

LAPORAN
TUGAS BESAR ARTIFICIAL INTELLIGENCE

KNAPSACK PROBLEM DENGAN ALGORITMA GENETIKA

Disusun Oleh :

Bayu Kusumo Hapsoro (113050220)

Barkah Nur Anita (113050228)

Radityo Basith (113050252)

Ilmi Hayyu Dinna (113050261)

IF 29 05



INSTITUT TEKNOLOGI TELKOM
DEPARTEMEN TEKNIK INFORMATIKA
BANDUNG
2008

1. KNAPSACK PROBLEM

Knapsack Problem merupakan suatu masalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang dimana setiap barang mempunyai weight dan profit masing-masing, sehingga dari pemilihan barang tersebut didapatkan profit yang maksimum.

Knapsack problem merupakan salah satu dari persoalan klasik yang banyak ditemukan dalam literatur-literatur lama dan hingga kini permasalahan tersebut masih sering ditemukan dalam kehidupan sehari-hari. Contoh nyata dari Knapsack Problem ini misalnya, jika ada seorang pedagang barang kebutuhan rumah tangga yang berkeliling menggunakan gerobak. Tentu saja gerobaknya memiliki kapasitas maksimum, sehingga ia tidak bisa memasukkan semua barang dagangannya dengan sekuat hatinya. Pedagang tersebut harus memilih barang-barang mana saja yang harus ia angkut, dengan pertimbangan berat dari barang yang dibawanya tidak melebihi kapasitas maksimum gerobak dan memaksimalkan profit dari barang-barang yang ia bawa.

Banyak algoritma yang dapat digunakan untuk menyelesaikan Knapsack Problem ini, misalnya Algoritma Brute Force, Branch and Bound, Greedy, dan lain-lain. Untuk tugas AI kali ini, kami akan mencoba menyelesaikan Knapsack Problem dengan menggunakan Algoritma Genetika.

2. ALGORITMA GENETIKA

Algoritma genetika adalah algoritma komputasi yang diinspirasi oleh teori evolusi yang kemudian diadopsi menjadi algoritma komputasi untuk mencari solusi suatu permasalahan dengan cara yang alamiah. Algoritma ini dikembangkan oleh Goldberg yang terinspirasi dari teori evolusi Darwin yang menyatakan bahwa kelangsungan hidup suatu makhluk dipengaruhi oleh aturan “yang kuat adalah yang menang”. Darwin juga mengatakan bahwa kelangsungan hidup suatu makhluk dapat dipertahankan melalui proses reduksi, crossover, dan mutasi.

Sebuah solusi yang dibangkitkan dalam Algoritma Genetika disebut sebagai kromosom, sedangkan kumpulan kromosom-kromosom tersebut disebut sebagai populasi. Sebuah kromosom dibentuk dari komponen-komponen penyusun yang disebut sebagai gen dan nilainya dapat berupa bilangan numerik, biner, simbol atau pun karakter tergantung dari permasalahan yang ingin diselesaikan. Kromosom-kromosom tersebut akan berevolusi secara berkelanjutan yang disebut dengan generasi. Dalam tiap generasi, kromosom-kromosom tersebut dievaluasi tingkat keberhasilan nilai solusinya terhadap masalah yang ingin diselesaikan (fungsi_objektif) menggunakan ukuran yang disebut fitness.

Untuk memilih kromosom yang tetap dipertahankan untuk generasi selanjutnya, dilakukanlah proses seleksi. Kromosom dengan nilai fitness tinggi akan memiliki peluang lebih besar untuk terpilih lagi pada generasi selanjutnya.

Offspring merupakan kromosom-kromosom baru yang dibentuk dengan cara melakukan perkawinan antar kromosom dalam satu generasi, atau sering disebut sebagai proses crossover. Jumlah kromosom yang mengalami crossover ditentukan oleh parameter $P_{crossover}$. Mekanisme perubahan susunan unsur penyusun makhluk hidup akibat adanya faktor alam disebut dengan mutasi. Jadi, mutasi direpresentasikan sebagai suatu proses berubahnya satu atau lebih nilai gen dalam kromosom dengan suatu nilai acak. Jumlah gen dalam populasi yang mengalami mutasi ditentukan oleh parameter P_{mutasi} . Setelah beberapa generasi akan dihasilkan kromosom-kromosom yang nilai gennya konvergen ke suatu nilai tertentu yang merupakan solusi terbaik yang dihasilkan oleh Algoritma Genetika terhadap permasalahan yang ingin diselesaikan.

