OPTIMASI KEUNTUNGAN PEMBANGUNAN PERUMAHAN BERDASARKAN JUMLAH RUMAH SETIAP TIPE MENGGUNAKAN *PARTICLE SWARM OPTIMIZATION (PSO)*

SKRIPSI

Disusun oleh:
Rozaq Akbar
NIM: 125150207111101

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
TAHUN 2016

PENGESAHAN

OPTIMASI KEUNTUNGAN PEMBANGUNAN PERUMAHAN BERDASARKAN JUMLAH RUMAH SETIAP TIPE MENGGUNAKAN PARTICLE SWARM OPTIMIZATION (PSO)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rozaq Akbar
NIM: 125150207111101

Skripsi ini telah diperiksa dan dinyatakan lulus pada
29 Desember 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dian Eka Ratnawati, S.Si, M.Kom	Ir. Sutrisno, M.T
NIP: 197306192002122001	NIP: 19570325198701 1001

Mengetahui
Ketua Program Studi Informatika

Tri Astoto Kurniawan , S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 29 Desember 2016

Rozaq Akbar

NIM: 125150200111105

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan YME karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan penyusunan skripsi yang berjudul "OPTIMASI KEUNTUNGAN PEMBANGUNAN PERUMAHAN BERDASARKAN JUMLAH RUMAH SETIAP TIPE MENGGUNAKAN PARTICLE SWARM OPTIMIZATION (PSO)" ini dengan baik. Skripsi ini disusun untuk memenuhi syarat memperoleh gelar Sarjana Komputer pada Program Studi Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam penyusunan skripsi ini, penulis telah mendapat banyak bantuan, baik berupa moral ataupun materiil dari berbagai pihak. Ucapan terimakasih penulis sampaikan kepada:

1. Ibu Dian Eka Ratnanawati, S.Si, M.kom selaku dosen pembimbing skripsi 1 yang telah menyediakan waktu, tenaga dan pemikirannya untuk membimbing dan membantu penyelesaian penelitian ini dengan baik.
2. Bapak Ir. Sutrisno M.T selaku dosen pembimbing skripsi 2 yang telah menyediakan waktu dan memberikan banyak masukan sehingga skripsi ini dapat terselesaikan dengan baik.
3. Bapak Lubi Aldiani, selaku marketing di PT. Goldenindo Lestari yang telah bersedia menjadi narasumber serta membimbing penulis dalam pengerjaan skripsi ini.
4. Kedua orangtua dan seluruh keluarga besar atas segala nasehat, motivasi, kasih sayang, serta senantiasa mendukung dalam doa demi terselesaikannya skripsi ini dengan baik.
5. Seluruh teman dan kerabat atas segala bentuk dukungan, baik moral atau materil, serta kebersediaan waktu untuk berdiskusi dan membagi ilmunya sehingga sangat membantu penulis dalam setiap proses pengerjaan skripsi ini.
6. Pihak-pihak lain yang tidak dapat penulis sebut satu persatu, yang turut membantu penyelesaian skripsi baik secara langsung maupun tidak langsung.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat banyak kesalahan dan kekurangan. Oleh karena itu, penulis bersedia menerima kritik dan saran yang membangun demi memperbaiki diri. Harapan penulis semoga skripsi ini dapat memberikan manfaat pada semua pihak, terutama bagi penulis dan pengembang yang akan mengembangkan penelitian dari penulis.

Malang, 29 Desember 2016

Penulis
rozaqakbar@gmail.com

ABSTRAK

Tempat tinggal merupakan kebutuhan setiap individu. Selain hal tersebut, laju pertumbuhan penduduk, serta adanya urbanisasi juga mempengaruhi banyaknya permintaan. Perumahan merupakan salah satu jawaban dalam mengatasi permintaan akan perumahan tersebut. Di Indonesia pembangunan perumahan dijalankan oleh Perusahaan Umum Pembangunan Perumahan Nasional atau Perumnas yang selanjutnya akan diteruskan oleh pihak swasta yang dalam hal ini adalah *developer* atau pengembang. *Developer* dalam hal ini melakukan berbagai upaya dalam mencapai keuntungan yang maksimal. Oleh karena itu, dibutuhkan sistem optimasi dalam menangani masalah ini. Penggunaan *Particle Swarm Optimization* (PSO) akan sangat membantu dalam hal pencarian optimasi keuntungan. Dilihat dari beberapa kasus yang menggunakan PSO, hasil yang didapatkan adalah penjadwalan sumber daya proyek yang optimal, penyusunan posisi barang dalam peti kemas yang lebih optimal, dan akurasi yang lebih baik dari fungsi keanggotaan. Sesuai dengan pengujian yang dilakukan dengan menggunakan data dari Permata Garden Regency, didapatkan jumlah keuntungan yang lebih optimal dibandingkan dengan keuntungan yang telah dicapai.

Kata kunci: Keuntungan, PSO, Perumahan, Tempat Tinggal, Particle Swarm Optimization, Kebutuhan Dasar

ABSTRACT

The residence is the every individual need. In addition, the population growth rate, as well as the urbanization also affect the number of requests. Housing is one of the answers in addressing the demand for such housing. Housing construction in Indonesia run by Perumnas *or* Perusahaan Umum Pembangunan Perumahan Nasional *which would then be forwarded by the private sector which in this case is the developer. In this case, developer made various efforts to achieve maximum benefit. Therefore, system optimization is needed in addressing this issue. Use of Particle Swarm Optimization (PSO) will be very helpful to search for benefit optimization. Judging from several cases using PSO, the results obtained are scheduling resources optimum project, preparation of the position of the goods in a container which is more optimal, and better accuracy in a membership function. In accordance with the testing conducted using data from the Permata Garden Regency, found the number of benefits that more optimal than the gains that have been achieved.*

Keywords: *Benefit, PSO, Housing, Shelter, Particle Swarm Optimization, Basic Needs*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Tempat tinggal merupakan kebutuhan setiap individu. Fungsi tempat tinggal itu sendiri adalah untuk keamanan, tempat beristirahat, dan beraktifitas. Tempat tinggal tidak hanya dalam bentuk rumah saja, akan tetapi memiliki beberapa jenis lain, seperti rumah susun, perumahan, rumah deret, dan apartemen.

Dijelaskan bahwa kebutuhan akan sebuah tempat tinggal merupakan kebutuhan dasar, maka permintaan akan tempat tinggal menjadi semakin besar. Selain hal tersebut, laju pertumbuhan penduduk, serta adanya urbanisasi juga mempengaruhi banyaknya permintaan. Perumahan merupakan salah satu jawaban dalam mengatasi permintaan akan tempat tinggal tersebut.

Menurut UU No.4 Tahun 1992 (sesuai dengan UUD 1945) tentang Perumahan dan Pemukiman dijelaskan bahwa perumahan merupakan bagian dari pemukiman, perumahan itu sendiri sebagai rumah yang berfungsi seperti lingkungan tempat tinggal atau lingkungan hunian yang dilengkapi dengan sarana dan prasarananya. Penataan ruang dan kelengkapan sarana dan prasarana lingkungan dan sebagainya, dimaksudkan supaya lingkungan tersebut menjadi lingkungan yang sehat, aman, serasi, dan teratur serta dapat berfungsi sebagaimana semestinya (Pasal 1 angka 2). Pembangunan perumahan tidak dilakukan oleh pihak pemerintah saja, akan tetapi pihak swasta-pun diharapkan ikut serta. Di Indonesia pembangunan perumahan dijalankan oleh Perumnas yang dimiliki oleh Badan Usaha Milik Negara (BUMN). Perusahaan Umum Pembangunan Perumahan Nasional (Perumnas) didirikan oleh pemerintah sebagai solusi dalam menyediakan perumahan layak huni.

Perumnas menjadi pembuka awal yang selanjutnya akan diteruskan oleh pihak swasta, yaitu *developer* atau pengembang. Berdasarkan penelitian yang telah dilakukan oleh *Real Estate* Indonesia (REI), yang dikerjakan pemerintah sebesar 5 persen. Sedangkan 95 persen oleh swasta, di mana sekitar 80 persennya dilakukan anggota REI (Nidia. 2013). Peluang ini dimanfaatkan dengan sangat baik oleh para *developer* dalam mengambil keuntungan. Akan tetapi, keuntungan yang didapat oleh developer tersebut tidak memenuhi target, karena developer hanya menyesuaikan pembangunan perumahan dengan permintaan pasar dan masih belum dapat mengoptimalkan jumlah rumah di lahan yang disediakan.

Developer melakukan berbagai upaya dalam mencapai keuntungan yang maksimal. Mengoptimalkan pemanfaatan lahan yang ada sangatlah penting, karena keuntungan maksimal bisa dicapai apabila penggunaan lahan yang ada juga maksimal.

Penggunaan *Particle swarm optimization* (PSO) akan membantu dalam hal pencarian optimasi keuntungan. Dikarenakan basic dari PSO itu sendiri adalah proses optimasi untuk menentukan suatu target keuntungan maksimal yang ingin didapatkan. Selain hal tersebut, setiap partikel bergerak melalui ruang solusi dan mempunyai kemampuan untuk mengingat hasil terbaik sebelumnya dan dapat digunakan, serta bertahan setiap generasinya.

Particle swarm optimization (PSO) banyak digunakan dalam berbagai tujuan optimasi. Salah satunya terdapat penelitian sebelumnya yang berjudul "*Analisis dan Penerapan Algoritma Particle Swarm Optimization (PSO) pada Optimasi Penjadwalan Sumber Daya Proyek*". Hasil yang didapatkan adalah penjadwalan yang lebih baik, dan memberikan hasil maksimal pada sumber daya proyek. Penelitian ini telah berhasil memberikan solusi optimum untuk penjadwalan sumber daya proyek.

Berdasarkan latar belakang di atas, penulis akan merancang sistem berbasis komputer untuk memberikan solusi dalam hal optimasi keuntungan pembangunan perumahan yang berjudul, "OPTIMASI KEUNTUNGAN PEMBANGUNAN PERUMAHAN BERDASARKAN JUMLAH RUMAH SETIAP TIPE MENGGUNAKAN *PARTICLE SWARM OPTIMIZATION* (PSO)".

1.2 Rumusan masalah

Berdasarkan latar belakang diatas, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana menerapkan metode *Particle Swarm Optimization* ke dalam sistem optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe agar diperoleh keuntungan yang optimal?
2. Bagaimana pengaruh parameter *Particle Swarm Optimization* terhadap seberapa banyak keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe yang diperoleh?

1.3 Tujuan

Berdasarkan latar belakang serta rumusan masalah di atas, maka tujuan dari sistem ini adalah sebagai berikut:

1. Menerapkan metode *Particle Swarm Optimization* ke dalam sistem optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe agar diperoleh keuntungan yang optimal.
2. Menguji pengaruh parameter *Particle Swarm Optimization* terhadap seberapa banyak keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe.

1.4 Manfaat

Pembuatan perangkat lunak memberikan manfaat sebagai berikut:

1. Mendapatkan perhitungan hasil keuntungan terbaik dari hasil pembagian luas tanah dalam menentukan jumlah rumah setiap tipe.
2. Membantu arsitek dalam mengambil keputusan pembagian luas tanah.

1.5 Batasan masalah

Dalam perancangan pembuatan sistem ini terdapat beberapa batasan masalah, yaitu:

1. Optimasi keuntungan didasarkan pada pembagian luas tanah dalam mendapatkan keuntungan terbaik.
2. Rencana anggaran biaya untuk setiap tipe rumah berbeda tergantung luas tanah dan luas bangunan akan tetapi sudah ditentutkan oleh arsitek.
3. Tidak ada penambahan lain dalam pembangunan rumah.

1.6 Sistematika pembahasan

Bab I Pendahuluan

Bab I berisi latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat penulisan, sistematika pembahasan, dan jadwal pelaksanaan.

Bab II Landasan Kepustakaan

Bab II berisi uraian serta pembahasan tentang teori, konsep, dan metode atau sistem dari literatur ilmiah, yang berhubungan dengan tema dan masalah atau pertanyaan penelitian.

Bab III Metodologi

Bab III berisi mengenai dasar teori, analisis kebutuhan, analisis hasil, dan kesimpulan.

Bab IV Perancangan Sistem

Bab IV berisi uraian mengenai rancangan sistem serta studi perancangan sistem.

Bab V Implementasi Sistem

Bab V berisi uraian mengenai implementasi pengujian, analisis, dan pengambilan kesimpulan.

Bab VI Pengujian dan Analisis Sistem

Bab VI berisi uraian mengenai implementasi pengujian dan analisis serta dasar pengambilan kesimpulan.

Bab VII Penutup

Bab VII berisi hasil kesimpulan dan saran-saran mengenai keseluruhan isi.

BAB 2 LANDASAN KEPUSTAKAAN

Pada landasan kepustakaan ini menjelaskan tentang teori dari sumber pustaka yang terkait dengan kasus optimasi atau penerapan algoritma *Particle Swarm Optimization*. Terdapat pula kajian pustaka yang menjelaskan penelitian-penelitian sebelumnya dan berhubungan dengan topik skripsi serta menunjukkan perbedaan skripsi tersebut terhadap penelitian yang sebelumnya.

2.1 Kajian Pustaka

Particle Swarm Optimization digunakan dalam berbagai tujuan optimasi. Salah satunya terdapat penelitian sebelumnya, dengan judul "Analisis dan Penerapan Algoritma *Particle Swarm Optimization* (PSO) pada Optimasi Penjadwalan Sumber Daya Proyek". Hasil yang didapatkan yaitu penjadwalan yang lebih baik, dan memberikan hasil maksimal pada sumber daya proyek. Penelitian ini telah berhasil memberikan solusi optimal dalam penjadwalan sumber daya proyek.

Selain itu pada tinjauan kedua yang berjudul "Optimasi Pola Penyusunan Barang dalam Peti Kemas Menggunakan Algoritma *Particle Swarm Optimization*". Penelitian ini telah berhasil membantu dalam proses penyusunan posisi barang didalam sebuah peti kemas. Model ini membantu tugas penyusunan, sehingga jumlah volume barang didalam peti kemas dapat lebih dioptimalkan.

Pada tinjauan ketiga yang berjudul "Implementasi Algoritma *Particle Swarm Optimization* Untuk Optimasi Fungsi Keanggotaan Pada Kondisi Penderita Penyakit Hepatitis". Penelitian ini telah berhasil mendapatkan kombinasi parameter PSO yang memberikan hasil terbaik, yang kemudian digunakan dalam mengukur tingkat akurasi yang dihasilkan menggunakan sistem inferensi *Fuzzy* Mamdani. Setelah dilakukan pengujian, nilai yang didapatkan lebih baik dibandingkan nilai akurasi yang dihasilkan oleh fungsi keanggotaan yang belum dioptimasi.

Pada penelitian ini, akan digunakan algoritma PSO untuk memecahkan kasus optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe.

Perbandingan objek dan metode penelitian yang telah dilakukan terhadap tinjauan pustaka dari penelitian sebelumnya ditunjukkan oleh Tabel 2.1 berikut.

Tabel 2.1 Tinjauan Pustaka

No.	Judul	Objek	Metode	Hasil
1.	*Analisis dan Penerapan Algoritma Particle Swarm Optimization (PSO) pada Optimasi Penjadwalan Sumber Daya Proyek*	Sumber daya proyek	*Particle Swarm Optimization (PSO)*	Penjadwalan sumber daya yang optimal
2.	Optimasi Pola Penyusunan Barang dalam Peti Kemas Menggunakan Algoritma Particle Swarm Optimization	Barang didalam peti kemas	*Particle Swarm Optimization (PSO)*	Penyusunan posisi dalam peti kemas yang lebih optimal
3.	Implementasi Algoritma Particle Swarm Optimization Untuk Optimasi Fungsi Keanggotaan Pada Kondisi Penderita Penyakit Hepatitis	Pasien hepatitis	*Particle Swarm Optimization (PSO)*	Akurasi yang lebih baik dari fungsi keanggotaan

2.2 Pembangunan Perumahan

Pembangunan perumahan perlu mendapatkan prioritas, dikarenakan kebutuhan manusia akan sebuah tempat tinggal. Undang-undang No. 4 Tahun 1992 tentang Perumahan dan Permukiman, menyatakan bahwa : Perumahan adalah sekelompok rumah yang berfungsi sebagai lingkungan tempat tinggal atau lingkungan hunian yang dilengkapi dengan prasarana dan sarana lingkungan. Didalam pembangunan perumahan terdapat beberapa tahapan (Aldaini, 2016), antara lain:

1. Penentuan Daerah Perumahan

Sebuah perusahan developer akan melakukan penentuan untuk memilih daerah yang akan dikembangkan serta dijadikan bisnis jual beli rumah. Daerah yang dipilih harus memiliki beberapa kriteria yang ditentukan oleh developer tersebut, sehingga daerah tersebut menjadi daerah yang layak dalam pembangunan perumahan, bahkan berprospek tinggi untuk harga jual unit rumah.

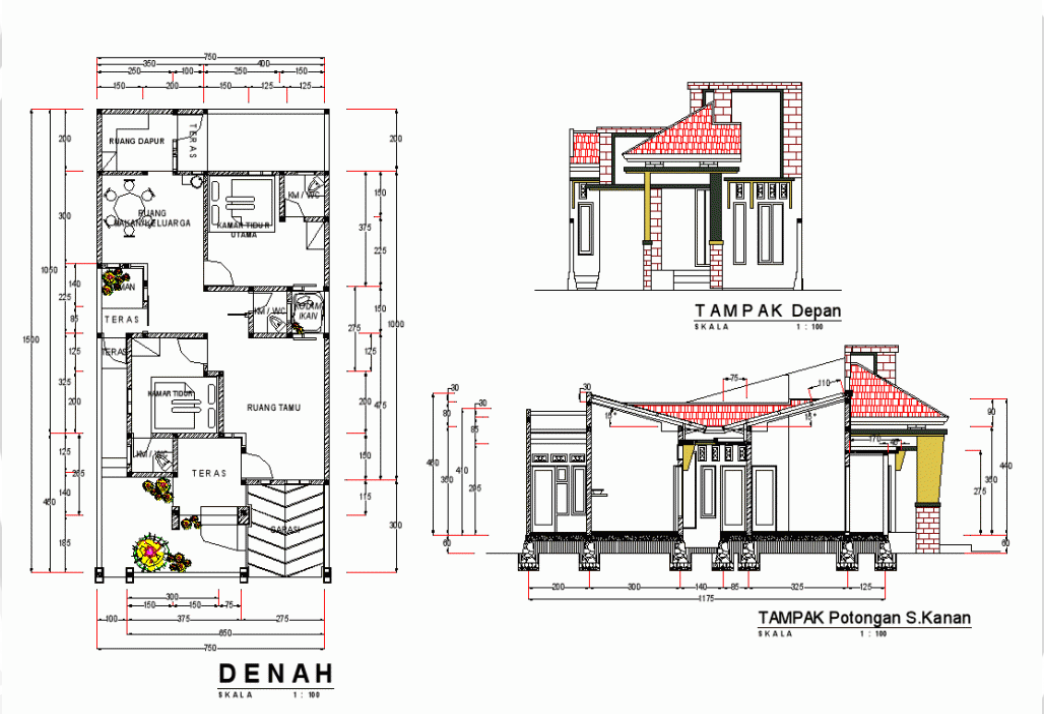
2. Pembelian Tanah

Setelah menetukan daerah, maka developer akan membeli tanah didaerah tersebut. Pembelian tanah dimaksudkan agar sertifikat tanah yang akan menjadi lahan bangunan dari perumahan tersebut sama, yaitu menjadi satu sertifikat induk. Karena tanah memiliki berbagai macam jenis sertifikat, antara lain sertifikat HGB (Hak Guna Bangunan), hak milik, girik, dan lain-lain.

3. Perancangan Denah serta Desain Rumah

Dalam hal ini, perancangan denah atau kerangka dilakukan oleh seorang arsitek. Arsitek melakukan pembagian luas tanah, menepatkan sarana serta prasarana disekitar perumahan, membuat gambar desain rumah dan rencana anggaran dari pembangunan tersebut.

Berikut ini merupakan contoh denah dan desain rumah yang ditunjukkan oleh Gambar 2.1 dan Gambar 2.2.



Gambar 2.1 Contoh denah

Gambar 2.2 Contoh desain rumah

Berikut Gambar 2.3 tentang rencana anggaran biaya pembangunan rumah.

