**Komparasi Algoritma *Dynamic Programming* dan Algoritma *Greedy* pada Permasalahan *Integer Knapsack* di Transportasi Angkut Barang**

**Global Ilham Sampurno1, Endang Sugiharti2, Alamsyah3**

1,2,3Jurusan Ilmu Komputer, FMIPA, Universitas Negeri Semarang

Email: 1 globalilhams@students.unnes.ac.id, 2endangsugiharti@mail.unnes.ac.id, 3Alamsyah@mail.unnes.ac.id

**Abstrak**

Pada saat ini pengiriman barang menjadi hal yang tidak asing lagi, karena penggunaan jasa pengiriman barang sangat memudahkan pelanggan. pt pos indonesia adalah salah satu penyedia jasa pengiriman barang. Pada pengiriman barang, sering kita jumpai pemilihan barang yang akan dimasukan ke transportasi terlebih dahulu dan yang ditahan pengirimannya. Pada saat pemilihan itulah terdapat permasalahan *Knapsack* yang membutuhkan solusi pemilihan yang optimal. *Knapsack* adalah tempat yang digunakan sebagai sarana menyimpan atau memuat suatu objek. Tujuan dari penelitian ini adalah mengetahui bagaimana cara mendapatkan hasil solusi yang optimal dalam menyelesaikan permasalahan *Integer Knapsack* pada transportasi angkut barang dengan menggunakan Algoritma *Dynamic Programming* dan Algoritma *Greedy* di PT Pos Indonesia Semarang. Serta mengetahui hasil komparasi Algoritma *Greedy* dengan Algoritma *Dynamic Programming* pada permasalahan *Integer* *Knapsack* pada pemilihan angkut barang di PT Pos Indonesia Semarang dengan menerapkan pada aplikasi *mobile*. Hasil pada penelitian ini terlihat dari hasil perhitungan yang diperoleh Algoritma *Dynamic Programming*, yaitu total berat pengiriman sebesar 5022 kg dalam 7 hari. Sedangkan hasil perhitungan yang diperoleh Algoritma *Greedy*, yaitu total berat pengiriman sebesar 4496 kg dalam 7 hari. Dapat disimpulkan bahwa hasil perhitungan yang diperoleh Algoritma *Dynamic Programming* dalam 7 hari memiliki total berat pengiriman 526 kg lebih besar jika dibandingkan dengan Algoritma *Greedy*.

**Keywords**: Permasalahan *Integer Knapsack*, Algoritma *Greedy*, Algoritma *Dynamic Programming*, Transportasi Angkut Barang.

1. **PENDAHULUAN**

Seiring dengan perkembangan teknologi, penyelesaian suatu permasalahan yang semula dilakukan secara manual, sekarang sudah bisa dilakukan secara sistematis melalui aplikasi. Proses penyelesaian permasalahan dapat dilakukan oleh Algoritma pada sebuah aplikasi. Algoritma adalah metode efektif dari perintah–perintah yang telah didefinisikan dengan baik untuk menghitung sebuah fungsi. Dimulai dari sebuah kondisi awal dan input awal, intruksi–intruksi tersebut menjelaskan sebuah komputasi yang bila dieksekusi atau diproses melalui sejumlah urutan kondisi terbatas dapat terdefinisi dengan baik dan menghasilkan keluaran (output)[1]. Pada saat ini pengiriman barang menjadi hal yang tidak asing lagi karena para pelaku bisnis sekarang ini banyak bertransaksi di internet. Orang-orang akan semakin mudah untuk berbelanja walaupun penjualan dan pembelian tidak saling bertemu langsung. Itulah sebabnya jasa pengiriman barang semakin dibutuhkan. Peluang bisnis dan prospek jasa pengiriman barang masih sangat bagus dan terus berkembang[2]. PT. Pos Indonesia merupakan salah satu perusahaan yang bergerak dalam bidang pengiriman barang maupun surat di Indonesia[3]. Pada pengiriman barang, sering kita jumpai pemilihan barang yang akan dimasukan ke transportasi terlebih dahulu dan yang ditahan pengirimannya. Pada saat pemilihan itulah terdapat permasalahan *Knapsack* yang membutuhkan solusi pemilihan yang optimal. Dari beberapa penelitian sebelumnya, permasalahan knapsack dapat diselesaikan menggunakan Algoritma *Greedy* Maupun Algoritma *Dynamic programming*[4,6,15].

