Optimasi Pencarian Kartu Berbasis Algoritma Genetika

Preprint · January 2021

DOI: 10.13140/RG.2.2.10701.36323

1 author:

Ide Yunianto
Universitas Gadjah Mada
4 PUBLICATIONS **1** CITATION

SEE PROFILE

Optimasi Pencarian Kartu Berbasis Algoritma Genetika

Ide Yunianto
Departemen Teknik Eektro dan Teknologi Informasi
Universitas Gadjah Mada
Yogyakarta, Indonesia
ide.y@mail.ugm.ac.id

Abstract*—Pencarian urutan sejumlah kartu dari tumpukan kartu yang disusun secara acak merupakan salah satu permasalahn kombinatorial. Permasalahan ini dapat diselesaikan dengan mencoba semua kemungkinan yang ada. Sayangnya, proses ini membutuhkan waktu yang lama untuk dapat menemukan urutan kartu yang diinginkan. Berdasarkan permasalahan tersebut, dilakukan penelitian untuk menyelesaikan permasalahan pencarian urutan kartu dengan menggunakan algoritma genetika. Algoritma genetika dalam penelitian ini dilakukan dalam serangkaian proses yaitu: pengkodean, penentuan representasi individu, inisialisasi populasi awal, evaluasi fitness, seleksi, crossover, dan mutasi. Dari penelitian ini, pencarian target urutan kartu As Sekop, 2 Keriting, 3 Hati, dan 4 Wajik dengan kode [0,14,28,42] berhasil dilakukan kurang dari 1000 iterasi dengan nilai fitness terbaik adalah 208. Algoritma genetika dapat mengoptimasi pencarian kartu dari kemungkinan sebanyak 6.497.400 urutan.

***Keywords*—algoritma genetika, permutasi, optimasi, pencocokan pola, kombinatorial**

I. PENDAHULUAN

Permasalahan kombinatorial merupakan permasalahan yang sering ditemui dalam kehidupan sehari-hari. Sebagai contoh seseorang yang membawa tas dengan fitur kunci kombinasi lupa kode sandi untuk membuka tas tersebut. Hal yang dapat dilakukan orang tersebut adalah mencoba semua kombinasi angka yang mungkin untuk memperoleh kode yang tepat. Apabila jumlah kombinasi yang tersedia banyak, tentu saja akan dibutuhkan waktu yang sangat lama untuk dapat memecahkan kode sandi tersebut. Contoh permasalahan yang lain yaitu menemukan rute terpendek untuk mengunjungi sejumlah tempat dalam sekali perjalanan. Setiap tempat dikunjungi hanya satu kali dan kembali lagi ke tempat semula ketika seluruh tempat telah dikunjungi. Permasalahan ini dinamakan Travelling Salesmen Problem (TSP). Sebagaimana kasus sebelumnya, solusi untuk dari permasalahan ini juga diperoleh dengan mencoba semua kemungkinan yang ada. Cara-cara yang digunakan untuk mendapatkan nilai optimum dari permasalahan-permasalahan yang telah disampaikan sebelumnya merupakan kegiatan optimasi [1].

Salah satu metode untuk menyelesaikan permasalahan kombinatorial adalah Algoritma Genetika [2]. Algoritma Genetika merupakan menggunakan pendekatan komputasi *evolutionary* yang meniru proses seleksi alami dan banyak digunakan untuk memecahkan permasalahan optimasi. Metode ini telah terbukti mampu memecahkan permasalahan TSP [1].

Seperti juga angka dan huruf, kartu permainan berisi simbol-simbol yang sering digunakan untuk menerangkan kasus-kasus matematika dan statistika, misalnya tentang probabilitas. Sehingga permasalahan pencocokan kata menggunakan algoritma genetika [3] sangat memungkinkan untuk diimplementasikan pada kartu permainan. Berdasarkan beberapa permasalahan yang telah disampaikan, peneliti bermaksud menguji algoritma genetika untuk memecahkan permasalahan optimasi kombinatorial pada pencarian sejumlah sampel berurutan dari tumpukan kartu yang disusun secara acak.

