PRISMA 3 (2020): 186-193
PRISMA, PROSIDING SEMINAR NASIONAL MATEMATIKA
https://journal.unnes.ac.id/sju/index.php/prisma/
ISSN 2613-9189

Optimisasi Tak Linear Dua Variabel Menggunakan Algoritme Genetika Pada Software Python

Amelia Chairunnisa[a], Elis Khatizah[a*], Prapto Tri Supriyo[a]

[a] *Departemen Matematika FMIPA IPB, Jalan Meranti Kampus IPB Dramaga, Bogor 16680, Indonesia*

* *elis_khatizah@apps.ipb.ac.id*

Algoritme genetika merupakan suatu teknik optimisasi yang cara kerjanya meniru proses evolusi dan perubahan struktur genetik pada makhluk hidup. Dalam seleksi alam, individu-individu yang lebih kuat akan mempunyai peluang untuk bertahan hidup lebih besar dan individu yang lebih kuat dari orang tuanya akan dilahirkan melalui proses penyilangan serta mutasi. Pada karya ilmiah ini akan diterapkan metode algoritme genetika untuk menyelesaikan masalah optimisasi pada fungsi objektif dua variabel. Algoritme genetika diaplikasikan menggunakan bantuan *software* Python. Fungsi objektif yang digunakan terdiri atas fungsi kuadrat, fungsi trigonometri dan fungsi Rastrigin yang mengandung bentuk kuadrat dan trigonometri. Hasil penelitian menunjukkan penyelesaian masalah optimisasi menggunakan metode algoritme genetika menghasilkan galat yang relatif kecil. Pada kasus minimisasi terkait fungsi Rastrigin, algoritme genetika terbukti efektif dalam menghindari minimum lokal. Namun, pada optimisasi dengan solusi tidak tunggal, *script* Python yang dirancang penulis hanya mampu menemukan satu solusi sehingga memerlukan pengembangan yang lebih lanjut.

Kata kunci:

Algoritme Genetika, Optimisasi, Python, Fungsi Rastrigin.

1. Pendahuluan

Algoritme genetika merupakan salah satu contoh metode metaheuristik dalam menyelesaikan masalah optimisasi. Sesuai namanya, cara kerja algoritme genetika adalah meniru proses evolusi dan perubahan struktur genetik pada makhluk hidup. Dalam seleksi alam, individu-individu yang lebih kuat akan mempunyai peluang untuk bertahan hidup lebih besar dan individu yang lebih kuat dari orang tuanya akan dilahirkan melalui proses penyilangan serta mutasi. Algoritme genetika diperkenalkan oleh John Holland melalui bukunya yang berjudul "*Adaptation in Natural and Artificial System"* pada tahun 1975 yang akhirnya menjadi pegangan utama bagi peneliti algoritme genetika di dunia (Arkeman *et al.*, 2012).

Sebagai salah satu contoh metode metaheuristik, terdapat beragam permasalahan optimisasi yang dapat diselesaikan menggunakan algoritma genetika. Beberapa diantaranya adalah pemodelan algoritme genetika pada sistem penjadwalan perkuliahan (Muliadi, 2014), penerapan algoritme genetika untuk masalah *vehicle routing* (Tanujaya *et al., 2011*) dan penggunaan algoritme genetika untuk optimasi pembelian komponen komputer serta aksesorisnya (Kholik *et al.*, 2018).

Pada karya ilmiah ini akan diterapkan metode algoritme genetika untuk menyelesaikan masalah optimisasi pada fungsi objektif berupa fungsi tak linear yang mengandung dua variabel. Fungsi objektif yang digunakan terdiri atas fungsi kuadrat, fungsi trigonometri dan fungsi Rastrigin yang mengandung bentuk kuadrat dan trigonometri. Keakuratan penyelesaian optimisasi menggunakan algoritme genetika akan ditinjau berdasarkan galat yang dihasilkan.

