Optimasi Alokasi Sumber Daya untuk Produksi Pasukan pada Gim *Age of Empires 2* dengan *Dynamic Programming* (Variasi *Unbounded Knapsack Problem*)

Tugas Makalah IF2211 Strategi Algoritma

Alvin Wilta - 13519163
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (13519163@std.stei.itb.ac.id):

***Abstrak*—Permainan *Age of Empires 2* merupakan permainan strategi yang membutuhkan pengelolaan sumber daya yang baik untuk memenangkan permainan baik itu untuk bertahan ataupun untuk menyerang sehingga membutuhkan alokasi sumber daya yang tepat untuk menghindari kerugian. Saat maju berperang, hal yang diperhatikan adalah *attack* dari penyerang dan health dari petahan. Ketika menyerang, dibutuhkan unit yang tepat dan membutuhkan sumber daya yang spesifik untuk mencapai *attack* yang besar sehingga untuk mengetahui apakah sebuah alokasi sumber daya untuk membuat unit adalah cukup untuk berperang dengan lawan, dibutuhkan pemodelan permasalahan dengan *unbounded knapsack problem* dengan optimasi *dynamic programming* untuk mempercepat pencarian karena terdapat banyak jenis unit dan bisa berulang kali.**

Keywords—unbounded knapsack problem; program dinamis; age of empires 2; alokasi sumber daya; unit; strategi

I. PENDAHULUAN

Age of Empires 2 adalah sebuah gim atau permainan *real-time strategy* (RTS) yang dikembangkan oleh Ensemble Studios dan dipublikasikan oleh Microsoft pada tahun 1999 untuk versi pertama dan tahun 2018 untuk versi *Definitive Edition* yang menambahkan jenis permainan dan grafik yang baru. Permainan ini adalah gim yang terfokus pada pengumpulan dan produksi sumber daya, pembangunan kota dan kerajaan, serta produksi pasukan untuk melawan musuh. Tujuan dari permainan ini adalah untuk mengalahkan kerajaan musuh dengan memanfaatkan perkembangan masa atau *ages* untuk meningkatkan efisiensi pasukan atau pengumpulan sumber daya. Pada permainan ini terdapat 4 masa utama yang berfungsi untuk meng-*unlock* kemampuan untuk memproduksi pasukan atau bangunan jenis lain yang lebih kuat dibandingkan masa sebelumnya. Permainan ini juga memiliki lebih dari 30 pilihan peradaban atau negara yang pasukan dan bangunannya memiliki keunikan dan bonusnya masing-masing.

Pada permainan ini, aspek yang perlu diperhatikan adalah pengelolaan sumber daya. Sumber daya utama pada permainan adalah makanan (*food*), kayu (*wood*), dan emas (*gold*) untuk membangun beragam jenis bangunan untuk bertahan, bangunan untuk menambah pemasukan sumber daya, atau pasukan untuk menyerang lawan. Strategi dari permainan ini adalah bagaimana alokasi sumber daya yang sudah didapatkan terhadap kebutuhan pasukan dan bangunan, serta pasukan atau bangunan apa yang akan dibutuhkan pada saat tertentu. Pada makalah ini, penulis akan membahas lebih dalam mengenai salah satu aspek permainan, yaitu peperangan. Khususnya alokasi dari sumber daya untuk kebutuhan pasukan dan jenis pasukan yang dibutuhkan dalam peperangan.

Selain alokasi sumber daya untuk pasukan, aspek seperti kuantitas pasukan, serta jenis pasukan juga berpengaruh terhadap hasil akhir dari peperangan. Alokasi sumber daya untuk pasukan akan menentukan apakah pasukan dapat cukup diproduksi (kuantitas dari pasukan), kuantitas pasukan akan menentukan apakah pasukan pemain dapat menghabisi dengan tuntas pasukan lawan, dan jenis pasukan akan menentukan keunggulan secara individual dari pasukan. Contoh dari keunggulan secara individual adalah pemanah dan kavaleri, kavaleri akan lebih unggul dibandingkan pemanah dan akan menghabisi pemanah dengan cepat. Tetapi jika kavaleri ditandingkan dengan infanteri maka kavaleri akan kalah oleh infanteri karena keunggulan infanteri terhadap kavaleri.