II. METODOLOGI

A. Kartu Permainan

Penelitian ini menggunakan kartu permainan berisi 52 lembar kartu yang terdiri dari empat bentuk kartu yaitu:

1. Sekop (*spades*)
2. Keriting (*clover/ club*)
3. Hati (*heart*)
4. Wajik/ Berlian (*diamond)*

Keempat jenis kartu tersebut terdiri dari dari 13 nilai kartu yang disimbolkan dengan angka 2 hingga 10, “As”, “King”, “Queen”, dan “Jack”.

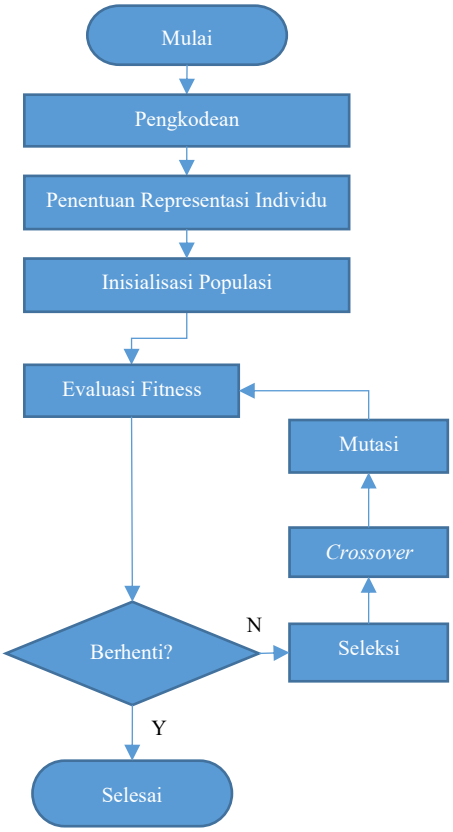
B. Alur Pencarian Kartu

Gambar 1 memberikan gambaran tentang permasalahan yang akan diselesaikan. Pertama kali tumpukan kartu diacak, kemudian diambil empat sampel kartu secara berurutan. Setelah itu keempat kartu dikembalikan ke tumpukan dan seluruh kartu diacak. Tantangan dari permasalahan ini adalah mencari keempat kartu secara berurutan menggunakan algoritma genetika.

Gambar 1 Alur Pencarian Kartu

C. Pengembangan *Algoritma Genetika*

Penelitian ini menggunakan algoritma genetika untuk optimasi pencarian sampel kartu di dalam tumpukan kartu yang disusun secara acak. Alur proses algoritma genetika dilakukan sebagai berikut (Gambar 2):



Gambar 2 Alur proses Algoritma Genetika.

1. Pengkodean Kartu

Setiap kartu dikodekan ke dalam bentuk *array* seperti yang ditunjukkan pada TABEL 1. "As Sekop" berada di posisi 0, "2 Sekop" berada di posisi 1, "3 Sekop" berada di posisi 2 dan seterusnya.

TABEL 1 Pengkodean Kartu

Indeks *Array*	**0**	…	**13**	…	**26**	…	**39**	…
Nilai	As Sekop	…	As Keriting	…	As Hati	…	As Wajik	…

2. Penentuan Representasi Individu

Apabila pada tumpukan kartu diambil empat kartu acak yang nantinya dijadikan target pencarian dan diperoleh urutan pengambilan "As Sekop", "2 Keriting", "3 Hati", dan "4 Wajik", maka representasi individu dari target yang akan dicari adalah [0, 14, 28, 42] (Gambar 3). Setiap individu direpresentasikan sebagai kromosom yang memiliki gen sejumlah *n*. Dalam contoh ini individu memiliki 4 gen.

As Sekop	2 Keriting	3 Hati	4 Wajik

0	14	28	42

Gambar 3 Representasi Individu dari target kartu yang akan dicari

3. Inisialisasi Populasi

Pada tahap ini dibangkitkan sejumlah individu secara acak sebanyak *m*. Masing-masing individu di dalam populasi ini nanti yang akan mengalami proses seleksi untuk dijadikan sebagai induk dari keturunan berikutnya. Pada penelitian ini ditetapkan sebanyak 100 individu per populasi.