Penerapan algoritme genetika dalam fungsi tak linear dua variabel dilakukan menggunakan *software* Python. Penggunaan *software* Python sangat membantu dalam memperoleh hasil algoritme genetika dengan iterasi yang relatif banyak. Banyaknya iterasi berpeluang menghasilkan calon solusi optimal karena terlahir dari individu-individu unggul seiring banyaknya perubahan generasi yang dilakukan. Selanjutnya, *script* Phyton yang dihasilkan diharapkan dapat dikembangkan lebih lanjut untuk masalah optimisasi fungsi objektif yang lebih kompleks.

To cite this article:
Chairunnisa, A., Khatizah, E., & Supriyo, P.T. (2020). Optimisasi Tak Linear Dua Variabel Menggunakan Algoritme Genetika Pada Software Python. *PRISMA, Prosiding Seminar Nasional Matematika 3*, 186-293

2. Pembahasan

Pada bagian ini dibahas mengenai penerapan algoritme genetika untuk menyelesaikan optimisasi fungsi taklinear dua variabel. Keakuratan hasil penyelesaian selanjutnya ditinjau berdasarkan aspek galat yang diperoleh.

2.1. Prosedur Algoritme Genetika

Prosedur algoritme genetika yang meniru proses evolusi dan perubahan struktur genetik pada makhluk hidup terdiri atas beberapa langkah, yaitu representasi kromosom, penentuan fungsi *fitness* (tujuan), seleksi kromosom, penyilangan dan mutasi (Arkeman *et al.*, 2012).

Representasi kromosom merupakan pembuatan kode (*encoding*) dari calon solusi. Salah satu jenis representasi kromosom adalah representasi *string* biner. Panjang string kromosom tergantung pada tingkat ketelitian yang dibutuhkan. Misalkan diketahui fungsi objektif $f(\mathbf{x})$ dengan $\mathbf{x}=[x_1, x_2, \ldots, x_j]$, domain variabel x_j adalah $[a_j, b_j]$ dan tingkat ketelitian yang dibutuhkan adalah *n* angka di belakang koma. Panjang kromosom yang diperlukan (m_j) dihitung menggunakan rumus sebagai berikut.

$$2^{m_j-1} < (b_j - a_j) \times 10^n \leq 2^{m_j} - 1. \quad (1)$$

Kromosom yang digunakan berbentuk biner sehingga deret biner ini perlu dikonversi ke dalam bentuk desimal menggunakan rumus berikut

$$x_j = a_j + \text{desimal}(substring) \times \frac{b_j - a_j}{2^{m_j}-1} \quad (2)$$

dengan desimal*(substring)* merepresentasikan nilai desimal dari *substring* bagi variabel keputusan. (Arkeman *et al*., 2012)

Penentuan fungsi *fitness* dalam algoritme genetika bisa dilakukan dengan cara menyamakan fungsi *fitness* dengan fungsi objektif atau memperoleh fungsi *fitness* dengan memodifikasi fungsi objektif. Jika masalahnya adalah maksimisasi, fungsi *fitness*-nya sama dengan fungsi objektif. Akan tetapi, jika masalahnya adalah masalah minimisasi, fungsi *fitness*-nya berbanding terbalik dengan fungsi objektif (Arkeman *et al*., 2012).

Kromosom-kromosom yang memiliki nilai *fitness* yang sangat baik akan memiliki peluang yang lebih besar untuk terpilih menjadi induk dan tetap bertahan pada generasi berikutnya, sedangkan kromosom-kromosom yang lebih buruk akan tergantikan oleh kromosom baru. Salah satu teknik seleksi dalam algoritme genetika adalah teknik seleksi turnamen. Seleksi turnamen merupakan teknik seleksi pada algoritme genetika yang paling populer untuk digunakan karena efisien dan mudah diimplementasikan. Pada seleksi turnamen, *n* individu dipilih secara acak dari populasi. Individu-individu yang dipilih akan bersaing. Individu dengan nilai *fitness* paling cocok akan diseleksi dan individu yang lolos seleksi akan maju ke tahap selanjutnya. Jumlah individu yang akan ambil bagian dalam setiap turnamen disebut ukuran turnamen. Ukuran turnamen yang paling sering digunakan adalah dua atau biasa disebut seleksi turnamen biner (Miller & Goldberg, 1995).

