Algoritma Local Search pada Bin Packing

Hubert, Dennis  
*Sekolah Tinggi Teknik Elektro*  
*Institut Teknologi Bandung*Bandung, Indonesia  
13222018@std.stei.itb.ac.id

*Abstract*—This electronic document is a “live” template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. *\*CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract*. (*Abstract*)

Keywords—component, formatting, style, styling, insert (key words)

# **Introduction**

This template, modified in MS Word 2007 and saved as a “Word 97-2003 Document” for the PC, provides authors with most of the formatting specifications needed for preparing electronic versions of their papers. All standard paper components have been specified for three reasons: (1) ease of use when formatting individual papers, (2) automatic compliance to electronic requirements that facilitate the concurrent or later production of electronic products, and (3) conformity of style throughout a conference proceedings. Margins, column widths, line spacing, and type styles are built-in; examples of the type styles are provided throughout this document and are identified in italic type, within parentheses, following the example. Some components, such as multi-leveled equations, graphics, and tables are not prescribed, although the various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

# **Studi Literatur**

## **Bin Packing Problem**

[1]Bin Packing Problem adalah masalah optimisasi di mana sejumlah item dengan ukuran tertentu harus dimasukkan ke dalam sejumlah wadah dengan kapasitas terbatas, dengan tujuan meminimalkan jumlah wadah yang digunakan. Masalah ini bersifat NP-hard dan banyak diterapkan dalam bidang logistik, manufaktur, serta ilmu komputer.

|  |
| --- |
| Schematic of the bin-packing problem for a sample case. | Download  Scientific Diagram |
| Gambar 2.1 Ilustrasi dari One Dimentional Bin Packing Problem[3] |

Dalam **masalah *one dimentional*** bin packing **klasik**, diberikan sebuah urutan item , di mana setiap item memiliki ukuran . Tujuannya adalah untuk **mengemas item-item tersebut ke dalam jumlah wadah (**bin**) berkapasitas satu seminimal mungkin**,  
yaitu membagi himpunan item tersebut ke dalam sejumlah minimum subset  
sedemikian sehingga:

Algoritma Worst case didapatkan pada saat:

RA​=inf{ r ≥1 ∣ RA​(L) ≤ r untuk semua L}

Dengan RA :

RA​(L) = A(L)/OPT(L)​

Keterangan :

RA : Rasio jumlah bin algoritma ke hasil optimal;

r : Batas atas algoritma

L : daftar weight

[4] Prosedur peningkatan solusi dalam *bin packing problem* umumnya terdiri dari tiga tahap utama:

1. **Konstruksi solusi awal (Initial solution construction)**  
   Algoritma Best Fit Decreasing (BFD) digunakan untuk membangun solusi awal yang layak — yaitu pembagian item ke dalam bin sedemikian rupa sehingga kapasitas tiap bin tidak terlampaui. Salah satu Konstruksi lainnya adalah Worst-Fit, yaitu membangun solusi dengan memasukkan masing masing weight satu weight ke dalam masing-masing bin sehingga weight tersebar merata dan jumlah ruang kosong paling tinggi dan membuat worst case scenario.
2. **Perbaikan solusi (Solution improvement)**  
   Setiap bin dalam solusi saat ini dihapus satu per satu, dan item-item di dalamnya dicoba untuk didistribusikan kembali ke bin lain sesuai dengan aturan tertentu (misalnya *best-fit*, *first-fit*, atau *worst-fit*).
   * Jika hasil redistribusi menghasilkan solusi baru yang feasible, maka jumlah bin berkurang satu, dan proses diulang.
   * Jika tidak feasible (ada bin yang kelebihan kapasitas), maka algoritma local search diterapkan untuk mencoba mengurangi tingkat ketidaklayakan tersebut (*reduce infeasibility*).
3. **Local search**  
   Pada tahap ini, pasangan bin dalam solusi sementara yang tidak layak diperiksa secara berurutan.  
   Item-item di antara kedua bin tersebut ditukar (redistributed) dengan harapan bahwa salah satu kombinasi pertukaran menghasilkan solusi yang layak kembali.  
   Bila solusi layak ditemukan, maka proses perbaikan solusi diulang dari tahap (b).

