IMPLEMENTASI SELF-ADAPTIVE DIFFERENTIAL EVOLUTION WITH NEIGHBORHOOD SEARCH (SANSDE) UNTUK OPTIMASI SISTEM POMPA AIR

Isnani Pramusinto[1]
[1]*Jurusan Teknik Informatika, Fakultas Teknologi Industri,Universitas Islam Indonesia*
Jl. Kaliurang Km. 14 Yogyakarta 55501
E-mail: isnanipramusinto@yahoo.com

ABSTRAKS
Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE) merupakan salah satu varian dari algoritma Differential Evolution (DE) yang dikembangkan dengan mengadopsi dua varian DE yang telah ada sebelumnya, yaitu Self-adaptive Differential Evolution (SaDE) dan Differential Evolution with Neighborhood Search (NSDE). Dalam pengujian terhadap fungsi – fungsi klasik, SaNSDE terbukti mampu menunjukkan performa yang jauh mengungguli algoritma SaDE dan NSDE. Di dalam penelitian ini SaNSDE digunakan untuk menyelesaikan salah satu permasalahan klasik dalam bidang mekanika fluida, yaitu optimasi sistem pompa air. Sistem pompa air merepresentasikan masalah optimasi non-linear yang sulit, dengan equality constraint. Performa SaNSDE dibandingkan dengan performa algoritma DE dengan 10 jenis strategi mutasinya, yang mana telah digunakan untuk menyelesaikan masalah yang sama pada penelitian terdahulu. Hasil penelitian ini menunjukkan bahwa SaNSDE mampu menemukan solusi optimal global dari formula optimasi sistem pompa air yang diberikan. Hasil penelitian juga mengindikasikan bahwa SaNSDE lebih baik daripada DE.

Kata Kunci: Differential Evolution, SaNSDE, optimasi, sistem pompa air

1. PENDAHULUAN

Perkembangan komputasi saat ini telah mengalami percepatan yang luar biasa. Berbagai teknik komputasi untuk mendapatkan solusi dan performa yang memuaskan terus bermunculan sebagai jawaban atas semakin banyaknya masalah optimasi nyata dalam kehidupan sehari – hari yang harus dipecahkan. Termasuk dalam salah satu bidang komputasi, *Evolutionary Computation.*

Ada beberapa algoritma yang termasuk dalam rumpun *Evolutionary Computation* yang selanjutnya dikenal dengan istilah *Evolutionary Algorithm.* Algoritma – algoritma tersebut antara lain *Genetic Algorithm, Genetic Programming, Evolutionary Strategies, Differential Evolution, Evolutionary Programming,* dan *Grammatical Evolution.* Di samping itu, menurut Dasgupta dan Michalewicz, masih banyak lagi sistem *hybrid*, yang menggabungkan berbagai fitur – fitur atau karakteristik yang dimiliki oleh algoritma – algoritma yang sudah disebutkan di atas, sehingga sulit untuk diklasifikasikan (Babu dan Angira, 2003).

Salah satu yang terbaik dari algoritma tersebut di atas adalah *Differential Evolution* (DE) yang dikenalkan oleh Storn dan Price (1995). Beberapa alasan yang membuat algoritma ini banyak mendapat pujian adalah karena implementasinya yang mudah dan kecepatan konvergensinya. Algoritma ini sukses digunakan untuk menyelesaikan masalah optimasi dalam berbagai bidang seperti klastering, desain filter digital, optimasi fungsi linier, tehnik kimia, dan optimasi *multi-objective* (Pant dkk., 2008).

DE mengalami perkembangan yang cukup signifikan sejak pertama kali dikenalkan. Berbagai varian dari DE bermunculan sebagai usaha untuk meningkatkan performa dari algoritma ini. Dan salah satunya adalah SaNSDE (*Self-adaptive Differential Evolution with Neighborhood Search*) yang dikenalkan oleh Yang dkk (2008).