Penyilangan adalah operator utama dalam algoritme genetika. Operator ini bekerja pada sepasang kromosom induk untuk menghasilkan dua kromosom anak dengan cara menukar beberapa elemen (gen) yang dimiliki masing-masing kromosom induk. Penyilangan adalah operator primer sehingga nilai peluang penyilangan (Pc) yang digunakan biasanya cukup tinggi (0,6 – 1). Penyilangan satu titik (*one-point crossover*) merupakan salah satu metode penyilangan yang cocok digunakan untuk kromosom dengan representasi biner (Arkeman *et al*., 2012).

Langkah terakhir dari algoritme genetika adalah mutasi yang berperan mengubah struktur kromosom secara spontan. Operator mutasi bekerja pada satu kromosom, tidak pada sepasang kromosom seperti yang dilakukan operator penyilangan. Mutasi sangat diperlukan dalam pencarian solusi optimum untuk

mengembalikan gen-gen yang hilang pada generasi sebelumnya dan memunculkan gen-gen baru yang belum pernah muncul pada generasi-generasi sebelumnya. (Gen & Cheng, 1997).

2.2. Penerapan algoritme genetika pada optimisasi fungsi tak linear dua variabel

Algoritme genetika akan diterapkan pada penyelesaian optimisasi dengan fungsi ojektif yang terdiri atas fungsi kuadrat, fungsi trigonometri dan fungsi Rastrigin. Algoritme genetika pada fungsi kuadrat dan fungsi trigonometri akan dibandingkan dengan hasil eksaknya. Untuk fungsi Rastrigin, hasil dari algoritme genetika akan dibandingkan dengan global minimum yang berada pada titik **x** = **0.**

Fungsi polinom kuadrat dua variabel yang akan dioptimalkan memiliki bentuk sebagai berikut.

$$f(x,y) = x^2 + y^2 - 2x + 1, -5 \leq x \leq 5,\ -5 \leq y \leq 5 \quad (3)$$

Berdasarkan penyelesaian analitik, nilai minimum global $f(x_1, x_2)$ berada pada titik $(1,0)$ dengan nilai 0 dan nilai maksimum global berada pada titik $(-5,-5)$ dan $(-5,5)$ dengan nilai 61.

Selanjutnya akan diterapkan algoritme genetika untuk masalah optimisasi (3) dengan langkah berikut

2.2.1 Representasi Kromosom

Berdasarkan Persamaan (1), dengan tingkat ketelitian yang diinginkan adalah lima angka di belakang koma, panjang kromosom yang diperlukan untuk variabel x_1 dan x_2 adalah $m_1 = m_2 = 20$. Dengan demikian, total panjang kromosom adalah 40 bit yang merupakan gabungan dari panjang kromosom x_1 dan panjang kromosom x_2. Misalkan diambil sebanyak 20 populasi awal yang dibangkitkan secara acak dengan representasi kromosom yang disajikan pada Tabel 1.

Tabel 1. Populasi awal yang dibangkitkan secara acak dengan v_k menyatakan kromosom ke *k*

v_k	Representasi Kromosom
v_1	0111001111100011111111000001011011110000
v_2	1000101010010001110001000011010101110101
v_3	0011001000111101011101110100010010101100
v_4	1010111110101000000110111110101011111100
v_5	1111010100111001101001011001111000111001
v_6	0111100000011110110110111100011111101000
v_7	1010100001001010101110101000110011101000
v_8	0010011000001010100010010101100110111100
v_9	0000101110000100000110100100001011110001
v_{10}	0001001110101100100000001001000110011101
v_{11}	1010100101100010011111010001000101111110
v_{12}	1001101011101010000100010011100100000110
v_{13}	0001000011011101001101101001111110011010
v_{14}	0101011001011100011010100000011000111011
v_{15}	0100101010111111100100010110101000100010
v_{16}	1011100010001111001101101001111011011111
v_{17}	0001100001110011101100111100001110111111
v_{18}	0101111100110010101001001101000010000111
v_{19}	1000100010100110100101000111101100101001
v_{20}	0011001111000011010101000000010010100111

Berdasarkan Persamaan (2), nilai x_1 dan x_2 pada Tabel 1 dikonversikan ke dalam desimal sehingga diperoleh Tabel 2 sebagai berikut.

