Optimasi Pemilihan Barang Dagangan bagi Pedagang Keliling dengan Algoritma Genetika

Iryanto[1], Eka Ismantohadi[2]

[1, 2] Teknik Informatika Politeknik Negeri Indramayu, Jl. Raya Lohbener Lama No. 08 Indramayu 45252
E-mail : iryanto@polindra.ac.id[1], eka@polindra.ac.id[2]

Abstrak

Adanya kendala meningkatkan tingkat kesulitan dalam melakukan pemilihan barang, tidak terkecuali bagi seorang pedagang keliling. Tentunya seorang pedagang ingin mengoptimalkan keuntungan yang diperolehnya. Namun keterbatasan tempat dan modal menjadikan pedagang tersebut memerlukan strategi khusus dalam proses pemilihan barang yang dibeli dan dijajakannya. Terlebih lagi suatu barang dagangan memiliki peluang yang berbeda untuk terjual sehari-harinya. Hal tersebut semakin menambah kerumitan pemilihan tersebut. Dalam artikel ini, masalah tersebut diselesaikan dengan menggunakan algoritma genetika. Hasil percobaan dan solusi analitik menunjukkan kesesuaian yang baik. Percobaan dengan N-jenis barang juga dipaparkan dalam artikel ini.

Kata Kunci: Algoritma Genetika, Knapsack Problem, Optimasi Pemilihan Barang

Abstract

Existence of constraints increases difficulty in choosing the right goods for a seller. Indeed, the seller wants to optimize the profit gained. Due to limitation of capital and maximum capacity, the seller needs a certain strategy to do the selection. Moreover the fact that each goods has its own probability to sell makes the problem becomes more complex. In this paper, the problem is solved using genetic algorithm. The result of simulation is in a good agreement with analytical solution. The simulation of N-goods selection is also given in the paper.

***Keywords*:** *Genetic Algorithm, Knapsack Problem, Optimization of goods selection*

I. PENDAHULUAN

Pemilihan suatu barang dapat menjadi sulit ketika melibatkan batasan atau kendala tertentu. Misalnya, pemilihan barang dagangan bagi seorang pedagang keliling. Tentunya seorang pedagang ingin mengoptimalkan keuntungan yang diperolehnya. Namun keterbatasan tempat dan modal menjadikan pedagang tersebut memerlukan strategi khusus dalam proses pemilihan barang yang dibeli dan dijajakannya. Terlebih lagi suatu barang dagangan memiliki peluang yang berbeda untuk terjual sehari-harinya. Hal ini semakin menambah kerumitan pemilihan tersebut.

Permasalahan tersebut dikenal dengan istilah *Knapsack problem*. Dari berbagai literatur *Knapsack problem* dapat dimaknai sebagai masalah optimasi pemilihan barang yang memiliki nilai dan bobot dengan kendala bobot (kapasitas) maksimum dari tempat penampung barang-barang tersebut. Masalah tersebut erat kaitannya dengan kehidupan sehari-hari tidak terkecuali bagi para pedagang keliling.

Dalam bidang optimasi pemilihan barang, proses pemilihan dilakukan sampai mendapatkan hasil (solusi) terbaik dari semua kemungkinan solusi yang ada. Penelitian dalam bidang ini telah banyak dikaji di berbagai literatur. Beberapa penelitian terkait dalam bidang kajian tersebut adalah :

- Penggunaan algoritma genetika untuk optimasi pemilihan buah kemasan kotak dikaji secara rinci di [7]. Dalam hal ini optimasi keuntungan mempertimbangkan tingkat kebutuhan pasar dan keawetan buah tersebut dengan kendala total berat buah dan volum kotak. Hasil simulasi menujukkan bahwa jumlah populasi berperan penting dalam perolehan *fitness* terbaik.
- Pemaparan perbandingan dari berbagai proses/metode mutasi algoritma genetika dalam penyelesaian 0/1 *Knapsack problem* oleh Hasan, dkk dibahas di [3]. Dari 50 kali percobaan yang dilakukan proses mutasi dengan inversi menghasilkan hasil dengan rata-rata tertinggi meski perbedaan performa secara umum tidak terlalu berbeda dibandingkan metode mutasi lainnya yang dipaparkan dalam artikel tersebut. Selain itu, representasi biner adalah cara terbaik untuk merepresentasikan *Knapsack problem.*
- Penelitian penataan letak barang 3D pada kontainer dengan menggunakan algoritma genetika dikaji secara rinci di [1]. Dalam

