalah langkah demi langkah, pada *Set*iap langkah, adapun urutan langkah algoritma *Greedy* adalah :
1. mengambil pilihan yang terbaik yang dapat diperoleh saat itu,
2. berharap bahwa dengan memilih solusi optimum lokal, pada *Set*iap langkah akan mencapai solusi optimum global. Algoritma *Greedy* mengasumsikan bahwa

optimum lokal merupakan bagian dari optimum global. optimum global.

2.2 Skema Umum Algoritma *Greedy*

Persoalan optimasi dalam konteks algoritma *Greedy* disusun oleh elemen-elemen sebagai berikut:

1. Himpunan kandidat, *C*.
   Himpunan ini berisi elemen-elemen pembentuk solusi. Pada *Set*iap langkah, satu buah kandidat diambil dari himpunannya.
2. Himpunan solusi, *S*.
   Merupakan himpunan dari kandidat-kandidat yang terpilih sebagai solusi persoalan. Himpunan solusi adalah himpunan bagian dari himpunan kandidat.
3. Fungsi seleksi – dinyatakan sebagai predikat SELEKSI
   Merupakan fungsi yang pada *Set*iap langkah memilih kandidat yang paling mungkin untuk mendapatkan solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.
4. Fungsi kelayakan (*feasible*) – dinyatakan dengan predikat LAYAK
   Merupakan fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar aturan yang ada.
5. Fungsi obyektif
   Merupakan fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Dalam strategi ini, kita berharap optimum global merupakan solusi optimum dari persoalan. Namun, adakalanya optimum global belum tentu merupakan solusi optimum (terbaik), tetapi dapat merupakan solusi sub-optimum atau pseudo-optimum. Hal ini dapat dijelaskan dari dua factor berikut:

1. algoritma *Greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada.
2. pemilihan fungsi SELEKSI: fungsi SELEKSI biasanya didasarkan pada fungsi obyektif (fungsi SELEKSI bisa saja identik dengan fungsi obyektif). Jika fungsi SELEKSI tidak identik dengan fungsi obyektif, kita harus memilih fungsi yang tepat untuk menghasilkan nilai yang optimum. Karena itu, pada sebagian masalah algoritma *Greedy* tidak selalu berhasil memberikan solusi yang benar-benar optimum. Tetapi, algoritma *Greedy* pasti memberikan solusi yang mendekati (*approximation*) nilai optimum.

2.3. Penjelasan Permainan *Pokemon TCG*

Ada 5 macam tipe kartu dalam list *Base Set Pokemon TCG*, yaitu *Pokemon Card*s, *Energy Card*s, *Trainer Card*s, *Stadium Card*s, dan *Supporter Card*s. Dalam membuat *deck Pokemon*, 1 *Deck* yang minimal terdiri dari 60 kartu hanya boleh memiliki maksimal 4 kartu bernama sama di dalamnya untuk tipe kartu apapun, kecuali *energy Card*s yang boleh dimasukkan berapapun jumlahnya ke dalam *deck*. Dalam makalah ini, yang dibahas lebih lanjut dibatasi dengan *Pokemon Card*s dan *Energy Card*s saja.

*Pokemon Card*s: *Pokemon Card*s adalah basis dari permainan ini, yang dapat digunakan untuk menyerang, yang dibagi menjadi 3 tipe *Pokemon*: *basic Pokemon*, *stage*1 *Pokemon*, dan *stage*2 *Pokemon*. Saat *stage*1 *Pokemon* menimpa *basic Pokemon*, atau *stage* 2 *Pokemon* menimpa *stage*1 *Pokemon*, peristiwa tersebut disebut *evolution*. Untuk *Pokemon Card*s dengan nama yang berbeda, akan terdapat perbedaan-perbedaan elemen di dalamnya, seperti *attack*, Hit Points (HP), *weakness*, *resistance*, dan lain-lain.
*Pokemon Card*s memiliki *energy type* masing-masing, dan untuk tiap *attack*nya membutuhkan *energy* yang spesifik dengan *attack effect* beragam dan *damage* yang berbeda-beda untuk nama *Pokemon Card*s yang berbeda, Beberapa *Pokemon Card*s memiliki *Weakness* atau *resistance* terhadap suatu *type energy*. Jika suatu *Pokemon Card*s mendapatkan *damage counter* dari serangan dengan *energy type* yang sama dengan *weakness*nya, maka *damage counter* yang diterima akan menjadi 2 kali lipat, Namun, jika *energy type* dari serangan tersebut sama dengan *resistance*-nya, *damage counter* yang diterima dikurangi 30 poin.