## **Local Search**

[2]Local search adalah metode optimisasi heuristik yang bekerja dengan memodifikasi solusi saat ini secara bertahap (melalui tetangga/neighbor) untuk menemukan solusi yang lebih baik, sampai tidak ada perbaikan lebih lanjut yang dapat dilakukan dalam lingkungan lokalnya. Pada paper ini dilakukan penelitian dengna 3 metode Local search yaitu Hill Climbing, Simmulated Annealing, dan Genetic Algorithm.

Pseudocode local search untuk Binary Packing :

|  |
| --- |
|  |
| Gambar 2.2 Local Search dari Binary Packing |

|  |
| --- |
| A diagram of a graph  AI-generated content may be incorrect. |
| Gambar 2.3 Visualisasi dari Local Search |

# **Metodologi**

## **Algoritma Inisiasi dan first fit**

**A.1 First Fit**

[5]Misalkan bin diberi indeks sebagai , dengan masing-masing awalnya terisi pada level nol.  
Angka-angka akan ditempatkan secara berurutan. Untuk menempatkan , carilah indeks terkecil sedemikian sehingga bin terisi hingga level , lalu tempatkan ke dalam . Setelah itu, bin kini terisi hingga level .

Keuntungan dari digunakannya Algoritma first fit adalah saat inisiasi terkadang dapat mendapatkan global optimal sehingga mempercepat proses local search. [7] Selain itu Worst case dari first fit algorithm adalah :

FF(L) ≤ c × OPT(L)

Berikut adalah flowchart dari first-fit algorithm :

|  |
| --- |
| A diagram of a process flow  AI-generated content may be incorrect. |
| Gambar 3.1 First Fit Algorithm Flowchart |

**A.2 Worst Fit**

Sementara algoritma worst-fit membangun solusi dengan memasukkan masing masing weight satu weight ke dalam masing-masing bin sehingga weight tersebar merata dan jumlah ruang kosong paling tinggi dan selalu membuat worst case scenario. Gambar dibawah adalah Flowchart dari worst fit algorithm.

|  |
| --- |
|  |
| Gambar 3.2 Worst Fit Algorithm Flowchart |

Pada Analisis akan digunakan Worst Fit agar visualisasi proses dari local search lebih mudah untuk diamati. Selain itu Perbandingan Performa akhir akan dibandingkan di bagian analisis.

Pada pengolahan digunakan tipe data pada bin sebagai list of dictionary. Masing-masing dictionary memiliki keys dan Value sebagai berikut:

['kontainer':(string)],

['remaining':(int)]

['total':(int)]

['barang':[['id':(string)],['ukuran':(int)]]]

## **Perhitungan Heuristic**

[6] Heuristik adalah aturan praktis atau educated guess yang mengurangi usaha pencarian dengan mengarahkan proses pencarian ke solusi yang lebih menjanjikan. Pada program karena objective dari bin packing adalah menggunakan bin sesedikit mungkin, heuristic merupakan fungsi minimalisasi. Heuristic dihitung dengan menjumlahkan space kosong dari masing-masing bin, jika ada bin yang space berlebih, diberi penalty sejumlah a. Pada percobaan yang dialkukan a ditune sebesar 10. Angka diambil tidak terlalu besar agar memungkinkan overfill dipilih dalam mencari hasil optimum.

Pada awalnya, perhitungan local search diberi constrain agar selalu mematuhi constraint besar container, namun seiring berjalannya waktu hasil banyak yang terjebak di local optimum. Sehingga diberikan penalti untuk overfitting pada Heuristic. Sehingga Heuristic didefinisikan sebagai Persamaan dibawah dengan c adalah space container :

## **Hill climbing algorithms**

[6] Hill climbing algorithm pada dasarnya adalah sebuah loop yang terus bergerak ke arah peningkatan atau penurunan nilai (sesuai fungsi objektif)—dengan kata lain, *hill climbing*. Algoritma ini berhenti ketika mencapai sebuah “puncak” atau local optimum dimana tidak ada *neighbor* yang memiliki nilai lebih tinggi. Algoritma ini tidak memelihara *search tree*, sehingga struktur data untuk *node* saat ini hanya perlu menyimpan *state* dan nilai dari fungsi objektif. Untuk mempermudah penggunaan istilah, karena pada bin packing heuristic didefinisikan sebagai minimum maka kata ascent di bawah dapat berarti juga optimalisasi dan Climbing = Descending, Naik = Turun (tergantung konteks).