penelitian ini, barang-barang tersebut berbentuk balok dan diletakkan hanya secara menyamping dengan dipisahkan berdasarkan tujuan, berat, volum dan jenis barang. Penelitian ini menghasilkan hasil yang menjanjikan.

- Implementasi algoritma genetika untuk optimasi persediaan barang dalam produksi jilbab di suatu perusahaan dipaparkan di [6]. Dalam artikel ini, nilai *fitness* ditentukan dengan meminimumkan biaya yang dibutuhkan selama proses produksi.
- Penggunaan algoritma genetika untuk menyelesaikan *Knapsack problem* juga dipaparkan di [8]. Dalam artikel ini fungsi objektif yang digunakan menyatakan optimasi total berat barang dengan kendala kapasitas maksimum dari tempat yang disediakan.

Dari pemaparan di atas, dapat dikatakan bahwa metode algoritma genetika baik digunakan untuk menyelesaikan *Knapsack problem.* Secara spesifik, dalam penelitian ini algoritma genetika tersebut diimplementasikan untuk menyelesaikan masalah pemilihan barang dagangan bagi pedagang keliling. Sehingga penelitian ini bertujuan untuk mengoptimalkan pemilihan barang dengan fungsi *fitness* dan kendala tertentu.

Adapun fungsi *fitness* dalam masalah ini adalah memaksimumkan keuntungan dengan mempertimbangkan nilai peluang suatu barang laku pada hari tertentu. Secara praktis nilai tersebut diperoleh dari riwayat penjualan atau dapat dilihat sebagai kecenderungan rasio antara jumlah barang yang laku terhadap jumlah total barang pada saat awal sebelum dijual.

Dalam penelitian ini kendala yang digunakan tidak hanya total bobot (berat) melainkan juga total biaya. Kedua kendala tersebut secara berurutan tidak boleh melebihi kapasitas maksimum tempat (gerobak) dan modal maksimum yang dimiliki pedagang setiap harinya.

Merujuk hasil penelitian yang disajikan di [3], algoritma genetika yang digunakan adalah algoritma genetika biner dengan proses mutasi inversi. Sedangkan dari referensi [7], nantinya akan digunakan jumlah populasi yang cukup besar dalam proses pencarian solusi dengan algoritma genetika tersebut.

Pemaparan mengenai algoritma genetika akan disajikan di Bab II. Proses implementasi dan perancangan sistem penyelesaian masalah tersebut akan dibahas lebih jauh di Bab III. Fungsi *fitness* dan kendala juga tersaji di bab tersebut. Hasil-hasil komputasi numerik akan diberikan di Bab IV. Penyajian ditutup dengan kesimpulan dan saran penelitian.

II. ALGORITMA GENETIKA

Algoritma genetika yang berkembang di saat sekarang dilatarbelakangi oleh penelitian yang dilakukan di Universitas Michigan yang dipimpin oleh John Holland di akhir tahun 1960an dan awal tahun 1970an [5]. Namun penggunaan algoritma genetika telah ada sejak awal tahun 1950an untuk simulasi masalah-masalah dalam bidang biologi [2].

Terinspirasi dari proses teori evolusi Darwin, algoritma genetika berkembang menjadi algortima komputasi untuk menyelesaikan masalah dengan proses-proses alamiah seperti proses reduksi, *crossover*, dan mutasi. Algoritma ini juga dikenal dengan istilah program evolusi [5] yaitu algoritma probabilistk yang mempertahankan individu-individu terbaik pada setiap generasinya.