Pada penelitian kali ini, SaNSDE akan diimplementasikan ke dalam program komputer untuk menyelesaikan salah satu masalah klasik dalam bidang mekanika fluida yaitu optimasi sistem pompa air. Masalah ini merepresentasikan masalah optimasi *non-linear* yang sulit dengan *equality constraint*. (Onwubolu dan Babu, 2004)

2. DASAR TEORI

2.1 Differential Evolution

Langkah – langkah penyelesaian masalah oleh DE secara umum sama dengan algoritma – algoritma evolusioner yang lain (Karaboga, 2004), yaitu:

```
Inisialisasi
Evaluasi
Repeat
    Mutasi
    Rekombinasi
    Evaluasi
    Seleksi
Until(kriteria berhenti tercapai)
```

Di dalam DE, individu – individu adalah nilai *real* yang merupakan nilai sebenarnya dari solusi yang dicari. Nilai *real* ini selanjutnya bisa disebut dengan istilah vektor.

DE membangkitkan suatu vektor (individu) baru dengan melibatkan tiga vektor (individu) sebagai orang tua. Pembangkitan vektor baru dilakukan dengan menambahkan selisih antara dua vektor (orang tua ke-1 dan ke-2) kepada vektor lainnya (orang tua ke-3). Proses inilah yang disebut mutasi. Ada dua skema mutasi yang diusulkan oleh Kenneth dan Price pada saat *paper* pertama mereka dipublikasikan. Salah satu skema ditunjukkan pada Persamaan (1).

Untuk setiap vektor $x_{i,G}, i = 0,1,2,...,NP-1$, suatu vektor baru v dibangkitkan berdasarkan rumus:

$$v_i = x_{r3,G} + F.\left(x_{r1,G} - x_{r2,G}\right) \quad (1)$$

di mana $r_1, r_2, r_3 \in [0, NP-1]$ adalah bilangan integer yang berbeda satu sama lain dan $F > 0$. Ketiga bilangan r_1, r_2, r_3 dipilih secara acak dalam interval $[0, NP-1]$. Sedangkan F disebut dengan faktor skala yang berupa bilangan *real* yang merupakan konstanta yang mengontrol penguatan *differential variation* $\left(x_{r1,G} - x_{r2,G}\right)$ (Storn, 1996). Faktor skala ini lebih berkaitan dengan kecepatan konvergensi (Qin dkk., 2006).

Dalam algoritma DE, sampai saat ini dikenal minimal 10 strategi mutasi dasar. Di dalam literatur DE, strategi ini dinyatakan dengan aturan penulisan *DE/a/b/c* di mana *a* menyatakan cara menghasilkan vektor mutan, *b* menyatakan banyaknya "selisih" yang dilibatkan dalam menghasilkan vektor mutan dan *c* menyatakan tipe rekombinasi (Zaharie, 2007).

Setelah didapatkan vektor v dari proses mutasi, maka proses selanjutnya adalah rekombinasi (*crossover*). Sampai saat ini dikenal dua metode rekombinasi dalam DE yaitu binomial dan eksponensial. Rekombinasi eksponensial adalah metode yang dikenalkan oleh Kenneth dan Price saat mereka mengenalkan DE pertama kali. Namun rekombinasi binomial justru lebih banyak digunakan dalam aplikasinya saat ini. (Zaharie, 2007)

Dalam rekombinasi, dikenal parameter CR yang mempunyai peranan penting dalam algoritma DE. CR lebih sensitif kepada kompleksitas masalah yang diselesaikan. Penentuan nilai CR yang tepat akan menghasilkan performa DE yang bagus, namun sebaliknya pemilihan nilai CR yang salah akan membawa DE ke dalam performa yang buruk (Qin dkk., 2006).

Berikut ini kedua tipe rekombinasi dalam DE, binomial dan eksponensial. (Suyanto, 2008) (Karaboga, 2004)

1. Binomial

$$u_i = \begin{cases} v_i & if\left(rand(0..1) \le CR\right) or\ j = rand(1..D) \\ x_i & if\left(rand(0..1) > CR\right) and\ j \ne rand(1..D) \end{cases} \quad (2)$$

2. Eksponensial

$$u_i = \begin{cases} v_i & if\ j = \langle n \rangle_D, \langle n+1 \rangle_D, ..., \langle n+L-1 \rangle_D \\ x_i & otherwise \end{cases} \quad (3)$$