Tabel 2. Konversi nilai x_1 dan x_2 ke desimal

v_k	x_1	x_2	$f(x_1, x_2)$
v_1	-0.47302768042343146	2.5560069618291488	8.70298
v_2	0.41286984717354525	-2.3694871611472714	5.95919
v_3	-3.0374985098824596	-0.4573397229573466	16.51055
v_4	1.8615883460887392	2.448699425410676	6.73846
v_5	4.579114512552751	-1.4887108695133873	15.02632
v_6	-0.30779391078368246	2.3630593901246932	7.29437
v_7	1.573902677443197	1.5940156879574658	2.87025
v_8	-3.514021410008821	0.8440836373173113	21.08887
v_9	-4.550160932694371	1.4134372839329563	32.80209
v_{10}	-4.231490355959278	-4.644498486040579	48.93986
v_{11}	1.6165891805545618	3.167713325227094	10.41459
v_{12}	1.0513458741625543	-4.235781894475837	17.94448
v_{13}	-4.341248837708319	-0.8603437999189376	29.26913
v_{14}	-1.626526476408459	1.2652170803232963	8.49942
v_{15}	-2.0801420976086593	-4.11588584507546	26.42779
v_{16}	2.2093555539660965	-0.8621271725913742	2.2058
v_{17}	-4.044846577498033	-2.6471020194072907	32.45763
v_{18}	-1.2813341916410366	-1.990897169968767	9.16816
v_{19}	0.33792051116992106	-2.1993133538373506	5.27533
v_{20}	-2.978008249290704	-2.4886393438714447	22.01788

Berikut adalah *script* dari representasi kromosom biner menggunakan *software* Python.

```
#Menentukan panjang kromosom
for x1 in range(100):
    if math.pow(2, x1 - 1) < (b - (a)) * 10 ** 5 and (b - (a)) *
10 ** 5 <= math.pow(2,x1) - 1:
        chrom.append(x1)
for x2 in range(100):
    if math.pow(2, x2 - 1) < (d - (c)) * 10 ** 5 and (d - (c)) *
10 ** 5 <= math.pow(2, x2) - 1:
        chrom.append(x2)
#Membangkitkan bilangan biner secara acak
randBinList = lambda n: [randint(0, 1) for b in range(1, n + 1)]
list1 = randBinList(chrom[0])
list2 = randBinList(chrom[1])
list3 = list1 + list2
#konversi bilangan biner ke desimal
dec1 = int(str1, 2)
dec2 = int(str2, 2)
a = -5  # batas bawah x1
b = 5  # batas atas x1
```

```
c = -5  # batas bawah x2
d = 5  # batas atas x2
x1 =  a +dec1 * ((b-(-a)) / (math.pow(2, chrom[0]) - 1))
x2 =  c +dec2 * ((d-(-c)) / (math.pow(2, chrom[1]) - 1))
```