Pada prakteknya nilai pobabilistik tersebut tercermin pada penggunaan bilangan acak. Nilai tersebut membuat solusi optimum global menjadi tidak pasti pula sehingga perancangan fungsi *fitness* dan pengaturan nilai parameter menjadi hal yang sangat diperlukan.

Adapun proses-proses pada algoritma genetika dalam menyelesaikan masalah optimasi dapat dinyatakan sebagai berikut (lihat [5] untuk lebih jelasnya) :

1. Representasi solusi yang merupakan pengkodean setiap individu
2. Pembangkitan populasi awal
3. Fungsi evaluasi yang dibuat berdasarkan masalah riil
4. Proses regenerasi (proses seleksi, *crossover*, dan mutasi)

III. IMPLEMENTASI DAN PERANCANGAN

Pada bab sebelumnya telah dijelaskan mengenai algoritma genetika berikut dengan rangkaian prosesnya. Pada bab ini hal-hal tersebut akan diimplementasikan untuk menyelesaikan masalah yang ada.

3.1 Representasi Solusi

- **Representasi Barang**
  Dalam hal ini barang direpresentasikan ke dalam empat buah array satu dimensi yaitu array peluang (menyatakan peluang barang tersebut terjual di hari tertentu), array bobot (menyatakan berat dari masing-masing barang), array biaya (menyatakan harga beli masing-masing barang), dan array keuntungan (menyatakan besarnya keuntungan tiap barang). Sehingga entri/isi dari array-array tersebut adalah nilai-nilai yang bersesuaian dengan keadaan barang yang sebenarnya seperti yang tersaji dalam Tabel 1.

Tabel 1 Daftar Barang

Barang	Peluang	Bobot	Biaya	Keuntungan
1	0,6	3	5.000	500
2	0,7	5	7.000	750
3	0,4	6	6.500	600
4	0,3	7	4.000	250
5	0,25	2	7.500	400
6	0,5	5	8.000	600
7	0,8	5	9.000	900
8	0,9	4	4.500	250
9	0,65	7	5.500	500

- **Kendala**
  Dalam masalah ini terdapat dua buah kendala yaitu berat total tidak boleh melebihi kapasitas

maksimum dan modal. Dalam hal ini jumlah total biaya pembelian barang tidak boleh melebihi batas modal yang dimiliki oleh pedagang tersebut.

- **Pengkodean Kromosom**
  Dalam algoritma genetika, kromosom yang tersusun atas gen-gen merepresentasikan calon solusi dari permasalahan yang ada [12].
  Pengkodean kromosom dinyatakan dalam array biner dua dimensi dengan jumlah baris sesuai dengan banyaknya populasi awal dan jumlah kolom sesuai dengan banyaknya barang yang akan dioptimasikan pemilihannya. Masing-masing gen dalam kromosom tersebut menyimpan informasi-informasi seperti pada poin representasi barang.

3.2 Pembangkitan Populasi Awal

Pembangkitan populasi awal dimulai dengan membangkitkan kromosom secara acak sebanyak jumlah populasi awal yang ditentukan.

3.3 Fungsi *Fitness*

Fungi *fitness* menyatakan tingkat kebugaran suatu kromosom. Fungsi ini digunakan untuk mengukur kelayakan suatu kromosom untuk dipertahankan [12].

Adapun fungsi *fitness* pada penelitian ini adalah fungsi yang memaksimumkan keuntungan dengan nilai peluang terjual dan kendala yang disebutkan pada poin pertama. Suatu kromosom dihitung nilai total keuntungan, biaya, dan beratnya. Kromosom dengan nilai keuntungan terbesar itulah yang diambil namun dengan total biaya dan berat tidak melebihi kendala. Jika total biaya atau berat melebihi kendala maka nilai keuntungan kromosom tersebut diatur menjadi nol.