*Energy Card*s: *energy Card*s adalah kartu yang ditambahkan ke dalam sebuah kartu *Pokemon* untuk memungkinkan *Pokemon* melakukan aksinya 1 *energy Card*s dalam *Trainer Base Set* menghasilkan 1 *type energy* untuk *Pokemon* yang kartu tersebut tambahkan. Tiap *attack* dari sebuah *Pokemon Card*s dipastikan membutuhkan minimal 1 *energy Card*s yang ditambahkan pada *Pokemon* tersebut. *Attack-attack* dari *Pokemon Card*s membutuhkan jenis-jenis *energy* yang spesifik untuk bisa digunakan. *Energy Card*s memiliki 6 elemen, yaitu fire, water, lightning, grass, physic, fighting, yang menghasilkan *energy type* yang sesuai dengan namanya masing-masing. *Attack* yang membutuhkan *energy* colorless dapat menggunakan *energy* manapun asalkan jumlahnya tetap sama.

Field , yaitu arena permainan dari tiap-tiap *player*, terdiri dari:

- hand : tempat kartu pemain yang dapat digunakan disimpan, maksimum 8 kartu
- *Pokemon field*: tempat *Pokemon* dan *energy Card*, digolongkan menjadi 2 jenis :
    - *active Pokemon*: *Pokemon* yang dapat memberikan *damage* pada *active Pokemon* lawan, namun dapat menerima *damage* dari *active Pokemon* lawan. Hanya boleh ada 1 *active Pokemon* dimajukan oleh tiap *player* dalam permainan. Namun, *active Pokemon* dapat diganti dengan membuang *energy Card* pada *active Pokemon* itu sebanyak jumlah *retreat cost Pokemon* itu.
    - *Benched Pokemon*: *Pokemon* yang berfungsi sebagai 'pemain cadangan', yang dapat dipindahkan menjadi *active Pokemon* jika *active Pokemon retreat* atau kalah. Maksimal hanya 5 *Pokemon* saja yang dapat menjadi *benched Pokemon*
- *library*: tempat mengambil(*draw*) kartu
- *discardpile*: tempat menaruh kartu yang terbuang
- *prize*: tempat menaruh *prize*

2.4. Alur Permainan *Pokemon TCG*

Permainan dimulai dengan mengacak pemain mana yang bergerak duluan. *player* yang saat ini sedang melakukan *turn* disebut *player*1, *player* lainnya disebut *player*2. *Set*elahnya *player*1 dan *player*2 masing-masing mengocok *deck* yang terdiri dari 60 kartu dan mengambil 7 kartu teratas, simpan *deck*. Lalu *player*1 dan *player*2 menaruh 1 *Pokemon* dengan *stage basic* sebagai *active Pokemon* dan boleh menaruh *Pokemon* dengan *stage basic* lainnya sebagai *benched Poke````mon*. *Player*1 dan *player*2 memiliki 6 *prize* di awal permainan

*Set*elahnya permainan berlangsung dalam *turn*, yang dilakukan bergilir antara *player*1 dan *player*2. Dalam tiap *turn* yang tiap *player* dapatkan, terdapat 2 *phase*, yaitu:

- *preparing phase*, yang terdiri dari: *draw* 1 *Card*, evolve *Pokemon*, menambahkan 1 *energy Card* ke *active* / *benched Pokemon Card* jika ada, dan *retreat* (maksimum 1 kali)
- *attack phase*, yang terdiri dari: pemilihan jenis *attack*, perhitungan *damage*, penambahan *damage counter*, pengambilan *prize* dan pergantian *active Pokemon* lawan jika kalah.