Untuk menentukan apakah alokasi sumber daya untuk pasukan dan jenis pasukan, dapat dilakukan pemodelan permasalahan optimasi produksi pasukan menggunakan model permasalahan *Unbounded Knapsack* dan diimplementasikan menggunakan konsep *Dynamic Programming* untuk mengoptimalisasi pencarian solusi.

II. Analisis Permasalahan

A. *Mekanisme Permainan Age of Empires 2*

Age of Empires 2 atau disingkat sebagai AoE2 adalah permainan yang terpusat pada pembangunan kota dan peperangan dengan kota lawan dan permainan dinyatakan selesai ketika lawan menyatakan menyerah atau ketika *Town Center* dari lawan hancur. *Town Center* adalah bangunan yang dapat memproduksi *Villager*, unit yang dapat mengumpulkan sumber daya berupa makanan, kayu, dan emas. Kemudian dari sumber daya tersebut, pemain dapat membuat bangunan lainnya yang berfungsi untuk meningkatkan ekonomi atau bangunan untuk memproduksi pasukan sesuai dengan jenis bangunannya. Setelah itu pemain dapat melakukan *Age up*, yaitu meningkatkan *Age* untuk meng-*unlock* unit pasukan baru yang lebih kuat untuk mengalahkan lawan yang masih memiliki unit pasukan yang belum cukup maju.

Permainan dimulai dengan memilih jenis peradaban dari pilihan peradaban dari 35 peradaban (tanpa *expansion pack*) yang tersedia, masing-masing memiliki keunggulannya sendiri yang tidak akan dijelaskan lebih jauh pada makalah ini karena kompleksitasnya dan dapat dilihat dari tautan berikut ini (*https://ageofempires.fandom.com/wiki/Civilizations_(Age_of_Empires_II)*)

Gambar 1. Layar Pemilihan Peradaban

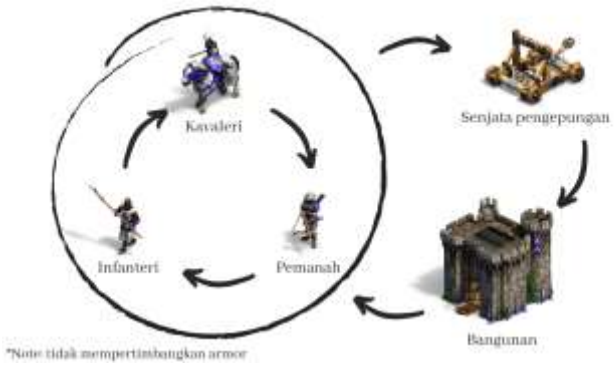
Kemudian setelah memilih peradaban dan memulai permainan, hal pertama yang diberikan oleh permainan umumnya adalah 3 unit *Villager* dan 1 Town Center dan diberikan 200 *food*, 200 *wood*, dan 100 *gold* sebagai sumber daya permulaan (permulaan permainan dapat bervariasi tergantung peradaban yang dipilih).

Gambar 2. Permulaan normal permainan Age of Empires 2

Umumnya pemain sebaiknya langsung memproduksi lebih banyak *villager* untuk meningkatkan pendapatan sumber daya yang akan digunakan untuk membuat bangunan yang dapat memproduksi pasukan. Pada umumnya bangunan tersebut masing-masing memproduksi terbatas hanya 1 kategori atau jenis pasukan saja. Pada permainan AoE2, pasukan dikategorikan menjadi 4 kategori terkecuali bangunan dan memiliki tandingannya (*counter*) masing-masing.