$$fit(i) = \sum_{k=1}^{Pkrom} keuntungan_k * peluang_k$$

$$fobj = \underset{i \in Np}{maks}\ fit(i)$$

Dengan Np menyatakan jumlah populasi dan $Pkrom$ panjang kromosom dengan kendala:

$$\sum_{k=1}^{Pkrom} bobot_k \leq w\_maks \ \ \& \ \ \sum_{k=1}^{Pkrom} biaya_k \leq modal$$

3.4 Proses Regenerasi

- **Proses Seleksi**
  Dalam penelitian ini proses seleksi menggunakan *roulette wheel*. Prinsip dari metode seleksi ini adalah setiap segmen *roulette* ditempati oleh masing-masing kromosom/individu yang nantinya akan digunakan sebagai orang tua (*parent*) jika terpilih. Besaran segmen *roulette* sesuai dengan rasio nilai *fitness* tiap individu dengan total nilai *fitness* [4]. Semakin besar segmen *roulette* tentunya peluang individu tersebut terpilih juga semakin besar. Selain itu, pada proses seleksi ini juga digunakan juga *elitilism* untuk mempertahankan nilai *fitness* terbaik suatu generasi agar tidak turun di generasi selanjutnya [9]. Dalam implementasinya, *elitilism* tersebut dilakukan dengan menyalin kromosom/individu dengan *fitness* terbaik tersebut sebanyak yang diinginkan. Penentuan individu/kromosom dengan *fitness* terbaik dengan melakukan pengurutan nilai *fitness.*
  Adapun proses seleksi dengan *roulette wheel* dapat dinyatakan sebagai berikut:
  - Menghitung *fitness* tiap individu
    $$fit(i) = \sum_{k=1}^{Pkrom} keuntungan_k * peluang_k$$
  - Menghitung total nilai *fitness* untuk populasi
    $$Fobj = \sum_{i=1}^{Np} fit(i)$$
  - Menghitung *fitness* relatif setiap kromosom
    $$P(i) = \frac{fit(i)}{Fobj}, i = 1, 2, 3, \dots, Np$$
  - Menghitung *fitness* kumulatif setiap kromosom
    $$Q(i) = \sum_{j=1}^{i} P(j),\ i = 1, 2, 3, \dots, Np$$

- **Proses *Crossover***
  Proses crossover ini bertujuan untuk menghasilkan individu/kromosom baru dari dua individu terpilih yang nantinya dijadikan sebagai orang tua/*parent.* Banyaknya jumlah orang tua tersebut ditentukan oleh parameter Pc (peluang *crossover*). Semakin besar nilai dari parameter Pc tersebut artinya akan semakin banyak pula orang tua yang terpilih yang nantinya akan mempengaruhi tingkat kevariatifan populasi individu tersebut.
  Lebih lanjut, proses *crossover* yang digunakan adalah *two-point crossover*. Metode ini salah satu dari metode *crossover* yang paling sederhana namun banyak digunakan [10]. Rincian dari proses tersebut dapat dilihat di referensi tersebut. Namun demi kejelasan proses, berikut adalah langkah-langkah dalam melakukan proses *two-point crossover* :
  - Menentukan orang tua/*parent crossover* dengan cara membangkitkan bilangan acak $r = [0,1]$ dan $r \leq Pc$.
  - Menentukan batas-batas *crossover* dengan cara membangkitkan bilangan acak $r1, r2 = [1, PKrom]$, dengan $PKrom$ menyatakan panjang kromosom.
  - Misalkan $r1 < r2$, selanjutnya tukar gen-gen dari *parent* pertama dengan *parent* kedua pada indeks $i \in [r1, r2]$
  - Misalkan diberikan dua buah *parent* dengan panjang kromosom sembilan, *parent1* dan *parent2*, dan $r1 = 3, r2 = 7$