2.2.2 Seleksi Kromosom

Proses penyeleksian kromosom dimulai dengan mengambil dua bilangan acak dari populasi yang disajikan pada Tabel 2. Misalkan bilangan acak yang diperoleh adalah 20 dan 11. Selanjutnya, kromosom 20 dan kromosom 11 dibandingkan nilai *fitness*-nya. Pada kasus minimisasi, kromosom yang diseleksi adalah kromosom yang memiliki nilai *fitness* lebih kecil. Nilai *fitness* pada kromosom 20 dan 11 berturut-turut adalah 22.01788 dan 10.41459. Kromosom 11 memiliki nilai *fitness* lebih kecil sehingga kromosom 11 akan maju ke tahap berikutnya. Sebaliknya, pada kasus maksimisasi kromosom yang lulus seleksi adalah kromosom yang memiliki nilai *fitness* lebih besar sehingga antara kromosom 20 dan 11 yang lulus seleksi adalah kromosom 20. Misalkan pada kasus minimisasi, yang lolos seleksi berturut-turut adalah kromosom 11 dan kromosom 4, maka kedua kromosom tersebut akan lanjut ke tahap penyilangan. Berikut adalah *script* Python untuk seleksi turnamen

```
#Tournament Selection
        while n < Population:
            pick = round(random.random(), 5)
            if  pick < 0.88:
                from random import sample
                i = [i for i in range(20)]
                picks = sample(i, 2)
                fitness1          =          max(np_fitness[picks[0]],
np_fitness[picks[1]])
                if fitness1 == np_fitness[picks[0]]:
                    parent1 = BinNum[picks[0]]
                else:
                    parent1 = BinNum[picks[1]]

                picks = sample(i, 2)
                fitness2          =          max(np_fitness[picks[0]],
np_fitness[picks[1]])
                if fitness2 == np_fitness[picks[0]]:
                    parent2 = BinNum[picks[0]]
                else:
                    parent2 = BinNum[picks[1]]
                n+=1
```

2.2.3 Penyilangan (Crossover)

Metode *crossover* yang digunakan adalah penyilangan satu titik (*one-point crossover*). Metode ini akan menyeleksi secara acak satu titik potong dan saling menukarkan bagian dari dua kromosom induk yang telah diseleksi untuk menghasilkan keturunan. Peluang penyilangan (p_c) yang ditetapkan penulis adalah 0.88. Dengan panjang kromosom sebesar 40, bilangan integer yang digunakan sebagai titik potong akan dibangkitkan secara acak pada *range* [1,39]. Misalkan bilangan acak yang dibangkitkan adalah 35, maka dua kromosom akan dipotong setelah gen ke 35 seperti yang ditunjukkan pada ilustrasi berikut.

v_{11} = **10101001011000100111110100010001011 11110**
v_4 = 10101111101010000001101111101010111 11100

$$\downarrow$$

$$v_{11}' = \mathbf{10101001011000100111110100010001011}\ \ 11100$$
$$v_4' = 10101111101010000001101111101010111\ \ \mathbf{11110}$$

Berikut adalah *script* Python dari proses penyilangan

```
c = random.randint(0, chrom[0] + chrom[1])
  parentx.append(parent1[:c] + parent2[c:])
  parentx.append(parent2[:c] + parent1[c:])
```

2.2.4 Mutasi

Mutasi akan mengganti satu atau lebih gen dengan peluang yang besarnya sama dengan nilai peluang mutasi (p_m) yang telah ditentukan. Peluang mutasi ditetapkan bernilai 0.1. Ilustrasi mutasi ditunjukkan pada sebagai berikut

$$v_{11}' = 1010100101110001001111101000100010111111\mathbf{1}0,$$
$$\downarrow$$
$$v_{11}'' = 1010100101111001001111101000100010111111\mathbf{0}0.$$

Berikut adalah *script* Python pada langkah mutasi

```
for a in range(Population - 1):
        for b in range((chrom[0] + chrom[1]) - 1):
            if random.random() < mutationrate:
                if np_parentx[a,b]==0:
                    np_parentx[a,b]=1
                else:
                    np_parentx[a,b]=0
```