Setiap unit memiliki 3 status utama yang akan berpengaruh ketika ditandingkan satu sama lain. Status tersebut adalah *health*, *cost*, dan *attack.* Health adalah status yang menyatakan berapa *attack* yang dapat ditanggung oleh unit tersebut sebelum meninggal. Cost adalah biaya yang dibutuhkan untuk membuat unit tersebut. Attack adalah kekuatan unit tersebut untuk melukai unit lainnya dalam satu kali serangan, dan khusus untuk *attack* dibagi lagi menjadi 2 tipe, yaitu *normal attack* dan *advantage attack. Normal attack* adalah nilai *attack* yang akan dikenakan kepada unit selain unit tandingan dan dihitung berdasarkan normal attack dikali dengan bonus *multiplier*nya, sedangkan ketika unit tersebut bertemu dengan unit tandingannya, maka unit tandingan tersebut akan dikenakan *advantage attack.*

Berikut adalah skema tandingan masing-masing kategori dengan catatan bahwa skema terkait tidak memperhatikan hal seperti *armor* yang dapat mengurangi serangan yang didapat.



Gambar 3. Skema tandingan antar kategori

Maksud dari arah panah adalah "... unggul terhadap ...". Contohnya kavaleri unggul terhadap pemanah tetapi lemah terhadap infanteri sehingga ketika kavaleri menyerang pemanah, maka pemanah akan dikenakan pengurangan *health* sebanyak

advantage attack dari kavaleri. Sedangkan jika kavaleri menyerang infanteri, maka *health* infanteri hanya dikurangi oleh *normal attack* dari kavaleri. Pada makalah ini, unit yang dibahas hanya unit kategori infanteri, kavaleri, dan pemanah saja untuk menyederhanakan persoalan.

Untuk memenangkan peperangan terhadap lawan, maka pemain harus menentukan unit apa saja yang perlu dibuat berdasarkan keunggulan unit pemain dengan unit lawan, alokasi sumber daya yang dapat . Oleh karena itu permasalahan ini dapat

B. *Permasalahan Knapsack (Knapsack Problem)*

Permasalahan knapsack adalah salah satu dari permasalahan yang melibatkan optimasi. Definisi dari permasalahan ini adalah pencarian kombinasi dari suatu himpunan barang yang memiliki nilai dan berat sedemikian rupa sehingga dari kombinasi tersebut didapati nilai maksimal dengan total berat barang tidak melebihi berat maksimal *knapsack*. Untuk maksimasi dan batasan dari pengambilan barang dapat ditulis sebagai berikut:

$$maksimasi = \sum_{i=1}^{n} v_i x_i$$

$$batasan = \sum_{i=1}^{n} w_i x_i \leq W \; dan \; x_i \in \{0,1\}$$

Permasalahan ini juga biasa disebut sebagai 0/1 *knapsack problem* karena barang yang diambil hanya terbatas antara diambil (1) atau tidak diambil (0). Tetapi selain 0/1 *knapsack problem*, terdapat varian lainnya seperti permasalahan knapsack terbatas (*bounded knapsack problem*) dan permasalahan knapsack tidak terbatas (*unbounded knapsack problem*). Perbedaan varian tersebut terletak pada teknis pengambilan barang.

Pada 0/1 *knapsack problem* setiap barang hanya bisa diambil sekali, pada *bounded knapsack problem* setiap barang hanya dapat diambil sebanyak c kali, dan pada *unbounded knapsack problem* barang dapat diambil berkali-kali tanpa batasan.

C. *Permasalahan Knapsack Tidak Terbatas (Unbounded Knapsack Problem)*

Permasalahan knapsack tidak terbatas adalah salah satu varian dari *permasalahan knapsack* yang memungkinkan barang yang sama terambil lebih dari satu kali tanpa batasan. Contohnya untuk kasus pemotongan kayu, yaitu terdapat sebuah kayu yang dapat dipotong menjadi berbagai jenis ukuran dengan harga yang berbeda-beda juga.

