Jurnal Ilmiah KURSOR
Menuju Solusi Teknologi Informasi
Vol. 4, No. 1, Januari 2008 ISSN 0216 - 0544

Optimasi Fungsi Tanpa Kendala Menggunakan Algoritma Genetika Dengan Kromosom Biner dan Perbaikan Kromosom *Hill-Climbing*

Wayan Firdaus Mahmudy, (wayanfm@ub.ac.id)
Program Studi Ilmu Komputer, Universitas Brawijaya

ABSTRAK

Optimasi merupakan masalah memaksimalkan atau meminimalkan suatu fungsi dengan kendala atau tanpa kendala. Dalam banyak kasus tidak mudah menyelesaikan masalah optimasi dengan fungsi tujuan non linier secara eksak. Algoritma genetika dengan representasi kromosom biner bisa digunakan untuk mendapatkan solusi hampiran. Algoritma genetika melakukan pencarian dalam area global sehingga hasilnya kurang akurat. Metode hill-climbing bisa digunakan untuk pencarian nilai optimum dan menjamin ditemukannya optimum lokal. Memadukan metode hill-climbing dengan algoritma genetika diharapkan meningkatkan akurasi hasil pencarian. Pada ujicoba dengan menggunakan beberapa fungsi uji, perbaikan kromosom dengan metode hill-climbing meningkatkan akurasi hasil secara nyata.

1. PENDAHULUAN

Pada pencarian solusi suatu masalah kadang-kadang dibutuhkan formulasi matematika yang kompleks untuk memberikan solusi yang pasti. Solusi optimum mungkin dapat diperoleh tetapi memerlukan proses perhitungan yang panjang dan tidak praktis. Untuk mengatasi kasus khusus seperti di atas dapat digunakan metode heuristik, yaitu suatu metode pencarian yang didasarkan atas intuisi atau aturan-aturan empiris untuk memperoleh solusi yang lebih baik daripada solusi yang telah dicapai sebelumnya [1]. Metode heuristik tidak selalu menghasilkan solusi terbaik tetapi jika dirancang dengan baik akan menghasilkan solusi yang mendekati optimum dalam waktu yang cepat. Algoritma genetika (*genetic algorithms* / *GAs*) adalah salah satu cabang *evolutionary algorithms*, yaitu suatu teknik optimasi yang didasarkan pada genetika alami. Dalam algoritma genetika untuk menghasilkan suatu solusi optimal, proses pencarian dilakukan di antara sejumlah alternatif titik optimal berdasarkan fungsi probabilistik [2].

Algoritma genetika murni memberikan hasil kurang optimum pada pencarian dalam area yang kompleks. Penggabungan (*hybridisation*) dengan teknik lain dapat meningkatkan hasil secara nyata dalam efisiensi pencarian. GAs yang telah dihibridisasi dengan teknik pencarian lokal (*local search techniques* / LS) biasa disebut *memetic algorithms* (MAs). MAs merupakan algoritma evolusioner yang menerapkan secara terpisah proses LS untuk perbaikan individu, misalnya dengan meningkatkan nilai fitness dengan *hill-climbing*. Aspek penting berkaitan dalam MAs adalah keseimbangan antara eksplorasi kemampuan GAs dan eksplorasi kemampuan LS [3].

Pada pendekatan hibrid, optimasi lokal diterapkan pada setiap individu baru (*offspring*) dengan menggerakkannya menuju optimum lokal sebelum dimasukkan ke dalam populasi. GAs digunakan pada eksplorasi global dan LS digunakan pada eksplorasi lokal di sekitar kromosom [4]. Apabila P(t) dan C(t) merupakan parents dan children pada generasi ke-t, maka struktur umum algoritma genetika dapat dideskripsikan sebagai berikut:

```
procedure AlgoritmaGenetika
begin
  t = 0
  inisialisasi P(t)
  while (bukan kondisi berhenti) do
    evaluasi P(t)
    seleksi P(t)
    reproduksi C(t) dari P(t)
    bentuk P(t+1) dari P(t) dan C(t) terbaik
    t = t + 1
  end while
end
```