Berikut adalah alur dari *turn player*1 sebagai penjelasan dari tahap-tahap *preparing phase* dan *attack phase*. Penjelasan mengenai alur ini dipersempit dengan tidak memasukkan penggunaan dari *Trainer Card*s, *supporter Card*s, *stadium Card*s, dan *poke-power*.

*−−−−−TURN player*1 -----
-----*PREPARING PHASE*-----
1) Jika bukan *turn* pertama, *draw* 1 kartu
2) taruh *Pokemon* dengan *stage basic* sebagai *benched Pokemon*
3) evolusi *Pokemon*, dengan *Pokemon* hasil evolusi memiliki minus hp yang sama dengan *Pokemon* sebelumnya, dan *energy*store yang sama dengan sebelumnya.
4) tambahkan maksimal *energyCard* ke dalam 1 *Pokemon player*1
5) *retreat* maksimal 1 *Pokemon* dari *active* menuju bench dan memindahkan *benched Pokemon* menjadi *active Pokemon*

-----*ATTACK PHASE*---
1) pilih *attack* pada *active Pokemon* yang akan digunakan untuk menambahkan minus hp *Pokemon player*2, gunakan yang memiliki mana yang cukup dengan mana yang dimiliki *active Pokemon*
2) cek efek *attack* yang dipilih
3) hitung dengan *resistance* dan juga *weakness player*2, lalu tambahkan minus HP *active Pokemon player*2 dengan *damage* yang telah dikalkulasi
4) jika hp *active Pokemon player*2 lebih kecil sama dengan minus HP *active Pokemon* tersebut, taruh kartu *active Pokemon* tersebut ke dalam *discard* pile, dan *player*2 memilih *benched Pokemon* yang akan dijadikan *active Pokemon*

5a) jika tidak ada *benched Pokemon*, *player*2 kalah,
5b) jika tidak, ambil *prize* dan taruh ke dalam hand
6) jika *prize* sudah habis, maka *player*2 kalah
7) jika *player*2 kalah, permainan berakhir dengan kemenangan *player*1
8) jika *player*2 tidak kalah, *turn* berpindah ke *player*2. Status *player*1 berpindah ke *player*2 dan sebaliknya

-----*END TURN player1*-----

Dalam pseudo code, alur keseluruhan permainan itu dapat ditulis sebagai berikut (di sini, tiap-tiap *actor* digambarkan sebagai objek):

```
STARTGAME:
        Match : game
        Player1, Player2 : player

        Random(player1, player2)
        Player1.Shuffle()
        Player2.Shuffle()
        Player1.Draw7Card()
        Player2.Draw7Card()
        Player1.PilihactivePokemon(player1.hand)
        Player2.PilihactivePokemon(player2.hand)
        Match.Turn(player1)
        Match.turnnum= 1

PREPARINGPHASE:
        If (Match.turnnum != 1) ->
player1.Draw1Card()
```

```
        Player1.SetBenchedPokemon()
        Player1.evolve()
        Player1.AttachEnergyCard()
        Player1.RetreatPokemon()

ATTACKPHASE:
        i : integer
        i := 0

        Player1.Chooseattack(listofattack[i],
        chosenattack)
        if
        (Player1.IsEnergyCukup(chosenattack.manan
        eed, manastore))
                i++
                Player1.Chooseattack(Player1.listo
        fattack[i], Player1.chosenattack)
        end if

        Player1.CheckCondition
        (chosenattack.cond)
        Player1.tempdamage =
        player1.chosenattack.damage
        if(Player2.isweakness()) -> tempdamage *
        2
        if(Player2.isresistance()) -> tempdamage
        - 30
        attackeffect(tempdamage)

        Player2.activePokemon.minus :=
        Player2.activePokemon.minus +tempdamage

        if (enemy.activePokemon.hp <= 0)
                Player2.moveactivetodiscardpile()
                If
        (Player2.listofbenchedPokemon.num = 0)
                        Player1.win = true
                        Player2.lose = true
                Endif

                Player1.Chooseactive()
                Player1.Getprize()
                if (Player1.prize.num = 0)
                        Player1.win = true
                        Player2.lose = true
                endif
        endif

        Match.turn(player2)
        Match.turnnum++
```