Knapsack problem merupakan permasalahan yang sering kita temukan dalam kehidupan sehari-hari[5]. *Knapsack* adalah tempat yang digunakan sebagai sarana menyimpan atau memuat suatu objek. Tempat tersebut hanya dapat menyimpan objek dengan ketentuan total ukuran yang yang terbatas dari kapasitas *Knapsack* tersebut. Setiap objek tidak harus dimasukan secara keseluruhan namun bisa sebagian objek saja. Banyak tahap-tahap yang diperlukan untuk mendapatkan penyelesaian masalah tersebut[6]. Permasalahan knapsack merupakan permasalahan program linear yang hanya memiliki satu kendala[7]. *Knapsack* adalah permasalahan mengenai optimalisasi kombinatorial di mana kita harus mencari solusi terbaik dari banyak kemungkinan yang dihasilkan[8]. Solusi yang dihasilkan yaitu Setiap objek memiliki berat tertentu dan memperoleh keuntungan tertentu. *Knapsack* dapat diisi dengan berat yang maksimal[9]. Tujuan Knapsack problem untuk mendapatkan keuntungan yang maksimum dari pemilihan barang tanpa melebihi kapasitas daya tampung media penyimpanan tersebut[10].

Dalam menyelesaikan permasalahan *Integer Knapsack* pada Algoritma *Greedy*, n masukan data dilakukan secara bertahap. Pertama dilakukan pemilihan solusi yang mungkin kemudian dari himpunan solusi yang mungkin tersebut akan diperoleh solusi optimal[11]. Pada setiap langkah dalam Algoritma Greedy, diambil keputusan yang paling optimal untuk langkah tersebut tanpa memperhatikan konsekuensi pada langkah selanjutnya[12]. Algoritma *Greedy* memecahkan masalah dengan membuat pilihan yang terbaik pada saat tertentu[13].

Pada Algoritma *Dynamic Programming* penyelesaian permasalahan *Integer Knapsack* dirancang secara matematis untuk memperbaiki tingkat efisiensi dengan menguraikannya menjadi bagian-bagian masalah yang lebih kecil, sehingga lebih sederhana dalam perhitungan. Untuk menyelesaikan usulan pemrograman matematis Model pendekatan *Dynamic Programming* dapat digunakan *Dynamic Programming* adalah pendekatan yang kuat melalui dimana solusi optimal global dapat diperoleh dalam kasus ruang solusi diskrit [14]. Ada beberpa variabel yang diperhatikan salah satunya yaitu jenis barang yang diangkut ke dalam alat pengangkut atau media transportasi[15].

Berdasarkan uraian di atas, tujuan penelitian ini adalah komparasi Algoritma *Dynamic Programming* dan Algoritma *Greedy* pada permasalahan *Integer Knapsack* di Transportasi angkut barang sehingga didapatkan solusi pengangkutan barang yang paling optimal.

1. **METODE**

Metode yang diterapkan pada penelitian ini adalah algoritma dynamic programming dan algoritma greedy. Berikut adalah penjelasan masing-masing algoritma:

* 1. **Algoritma *Dynamic Programming***

*Dynamic Programming* merupakan metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (step) atau tahapan (stage) sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan[16].