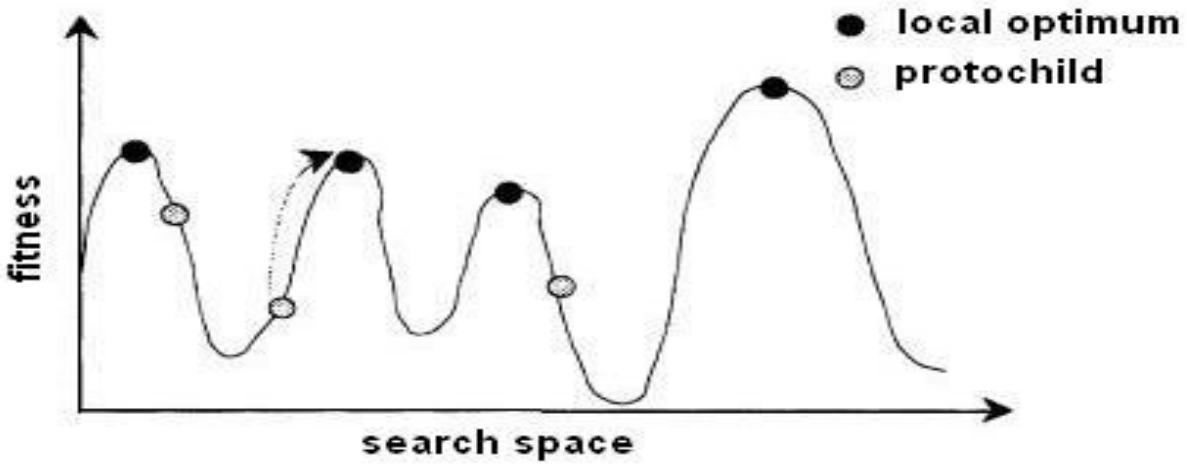
Gambar 1. Struktur Algoritma Genetika

Pada struktur Algoritma Genetika Hibrid ditambahkan perbaikan lokal sebagai berikut:

```
procedure AlgoritmaGenetikaHibrid
begin
  t = 0
  inisialisasi P(t)
  while (bukan kondisi berhenti) do
    evaluasi P(t)
    seleksi P(t)
    reproduksi C(t) dari P(t)
    perbaiki C(t)
    bentuk P(t+1) dari P(t) dan C(t) terbaik
    t = t + 1
  end while
end
```

Gambar 2. Struktur Algoritma Genetika Hibrid

Crossover dan mutasi seringkali menghasilkan anak di luar optimum lokal. MAs dapat memberikan hasil yang lebih baik seperti ditunjukkan pada Gambar 3. Anak yang baru terbentuk (*protochild*) akan didorong menuju optimum lokal.



Gambar 3. MAs dan optimasi lokal [4].

2. ALGORITMA GENETIKA DENGAN KROMOSOM BINER

Ada banyak cara untuk merepresentasikan kromosom, baik dengan bilangan real maupun bilangan biner [3]. Karena sifatnya yang lebih alami, algoritma genetika dalam makalah ini menggunakan representasi biner. Bagian ini menjelaskan representasi kromosom dengan kode biner serta metode evaluasi, seleksi, crossover dan mutasi yang digunakan.

2.1. Representasi Biner

Representasi kromosom biner ditentukan oleh variabel keputusan. Panjang string kromosom tergantung pada presisi yang dibutuhkan. Misalkan domain variabel x_j adalah $[a_j,b_j]$ dan presisi yang dibutuhkan adalah lima angka di belakang koma maka interval dari domain tiap variabel setidaknya mempunyai lebar $(b_j - a_j)\times 10^5$. Banyaknya bit yang dibutuhkan (m_j) untuk variabel x_j adalah [5]

$$2^{m_j-1} < (b_j - a_j)\times 10^5 \le 2^{m_j-1} - 1$$

Pemetaan dari string biner menjadi bilangan real untuk variabel x_j adalah sebagai berikut

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

decimal(substring) merepresentasikan nilai desimal dari *substring* bagi variabel keputusan x_j.

