 ISSN: 2684-8473

Teknik-teknik optimasi *knapsack problem*

Riri Nada Devita[1], Aji Prasetya Wibawa[2]

Jurusan Teknik Elektro, Universitas Negeri Malang, Indonesia
[1] ririnadadevita@gmail.com; [2] aji.prasetya.ft@um.ac.id

INFORMASI ARTIKEL

Histori Artikel
Diterima : 8 Februari 2020
Direvisi : 22 Februari 2020
Diterbitkan : 4 April 2020

Kata Kunci:
Algoritma
Knapsack
Optimasi

ABSTRAK

Optimasi merupakan sebuah teknik yang identik dengan memaksimalkan sumber daya yang terbatas. Salah satunya adalah permasalahan nyata yang memperlukan teknik optimasi adalah cara mengatur barang-barang yang dimuat dalam suatu knapsack (karung/ Tas). Knapsack problem merupakan masalah dimana orang dihadapkan pada persoalan optimasi pemilihan benda yang dapat di tampung ke dalam sebuah knapsack (karung) yang memiliki keterbatasan daya dan ruang tampung. Oleh karena itu, dengan adanya optimasi dalam pemilihan barang yang akan ditampung dalam knapsak tersebut diharapkan dapat menghasilkan efisiensi yang maksimal. Oleh karena itu, paper ini bertujuan untuk mendiskusikan beberapa algoritma optimasi yang sesuai untuk masalah knapsack. Hasil dari studi ini menunjukkan bahwa algoritma Dynamic Programming adalah algoritma yang paling sesuai dalam penyelesaian masalah knapsack karena menghasilkan solusi yang optimum dan waktu running yang tidak lama.

I. Pendahuluan

Pengertian optimasi berhubungan erat dengan maksimasi, tetapi dengan batasan yang *constrain*. Mengoptimumkan identik dengan memaksimum-kan dengan sumber daya terbatas [1]. Dalam kehidupan sehari hari, banyak ditemukan permasalahan yang menyangkut permasalahan optimasi. Salah satunya adalah permasalahan cara mengatur barang-barang yang dimuat dalam suatu tempat atau dianalogikan akan disimpan dalam suatu *knapsack* (karung/Tas).

Knapsack adalah tas atau karung yang digunakan untuk memuat sesuatu yang tentunya tidak semua objek dapat ditampung di dalam karung tersebut. Karung tersebut hanya dapat menyimpan beberapa objek dengan total ukuran *(weight)* lebih kecil atau sama dengan ukuran kapasitas karung [2]. *Knapsack* problem merupakan masalah dimana orang dihadapkan pada persoalan optimasi pemilihan benda yang dapat di tampung ke dalam sebuah *knapsack* (karung) yang memiliki keterbatasan daya dan ruang tampung. Oleh karena itu, dengan adanya optimasi dalam pemilihan barang yang akan ditampung dalam *knapsak* tersebut diharapkan dapat menghasilkan keuntungan yang maksimum.

Saat ini, terdapat banyak algoritma yang dapat digunakan dalam permasalahan optimasi *knapsack*, diantaranya adalah algoritma yang dijabarkan pada jurnal *Permasalahan Optimasi 0-1 Knapsack dan Perbandingan beberapa Algoritma Pemecahannya* [3]. Dalam jurnal tersebut dibandingkan beberapa algoritma optimasi untuk permasalahan *knapsack* diantaranya adalah algoritma *Brute Force* yang memiliki sifat optimal akan tetapi lama dalam waktu *running* [4][5], algoritma *Greedy* memiliki sifat cepat tapi tidak optimal [6][7], dan algoritma *Dynamic Programming* memiliki sifat optimal dan agak cepat dalam waktu *running* [4][8]. Beberapa algoritma di atas merupakan algoritma optimasi yang sering digunakan untuk menyelesaikan masalah *knapsack.* Terdapat banyak algoritma optimasi yang umumnya tidak diketahui oleh orang awam, oleh karena itu pada penelitian ini akan dibahas berbagai macam algoritma optimasi yang dapat digunakan untuk menyelesaikan *knapsack problem.* Tujuan dari penelitian ini adalah menyediakan sebuah referensi sebagai bahan pertimbangan untuk menentukan algoritma yang paling sesuai dalam penyelesaian masalah *knapsack.*

II. Kajian Pustaka

A. Optimasi

Optimasi adalah menemukan solusi yang berada dalam daerah yang mungkin *(feasible region)* yang memiliki nilai minimum atau maksimum dari fungsi objektif [9]. Optimasi merupakan permasalahan

komputasional yang bertujuan untuk menemukan solusi terbaik dari beberapa solusi yang mungkin dari sejumlah alternatif solusi dengan memenuhi sejumlah batasan *(constraints)*.