Langkah pertama pada tahap Algoritma Dynamic Programming menentukan struktur dari masalah. Dari masalah diketahui n barang. Kapasitas maksimal transportasi sebesar 750 kg dinotasikan sebagai M. Wi dinotasikan sebagai berat barang i, dan Pi dinotasikan sebagai ongkos kirim barang i. Dirumuskan sebagai M∈n,Wi ≤ M , i ∈ n, dengan batasan . Total keuntungan optimal dipengaruhi oleh ongkoskirim dinotasikan sebagai . Untuk memperoleh total keuntungan optimal dilakukan perhitungan dengan prosedur rekursif maju. Langkah selanjutnya adalah menentukan persamaan rekursif untuk memproleh total keuntungan optimal. Dari persamaan Z dapat ditulis suatu persamaan rekursif untuk mencari semua kemungkinan pengangkutan jenis barang ke-i sehingga diperoleh solusi pengangkutan optimal. Nilai dari solusi pengangkutan optimal dimasukan ke dalam z(i, j), sehingga dapat ditulis persamaan rekursifnya fk(y) = max { fk-1(y),pk + fk-1 (y – wk) } , k = 1,2,.. ,n.

pseudecode Algoritma Dynamic Programming ditunjukan pada Gambar 1.

|  |
| --- |
| Input: ongkoskirim (v[]) Berat barang (w[]) Jumlah item barang (n) Kapasitas Knapsack (W)for j from 0 to W do m[0, j] := 0end for for i from 1 to n do for j from 0 to W do if w[i] <= j then m[i, j] := max(m[i-1, j], m[i-1, j w[i]] + v[i]) else m[i, j] := m[i-1, j] end if end forend for |

**Gambar 1.** Pseudecode Algoritma Dynamic Programming

Kemudian menghitung nilai dari solusi optimal, prosedur perhitungan yang digunakan adalah prosedur rekursif maju. Perhitungan dimulai dari mencari nilai profit untuk barang pertama hingga barang terakhir sejumlah keseluruhan barang. Hasil perhitungan data menggunakan Algoritma Dynamic Programming disajikan pada Tabel 1 seperti berikut.

 **Tabel 1.** Hasil perhitungan menggunakan Algoritma Dynamic Programming.

|  |  |  |  |
| --- | --- | --- | --- |
| Tanggal  | Total Berat | Total Ongkos kirim | Jumlah data |
| 1 juli 2017 | 772 kg |  Rp. 25.495.500 | 31 data |
| 2 juli 2017 | 581 kg |  Rp. 6.857.000 | 10 data |
| 3 juli 2017 | 750 kg |  Rp. 35.730.500 | 35 data |
| 4 juli 2017 | 750 kg |  Rp. 40.233.000 | 39 data |
| 5 juli 2017 | 744 kg |  Rp. 31.698.000 | 34 data |
| 6 juli 2017 | 725 kg |  Rp. 37.829.000 | 32 data |
| 7 juli 2017 | 750 kg |  Rp. 32.292.500 | 25 data |
| 8 juli 2017 | 719 kg |  Rp. 11.457.500 | 15 data |
| 9 juli 2017 | 723 kg | Rp. 7.297.000 | 2 data |
| 10 juli 2017 | 516 kg | Rp. 4.881.000 | 2 data |
| 11 juli 2017 | 278 kg | Rp. 2.000.000 | 1 data |

Berdasarkan hasil perhitungan pada Tabel 1 diperoleh bahwa pada Algoritma *Dynamic Programming* sisa pengiriman membutuhkan 4 kali angkut, yaitu tanggal 8,9,10 dan 11 juli 2017.

* 1. **Algoritma *Greedy***

Langkah pertama pada tahap Algoritma *Greedy* adalah mengurutkan data. *Greedy* yang digunakan pada penelitian ini adalah *Greedy by density* data diurutkan berdasarkan *density* (ongkos kirim dibagi berat ) terbesar, kemudian diambil satu-persatu objek yang dapat ditampung oleh tempat penyimpanan hingga tempat tempat penyimpanan penuh atau sudah tidak ada lagi objek yang dimasukan ke dalam tempat penyimpanan. Hasil perhitungan data menggunakan Algoritma *Greedy* disajikan pada Tabel 2 seperti berikut.