4. Evaluasi Fitness

Evaluasi fitness dilakukan dengan menentukan fungsi fitness terlebih dahulu. Fungsi fitness digunakan untuk memberikan probabilitas yang lebih besar bagi individu-individu yang terbaik di dalam populasi untuk terpilih menjadi calon induk. Fungsi ini akan berbeda-beda tergantung dari permasalahan yang ingin dipecahkan. Berikut ini adalah fungsi fitness yang digunakan dalam penelitian ini:

$$fitness = 52\ (n) - \sum_{i=1}^{n} |g_i^k - t_i|$$

di mana:
Nilai 52 adalah total gen unik yang tersedia
n = jumlah gen
i = nomor indeks gen
g_i^k = gen ke-i dari individu ke-k
t_i = gen ke-i dari individu target

Pada Gambar 4 ditunjukkan cuplikan kode fungsi *fitness* menggunakan bahasa pemrograman Python.

```
#Hitung Fitness Individu/ Kromosom
def hitung_fitness(xkromosom):
    E=0
    for i in range(len(lst_pilihan_kartu_idx)):
        E+= abs(xkromosom[i]-lst_pilihan_kartu_idx[i])

    fitness =len(lst_all_kartu)*len(lst_pilihan_kartu_idx)-E

    return fitness,E
```

Gambar 4 Fungsi *fitness* dengan bahasa pemrograman Python

5. Seleksi

Metode seleksi yang digunakan pada penelitian ini adalah *roulette wheel*. Nilai Seleksi individu dilakukan berdasarkan nilai *fitness* yang diperoleh pada proses sebelumnya. Individu yang memiliki nilai *fitness* lebih besar akan mendapatkan porsi potongan yang lebih besar pada *roulette wheel* sehingga kemungkinan terpilih sebagai calon induk menjadi lebih besar daripada individu-individu yang lain.

Roulette wheel diputar sebanyak jumlah individu dalam populasi. Dengan cara ini akan didapatkan populasi baru dengan jumlah individu yang sama dengan populasi sebelumnya. Di dalam populasi baru tersebut memungkinakan munculnya individu yang sama karena terpilih lebih dari satu kali. Pada Gambar 5 ditunjukkan

cuplikan kode proses seleksi dengan *roulette wheel* menggunakan bahasa pemrograman Python.

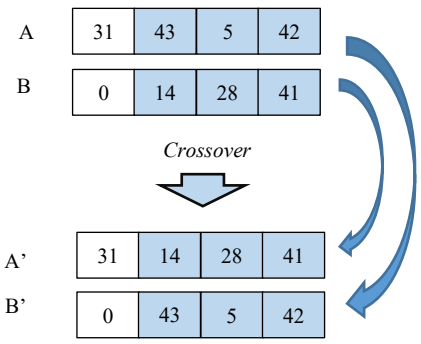
```
#Roulette Selection
lst_individu_terpilih=[]
for i in range(jumlah_individu_per_populasi):
    r = random.randrange(0, 100)
    for idx,pct_map in enumerate(lst_map_fitness_percent):
        if pct_map > r:
            lst_individu_terpilih.append(populasi[idx])
            break
```

Gambar 5 Proses seleksi dengan *Roulette Wheel* menggunakan pemrograman Python

6. *Crossover*

Pada bagian ini, induk di dalam populasi akan diseleksi lagi secara acak untuk proses pertukaran gen. Parameter yang digunakan untuk seleksi dinamakan nilai probabilitas *crossover*. Pada penelitian ini digunakan nilai probabilitas *crossover* sebesar 0.9. Dengan 100 individu per populasi, maka akan didapatkan 50 pasangan individu yang digunakan sebagai jumlah iterasi. Pada setiap iterasi ke-*i* dibangkitkan 100 bilangan random dari 0 hingga 1. Jika pada iterasi ke-*i* didapatkan nilai kurang dari nilai probabilitas *crossover*, maka pasangan ke-*i* akan mengalami pertukaran gen.