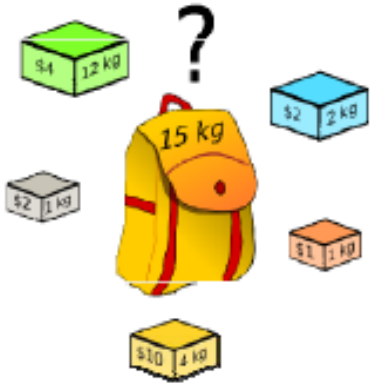
B. Knapsack Problem

Pada metode MSB, pesan disisipkan pada *bit* ke-1. Sebagai contoh, misalkan tiga piksel yang berdekatan (sembilan *bytes*) dengan kode RGB seperti pada metode LSB+1 akan disisipkan adalah karakter "R" dengan menggunakan metode MSB, maka akan dihasilkan citra hasil dengan urutan *bit* sebagai berikut:

Knapsack adalah tas atau karung yang digunakan untuk memuat sesuatu yang tentunya tidak semua objek dapat ditampung di dalam karung tersebut. Karung tersebut hanya dapat menyimpan beberapa objek dengan total ukuran *(weight)* lebih kecil atau sama dengan ukuran kapasitas karung [10][11]. Ilustrasi *knapsack* dapat dilihat pada Gambar 1. Terdapat sebuah tas yang memiliki kapasitas sebesar 15 kg dan terdapat 5 barang dengan berat dan keuntungan masing-masing, yang menjadi persoalan adalah barang mana saja yang harus dimasukkan ke dalam tas.

Masalah *knapsack* merupakan masalah yang nyata dalam kehidupan sehari-hari. Permasalahan *knapsack* ini sering digunakan pada bidang jasa pengangkutan barang seperti pengangkutan peti kemas dalam sebuah media pengangkut. Dalam kegiatan tersebut, diinginkan keuntungan yang maksimal untuk mengangkut semua barang dengan tidak melebihi kapasitas yang ada. *Knapsack* sendiri terdiri dari beberapa persoalan, yaitu:

- Knapsack 0-1 (integer knapsack) yaitu, objek yang dimasukkan ke dalam media penyimpanan, dimana dimensinya harus dimasukkan semua atau tidak sama sekali.
- Bounded Knapsack, yaitu objek yang dimasukkan ke dalam media penyimpanan, dimana dimensinya bisa dimasukkan sebagian atau seluruhnya.
- Unbounded knapsack, yaitu jumlah objek yang dimasukkan ke dalam media penyimpanan yang macamnya tidak terbatas [5][12].



Gambar. 1. Ilustrasi *Knapsack Problem*

Knapsack problem mempunyai total ukuran atau kapasitas yang disimbolkan dengan *V*, ada *n* jenis barang berbeda yang dimasukkan dalam *knapsack*. Barang ke-*i* mempunyai bobot v_i serta profit b_i. X_i merupakan total barang ke-*i* yang ditempatkan pada *knapsack* [13]. Fungsi tujuan dari permasalahan *knapsack* adalah:

$$\sum_{i=1}^{\pi} b_i . v_i \quad (1)$$

dengan *constraint*:

$$\sum_{i=1}^{\pi} v_i . X_i \leq V_i \quad (2)$$

Formulasi yang paling sering dan umum dari masalah *knapsack* adalah *Knapsack Problem* 0-1. Hal itu membatasi kemungkinan pilihan yang diambil antara 1 atau 0 [14][15].

III. Algoritma Optimasi

Terdapat banyak algoritma optimasi yang telah ada, tetapi pada penelitian ini hanya akan dibahas beberapa algoritma optimasi yang sering digunakan untuk menyelesaikan masalah *knapsack*. Berikut adalah beberapa algoritma optimasi yang sering digunakan untuk menyelesaikan *knapsack*.