**Tabel 2.** Hasil perhitungan menggunakan Algoritma *Greedy*

|  |  |  |  |
| --- | --- | --- | --- |
| Tanggal  | Total Berat | Total Ongkos kirim | Jumlah data |
| 1 juli 2017 | 722 kg |  Rp. 25.495.500 | 31 data |
| 2 juli 2017 | 581 kg |  Rp. 6.857.000 | 10 data |
| 3 juli 2017 | 560 kg |  Rp. 30.270.500 | 37 data |
| 4 juli 2017 | 659 kg |  Rp. 34.609.000 | 34 data |
| 5 juli 2017 | 706 kg |  Rp. 32.937.000 | 32 data |
| 6 juli 2017 | 570 kg |  Rp. 30.562.000 | 28 data |
| 7 juli 2017 | 698 kg |  Rp. 36.651.500 | 30 data |
| 8 juli 2017 | 672 kg |  Rp. 14.456.000 | 5 data |
| 9 juli 2017 | 724 kg |  Rp. 8.781.000 | 10 data |
| 10 juli 2017 | 706 kg |  Rp. 8.526.000 | 4 data |
| 11 juli 2017 | 660 kg |  Rp. 6.625.000 | 5 data |

Berdasarkan hasil perhitungan pada Tabel 2 diperoleh bahwa pada Algoritma *Greedy* sisa pengiriman membutuhkan 4 kali angkut, yaitu tanggal 8,9,10 dan 11 juli 2017 .

Algoritma *Greedy* adalah Algoritma untuk menyelesaikan permasalahan secara bertahap [6].

Tahap penyelesaiannya adalah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan.

2 .Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

pseudecode Algoritma Greedy ditunjukan pada Gambar 2.

|  |
| --- |
| Input:p(1:n), w(1:n), x(1:n), m, isi,nx(1:n) = 0isi = mfor i = 1 to n do if w(i) > isi then  Exit endif x(i) = 1 isi = isi – w(i)repeat if i ≤ n then x(i) = isi/w(i)  endif |

**Gambar 2.** Pseudecode Algoritma Greedy

* 1. **Perancangan Proses**

Langkah-langkah pembuatan sistem:

1. Tahap pengumpulan data

Tahap pengumpulan data adalah tahap yang dilakukan untuk mendapatkan data-data yang berkaitan dan dibutuhkan dalam penelitian[17].

1. Studi Literatur

Tahapan ini merupakan tahapan melalui literatur yang relevan dengan penelitian. Tahap ini dilakukan untuk mendapatkan informasi yang relevan terkait dengan penelitian [18]. Pada tahap ini juga dilakukan pengumpulan bahan, informasi, keterangan, dan teori dalam buku serta konsultasi dengan para ahli atau narasumber serta rujukan dari artikel, jurnal, dan karya ilmiah lainnya yang relevan dan berhubungan dengan objek penelitian, serta metode yang digunakan dalam penelitian[19].

1. Studi Lapangan

Tahap ini adalah pencarian data di lapangan. tahap ini dilakukan untuk memahami, mempelajari, dan mendapatkan informasi. Metode observasi berguna bagi peneliti untuk mengumpulkan data dalam berbagai cara[20].

1. Implementasi

Tahap ini adalah proses membangun suatu sistem berdasarkan hasil analisis dan perancangan sistem yang telah dilakukan sebelumnya. Begitu sistem dibangun, data yang didapat pada pengumpulan data diolah pada sistem yang telah dibangun[21].

1. Pengujian

Tahap ini merupakan proses pengujian yang dilakukan dengan menggunakan data dan hasil implementasi.

* 1. **Perancangan Sistem**

Perancangan sistem ini menggunakan model waterfall. Waterfall adalah sebuah pendekatan berdasarkan asumsi bahwa keputusan besar harus dibuat sebelum pengkodean dimulai [22]. Waterfall adalah metodologi pengembangan perangkat lunak yang mengusulkan sebuah pendekatan perangkat lunak yang sistematis dan berurutan yang dimulai pada tingkat kemajuan sistem seluruh analisis, desain, kode, pengujian dan pemeliharaan [23].Model ini diberi nama "Waterfall" karena representasi diagramnya terlihat seperti air terjun, Ini juga dikenal sebagai classical lifecycle model[24]. Model ini yang sering digunakan oleh penganalisis sistem pada umumnya [25]. Ada empat tahap dalam waterfall yaitu analisis kebutuhan, desain, implementasi dan pengujian [26].