Gen mana saja yang akan ditukarkan oleh pasangan yang terpilih juga dilakukan secara acak. Sebagai contoh jika dari proses pembangkitan bilangan acak didapatkan nilai 2, maka gen ke-2 hingga gen terakhir dari pasangan akan saling ditukarkan. Sedangkan gen lainnya tidak mengalami pertukaran (Gambar 6). Individu hasil pertukaran tersebut menggantikan posisi induknya di dalam populasi.



Gambar 6 Proses *crossover* pada pasangan terpilih

Pada Gambar 7 ditunjukkan cuplikan kode pada proses *crossover* dengan menggunakan bahasa pemrograman Python.

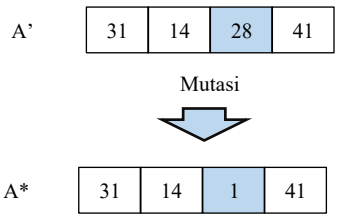
```
#Crossover
lst_individu_terpilih_cross = [lst[:] for lst in lst_individu_terpilih]
for i in range(jumlah_individu_per_populasi//2):
    r = random.randrange(0, 10)/10
    if r < probabilitas_crossover:
        r_pos_gen = random.randrange(0,jumlah_gen)

        # Temporary Parent
        parent1 = lst_individu_terpilih_cross[2*i].copy()
        parent2 = lst_individu_terpilih_cross[2*i+1].copy()

        if best_individu == parent1 or best_individu == parent2:
            continue;

        # Do CrossOver
        lst_individu_terpilih_cross[2*i]=parent1[0:r_pos_gen] + parent2[r_pos_gen:]
        lst_individu_terpilih_cross[2*i+1]=parent2[0:r_pos_gen] + parent1[r_pos_gen:]
```

Gambar 7 Proses *Crossover* dengan bahasa pemrograman Python

7. Mutasi

Pada proses ini dilakukan pemilihan individu yang akan dimutasi secara acak. Proses seleksi dilakukan dengan membangkitkan 1000 bilangan acak dari 0 hingga 1. Dengan jangkauan yang lebar tersebut, kemungkinan terjadinya mutasi individu menjadi sangat kecil. Parameter yang digunakan untuk proses seleksi mutasi dinamakan nilai probabilitas mutasi. Nilai probabilitas mutasi lebih kecil dari nilai probabilitas *crossover*. Pada penelitian ini digunakan nilai probabilitas mutasi sebesar 0,005. Apabila nilai acak yang didapatkan kurang dari nilai probabilitas mutasi, maka individu akan dimutasi. Posisi gen yang mengalami mutasi ditentukan secara acak dan dapat diganti dengan nilai gen lain yang didapatkan secara acak.

Misalkan individu A' yang didapatkan pada proses sebelumnya mengalami proses mutasi pada gen ke 3, hasil mutasi dapat menjadi seperti terlihat pada Gambar 8.



Gambar 8 Hasil mutasi pada individu terpilih

Pada Gambar 9 ditunjukkan cuplikan kode pada proses mutasi dengan menggunakan bahasa pemrograman Python.

```
#Mutasi
lst_individu_terpilih_mutasi = [lst[:] for lst in lst_individu_terpilih_cross]
for i in range(jumlah_individu_per_populasi):
    r = random.randrange(0,1000)/1000

    if best_individu == lst_individu_terpilih_mutasi[i][r_pos_gen]:
        continue;

    if r < probabilitas_mutasi:
        r_pos_gen = random.randrange(0,jumlah_gen)
        lst_individu_terpilih_mutasi[i][r_pos_gen] = random.randrange(0,len(lst_all_kartu))

return lst_individu_terpilih_mutasi,average_error,average_fitness,best_individu,best_fitness
```