    Parent1

1	0	0	0	0	1	1	1	0

    Parent2

1	0	1	1	0	0	0	0	0

Sehingga hasil *crossover* dari dua *parent* tersebut adalah

Anak1

1	0	1	1	0	0	0	1	0

Anak2

1	0	0	0	0	1	1	0	0

- **Proses Mutasi**

  Proses mutasi ini dilakukan dengan cara menginverskan nilai dari gen. Dalam implementasinya, proses ini dilakukan dengan mengubah nilai gen (*allele*) 1 menjadi 0 dan 0 menjadi 1 [3]. Layaknya proses *crossover*, banyaknya gen yang mengalami mutasi dibatasi oleh suatu parameter tertentu. Dalam hal ini, parameter tersebut adalah Pm. Nilai Pm berpengaruh terhadap *fitness* yang dihasilkan. Nilai Pm yang besar dapat menyebabkan penurunan *fitness* suatu individu.

  Adapun tahapan-tahapan proses tersebut adalah sebagai berikut:

  - Menemtukan orang tua/*parent* mutasi dengan cara membangkitkan bilangan acak $r = [0,1]$ dan $r \leq Pm$.
  - Menginverskan nilai gen ke rr pada suatu kromosom dengan cara membangkitkan bilangan acak $rr = [1, PKrom]$, dengan $PKrom$ menyatakan panjang kromosom.
  - Misalkan diberikan *parent* dengan panjang kromosom sembilan dan nilai tiap gen sebagai berikut

1	0	1	1	0	0	0	0	0

Bilangan acak $rr = 6$ sehingga diperoleh hasil mutasi

1	0	1	1	0	1	0	0	0

3.5 Algoritma

Berikut adalah algoritma yang digunakan untuk menyelesaikan permasalahan yang ada

1. Inisialisasi parameter-parameter simulasi
2. Inisialisasi populasi awal secara acak
3. Hitung nilai fungsi objektif, *fitness* berat (*weight*), dan biaya
4. Tetapkan $fobj = 0$ jika kendala tidak terpenuhi
5. Pilih calon induk *crossover* dan mutasi
6. Lakukan proses *crossover*
7. Lakukan proses mutasi
8. Tetapkan kromosom-kromosom baru dalam populasi tersebut
9. Ulangi langkah 3 – 8 sampai batas evolusi maksimum

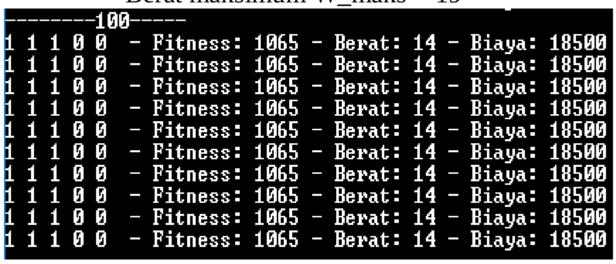
IV. HASIL DAN PEMBAHASAN

Validasi program/koding yang digunakan dilakukan dengan cara membandingkan hasil keluaran program dengan solusi analitik *Knapsack problem*. Dalam hal ini percobaan dilakukan dengan data yang tertera dalam Tabel 1. Percobaan pertama dilakukan untuk lima jenis barang pertama saja dan percobaan berikutnya semua data dalam Tabel 1 digunakan. Adapun solusi analitik diperoleh dari perhitungan manual dengan membandingkan nilai *fitness* untuk semua kemungkinan solusi. Misalnya, untuk kasus lima jenis barang akan terdapat $2^5 = 32$ calon solusi sedangkan untuk sembilan jenis barang akan terdapat $2^9 = 512$ kemungkinan solusi. Percobaan lainnya dilakukan untuk N-jenis barang yang cukup banyak. Pada percobaan ini waktu simulasi juga akan diamati.