1) Tahap analisis kebutuhan yaitu mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan dan garis besar sistem yang akan dibuat [27]. tahap ini bermaksud untuk mengidentifikasi dan mengevaluasi masalah dan kebutuhan yang dibutuhkan, tahap ini meliputi analisis kebutuhan perangkat keras, analisis persyaratan perangkat lunak, analisis kebutuhan pengguna dan persyaratan proses analisis [28].

2) Tahap desain yaitu melakukan desain aplikasi meliputi desain antarmuka, dan desain struktur database [29]. Tahap Desain adalah proses penerjemahan sistem sesuai algoritma yang digunakan [30].

3) Tahap implementasi yaitu merancang perangkat lunak yang kemudian direalisasikan sebagai serangkaian program atau unit program [31].

4) Tahap pengujian yaitu menguji apakah sistem sudah siap dan layak untuk dipakai. Tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program [32]. Untuk mengecek keberhasilan dari system, dilakukan tahap pengujian. Tahap ini adalah dengan melakukan uji mengenai kesalahan-kesalahan yang ada dalam pembuatan program[33]. Verifikasi adalah proses evaluasi perangkat lunak untuk menentukan apakah suatu produk dari fase pengembangan tertentu memenuhi persyaratan yang diberlakukan pada awal tahap tersebut[34].

1. **HASIL DAN PEMBAHASAN**
2. **Pengumpulan Data**

Pada penelitian ini, data yang digunakan adalah data catatan pengiriman barang di PT Pos Indonesia di Semarang. Data diperoleh dari PT Pos Indonesia di Semarang sebanyak 226 data dalam seminggu di bulan juli. Berikut disajikan tabel mengenai data catatan pengiriman barang tanggal 1 – 7 Juli 2017.

**Tabel 1.** Data catatan pengiriman barang tanggal 1 – 7 Juli 2017

|  |  |  |  |
| --- | --- | --- | --- |
| Tanggal  | Total Berat | Total Ongkos kirim | Jumlah data |
| 1 juli 2017 | 802 kg |  Rp. 26.581.500 | 32 data |
| 2 juli 2017 | 501 kg |  Rp. 5.771.000 | 9 data |
| 3 juli 2017 | 1057 kg |  Rp. 42.397.500 | 43 data |
| 4 juli 2017 | 739 kg |  Rp. 39.318.000 | 36 data |
| 5 juli 2017 | 1764 kg |  Rp. 46.103.000 | 38 data |
| 6 juli 2017 | 1329 kg |  Rp. 44.748.500 | 33 data |
| 7 juli 2017 | 616 kg |  Rp. 30.851.500 | 35 data |

Dalam permasalahan *Knapsack* ditemukan masalah dalam tempat penyimpanan. tempat tersebut hanya dapat menyimpan objek dengan ketentuan total ukuran yang yang terbatas dari kapasitas *Knapsack* tersebut. Keadaan tersebut dapat digambarkan sebagai keadaan kelebihan muatan. Pada penelitian ini data catatan pengiriman barang di PT Pos Indonesia Semarang terdapat permasalahan *Knapsack*. Karena jika diketahui Kapasitas maksimal transportasi sebesar 750 kg sedangkan total berat pengiriman pada tanggal 1,3,5, dan 6 Juli 2017 melebihi kapaitas maksimal transportasi. Sehingga pada data tersebut terdapat permasalahan *Knapsack.* selama terdapat keadaan kelebihan muatan dapat di cari solusi *optimal* menggunakan Algoritma *Greedy* maupun Algoritma *Dynamic Programming.*

1. **Perhitungan Data**

Pada penerapan Algoritma *Greedy*, dengan menggunakan data tanggal 1 juli 2017 sampai dengan 7 juli 2017. Diperoleh total berat sisa pengiriman sebesar 2762 kg. Sedangkan pada penerapan Algoritma *Dynamic Programming* diperoleh total berat sisa pengiriman sebesar 2236 kg. Perbandingan solusi pengambilan barang dari kedua Algoritma dapat dilihat pada Tabel 4 seperti berikut.