A. Algoritma Greedy

Metode *greedy* dapat diartikan rakus/tamak [6]. Algoritma *greedy* merupakan solusi langkah per langkah dan pada setiap langkahnya banyak pilihan yang perlu dieksplorasi oleh karena itu pada setiap langkah perlu dibuat keputusan yang terbaik dalam menentukan pilihan, dan diharapkan langkah sisanya akan mengarah ke optimum global (*global optimum*) [16]. Tahap penyelesaian algoritma *greedy* adalah:

- Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan.
- Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global [2].

Pada penyelesaian *Knapsack Problem* terdapat 3 jenis algoritma *Greedy* yang dapat digunakan. Berikut adalah penjelasannya:

- *Greedy by profit*. *Knapsack* diisi dengan objek yang mempunyai keuntungan terbesar pada setiap tahap.
- *Greedy by Weights*. *Knapsack* diisi dengan objek yang mempunyai berat paling ringan pada setiap tahap.
- *Greedy by density*. *Knapsack* diisi dengan objek yang mempunyai densitas terbesar pada setiap tahap [6].

Penelitian menggunakan algoritma *greedy* untuk penyelesaian *knapsack problem* pernah dilakukan oleh Gazali dan Manik pada tahun 2010, untuk penyusunan barang dalam kontainer [16]. Menurut penelitian tersebut algoritma *greedy* mampu mencari solusi optimal dalam permasalahan *Three dimensional container loading problem*, dan waktu yang dibutuhkan untuk melakukan proses optimalisasi berbanding lurus dengan jumlah barang yang diinputkan.

B. Algoritma Dynamic Programming

Program dinamis *(Dynamic Programming)* adalah suatu teknik matematis digunakan untuk membuat suatu keputusan dari serangkaian keputusan yang saling berkaitan [17]. Strategi *dynamic programming* adalah membangun masalah optimasi bertingkat, yaitu masalah yang dapat digambarkan dalam bentuk serangkaian tahapan (*stage*) yang saling mempengaruhi. *Dynamic programming* memiliki karakteristik *overlapping subproblem* dan *optimal substructure* [18][19]. Metode Program Dinamis dibagi menjadi dua jenis penyelesaian yaitu Program Dinamis perhitungan rekursif maju dan rekursif mundur. Langkah penyelesaian Program Dinamis perhitungan rekursif maju dimulai dari iterasi ke-1 sampai iterasi ke- , dan penyelesaian Program Dinamis perhitungan rekursif mundur yaitu dari ke- sampai iterasi ke-1 [20]. Pembangunan algoritma *dynamic programming* dapat dipecah menjadi empat langkah yang berurutan yaitu:

- Karakterisasi struktur dari solusi optimal.
- Mendefinisikan nilai dari solusi optimal secara rekursif.
- Menghitung nilai dari solusi optimal secara *bottom-up*.
- Membuat sebuah solusi optimal berdasarkan informasi hasil komputasi [14].

Penelitian menggunakan algoritma *dynamic programming* pernah dilakukan oleh Prasetiyowati dan Wicaksana pada tahun 2013 untuk menyelesaikan *Multiple Constraint Knapsack Problem* [14]. Dalam penelitian tersebut telah dibuktikan bahwa algoritma *dynamic programming* terbukti telah menghasilkan solusi optimal yang akurat, hal ini diketahui dari pengujian setiap variabel. Sedangkan untuk kecepatan waktu optimasi algoritma *dynamic programming* berbanding lurus dengan banyaknya inputan, tetapi telah tercatat memiliki rata-rata waktu yang cepat, sekitar 29,932 *micro second* untuk inputan sebanyak 20 data.

C. Algoritma Branch and Bound

Algoritma *Branch and Bound* merupakan metode pencarian di dalam ruang solusi secara sistematis. Ruang solusi diorganisasikan ke dalam pohon ruang status yang dibangun secara dinamis berdasarkan skema DFS *(Depth First Search)*, maka pada algoritma *Branch and Bound* ruang solusi dibangun dengan skema BFS *(Breadth First Search)*. Pada algoritma ini, permasalahan dibagi bagi menjadi *subregion-subregion* yang mungkin mengarah ke solusi [21].

Algoritma *Branch and Bound* menentukan keuntungan maksimal pada tiap simpulnya, proses pencarian akan menemukan solusi optimum, karena tidak semua objek pada permasalahan ini dapat dimasukkan ke dalam *knapsack*, maka kemungkinan solusi akan sampai pada keadaan dimana tidak ada lagi simpul yang dapat dibangkitkan karena telah melewati batas kapasitas daya angkut membuat kita harus menentukan solusi optimum dengan membandingkan lintasan-lintasan mana yang berakhir di daun pada pohon yang akan menghasilkan keuntungan paling besar maka objek-objek tersebutlah yang akan dipilih untuk dimasukkan ke dalam *knapsack*/gerobak.