NO.	URAIAN PEKERJAAN	Satuan	harga Satuan	Volume	jumlah
I. PEKERJAAN PERSIAPAN					
1	Air kerja	Ls	500.000,00	1,00	500.000,00
2	Lisrik kerja	Ls	475.000,00	1,00	475.000,00
3	Pembongkaran Dinding Tembok Bata merah	M2	31.600,00	9,90	312.840,00
4	Pembongkaran Beton bertulang	m3	256.060,00	2,52	645.271,20
5	Pembongkaran atap genteng dan rangka atap lama	M2	22.500,00	70,00	1.575.000,00
6	Pembongkaran Plafon GRC/asbes, tripleks > 9mm	M2	15.500,00	80,00	1.240.000,00
7	pembongkaran Keramik Lama	M2	17.500,00	47,00	822.500,00
8	Mobilisasi Dalam Kota Bogor (5s/d15orang pekerja bangunan)	Ls	200.000,00	1,00	200.000,00
				sub Total	**5.770.611,20**
II. PEKERJAAN TANAH					
1	Galian tanah untuk pondasi telapak sedalam 1 m	M3	58.500,00	12,00	702.000,00
2	Galian tanah untuk pondasi Jalur sedalam 0.5 m	M3	58.500,00	1,00	58.500,00
3	Urugan pasir	M3	210.000,00	6,00	1.260.000,00
4	Urugan tanah	M3	148.000,00	6,00	888.000,00
				sub Total	**2.908.500,00**
III. PEKERJAAN PONDASI					
1	Pasangan Pondasi Batu Kali 1:5	M3	672.000,00	1,20	806.400,00
2	Sloof 20x30 cm Beton K-225	m3	5.542.684,00	1,20	6.651.220,80
3	Pondasi Telapak 60x60 cm Beton K-300	m3	6.372.777,62	3,17	20.188.959,50
				sub Total	**27.646.580,30**

IV. PEKERJAAN BETON NON STRUKUR DAN BETON STRUKTUR					
1	Membuat Balok Beton Bertulang (125 Kg Besi + Bekisting)	M3	5.895.014,50	2,07	12.202.680,02
2	Membuat Plat Lantai tebal 10 Cm Beton Bertulang (100 Kg Besi + Bekisting)	M3	6.372.777,62	9,60	61.178.665,15
3	Membuat Kolom Beton Bertulang (125 Kg Besi + Bekisting)	M3	4.390.000,00	2,64	11.589.600,00
4	Membuat Ring Balk Beton Bertulang (175 Kg Besi + Bekisting)	M3	5.542.684,00	1,04	5.764.391,36
				sub Total	**90.735.336,53**
V. PEKERJAAN DINDING					
1	Pasangan Bata Merah Tebal 1 Bata, Campuran 1 Pc : 3 Psr	M2	115.000,00	297,50	34.212.500,00
2	Plesteran dan aci 1 : 3	M2	28.480,00	587,00	16.717.760,00
				sub Total	**50.930.260,00**
VI. PEKERJAAN PELAPIS LANTAI					
1	Screeding di bawah lantai T=5cm	m3	850.000,00	13,12	11.154.125,00
2	Pasang Lantai keramik ukuran 40 x 40 cm exs Hercules	M2	135.500,00	120,70	16.354.850,00
3	Pasang Lantai Keramik polos 20 x 20 cm exs Pegasus	M2	115.000,00	4,50	517.500,00
4	Pasang Lantai Keramik polos 30 x 30 cm exs arwana	M2	127.300,00	15,00	1.909.500,00
5	Roman 40 x 40 Golongan A	M2	165.500,00	30,00	4.965.000,00
6	Pasang lantai Parquette tangga Exs China	M2	385.700,00	9,76	3.764.432,00
7	Pasang list sudut lantai Parquette tangga	M'	35.500,00	18,00	639.000,00
8	Pasang Batu templek lantai	M2	285.000,00	24,00	6.840.000,00
9	pasang keramik meja beton kitchen &dapur30x30 set exs roman golongan C	M2	163.500,00	10,00	1.635.000,00
				sub Total	**47.779.407,00**
VII. PEKERJAAN PELAPIS DINDING					
1	Pasang Dinding Keramik motif Roman 30 x 60 Golongan A	M2	188.419,55	70,48	13.280.375,14
2	Pasang Dinding Keramik polos Roman 20 x 25 exs arwana	M2	132.500,00	16,50	2.186.250,00
3	pasang keramik motif 20X25 dinding meja Beton kitchen set Exs arwana	M2	122.500,00	3,00	367.500,00
				sub Total	**15.834.125,14**
VIII. PEKERJAAN KUSEN, PINTU, JENDELA, GLASS BOX					
1	Pasang Kusen Pintu & Jendela Kayu kamper medan	M'	335.000,00	90,31	30.253.850,00
2	Pasang Pintu utama panel double kayu kamper medan uk 2x(2x0.7m)	BH	1.240.000,00	2,00	2.480.000,00
3	Pasang Pintu Triplek Rangkap , Rangka Kayu kamper medan	BH	760.000,00	3,00	2.280.000,00
4	Pasang jendela kaca rayban uk. 190x70 kayu kamper medan	BH	675.000,00	2,00	1.350.000,00
5	Pasang jendela kaca rayban uk. 125x65 kayu kamper medan	BH	530.000,00	13,00	6.890.000,00
6	Pasang jendela kaca rayban uk. 55x45 kayu kamper medan	BH	230.000,00	2,00	460.000,00
7	Pasang Pintu UPVC kamar mandi	BH	630.000,00	4,00	2.520.000,00
				sub Total	**46.233.850,00**
IX. PEKERJAAN RANGKA ATAP DAN PENUTUP ATAP					
A	**PEKERJAAN RANGKA ATAP**				
1	Pekerjaan Konstruksi Rangka atap Baja Ringan 0,75mm Zinc alume exs blue scope	M2	127.500,00	162,40	20.706.000,00
2	Pekerjaan List Plang GRC	M'	45.000,00	39,20	1.764.000,00
				sub Total	**22.470.000,00**
B	**PEKERJAAN PENUTUP ATAP**				
1	Pasang Nok/Bubungan Genteng keramik glazur	M'	135.000,00	11,70	1.579.500,00
2	plastik fiber pelapis sebelum atap anti bocor	M2	30.000,00	165,00	4.950.000,00
3	Pasang Atap Genteng morando glazur	M2	115.000,00	165,00	18.975.000,00
				sub Total	**25.504.500,00**
X. PEKERJAAN PENUTUP PLAFOND					
1	Plafond gypsum Exs jayaboard 9 mm, rangka besi Hollow biasa*	M2	87.500,00	197,00	17.237.500,00
2	List Plafon gypsum profile Minimalis 10cm	M'	28.500,00	262,80	7.489.800,00
3	cat plafond Exs vinilex	M2	28.000,00	197,00	5.516.000,00
				sub Total	**30.243.300,00**

XI. PEKERJAAN SANITAIR					
1	Memasang Kloset Duduk Toto type CW660J/SW660J/setara	BH	2.395.000,00	2,00	4.790.000,00
2	Memasang Kloset Duduk Toto type CW823J/setara	BH	6.250.000,00	1,00	6.250.000,00
3	Memasang Kloset Jongkok Toto type CE7/setara	BH	375.000,00	1,00	375.000,00
4	Memasang tempat sabun Toto S156N setara	BH	92.500,00	4,00	370.000,00
5	Memasang kran dan Shower dinding Toto TX 432 SD	BH	844.000,00	3,00	2.532.000,00
6	memasang Lavatory/wastafel Toto L529V1/setara	BH	412.000,00	2,00	824.000,00
7	*Memasang Bak mandi Fibreglass ukuran 60*	*BH*	*610.000,00*	*1,00*	*610.000,00*
8	memasang Floordrain Toto TX1AN	BH	315.000,00	6,00	1.890.000,00
9	memasang Jet Washer toto THX 20 NBPIV	BH	235.000,00	3,00	705.000,00
10	Memasang Kran dinding Toto T23B13	BH	233.500,00	6,00	1.401.000,00
11	Memasang Kran angsa kitchen zink Toto TX 604 TDN Kitchen Taps	BH	835.000,00	1,00	835.000,00
12	Memasang Kitchen zinc (2 lubang) Exs Butter fly	BH	776.000,00	1,00	776.000,00
13	memasang Exhause ventilator panasonic 12"	BH	577.000,00	1,00	577.000,00
14	memasang Exhause fan kamar mandi exs panasonic FV-15EGK	BH	555.450,00	1,00	555.450,00
				sub Total	**22.490.450,00**
XII. PEKERJAAN PERPIPAAN					
1	Memasang pipa PVC type D diameter 3" exs wavin	M'	37.500,00	32,00	1.200.000,00
2	piting-piting Memasang pipa PVC type D diameter 3" exs rucika	Ls	74.200,00	1,00	74.200,00
3	Memasang pipa PVC type D diameter 4" exs wavin	M'	39.500,00	27,00	1.066.500,00
4	piting-piting Memasang pipa PVC type D diameter 4" exs rucika	Ls	88.500,00	1,00	88.500,00
5	Memasang pipa PVC type D diameter 2" exs wavin	M'	25.000,00	12,00	300.000,00
6	piting-piting Memasang pipa PVC type D diameter 2" exs rucika	Ls	50.000,00	1,00	50.000,00
7	Memasang pipa PVC type AW diameter 1/2" exs wavin	M'	21.000,00	49,00	1.029.000,00
8	piting-piting Memasang pipa PVC type D diameter 1/2" exs rucika	Ls	98.500,00	1,00	98.500,00
9	membuat bak control 40x60 tutup beton	BH	250.000,00	1,00	250.000,00
				sub Total	**4.156.700,00**
XIII. PEKERJAAN PENUTUP BESI					
1	Pasang railling balkon dan tangga	M2	395.000,00	55,60	21.962.000,00
2	Pasang Canopy Polycarbonat	M2	450.000,00	27,00	12.150.000,00
3	Pasang Pagar Hollo minimalis Tinggi 1,8 meter*	M'	395.000,00	15,30	6.043.500,00
4	Pasang gerbang Lipat Pagar Teralis*	M'	495.000,00	6,60	3.267.000,00
5	Pasang gerbang sliding Pagar Teralis Tinggi 2,2m panjang 5m	M'	495.000,00	11,00	5.445.000,00
				sub Total	**48.867.500,00**
XIV. PEKERJAAN ACCESSORIES, PENGUNCI DAN PENGGANTUNG					
1	Pasang Kunci bulat putar Kamar Mandi	BH	45.000,00	4,00	180.000,00
2	Pasang Engsel Pintu	BH	29.000,00	10,00	290.000,00
3	Pasang Engsel Jendela	BH	23.000,00	34,00	782.000,00
4	Pasang Kait Angin	BH	17.500,00	34,00	595.000,00
5	Pasang Pegangan Pintu / Door Handle pintu double tripleks	BH	135.000,00	3,00	405.000,00
6	Pasang Pegangan Pintu / Door Handle pintu kayu solid panel utama	BH	189.000,00	4,00	756.000,00
7	Pasang Kunci pintu double panel Exs solid	BH	199.750,00	1,00	199.750,00
8	Pasang Grendel / sloot jendela	BH	19.500,00	34,00	663.000,00
9	Pasang Tarikan jendela	BH	19.000,00	34,00	646.000,00
				sub Total	**4.516.750,00**
XV. PEKERJAAN PENGECATAN					
1	Pengecatan dinding interior Kwalitas Sedang Vinilex / setara	M2	27.500,00	729,30	20.055.750,00
2	Pengecatan dinding eksterior Kwalitas Baik Dulux weathershield/ setara	M2	43.500,00	135,00	5.872.500,00
3	Aquaproofing dinding eksterior (Lt2.belakang)	M2	33.500,00	35,00	1.172.500,00
4	finishing cat fliture kusen, daun pintu dan daun jendela baru	M2	30.060,50	34,00	1.022.057,00
				sub Total	**28.122.807,00**

XVI. PEKERJAAN LISTRIK					
1	Pasang titik Stop kontak cab: eterna, panasonic	TTK	155.000,00	18,00	2.790.000,00
2	Pasang titik Lampu cab : eterna	TTK	125.000,00	57,00	7.125.000,00
3	Pasang saklar ganda panasonic	BH	31.500,00	16,00	504.000,00
4	Pasang saklar tunggal panasonic	BH	28.500,00	5,00	142.500,00
5	Pasang titik listrik Exhause fan kamar mandi	TTK	125.000,00	2,00	250.000,00
6	Pasang Lampu SL :philips 9 = 20 watt + tabung down light	M2	115.000,00	41,00	4.715.000,00
7	Pasang Lampu SL :philips 9 = 20 watt	M2	65.000,00	15,00	975.000,00
8	Tabung lampu dinding	BH	165.000,00	2,00	330.000,00
				sub Total	**16.831.500,00**
XVII. LAIN-LAIN					
1	kithen set multipleks 18mm, minimalis finishing melamics	M'	1.800.000,00	6,00	10.800.000,00
2	kithen set multipleks 18mm, minimalis finishing TACONSHEET	M'	1.700.000,00	8,00	13.600.000,00
3	Teralis Jendela-jendela Lantai 2=16titik(hanya bahan besi dibyr,biaya pembuatan free)	Ls	500.000,00	1,00	500.000,00
4	bongkar muat dan Buang puing bongkaran keluar lokasi(dengan dump truck kaps 7m3)	Trip	500.000,00	7,00	3.500.000,00
5	Pembersihan Akhir	m2	2.000,00	391,00	782.000,00
				sub Total	**29.182.000,00**
					520.224.177,17

Gambar 2.3 Rencana Anggaran Biaya

4. Pembangunan

Pekerjaan pembangunan diserahkan pada pekerja menurut hasil desain serta rencana anggaran yang diberikan oleh arsitek tersebut sesuai dengan dana yang dilimpahkan oleh pihak developer.

2.3 Tipe Rumah

Tipe rumah merupakan ukuran (luas tanah dan luas bangunan) atau jenis dari sebuah rumah, dimana rumah tersebut dibedakan. Tipe rumah didasarkan pada hasil yang diberikan arsitek yang membuat denah dan desain rumah (hasil wawancara: Aldaini, 2016).

1. Luas Tanah dan Luas Bangunan

Luas tanah (LT) merupakan ukuran yang menyatakan panjang dan lebar rumah tersebut. Luas tanah ini biasanya digunakan dalam tipe rumah. Sedangkan untuk luas bangunan (LB) adalah seberapa besar rumah tersebut dibangun diatas tanah yang disediakan.

2. Model Bangunan

Saat ini terdapat banyak sekali model rumah, seperti klasik, minimalis, dll. Model seperti ini dapat dibedakan berdasarkan desain tampilan depan maupun bentuk ruangan rumah tersebut. Sehingga setiap rumah yang awalnya memiliki bentuk bangunan dan desain yang sama, kemudian dibedakan. Terkadang model juga bergantung terhadap luas tanah, karena beberapa model tidak bisa diaplikasikan pada luas tanah yang minim.

2.4 Keuntungan

Keuntungan adalah selisih lebih antara harga penjualan yang lebih besar dan harga pembelian atau biaya produksi; keuntungan (Kamus Besar Bahasa Indonesia). Dalam bidang pembangunan perumahan, keuntungan dihitung berdasarkan pembagian lahan yang tersedia serta penentuan harga setiap

bangunan yang terdapat dilahan tersebut. Sehingga hasil dari penjualan setiap bangunan yang terdapat diatas lahan tersebut melebihi harga pembelian lahan, pembelian bahan bangunan, dan biaya lainnya.

2.5 *Particle Swarm Optimization (PSO)*

Algoritma ini pertama kali dikenalkan oleh Dr. Eberhart dan Dr. Kennedy pada tahun 1995 dalam sebuah konferensi jaringan syaraf di Perth, Australia. Algoritma optimasi ini meniru proses yang terjadi dalam kehidupan populasi burung dan ikan dalam bertahan hidup. Algoritma *Particle Swarm Optimization* merupakan teknik optimasi berbasis *stochastic* yang terinspirasi dari tingkah laku sosial sekawanan burung atau sekumpulan ikan. Analoginya diambil dari kebiasaan sekolompok burung yang secara acak mencari makanan disuatu area. Di area tersebut hanya ada sepotong makanan yang akan dicari. Seluruh burung tidak mengetahui dimana makanan tersebut. Mereka mengetahui jarak makanan tersebut di setiap iterasi. Jadi strategi apa yang terbaik untuk menemukan makanan (Zerda, 2009). Berdasarkan pemikiran tersebut, algoritma ini menjadi salah satu metode optimasi yang termasuk kategori algoritma *behaviorally inspired* sebagai alternatif dari algoritma genetika, yang biasa disebut *evolution-based procedures*.

Particle Swarm Optimization diinisialisasi oleh sebuah populasi solusi secara acak, selanjutnya mencari titik optimum dengan cara meng-*update* tiap hasil pembangkitan (Universitas Sumatera Utara). Dalam konteks optimasi *multi variabel*, kawanan diasumsikan memiliki ukuran tertentu dengan setiap posisi partikel awalnya terletak di suatu lokasi yang acak dalam ruang multi dimensi. Setiap partikel diasumsikan memiliki dua karakteristik, posisi dan kecepatan. Setiap partikel bergerak dalam ruang tertentu dan mengingat posisi paling baik yang pernah dilalui atau ditemukan terhadap sumber makanan, dalam hal ini adalah nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi paling baik kepada partikel lain dan menyesuaikan posisi serta kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi terbaik tersebut.

2.5.1 Parameter PSO

Ada dua langkah utama ketika menerapkan algoritma PSO pada masalah optimasi, yaitu (Zerda, 2009):

1. Representasi solusi
2. Fungsi obyektif atau fungsi *fitness*-nya

Parameter yang dibutuhkan pada algoritma PSO antara lain (Ria, 2009) :

1. Jumlah partikel *range*-nya 20 - 40. Sebenarnya dalam sebagian masalah 5 - 10 partikel cukup besar untuk mendapatkan hasil yang bagus. Untuk masalah yang sangat sulit atau khusus, bagus untuk mencoba 100 atau 200 partikel.

2. Dimensi dari partikel Ini ditentukan dari masalah yang dioptimasi.
3. Range dari partikel Ini juga ditentukan dari masalah yang dioptimasi. Dapat menspesifikasikan *range* yang berbeda untuk dimensi yang berbeda dari partikel.
4. Kondisi berhenti mencapai nilai iterasi maksimum, perulangan telah mencapai kondisi nilai optimal atau minimal *error* yang diinginkan.
5. *Inertia weight* (w), dalam algoritma PSO digunakan sebagai keseimbangan eksplorasi global dan lokal secara utama, yaitu kontrol oleh *inertia weight* yang merupakan parameter penurunan kecepatan untuk menghindari stagnasi partikel di lokal optimum.

2.5.2 Tahapan PSO

Proses algoritma PSO yaitu sebagai berikut (Ria, 2009):

1. Inisialisasi partikel secara acak (setiap partikel merepresentasikan solusi yang mungkin untuk masalah optimasi).
2. Inisialisasi posisi (*Xid*) dan kecepatan dari setiap partikel (*Vid*).
3. Menghitung nilai fluktuasi dari setiap partikel F_i berdasarkan formula fungsi objektif (*fitness*) dan model yang telah ditentukan sesuai dengan masalah optimasinya.

$$\text{Fitness}=\frac{1}{(Tipe1*harga+Tipe2*harga+Tipe3*harga)+(Tipe1.selisih+Tipe2*selisih+Tipe3*selisih)*C} \quad \text{.....(2.1)}$$

Keterangan :

Tipe1 = jumlah rumah dengan tipe 30/72

Tipe2 = jumlah rumah dengan tipe 38/78

Tipe3 = jumlah rumah dengan tipe 45/91

C = konstanta

Harga = harga jual tiap rumah

Selisih = selisih jumlah rumah rekomendasi pakar dan keluaran sistem

4. Membandingkan nilai fluktuasi *Fi* dengan nilai terbaiknya *Pid (local best)*,

   jika *Fi < Pid* , maka *Pid* diganti dengan *Fi* (2.2).

5. Untuk setiap partikel, bandingkan nilai fluktuasi *Fi* dengan nilai terbaiknya dalam populasi *Pgd (global best)*,

   jika *Fi < Pgd* , maka *Pgd* diganti dengan *Fi* (2.3).

6. Berdasarkan persamaaan 4 dan 5 , kecepatan *(Vi)* dan posisi dari partikel *(Xi)* diubah.

   Rumus perubahan kecepatan *(Vi)*:

$$V_{id}^{t+1} = w * v_{id}^{t} + c_1 * Rand() * \left(p_{id} - x_{id}^{t}\right) + c_2 * Rand() * \left(p_{gd} - x_{id}^{t}\right) \text{..... (2.4)}$$

Rumus perubahan posisi *(Xi)*:

$$x_{id}^{t+1} = x_{id}^{t} + V_{id}^{t+1} \quad \text{..... (2.5)}$$

Rumus W *inersia (w)*:

$$w = w_{max} - \frac{w_{max} - w_{min}}{I_{max}} * I \quad \text{..... (2.6)}$$

Keterangan:

v_{id}^{t} = kecepatan partikel ke-i pada dimensi ke-d pada iterasi ke-t

x_{id}^{t} = posisi partikel ke-i pada dimensi ke-d pada iterasi ke-t

w = bobot inersia

I = Iterasi

I_{max}= iterasi maksimum

c_1, c_2 = learning rate

p_{id} = P best atau posisi terbaik pada partikel x_{id}^{t}

p_{gd} = P global best atau posisi terbaik pada partikel secara keseluruhan

7. Jika telah mencapai kondisi akhir (mencapai nilai iterasi maksimum) maka perulangan berhenti dan nilai optimumnya didapatkan namun jika belum maka diulangi pada nomor 3 dan seterusnya.