**Tabel 4.** Perbandingan solusi pengambilan barang dari kedua Algoritma.

|  |  |
| --- | --- |
| Tanggal | Total Hasil Perhitungan |
|  | Algoritma | Berat | Ongkos Kirim |
| 1 juli 2017 | *Greedy* | 722 kg | Rp. 25.495.500 |
| *Dynamic Programming* | 722 kg | Rp. 25.495.500 |
| 2 juli 2017 | *Greedy* | 581 kg | Rp. 6.857.000 |
| *Dynamic Programming* | 581 kg | Rp. 6.857.000 |
| 3 juli 2017 | *Greedy* | 560 kg | Rp. 30.270.500 |
| *Dynamic Programming* | 750 kg | Rp. 35.730.500 |
| 4 juli 2017 | *Greedy* | 659 kg | Rp. 34.609.000 |
| *Dynamic Programming* | 750 kg | Rp. 40.233.000 |
| 5 juli 2017 | *Greedy* | 706 kg | Rp. 32.937.000 |
| *Dynamic Programming* | 744 kg | Rp. 31.698.000 |
| 6 juli 2017 | *Greedy* | 570 kg | Rp. 30.562.000 |
| *Dynamic Programming* | 725 kg | Rp. 37.829.000 |
| 7 juli 2017 | *Greedy* | 698 kg | Rp. 36.651.500 |
| *Dynamic Programming* | 750 kg | Rp. 32.292.500 |
| 8 juli 2017 | *Greedy* | 672 kg | Rp. 14.456.000 |
| *Dynamic Programming* | 719 kg | Rp. 11.457.500 |
| 9 juli 2017 | *Greedy* | 724 kg | Rp. 8.781.000 |
| *Dynamic Programming* | 723 kg | Rp. 7.297.000 |
| 10 juli 2017 | *Greedy* | 706 kg | Rp. 8.526.000 |
| *Dynamic Programming* | 516 kg | Rp. 4.881.000 |
| 11 juli 2017 | *Greedy* | 660 kg | Rp. 6.625.000 |
| *Dynamic Programming* | 278 kg | Rp. 2.000.000 |

Berdasarkan hasil perhitungan dari kedua Algoritma tersebut, dapat diketahui bahwa solusi pengambilan barang menggunakan Algoritma *Dynamic Programming* lebih baik jika dibandingkan dengan Algoritma *Greedy* maupun memilih secara acak tanpa Algoritma*.* Hal ini terlihat dari hasil perhitungan yang diperoleh Algoritma *Dynamic Programming*, yaitu total berat pengiriman sebesar 5022 kg dalam 7 hari. Sedangkan hasil perhitungan yang diperoleh Algoritma *Greedy*, yaitu total berat pengiriman sebesar 4496 kg dalam 7 hari.

1. **KESIMPULAN**

Dari hasil penelitian terkait hasil komparasi Algoritma *Greedy* dan algorima *Dynamic Programming* pada Permasalahan *Integer Knapsack* di Transportasi Angkut Barang dengan menggnakan data catatan pengiriman barang di PT Pos Indonesia di Semarang, dapat ditarik kesimpulan bahwa solusi pengambilan barang menggunakan Algoritma *Dynamic Programming* lebih baik jika dibandingkan dengan Algoritma *Greedy* maupun memilih secara acak tanpa Algoritma*.* Hal ini terlihat dari hasil perhitungan yang diperoleh Algoritma *Dynamic Programming*, yaitu total berat pengiriman sebesar 5022 kg dalam 7 hari. Sedangkan hasil perhitungan yang diperoleh Algoritma *Greedy*, yaitu total berat pengiriman sebesar 4496 kg dalam 7 hari. solusi pengambilan barang menggunakan Algoritma *Dynamic Programming* dalam 7 hari memiliki total berat pengiriman 526 kg lebih besar jika dibandingkan dengan Algoritma *Greedy*.