2.2 SaNSDE

SaNSDE diusulkan oleh Yang Z. dkk (2008) pada paper yang dipublikasikan pada *IEEE Congress on Evolutionary Computation (CEC 2008)*. Ide dari SaNSDE adalah menggabungkan dua varian algoritma DE yang sudah ada sebelumnya, SaDE (Qin dan Suganthan, 2005) dan NSDE (Yang dkk, 2008). Secara umum berikut ini karakteristik SaNSDE yang membedakannya dengan varian DE yang lain:

1. Mutasi pada SaNSDE mengadopsi secara utuh strategi mutasi yang digunakan oleh SaDE, yaitu:

$$v_i = \begin{cases} strategi\ 1 & if\ U_i(0,1) < p \\ strategi\ 5 & otherwise \end{cases} \quad (4)$$

Di mana mutasi strategi 1 ➔ $v_i = x_{r1} + F.\left(x_{r2} - x_{r3}\right)$ sedangkan mutasi strategi 5 ➔ $v_i = x_{r3} + F.\left(x_{best} + x_{r1} - x_{r2} - x_{r3}\right)$. Di sini p diberi nilai awal 0,5. Setelah evaluasi dari semua *offspring*, jumlah *offspring* yang sukses masuk generasi berikutnya dengan menggunakan strategi 1 dan 5 disimpan dalam ns_1 dan ns_5. Sedangkan jumlah *offspring* yang gagal masuk generasi berikutnya dengan menggunakan strategi 1 dan 5 disimpan dalam nf_1 dan nf_5. Kemudian angka – angka tersebut dioperasikan setelah sampai pada jumlah generasi tertentu, yang disebut dengan *learning period.* Dalam SaDE, *learning period* adalah 50. Kemudian probabilitas p di*update* dengan rumus:

$$p = \frac{ns_1\left(ns_5 + nf_5\right)}{ns_5\left(ns_1 + nf_1\right) + ns_1\left(ns_5 + nf_5\right)} \quad (5)$$

2. Faktor skala F mengadopsi metode yang digunakan NSDE dengan beberapa modifikasi.

$$F_i = \begin{cases} N_i(0.5,0.3) & if\ U_i(0,1) < f_p \\ \delta_i & otherwise \end{cases} \quad (6)$$

di mana f_p *self-adapted* sebagaimana p dalam Persamaan (5).

3. Proses rekombinasi menggunakan metode yang mirip dengan metode yang digunakan oleh SaDE. Namun setiap kali nilai CR yang sukses masuk generasi berikutnya

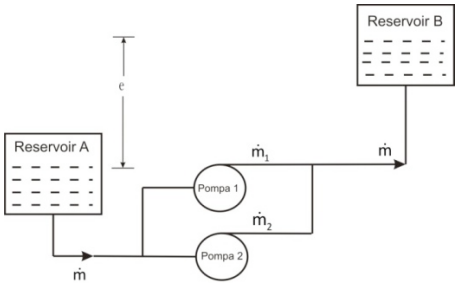
disimpan dalam *array* CR_{rec}, maka kenaikan nilai *fitness* juga disimpan dalam *array* Δf_{rec}, dengan $\Delta f_{rec}(k) = f(k) - f_{new}(k)$. Dan persamaan rekombinasi pada SaDE (Qin dan Suganthan, 2005) diubah menjadi:

$$CRm = \sum_{k=1}^{|CR_{rec}|} w_k * CR_{rec}(k) \qquad (7)$$

$$w_k = \Delta f_{rec}(k) / \left(\sum_{k=1}^{|\Delta f_{rec}|} \Delta f_{rec}(k) \right) \qquad (8)$$

2.3 Sistem Pompa Air

Desain sistem pompa air yang banyak terdapat di dalam literatur adalah sistem yang terdiri dari dua pompa paralel yang digunakan untuk mengalirkan air dari tempat yang lebih rendah baik itu reservoir, sumur, maupun sungai, menuju tempat yang lebih tinggi. Perhatikan desain sistem yang dimaksud pada Gambar 1.