4.1 Validasi Program

Percobaan dijalankan dengan parameter-parameter algoritma genetika sebagai berikut:

- Peluang mutasi Pm = 0.05
- Peluang *crossover* Pc = 0.8
- Banyaknya populasi awal Np = 15
- Elitilism = 10
- Evolusi maksimum = 100
- Modal = 25.000
- Berat maksimum W_maks = 19

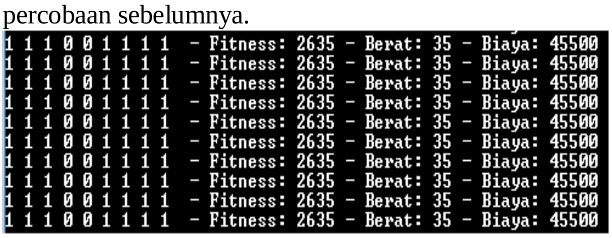


```
--------100-----
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
1 1 1 0 0  - Fitness: 1065 - Berat: 14 - Biaya: 18500
```

Gambar 1 Hasil Validasi 5 Jenis Barang

Berdasarkan Gambar 1 nilai *fitness* terbaik bernilai 1065 dengan total berat 14 dan total biaya 18.500 dengan rincian barang yang terpilih adalah barang pertama, kedua, dan ketiga. Hasil perhitungan analitik nilai *fitness* juga bernilai 18.500 dengan kromosom yang sama.

Percobaan selanjutnya menggunakan sembilan jenis barang dengan *elitilism* 90 dan banyaknya populasi 100, *elitilism* 25 dan banyaknya populasi 30, berat maksimum 35, dan modal 65.000. Parameter lainnya sama dengan percobaan sebelumnya.



```
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
1 1 1 0 0 1 1 1 1  - Fitness: 2635 - Berat: 35 - Biaya: 45500
```

Gambar 2 Hasil Validasi 9 Jenis Barang

Hasil perhitungan analitik diperoleh nilai *fitness* terbaik sebesar 2635 dengan rincian barang yang terpilih adalah barang pertama, kedua, ketiga, keenam, ketujuh, kedelapan, dan kesembilan. Baik percobaan dengan banyaknya populasi 100 maupun 30 juga sama-sama menghasilkan nilai *fitness* terbesar senilai 2635 dengan rincian keluaran program seperti yang disajikan di

Gambar 2. Namun dari 11 kali menjalankan program hasilnya berbeda-beda. Nilai rata-rata percobaan dengan 100 populasi sebesar 2.586 sedangkan dengan 30 populasi sebesar 2533.

Dari dua hasil validasi tersebut dapat dikatakan bahwa program tersebut menghasilkan keluaran yang masuk akal dan baik sehingga program yang ada dapat digunakan untuk menyelesaikan masalah yang ada.

4.2 Percobaan untuk N Jenis Barang

Dalam hal ini diperlukan pengaturan memori untuk memfasilitasi percobaan tersebut. Di sini alokasi memori dilakukan secara dinamis, *dynamic allocation memory*, lihat [11] untuk lebih jelasnya. Sebagai contohnya percobaan dilakukan dengan 200 jenis barang dan 250 populasi awal, *elitilisim* 225, bobot maksimum 1000, modal 1.300.000 serta nilai parameter algoritma genetika lainnya sama dengan nilai parameter yang digunakan di Subbab 4.1.

Nilai masing-masing array peluang, bobot, biaya, dan keuntungan dibangkitkan secara acak memenuhi

$$r(i) = random[0,1]$$
$$peluang(i) = r(i)$$
$$bobot(i) = r(i) * 10$$
$$keuntungan(i) = r(i) * 1.000$$
$$biaya(i) = r(i)*10.000$$

Dari 11 kali menjalankan program, diperoleh hasil *fitness* terbaik sebesar 73.149 dengan rincian kromosom seperti pada Gambar 3. Adapun nilai rata-rata *fitness* dari 11 kali menjalankan program tersebut adalah 67.795,27.

Gambar 3 Hasil N-Jenis Barang

Selain itu, waktu simulasi menujukkan perbedaan yang signifikan. Dapat dilihat pada Gambar 3 waktu simulasi percobaan tersebut adalah 4,328 detik sedangkan waktu simulasi untuk percobaan dengan 100 populasi dengan 5 dan 9 jenis barang secara berturut-turut bernilai 0,062 detik dan 0,093 detik. Hal ini menunjukkan bahwa banyaknya populasi dan panjang kromosom berpengaruh terhadap waktu komputasi.