1. **DAFTAR PUSTAKA**

[1] Alamsyah & Putri, T, I. 2014. Penerapan Algoritma Greedy Pada Mesin Penjual

 Otomatis (*Vending Machine*). *Scientific Journal of Informatics,* 1(2): 201-209.

[2] Trianto, E & Revina, W. 2016. Perancangan Sistem Iinformasi Pencatatan Pengiriman Barang di PT. TIKI Jalur Nugraha Ekakurir Cabang Bandung. (online), (http://ejournal.lpkia.ac.id, diakses 4 Maret 2017).

[3] Fitriana, E.N., & Sugiharti, E. 2015. Implementasi Algoritma Genetika Dengan Teknik Kendali Logika Fuzzy Untuk Mengatasi Travelling Salesman Problem Menggunakan Matlab. *Unnes Journal of Mathematics,* 4(2): 114.

[4] Husni, T. S. & Arief, A. 2008. Implementasi 0-1 KnapsackMenggunakan AlgoritmaDynamic ProgrammingPada Aplikasi Perhitungan Harga Satuan Produk Percetakan Berbasis Web (Studi Kasus: Cv Tunas Utama). *Jurnal InforSAINS* 2(3): 31-35.

[5] Supriana, I, Wayan. 2016. Optimalisasi Penyelesaian Knapsack Problem Dengan Algoritma Genetika. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*. 7(3):128.

[6] Paryati. 2009. Optimasi Strategi Algoritma Greedy Untuk Menyelesaikan Permasalahan Knapsack 0-1. Seminar Nasional Informatika 2009. Yogyakarta: UPN Veteran Yogyakarta.

[7] Isnaeni, S., Supriyono., & Arini, F. Y. 2014. Implementasi Algoritma Pemrograman Dinamik Untuk Penyelesaian Persoalan Knapsack Dalam Penentuan Keuntungan Optimal Penjualan Barang. *Unnes Journal of Mathematics,* 3(2): 98.

[8] Puspa, S. K., Mrunal, T. V. & Suhas. C. 2016. A Study of Performance Analysis on Knapsack Problem. *International Journal of Computer Applications,* 0975 – 8887:1

[9] Pajjan, S. P., Roogi, R., Badiger, V., & Amaragatti, S. 2014. A New Approach to Solve Knapsack Problem. *Oriental Journal Of Computer Science & Technology,* 7(2):219.

[10] Supriadi, D. 2016. Perbandingan Penyelesaian Knapsack Problem Secara Matematika, Kriteria Greedy Dan Algoritma Greedy. *Indonesian Journal on Computer and Information Technology*,1(2):91.

[11] V. Springer. 2005. Knapsack 0-1 Problem. Yogyakarta: Graha Ilmu.

[12] Faisal. 2014. Penerapan Metode Greedy Knapsack Dalam Menentukan Komposisi Buah Pada Masalah Keranjang. *Jurnal Teknologi Informasi*, 10(2): 33.

[13] Malik, A., Sharma, A., Saroha, V. 2013. Greedy Algorithm. *International Journal of Scientific and Research Publications,* 3(8):1.

[14] Tari, F.G. 2016. A Hybrid Dynamic Programming for Solving Fixed Cost Transportation with Discounted Mechanism. *Hindawi Publishing Corporation Journal of Optimization,* 2016(8518921):1.

[15] Pratiwi, A & Rochmad M. 2014. Implementasi Algoritma Branch And Bound Pada 0-1 Knapsack Problem Untuk Mengoptimalkan Muatan Barang. *Unnes Journal of Mathematics,* 3(2): 92.

[16] Surjawan, D, J & Susanto I. 2015. Aplikasi Optimalisasi Muat Barang Dengan Penerapan Algoritma Dynamic Programming Pada Persoalan Integer Knapsack. *Jurnal Teknik Informatika dan Sistem Informasi,* 1(2):151-162.

[17] Ashari A. I., Muslim, M.A., & Alamsyah. 2016. Comparison Performance of Genetic Algorithm and Ant Colony Optimization in Course Scheduling Optimizing. *Scientific Journal of Informatics*, 3(2): 150.