2.6 *Random Injection*

Pada ruang pencarian yang tidak terlalu besar, sering dijumpai pencapaian konvergensi dini. Hal ini disebabkan karena partikel lebih cepat menemukan posisi terbaik global dalam ruang pencarian yang kecil dan disebabkan oleh kurangnya diversitas populasi setelah melewati sekian generasi (Mahmudy, 2014). Untuk mengatasi hal tersebut dan membuat partikel lebih teliti atau bertahap dalam melakukakan eksploitasi lokal dan eksplorasi global, maka diterapkan sistem *random injection. Random injection* dilakukan dengan menginisialisasi kembali posisi n partikel setiap g *interval* iterasi.

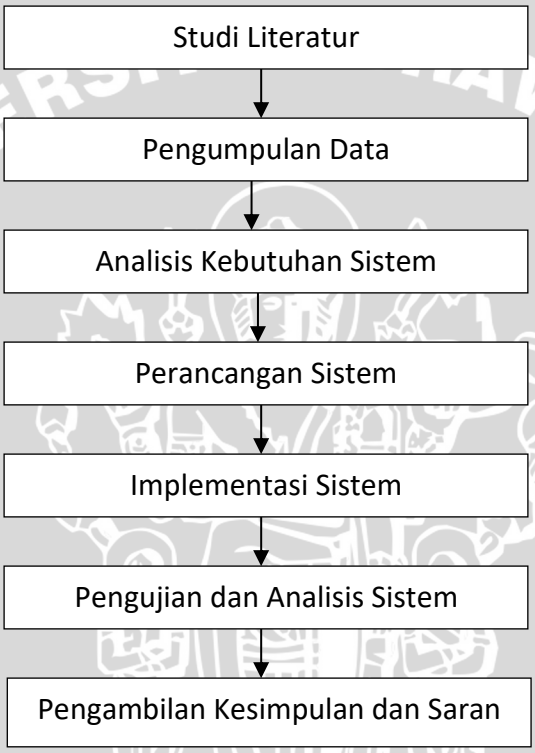
Pada kasus ini ditentukan jumlah partikel yang disisipkan adalah 60% dari ukuran *swarsize* dan *interval injection* adalah setiap kelipatan 2 iterasi. *Random injection* dilakukan dengan mengevaluasi nilai *fitness* partikel saat memasuki *interval injection*. Nilai *fitness* akan diurutkan, kemudian 60% dari jumlah partikel dengan nilai *fitness* yang buruk akan digantikan dengan partikel acak yang baru. Untuk partikel baru tersebut, maka diberikan nilai kecepatan $V = 0$ dan *pBest* sama dengan partikel itu sendiri. Hal ini sama seperti proses inisialisasi awal partikel.

2.7 Pengujian

Uji coba dilakukan untuk melihat kombinasi nilai parameter yang akan memberikan hasil terbaik dan kemudian mengukur hasil akurasi yang dihasilkan oleh proses optimasi tersebut (Ratna, dkk. 2013). Hasil dari pengujian parameter ini akan mempengaruhi *fitness*, dimana *fitness* tersebut merupakan jawaban dari sistem optimasi ini. Selanjutnya, hasil dari sistem ini diharapkan dapat diterapkan dan menjadi acuan dalam perhitungan-perhitungan lainnya.

BAB 3 METODOLOGI

Dalam metodologi penelitian dan perancangan akan dijelaskan secara umum tahapan dalam implementasi *PSO* untuk optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe. Tahapan penelitian meliputi analisis kebutuhan sistem, deskripsi umum sistem, data yang digunakan, perancangan sistem, proses PSO, perancangan *user interface*, dan perancangan pengujian.Untuk dapat memberikan kemudahan dalam menjelaskan metodologi yang kami gunakan, maka kami menggunakan diagram alir seperti pada Gambar 3.1.



Gambar 3.1 Diagram Alir Optimasi Keuntungan Pembangunan Perumahan Berdasarkan Jumlah Rumah Setiap Tipe Menggunakan*Particle Swarm Optimization (PSO)*

3.1 Studi Literatur

Studi literatur yang kami lakukan adalah dengan cara mengumpulkan serta mempelajari berbagai macam literatur yang berkaitan dengan optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe menggunakan *Particle Swarm Optimization (PSO)*, diantaranya :

1. Pembangunan perumahan.
2. Rencana Anggaran Biaya (RAB) setiap pembangunan satu unit tipe rumah.

3. Perhitungan keuntungan setiap satu unit tipe rumah.
4. Algoritma *Particle Swarm Optimization (PSO)*.
5. *Random Injection*

Sumber yang saya peroleh untuk melakukan studi literatur ini berupa jurnal, laporan ilmiah, dan paper.

3.2 Pengumpulan Data

Pengumpulan data dilakukan dengan cara wawancara pada bagian marketing yang bertugas dalam pembangunan perumahan. Wawancara ini dilakukan untuk memahami bagaimana proses dalam membuat denah atau pembagian lahan untuk satu unit tipe rumah yang akan dibangun. Selain itu wawancara dilakukan untuk mengetahui rencana anggaran biaya untuk satu unit tipe rumah yang dibangun. Untuk mengetahui harga jual satu unit tipe rumah dilakukan wawancara dengan bagian *marketing* pengembang. Sehingga dapat dibandingkan antara rencana anggaran biaya awal dengan harga jual dari setiap unit tipe rumah tersebut. Data-data pada penelitian ini digunakan untuk menghitung jumlah tiap tipe rumah yang dapat menghasilkan keuntungan pembangungan perumahan yang optimal.

3.3 Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk menganalisis dan mendapatkan semua kebutuhan yang diperlukan dalam pembuatan sistem optimasi keuntungan pembangunan perumahan berdasarkan jumlah unit tipe rumah menggunakan *Particle Swarm Optimization (PSO)*. Analisis kebutuhan disesuaikan dengan kebutuhan hardware dan software serta kebutuhan secara fungsional dan non fungsional. Secara keseluruhan, kebutuhan yang digunakan dalam aplikasi ini meliputi:

1. Data yang dibutuhkan, antara lain:

   - Tipe dari setiap rumah yang akan dibangun.

   - *Range* dari setiap tipe rumah yang akan dibangun

   - Keuntungan setiap tipe rumah yang akan dibangun

2. Kebutuhan fungsional

   Sistem dapat menentukan komposisi terbaik dari jumlah setiap unit tipe rumah yang akan dibangun, serta menampilkan hasil keuntungan penjualan rumah yang terbaik.

3. Kebutuhan non-fungsional

   - Sistem dapat diakses selama 24 jam *(Availability)*.

   - Sistem dapat diimplementasikan menggunakan bahasa pemrograman *Java (Implementation)*.

3.4 Perancangan Sistem

Perancangan sistem dibangun berdasarkan hasil analisis kebutuhan dan pengambilan data yang dilakukan pada optimasi keuntungan pembangunan perumahan berdasarkan jumlah unit tipe rumah menggunakan *Particle Swarm Optimization (PSO)*. Pada perancangan sistem dijelaskan tentang rancangan langkah kerja dari sistem secara menyeluruh untuk mempermudah implementasi, pengujian, dan analisis. Langkah-langkah yang dilakukan dalam perancangan sistem ini disesuaikan dengan perancangan algoritma *Particle Swarm Optimization (PSO)*.

3.5 Pengujian dan Analisis Sistem

Pengujian dan analisis sistem dilakukan untuk dapat mengetahui apakah sistem berjalan sesuai dengan spesifikasi kebutuhan yang telah ditetapkan. Detail pengujian akan dipaparkan pada bab tentang pengujian dan analisis sistem.

3.6 Kesimpulan

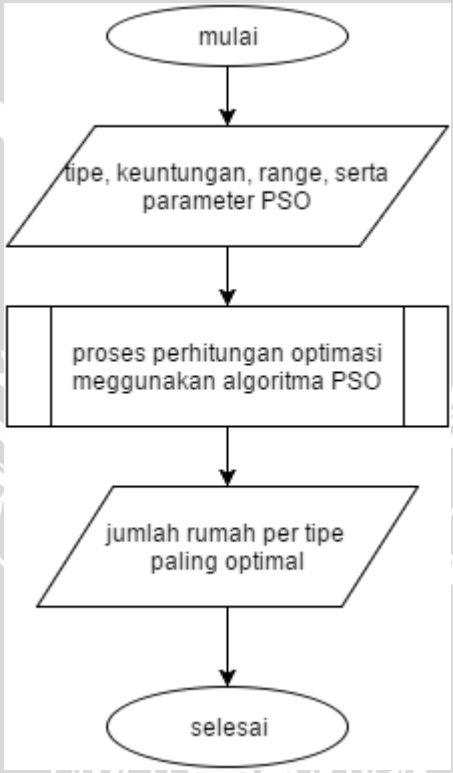
Kesimpulan dilakukan apabila semua tahapan dari perancangan, implementasi, dan pengujian yang diterapkan telah dilakukan. Kesimpulan diambil dari tahap analisis dan pengujian sistem tersebut. Tahap terakhir dari penulisan adalah saran yang berkenaan dengan hasil yang dicapai dan berguna untuk memperbaiki kesalahan untuk pengembangan lebih lanjut.

BAB 4 PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai hal-hal yang berhubungan dengan perancangan sistem, dimulai dari alir perancangan sistem, perancangan antarmuka pengguna, perancangan pengujian, serta analisis sistem.

4.1 Alir Perancangan Sistem

Pada tahap alir perancangan sistem, terdapat penjelasan mengenai bagaimana proses yang terjadi didalam system menggunakan algoritma *particle swarm optimization (PSO)*, yang ditunjukkan oleh Gambar 4.1 berikut ini.

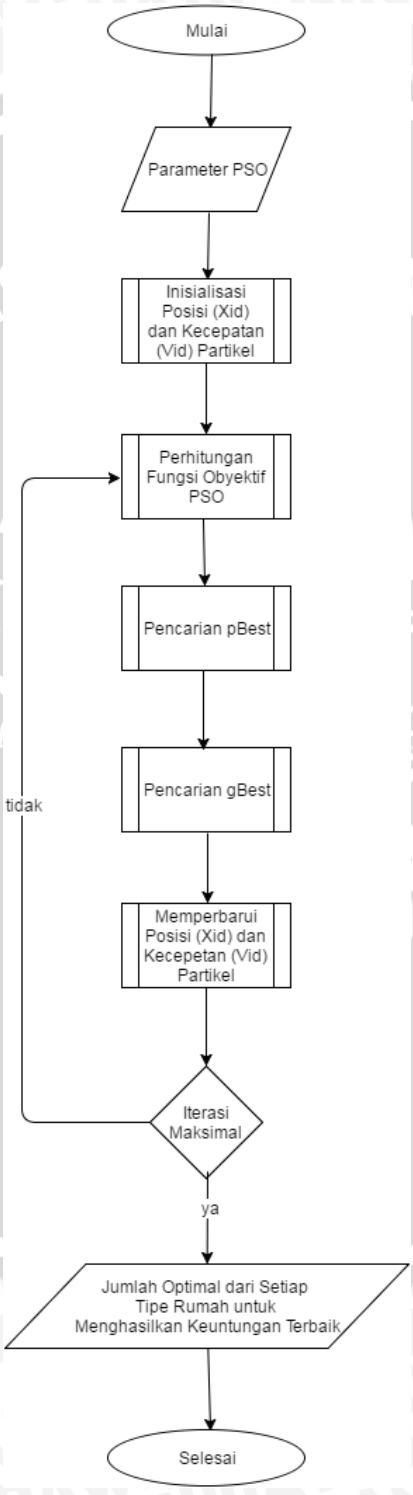


Gambar 4.1 Diagram alir perancangan sistem

Pada Gambar 4.1 ditunjukkan bagaimana alir perancangan sistem untuk optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe menggunakan *Particle Swarm Optimization (PSO)*. Dimulai dari memasukkan tipe rumah yang akan dibangun, kemudian memasukkan keuntungan yang didapatkan dari setiap satu tipe rumah yang akan dijual , memasukkan range dari setiap tipe rumah yang akan dibangun. Selain itu, memasukkan masukan berupa parameter dari PSO tersebut. Kemudian data tersebut akan digunakan untuk proses perhitungan menggunakan algoritma *Particle Swarm Optimization* (PSO) untuk menghasilkan jumlah rumah setiap tipe yang akan digunakan untuk mendapatkan keuntungan maksimal.

4.1.1 Proses Perhitungan Algoritma *Particle Swarm Optimization (PSO)*

Proses perhitungan menggunakan algoritma PSO untuk menghasilkan output terbaik. Proses-proses tersebut dijelaskan dalam diagram alir dibawah ini.

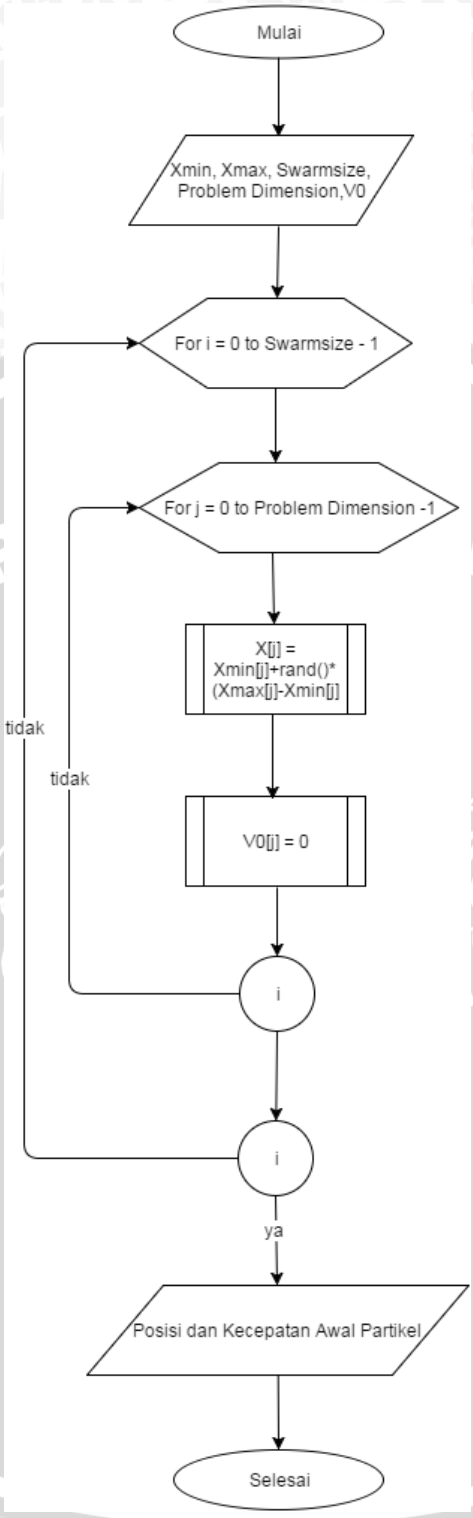


Gambar 4.2 Diagram Alir Algoritma *Particle Swarm Optimization (PSO)*

Pada Gambar 4.2 ditunjukkan bagaimana alir algoritma *Particle Swarm Optimization (PSO)* untuk optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe menggunakan *Particle Swarm Optimization (PSO)*. Dimulai dari memasukkan tipe rumah yang akan dibangun, kemudian memasukkan keuntungan yang didapatkan dari setiap satu tipe rumah yang akan dijual , memasukkan range dari setiap tipe rumah yang akan dibangun. Selain itu, memasukkan masukan berupa parameter dari PSO tersebut. Didalam gambar dijelaskan bahwa setelah proses memasukkan inputan akan dilakukan proses inisialisasi partikel. Dalam algoritma *Particle Swarm Optimization (PSO)* dilakukan pula perhitungan fungsi obyektif. Setelah proses tersebut berjalan dilakukan pencarian Pbest dan Gbest yang dilanjutkan dengan memperbarui posisi Xid dan kecepatan partikel Vid. Kemudian dilakukan pengulangan sampai mencapai iterasi maksimal. Hal ini dilakukan untuk menghasilkan jumlah rumah setiap tipe yang akan digunakan untuk mendapatkan keuntungan maksimal.

4.1.2 Proses Inisialisasi Partikel

Proses pertama dalam PSO adalah inisialisasi partikel. Sebuah partikel didefinisikan oleh dua vektor, x_min dan x_max yang mewakili batas bawah dan batas atas setiap dimensi. Secara garis besar proses dapat dilihat pada Gambar 4.3 berikut.



Gambar 4.3 Proses Inisialisasi Partikel

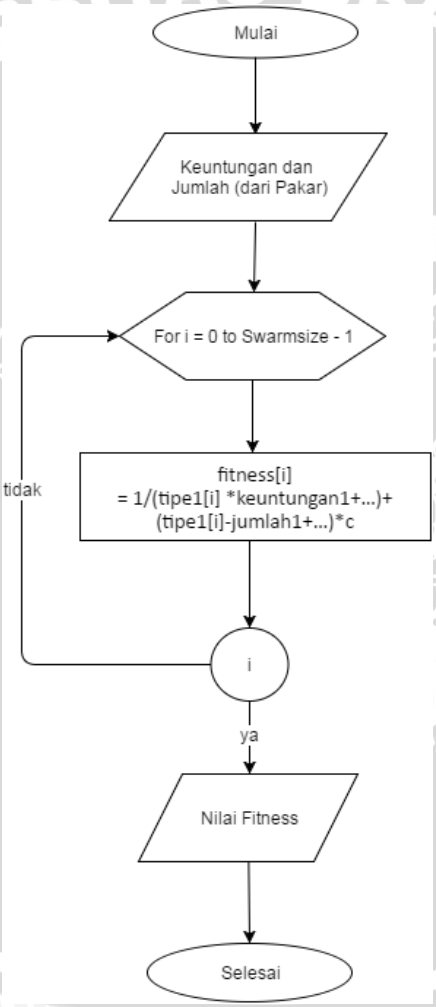
Berdasarkan diagram alir pada Gambar 4.3, proses inisialisasi partikel dapat dijelaskan sebagai berikut:

1. Memberikan perulangan dengan memberi nilai $i = 0$ to *swarmsize.*
2. Memberikan perulangan dengan memberi nilai $j = 0$ to *problem dimension.*

3. Menentukan posisi partikel awal
4. Memberi nilai kecepatan awal $v_0 = 0$
5. Setelah perulangan dilakukan akan terbentuk partikel x yang memiliki posisi dan kecepatan.

4.1.3 Proses Perhitungan Fungsi Obyektif Algoritma *Particle Swarm Optimization (PSO)*

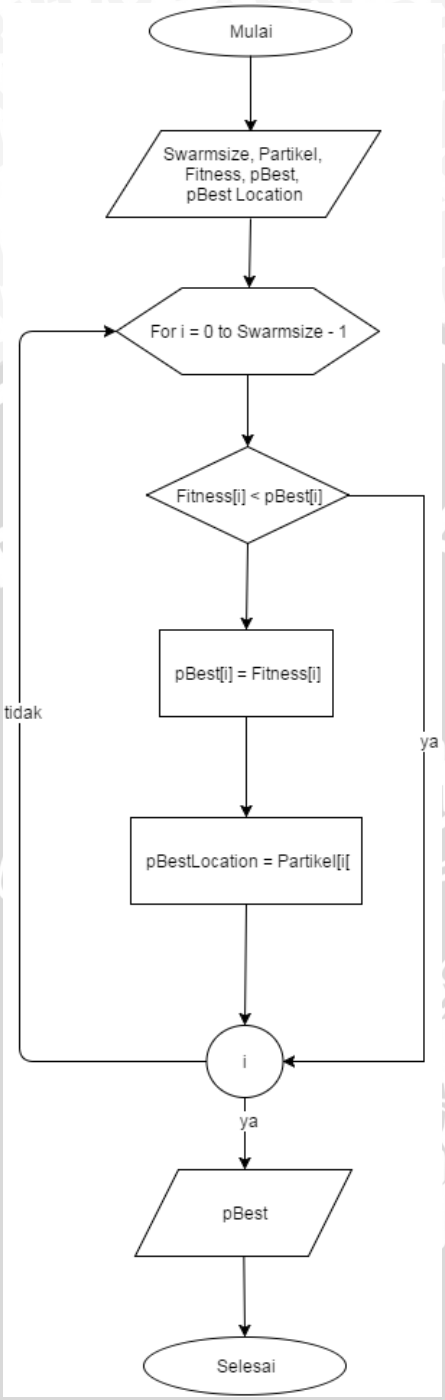
Dalam penelitian ini PSO digunakan untuk medapatkan perhitungan nilai *fitness* yang paling optimal , yaitu hasil maksimal keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipenya. Perhitungan fungsi obyektif menggunakan Persamaan 2.1, namun lebih jelasnya digambarkan melalui diagram alir pada Gambar 4.4 sebagai berikut.