Penyelesaian *knapsack problem* dengan menggunakan algoritma *Branch and bound* telah dilakukan oleh A. S. Permata pada tahun 2007 [22]. Dalam penelitiannya, Permata menyimpulkan bahwa algoritma *branch and bound* kurang mangkus untuk diterapkan sebagai solusi permasalahan *knapsack*, karena algoritma ini tidak memperhitungkan semua kemungkinan yang ada sehingga akan mengefisienkan waktu pemrosesan, tetapi tidak diketahuinya batasan untuk menemukan solusi optimal.

D. Algoritma Brute Force

Brute Force adalah sebuah pendekatan langsung (*straight forward*) untuk memecahkan suatu masalah, yang biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Pada dasarnya algoritma *Brute Force* adalah alur penyelesaian suatu permasalahan dengan cara berpikir yang sederhana dan tidak membutuhkan suatu pemikiran yang lama karena pada dasarnya alur pikir manusia adalah langsung atau *to the point* [6]. Prinsip pencarian solusi permasalahan *Knapsack 0/1* menggunakan algoritma *brute force* adalah:

- Mengenumerasikan *list* semua himpunan bagian dari himpunan dengan *n* objek.
- Menghitung total keuntungan dari setiap himpunan bagian dari langkah 1.
- Memilih himpunan bagian yang menerbitkan total keuntungan terbesar [23].

Algoritma *brute force* telah dimanfaatkan oleh Fitriana pada tahun 2009 untuk penelitian *Perbandingan beberapa algoritma knapsack 0-1* [3]. Di antara algoritma yang dibahas pada penelitian Fitriana, algoritma *brute force* terbukti mampu memberikan solusi paling optimal, tetapi kompleksitas *brute force* paling buruk di antara algoritma lain, sehingga memperlambat proses pencarian solusi optimum.

E. Algoritma Genetika

Algoritma genetika dalam penyelesaian masalah meniru teori evolusi makhluk hidup [24]. Adapun komponen-komponen algoritma genetika tersusun dari populasi yang terdiri dari kumpulan individu-individu yang merupakan calon solusi dari permasalahan *knapsack* [13]. Secara umum, dalam algoritma genetika terdapat lima proses, yaitu:

- Pembentukan Populasi Awal.
- Perhitungan Nilai *Fitness*.
- Seleksi.
- Regenerasi (*Crossover* dan Mutasi),
- Penciptaan Populasi Baru Hasil Regenerasi [25].

Pada tahun 2010, Komang Setemen telah melakukan penelitian dengan memanfaatkan algoritma genetika pada *knapsack problem* untuk optimasi pemilihan buah kemasan kotak [25]. Hasil penelitian tersebut menunjukkan bahwa algoritma genetik cukup baik digunakan pada *knapsack problem.* Sedangkan pada tahun yang sama, penelitian yang dilakukan Diah *et al.* dengan algoritma yang sama pula menunjukkan hasil optimasi menggunakan algoritma genetik masih mendekati solusi optimalnya.

IV. Perbandingan Algoritma Optima

Dari penjelasan setiap algoritma di atas, maka dapat diketahui perbandingan pada setiap algoritma optimasi dalam penyelesaian *knapsack problem.* Tabel 1 menunjukkan perbandingan dari setiap algoritma optimasi. Dari kajian pustaka yang telah dilakukan, diketahui bahwa setiap algoritma optimasi dalam penyelesaian *knapsack problem* memiliki kelemahan dan kelebihan tersendiri. Akan tetapi algoritma *dynamic programming* memberikan solusi yang lebih optimal dari algoritma lain. Hal ini terbukti dalam penelitian yang dilakukan Fitriana pada tahun 2009, Fitriana melakukan penelitian dengan membandingkan beberapa algoritma optimasi untuk *knapsack problem*, dan hasilnya menunjukkan bahwa algoritma *dynamic programming* yang paling cocok dan mangkus dalam penyelesaian *knapsack problem*. Hal ini pun diakui oleh banyak penelitian lain, salah satunya adalah penelitian yang dilakukan Prasetiyowati dan Wicaksana untuk menyelesaikan *multiple constraint knapsack problem*. Prasetiyowati dan Wicaksana mengungkapkan bahwa algoritma *dynamic programming* mampu menghasilkan solusi optimal yang akurat dan memiliki waktu pemrosesan yang tergolong cepat.