2.2. Evaluasi

Untuk mengevaluasi nilai fitness dari kromosom dilakukan langkah-langkah berikut

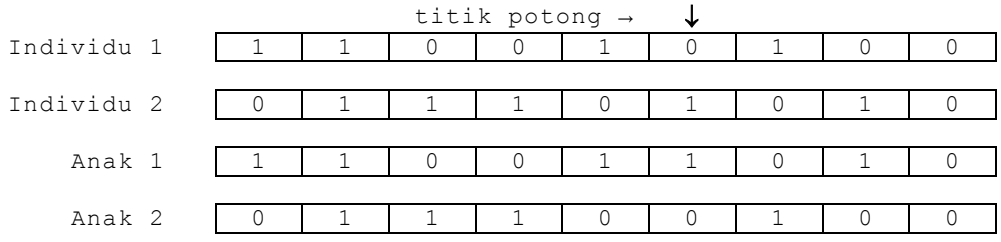
- Konversi *genetype* dari kromosom menjadi *phenotype*. Artinya konversi string biner menjadi nilai real dari $x^k = (x_1^k,...,x_n^k)$; k=1,2,...,ukuran_populasi, n=banyaknya_variabel_keputusan.
- Evaluasi nilai fungsi tujuan $f(x^k)$
- Konversi nilai fungsi tujuan menjadi nilai *fitness*. Untuk masalah maksimasi nilai *fitness*= $f(x^k)$. Untuk masalah minimasi nilai *fitness*=z - $f(x^k)$; z adalah sembarang bilangan real.

2.3. Seleksi

Seleksi digunakan untuk memilih dua individu/kromosom sebagai induk untuk anak pada generasi berikutnya. Pada makalah ini digunakan metode pemilihan seragam, artinya setiap individu dalam populasi memiliki peluang yang sama untuk terpilih. Metode ini dipilih untuk menghemat waktu yang digunakan pada komputasi nilai probabilitas kumulatif yang digunakan pada metode seleksi *roulette-wheel*.

2.4. Crossover

Metode crossover yang digunakan adalah metode *one-cut-point*, yang secara acak memilih satu titik potong dan menukarkan bagian kanan dari tiap induk untuk menghasilkan *offspring*. Ilustrasi metode ini digambarkan pada Gambar 4.



Gambar 4. Ilustrasi crossover dengan 2 titik tukar

2.5. Mutasi

Metode mutasi yang digunakan adalah dengan memilih satu titik acak kemudian mengubah nilai gen pada titik tersebut.

Gambar 5. Ilustrasi mutasi dengan perubahan 1 titik

3. ALGORITMA HILL-CLIMBING

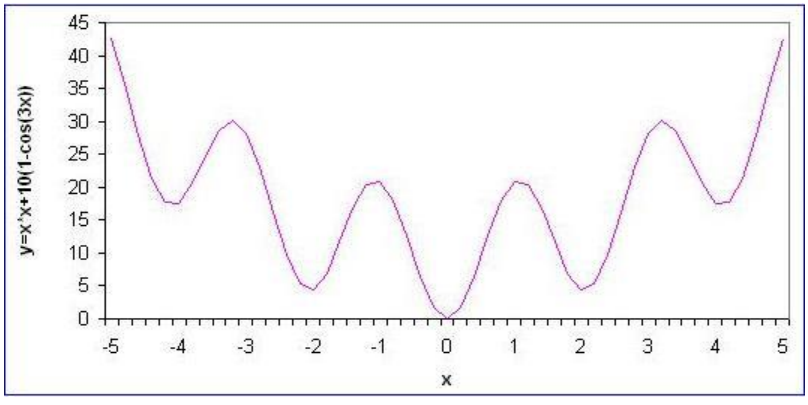
Algoritma *hill-climbing* adalah metode pencarian berdasarkan nilai tetangga. Algoritma ini menggerakan variabel keputusan x_j menuju (x_j + *step*) atau (x_j - *step*) berdasarkan nilai fungsi obyektive. Pada masalah maksimasi, x_j menuju (x_j + *step*) jika $f(x_j + step) > f(x_j - step)$, x_j menuju (x_j - *step*) jika $f(x_j + step) < f(x_j - step)$. Pada masalah minimasi berlaku sebaliknya. *Step* adalah nilai perubahan yang diinginkan, jika dipilih terlalu kecil maka pencarian berjalan lambat, jika terlalu besar maka bisa jadi titik optimum akan terlewati. Karena pada makalah ini digunakan representasi kromosom biner maka digunakan kenaikan dan penurunan nilai sebesar 1 pada bilangan biner.