Pada percobaan di paper ini dilakukan 4 jenis hill climbing, yaitu steepest ascent hill climbing (SAHC), sideway move hill climbing (SMHC), random restart hill climbing (RRHC), dan stochastic hill climbing (SHC).

Tipe data input : [['kontainer':(string)], ['remaining':(int)], ['total':(int)], ['barang':[['id':(string)],['ukuran':(int)]]]

Tipe data output :[['kontainer':(string)],['remaining':(int)], ['total':(int)], ['barang':[['id':(string)],['ukuran':(int)]]]

|  |
| --- |
|  |
| Gambar 3.3.1 Ilustrasi algoritma Hill Climbing. |

**C1. Steepest ascent hill climbing (SAHC)**

[6]Pada Steepest Ascent hill climbing, pergeseran current dilakukan saat didapatkan hasil yang lebih baik. Hill climbing kadang disebut *greedy local search* karena algoritma ini langsung memilih *neighbor state* yang lebih baik tanpa memikirkan langkah berikutnya. Steepest Ascent Hill climbing biasanya membuat optimum solution yang cepat menuju solusi karena umumnya cukup mudah untuk memperbaiki keadaan yang buruk.

Namun sayangnya, SAHC sering terjebak karena beberapa alasan berikut:

* **Local maximum (maksimum lokal):**  
  Sebuah *local maximum* adalah puncak yang lebih tinggi dari semua keadaan tetangganya, tetapi lebih rendah dari *global maximum*. Algoritma *SAHC* yang mencapai daerah sekitar *local maximum* akan terdorong naik ke puncak tersebut, namun kemudian terjebak karena tidak ada lagi arah untuk nai
* **Ridge:**

*Ridge* menghasilkan serangkaian *local maximum* yang sangat sulit dilalui oleh greedy algorithm.

* **Plateau:**  
  *Plateau* adalah area datar pada lanskap ruang keadaan (*state-space landscape*).  
  Area ini bisa berupa *flat local maximum* — yaitu puncak datar tanpa jalan naik sama sekali — atau *shoulder*, yaitu daerah yang masih memungkinkan untuk naik lebih lanjut (Gambar 3.3.1).  
  Pencarian *SAHC* bisa tersesat di *plateau* ini.

|  |
| --- |
|  |
| Gambar 3.3.2 Pseudocode dari SAHC, arah fitness function dibalik pada Bin Packing[8] |

**C2. Sideway move hill climbing (SMHC)**

Karena untuk menyelesaikan masalah Plateau ini, SAHC hanya dapat menyelesaikan 86% problem SAHC[6]. Salah Satu solusi dari problem ini adalah memperbolehkan sideways move. Akibatnya dilakukan sideway move agar state dapat bergerak dari plateau ke global maximum. Perbedaan pseudocode sari SMHC dan SAHC adalah jika fitness\_baru >= fitness\_lama, dapat terjadi pertukaran state. Pertukaran state kemudian diconstraint dengan Max\_no\_improve.

**C3. Stochastic hill climbing (SHC)**

Stochastic hill climbing memilih secara acak langkah-langkah yang menuju ke arah naik, namun akan selalu dipilih arah terbaik. Probabilitas pemilihannya dapat bervariasi tergantung pada tingkat *steepness* dari langkah tersebut. Metode ini biasanya berkonvergensi lebih lambat dibandingkan dengan steepest ascent hill climbing, tetapi pada beberapa bentuk *state landscape*, metode ini dapat menemukan solusi yang lebih baik.

|  |
| --- |
| i = initial solution  j = 0  While (j < max\_iter)  Generates neighbours(i);  s\_rand = choose randomly(s);  if (fitness(s\_rand) > fitness(i) ) then  replace s with i;  end if  j++ |
| Gambar 3.3.2 Pseudocode dari SHC, arah fitness function dibalik pada Bin Packing. |