Setelah tahapan mutasi selesai, akan diperoleh kembali populasi sebanyak 20 dengan representasi kromosom yang menggambarkan generasi pertama. Selanjutnya proses yang sama dilakukan sebanyak 5000 iterasi sehingga algoritme genetika akan berakhir setelah generasi ke 5000 dengan jumlah populasi 20, panjang kromosom 40, peluang penyilangan 0.88 dan peluang mutasi 0.1. Hasil yang didapatkan setelah 5000 iterasi pada kasus minimisasi Persamaan (3) adalah $f(x_1, x_2) = 0.0$, dengan $x_1 = 0.99927$ dan $x_2 = 0.00095$ yang ditemukan pada generasi ke 710. Hasil maksimisasi Persamaan (3) ditemukan pada generasi ke 789 dengan nilai $f(x_1, x_2) = 60.99847$, $x_1 = -5.0$ dan $x_2 = -4.99985$. Perlu diketahui, hasil optimisasi tidak selalu ditemukan pada generasi yang sama untuk setiap *running* program. Selanjutnya, dapat diperhatikan bahwa pada hasil maksimisasi, algoritma genetika hanya mampu menghasilkan nilai maksimum yang terletak di satu titik.

Dengan langkah yang sama, algoritma genetika diterapkan untuk optimasi terkait fungsi trigonometri berikut.

$$f(x_1, x_2) = \sin(x_1) + \sin(x_2), 0 \le x_1 \le \pi, 0 \le x_2 \le \pi \quad (4)$$

Berdasarkan penyelesaian analitik, minimum global $f(x_1, x_2)$ terdapat di empat titik, yaitu $(0,0)$, $(0, \pi)$, $(\pi, 0)$ dan (π, π) dengan nilai 0 sedangkan maksimum global $f(x_1, x_2)$ berada pada titik $\left(\frac{\pi}{2}, \frac{\pi}{2}\right)$ dengan nilai 2.

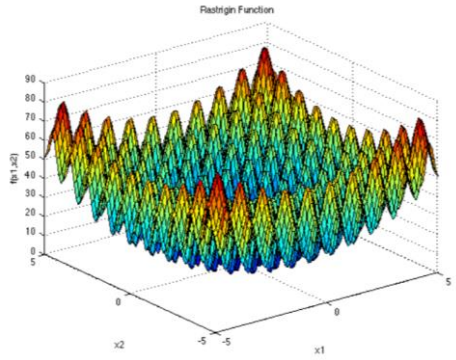
Menggunakan algoritme genetika yang berakhir setelah generasi ke 5000 dengan jumlah populasi 20, panjang kromosom 34, peluang penyilangan 0.88 dan peluang mutasi 0.1, diperoleh minimisasi Persamaan (4) dengan nilai $f(x_1, x_2) = 0.00024$, $x_1 = 0.00005$, dan $x_2 = 0.00019$. Nilai x_1 dan x_2 ditemukan pada generasi ke 3820. Untuk maksimisasi Persamaan (4) ditemukan pada generasi ke 75

dengan nilai $f(x_1, x_2) = 2.0$, $x_1 = 1.57025$ dan $x_2 = 1.57109$. Sebagaimana kasus maksimisasi Persamaan (3), pada kasus minimisasi Persamaan (4) ini, algoritma genetika hanya mampu menghasilkan nilai minimum yang terletak di satu titik.

Selanjutnya, algoritme genetika diterapkan untuk optimisasi fungsi Rastrigin dua variabel berikut.

$$\text{minimumkan } f(x_1, x_2) = 20 + {x_1}^2 + {x_2}^2 - 10(\cos(2\pi x_1) + \cos(2\pi x_2))$$
$$x_1 \in [-5.12, 5.12]$$
$$x_2 \in [-5.12, 5.12] \quad (5)$$

Nilai minimum global dari fungsi ini adalah $f(\mathbf{x}) = \mathbf{0}$. Sulit untuk menemukan minimum global karena fungsi ini memiliki banyak minimum lokal (Neydorf et al, 2016). Grafik fungsi Rastrigin disajikan pada Gambar 1 berikut



Gambar 1. Grafik fungsi Rastrigin dua variabel

Menggunakan algoritme genetika sebanyak 5000 generasi, jumlah populasi 20, panjang kromosom 40, peluang penyilangan 0.88, dan peluang mutasi 0.1 diperoleh nilai minimum pada generasi ke 4589 sebagai berikut $f(x_1, x_2) = 0.00001$, $x_1 = 0.00007$, dan $x_2 = 0.00021$.