Misalkan terdapat list harga dari panjang kayu yang dapat dilihat pada tabel berikut:

Tabel 1. Contoh Daftar Harga Potongan Kayu

No.	Potongan (cm)	Nilai ($)
1	1	1
2	2	5
3	3	8
4	4	9

Kemudian diketahui terdapat batangan kayu dengan panjangnya 4 cm. Berdasarkan data yang didapati pada Tabel 1 di atas, bisa dibuat kombinasi pemotongan kayu dengan kemungkinan sebagai berikut:

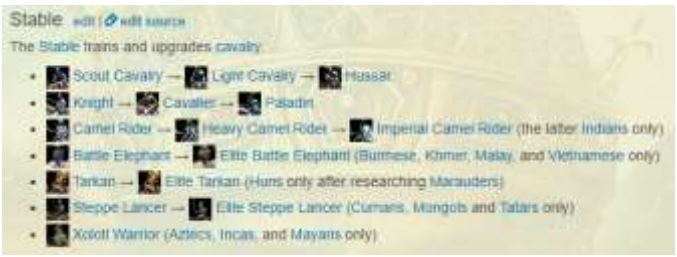
Tabel 2. Kombinasi Potongan Batang dan Nilai Totalnya

No.	Kombinasi potongan (cm)	Nilai total ($)
1	1, 1, 1, 1	4
2	1, 1, 2	7
3	2,2	10
4	3,1	9
5	4	9

Didapati pada Tabel 2 bahwa kombinasi potongan no 3 menghasilkan nilai total maksimum untuk batang kayu sepanjang 4 cm. Maka didapati nilai maksimal yang bisa diperoleh dari potongan kayu sepanjang 4 cm adalah 10 dollar.

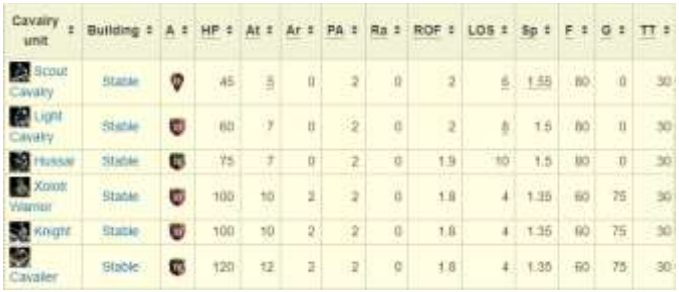
D. *Keterkaitan Permasalahan Knapsack dengan Permasalahan Pemilihan Pasukan pada permainan AoE2*

Pemilihan kategori jenis pasukan yang akan dibuat tidak akan dikaitkan dengan permasalahan *knapsack*, tetapi permasalahan *knapsack* dapat digunakan untuk menentukan total *attack* maksimal yang bisa dihasilkan oleh berbagai jenis unit dari satu kategori (misalnya *Scout Cavalry* dan *Knight* yang berbeda unit tetapi masih dikategorikan sebagai kavaleri) dengan alokasi sumber daya terbatas.



Gambar 4. List Unit Kategori Kavaleri

Masing-masing dari contoh unit pada kategori kavaleri tersebut *Gambar 4* membutuhkan sumber daya yang berbeda-beda dan kriteria yang berbeda juga, seperti yang ditunjukkan pada *Gambar 5* yang menunjukkan statistik dari beberapa unit kavaleri dasar yang dimiliki oleh semua peradaban.



Cavalry unit	Building	A	HP	At	Ar	PA	Ra	ROF	LOS	Sp	F	G	TT
Scout Cavalry	Stable		45	5	0	2	0	2	6	1.55	80	0	30
Light Cavalry	Stable		60	7	0	2	0	2	6	1.5	80	0	30
Hussar	Stable		75	7	0	2	0	1.9	10	1.5	80	0	30
Xolotl Warrior	Stable		100	10	2	2	0	1.8	4	1.35	60	75	30
Knight	Stable		100	10	2	2	0	1.8	4	1.35	60	75	30
Cavalier	Stable		120	12	2	2	0	1.8	4	1.35	60	75	30

Gambar 5. Status dan Cost dari Kavaleri

Pada *Gambar 5*, masih terdapat beberapa statistik yang tidak dibutuhkan untuk penyelesaian masalah ini seperti *Armor*, *Pierce Armor*, *Range*, *Rate of Fire*, dll sehingga dapat disederhanakan menjadi sebagai berikut:

Tabel 3. Attack, dan Cost dari Unit Kavaleri

Nama	Attack	Food	Wood	Gold
Scout Cavalry	5	80	0	0
Light Cavalry	7	80	0	0
Hussar	7	80	0	0
Xolotl Warrior	10	60	0	75
Knight	10	60	0	75
Cavalier	12	60	0	75

Sehingga pada dasarnya akan dicari *attack* maksimum yang bisa diperoleh dari alokasi sumber daya yang diberikan pemain, kemudian dibandingkan dengan *health* dari musuh. Dalam permainan standar, umumnya unit akan menyerang sebanyak 8 kali, sehingga ketika total attack maksimum yang sudah dikalikan dengan 20 kurang dari *health* lawan, maka alokasi sumber daya dinyatakan tidak cukup.