4. METODE

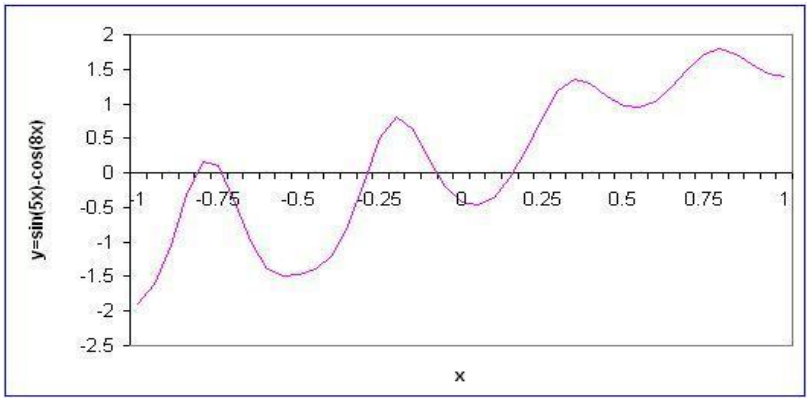
Pada makalah ini digunakan empat fungsi untuk ujicoba perangkat lunak yang dibangun berdasarkan algoritma genetika. Fungsi-fungsi yang digunakan dalam pengujian adalah

	Fungsi	tujuan	Range
1	$f(x) = x^2 + 10x(1-\cos(3x))$	cari minimum	-5≤ x ≤5
2	$f(x) = \cos(\sin(4x) + 2\cos(6x)) + x$	cari maksimum	-1≤ x ≤1
3	$f(x,y) = 1.1x\sin(2x) + y\sin(4y)$	cari minimum	0 ≤ x ≤ 10 0 ≤ y ≤ 10
4	$f(x,y) = 21.5 + x\sin(4\pi x) + y\sin(20\pi y)$	cari maksimum	-3 ≤ x ≤ 12,1 4,1 ≤ y ≤ 5,8

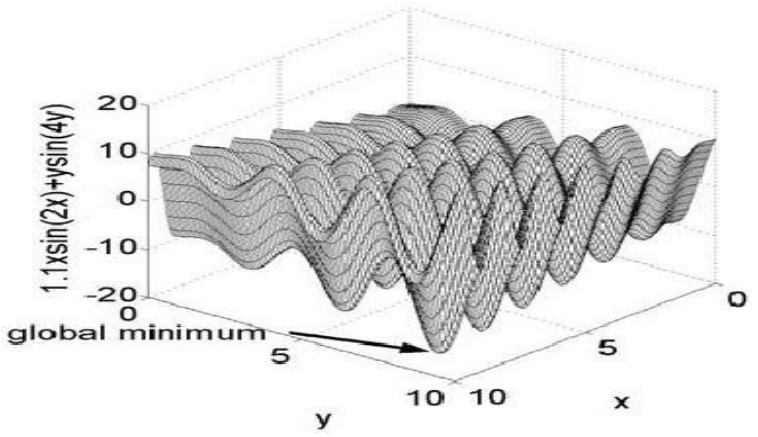
Fungsi-fungsi yang digunakan semuanya memiliki maksimum dan minimum lokal seperti terlihat pada Gambar 6 sampai Gambar 9.



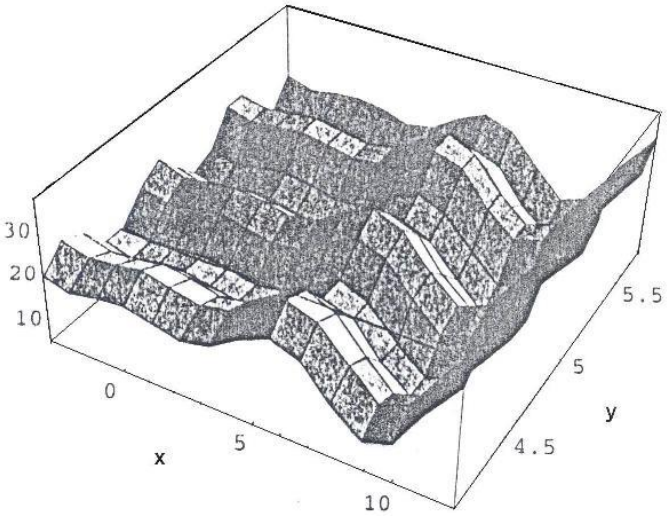
Gambar 6. Plot fungsi uji ke-1



Gambar 7. Plot fungsi uji ke-2



Gambar 8. Plot fungsi uji ke-3 [7]