Gambar 4.4 Proses perhitungan fungsi obyektif algoritma PSO

4.1.4 Pencarian *Pbest*

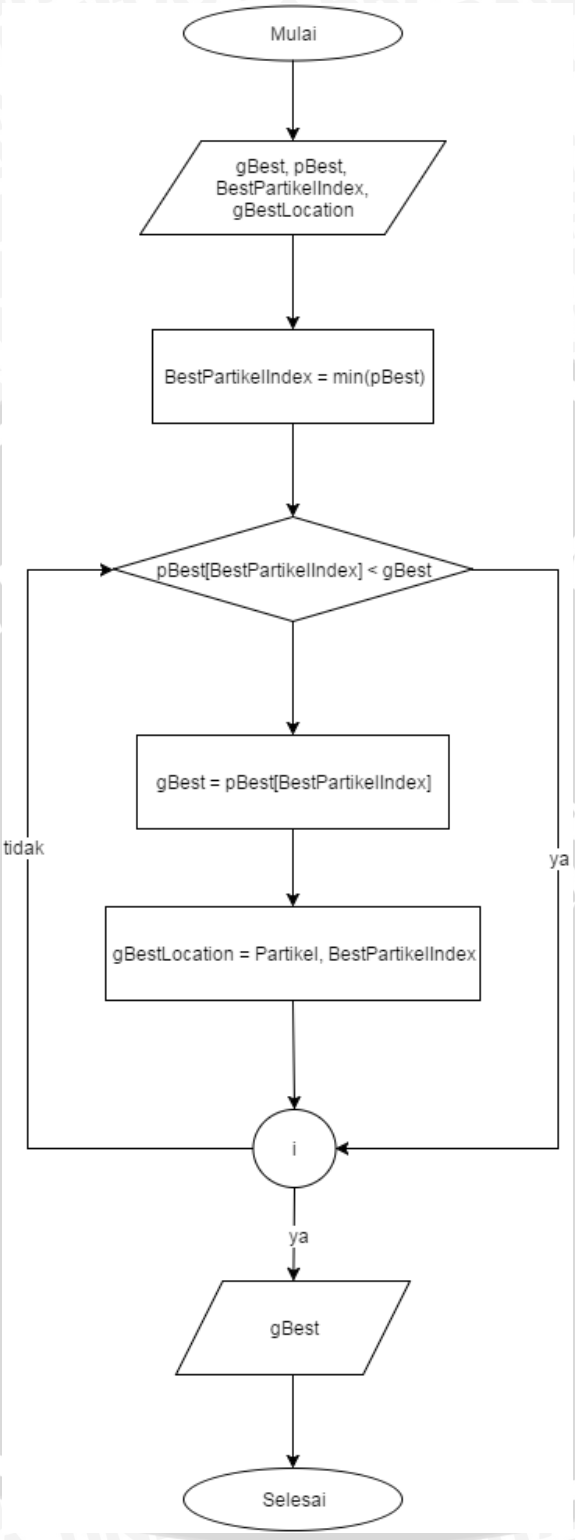
Proses selanjutnya adalah pencarian *pBest*. Sesuai dengan persamaan 2.2, maka secara garis besar proses dapat dilihat pada Gambar 4.5 berikut.



Gambar 4.5 Proses pencarian pBest

4.1.5 Pencarian *Gbest*

Hal yang sama juga dilakukan dalam rangka pencarian *gBest*. Sesuai dengan persamaan 2.3, maka secara garis besar proses dapat dilihat pada Gambar 4.6 berikut.

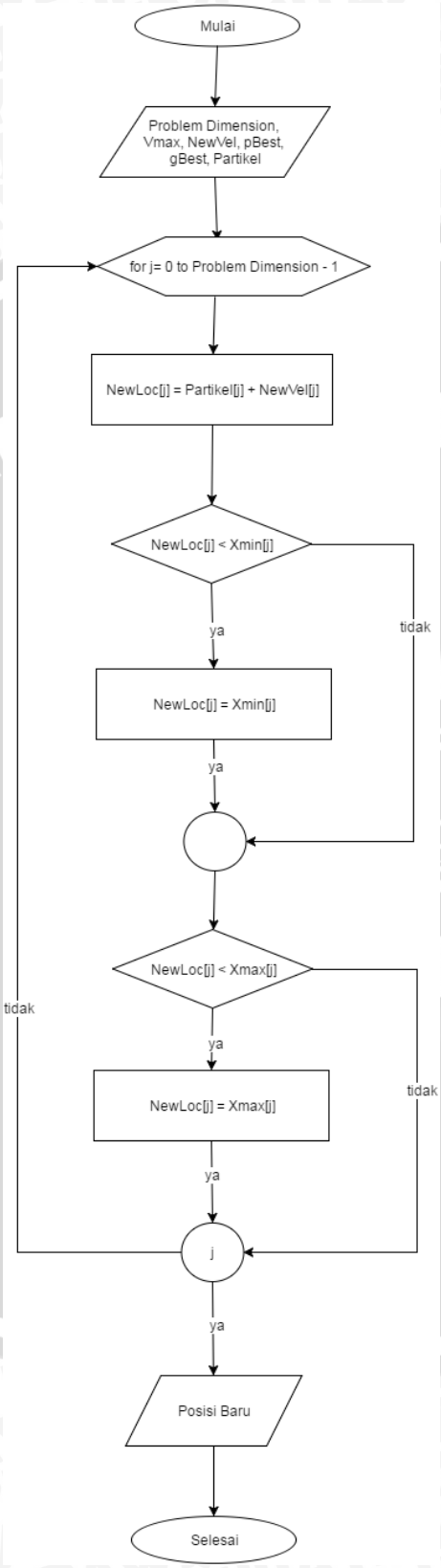


Gambar 4.6 Proses pencarian gBest

4.1.6 Memperbarui Posisi Partikel

Dalam rangka untuk mendapatkan posisi baru partikel, maka diperlukan kecepatan partikel yang baru pada posisi partikel saat ini. Persamaan untuk perubahan posisi (x_{id}) dijelaskan pada Persamaan 2.5, sedangkan diagram alir

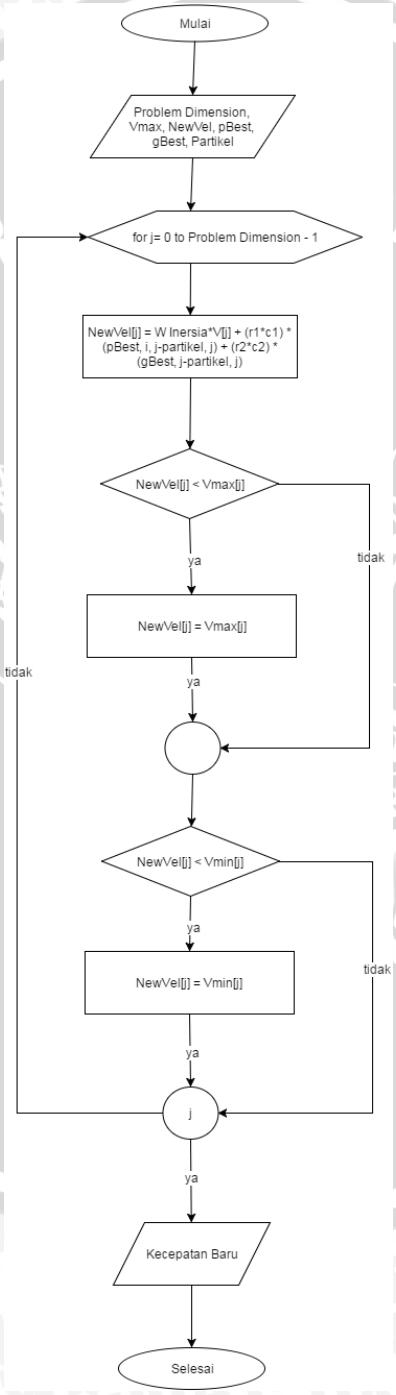
proses memperbarui posisi partikel akan dijelaskan dalam Gambar 4.7 berikut ini.



Gambar 4.7 Proses memperbarui posisi partikel

4.1.7 Memperbarui Kecepatan Partikel

Berdasarkan $pBest$ dan $gBest$ yang diperoleh, maka kecepatan (v_{id}) dan posisi dari particle (x_{id}) diubah. Proses untuk memperbarui kecepatan partikel dengan menggunakan Persamaan 2.4. Dalam persamaan 2.4 terdapat perhitungan awal w inersia yang sesuai dengan persamaan 2.6. Proses untuk memperbarui kecepatan partikel dijelaskan dalam diagram alir pada Gambar 4.8 berikut ini.



Gambar 4.8 Proses memperbarui kecepatan partikel

4.2 Perhitungan Manual Algoritma *Particle Swarm Optimization (PSO)*

Pada contoh perhitungan manual ini, penulis mendapatkan data dari PT. Goldenindo Lestari Housing Developer. Perhitungan manual ini berdasarkan data Cluster Permata Garden Regency dan hasil wawancara dengan Sdr. Lubi Aldiani selaku Marketing. Data yang didapatkan ada pada Tabel 4.1 sebagai berikut.

Tabel 4.1 Keuntungan Penjualan dari Setiap Tipe Rumah

Rumah	Keuntungan
tipe40	Rp. 58.250.000,-
tipe45	Rp. 54.000.000,-
tipe60	Rp. 92.750.000,-
tipe65	Rp. 90.000.000,-
tipe90	Rp. 376.500.000,-
tipe100	Rp. 378.500.000,-
tipe120	Rp. 453.500.000,-

Keuntungan yang didapat berdasarkan harga jual dikurangi harga beli tanah serta harga dalam membangun setiap rumah, yang menurut hasil wawancara adalah harga bangunan rumah per meter persegi adalah Rp. 3.500.000,- serta harga beli tanah awal per meter persegi adalah Rp. 500.000,-.

Tabel 4.2 Jumlah Setiap Tipe Rumah yang Telah Dibangun

Rumah	Jumlah
tipe40	16
tipe45	21
tipe60	1
tipe65	1
tipe90	5
tipe100	5
tipe200	3

Selain data mengenai keuntungan, didapatkan pula jumlah rumah setiap tipe yang telah terbangun berdasarkan siteplan Permata Garden Regency.

4.2.1 Inisialisasi Parameter Algoritma *Particle Swarm Optimization (PSO)*

Hal yang akan dilakukan pertama kali adalah insisalisasi parameter PSO, parameter PSO digunakan dalam proses perhitungan. Parameter PSO diinisialisasikan sebagai berikut.

swarm size : 10
problem dimension : 7
c1 : 2
c2 : 2
Wmin : 0
Wmax : 1

r1 : 0.931431
r2 : 0.313371
V(1,0) : 0
max iteration : 10

4.2.2 Inisialisasi Partikel *Random*

Setelah menentukan parameter PSO, langkah selanjutnya adalah inisialisasi partikel. *Range* merupakan jarak antara batasan maksimal (batas atas) dan batasan minimal (batas bawah) suatu partikel. Pada perhitungan ini digunakan tipe setiap rumah yang akan dibangun, antara lain tipe 40, tipe 45, tipe 60, tipe 65, tipe 90, tipe100, dan tipe 120. Untuk keseluruhan *range* ditunjukkan pada Tabel 4.3 berikut ini.

Tabel 4.3 Range dari Setiap Tipe Rumah yang akan Dibangun

	tipe 40	tipe 45	tipe 60	tipe 65	tipe 90	tipe 100	tipe 200
Min	14	15	1	1	3	4	2
Max	21	23	3	3	9	9	5

Setelah mengetahui *range* dari setiap tipe, maka inisialisasi partikel random dilakukan. Seperti dijelaskan pada Gambar 4.1.2 diatas, maka inisialisasi partikel dilakukan seperti Tabel 4.4 berikut ini.

Tabel 4.4 Proses Inisialisai Partikel

		x=particle							
x(i)	*t*	1	2	3	4	5	6	7	*fitness*
x1	0	17	23	1	1	7	5	2	1.34228E-10
x2	0	16	22	1	1	4	5	5	1.28721E-10
x3	0	20	18	2	2	3	8	4	1.25408E-10
x4	0	21	15	3	3	4	4	5	1.32118E-10
x5	0	16	15	1	3	7	4	4	1.20945E-10
x6	0	21	20	1	1	9	8	4	1.05081E-10
x7	0	21	16	1	3	4	6	5	1.23541E-10
x8	0	15	21	3	3	8	9	4	1.03215E-10
x9	0	19	23	2	3	9	7	4	1.09215E-10
x10	0	14	21	2	3	7	9	5	1.0189E-10

4.2.3 Pencarian *Pbest*

Nilai kecepatan awal partikel untuk setiap dimensi adalah 0 dan posisi terbaik individu ke-i $(pBest_i)$ awal untuk setiap partikel pada waktu $t = 0$ adalah posisi partikel itu sendiri untuk setiap *problem dimension*. Sesuai dengan persamaan 2.2, dan garis besar proses yang dapat dilihat pada Gambar 4.5, maka nilai awal pbest ditunjukkan pada Tabel 4.5 berikut ini.

Tabel 4.5 Proses Memperbarui Kecepatan Partikel

x(i)	t	Pbest							fitness
		1	2	3	4	5	6	7	
x1	0	17	23	1	1	7	5	2	1.34228E-10
x2	0	16	22	1	1	4	5	5	1.28721E-10
x3	0	20	18	2	2	3	8	4	1.25408E-10
x4	0	21	15	3	3	4	4	5	1.32118E-10
x5	0	16	15	1	3	7	4	4	1.20945E-10
x6	0	21	20	1	1	9	8	4	1.05081E-10
x7	0	21	16	1	3	4	6	5	1.23541E-10
x8	0	15	21	3	3	8	9	4	1.03215E-10
x9	0	19	23	2	3	9	7	4	1.09215E-10
x10	0	14	21	2	3	7	9	5	1.0189E-10

4.2.4 Pencarian *Gbest*

Selain pencarian pBest, dilakukan pula pencarian gBest yang sesuai dengan persamaan 2.3, dan garis besar proses yang dapat dilihat pada gambar 4.6. Persamaan 2.3 untuk memperoleh nilai adalah $gBest(0) = argMin_{i=1}^{n}\{f(pBest_i(0))\}$ yang perhitungannya ditunjukkan pada Tabel 4.6 berikut ini.

Tabel 4.6 Proses Pencarian Gbest

x(i)	t	Gbest							fitness
		1	2	3	4	5	6	7	
x10	0	14	21	2	3	7	9	5	1.0189E-10

4.2.5 Memperbarui Posisi dan Kecepatan Partikel

Kecepatan partikel baru diperlukan dalam menentukan posisi partikel, sesuai dengan persamaan 2.4 dan diagram alir pada Gambar 4.8, maka hasil perhitungan manualisasi kecepatan partikel baru pada Tabel 4.7 berikut ini.

Tabel 4.7 Proses Memperbarui Kecepatan Partikel

x(i)	T	Vbaru						
		1	2	3	4	5	6	7
x1	1	-0.19	-0.1	0.06	0.13	0	0.26	0.19
x2	1	-0.13	-0.1	0.06	0.13	0.2	0.26	0
x3	1	-0.39	0.19	0	0.06	0.3	0.06	0.06
x4	1	-0.45	0.39	-0.1	0	0.2	0.32	0
x5	1	-0.13	0.39	0.06	0	0	0.32	0.06
x6	1	-0.45	0.06	0.06	0.13	-0.1	0.06	0.06
x7	1	-0.45	0.32	0.06	0	0.2	0.19	0
x8	1	-0.06	0	-0.1	0	-0.1	0	0.06
x9	1	-0.32	-0.1	0	0	-0.1	0.13	0.06

x(i)	T	Vbaru						
		1	2	3	4	5	6	7
x10	1	0	0	0	0	0	0	0

Dalam rangka menetukan posisi partikel baru yang sesuai dengan persamaan 2.5 dan diagram alir pada Gambar 4.7, maka hasil dari posisi baru adalah pada Tabel 4.8 berikut ini.

Tabel 4.8 Proses memperbarui posisi partikel

x(i)	t	pbaru							fitness
		1	2	3	4	5	6	7	
x1	1	16.81	22.9	1.06	1.13	7	5.26	2.19	1.31525E-10
x2	1	15.87	21.9	1.06	1.13	4.2	5.26	5	1.26563E-10
x3	1	19.61	18.2	2	2.06	3.3	8.06	4.06	1.23561E-10
x4	1	20.55	15.4	2.94	3	4.2	4.32	5	1.29627E-10
x5	1	15.87	15.4	1.06	3	7	4.32	4.06	1.19497E-10
x6	1	20.55	20.1	1.06	1.13	8.9	8.06	4.06	1.04868E-10
x7	1	20.55	16.3	1.06	3	4.2	6.19	5	1.21864E-10
x8	1	14.94	21	2.94	3	7.9	9	4.06	1.03128E-10
x9	1	18.68	22.9	2	3	8.9	7.13	4.06	1.08709E-10
x10	1	14	21	2	3	7	9	5	1.0189E-10

4.2.6 Hasil Perhitungan Manual Menggunakan Algoritma *Particle Swarm Optimization (PSO)*

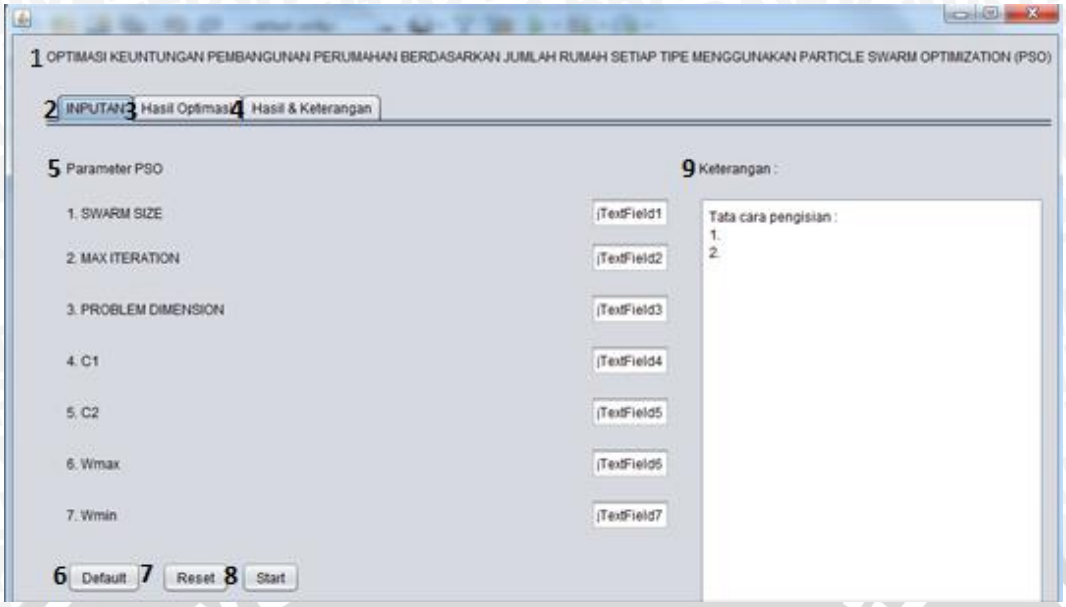
Dari perhitungan yang telah dilakukan menggunakan proses perhitungan berdasarkan algoritma PSO, didapatkan hasil sebagai berikut pada tabel 4.9. Hasil tersebut kemudian akan dikalikan dengan keutungan yang didapatkan per tipe rumah yang akan dibangun.