3. PEMBAHASAN PENELITIAN

3.1.. Pendekatan Algoritma *Greedy*

3.1.1. Algoritma *Greedy*

Dalam permainan ini, dapat kita lihat bahwa ada saat di mana *player*1 membutuhkan penentuan untuk menentukan *attack* mana yang akan diambil untuk melakukan *damage* terhadap *Pokemon* player2, dengan diikuti pula perhitungan dari *resistance* dan *weakness* musuh, serta efek dari *attack* yang dipilih, yaitu *attackeffect*. Penyelesaian untuk menghasilkan *damage* yang optimal dalam tiap *attack phase* ini dapat menggunakan algoritma *Greedy*.

Elemen-elemen yang terdapat pada algoritma *Greedy* dapat diputuskan sebagai berikut:

- Himpunan kandidat : list of *attack* pada *active Pokemon Card player*1
- Himpunan solusi : *attack* terpilih yang merupakan *attack* dengan nilai *damage* akhir tertinggi yang dapat dicapai turn ini
- Fungsi seleksi (*selection function*) :pilihlah *attack* yang memiliki *damage* tertinggi pada list of *attack* dengan memperhitungkan *weakness* dan *resistance* serta *attack effect* dari tiap-tiap *attack* (dibahas kemudian).
- Fungsi kelayakan (*feasible*) : cukupnya *energy Card* yang dibutuhkan oleh *attack*
- Fungsi Obyektif: *attack* yang dipilih cukup memiliki *damage* yang dapat mengalahkan musuh pada turn ini atau *attack* yang dipuluh dapat melakukan *damage* pada musuh dengan nilai *damage* terbesar jika *damage* yang dihasilkan *attack* yang dipilih tidak dapat mengalahkan musuh pada turn ini.

Skema Algoritma *Greedy* itu sendiri mengikuti skema *Greedy* umum yang telah ada, yang dapat dirumuskan sebagai berikut:

1) inisialisasi C dengan kosong
2) pilih sebuah kandidat dengan fungsi seleksi dari C
3) kurangi C dengan kandidat yang sudah dipilih dari langkah (2)di atas
4) periksa apakah kandidat yang dipilih tersebut bersama-sama dengan himpunan solusi membentuk solusi yang layak atau feasible (dilakukan dengan fungsi LAYAK). Jika ya, masukkan kandidat tersebut ke dalam himpunan solusi. Jika tidak, buang kelayakan tersebut dan tidak perlu dipertunbangkan lagi.
5) periksa apakah himpunan solusi sudah memberikan solusi yang lengkap dengan menggunakan fungsi solusi. Jika ya, berhenti (selesai). Jika tidak, ulangi langkah (2)

Pada kasus penentuan *attack* yang memiliki nilai optimum, skema algoritma *Greedy* dapat dituliskan pula sebagai berikut

1) membuat attribut chosen*attack* yang masih kosong
2) memilih sebuah *attack* dari *active Pokemon player*1
3) kurangi *attack* yang telah dimasukkan tadi dari list of *attack*
4) periksa apakah *energy* yang dimiliki oleh *active Pokemon* tersebut dapat memenuhi *energy* yang

dibutuhkan oleh *attack* yang dipilih tadi. Jika cukup, masukkan ke dalam atribut chosen*attack*. Jika tidak, lewati *attack* tersebut

5) Periksa apakah *attack* yang kita pilih sudah dapat memberikan *damage* yang dapat mengalahkan *active Pokemon player*2pada turn ini atau telah dapat memberikan *damage* terbesar jika *active Pokemon player*2 tidak dapat dikalahkan pada turn ini (dengan membandingkan dengan chosen*attack* yang dibuat pada loop sebelumnya, ambil chosen*attack* yang memberikan *damage* tertinggi). Jika sudah, berhenti. Jika belum, lakukan langkah (2) lagi untuk melakukan pengecekan *attack* lainnya hingga list of *attack* habis.