Gambar 1. Sistem pompa air dengan 2 pompa paralel

Perbedaan ketinggian antara reservoir A dan B yang diistilahkan dengan elevasi adalah sebesar e meter. Karakteristik pompa 1 dan pompa 2 yang dituliskan dalam persamaan perbedaan tekanan (*pressure difference*) Δp dan laju aliran $\dot{m}$, masing – masing adalah

$$\Delta p = A_0 - A_1 \dot{m}_1 - A_2 \dot{m}_1^{\,2} \qquad (9)$$

dan

$$\Delta p = B_0 - B_1 \dot{m}_2 - B_2 \dot{m}_2^{\,2} \qquad (10)$$

di mana tekanan dalam kPa dan laju aliran dalam kg/s. Perbedaan tekanan yang disebabkan karena elevasi dan gesekan dalam pipa adalah

$$\Delta p = H + C(\dot{m})^2 \qquad (11)$$

Dengan H adalah tekanan yang disebabkan karena elevasi e dalam kPa, dan $C(\dot{m})^2$ adalah tekanan yang disebabkan karena gesekan dalam kPa. H didapatkan dari rumus tekanan $\rho.g.h$, dengan ρ = massa jenis air, g = grafitasi, h = tinggi/elevasi. Karena $\rho = 1000 kg/m^3$, $g = 10 m/s^2$, $h = (e)m$, maka

$$H = 1000 kg/m^3 . 10 m/s^2 . (e)m$$
$$H = (1000.e)Pa$$
$$H = (10.e)kPa \qquad (12)$$

Total laju aliran $\dot{m}$ adalah jumlah dari laju aliran di dua pompa $\dot{m}_1$ dan $\dot{m}_2$, sehingga

$$\dot{m} = \dot{m}_1 + \dot{m}_2 \qquad (13)$$

Kemudian dari Persamaan (9), (10), (11), (12), (13), formula optimasi untuk sistem pompa air pada Gambar 1. adalah

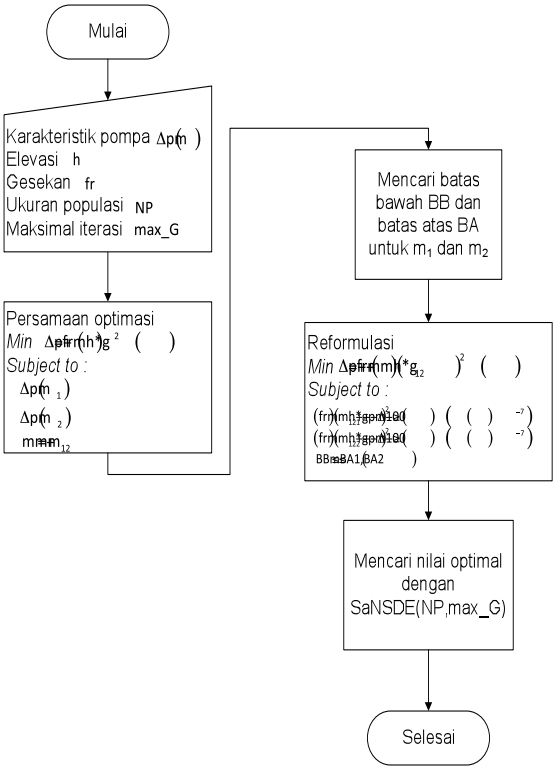
$$Min. \quad f = \Delta p$$
$$subject\ to:$$
$$\Delta p = H + C(\dot{m})$$
$$\Delta p = A_0 - A_1 \dot{m}_1 - A_2 \dot{m}_1^{\,2}$$
$$\Delta p = B_0 - B_1 \dot{m}_2 - B_2 \dot{m}_2^{\,2}$$
$$\dot{m} = \dot{m}_1 + \dot{m}_2 \qquad (14)$$

Tujuan dari optimasi ini adalah untuk mencari nilai laju aliran $\dot{m}_1$ dan $\dot{m}_2$ yang optimal agar perbedaan tekanan (*pressure difference*) Δp dalam sistem menjadi minimal. (Jaluria, 2007),(Babu dan Angira, 2003).

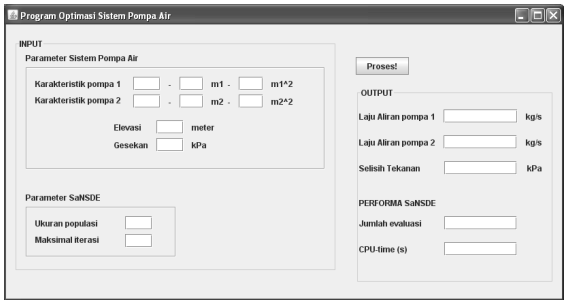
3. IMPLEMENTASI

Aplikasi ini dibangun dengan menggunakan bahasa pemrograman Java dengan editor NetBeans 6.7.1 dan kompiler Java Development Kit 1.6. Urutan proses penyelesaian masalah pompa air ini digambarkan dalam diagram alir pada Gambar 2.