**C4. Random restart hill climbing (RRHC)**

Algoritma ini melakukan serangkaian pencarian *hill climbing* yang dimulai dari *initial state* yang dihasilkan secara acak, hingga akhirnya ditemukan *goal state*. Algoritma ini dapat dikatakan *trivially complete* dengan probabilitas mendekati 1, karena pada akhirnya algoritma akan menghasilkan keadaan tujuan sebagai *initial state* setelah cukup banyak percobaan acak dilakukan.

|  |
| --- |
| i = initial solution  j = 0  while (j < max\_restart) do  While (f(s) < f(i)) do  Generates neighbours(i);  if fitness(s) > fitness(i) then  replace s with i;  end if  random\_restart(bins)  j++ |
| Gambar 3.3.3 Pseudocode dari RRHC, arah fitness function dibalik pada Bin Packing. |

## **Simmulated Annealing**

Sebuah algoritma hill climbing yang tidak pernah melakukan langkah “menurun” menuju keadaan dengan nilai lebih rendah (atau biaya lebih tinggi) dijamin akan tidak lengkap, karena dapat terjebak pada maksimum lokal.

Sebaliknya, random walk murni — yaitu bergerak menuju keadaan penerus yang dipilih secara acak uniform dari seluruh himpunan penerus — bersifat lengkap, tetapi sangat tidak efisien.

Oleh karena itu, masuk akal untuk mencoba menggabungkan hill climbing dengan random walk agar diperoleh algoritma yang memiliki efisiensi sekaligus kelengkapan. Salah satu algoritma yang melakukan hal ini adalah simulated annealing. Algoritma dari simulated annealing ditampilkan di gambar 3.4.1 dibawah.

Dalam metalurgi, *annealing* adalah proses untuk mengeraskan atau memperkuat logam dan kaca dengan cara memanaskannya hingga suhu tinggi, kemudian mendinginkannya secara bertahap, sehingga material dapat mencapai keadaan kristal dengan energi rendah.

|  |
| --- |
|  |
| Gambar 3.4.1 Pseudocode dari Simulated Annealing[6][page 126] |

## **Genetic Algorithms**

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed

|  |
| --- |
|  |
| Gambar 3.5.1 Ilustrasi dari Genetic Algorithm pada [6][page 127] |

|  |
| --- |
|  |
| Gambar 3.5.2 Pseudocode dari Genetic Algorithm[6][page 129] |

using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

## **Perhitungan Global Minimum**

[9]Mencari global minimum pada Bin Packing Algorithm Secara merupakan NP-Hard Problem. Sebagai contoh pada Algoritma Integer Linear Programming didapatkan time complexity sebesar 2^O(1/e log(1/e)) Dengan e = /(1OPT (I)+1). Namun batas bawah dari Bin Packing dengan mudah didapatkan dengan :

Sehingga jika ada salah satu nilai heuristic pada percobaan mencapai batas bawah, sudah dipastikan hasil tersebut adalah global minimum berdasarkan *squeeze theorem*.

# Hasil dan Diskusi

Pada experiment digunakan 3 testcase, 3 weight yang didefinisikan sebagai list of integer berikut :

weights\_1 = [2, 5, 4, 7, 1, 3, 6, 8, 9, 2, 5, 3, 7, 4, 6, 1, 8, 9, 2, 5]

weights\_2 = [1, 2, 1, 3, 2, 4, 3, 1, 2, 3, 4, 2, 1, 3, 2, 4, 3, 1, 2, 3]

weights\_3 = [2, 5, 4, 7, 1, 3, 6, 8, 9, 2, 5, 3, 7, 4, 6, 1, 8, 9, 2, 5, 3, 7, 2, 4, 5]

Berdasarkan metodologi yang digunakan, lower bound ditentukan sebagai berikut :

Tabel 4.1 Lower bound dari masing-masing weights

|  |  |  |
| --- | --- | --- |
| weights\_1 | weights\_2 | weights\_3 |
| 3 | 3 | 2 |

Berikut adalah variable control yang digunakan :

* Testcase : weights\_3
* Iterasi-HC : 100
* Max\_sidestep : 10
* Max\_restart : 10
* Iterasi Simulated Annealing : 200
* Simulated Annealing Temperature : 100
* Penurunan Temperature : T \*= 0.95
* Populasi : 10
* Generasi : 10
* Fitness threshold : 10% (diambil 10% terbaik) agar populasi tidak punah
* Mutation rate : 10%