3.1.2. Pengaruh *attack effect*

Perlu diperhatikan, pada bagian fungsi seleksi, terdapat efek dari tiap-tiap *attack*, yang disebut *attack effect*. Perlakuan untuk tiap-tiap jenis efek tersebut terhadap nilai akhir suatu *attack* akan berbeda-beda tergantung jenisnya. Nilai probabilitas didapatkan dari perhitungan didapatkannya *attack* maksimal yang berhasil dilakukan.

- Untuk *attack* yang tidak memiliki *effect attack*, nilai *attack* tetap
- Untuk *attack* yang memiliki *effect attack* melakukan penambahan dari *attack* berdasarkan suatu kondisi, seperti "*if fire energy Card is attached to this Pokemon, plus 10 damage",* tambahkan penambahan itu jika kondisi terpenuhi
- Untuk *attack* yang memiliki *effect attack* yang melakukan pengalian dari *attack* dasar berdasarkan suatu kondisi, seperti *"deal damage for each water energy Card attached to this Pokemon",* kalikan *damage* tersebut sesuai kondisi yang terpenuhi
- Untuk *attack* yang memiliki *effect attack* yang melakukan penambahan nilai *attack* berdasarkan probabilitas, seperti *"flip 1 coin, if head, deal 10 plus damage"*, lakukan penambahan *damage* tambahan itu *Set*elah membagi *damage* tambahan dengan nilai probabilitas didapatkannya tambahan *damage* itu
- Untuk *attack* yang memiliki *effect attack* yang melakukan pengalian keseluruhan *damage* dan terjadinya *damage* berdasarkan probabilitas, seperti *"flip 4 coins, deal damage for each head"*, nilai *attack* akhir ini dikalikan dengan nilai probabilitas terjadinya *attack* maksimum
- Untuk *attack* yang memiliki *effect attack* probabilitas terjadinya *damage* seluruhnya, seperti *"flip a coin, if tail, attack fail"*, *attack* tersebut memiliki perhitungan perkalian nilai *attack* akhir dari *attack* tersebut dengan nilai probabilitas terjadinya *damage* yang sukses.
- Untuk *attack* yang memiliki *attack effect* yang melakukan *damage* terhadap diri sendiri, akan menghasilkan nilai akhir *Set*elah mengurangi nilai *damage* total yang telah dijjumlahkan dengan *resistance* dan *weakness* dengan *damage* yang diterima *Pokemon* ini.
- Untuk *attack* yang memiliki probabilitas untuk terjadinya kejadian lain yang menguntungkan *player*1 itu sendiri ataupun tidak sama sekali, seperti *"you may remove 1 damage counter from 1 of your benched Pokemon"* atau *"shuffle your deck"*, nilai *attack* akhir tidak melakukan penambahan apapun.

3.2. Perhitungan *Damage* akhir tiap-tiap *attack*

Perhitungan *damage* akhir tiap-tiap *attack* mengikuti urutan berikut (dimisalkan *attack* yang dipilih disebut dengan variable X):

1. Dapatkan nilai *damage* dasar X
2. Jika *energytype* dari *resistance active Pokemon player*2 sama dengan salahsatu *energytype* yang digunakan untuk menggunakan X, kalikan *damage* dasar X dengan 2. Jika tidak, *damage* tetap
3. Jika *energytype* dari *resistance active Pokemon player*2 sama dengan salahsatu *energytype* yang digunakan untuk menggunakan X, kurangi *damage* yang didapat dari langkah (2) dengan 30, atau dengan angka yang tertera pada *resistance active Pokemon player*2. Jika tidak, *damage* pada akhir tahap (2) masih tetap
4. Perlakukan nilai *damage* hingga tahap (3) sesuai dengan *attack effect* dari X, dan didapatkan nilai akhir *damage* X