Tabel 1 Kelebihan dan Kekurangan Algoritma

Algoritma	Kelebihan	Kekurangan
Greedy	• Pemrosesan cepat [16] • Kompleksitas waktu $O(n^2)$ [26]	• Tidak selalu menghasilkan solusi optimal [3]
Dynamic Programming	• Menghasilkan solusi optimal [3][26].	• Waktu pemrosesan standart [27] • Sulit di implementasikan [26]
Branch and Bound	• Memiliki tingkat kesalahan yang sedikit	• Tidak selalu menghasilkan solusi optimal [3] • Waktu pemrosesan lama [22]
Brute Force	• Selalu menghasilkan solusi optimal [3] • Mudah diimplementasikan [26]	• Waktu pemrosesan sangat lama [3] • Kompleksitas waktu eksponensial yaitu $O(n.2^n)$ [26]
Genetika	• Waktu pemrosesan algoritma cepat [3] • Iterasinya sedikit [27]	• Ketidakpastian untuk menghasilkan solusi optimum

V. Kesimpulan

Berdasarkan studi yang telah dilakukan dapat diketahui bahwa terdapat beberpa algoritma yang dapat digunakan untuk permasalahan *knapsack*. Setiap algoritma optimasi tersebut memiliki kelemahan dan kelebihan tersendiri. Namun dapat diasumsikan bahwa algoritma yang paling sesuai adalah *dynamic programming*. Pemilihan algoritma ini berdasarkan dari karakteristik permasalahan *knapsack*. Selain itu banyak penelitian sebelumnya yang menyatakan bahwa algoritma *dynamic programming* merupakan algoritma yang paling sesuai untuk permasalahan *knapsack*. Hasil dari studi ini diharapkan dapat menjadi bahan pertimbangan dan dapat menambah referensi untuk penelitian terkait selanjutnya.

Daftar Pustaka

[1] M. Hilmi, A. Masrevaniah, and W. Soetopo, "Optimasi Pola Operasi Waduk Pelaparado Di Kabupaten Bima," *J. Tek. Pengair.*, vol. 3, pp. 132–142, 2012.

[2] Paryati, "Optimasi Strategi Algoritma Greedy Untuk Menyelesaikan Permasalahan Knapsack 0-1," *Yogyakarta, UPN Veteran*, vol. 2009, no. semnasIF, pp. 101–110, 2009.

[3] F. Passa, "Permasalahan Optimasi 0-1 Knapsack dan Perbandingan Beberapa Algoritma Pemecahannya," pp. 1–6, 2009.

[4] D. K. Ningtyas, V. Evania, and Ernastuti, "Evaluasi Kinerja Algoritma Traveling Salesman Problem dengan Teknik Pemrograman Dinamik," in *Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008)*, 2008, pp. 143–146.

[5] D. J. Surjawan and I. Susanto, "Aplikasi Optimalisasi Muat Barang dengan Penerapan Algoritma Dynamic Programming pada Persoalan Integer Knapsack," *J. Tek. Inform. dan Sist. Inf.*, vol. 1, no. 2, pp. 151–162, 2015.

[6] D. Rachmawati and A. Candra, "Implementasi Algoritma Greedy untuk Menyelesaikan Masalah Knapsack Problem," *J. Ilm. Saintikom*, vol. 12, no. 3, pp. 185–192, 2013.

[7] D. T. Wijayanti, "Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem," *J. Transform.*, vol. 11, no. 1, pp. 1–6, 2013.

[8] A. Mulyanto, "ANALISIS EDIT DISTANCE MENGGUNAKAN ALGORITMA DYNAMIC PROGRAMMING," vol. 5, no. 2, 2010.

[9] H. A. Saputro, W. F. Mahmudy, and C. Dewi, "Implementasi Algoritma Genetika Untuk Optimasi Penggunaan Lahan Pertanian Dengan," *J. Mhs. PTIIK Univ. Brawijaya*, vol. 5, no. 12, pp. 1–12, 2015.

[10] a. J. Umbarkar and M. S. Joshi, "0/1 Knapsack Problem using Diversity based Dual Population Genetic Algorithm," *Int. J. Intell. Syst. Appl.*, vol. 6, no. 10, pp. 34–40, 2014.