Tabel 4.9 Hasil optimasi mengunakan algoritma PSO

x(i)	T	gbest							fitness
		1	2	3	4	5	6	7	
x10	1	14	21	2	3	7	9	5	1.0189E-10

4.3 Perancangan Antar Muka atau *User Interface(UI)*

Antarmuka sistem ini terdiri dari halaman utama sistem yang berisi masukan masukan parameter algoritma PSO dan hasil optimasi oleh sistem. Rancangan antar muka ditunjukkan pada gambar 4.9 sebagai berikut:



Gambar 4.9 Rancangan antarmuka

Rancangan tampilan sistem adalah tampilan yang berisi formulir untuk memasukkan nilai-nilai parameter algoritma yang akan diolah untuk menampilkan hasil optimasi berupa jumlah rumah setiap tipenya. Rancangan tampilan sistem dijelaskan sebagai berikut:

1. Judul sistem
2. *Tabbutton* untuk berpindah ke halaman masukan
3. *Tabbutton* untuk berpindah ke halaman hasil optimasi yang berisi hasil perhitungan di setiap iterasi berdasarkan masukan parameter algoritma PSO
4. *Tabbutton* untuk berpindah ke halaman hasil dan keterangan yang menampilkan hasil terbaik dan perhitungan keuntungannya
5. *Textbox* untuk memasukkan nilai-nilai parameter algoritma PSO
6. *Button* "Default" untuk memproses masukan parameter algoritma yang nantinya akan menampilkan hasil optimasi keuntungan jumlah rumah setiap tipenya
7. *Button* "Reset" untuk memproses masukan parameter algoritma yang nantinya akan menampilkan hasil optimasi keuntungan jumlah rumah setiap tipenya
8. *Button* "Start" untuk memproses masukan parameter algoritma yang nantinya akan menampilkan hasil optimasi keuntungan jumlah rumah setiap tipenya
9. *Text* yang menampilkan tata cara pengisian nilai-nilai parameter PSO

4.4 Perancangan Pengujian dan Analisis Sistem

Tidak adanya metode tertentu untuk menentukan parameter algoritma PSO membuat perlu dilakukannya evaluasi dengan beberapa pengujian agar mendapatkan parameter yang optimal. Pengujian tersebut antara lain:

1. Pengujian untuk menentukan jumlah partikel (*Swarmsize*) yang optimal.
2. Pengujian untuk menentukan jumlah iterasi yang optimal.

4.4.1 Pengujian Jumlah Partikel

Uji coba terhadap banyaknya jumlah partikel (swarmsize) untuk mengetahui jumlah partikel yang optimal dan dapat menghasilkan solusi yang lebih baik. Pengujian ini menggunakan beberapa ukuran swarmsize yang berbeda dengan kelipatan 2. Rancangan pengujian jumlah partikel (swarmsize) dapat dilihat pada Tabel 4.10 berikut.

Tabel 4.10 Pengujian Jumlah Partikel

Jumlah partikel (*Swarmsize)*	Nilai *cost* pada percobaan ke-					Rata-rata *fitness*
	1	2	3	...	10	
5						
7						
9						
11						
13						
15						

4.4.2 Pengujian Jumlah Iterasi

Pengujian terhadap banyaknya jumlah maksimum iterasi untuk mengetahui maksimum iterasi yang optimal dan dapat menghasilkan solusi yang lebih baik. Pengujian dilakukan sebanyak 5 kali untuk masing-masing iterasi. Pengujian menggunakan ukuran yang berbeda dengan kelipatan 20. Kemudian masing-masing data akan dicatat untuk mendapatkan rata-rata nilai cost terbaik. Rancangan pengujian jumlah iterasi dapat dilihat pada Tabel 4.11 berikut.

Tabel 4.11 Pengujian Jumlah Iterasi

Iterasi	Nilai *cost* pada percobaan ke-					Rata-rata *fitness*
	1	2	3	4	5	
20						
35						
50						
65						
80						

4.4.3 Pengujian Waktu Komputasi

Pengujian terhadap waktu komputasi pada setiap jumlah maksimum iterasi untuk mengetahui waktu yang dibutuhkan dalam memproses data yang ada. Pengujian dilakukan sebanyak 5 kali, sesuai dengan jumlah setiap kelipatan 20. Pengujian menggunakan ukuran yang berbeda dengan kelipatan 5. Kemudian masing-masing data akan dicatat untuk mendapatkan waktu komputasi. Rancangan pengujian waktu komputasi dapat dilihat pada Tabel 4.12 berikut.

Tabel 4.12 Pengujian Waktu Komputasi

iterasi	Waktu komputasi
20	
35	
50	
65	
80	

BAB 5 IMPLEMENTASI SISTEM

Bab ini membahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah diperoleh dari analisis kebutuhan dan proses perancangan perangkat lunak yang telah dibuat. Pembahasan pada bab ini terdiri dari implementasi program dan implementasi antarmuka sistem "OPTIMASI KEUNTUNGAN PEMBANGUNAN PERUMAHAN BERDASARKAN JUMLAH RUMAH SETIAP TIPE MENGGUNAKAN *PARTICLE SWARM OPTIMIZATION (PSO)*"

5.1 Implementasi program

Berdasarkan perancangan yang telah dibahas pada bab 4, maka akan dibuat implementasi program sesuai dengan perancangan tersebut. Program menggunakan bahasa pemrograman *Java* berbasis *desktop*.

5.1.1 Implementasi Proses PSO

Proses ini merupakan inti dari perhitungan yang ada. Proses optimasi ini menggunakan *Particle Swarm Optimization* (PSO). Proses ini sesuai dengan diagram alir pada sub bab 4.1.1. Proses diawali dengan proses inisialisasi partikel, dilanjutkan dengan proses *random location* berdasarkan *range* yang ditentukan, *update fitness*, kemudian proses pencarian *fitness* paling kecil, *update pBest* serta *update gBest*, proses *update velocity*, proses *update location*, sampai pada akhirnya telah mencapai iterasi maksmimal, dan yang terakhir menampilkan keluaran yang akan dijadikan tabel di tabel hasil optimasi pada implementasi antar muka. Keseluruhan proses tersebut ditunjukkan oleh Kode Program 5.1 dibawah ini.

Kode Program 5.1 Implementasi Proses PSO

```
1  package PsoSkripsi;
2
3  import java.math.BigDecimal;
4  import java.util.Random;
5  import java.util.Vector;
6  import javax.swing.table.DefaultTableModel;
7
8  public class hewekHewek extends javax.swing.JFrame {
9
10     int SWARM_SIZE;
11     int MAX_ITERATION;
12     int PROBLEM_DIMENSION;
13     double C1;
14     double C2;
15     double W_UPPERBOUND;
16     double W_LOWERBOUND;
17
18     private Vector<Particle> swarm;
19     private double[] pBest;
20     private Vector<Location> pBestLocation;
21     private double gBest;
```

```
22      private Location gBestLocation;
23      private double[] fitnessValueList;
24
25      DefaultTableModel optimasi, hasil;
26
27      Random rand = new Random();
28
29      void setParam() {
            SWARM_SIZE =
30  Integer.valueOf(fieldSwarmSize.getText());
            MAX_ITERATION =
31  Integer.valueOf(fieldMax.getText());
            PROBLEM_DIMENSION =
32  Integer.valueOf(fieldProblem.getText());
33          C1 = Double.valueOf(fieldC1.getText());
34          C2 = Double.valueOf(fieldC2.getText());;
35          W_UPPERBOUND = Double.valueOf(fieldWUp.getText());;
            W_LOWERBOUND =
36  Double.valueOf(fieldWLow.getText());;
37
38          swarm = new Vector<Particle>();
39          pBest = new double[SWARM_SIZE];
40          pBestLocation = new Vector<Location>();
41          fitnessValueList = new double[SWARM_SIZE];
42      }
43
44      public void initializeSwarm() {
45          Particle p;
46          for (int i = 0; i < SWARM_SIZE; i++) {
47              p = new Particle();
48
49              // inisialisasi partikel random
50              double[] loc = new double[PROBLEM_DIMENSION];
51              for (int a = 0; a < PROBLEM_DIMENSION; a++) {
                    loc[a] = ProblemSet.range[0][a] +
52  rand.nextDouble() * (ProblemSet.range[1][a]
53                          - ProblemSet.range[0][a]);
54              }
55
56              Location location = new Location(loc);
57
                // random velocity berdasarkan range di Problem
58  Set
59              double[] vel = new double[PROBLEM_DIMENSION];
60              //diubah menggunakan for
61              for (int j = 0; j < PROBLEM_DIMENSION; j++) {
62                  vel[j] = 0;
63              }
64              Velocity velocity = new Velocity(vel);
65
66              p.setLocation(location);
67              p.setVelocity(velocity);
68              swarm.add(p);
69          }
```

```
70      }
71
72      public void updateFitnessList() {
73          for (int i = 0; i < SWARM_SIZE; i++) {
                fitnessValueList[i] =
74  swarm.get(i).getFitnessValue();
75          }
76      }
77
78      public static int getMinFitness(double[] list) {
79          int pos = 0;
80          double minValue = list[0];
81          for (int i = 0; i < list.length; i++) {
82              if (list[i] < minValue) {
83                  pos = i;
84                  minValue = list[i];
85              }
86          }
87          return pos;
88      }
89
90      public hewekHewek() {
91          initComponents();
92      }
93
94      @SuppressWarnings("unchecked")
        // <editor-fold defaultstate="collapsed"
95  desc="Generated Code">
96      private void initComponents() {
97
98          jLabel1 = new javax.swing.JLabel();
99          jTabbedPane1 = new javax.swing.JTabbedPane();
100         jPanel1 = new javax.swing.JPanel();
101         jScrollPane4 = new javax.swing.JScrollPane();
102         jTextArea2 = new javax.swing.JTextArea();
103         jLabel2 = new javax.swing.JLabel();
104         jLabel3 = new javax.swing.JLabel();
105         jLabel4 = new javax.swing.JLabel();
106         jLabel5 = new javax.swing.JLabel();
107         jLabel6 = new javax.swing.JLabel();
108         jLabel7 = new javax.swing.JLabel();
109         jLabel8 = new javax.swing.JLabel();
110         jLabel9 = new javax.swing.JLabel();
111         jLabel10 = new javax.swing.JLabel();
112         fieldSwarmSize = new javax.swing.JTextField();
113         fieldMax = new javax.swing.JTextField();
114         fieldProblem = new javax.swing.JTextField();
115         fieldC1 = new javax.swing.JTextField();
116         fieldC2 = new javax.swing.JTextField();
117         fieldWUp = new javax.swing.JTextField();
118         fieldWLow = new javax.swing.JTextField();
119         buttonDefault = new javax.swing.JButton();
120         buttonReset = new javax.swing.JButton();
```

```
        buttonStart = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        tableOptimasi = new javax.swing.JTable();
        jPanel2 = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        tableHasil = new javax.swing.JTable();
        jScrollPane3 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_O
N_CLOSE);

        jLabel1.setText("OPTIMASI KEUNTUNGAN PEMBANGUNAN
PERUMAHAN BERDASARKAN JUMLAH RUMAH SETIAP TIPE MENGGUNAKAN
PARTICLE SWARM OPTIMIZATION (PSO)");

        jTextArea2.setEditable(false);
        jTextArea2.setColumns(20);
        jTextArea2.setRows(5);
        jTextArea2.setText("Tata cara pengisian :\n1. Swarm
size(jumlah partikel) diisi\n    bilangan bulat dengan
range 10\n    hingga 30 \n2. Max iteration diisi dengan
bilangan\n    bulat dengan range 50 hingga 150\n3. Nilai C1
dan C2 diisi dengan\n    bilangan bulat atau desimal
dengan\n    range 2 hingga 3\n4. Nilai WMax dan WMin diisi
dengan\n    bilangan desimal dengan range 0\n    hingga
1\n\n");
        jScrollPane4.setViewportView(jTextArea2);

        jLabel2.setText("Keterangan :");

        jLabel3.setText("Parameter PSO");

        jLabel4.setText("1. SWARM SIZE");

        jLabel5.setText("2. MAX ITERATION");

        jLabel6.setText("3. PROBLEM DIMENSION");

        jLabel7.setText("4. C1");

        jLabel8.setText("5. C2");

        jLabel9.setText("6. Wmax");

        jLabel10.setText("7. Wmin");

        fieldSwarmSize.setColumns(10);

        fieldMax.setColumns(10);