Hasil pengkodean aplikasi ini dapat dilihat pada Gambar 3. Di sana terlihat bahwa aplikasi menyediakan dua kelompok kotak input. Kelompok yang pertama (kiri atas dalam gambar) merupakan kotak input yang digunakan untuk menginputkan parameter dari sistem pompa air. Parameter – parameter tersebut adalah karakteristik pompa air, elevasi dan gesekan. Kelompok yang kedua (kiri bawah dalam gambar) merupakan kotak – kotak input yang digunakan menginputkan parameter dari algoritma SaNSDE, berupa ukuran populasi tiap generasi dan jumlah maksimal iterasi.



Gambar 2. Diagram alir penyelesaian masalah



Gambar 3. Tatap muka aplikasi

4. PENGUJIAN

4.1 Data pengujian

Data yang digunakan untuk pengujian aplikasi ini adalah formula optimasi dari sebuah sistem pompa air yang diselesaikan oleh Babu dan Angira (2003), yaitu:

$$
\begin{aligned}
&Min. \quad f = \Delta p \\
&subject\ to: \\
&\Delta p = 250 + 30\dot{m}_1 - 6\dot{m}_1^{\,2} \\
&\Delta p = 300 + 20\dot{m}_2 - 6\dot{m}_2^{\,2} \\
&\Delta p = 150 + 0{,}5\left(\dot{m}_1 + \dot{m}_2\right)^2 \\
&0 \le \dot{m} \le \left(9{,}422;5{,}903\right) \qquad (15)
\end{aligned}
$$

Penggunaan formula Persamaan (15) untuk pengujian dikarenakan formula tersebut sudah pernah diselesaikan dan sudah diketahui berapa nilai optimal yang diperoleh sehingga bisa digunakan untuk melihat performa SaNSDE. Dua penelitian sebelumnya, yang pertama Ryoo dan Sahindis dalam Babu dan Angira (2003), menyelesaikan formula ini dengan algoritma Branch and Reduce. Penelitian yang kedua adalah Babu dan Angira (2003) yang menyelesaikan formula ini dengan algoritma *Differential Evolution*. Kedua penelitian mampu menemukan nilai optimal global dengan nilai laju aliran di dua pompa adalah 6,293430 dan 3,821839 serta nilai fungsi obyektif dalam hal ini selisih tekanan adalah 201,159334.

4.2 Hasil pengujian

Pengujian dilakukan dengan mengubah – ubah nilai parameter ukuran populasi. Dalam pengujian ini dilakukan sepuluh kali eksekusi untuk setiap ukuran populasi, dimulai dengan ukuran populasi sebesar 10, kemudian 11, 15, 20, 30 dan 40. Tabel 1 menunjukkan hasil lengkap dari pengujian untuk setiap ukuran populasi dengan NP = ukuran populasi, KGO = berapa kali mencapai optimal global dari sepuluh kali eksekusi, REF = rata – rata banyaknya evaluasi fungsi obyektif yang dilakukan, dan CPU-time = rata – rata waktu yang diperlukan untuk mencapai optimal global.

Pengujian ini dilakukan dengan dua jenis tingkat akurasi, 10^{-6} dan 10^{-7}. Tingkat akurasi di sini adalah selisih individu terbaik dengan individu terburuk yang digunakan sebagai kriteria pemberhenti iterasi. Dengan kata lain, jika selisih individu terbaik dengan individu terburuk dalam suatu generasi lebih besar dari tingkat akurasi, maka iterasi akan terus berlanjut.