Gambar 9. Plot fungsi uji ke-4 [5]

Perangkat lunak dijalankan dengan ukuran populasi sebesar 10, iterasi berhenti jika dalam 10.000 generasi ke depan tidak ada perubahan fitness (konvergen). Nilai *crossover rate* dipilih sebesar 0,8. Nilai *mutation rate* dipilih sebesar 0,1.

5. HASIL DAN PEMBAHASAN

Untuk mengetahui keoptimalan hasil MAs dibandingkan GAs, berikut ini diberikan tabel perbandingannya pada empat fungsi uji. Tabel 1 menampilkan hasil rata-rata 10 kali uji coba untuk tiap fungsi

Tabel 1. Perbandingan Hasil Perhitungan

fungsi ke	Tujuan pencarian	konvergen pada generasi		waktu proses (detik)			nilai fungsi tujuan	
		GAs	MAs	GAs	MAs	Beda (%)	GAs	MAs
1	Minimum	631	520	2,4	2,1	14,2	0,00001	0
2	Maksimum	892	632	4,2	3,4	23,5	1,7245	1,7933
3	Minimum	10.279	3592	10,4	8,7	29,5	-17,9872	-18,5547
4	Maksimum	14.890	8443	14,9	10,3	44,7	37,8590	38,8495

Dari Tabel 1 terlihat untuk semua fungsi uji, MAs akan mencapai titik konvergen lebih cepat. Pada fungsi uji sederhana (fungsi ke-1 dan ke-2) perbedaan waktu konvergen tidak terlalu besar. Pada fungsi uji yang lebih kompleks (fungsi ke-3 dan ke-4) perbedaan waktu konvergen cukup signifikan. Dari semua fungsi uji, MAs memberikan hasil nilai fungsi tujuan lebih baik (lebih mendekati optimum) dibandingkan GAs.

Dari aspek waktu konvergen dan nilai fungsi tujuan MAs memberikan hasil yang lebih baik karena MAs tidak perlu berlama-lama melakukan pencarian dalam area global. Jika satu individu/kromosom diletakkan dalam daerah pencarian (*search space*), dia akan segera bergerak menuju optimum lokal. Ini berarti MAs telah berhasil melakukan perbaikan lokal pada daerah pencarian dengan metode *hill-climbing*.

6. KESIMPULAN DAN SARAN

– MAs menghasilkan waktu konvergen dan fungsi tujuan lebih baik dibandingkan GAs.

– MAs akan memberikan hasil perbaikan yang sangat nyata jika digunakan pada fungsi yang kompleks.

DAFTAR PUSTAKA

[1] Dimyati, T,T, dan Dimyati, A. (2003), *Operations Research, Model-Model Pengambilan Keputusan*, Sinar Baru Algensindo, Bandung,

[2] Michalewicz, Zbigniew. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Heidelberg.

[3] Lozano, M; Herrera, F; Krasnogor, N; Molina, D. (2004). Real-Coded Memetic Algorithms with Crossover Hill-Climbing. Evolutionary Computation 12(3): 273-302.

[4] Gen, M and Cheng, R. (2000). Genetic Algorithm and Engineering Optimization. John Wiley Sons, Inc, New York.

[5] Gen, M and Cheng, R. (1997). Genetic Algorithm and Engineering Design. John Wiley Sons, Inc, New York.

[6] Lozano, M; Herrera, F; Verdegay, J.L. (1998). Tackling Real-Coded Genetic Algorithms Operators and Tools for Behavioural Analysis. Artificial Intellegence Review 12: 256-319.

[7] Randy L Haupt dan Sue Ellen Haupt. (2004). Practical Genetic Algorithms, second edition. John Wiley Sons, Inc, New York.