```

```
164         fieldProblem.setEditable(false);
165         fieldProblem.setColumns(10);
166         fieldProblem.setText("7");
167 
168         fieldC1.setColumns(10);
169 
170         fieldC2.setColumns(10);
171 
172         fieldWUp.setColumns(10);
173 
174         fieldWLow.setColumns(10);
175 
176         buttonDefault.setText("Default");
            buttonDefault.addActionListener(new
177 java.awt.event.ActionListener() {
                public void
178 actionPerformed(java.awt.event.ActionEvent evt) {
179                 buttonDefaultActionPerformed(evt);
180             }
181         });
182 
183         buttonReset.setText("Reset");
            buttonReset.addActionListener(new
184 java.awt.event.ActionListener() {
                public void
185 actionPerformed(java.awt.event.ActionEvent evt) {
186                 buttonResetActionPerformed(evt);
187             }
188         });
189 
190         buttonStart.setText("Start");
            buttonStart.addActionListener(new
191 java.awt.event.ActionListener() {
                public void
192 actionPerformed(java.awt.event.ActionEvent evt) {
193                 buttonStartActionPerformed(evt);
194             }
195         });
196 
            javax.swing.GroupLayout jPanel1Layout = new
197 javax.swing.GroupLayout(jPanel1);
198         jPanel1.setLayout(jPanel1Layout);
199         jPanel1Layout.setHorizontalGroup(

    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.A
200 lignment.LEADING)

    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
201 jPanel1Layout.createSequentialGroup()
202                 .addGap(18, 18, 18)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
203 upLayout.Alignment.LEADING)

204 .addGroup(jPanel1Layout.createSequentialGroup()
205 
```

```
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
    upLayout.Alignment.LEADING)
                                .addComponent(jLabel4,
    javax.swing.GroupLayout.DEFAULT_SIZE,
206 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(jLabel5,
    javax.swing.GroupLayout.DEFAULT_SIZE, 259,
207 Short.MAX_VALUE))
208                         .addGap(48, 48, 48)
                            .addComponent(fieldMax,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
209 javax.swing.GroupLayout.PREFERRED_SIZE))

    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
210 jPanel1Layout.createSequentialGroup()
211                         .addGap(0, 0, Short.MAX_VALUE)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
212 upLayout.Alignment.LEADING)
                                .addComponent(fieldSwarmSize,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
213 javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(fieldProblem,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
214 javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(fieldC1,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
215 javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(fieldC2,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
216 javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(fieldWUp,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
217 javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(fieldWLow,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
218 javax.swing.GroupLayout.PREFERRED_SIZE)))

219 .addGroup(jPanel1Layout.createSequentialGroup()

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
220 upLayout.Alignment.LEADING)
221                         .addComponent(jLabel3)


222 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
```

```
      upLayout.Alignment.TRAILING, false)
                                    .addComponent(jLabel10,
      javax.swing.GroupLayout.Alignment.LEADING,
      javax.swing.GroupLayout.DEFAULT_SIZE,
223   javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                    .addComponent(jLabel9,
      javax.swing.GroupLayout.Alignment.LEADING,
224   javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                                    .addComponent(jLabel8,
      javax.swing.GroupLayout.Alignment.LEADING,
      javax.swing.GroupLayout.DEFAULT_SIZE,
225   javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                    .addComponent(jLabel7,
      javax.swing.GroupLayout.Alignment.LEADING,
      javax.swing.GroupLayout.DEFAULT_SIZE,
226   javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
227                                 .addComponent(jLabel6)

228   .addGroup(jPanel1Layout.createSequentialGroup()

229   .addComponent(buttonDefault)

      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
230   .RELATED)
231                                 .addComponent(buttonStart)

      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
232   .RELATED)

233   .addComponent(buttonReset)))
234                       .addGap(0, 0, Short.MAX_VALUE)))
235                 .addGap(26, 26, 26)

      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
236   upLayout.Alignment.LEADING)
237                     .addComponent(jLabel2)
                        .addComponent(jScrollPane4,
      javax.swing.GroupLayout.PREFERRED_SIZE, 309,
238   javax.swing.GroupLayout.PREFERRED_SIZE))
239                 .addContainerGap())
240         );
241         jPanel1Layout.setVerticalGroup(

      jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.A
242   lignment.LEADING)

      .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
243   jPanel1Layout.createSequentialGroup()
244                 .addContainerGap(34, Short.MAX_VALUE)

      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
245   upLayout.Alignment.BASELINE)
246                     .addComponent(jLabel2)
247                     .addComponent(jLabel3))
248                 .addGap(18, 18, 18)

      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
249   upLayout.Alignment.LEADING, false)
```

```
250 .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jScrollPane4,
    javax.swing.GroupLayout.PREFERRED_SIZE, 382,
251 javax.swing.GroupLayout.PREFERRED_SIZE)
252                     .addContainerGap())

253 .addGroup(jPanel1Layout.createSequentialGroup()

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
254 upLayout.Alignment.BASELINE)
255                         .addComponent(jLabel4)
                            .addComponent(fieldSwarmSize,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
256 javax.swing.GroupLayout.PREFERRED_SIZE))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
257 .UNRELATED)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
258 upLayout.Alignment.BASELINE)
259                         .addComponent(jLabel5)
                            .addComponent(fieldMax,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
260 javax.swing.GroupLayout.PREFERRED_SIZE))
261                     .addGap(18, 18, 18)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
262 upLayout.Alignment.BASELINE)
263                         .addComponent(jLabel6)
                            .addComponent(fieldProblem,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
264 javax.swing.GroupLayout.PREFERRED_SIZE))
265                     .addGap(18, 18, 18)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
266 upLayout.Alignment.BASELINE)
267                         .addComponent(jLabel7)
                            .addComponent(fieldC1,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
268 javax.swing.GroupLayout.PREFERRED_SIZE))
269                     .addGap(18, 18, 18)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
270 upLayout.Alignment.BASELINE)
271                         .addComponent(jLabel8)
                            .addComponent(fieldC2,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
272 javax.swing.GroupLayout.PREFERRED_SIZE))
273                     .addGap(18, 18, 18)

274 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
```

```
    upLayout.Alignment.BASELINE)
275                             .addComponent(jLabel9)
                                .addComponent(fieldWUp,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
276 javax.swing.GroupLayout.PREFERRED_SIZE))
277                         .addGap(18, 18, 18)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
278 upLayout.Alignment.BASELINE)
279                             .addComponent(jLabel10)
                                .addComponent(fieldWLow,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
280 javax.swing.GroupLayout.PREFERRED_SIZE))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
    .RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
281 Short.MAX_VALUE)

    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gro
282 upLayout.Alignment.BASELINE)
283                             .addComponent(buttonDefault)
284                             .addComponent(buttonStart)
285                             .addComponent(buttonReset))
286                         .addGap(34, 34, 34))))
287         );
288
289         jTabbedPane1.addTab("INPUTAN", jPanel1);
290
            tableOptimasi.setModel(new
291 javax.swing.table.DefaultTableModel(
292             new Object [][] {
293
294             },
295             new String [] {
                    "Iterasi", "Swarm ke-", "Tipe 40", "Tipe
    45", "Tipe 60", "Tipe 65", "Tipe 90", "Tipe 100", "Tipe
296 120", "Fitness"
297             }
298         ));
299         jScrollPane1.setViewportView(tableOptimasi);
            if (tableOptimasi.getColumnModel().getColumnCount()
300 > 0) {

    tableOptimasi.getColumnModel().getColumn(0).setPreferredWid
301 th(50);

    tableOptimasi.getColumnModel().getColumn(9).setPreferredWid
302 th(100);
303         }
304
            jTabbedPane1.addTab("Hasil Optimasi",
305 jScrollPane1);
306
            tableHasil.setModel(new
307 javax.swing.table.DefaultTableModel(
```

```
            new Object [][] {

            },
            new String [] {
                "Tipe 40", "Tipe 45", "Tipe 60", "Tipe 65",
"Tipe 90", "Tipe 100", "Tipe 120"
            }
        ));
        jScrollPane2.setViewportView(tableHasil);

        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane3.setViewportView(jTextArea1);

        jLabel11.setText("Hasil");

        jLabel12.setText("Keterangan");

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
                    .addComponent(jScrollPane3,
javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 736, Short.MAX_VALUE)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
                            .addComponent(jLabel11)
                            .addComponent(jLabel12))
                        .addGap(0, 0, Short.MAX_VALUE)))
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap(13, Short.MAX_VALUE)
                .addComponent(jLabel11)
                .addGap(18, 18, 18)
                .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 51,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED)
                .addComponent(jLabel12)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
                .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 321,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        jTabbedPane1.addTab("Hasil & Keterangan", jPanel2);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                    .addComponent(jTabbedPane1)
                    .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 27,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jTabbedPane1)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void
buttonStartActionPerformed(java.awt.event.ActionEvent evt)
{
        optimasi = (DefaultTableModel)
```

```
    tableOptimasi.getModel();
383         hasil = (DefaultTableModel) tableHasil.getModel();
384
385         setParam();
386
387         initializeSwarm();
388         updateFitnessList();
389
390         //pbest awal
391         for (int i = 0; i < SWARM_SIZE; i++) {
392             pBest[i] = fitnessValueList[i];
393             pBestLocation.add(swarm.get(i).getLocation());
394         }
395
396         int t = 0;
397         double w;
398
399         while (t < MAX_ITERATION) {
400
401             // step 2 - update gBest
                int bestParticleIndex =
402 getMinFitness(fitnessValueList);
                if (t == 0 ||
403 fitnessValueList[bestParticleIndex] < gBest) {
                    gBest =
404 fitnessValueList[bestParticleIndex];
                    gBestLocation =
405 swarm.get(bestParticleIndex).getLocation();
406             }
407             //diubah sesuai manualisasi
                w = W_UPPERBOUND - ((W_UPPERBOUND -
408 W_LOWERBOUND) / MAX_ITERATION) * ((double) t);
409
410             for (int i = 0; i < SWARM_SIZE; i++) {
411                 double r1 = rand.nextDouble()/10;
412                 double r2 = rand.nextDouble()/10;
413
414                 Particle p = swarm.get(i);
415
416                 // step 3 - update velocity
                    double[] newVel = new
417 double[PROBLEM_DIMENSION];
                    double[] Vmax = new
418 double[PROBLEM_DIMENSION];
                    double[] Vmin = new
419 double[PROBLEM_DIMENSION];
420
                    for (int j = 0; j < PROBLEM_DIMENSION; j++)
421 {
                        Vmax[j] = 0.6 *
422 ((ProblemSet.range[1][j]-ProblemSet.range[0][j])/2);
423                     Vmin[j] = -Vmax[j];
                        newVel[j] = (w *
424 p.getVelocity().getPos()[j])
                            + (r1 * C1) *
425 (pBestLocation.get(i).getLoc()[j] -
```

```
    p.getLocation().getLoc()[j])
                                + (r2 * C2) *
426 (gBestLocation.getLoc()[j] - p.getLocation().getLoc()[j]);
427                     if (newVel[j] < Vmin[j]) {
428                         newVel[j] = Vmin[j];
429                     } else if (newVel[j] > Vmax[j]) {
430                         newVel[j] = Vmax[j];
431                     } else {
432                         newVel[j] = newVel[j];
433                     }
434                     System.out.println(newVel[j]);
435                     System.out.println("");
436                 }
437                 Velocity vel = new Velocity(newVel);
438                 p.setVelocity(vel);
439
440                 // step 4 - update location
                    double[] newLoc = new
441 double[PROBLEM_DIMENSION];
                    for (int j = 0; j < PROBLEM_DIMENSION; j++)
442 {
                        newLoc[j] = p.getLocation().getLoc()[j]
443 + newVel[j];
                        if (newLoc[j] < ProblemSet.range[0][j])
444 {
445                         newLoc[j] = ProblemSet.range[0][j];
                        } else if (newLoc[j] >
446 ProblemSet.range[1][j]) {
447                         newLoc[j] = ProblemSet.range[1][j];
448                     } else {
449                         newLoc[j] = newLoc[j];
450                     }
451                 }
452                 Location loc = new Location(newLoc);
453                 p.setLocation(loc);
454
455             }
456             updateFitnessList();
457
458             //random injection
459             if (t % 2 == 0) {
                    int jumInjec = (int) Math.round(0.6 *
460 SWARM_SIZE);
461                 for (int q = 0; q < SWARM_SIZE; q++) {
                        double[] newPosInjec = new
462 double[PROBLEM_DIMENSION];
                        double[] newVelInjec = new
463 double[PROBLEM_DIMENSION];
464                     Particle partikel = swarm.get(q);
465                     for (int j = 0; j < jumInjec; j++) {
                            if (partikel.getFitnessValue() ==
466 fitnessValueList[j]) {
                                for (int k = 0; k <
467 PROBLEM_DIMENSION; k++) {
                                    newPosInjec[k] =
468 ProblemSet.range[0][k] + rand.nextDouble() *
```

```
    (ProblemSet.range[1][k] - ProblemSet.range[0][k]);
469                                 newVelInjec[k] = 0;
                                    Location lokasi = new
470 Location(newPosInjec);

471 partikel.setLocation(lokasi);
                                    Velocity kec = new
472 Velocity(newVelInjec);
473                                 partikel.setVelocity(kec);
474                             }
475                         }
476                     }
477                 }
478                 updateFitnessList();
479             }
480
481             //update pBest
482             for (int i = 0; i < SWARM_SIZE; i++) {
483                 if (fitnessValueList[i] <= pBest[i]) {
484                     pBest[i] = fitnessValueList[i];
                        pBestLocation.set(i,
485 swarm.get(i).getLocation());
486                 }
487             }
488             //update gBest
                if (t == 0 ||
489 fitnessValueList[bestParticleIndex] < gBest) {
                    gBest =
490 fitnessValueList[bestParticleIndex];
                    gBestLocation =
491 swarm.get(bestParticleIndex).getLocation();
492             }
                optimasi.addRow(new Object[]{t,
    bestParticleIndex, gBestLocation.getLoc()[0],
493 gBestLocation.getLoc()[1], gBestLocation.getLoc()[2],
                    gBestLocation.getLoc()[3],
    gBestLocation.getLoc()[4], gBestLocation.getLoc()[5],
494 gBestLocation.getLoc()[6],
495                 ProblemSet.evaluate(gBestLocation)});
496             t++;
497             //updateFitnessList();
498
499         }
500
501         hasil.addRow(
                    new
502 Object[]{Math.round(gBestLocation.getLoc()[0]),
                        Math.round(gBestLocation.getLoc()[1]),
503 Math.round(gBestLocation.getLoc()[2]),
                        Math.round(gBestLocation.getLoc()[3]),
504 Math.round(gBestLocation.getLoc()[4]),
                        Math.round(gBestLocation.getLoc()[5]),
505 Math.round(gBestLocation.getLoc()[6])
506                 }
507         );
508         double total = 0;
```

```
509             for (int a = 0;
510                     a < PROBLEM_DIMENSION;
511                     a++) {
                    total = total +
    (Math.round(gBestLocation.getLoc()[a]) *
512 ProblemSet.keuntungan[a]);
513             }
514
515             jTextArea1.setText(
                    "Keuntungan optimal yang bisa
516 didapatkan:\n"
                    + "Jumlah rumah tipe 40: " +
517 Math.round(gBestLocation.getLoc()[0])
                    + " keuntungan per rumah: " + new
518 BigDecimal(ProblemSet.keuntungan[0]).toPlainString()
                    + " total = " + new
    BigDecimal((Math.round(gBestLocation.getLoc()[0]) *
519 ProblemSet.keuntungan[0])).toPlainString() + " \n"
                    + "Jumlah rumah tipe 45: " +
520 Math.round(gBestLocation.getLoc()[1])
                    + " keuntungan per rumah: " + new
521 BigDecimal(ProblemSet.keuntungan[1]).toPlainString()
                    + " total = " + new
    BigDecimal((Math.round(gBestLocation.getLoc()[1]) *
522 ProblemSet.keuntungan[1])).toPlainString() + " \n"
                    + "Jumlah rumah tipe 60: " +
523 Math.round(gBestLocation.getLoc()[2])
                    + " keuntungan per rumah: " + new
524 BigDecimal(ProblemSet.keuntungan[2]).toPlainString()
                    + " total = " + new
    BigDecimal((Math.round(gBestLocation.getLoc()[2]) *
525 ProblemSet.keuntungan[2])).toPlainString() + " \n"
                    + "Jumlah rumah tipe 65: " +
526 Math.round(gBestLocation.getLoc()[3])
                    + " keuntungan per rumah: " + new
527 BigDecimal(ProblemSet.keuntungan[3]).toPlainString()
                    + " total = " + new
    BigDecimal((Math.round(gBestLocation.getLoc()[3]) *
528 ProblemSet.keuntungan[3])).toPlainString() + " \n"
                    + "Jumlah rumah tipe 90: " +
529 Math.round(gBestLocation.getLoc()[4])
                    + " keuntungan per rumah: " + new
530 BigDecimal(ProblemSet.keuntungan[4]).toPlainString()
                    + " total = " + new
    BigDecimal((Math.round(gBestLocation.getLoc()[4]) *
531 ProblemSet.keuntungan[4])).toPlainString() + " \n"
                    + "Jumlah rumah tipe 100: " +
532 Math.round(gBestLocation.getLoc()[5])
                    + " keuntungan per rumah: " + new
533 BigDecimal(ProblemSet.keuntungan[5]).toPlainString()
                    + " total = " + new
    BigDecimal((Math.round(gBestLocation.getLoc()[5]) *
534 ProblemSet.keuntungan[5])).toPlainString() + " \n"
                    + "Jumlah rumah tipe 120: " +
535 Math.round(gBestLocation.getLoc()[6])
                    + " keuntungan per rumah: " + new
536 BigDecimal(ProblemSet.keuntungan[6]).toPlainString()
```

```
                + " total = " + new
BigDecimal((Math.round(gBestLocation.getLoc()[6]) *
ProblemSet.keuntungan[6])).toPlainString() + " \n"
                + "Total keuntungan seluruhnya yaitu = " +
new BigDecimal(total).toPlainString() + " \n"
        );
    }

    private void
buttonDefaultActionPerformed(java.awt.event.ActionEvent
evt) {
        fieldSwarmSize.setText("10");
        fieldMax.setText("50");
        fieldProblem.setText("7");
        fieldC1.setText("2");
        fieldC2.setText("2");
        fieldWUp.setText("1");
        fieldWLow.setText("0");
    }

    private void
buttonResetActionPerformed(java.awt.event.ActionEvent evt)
{
        fieldSwarmSize.setText("");
        fieldMax.setText("");
        fieldProblem.setText("");
        fieldC1.setText("");
        fieldC2.setText("");
        fieldWUp.setText("");
        fieldWLow.setText("");
        jTextArea1.setText("");
        optimasi.setRowCount(0);
        hasil.setRowCount(0);
    }

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look
and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not
available, stay with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandf
eel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info
: javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;

                }
            }
```

```
579             } catch (ClassNotFoundException ex) {
580 java.util.logging.Logger.getLogger(hewekHewek.class
581 .getName()).log(java.util.logging.Level.SEVERE, null, ex);
582             } catch (InstantiationException ex) {
583 java.util.logging.Logger.getLogger(hewekHewek.class
584 .getName()).log(java.util.logging.Level.SEVERE, null, ex);
585             } catch (IllegalAccessException ex) {
586 java.util.logging.Logger.getLogger(hewekHewek.class
587 .getName()).log(java.util.logging.Level.SEVERE, null, ex);
            } catch
588 (javax.swing.UnsupportedLookAndFeelException ex) {
589 java.util.logging.Logger.getLogger(hewekHewek.class
590 .getName()).log(java.util.logging.Level.SEVERE, null, ex);
591             }
592             //</editor-fold>
593 
594             /* Create and display the form */
595             java.awt.EventQueue.invokeLater(new Runnable() {
596                 public void run() {
597                     new hewekHewek().setVisible(true);
598                 }
599             });
600     }
601 
602     // Variables declaration - do not modify
603     private javax.swing.JButton buttonDefault;
604     private javax.swing.JButton buttonReset;
605     private javax.swing.JButton buttonStart;
606     private javax.swing.JTextField fieldC1;
607     private javax.swing.JTextField fieldC2;
608     private javax.swing.JTextField fieldMax;
609     private javax.swing.JTextField fieldProblem;
610     private javax.swing.JTextField fieldSwarmSize;
611     private javax.swing.JTextField fieldWLow;
612     private javax.swing.JTextField fieldWUp;
613     private javax.swing.JLabel jLabel1;
614     private javax.swing.JLabel jLabel10;
615     private javax.swing.JLabel jLabel11;
616     private javax.swing.JLabel jLabel12;
617     private javax.swing.JLabel jLabel2;
618     private javax.swing.JLabel jLabel3;
619     private javax.swing.JLabel jLabel4;
620     private javax.swing.JLabel jLabel5;
621     private javax.swing.JLabel jLabel6;
622     private javax.swing.JLabel jLabel7;
623     private javax.swing.JLabel jLabel8;
624     private javax.swing.JLabel jLabel9;
```

```
625     private javax.swing.JPanel jPanel1;
626     private javax.swing.JPanel jPanel2;
627     private javax.swing.JScrollPane jScrollPane1;
628     private javax.swing.JScrollPane jScrollPane2;
629     private javax.swing.JScrollPane jScrollPane3;
630     private javax.swing.JScrollPane jScrollPane4;
631     private javax.swing.JTabbedPane jTabbedPane1;
632     private javax.swing.JTextArea jTextArea1;
633     private javax.swing.JTextArea jTextArea2;
634     private javax.swing.JTable tableHasil;
635     private javax.swing.JTable tableOptimasi;
636     // End of variables declaration
637 }
```

Penjelasan Kode program 5.1 adalah sebagai berikut :

1. Baris 49-54 merupakan proses inisialisasi partikel
2. Baris 56-68 merupakan proses *random location* berdasarkan *range* yang ditentukan dan memberikan kecepatan awal pada setiap partikel
3. Baris 72-76 merupakan *update fitness* memanggil *method getFittnessValue*
4. Baris 78-88 merupakan proses pencarian *fitness* paling kecil
5. Baris 381-383 merupakan proses eksekusi program
6. Baris 391-394 merupakan proses perhitungan *Pbest* awal
7. Baris 402-406 merupakan proses perhitungan *Gbest* awal
8. Baris 408-412 merupakan proses inisialisasi *w* sesuai manualisasi
9. Baris 417-438 merupakan proses *update velocity*
10. Baris 441-455 merupakan proses *update location*
11. Baris 459-477 merupakan proses *random injection*
12. Baris 482-487 merupakan proses *update Pbest*
13. Baris 489-496 merupakan proses *update Gbest*
14. Baris 502-507 merupakan hasil perhitungan pada setiap iterasi
15. Baris 515-540 merupakan keluaran yang berisi hasil optimasi keuntungan

5.1.2 Implementasi Perhitungan Fitness

Proses ini menunjukkan *problem set* yang akan dihitung menggunakan algoritma PSO, dalam hal ini yaitu proses perhitungan jumlah rumah setiap tipe untuk mendapatkan keuntungan tertinggi. Didalam implementasi berikut ini, mengenai kasus (Permata *Garden Regency*) yang akan dihitung. Berdasarkan hasil wawancara serta data yang didapat, didapatkan masukan sebagai berikut.

Kode Program 5.1 Implementasi Perhitungan Fitness

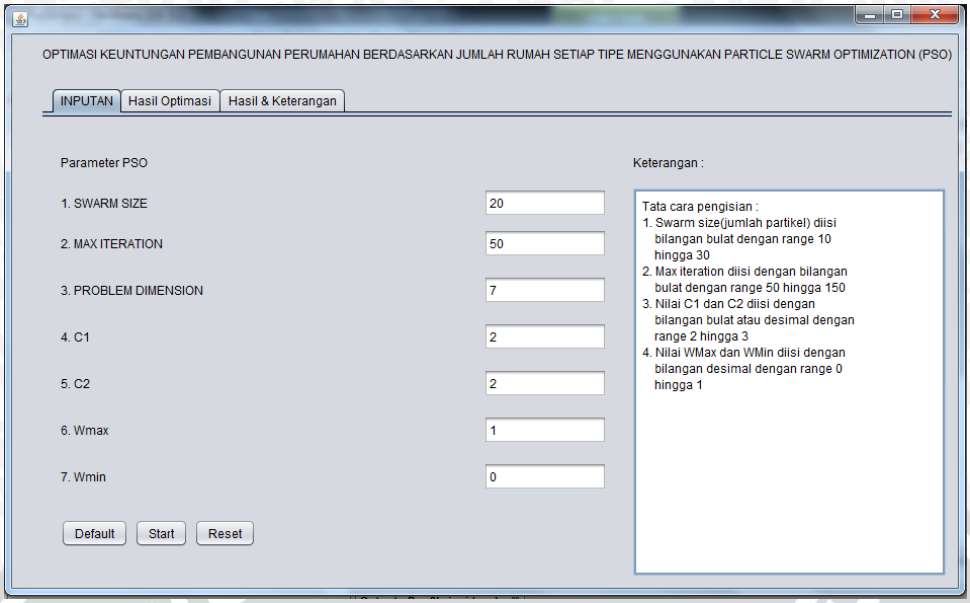
```
1   package PsoSkripsi;
2
3   public class ProblemSet {
4
5       public static final double[][] range = {{14, 15, 1, 1, 3,
6   4, 2}, //min
7       {21, 23, 3, 3, 9, 9, 5}}; //max
8       static double[] keuntungan = {58250000, 54000000,
9   92750000, 90000000, 376500000, 378500000, 453500000};
10      static double[] jmlhpakar = {16, 21, 1, 1, 5, 5, 3}; //52
11      //menghitung fitness
12      public static double evaluate(Location location) {
13          double result = 0;
14          double untung = 0;
15          double selisih = 0;
16          for (int a = 0; a < 7; a++) {
17              untung = untung + (location.getLoc()[a] *
18  keuntungan[a]);
19              selisih = selisih + Math.abs(location.getLoc()[a]
20  - jmlhpakar[a]);
21          }
22          result = 1/(untung + (selisih * 1000000000));
23          return result;
24      }
25  }
```

Penjelasan Kode program 5.2 adalah sebagai berikut:

1. Baris 5-7 merupakan *range* dari setiap tipe rumah yang akan dibangun
2. Baris 8 dan 9 merupakan keuntungan dari setiap tipe rumah yang akan dibangun
3. Baris 10 merupakan jumlah dari setiap rumah yang telah dibangun (diberikan oleh sumber dari hasil wawancara)
4. Baris 12-24 merupakan proses perhitungan *fitness*

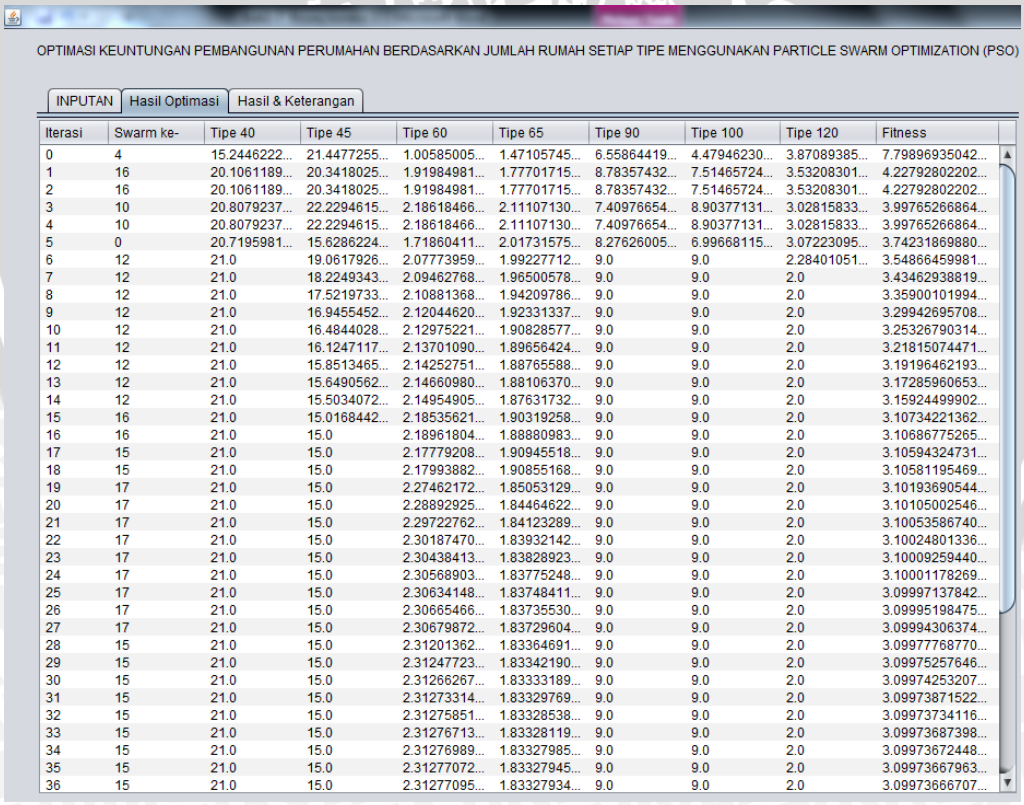
5.2 Implementasi Antarmuka

Antarmuka sistem terdiri dari halaman inputan parameter PSO, selain itu terdapat halaman hasil optimasi yang berisi mengenai iterasi beserta *fitness*, serta halaman hasil dan keterangan yang merupakan jumlah rumah setiap tipe untuk menghasilkan keuntungan terbaik, beserta keterangan perhitungan keuntungannya. Tampilan antarmuka program yang merupakan halaman masukan ditunjukkan dalam Gambar 5.1 berikut:



Gambar 5.1 Tampilan Antarmuka

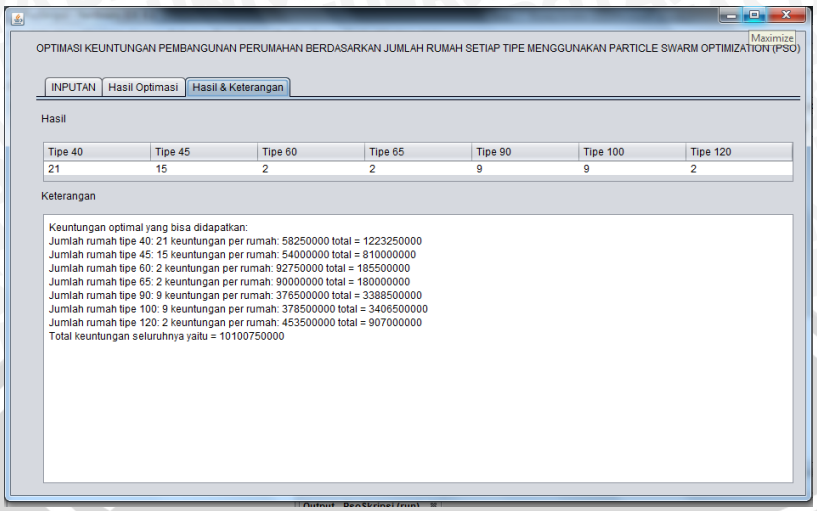
Pada halaman berikutnya merupakan hasil optimasi, dimana terdapat hasil dari setiap iterasi beserta *fitness* dari setiap jumlah tipe rumah. Tampilan halaman hasil optimasi ditunjukkan dalam Gambar 5.2 berikut:



Iterasi	Swarm ke-	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
0	4	15.2446222...	21.4477255...	1.00585005...	1.47105745...	6.55864419...	4.47946230...	3.87089385...	7.79896935042...
1	16	20.1061189...	20.3418025...	1.91984981...	1.77701715...	8.78357432...	7.51465724...	3.53208301...	4.22792802202...
2	16	20.1061189...	20.3418025...	1.91984981...	1.77701715...	8.78357432...	7.51465724...	3.53208301...	4.22792802202...
3	10	20.8079237...	22.2294615...	2.18618466...	2.11107130...	7.40976654...	8.90377131...	3.02815833...	3.99765266864...
4	10	20.8079237...	22.2294615...	2.18618466...	2.11107130...	7.40976654...	8.90377131...	3.02815833...	3.99765266864...
5	0	20.7195981...	15.6286224...	1.71860411...	2.01731575...	8.27626005...	6.99668115...	3.07223095...	3.74231869880...
6	12	21.0	19.0617926...	2.07773959...	1.99227712...	9.0	9.0	2.28401051...	3.54866459981...
7	12	21.0	18.2249343...	2.09462768...	1.96500578...	9.0	9.0	2.0	3.43462938819...
8	12	21.0	17.5219733...	2.10881368...	1.94209786...	9.0	9.0	2.0	3.35900101994...
9	12	21.0	16.9455452...	2.12044620...	1.92331337...	9.0	9.0	2.0	3.29942695708...
10	12	21.0	16.4844028...	2.12975221...	1.90828577...	9.0	9.0	2.0	3.25326790314...
11	12	21.0	16.1247117...	2.13701090...	1.89656424...	9.0	9.0	2.0	3.21815074471...
12	12	21.0	15.8513465...	2.14252751...	1.88765588...	9.0	9.0	2.0	3.19196462193...
13	12	21.0	15.6490562...	2.14660980...	1.88106370...	9.0	9.0	2.0	3.17285960653...
14	12	21.0	15.5034072...	2.14954905...	1.87631732...	9.0	9.0	2.0	3.15924499902...
15	16	21.0	15.0168442...	2.18535621...	1.90319258...	9.0	9.0	2.0	3.10734221362...
16	16	21.0	15.0	2.18961804...	1.88880983...	9.0	9.0	2.0	3.10686775265...
17	15	21.0	15.0	2.17779208...	1.90945518...	9.0	9.0	2.0	3.10594324731...
18	15	21.0	15.0	2.17993882...	1.90855168...	9.0	9.0	2.0	3.10581195469...
19	17	21.0	15.0	2.27462172...	1.85053129...	9.0	9.0	2.0	3.10193690544...
20	17	21.0	15.0	2.28892925...	1.84464622...	9.0	9.0	2.0	3.10105002546...
21	17	21.0	15.0	2.29722762...	1.84123289...	9.0	9.0	2.0	3.10053586740...
22	17	21.0	15.0	2.30187470...	1.83932142...	9.0	9.0	2.0	3.10024801336...
23	17	21.0	15.0	2.30438413...	1.83828923...	9.0	9.0	2.0	3.10009259440...
24	17	21.0	15.0	2.30568903...	1.83775248...	9.0	9.0	2.0	3.10001178269...
25	17	21.0	15.0	2.30634148...	1.83748411...	9.0	9.0	2.0	3.09997137842...
26	17	21.0	15.0	2.30665466...	1.83735530...	9.0	9.0	2.0	3.09995198475...
27	17	21.0	15.0	2.30679872...	1.83729604...	9.0	9.0	2.0	3.09994306374...
28	15	21.0	15.0	2.31201362...	1.83364691...	9.0	9.0	2.0	3.09977768770...
29	15	21.0	15.0	2.31247723...	1.83342190...	9.0	9.0	2.0	3.09975257646...
30	15	21.0	15.0	2.31266267...	1.83333189...	9.0	9.0	2.0	3.09974253207...
31	15	21.0	15.0	2.31273314...	1.83329769...	9.0	9.0	2.0	3.09973871522...
32	15	21.0	15.0	2.31275851...	1.83328538...	9.0	9.0	2.0	3.09973734116...
33	15	21.0	15.0	2.31276713...	1.83328119...	9.0	9.0	2.0	3.09973687398...
34	15	21.0	15.0	2.31276989...	1.83327985...	9.0	9.0	2.0	3.09973672448...
35	15	21.0	15.0	2.31277072...	1.83327945...	9.0	9.0	2.0	3.09973667963...
36	15	21.0	15.0	2.31277095...	1.83327934...	9.0	9.0	2.0	3.09973666707...

Gambar 5.2 Halaman Hasil Optimasi

Selain itu terdapat hasil dan keterangan yang berisi jumlah rumah setiap tipe untuk menghasilkan keuntungan terbaik, beserta keterangan perhitungan keuntungannya. Tampilan halaman hasil ditunjukkan dalam Gambar 5.3 berikut:



Gambar 5.3 Halaman Hasil dan Keterangan

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini dibahas mengenai pengujian dan analisis terhadap sistem yang telah diimplementasikan sebelumnya. Proses pengujian dilakukan sesuai dengan perancangan pengujian pada Bab 4.

6.1 Pengujian

Pada sub bab ini akan dijelaskan mengenai pengujian yang akan dilakukan pada sistem untuk menghasilkan solusi optimal. Proses tersebut melibatkan parameter-parameter yang harus ditentukan untuk menjalankan algoritma PSO. Oleh karena itu, pengujian terhadap parameter diperlukan untuk menentukan nilai parameter terbaik.

Tidak adanya metode tertentu untuk menentukan parameter algoritma PSO membuat perlu dilakukannya evaluasi dengan beberapa pengujian agar mendapatkan parameter yang optimal. Pengujian tersebut antara lain:

1. Pengujian untuk menentukan jumlah partikel (*Swarmsize*) yang optimal.
2. Pengujian untuk menentukan jumlah iterasi yang optimal.
3. Pengujian jumlah iterasi terhadap waktu komputasi.

6.2 Hasil Pengujian

Berdasarkan skenario pengujian tersebut, maka hasil pengujian dari sistem dijelaskan pada sub bab 6.2.1 sampai dengan sub bab 6.2.2.

6.2.1 Menentukan Jumlah Partikel (*Swarmsize*) yang Optimal.

Pengujian banyaknya jumlah partikel digunakan untuk menentukan jumlah partikel yang digunakan untuk untuk isian *default* sistem, sehingga pengguna tidak perlu mengisikan parameter-parameter PSO, dan sudah memberikan solusi yang terbaik. Pengujian ini menggunakan beberapa ukuran swarmsize yang berbeda dengan kelipatan 2. Pengujian dilakukan sebanyak 5 kali untuk setiap jumlah partikel. Dalam hal ini, parameter yang digunakan adalah:

problem dimension:		7
c1	:	2
c2	:	1
Wmin	:	0
Wmax	:	1
r1	:	0.163251
r2	:	0.183259
V(1,0)	:	0
max iteration	:	50

Pengujian ini dilakukan sesuai dengan perancangan pengujian pada bab 4.4.1 yang hasilnya dapat dilihat pada Tabel 6.2 berikut ini.

Tabel 6.1 Hasil Pengujian Jumlah Partikel

Jumlah partikel (*Swarmsize)*	Nilai *fitness* pada percobaan ke-				
	1	2	3	4	5
5	3.57E-11	3.31E-11	3.36E-11	3.44E-11	3.00E-11
7	3.07E-11	3.12E-11	2.99E-11	3.06E-11	2.77E-11
9	3.15E-11	3.22E-11	3.54E-11	3.18E-11	3.41E-11
11	3.06E-11	2.93E-11	3.04E-11	2.96E-11	3.02E-11
13	2.75E-11	2.83E-11	2.73E-11	3.00E-11	2.74E-11
15	3.24E-11	3.22E-11	3.11E-11	3.25E-11	3.09E-11

Tabel 6.2 Rata-Rata Fitness dari Hasil Pengujian Jumlah Partikel

Jumlah partikel (*Swarmsize)*	Rata-rata *fitness*
5	3.34E-11
7	3.00E-11
9	3.30E-11
11	3.00E-11
13	2.81E-11
15	3.18E-11

Tabel 6.2 menunjukkan bahwa nilai fitness tidak selalu semakin membaik seiring dengan bertambahnya jumlah partikel.

6.2.2 Menentukan Jumlah Maksimum Iterasi yang Optimal

Pengujian terhadap banyaknya jumlah maksimum iterasi untuk mengetahui maksimum iterasi yang optimal dan dapat menghasilkan solusi yang lebih baik. Pada pengujian jumlah partikel telah didapatkan jumlah partikel yang tidak terlalu banyak, akan tetapi memliki hasil yang bagus yaitu 13. Nilai parameter tersebut akan digunakan dalam pengujian ini. Pengujian ini menggunakan beberapa ukuran maksimum iterasi yang berbeda dengan kelipatan 15. Pengujian dilakukan sebanyak 5 kali untuk masing-masing ketentuan maksimum iterasi. Kemudian masing-masing data akan dicatat nilai *fitness* untuk didapatkan rata-rata nilai *fitness* terbaik. Indikator pengujian adalah nilai rata-rata *fitness* terendah atau terbaik. Nilai parameter yang digunakan adalah sebagai berikut.

swarm size : 13 (merupakan rata-rata hasil fitness terbaik)
problem dimension : 7
c1 : 2
c2 : 1
Wmin : 0
Wmax : 1
r1 : 0.163251
r2 : 0.183259
V(1,0) : 0
max iteration : Parameter yang diuji coba

Pengujian ini dilakukan sesuai dengan perancangan pengujian pada sub bab 4.4.2. Hasil pengujian dapat dilihat pada Tabel 6.3 berikut ini.

Tabel 6.3 Hasil pengujian maksimum iterasi

Max Iteration	Nilai *fitness* pada percobaan ke-				
	1	2	3	4	5
20	3.19E-11	3.23E-11	3.61E-11	3.08E-11	3.49E-11
35	2.86E-11	3.03E-11	2.88E-11	3.35E-11	3.42E-11
50	3.03E-11	3.00E-11	3.08E-11	3.12E-11	2.89E-11
65	2.75E-11	3.11E-11	2.90E-11	2.90E-11	3.09E-11
80	2.73E-11	2.92E-11	3.04E-11	2.90E-11	3.31E-11

Tabel 6.4 Rata-rata fitness dari hasil pengujian maksimum iterasi

$t\ max$	Rata-rata *fitness*
20	3.32E-11
35	3.11E-11
50	3.02E-11
65	2.95E-11
80	2.98E-11

Berdasarkan Tabel 6.4 menunjukkan bahwa nilai fitness tidak selalu semakin membaik seiring dengan bertambahnya jumlah iterasi.

6.2.3 Menentukan Waktu Komputasi Berdasarkan Jumlah Maksimum Iterasi

Pengujian terhadap waktu komputasi pada setiap jumlah maksimum iterasi untuk mengetahui waktu yang dibutuhkan sistem dalam memproses. Pada pengujian jumlah partikel telah didapatkan jumlah partikel yang terbaik yaitu 13. Nilai parameter tersebut akan digunakan dalam pengujian ini. Pengujian ini menggunakan beberapa ukuran maksimum iterasi yang berbeda dengan kelipatan 15. Kemudian masing-masing waktu komputasi yang dibutuhkan data akan dicatat. Indikator pengujian adalah waktu komputasi. Nilai parameter yang digunakan adalah sebagai berikut.

swarm size : 13 (merupakan rata-rata hasil fitness terbaik)
problem dimension : 7
c1 : 2
c2 : 1
Wmin : 0
Wmax : 1
r1 : 0.163251
r2 : 0.183259
V(1,0) : 0
max iteration : Parameter yang diuji coba

Pengujian ini dilakukan sesuai dengan perancangan pengujian pada sub bab 4.4.3. Hasil pengujian dapat dilihat pada Tabel 6.5 berikut ini.

Tabel 6.5 Waktu komputasi terhadap jumlah maksimum iterasi

$t\ max$	Waktu komputasi (detik)
20	5
35	5
50	6
65	7
80	8

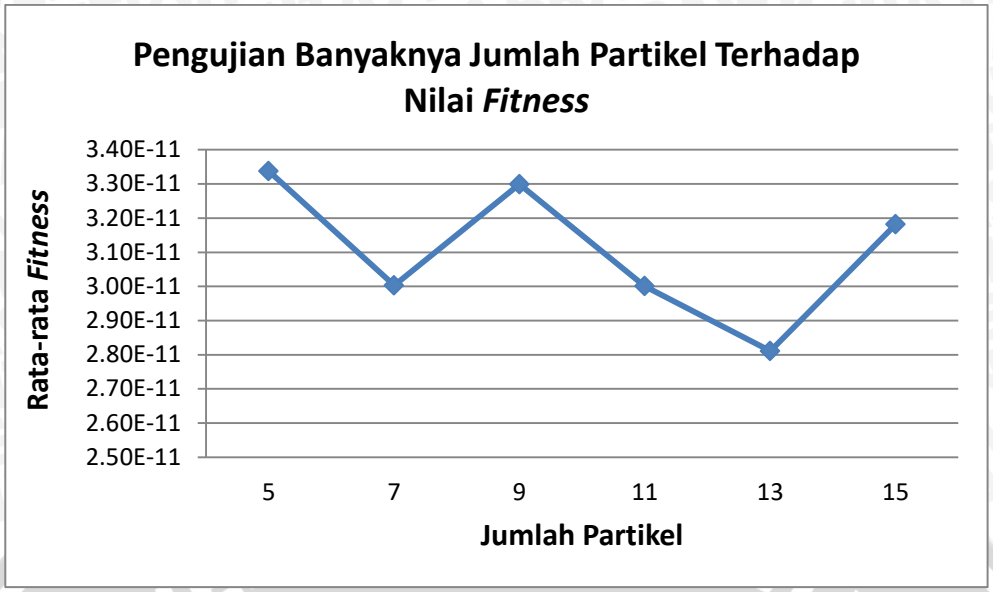
Berdasarkan Tabel 6.5 dapat disimpulkan bahwa semakin banyak jumlah iterasi akan dibutuhkan pula waktu komputasi yang cukup lama.

6.3 Analisis hasil pengujian

Pada sub bab analisis hasil pengujian ini akan dilakukan analisis terhadap data hasil pengujian yang telah dilakukan sebelumnya berdasarkan sub bab 6.1.1 dan 6.1.2.

6.3.1 Analisis Hasil Pengujian Banyaknya Jumlah Partikel

Setelah dilakukan pengujian terhadap banyaknya jumlah partikel yang akan digunakan dalam memperoleh hasil optimasi didapatkan hasil rata-rata seperti pada tabel 6.2. Pada Gambar 6.1 dibawah ini, ditunjukkan bagaimana jumlah partikel berpengaruh terhadap hasil *fitness* yang ada. Seperti tertera pada grafik dibawah ini

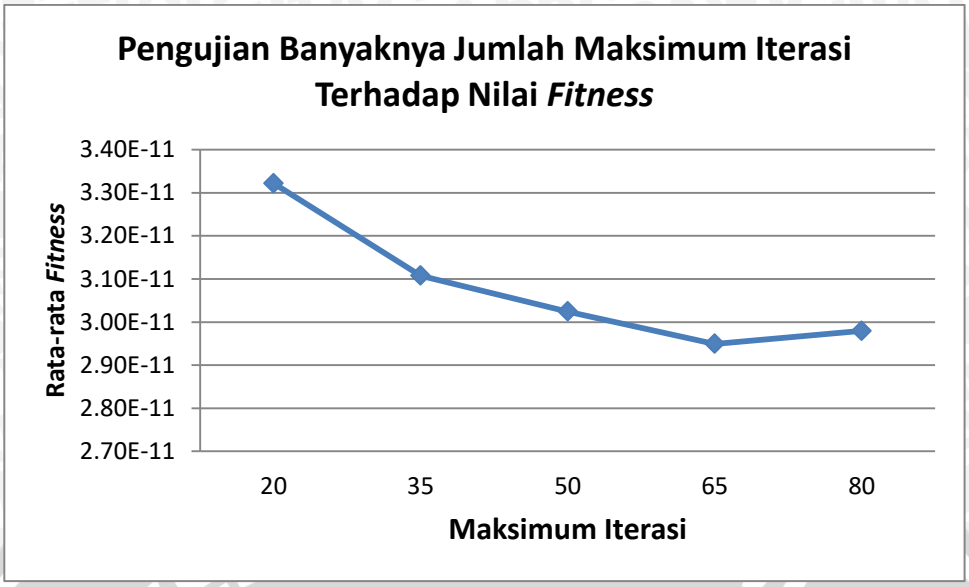


Gambar 6.1 Grafik Hasil Pengujian Banyaknya Jumlah Partikel Terhadap Nilai *Fitness*

Berdasarkan grafik pada Gambar 6.1 menunjukkan bahwa tidak selalu semakin besar jumlah partikel, maka rata-rata fitness yang didapatkan akan semakin bagus. Dalam hal ini, dapat dikatakan bahwa solusi yang didapatkan untuk jumlah partikel adalah 13. Tidak menuntut kemungkinan bahwa jumlah partikel yang lebih banyak akan membuat hasil *fitness* lebih optimal, dikarenakan jumlah partikel yang banyak akan memberikan inisialisasi partikel yang lebih bervariasi dan lebih menyeluruh.

6.3.2 Analisis Hasil Pengujian Banyaknya Jumlah Maksimum Iterasi

Selain pengujian terhadap jumlah partikel, dilakukan pula pengujian terhadap jumlah maksimum iterasi. Dari hasil pengujian sebelumnya, didapatkan jumlah partikel yang optimal adalah 13. Kemudian dibuatlah Tabel 6.4 berdasar pada jumlah maksimum iterasi yang optimal. Maka dibuatlah grafik yang sesuai dengan Tabel 6.4 seperti berikut ini.

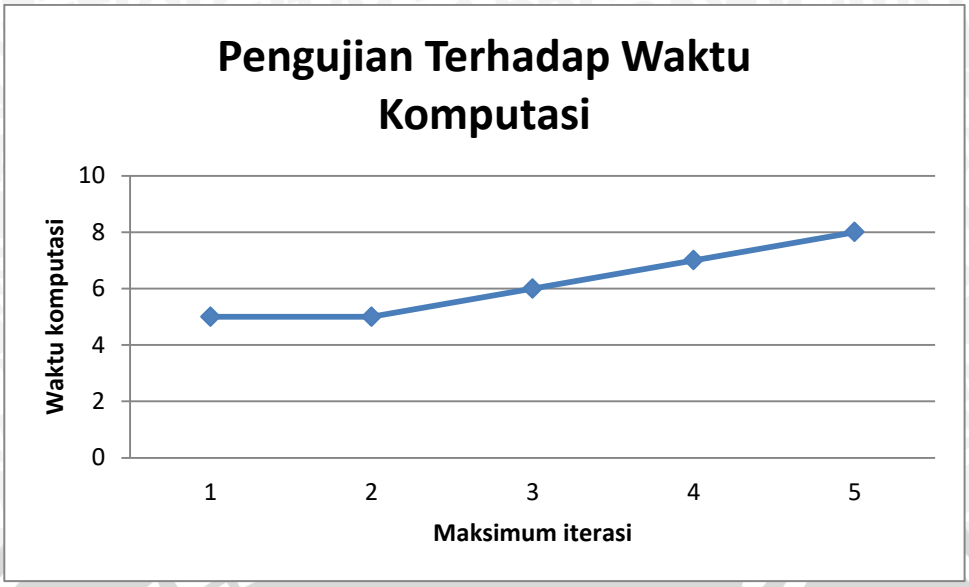


Gambar 6.2 Grafik Hasil Pengujian Banyaknya Jumlah Maksimum Iterasi Terhadap Nilai *Fitness*

Berdasarkan grafik hasil pengujian diatas dapat dikatakan bahwa tidak selalu jumlah iterasi yang semakin banyak menghasilkan *fitness* yang semakin kecil atau baik. Dilihat dari hasil yang ada saat melakukan perhitungan *fitness*, tidak ada perubahan pada hampir 5-10 iterasi terakhir, hal ini dikarenakan partikel tetap diam atau dalam posisi tersebut atau tidak mengalami perpindahan sama sekali karena hasil tersebut adalah hasil maksimal. Dapat disimpulkan dari gambar diatas bahwa 65 adalah jumlah maksimum iterasi yang dapat menghasilkan solusi yang optimal.