Gambar 9 Prosse Mutasi dengan bahasa pemrograman Python

8. Pembentukan Generasi Baru

Individu-individu baru yang didapatkan melalui proses *crossover* dan mutasi merupakan generasi baru yang menggantikan populasi sebelumnya. Generasi baru ini dijadikan populasi baru yang akan menjalani proses seleksi, *crossover,* dan mutasi seperti yang dilakukan sebelumnya. Proses tersebut dilakukan secara berulang dengan nilai iterasi tertentu hingga didapatkan individu yang terbaik dan individu yang menjadi target ditemukan. Pada Gambar 10 ditunjukkan cuplikan kode pada proses pembentukan generasi baru dengan menggunakan bahasa pemrograman Python.

```
#Pemrosesan algoritma Genetika sejumlah generasi tertentu
for i in range(xjumlah_generasi-1):
    populasi_pilihan,avg_error,avg_fitness,best_individu,best_fitness = proses_GA(cur_generasi)
    cur_generasi = [gen[:] for gen in populasi_pilihan]
    lst_generasi.append(cur_generasi)
    lst_avg_error.append(avg_error)
    lst_avg_fitness.append(avg_fitness)
    lst_best_fitness.append(best_fitness)


    if(i%100==0):
        print(i,';',best_individu,';',decode_kromosom(best_individu),';',best_fitness,';', avg_fitness)
```

Gambar 10 Pembentukan generasi baru dengan bahasa pemrograman Python

D. *Pengujian*

Pengujian dilakukan dengan cara membandingkan antara individu target dengan individu terbaik yang memiliki nilai *fitness* tertinggi yang diperoleh melalui serangkaian proses algoritma genetika. Rata-rata nilai *fitness* untuk setiap generasi di-plot ke dalam grafik untuk melihat konvergensinya. Apabila nilai *fitness* konvergen pada nilai terbaik dan individu target berhasil ditemukan, maka proses optimasi dianggap berhasil.

III. HASIL DAN PEMBAHASAN

Berdasarkan penelitian yang dilakukan, didapatkan hasil seperti yang ditunjukkan pada TABEL 2. Dengan menggunakan algoritma genetika, kartu yang dicari yaitu As Sekop, 2 Keriting, 3 Hati, dan 4 Wajik dengan kode [0,14,28,42] berhasil ditemukan.

TABEL 2 Hasil dari Serangkain proses algoritma genetika untuk mencari urutan kartu dari tumpukan kartu yang disusun secara acak.

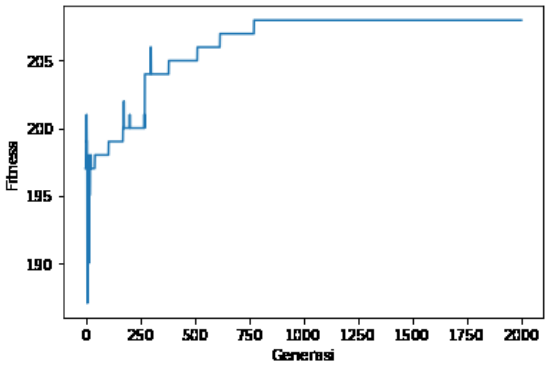
Generasi ke-	Individu Terbaik	*Fitness Terbaik*	Rata-rata *Fitness*
1	[2, 13, 31, 37]	197	135.72
101	[1, 17, 23, 43]	198	198
201	[0, 16, 23, 43]	200	200
301	[0, 16, 29, 43]	204	204
401	[0, 16, 28, 43]	205	204.52
501	[0, 16, 28, 43]	205	204.61
601	[0, 13, 28, 43]	206	205.56
701	[0, 13, 28, 42]	207	207
801	**[0, 14, 28, 42]**	**208**	**207.19**
901	[0, 14, 28, 42]	208	207.85
1001	[0, 14, 28, 42]	208	207.66
1101	[0, 14, 28, 42]	208	207.4
1201	[0, 14, 28, 42]	208	207.6
1301	[0, 14, 28, 42]	208	208
1401	[0, 14, 28, 42]	208	207.26
1501	[0, 14, 28, 42]	208	207.89
1601	[0, 14, 28, 42]	208	207.5
1701	[0, 14, 28, 42]	208	208
1801	[0, 14, 28, 42]	208	207.99
1901	[0, 14, 28, 42]	208	208