III. Implementasi

Berdasarkan Tabel 3, yang akan diimplementasikan menggunakan *dynamic programming* dengan model *unbounded knapsack problem* hanya data sumber daya (*food*, *wood*, *gold*) dan *attack*. Untuk menyederhanakan persoalan ini, sumber daya akan digabung menjadi satu data, yaitu *resources* dengan nilai berasal dari jumlah *food* + *wood* + *gold*. Pada aplikasinya, permainan ini memiliki *market* untuk menukar *food*, *wood*, dan *gold* dengan satu sama lain sehingga masih relevan.

Pada implementasi ini, constraint untuk permasalahannya adalah *resources* yang merupakan gabungan dari *food*, *wood*, dan *gold*, dan akan menentukan total *attack* maksimal dari gabungan unit tersebut berdasarkan *resources* yang sudah dialokasikan dari pemain.

A. Langkah Algoritma

1. Menentukan list dari pasukan yang akan dipilih
2. Menentukan batasan resource yang akan dialokasikan
3. Menentukan jumlah *health* dari lawan
4. Menentukan kategori unit lawan (0/1/2)
5. Mencari nilai attack terbesar
6. Mengalikan nilai attack terbesar dengan 20 dan membandingkannya dengan *health* musuh
7. Jika *attack* lebih besar, maka alokasi dapat digunakan
8. Jika *attack* gagal, maka alokasi tidak dapat digunakan

Untuk mendapatkan nilai attack terbesar, digunakan fungsi utbk yang mengembalikan nilai dari ret[res] dimana ret[i] mengindikasikan profit maksimum dengan kapasitas knapsack sebesar i, sehingga jawaban dari knapsack berada pada ret[res] dimana res adalah nilai alokasi resources yang akan digunakan.

B. Implementasi Kode Program

Berikut adalah fungsi untuk menghitung nilai maksimal *attack* yang bisa didapatkan dari resource berjumlah *res*.

```
# Algoritma unbounded knapsack problem
dengan dynamic programming
def unbk(res, n, val, wt):
    ret = [0 for i in range(res + 1)]

    for i in range(res + 1):
        for j in range(n):
            if (wt[j] <= i):
                ret[i] = max(ret[i],
ret[i - wt[j]] + val[j])

    return ret[res]
```

Berikut adalah fungsi untuk membaca file csv yang berisikan statistik list pasukan dengan perbarisnya berisi sesuai urutan nama, attack (advantage), food, wood, gold, category. Category adalah id yang membedakan antara kategori infanteri (0), kavaleri (2), dan pemanah (1).

```
def readcsv(filename):
    import csv
    ifile = open(filename)
    reader = csv.reader(ifile,
delimiter=",")
    result = []

    for row in reader:
        result.append (row)

    ifile.close()
    return result
```

Contoh isi file csv:

Scout cavalry,7,80,0,0,2

Knight,14,60,0,75,2

Steppe lancer,9,70,0,40,2

Archer,6,0,25,45,1

Skirmisher,4,25,35,0,1

Hand cannoneer,17,45,0,50,1

Militia,6,60,0,20,0

Spearman,5,35,25,0,0

Eagle warrior,7,30,0,50,0

Berikut adalah fungsi-fungsi untuk mengolah data objek hasil baca csv menjadi data yang dapat digunakan pada algoritma penyelesaian *unbounded knapsack problem* berupa *array of value* dan *array of weight*.