3.3. Contoh kartu *Pokemon* dan kasus *attack*

Contoh kasus:

*Player*1.*activePokemon*:	Vulpix
Vulpix.name:	vulpix
Vulpix.hp	50
Vulpix.minus:	30
Vulpix..*energy*:	fire
Vulpix.*energy*store	(fire,fire,
lightning)	
Vulpix.*stage*:	*basic*
Vulpix.prev	–
Vulpix.*weakness*	water
Vulpix.*resistance*	–
Vulpix.*retreatcost*	1
List of *attack*:	

```
Vulpix.attacklist[0].name     bite
Vulpix.attacklist[0].mananeed (colorless)
Vulpix.attacklist[0].damage   10
Vulpix.attacklist[0].effect           -
Vulpix.attacklist[0].name     firebreathing
Vulpix.attacklist[0].mananeed (fire, colorless)
Vulpix.attacklist[0].damage   20
Vulpix.attacklist[0].effect   flip a coin. If
heads, this attack does 20 damage plus 10 more
damage

Player2.activePokemon:        Butterfree
Butterfree.name:              butterfree
Butterfree.hp:                100
Butterfree.minus:             80
Butterfree.energy:            grass
Butterfree.energystore:       (grass, grass,
fire, fire)
Butterfree.stage:             stage2
Butterfree.prev:              metapod
Butterfree.weakness:          lightning
Butterfree.resistance:        fighting
Butterfree.retreatcost:       3
List of attack:
Butterfree.attacklist[0].name:        whirlwind
Butterfree.attacklist[0].mananeed:    (colorless,
colorless)
Butterfree.attacklist[0].damage:      30
Butterfree.attacklist[0].effect:      oppenent
switch the defending Pokemon with 1 of his or her
benched Pokemon
Butterfree.attacklist[1].name:        gust
Butterfree.attacklist[1].mananeed:    (grass,
colorless, colorless)
Butterfree.attacklist[1].damage:      50
Butterfree.attacklist[1].effect:      -
Butterfree.power: as long as butterfree is your
active Pokemon, remove 1 damage counter from each
of your Pokemon between turns <tidak dibahas>
```

Pada keadaan di atas, *attack* yang paling optimal untuk dilakukan oleh *active Pokemon player*1 (*Vulpix*) adalah FireBreathing, yang memberikan *damage* sebesar 20 yang memiliki tambahan tingkat probabilitas 50% untuk menghasilkan 10 *damage* tambahan sehingga nilai *damage attack-attack* yang akan masuk ke dalam algoritma adalah:

Bite: 10 dmg

FireBreathing: 20+(10*50%)=25dmg → dipilih

Hasil dari *damage attack* yang dipilih diperhitungkan bisa mengalahkan *active Pokemon player*2 (*Butterfree*) yang memiliki sisa HP 20.

4. KESIMPULAN

Algoritma *Greedy* merupakan algoritma yang efektif untuk dapat menyelesaikan penentuan *attack* paling optimal yang akan dilakukan oleh *player* dalam permainan *Pokemon Trading Card Game* (*Pokemon TCG*)dengan *Cardlist Trainer Base Set*. Permasalahan ini merupakan permasalahan yang menjadi salah satu faktor kemenangan pemain dalam permainan *Pokemon TCG, baik skala kecil maupun turnamen dunia*. Himpunan Solusi dari algoritma *Greedy* merupakan *attack* yang memiliki nilai *damage* paling optimal yang didapat *player* pada turn ini, *Set*elah memperhitungkan segala situasi yang ada.

REFERENSI

[1] Munir,Rinaldi. 2003.*Matematika Diskrit.* Bandung: Informatika.

[2] Silvertsen, Jimmer. 2005.*Pokemon Trading Card Game Rule Book*. Redmond: Pokemon USA, Inc.