## Hill Climbing Algorithm

## A.1. Steepest ascent hill climbing

Tabel 4.2 plot dari SAHC

|  |  |  |  |
| --- | --- | --- | --- |
| TC | Plot | Final value | Bin Used |
| 1 |  | 13 | 11 |
| 2 |  | 3 | 5 |
| 3 |  | 12 | 13 |

## A.2. Sideways Move Hill Climbing

Tabel 4.3 plot dari SMHC

|  |  |  |  |
| --- | --- | --- | --- |
| TC | Plot | Final value | Bin Used |
| 1 |  | 13 | 11 |
| 2 |  | 3 | 5 |
| 3 |  | 12 | 13 |

## A.3. Stochastic Hill Climbing

Tabel 4.4 plot dari SHC

|  |  |  |  |
| --- | --- | --- | --- |
| TC | Plot | Final value | Bin Used |
| 1 |  | 23 | 12 |
| 2 |  | 33 | 8 |
| 3 |  | 32 | 15 |

## A.4. Random Restart Hill Climbing

Tabel 4.5 plot dari SAHC

|  |  |  |  |
| --- | --- | --- | --- |
| TC | Plot | Best val | Bin Used |
| 1 |  | 3 | 10 |
| 2 |  | 3 | 5 |
| 3 |  | 2 | 12 |

Tabel 4.6 Hasil Global optimum pada TC\_3

|  |
| --- |
| Kontainer 1 (total: 10)    \* BRG009 (9)    \* BRG005 (1)  Kontainer 2 (total: 10)    \* BRG008 (8)    \* BRG001 (2)  Kontainer 3 (total: 10)    \* BRG013 (7)    \* BRG021 (3)  Kontainer 4 (total: 10)    \* BRG015 (6)    \* BRG003 (4)  Kontainer 5 (total: 9)    \* BRG017 (8)    \* BRG016 (1)  Kontainer 6 (total: 10)    \* BRG007 (6)    \* BRG019 (2)    \* BRG023 (2)  Kontainer 7 (total: 10)    \* BRG004 (7)    \* BRG012 (3)  Kontainer 8 (total: 10)    \* BRG022 (7)    \* BRG006 (3)  Kontainer 9 (total: 10)    \* BRG011 (5)    \* BRG002 (5)  Kontainer 10 (total: 9)    \* BRG018 (9)  Kontainer 11 (total: 10)    \* BRG024 (4)    \* BRG010 (2)    \* BRG014 (4)  Kontainer 12 (total: 10)    \* BRG020 (5)    \* BRG025 (5) |

## Simulated Annealing

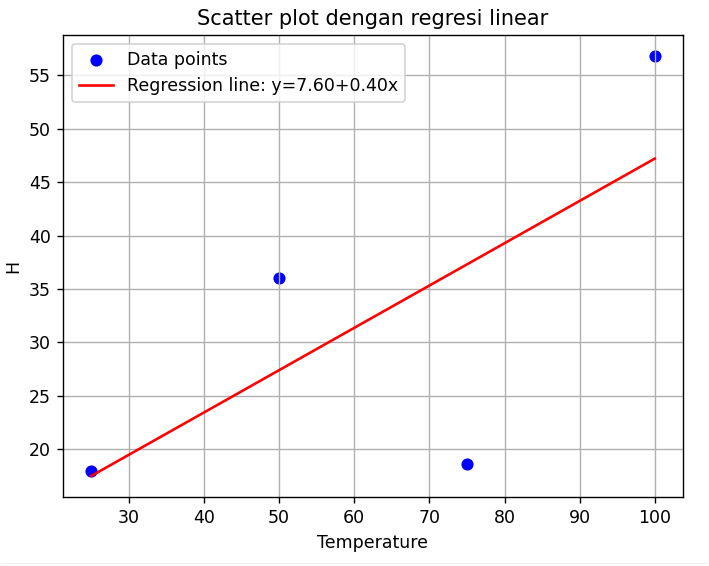
Tabel 4.8 plot dari Simulated Annealing dan probabilitas Worse move dipilih

|  |  |  |  |
| --- | --- | --- | --- |
| TC | plot | Best val | Bin |
| 1 |  | 25 | 10 |
| 2 |  | 13 | 6 |
| 3 |  | 57 | 12 |