V. PENUTUP

Kesimpulan

Penyelesaian masalah pemilihan barang dagangan bagi pedagang keliling telah dipaparkan dengan lengkap. Untuk pemilihan barang dengan data seperti pada Tabel 1, barang yang terpilih adalah barang pertama, kedua, ketiga, keenam, ketujuh, kedelapan, dan kesembilan dengan nilai *fitness* sebesar 2635. Hasil percobaan dengan algoritma genetika tersebut menunjukkan kesesuaian yang baik dengan solusi analitik. Pada percobaan N-jenis barang diperoleh hasil *fitness* terbaik sebesar 73.149 dengan nilai rata-rata *fitness* sebesar 67.795,27. Selain itu pada percobaan ini terjadi peningkatan waktu komputasi yang signifikan.

Saran

Saran yang dapat penulis sampaikan adalah verifikasi hasil pada percobaan N-jenis barang, misalnya dengan melakukan studi kasus tertentu. Saran lainnya adalah paralelisasi program untuk menurunkan waktu simulasi terutama untuk kasus pemilihan N-jenis barang.

VI. DAFTAR PUSTAKA

[1] Farosanti, Lafnidita. "Simulasi 3D optimasi penataan barang pada kontainer menggunakan Algoritma Genetika." Skripsi. Universitas Islam Negeri Maulana Malik Ibrahim, 2015.

[2] Goldberg, David E., and Robert Lingle. "Alleles, loci, and the traveling salesman problem." Proceedings of an International Conference on Genetic Algorithms and Their Applications, vol. 154. Lawrence Erlbaum, Hillsdale, NJ, 1985.

[3] Hasan, Basima Hani F., and Moutaz Saleh M. Saleh. "Evaluating the effectiveness of mutation operators on the behavior of genetic algorithms applied to non-deterministic polynomial problems." Informatica, 2011, vol. 35(4), pp. 513-519.

[4] Kumar, Rakesh. "Blending roulette wheel selection & rank selection in genetic algorithms." International Journal of Machine Learning and Computing, 2012, vol. 2(4), pp. 365-370.

[5] Michalewicz, Zbigniew. "Genetic Algorithms+ Data Structures= Evolution Programs." Springer Berlin Heidelberg, 1996.

[6] Ramuna, Maretta Dwi Tika, and Wayan Firdaus Mahmudy. "Optimasi Persediaan Barang Dalam Produksi Jilbab Menggunakan Algoritma Genetika." DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya, 2015, vol. 5(14), pp. 1-10.

[7] Setemen, Komang. "Implementasi Algoritma Genetika Pada Knapsack Problem Untuk Optimasi Pemilihan Buah Kemasan Kotak." Seminar Nasional Aplikasi Teknologi Informasi (SNATI), 2010, pp. 21-25.

[8] Supriana, I. Wayan. "Optimalisasi Penyelesaian Knapsack Problem Dengan Algoritma Genetika." Lontar Komputer: Jurnal Ilmiah Teknologi Informasi, 2016, vol. 7(3), pp. 858-868.

[9] Dyer, DW. "Evolutionary Computation in Java: A Practical Guide to the Watchmaker Framework." 2008,http://watchmaker.uncommons.org/manual/ch03s06.html. Diakses 29 Januari 2017.

[10] Sastry, Kumara, David E. Goldberg, and Graham Kendall. "Genetic algorithms." Search methodologies. Springer US, 2014, pp. 93-117.

[11] Séméria, Luc, Koichi Sato, and Giovanni De Micheli. "Resolution of dynamic memory allocation and pointers for the behavioral synthesis form C." Proceedings of the conference on Design, automation and test in Europe. ACM, 2000, pp 312-319.

[12] Mitchell, Melanie. "An introduction to genetic algorithms." MIT press, 1998.