Fungsi untuk memilah nilai *attack* yang sesuai dengan kategori *counter* nya. Jika musuh archer maka nilai *attack* yang digunakan berasal dari kavaleri, dan lainnya.

```
def countatk(object,code):
    #mengambil attack saja
    result = []
    temp = len(object)
    for i in range(temp):
        if (code == '0' and
object[i][5] == '1'):

result.append(int(object[i][2]))
        elif (code == '1' and
object[i][5] == '2'):

result.append(int(object[i][2]))
        elif (code == '2' and
object[i][5] == '0'):

result.append(int(object[i][2]))
    return result
```

Fungsi untuk memilah nilai *resource* yang merupakan gabungan dari nilai *food*, *wood*, dan *gold*.

```
def countres(object,code):
    result = []
    temp = len(object)
    for i in range(temp):
        if (code == '0' and
object[i][5] == '1'):

result.append(int(object[i][3])+int(obj
ect[i][4])+int(object[i][5]))
        elif (code == '1' and
object[i][5] == '2'):

result.append(int(object[i][3])+int(obj
ect[i][4])+int(object[i][5]))
        elif (code == '2' and
object[i][5] == '0'):

result.append(int(object[i][3])+int(obj
ect[i][4])+int(object[i][5]))
    return result
```

Berikut ini adalah implementasi dari program utama yang akan menjalankan fungsi-fungsi yang sudah dideklarasikan sebelumnya.

```
# Driver program
# name | (advantage) attack | food |
wood | gold | category
# category = {0,1,2}
# 0 = infantry
# 1 = archer
# 2 = cavalry

# input (asumsi semua input sudah
valid)
filename = input('enter filename path:
') # nama file (list semua pasukan)
enemyhp = int(input('enter enemy hp:
')) # health lawan
enemycode = str(input('enter enemy
code: ')) # kategori unit lawan
res = int(input('enter allocation of
resources: ')) # resource yang
dialokasikan

# inisialisasi
obj = readcsv(filename)
val = countatk(obj,enemycode)
wt = countres(obj,enemycode)
n = len(val)

maks = unbk(res, n, val, wt)
if (maks*20 > enemyhp):
    print('total attack: ' + str(maks))
    print('Alokasi resource valid dan
bisa melawan musuh')
    print('Kategori yang harus dipakai
adalah ', end='')
    if (enemycode == '0'):
        print('archer')
    elif (enemycode == '1'):
        print('cavalry')
    elif (enemycode == '2'):
        print('infantry')

else:
    print(maks)
    print('Alokasi resource sebesar ' +
str(res) + ' tidak cukup untuk melawan
musuh')
```

C. Studi Kasus

Diberikan data unit sebagai berikut dengan keterangan Atk sebagai attack, F sebagai food, W sebagai wood, G sebagai gold, dan C sebagai kategori:

Tabel 4. List Pasukan Studi Kasus

No	Name	Atk	F	W	G	C
1	Scout Cavalry	7	80	0	0	2
2	Knight	14	60	0	75	2
3	Steppe Lancer	9	70	0	40	2
4	Archer	6	0	25	45	1
5	Skirmisher	4	25	35	0	1
6	Hand Cannoneer	17	45	0	50	1
7	Militia	6	60	0	20	0
8	Spearman	5	35	25	0	0
9	Eagle Warrior	7	30	0	50	0

1. Testcase 1

   Enemy HP: 300

   Enemy Type: 2

   Allocation of Resources: 1000

```
enter filename path: ./data.csv
enter enemy hp: 300
enter enemy code: 2
enter allocation of resources: 1000
============================
total attack: 3000
Alokasi resource valid dan bisa melawan musuh
Kategori yang harus dipakai adalah infantry
```

Pada kasus ini, total attack yang didapat berupa 3000, maka maksimal *attack* yang didapat adalah 150 dari kategori infantry

2. Testcase 2

   Enemy HP: 400

   Enemy Type: 1

   Allocation of Resources: 300

```
enter filename path: ./data.csv
enter enemy hp: 12000
enter enemy code: 0
enter allocation of resources: 500
============================
430
Alokasi resource sebesar 500 tidak cukup untuk melawan musuh
```