Tabel 1. Hasil pengujian dengan tingkat akurasi 10^{-6}

NP	KGO	REF	CPU-time
10	5	1215	0,0112
11	7	1426	0,0186
15	9	1485	0,0156
20	10	1852	0,0172
30	10	2721	0,0299
40	10	3624	0,0361

Tabel 2. Hasil pengujian dengan tingkat akurasi 10^{-7}

NP	KGO	REF	CPU-time
10	3	2949	0,0373
11	7	3091	0,0298
15	10	1758	0,0142
20	10	2048	0,0234
30	10	3291	0,0327
40	10	4345	0,0453

Tabel 1 dan 2 memperlihatkan bahwa pada pengujian dengan tingkat akurasi 10^{-6}, untuk ukuran populasi sebesar 10, 11 dan 15, program ternyata tidak selalu bisa mencapai optimal global. Dalam tabel juga terlihat bahwa dengan ukuran populasi 20,

30 dan 40, program mampu mencapai optimal global.

Pada pengujian dengan tingkat akurasi 10^{-7}, ukuran populasi 10 dan 11 tidak mampu selalu mencapai optimal global. Bahkan untuk ukuran populasi 10, program hanya bisa mencapai global optimal sebanyak tiga kali. Angka rata – rata evaluasi fungsi obyektif yang besar pada ukuran populasi 10 di atas disebabkan oleh eksekusi yang gagal mencapai optimal global. Pada eksekusi yang berhasil mencapai optimal global rata – rata evaluasi fungsi obyektif sebesar 1286. Begitu juga yang terjadi pada pengujian ukuran populasi 11.

Hasil pengujian dengan tingkat akurasi 10^{-7} berikutnya dengan ukuran populasi sebesar 15, 20, 30 dan 40, tiap pengujian dengan sempurna mampu mencapai global optimal sepuluh kali dari sepuluh kali eksekusi. Dari keempat pengujian ini, terlihat pada tabel bahwa pengujian dengan ukuran populasi 15 memberikan performa yang paling baik dengan rata – rata evaluasi fungsi obyektif dan waktu yang paling minimal.

4.3 Perbandingan performa dengan DE

Babu dan Angira (2003) telah menyelesaikan masalah optimasi sistem pompa air ini dengan algoritma *Differential Evolution* yang masih asli. Mereka menggunakan sepuluh strategi mutasi yang berbeda dengan masing – masing strategi mendapatkan sepuluh kali eksekusi. Parameter yang digunakan adalah ukuran populasi NP = 20, probabilitas *crossover* CR = 0.5 dan faktor skala F = 0.8.

Tabel 3 dan Tabel 4 menunjukkan performa yang ditunjukkan oleh sepuluh strategi DE dibandingkan dengan SaNSDE dengan tingkat akurasi yang berbeda. Tabel 6.3 untuk tingkat akurasi 10^{-6} dan tabel 6.4 untuk tingkat akurasi 10^{-7}.

Tabel 3. Performa DE dan SaNSDE dengan akurasi 10^{-6}

No.	Algoritma	REF	CPU-time	KGO
1.	DE/rand/1/bin	3134	0,1319	10
2.	DE/best/1/bin	2406	0,0879	10
3.	DE/best/2/bin	4444	0,1758	10
4.	DE/rand/2/bin	4644	0,1758	10
5.	DE/randToBest/1/bin	2364	0,0879	10
6.	DE/rand/1/exp	3214	0,1154	10
7.	DE/best/1/exp	2372	0,0934	10
8.	DE/best/2/exp	4506	0,1648	10
9.	DE/rand/2/exp	4652	0,1868	10
10.	DE/randToBest/1/exp	2162	0,0714	10
11.	SaNSDE (NP = 15)	1485	0,0156	9
12.	SaNSDE (NP = 20)	1852	0,0172	10

Tabel 4. Performa DE dan SaNSDE dengan akurasi 10^{-7}

No.	Algoritma	REF	CPU-time	KGO
1.	DE/rand/1/bin	3524	0,1374	10
2.	DE/best/1/bin	2624	0,1099	10
3.	DE/best/2/bin	5016	0,1868	10
4.	DE/rand/2/bin	5158	0,2033	10
5.	DE/randToBest/1/bin	4146	0,1648	9
6.	DE/rand/1/exp	3542	0,1319	10
7.	DE/best/1/exp	2636	0,0989	10
8.	DE/best/2/exp	5048	0,1923	10
9.	DE/rand/2/exp	5206	0,1978	10
10.	DE/randToBest/1/exp	2484	0,0989	10
11.	SaNSDE (NP = 15)	1758	0,0142	10
12.	SaNSDE (NP = 20)	2048	0,0234	10

Tabel 3 menunjukkan bahwa performa SaNSDE dengan NP = 20 lebih baik dibanding dengan yang lain. Sebenarnya jika dilihat dari banyaknya evaluasi fungsi obyektif yang dilakukan serta waktu yang diperlukan, SaNSDE dengan NP = 15 paling baik dibanding lainnya, namun ini menjadi buruk karena tidak selalu bisa mencapai optimal global.