**Pengaruh temperature**

Tabel 4.9 Pengaruh Temperature pada Simulated Annealing

|  |  |  |  |
| --- | --- | --- | --- |
| T | Iter 1-5 | mean | Std |
| 100 | 101, 57, 46, 23, 57 | 56.8 | 25.35 |
| 75 | 23, 34, 12, 12, 12 | 18.6 | 8.8 |
| 50 | 34, 33, 56, 23, 34 | 36 | 10.83 |
| 25 | 12, 22, 22, 22, 12 | 18 | 4.9 |



Gambar 4.1 Pengaruh Temperature pada Simulated Annealing

Koefisien korelasi r = -0.990

Dari Tabel diatas tidak ada satupun hasil dari simulated annealing yang mencapai local Optima.

## Genetic Algorithm

**Best H over generation:**

Tabel 4.10 Pengaruh testcase pada best H

|  |  |  |  |
| --- | --- | --- | --- |
| TC | plot | Best val | Bin |
| 1 |  | 3 | 10 |
| 2 |  | 3 | 5 |
| 3 |  | 12 | 13 |

**Fitness over generation :**

Tabel 4.11 Pengaruh testcase pada Fitness

|  |  |  |
| --- | --- | --- |
| TC | plot | Best val |
| 1 | A graph with blue lines  AI-generated content may be incorrect. | 3 |
| 2 |  | 3 |
| 3 |  | 2 |

**Pengaruh Populasi :**

Tabel 4.12 Pengaruh populasi pada Heuristic

|  |  |  |  |
| --- | --- | --- | --- |
| P | Iterasi 1-5 | mean | std |
| 5 | 12, 2, 12, 12, 12 | 10 | 10 |
| 10 | 2, 12, 12, 12, 12 | 10 | 4 |
| 15 | 2, 12, 2, 12, 8 | 7.2 | 4.49 |
| 20 | 7, 8, 2, 11, 12 | 8 | 3.52 |

A graph with a red line

AI-generated content may be incorrect.

Gambar 4.2 Pengaruh populasi pada H Genetic Algorithm

Koefisien korelasi = -0.798

**Pengaruh mutation rate :**

Tabel 4.13 Pengaruh Mutation Rate pada Heuristic

|  |  |  |  |
| --- | --- | --- | --- |
| M | Iterasi 1-5 | Mean | std |
| 5 | 12, 12, 2, 12, 12 | 10 | 4 |
| 10 | 2, 12, 12, 12, 12 | 10 | 4 |
| 15 | 2, 2, 12, 2, 2 | 4 | 4 |
| 20 | 2, 2, 12, 12, 2 | 6 | 4.9 |

Koefisien korelasi = -0.775

A graph with red line and blue dots

AI-generated content may be incorrect.

Gambar 4.3 Pengaruh Mutation Rate pada H Genetic Algorithm

**Pengaruh Fitness threshold :**

Tabel 4.14 Pengaruh Fitness threshold pada Heuristic

|  |  |  |  |
| --- | --- | --- | --- |
| T | Iterasi 1-5 | mean | std |
| 5 | 2, 12, 2, 12, 2 | 6 | 4.9 |
| 10 | 2, 12, 12, 12, 12 | 10 | 4 |
| 15 | 12, 11, 12, 12, 2 | 9.80 | 3.92 |
| 20 | 2, 3, 2, 12, 12 | 6.2 | 4.75 |

A graph with red lines and blue dots

AI-generated content may be incorrect.