6.3.3 Analisis Hasil Pengujian Waktu Komputasi

Selain pengujian terhadap jumlah partikel, dilakukan pula pengujian terhadap jumlah maksimum iterasi. Dari hasil pengujian sebelumnya, didapatkan jumlah partikel yang optimal adalah 13 dan maksimum iterasi adalah 65. Kemudian dibuatlah Tabel 6.5 berdasar pada jumlah partikel yang optimal pada Tabel 6.2 dan jumlah maksimum iterasi pada Tabel 6.4, maka dibuatlah grafik yang sesuai dengan Tabel 6.5 seperti berikut ini.



Gambar 6.3 Grafik Hasil Pengujian Banyaknya Jumlah Maksimum Iterasi Terhadap Waktu Komputasi

Berdasarkan grafik hasil pengujian diatas dapat dikatakan bahwa banyak jumlah iterasi, maka dibutuhkan waktu komputasi yang lebih lama. Dilihat dari pengujian sebelumnya bahwa maksimal iterasi sebanyak 65 memberikan rata-rata *fitness* yang baik serta waktu komputasi yang tidak terlalu lama.

6.3.4 Analisis Keseluruhan dari Hasil pengujian

Berdasarkan hasil pengujian yang dilakukan pada sistem, maka parameter algoritma yang dapat dikatakan baik adalah sebagai berikut.

swarm size : 13 (merupakan rata-rata hasil fitness terbaik)
problem dimension : 7
c1 : 2
c2 : 1
Wmin : 0
Wmax : 1
r1 : 0.163251
r2 : 0.183259
V(1,0) : 0
max iteration : 65 (merupakan rata-rata hasil fitness terbaik)

Dari pengujian yang dilakukan menggunakan parameter yang sudah diuji dan dioptimalkan, didapatkan hasil optimasi keuntungan pembangunan perumahan berdasarkan jumlah rumah setiap tipe yang tertera pada Tabel 6.6 berikut ini.

Tabel 6.6 Hasil Perhitungan Sistem

Iterasi	Swarm ke-	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness

Iterasi	Swarm ke-	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
0	12	16.49517	17.04019	2.192441	1.76056	8.291324	7.751084	3.50111	4.38E-11
1	12	16.49517	17.04019	2.192441	1.76056	8.291324	7.751084	3.50111	4.38E-11
2	12	16.49517	17.04019	2.192441	1.76056	8.291324	7.751084	3.50111	4.38E-11
3	3	20.48608	20.90394	2.196739	2.469976	6.413432	8.033609	4.898866	4.16E-11
4	3	20.48608	20.90394	2.196739	2.469976	6.413432	8.033609	4.898866	4.16E-11
5	7	18.48938	16.22919	1.783086	1.856263	8.603045	7.599055	3.457319	3.92E-11
6	10	19.20325	18.39655	2.405036	1.832491	8.286733	9	4.164768	3.65E-11
7	10	19.80944	18.50319	2.453455	1.807941	8.428208	9	4.264392	3.53E-11
8	10	20.34103	18.5967	2.495915	1.786412	8.552271	9	4.351755	3.44E-11
9	10	20.79901	18.67727	2.532495	1.767864	8.659156	9	4.427022	3.36E-11
10	10	21	18.74544	2.563448	1.75217	8.749597	9	4.490709	3.31E-11
11	8	21	17.72976	2.629841	1.955143	8.988742	9	4.104998	3.20E-11
12	8	21	17.72976	2.629841	1.955143	8.988742	9	4.104998	3.20E-11
13	8	21	17.72976	2.629841	1.955143	8.988742	9	4.104998	3.20E-11
14	11	21	16.85569	2.637954	1.773529	9	9	3.962994	3.16E-11
15	11	21	16.0729	2.596366	1.686227	9	9	3.789549	3.13E-11
16	11	21	15.4828	2.565015	1.620415	9	9	3.658797	3.10E-11
17	11	21	15.04703	2.541864	1.571815	9	9	3.562242	3.08E-11
18	11	21	15.04703	2.541864	1.571815	9	9	3.562242	3.08E-11
19	9	21	15.28682	2.774814	1.849727	9	9	3.640696	3.04E-11
20	9	21	15	2.77186	1.839003	9	9	3.54057	3.03E-11
21	9	21	15	2.77186	1.839003	9	9	3.54057	3.03E-11
22	9	21	15	2.77186	1.839003	9	9	3.54057	3.03E-11
23	11	21	15.00218	2.761913	1.820042	9	9	3.568626	3.03E-11
24	11	21	15.00204	2.785372	1.849923	9	9	3.571612	3.02E-11
25	11	21	15.001	2.7998	1.8683	9	9	3.5734	3.02E-

Iterasi	Swarm ke-	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
			96	09	11			5	11
26	11	21	15.001 91	2.8084 71	1.8793 43	9	9	3.5745 53	3.02E-11
27	11	21	15.001 89	2.8135 35	1.8857 93	9	9	3.5751 97	3.02E-11
28	11	21	15.001 87	2.8164 17	1.8894 65	9	9	3.5755 64	3.02E-11
29	11	21	15.001 86	2.8180 14	1.8914 98	9	9	3.5757 67	3.02E-11
30	11	21	15.001 86	2.8188 73	1.8925 93	9	9	3.5758 77	3.02E-11
31	11	21	15.001 85	2.8193 23	1.8931 66	9	9	3.5759 34	3.01E-11
32	11	21	15.001 85	2.8195 51	1.8934 57	9	9	3.5759 63	3.01E-11
33	11	21	15.001 85	2.8196 63	1.8936	9	9	3.5759 77	3.01E-11
34	11	21	15.001 85	2.8197 17	1.8936 68	9	9	3.5759 84	3.01E-11
35	9	21	15.000 63	2.8186 04	1.8925 25	9	9	3.5794 71	3.01E-11
36	9	21	15.000 69	2.8191 37	1.8931 36	9	9	3.5793 7	3.01E-11
37	9	21	15.000 71	2.8193 67	1.8934	9	9	3.5793 26	3.01E-11
38	9	21	15.000 72	2.8194 62	1.8935 09	9	9	3.5793 08	3.01E-11
39	9	21	15.000 73	2.8195	1.8935 53	9	9	3.5793 01	3.01E-11
40	9	21	15.000 73	2.8195 15	1.8935 7	9	9	3.5792 98	3.01E-11
41	9	21	15.000 73	2.8195 2	1.8935 76	9	9	3.5792 97	3.01E-11
42	9	21	15.000 73	2.8195 22	1.8935 78	9	9	3.5792 96	3.01E-11
43	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
44	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
45	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
46	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
47	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
48	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
49	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
50	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11

Iterasi	Swarm ke-	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
51	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
52	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
53	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
54	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
55	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
56	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
57	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
58	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
59	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
60	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
61	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
62	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
63	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11
64	9	21	15.000 73	2.8195 23	1.8935 79	9	9	3.5792 96	3.01E-11

Hasil keluaran sistem menunjukkan bahwa hasil optimal dari perhitungan didapatkan pada iterasi ke-49 dan swarm ke-0 yang ditunjukkan pada Tabel 6.10 berikut ini.

Tabel 6.7 Hasil Optimal Sistem

iterasi	swarm ke-	tipe 40	tipe 45	tipe 60	tipe 65	tipe 90	tipe 100	tipe 120	Fitness
64	9	21	15	3	2	9	9	4	3.01E-11

Berdasarkan Tabel 6.7 dapat dihitung berapa jumlah keuntungan yang didapatkan, kemudian dibandingkan dengan hasil dari pakar adalah sebagai berikut.

Tabel 6.8 Hasil Perhitungan Keuntungan oleh Sistem dan Hasil Perhitungan Keuntungan oleh Pakar

Tipe Rumah	Sistem		Pakar	
	Jumlah Rumah	Keuntungan	Jumlah Rumah	Keuntungan
tipe 40	21	Rp.1.223.250.000,-	16	Rp.932.000.000,-
tipe 45	15	Rp.810.000.000,-	21	Rp.1.134.000.000,-
tipe 60	3	Rp.278.250.000,-	1	Rp.92.750.000,-
tipe 65	2	Rp.180.000.000,-	1	Rp.90.000.000,-
tipe 90	9	Rp.3.388.500.000,-	5	Rp.1.882.500.000,-
tipe 100	9	Rp.3.406.500.000,-	5	Rp.1.892.500.000,-
tipe 120	4	Rp.1.814.000.000,-	3	Rp.1.360.500.000,-
Total	63	Rp.11.100.500.000,-	52	Rp.4.131.250.000,-

Dari perhitungan di atas dapat dihitung perbandingan keuntungan antara data yang diberikan oleh hasil wawancara dan data hasil optimasi menggunakan algoritma PSO adalah 11.100.500.000 - 4.131.250.000 = Rp.6.969.250.000,-

BAB 7 PENUTUP

Dalam Bab 7 ini akan diberikan kesimpulan serta saran dari hasil penelitian yang telah dilakukan. Kesimpulan yang diambil adalah untuk menjawab rumusun masalah dan menyelesaikan permasalahan yang telah dijelaskan pada Bab 1. Kesimpulan akan dijelaskan dalam sub bab 7.1 dan saran akan diberikan pada sub bab 7.2.

7.1 Kesimpulan

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut.

1. Proses PSO dapat diterapkan pada masalah perhitungan optimasi pembangunan perumahan. Dengan mendefinisikan sejumlah partikel sebagai solusi dalam setiap ruang pencarian secara random. Kemudian pada proses pencarian solusi terbaik, dihitung kecepatan perpindahan partikel dalam ruang pencarian dan posisi partikel setelah perpindahan. Posisi partikel inilah yang merepresentasikan jumlah rumah yang akan dibangun untuk setiap tipe. Setiap perpindahan posisi, partikel akan mengevaluasi dan membandingkan posisi yang pernah dilalui dan diambil menjadi posisi terbaik individu. Selanjutnya solusi terbaik dipilih dari posisi terbaik *global* yang diambil dengan cara membandingkan seluruh posisi terbaik individu.
2. Penentuan nilai parameter yang tepat sangat penting untuk mendapatkan hasil optimasi keuntungan. Didapatkan pada pengujian-pengujian diatas bahwa parameter terbaik yang dapat digunakan pada kasus ini adalah jumlah partikel sebanyak 13, sedangkan untuk iterasi maksimal sebanyak 65.
3. Didapatkan pula perbedaan jumlah rumah setiap tipe antara pakar dan sistem adalah sebanyak 11.

7.2 Saran

1. Penelitian ini dilakukan pada cluster Permata Garden Regency, diharapkan penelitian dapat diterapkan dalam optimasi keuntungan untuk kasus lain.
2. Dalam parameter PSO tidak terdapat perhitungan dalam menentukan parameter tersebut. Diharapkan dengan lebih banyak percobaan mengenai penentuan parameter, akan menghasilkan keuntungan yang lebih optimal.

DAFTAR PUSTAKA

Ashri F., Putri Y. E., & Indryani R. 2011. Optimasi Jumlah Unit Rumah Tiap Tipe Pada Perumahan Green Hill Gresik. Surabaya.

Bisilisin F. Y., Herdiyeni Y., & Silalahi B. P. 2014. Optimasi K-Means Clustering Menggunakan Particle Swarm Optimization pada Sistem Identifikasi Tumbuhan Obat Berbasis Citra. Bogor.

BUMN. Tentang Perusahaan. Tersedia di <http://bumn.go.id/perumnas/halaman/41/tentang-perusahaan.html> [Diakses 7 Februari 2016]

Erny. 2013. Optimasi Pola Penyusunan Barang dalam Peti Kemas Menggunakan Algoritma Particle Swarm Optimization. Makassar.

Kennedy J. & Eberhart R. 1995. Particle Swarm Optimization. International Conference on Neural Networks, vol. 4, Nov 1995, 19421948. New Jersey.

Mahmudy, W. F. (2014). Optimasi penjadwalan two-stage assembly flowshop menggunakan algoritma genetika yang dimodifikasi. Konferensi Nasional Sistem Informasi (KNSI), pp. 478-483.

Mahmudy, W. F. (2014). Optimasi Part Type Selection And Machine Loading Problems Pada FMS Menggunakan Metode Particle Swarm Optimization. Konferensi Nasional Sistem Informasi 2014. Makassar.

Mansur, Prahasto T., & Farikhin. 2014. Particle Swarm Optimization Untuk Sistem Informasi Penjadwalan Resource Di Perguruan Tinggi. Semarang.

Nadiasa M. 2012. OPTIMALISASI PEMBANGUNAN PERUMAHAN GRAND RENON PRIME RESIDENCE. Denpasar.

Nidia Z. 2013. REI: Swasta Lebih Berperan Dalam Sektor Perumahan. Jakarta. Tersedia di <http://www.republika.co.id/berita/ekonomi/bisnis/13/11/20/mwjupo-rei swasta-lebih-berperan-dalam-sektor-perumahan> [Diakses 7Februari 2016]

Ratna P. P. S., Dewi C., Indriati. 2013. IMPLEMENTASI ALGORITMA PARTICLE SWARM OPTIMIZATION UNTUK OPTIMASI FUNGSI KEANGGOTAAN PADA KONDISI PENDERITA PENYAKIT HEPATITIS. Malang.

Real Estate Indonesia. 2015. Bangun Satu Juta Rumah, REI Beri 14 Syarat ke Pemerintah. Jakarta. Tersedia di <http://www.rei.or.id/berita.php> [Diakses 7Februari 2016]

Tuegeh M., Soeprijanto A., & Hery M. P. 2009. OPTIMAL GENERATOR SCHEDULING BASED ON PARTICLE SWARM OPTIMIZATION. *Seminar Nasional Informatika 2009 (semnasIF 2009)* . Yogyakarta.

Tuegeh M., Soeprijanto A., & Hery M. P. (2009). MODIFIED IMPROVED PARTICLE SWARM OPTIMIZATION FOR OPTIMAL GENERATOR SCHEDULING. Seminar

Nasional Aplikasi Teknologi Informasi 2009 (SNATI 2009). ISSN: 1907-5022. Yogyakarta.

UNDANG-UNDANG REPUBLIK INDONESIA NOMOR 4 TAHUN 1992. 1992. Tersedia di <http://www.sjdih.depkeu.go.id/fulltext/1992/4TAHUN~1992UUPenj.htm> [Diakses 7Februari 2016]

Wati D. A. R. & Rochman Y. A. (2013). Model Penjadwalan Matakuliah Secara Otomatis Berbasis Algoritma *Particle Swarm Optimization (PSO).* Yogyakarta.

Zerda E. R. 2009. Analisis dan Penerapan Algoritma *Particle Swarm Optimization (PSO)* pada Optimasi Penjadwalan Sumber Daya Proyek. Bandung.

LAMPIRAN 1 SURAT KETERANGAN

Lampiran surat keterangan wawancara dan perhitungan keuntungan di PT. Goldenindo Lestari mengenai Cluster Permata Garden Regency.

LAMPIRAN 2 SURAT KETERANGAN

Lampiran surat keterangan pengaplikasian hasil di PT. Goldenindo Lestari mengenai Cluster Permata Garden Regency.

LAMPIRAN 3 PENGUJIAN JUMLAH PARTIKEL

Swarmsize 5

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
49	3	21	15.55128	1.331062	3	4.231661	8.078285	5	3.57E-11
49	3	14	15	2.811233	1	9	8.50873	5	3.31E-11
49	3	20.71093	15.68144	2.884107	2.178686	7.232054	7.222916	5	3.36E-11
49	3	20.48954	16.48801	2.737288	1.626206	7.993709	7.803518	4.447729	3.44E-11
49	3	21	15	3	1.9279	9	9	3.491272	3.00E-11

Swarmsize 7

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
49	4	21	15	1.880357	3	7.325301	9	4.620819	3.07E-11
49	6	19.02447	15	1.370071	3	9	8.499582	5	3.12E-11
49	4	21	15	3	3	6.653771	9	5	2.99E-11
49	6	21	16.71194	2.522192	2.248644	9	8.834897	4.459621	3.06E-11
49	5	21	15	2.881407	2.595871	9	9	5	2.77E-11

Swarmsize 9

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
49	6	21	15	1.03571	2.668634	9	9	2	3.15E-11
49	5	21	15	1.422742	1.703521	9	9	2.189543	3.22E-11
49	7	21	15	1.230134	1	3	9	5	3.54E-11
49	5	21	15	2.663275	3	3.842128	9	5	3.18E-11
49	6	21	15	3	3	3	8.271343	3.571444	3.41E-11

Swarmsize 11

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
49	9	20.97527	15.64738	2.759766	2.10915	8.529724	8.512642	4.479417	3.06E-11
49	8	21	15	3	3	9	9	2.183431	2.93E-11
49	10	21	15	1.33265	1.243578	9	8.999357	5	3.04E-11
49	9	21	15	2.819574	3	9	9	2.538907	2.96E-11
49	10	21	15	2.550116	2.381832	9	9	2	3.02E-11

Swarmsize 13

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
49	12	21	15	3	3	9	9	4.791155	2.75E-11
49	8	21	15	2.378084	2.450202	9	9	5	2.83E-11
49	8	21	15	3	3	9	9	5	2.73E-11
49	10	21	15	1	2.995718	9	9	4.220709	3.00E-11
49	10	21	15.22174	3	3	9	9	5	2.74E-11

Swarmsize 15

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
49	12	21	15	3	2.907367	9	4	4.457524	3.24E-11
49	12	14	15	3	2.050735	9	9	4.232855	3.22E-11
49	9	21	15	3	1	9	9	2	3.11E-11
49	14	20.85253	17.71477	2.244083	2.359224	8.59737	8.433039	4.808895	3.25E-11
49	13	14	15	3	3	9	8.395575	5	3.09E-11

LAMPIRAN 4 PENGUJIAN JUMLAH ITERASI

Iterasi 20

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
19	10	19.98523	15	1.100338	1.524705	8.694774	8.89813	5	3.19E-11
19	12	19.70211	15	2.90086	3	7.655002	9	2	3.23E-11
19	8	20.50173	16.02966	2.731103	2.645703	3.098116	8.946013	2.071279	3.61E-11
19	11	21	16.77543	2.333189	2.284391	9	9	4.312688	3.08E-11
19	11	19.82616	19.36599	1.601498	1.98064	9	9	5	3.49E-11

Iterasi 35

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
34	9	20.60457	15	3	1.896541	9	9	5	2.86E-11
34	9	21	15	1	1.753972	9	9	4.935808	3.03E-11
34	11	21	17.02375	3	3	9	9	5	2.88E-11
34	8	20.98999	15	2.572182	1.700202	3	9	4.568553	3.35E-11
34	10	19.23716	15	2.151772	3	7.654835	9	2.803631	3.42E-11

Iterasi 50

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
49	11	21	15	2.920324	1.252626	8.566016	8.249127	5	3.03E-11
49	12	21	15	3	2.189518	9	8.997064	2.142436	3.00E-11
49	9	14	15	3	2.629638	9	9	4.734449	3.08E-11
49	9	21	19.67605	2.85514	3	9	9	5	3.12E-11
49	11	21	15.04481	1.467937	3	9	9	4.768835	2.89E-11

Iterasi 65

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
64	12	21	15	3	2.794873	9	9	5	2.75E-11
64	8	21	15	3	1	9	9	2	3.11E-11
64	8	21	15	3	1	9	9	5	2.90E-11
64	11	21	15	2.066181	2.464113	9	8.625391	5	2.90E-11
64	12	21	15	2.710577	1	9	9	3.787326	3.09E-11

Iterasi 80

Iterasi	Swarm ke	Tipe 40	Tipe 45	Tipe 60	Tipe 65	Tipe 90	Tipe 100	Tipe 120	Fitness
79	8	21	15	3	3	9	9	5	2.73E-11
79	8	21	15	3	3	9	9	2	2.92E-11
79	9	14	15	2.694129	3	9	9	5	3.04E-11
79	8	21	15	1	3	9	9	5	2.90E-11
79	8	21	15	3	1	3	9	5	3.31E-11