Pada kasus ini, alokasi resource yang diberikan tidak cukup untuk melawan musuh

3. Testcase 3

   Enemy HP: 1200

   Enemy Type: 1

   Allocation of Resources: 600

```
enter filename path: ./data.csv
enter enemy hp: 1200
enter enemy code: 1
enter allocation of resources: 600
============================
total attack: 24000
Alokasi resource valid dan bisa melawan musuh
Kategori yang harus dipakai adalah cavalry
```

Pada kasus ini, total attack yang didapat berupa 24000, maka maksimal *attack* yang didapat adalah 1200 dari kategori cavalry

IV. Penutup

A. *Kesimpulan*

Program dinamis dapat diaplikasikan pada berbagai aspek dan disiplin ilmu karena program dinamis merupakan suatu algoritma optimasi yang berguna untuk mempercepat kinerja dari suatu algoritma, spesifiknya pada kasus permasalahan *unbounded knapsack* yang dibahas pada makalah ini. Algoritma program dinamis dapat digunakan untuk mempercepat pencarian *attack* terbesar yang bisa didapatkan dari alokasi sumber daya terkait pada permainan *Age of Empires 2* menggunakan pemodelan *unbounded knapsack*. Dengan demikian pemain dapat mengetahui berapa banyak sumber daya yang harus disiapkan sebelum memulai peperangan dengan lawannya.

B. *Saran*

Didalam permainan, masih banyak faktor yang perlu dipertimbangkan seperti *attack speed* karena permainan merupakan permainan RTS, *armor* dan *armor piercing* yang menegasikan serangan, dan berbagai hal lainnya yang belum dipertimbangkan. Selain itu, penggabungan *resources* kurang efektif karena penggunaan market untuk menukar antar sumber daya memiliki pajak sehingga sumber daya yang dibutuhkan lebih mahal. Oleh karena itu pemodelan yang seharusnya digunakan adalah *unbounded knapsack with multiple constraint* untuk mengukur alokasi sumber daya yang lebih akurat.

Acknowledgment

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah membantu dalam proses pembuatan makalah ini. Pertama penulis memanjatkan puji syukur kepada Tuhan Yesus Kristus yang selalu memberikan berkat dan kasih karunia-Nya sehingga makalah ini dapat terselesaikan dengan baik. Kedua penulis berterima kasih kepada Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D.selaku dosen mata kuliah IF2211 Strategi Algoritma K-3 yang sudah berbagi ilmu dan membimbing penulis untuk memahami materi yang terpakai di dalam makalah ini. Ketiga kepada penulis lainnya pada bagian daftar referensi yang sudah berbagi ilmunya sehingga memperkaya isi dari makalah ini. Keempat kepada orang tua dan teman-teman penulis yang senantiasa memberikan dukungan dan semangat untuk menyelesaikan makalah ini.

References

[1] IF2211 Strategi Algoritma (2020). Program Dinamis (Dynamic Programming). KK Informatika STEI ITB. (diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf pada 10 Mei 2021 pukul 11:22 WIB)

[2] Hu T.C., Landa L., Shing MT. (2009) The Unbounded Knapsack Problem. In: Cook W., Lovász L., Vygen J. (eds) Research Trends in Combinatorial Optimization. Springer, Berlin, Heidelberg. (diakses melalui https://doi.org/10.1007/978-3-540-76796-1_10 pada 10 Mei 2021 pukul 20:56 WIB)

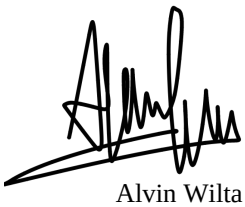
[3] Kellerer, Hans; Pferschy, Ulrich; Pisinger, David (2004). Knapsack Problems. Springer. (diakses melalui https://link.springer.com/book/10.1007%2F978-3-540-24777-7 pada 11 Mei 2021 pukul 15:44 WIB)

[4] Goddard, Steve. Dynamic Programming. CSCE 310J Computer Science and Engineering. (diakses melalui http://cse.unl.edu/~goddard/Courses/CSCE310J/Lectures/Lecture8-DynamicProgramming.pdf pada 11 Mei pukul 17.32 WIB)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Jakarta, 11 Mei 2021



Alvin Wilta
13519163