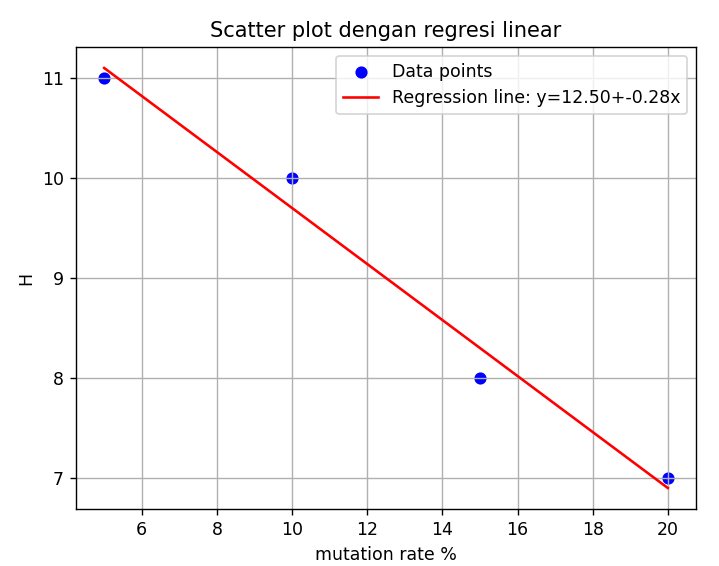
Gambar 4.4 Pengaruh Fitness Threshold pada H Genetic Algorithm

Koefisien korelasi r = 0.024

**Pengaruh Generasi :**

Tabel 4.15 Pengaruh Fitness threshold pada Heuristic

|  |  |  |  |
| --- | --- | --- | --- |
| G | Iterasi 1-5 | mean | std |
| 5 | 12, 7, 12, 12, 12 | 11 | 2 |
| 10 | 2, 12, 12, 12, 12 | 10 | 4 |
| 15 | 12, 12, 2, 7, 7 | 8.0 | 3.74 |
| 20 | 12, 12, 7, 2, 2 | 7 | 4.47 |



Gambar 4.5 Pengaruh Generasi pada H Genetic Algorithm

Koefisien korelasi r = -0.990

## Time Analysis

Dengan menggunakan variable control diatas, berikut adalah waktu run masing-masing algoritma :

|  |  |
| --- | --- |
| **Algoritma** | **Time Elapsed** |
| SAHC | 1.154 |
| SMHC | 1.772 |
| SHC | 3.971 |
| RRHC | 8.375 |
| Simulated Annealing | 6.778 |
| Genetic Algorithm | 10.399 |

## Initial State Analysis

Dengan menggunakan variable control diatas, berikut adalah waktu hasil masing-masing algoritma :

|  |  |  |
| --- | --- | --- |
| Algorima | Worst  (Final value) | First fit  (Final value) |
| SAHC | 12 | 12 |
| RRHC | 2 | 12 |
| Simulated Annealing | 12 | 113 |
| Genetic Algorithm | 2 | 2 |

.

## Figures and Tables

#### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### Referensi

1. Coffman, E. G., Garey, M. R., & Johnson, D. S. (1997). Approximation Algorithms for Bin Packing: A Survey. In D. S. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems (pp. 46–93). PWS Publishing Company.
2. Aarts, E., & Lenstra, J. K. (Eds.). (2003). Local Search in Combinatorial Optimization. Princeton University Press.
3. Kang, Seokchan & Lee, Jiyeong. (2017). Developing a Tile-Based Rendering Method to Improve Rendering Speed of 3D Geospatial Data with HTML5 and WebGL. Journal of Sensors. 2017. 1-11. 10.1155/2017/9781307.
4. Alvim, Adriana & Glover, Fred & Ribeiro, Celso. (1999). Local Search For The Bin Packing Problem.

[5] Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L. (1974). *Worst-case performance bounds for simple one-dimensional packing algorithms.* SIAM Journal on Computing, 3(4), 299–325. <https://doi.org/10.1137/0203025> .

[6] Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.

[7] Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L. (1974).*Worst-case performance bounds for simple one-dimensional packing algorithms.* SIAM Journal on Computing, 3(4), 299–325. <https://doi.org/10.1137/0203025> .

[8] Shehab, Mohammad & Khader, Ahamad Tajudin & Makhlouf, Laouchedi. (2018). A hybrid method based on Cuckoo search algorithm for global optimization problems. Journal of Information and Communication Technology. 17. 10.32890/jict2018.17.3.4.

[9] Jansen, Klaus & Kratsch, Stefan & Marx, Dániel & Schlotter, Ildikó. (2010). Bin Packing with Fixed Number of Bins Revisited. Journal of Computer and System Sciences. 79. 260-272. 10.1007/978-3-642-13731-0\_25.

[10] sdc

[11] dvsdv