Tabel 4 menunjukkan bahwa performa SaNSDE dengan NP = 15 lebih baik dibanding dengan yang lain. Bahkan kedua kondisi SaNSDE mampu mengungguli semua performa DE dengan sepuluh strategi mutasinya, baik dari banyaknya evaluasi fungsi obyektif maupun dari waktu yang diperlukan untuk menemukan solusi optimal global (sebenarnya untuk waktu tidak bisa dibandingkan secara langsung karena perbedaan mesin komputer yang digunakan).

5. KESIMPULAN

Dalam penelitian ini, *Self-adaptive Differential Evolution with Neighborhood Search* (SaNSDE) telah diimplementasikan untuk menyelesaikan masalah optimasi sistem pompa air. Dari proses yang dilakukan dan hasil yang diperoleh, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Dari hasil pengujian, dapat dilihat bahwa algoritma SaNSDE mampu menyelesaikan optimasi sistem pompa air dengan baik. Algoritma ini berhasil menemukan solusi optimal global dari permasalahan tersebut berupa laju aliran pompa serta selisih tekanan minimal.
2. SaNSDE bekerja dengan baik dengan ukuran populasi NP = 20. Di samping selalu menemukan global optimal, dia melakukan lebih sedikit evaluasi terhadap fungsi obyektif dibandingkan dengan ukuran populasi lebih dari 20.
3. Dibandingkan dengan DE, SaNSDE menunjukkan performa yang lebih baik. Tetapi keunggulan ini menjadi tidak terlalu signifikan jika melihat lebih rumitnya

implementasi SaNSDE ke dalam kode bahasa pemrograman dibandingkan dengan DE yang lebih sederhana.

4. Dari sisi penentuan nilai parameter, SaNSDE lebih unggul dibandingkan DE. Dalam SaNSDE, tidak perlu mencoba – coba berbagai nilai untuk mencari kombinasi parameter (NP, CR dan F) yang menghasilkan performa yang baik.

PUSTAKA

Babu, B.V., Angira, R., 2003, *Optimization of Water Pumping System Using Differential Evolution Strategies*, Proceedings of The Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS-2003), Singapore.

Jaluria, Yogesh, 2007, *Design and Optimization of Thermal System. 2nd ed.,* Boca Raton, CRC Press, hal 350 – 360.

Karaboga, D., Okdem, S., 2004, *A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm*, Turk J Elec Engin, Vol. 12.

Onwubolu, G.C., Babu, B.V., 2004,Optimization of Non-Linear Chemical Processes Using Evolutionary Computation, *New Optimization Techniques in Engineering,* Berlin, Springer.

Pant, M., Radha, T., Rani, D., Abraham, A.,Srivastava, D.K., 2008, Estimation Using Differential Evolution for Optimal Crop Plan, *Hybrid Artificial Intelligence System.,* Springer, Berlin, hal 289 – 297.

Qin, A.K., Huang, V.L., Suganthan, P.N., 2006, *Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization,* IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, Kanada.

Qin, A.K., Suganthan, P.N., 2005, *Self-adaptive Differential Evolution Algorithm for Numerical Optimization,* IEEE Congress on Evolutionary Computation (CEC 2005), Edinburgh, Skotlandia.

Storn, R., 1996, *On the Usage of Differential Evolution for Function Optimization,* Biennial Conference of North American Fuzzy Information Processing Society, IEEE Press hal 519 – 523.

Suyanto, 2008, *Evolutionary Computation: Evolusi Berbasis Evolusi dan Genetika*, Informatika, Bandung, hal 139 – 154.

Zaharie, D., 2007, *A Comparative Analysis of Crossover Variants in Differential Evolution*, Proceedings of the International Multiconference on Computer Science and Information Technology hal 171-181.