[18] Sugiharti, E., Firmansyah, S., & Devi, F.R. 2017. Redictive Evaluation Of Performance Of Computer Science Students Of Unnes Using Data Mining Based On Naïve Bayes Classifier (Nbc) Algorithm. *Journal of Theoretical and Applied Information Technology*, 95(4): 905.

[19] Amin, S., Muslim, M.A., & Alamsyah. 2012. Sistem Deteksi Dini Hama Wereng Batang Coklat Menggunakan Jaringan Syaraf Tiruan Backpropagation. *Unnes Journal of Mathematics*, 1(2): 119.

[20] Alamsyah & Arus, A.A. 2014. Analisis Sistem Pendaftaran pada Web Forum Ilmiah Matematika Unnes 2014. *Scientific Journal of Informatics*, 1(1): 110.

[21] Sugiharti, E., & Muslim, M.A. 2016. On-Line Clustering Of Lecturers Performance Of Computer Science Department Of Semarang State University Using K-Means Algorithm. *Journal of Theoretical and Applied Information Technology*, 83 (1): 66.

[22] Patel, U.A., & Jain N.K. 2013. New Idea In Waterfall Model For Real Time Software Development. *International Journal of Engineering Research & Technology (IJERT)*, 2(4): 115.

[23] Rukmana, S.H., & Muslim, M. A. 2016. Sistem Pendukung Keputusan Tender Proyek Menggunakan Metode Benefit Cost Ratio*. Jurnal Sains & Teknologi*, 5(2): 817-822.

[24] Saxena, A., & Upadhyay, P. 2016. Waterfall vs. Prototype: Comparative Study of SDLC. *Imperial Journal of Interdisciplinary Research (IJIR)*, 2(6):1012-1013.

[25] Roviaji, R., & Muslim M.A. 2017. Pembuatan Sistem Informasi Gardu Induk PT. Pln (Persero) App Semarang Se-Kota. Prosiding Seminar Ilmu Komputer dan Teknologi Informasi. Samarinda: Universitas Mulawarman.

[26] Pressman, R.S. 2001. Software Engineering. Online. Tersedia di http://www.resource.mitfiles.com/ [diakses 18-2-2017].

[27] Nugroho, Z.A., & Arifudin, R. Sistem Informasi Tracer Study Alumni Universitas Negeri Semarang Dengan Aplikasi Digital Maps. *Scientific Journal of Informatics*, 1(2): 154.

[28] Rukhansah, N., Muslim, M. A., & Arifudin, R. 2016. Peramalan Harga Emas Menggunakan Fuzzy Time Series Markov Chain Model. *KOMPUTAKI*, 1(1): 68.

[29] Putra, A.T. 2014. Pengembangan E-Lecture menggunakan Web Service Sikadu untuk Mendukung Perkuliahan di Universitas Negeri Semarang. *Scientific Journal of Informatics*, 1(2): 170.

[30] Muslim, M.A., Kurniawati I., & Sugiharti E. 2015. Expert System Diagnosis Chronic Kidney Disease Based on Mamdani Fuzzy Inference System. *Journal of Theoretical and Applied Information Technology*, 78(1): 71.

[31] Purwinarko, A. 2014. Model Expertise Management System di Universitas Negeri Semarang. *Scientific Journal of Informatics*, 1(2): 178.

[32] Mustaqbal, M.S., Firdaus, F.S., Rahmadi, H. 2015. Pengujian Aplikasi Menggunakan Black Box Testing Boundary Value Analysis. *JITTER*, 1(3): 35.

[33] Sugiharti E & Triliani, S. E. 2014. Perancangan Aplikasi Surat Masuk dan Keluar pada PT. Angkasa Pura 1 Semarang. *Scientific Journal of Informatics*, 1(1): 41.

[34] Hardyanto, W., Purwinarko, A., Sujito, F., Masturi., Alighiri., D. 2016. Applying an MVC Framework For The System Development Life Cycle With Waterfall Model Extended*. Journal of Physics: Conference Series*, 824 (1): 3.