2.3. Galat Metode Algoritme Genetika

Galat penyelesaian optimisasi fungsi tak linear dua variabel menggunakan metode algoritme genetika disajikan dalam Tabel 3 berikut

Tabel 3. Galat penyelesaian optimisasi fungsi tak linear dua variabel menggunakan metode algoritme genetika

Fungsi Objektif		Galat		
		$f(x_1, x_2)$	x_1	x_2
Fungsi Kuadrat	Minimisasi	0	0.00073	0.00005
	Maksimisasi	0.00153	0	0.00015
Fungsi Trigonometri	Minimisasi	0.00024	0.00005	0.00019
	Maksimisasi	0	0.00054	0.00030
Fungsi Rastrigin	Minimisasi	0.00001	0.00007	0.00021

Berdasarkan Tabel 3, dapat dilihat bahwa secara umum galat yang dihasilkan dalam penyelesaian optimisasi terkait fungsi tak linear dua variabel relatif kecil. Dengan demikian, dapat disimpulkan bahwa algoritme genetika cukup tangguh (*robust*) untuk optimisasi fungsi tak linear dua variabel khususnya fungsi Rastrigin yang sulit ditemukan nilai minimum globalnya.

3. Simpulan

Algoritme genetika dapat digunakan untuk menyelesaikan masalah optimisasi fungsi objektif berupa fungsi taklinear dua variabel menggunakan *software* Python. Berdasarkan penyelesaian masalah

optimisasi pada fungsi kuadrat, fungsi trigonometri dan fungsi Rastrigin, diperoleh galat yang relatif kecil. Dengan demikian, algoritme genetika cukup tangguh (*robust*) untuk optimisasi fungsi tak linear dua variabel baik yang memiliki penyelesaian eksak maupun yang tidak bisa diselesaikan secara analitik. Berdasarkan penyelesaian optimisasi fungsi Rastrigin, algoritme genetika juga terbukti efektif dalam menghindari minimum lokal sehingga nilai minimum global dapat ditemukan meski fungsi memiliki banyak minimum lokal. Namun, pada optimisasi dengan solusi tidak unik, *script* Python yang dirancang penulis hanya mampu menemukan satu solusi sehingga perlu pengembangan lebih lanjut.

Daftar Pustaka

Arkeman, Y., Seminar, K.B. & Gunawan, H. (2012). *Algoritme Genetika Teori dan Aplikasinya untuk Bisnis dan Industri.* Bogor(ID):PT Penerbit IPB Press.

Neydorf, R., Chernogorov, I., Yarakhmedov, V.P.O., & Goncharova, Y. (2016). Study of Search Optimization Opportunities of Heuristic Algorithms for Solving Multi-Extermal Problems. *The tenth International Conference on Advanced Engineering Computing and Applications in Sciences*,10: 48-51

Gen, M. & Cheng, R.(1997). *Genetic Algorithms and Engineering Design.* Hoboken(US): John Wiley & Son, Inc.

Kholik, E.E., Devianto, W.K., Sholihah, N., & Santosa, Y.M. (2018). Optimasi Pembelian Komponen Komputer dan Aksesorisnya Menggunakan Algoritma Genetika. *SNATI UII.*

Miller, B. L. & Goldberg, D. E. 1995.Genetic Algorithm, Tournamen Selection, and the Effects of Noise. *Complex System, 9*, 193-212.

Muliadi (2014). Pemodelan Algoritma Genetika Pada Sistem Perkuliahan Prodi Ilmu Komputer Universitas Lambung Mangkurat. *Kumpulan Jurnal Ilmu Komputer,1*(1), 67-78.

Tanujaya, W., Dewi, D.R.S, & Endah, D., (2011). Penerapan Algoritma Genetika Untuk Penyelesaian Masalah *Vehicle Routing* di PT MIF , *WIDYA TEKNIK, 10*(1), 92-102.