Pada TABEL 2 ditunjukkan 20 urutan sampel dengan jarak 100 iterasi dari total 2000 iterasi untuk pembentukan generasi. Dapat dilihat bahwa individu terbaik mulai muncul pada generasi ke 701 dan 801. Hal ini dapat juga dilihat pada Gambar 11, di mana pada generasi ke-750 mulai menunjukkan nilai *fitness* yang konvergen sebesar 208. Gambar 12 menunjukkan rata-rata nilai *fitness* dengan tren yang konvergen walaupun masih terlihat adanya *noise*.

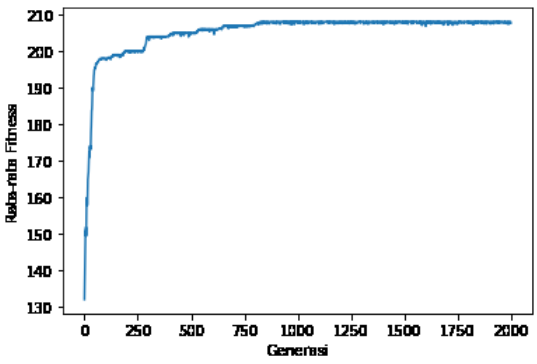
Proses pencarian kartu pada penelitian ini termasuk salah satu permasalahan kombinatorial, dalam hal ini adalah permutasi. Untuk menghitung jumlah permutasi $P(n,r)$ dari sejumlah sampel r dari sejumlah obyek n, digunakan persamaan berikut ini [4]:

$$P(n,r) = \frac{n!}{(n-r)!}$$

Apabila persamaan permutasi tersebut diimplementasikan pada kasus yang ingin dipecahkan pada penelitian ini, yaitu urutan dari sejumlah 4 sampel kartu ingin dicari dari tumpukan kartu sebanyak 52, maka jumlah kemungkinan urutan sampel yaitu sebanyak 6.497.400. Dengan algoritma genetika, target berhasil didapatkan pada generasi kurang dari 1000. Dapat disimpulkan bahwa algoritma genetika mampu melakukan optimasi pencarian urutan sampel kartu dari dalam tumpukan kartu yang disusun secara acak.



Gambar 11 Nilai f*itness* terbaik per Generasi



Gambar 12 Rata-rata nilai *fitness* terbaik per Generasi

IV. KESIMPULAN

Berdasarkan penelitian yang dilakukan, didapatkan hasil bahwa algoritma genetika dapat digunakan untuk mencari urutan sejumlah sampel kartu dari dalam tumpukan kartu yang disusun secara acak. Jumlah total kemungkinan untuk mendapatkan urutan 4 kartu dari 52 kartu adalah sebesar

6.497.400. Akan tetapi dengan menggunakan algoritma genetika proses tersebut dapat dioptimasi menjadi kurang dari 1000 iterasi.

REFERENCES

[1] A. Wahyu Widodo and W. Mahmudy, "Penerapan algoritma genetika pada sistem rekomendasi wisata kuliner," *Kursor*, vol. 5, pp. 205–211, Jan. 2010.

[2] G. Ucoluk, "A method for chromosome handling of r-permutations of n-element set in genetic algorithms," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, Apr. 1997, pp. 55–58, doi: 10.1109/ICEC.1997.592268.

[3] M. Kartika, "Pencocokan Kata Secara Acak Dengan Metode Algoritma Genetika Menggunakan Program Pascal," *Jurnal Matematika UNAND*, vol. 2, p. 1, Jun. 2013, doi: 10.25077/jmu.2.2.1-9.2013.

[4] R. Munir, "Kombinatorial, Bahan Kuliah IF2120 Matematika Diskrit," 2017.