[11] A. Lamine, M. Khemakhem, and H. Chabchoub, "Knapsack Problems involving dimensions , demands and multiple choice constraints : generalization and transformations between formulations," *Int. J. Adv. Sci. Technol.*, vol. 46, pp. 71–94, 2012.

[12] K. D. KW, M. Fadhli, and C. Sutanto, "Penyelesaian Knapsack Problem menggunakan Algoritma Genetika," in *Seminar Nasional Informatika 2010 (semnasIF 2010)*, 2010, pp. 28–33.

[13] I. W. Supriana, "Optimalisasi Penyelesaian Knapsack Problem dengan Algoritma Genetika," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 7, no. 3, pp. 182–192, 2016.

[14] M. I. Prasetiyowati and A. Wicaksana, "Implementasi Algoritma Dynamic Programming untuk Multiple Constraints Knapsack Problem," in *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*, 2013, pp. 6–13.

[15] S. Kosasi, "PENYELESAIAN BOUNDED KNAPSACK PROBLEM MENGGUNAKAN DYNAMIC PROGRAMMING (Studi Kasus : CV . Mulia Abadi)," vol. 8, no. 2, pp. 35–43, 2013.

[16] W. Gazali and N. I. Manik, "Perancangan Program Simulasi Optimasi Penyusunan Barang dalam Kontainer menggunakan Algoritma Greedy," *J. Mat Stat*, vol. 10, no. 2, pp. 100–113, 2010.

[17] T. Y. Anggun and A. Munawar, "Analisis Sistem Transportasi Sampah Kota Tuban menggunakan Dynamic Programming," *J. Ilm. Tek. Lingkung.*, vol. 6, no. 1, pp. 45–52, 2014.

[18] B. Bhowmik, M. Mieee, Miacsit, Miaeng, Mpass, Miaoe, and Lecturer, "Dynamic Programming – Its Principles, Applications, Strengths, and Limitations," *Biswajit Bhowmik / Int. J. Eng. Sci. Technol.*, vol. 2, no. 9, pp. 4822–4826, 2010.

[19] N. S. Tung, A. Bhadoria, K. Kaur, and S. Bhadauria, "Dynamic Programming Model based on Cost Minimization Algorithms for Thermal Generating Units," *Int. J. All Res. Educ. Sci. Methods*, vol. 1, no. 1, pp. 19–27, 2013.

[20] H. Yunus and S. Martha, "Metode Program Dinamis Pada Penyelesaian Traveling Salesman Problem," vol. 04, no. 3, pp. 329–336, 2015.

[21] E. N. Hayati, "Aplikasi Algoritma Branch and Bound untuk Menyelesaikan Integer Programming," *Din. Tek.*, vol. 4, no. 1, pp. 13–23, 2010.

[22] A. S. Permata, "Pemecahan Masalah Knapsack dengan Menggunakan Algoritma Branch and Bound," 2007.

[23] H. Rija, M. Fadhli, and W. N. Sikumbang, "Aplikasi Agenda Latihan di Pusat Kebugaran dengan Pendekatan Algoritma Brute Force berbasis Android," *J. Aksara Komput. Terap.*, vol. 1, no. 1, pp. 1–10, 2012.

[24] T. N. Panggabean, "Analisis Tingkat Optimasi Algoritma Genetika Dalam Hukum Ketetapan Hardy-Weinberg Pada Bin Packing Problem," *CESSJournal Comput. Eng. Syst. Sci.*, vol. 1, no. 2, pp. 12–18, 2016.

[25] K. Setemen, "Implementasi Algoritma Genetika pada Knapsack Problem untuk Optimasi Pemilihan Buah Kemasan Kotak," in *Seminar Nasional Aplikasi Teknologi Informasi 2010 (SNATI 2010)*, 2010, pp. 21–25.

[26] P. A. Wicaksono, "Eksplorasi Algoritma Brute Force , Greedy Dan Pemrograman Dinamis Pada Penyelesaian Masalah 0 / 1 Knapsack," pp. 1–5, 2007.

[27] V. Thada and S. Dhaka, "Genetic Algorithm based Approach to Solve Non Fractional (0/1) Knapsack Optimization Problem," *Int. J. Comput. Appl.*, vol. 100, no. 15, pp. 21–26, 2014.