Algoritma Genetika sangat cocok untuk menyelesaikan masalah optimasi dengan ruang lingkup yang besar, karena Algoritma Genetika selalu bergerak dengan mencari sejumlah solusi sekaligus, selama solusi tersebut masih bersifat feasible (tidak melanggar constraint). Dengan setting parameter yang tepat, diharapkan salah satu dari sekian banyak solusi yang dibangkitkan oleh Algoritma Genetika merupakan solusi optimum global.

Akan tetapi, Algoritma Genetika ini juga masih memiliki kelemahan yaitu ketidakpastian untuk menghasilkan solusi optimum global, karena sebagian besar dari algoritma ini berhubungan dengan bilangan random yang bersifat probabilistik. Peranan programmer disini adalah memaksimalkan probabilitas dalam menghasilkan solusi optimum global dengan cara membuat suatu skema pengolahan input argumen (fungsi fitness) dan setting parameter yang tepat.

3. PENERAPAN ALGORITMA GENETIKA DALAM KNAPSACK PROBLEM

Berikut adalah pengolahan fitness dan setting parameter yang kami terapkan :

- Representasi Barang

Kami merepresentasikan barang dalam dua array, dimana array pertama berisi weight (berat) barang, dan array kedua berisi profit (keuntungan) barang.

Weight :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
180	170	100	190	270	120	190	140	180	100	140	70	150	120	190	140	80	150	200	130

Profit :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
200	150	90	220	250	80	170	120	190	70	160	110	120	160	220	140	120	110	160	180

- Constraint

Adapun constraint yang kami gunakan dalam aplikasi ini adalah weight. Jadi, total berat dari sekumpulan barang yang dipilih tidak boleh melebihi kapasitas Knapsack.

- Encoding Kromosom

Untuk merepresentasikan kromosom, kami menggunakan array 1 dimensi yang berisi 1 atau 0.

Misal :

Kromosom : 1 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0

Arti : Barang 1, 4, 8, 9, 10, 12, 14, 16, 18 diambil

Barang 2, 3, 5, 6, 7, 11, 13, 15, 17, 19, 20 tidak diambil

- Termination Conditions

Pencarian solusi berhenti jika terdapat > 60% kromosom yang mempunyai nilai fitness maksimum ATAU jumlah evolusi lebih besar limit evolusi yang telah ditentukan (jika jumlah evolusi > 1000).

- Fitness Function

Pada evolusi di dunia nyata, individu bernilai fitness tinggi akan bertahan hidup. Sedangkan individu bernilai fitness rendah akan mati. Pada AG, suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran nilai fitness-nya. Pada aplikasi ini, fitness dihitung dengan menjumlahkan profit tiap barang yang masuk ke dalam knapsack. Jika berat total dalam satu kromosom lebih besar daripada kapasitas maksimum knapsack, maka nilai fitnessnya diassign 0.

Selain dihitung nilai fitnessnya, dihitung pula berat total dari tiap kromosom untuk kemudian dilakukan pengecekan, dimana apabila ada kromosom yang berat totalnya melebihi kapasitas dari knapsack, maka akan dilakukan pencarian gen dalam kromosom tersebut yang bernilai 1 untuk diganti dengan nilai 0. Hal ini dilakukan terus menerus sampai dipastikan bahwa semua kromosom tidak ada yang melanggar constraint.

Untuk mencegah adanya individu yang dominan dalam suatu populasi (dalam pemilihan parent untuk dicrossover), maka diperlukan suatu fungsi Linier Fitness Ranking. Fungsi ini akan menurunkan perbedaan nilai fitness antar individu, sehingga perbedaan antara nilai fitness terbaik dengan nilai fitness terendah dapat diperkecil. Dengan begitu setiap kromosom memiliki kemungkinan untuk terpilih menjadi parent secara lebih merata (lebih adil).

- Selection Function

Aplikasi ini menggunakan metode seleksi Roulette Wheel yang dikombinasikan dengan Elitism. Roulette Wheel merupakan suatu metode pemilihan kromosom untuk dijadikan parent, dimana kromosom dengan fitness tinggi mempunyai peluang lebih besar untuk dijadikan parent. Sedangkan Elitism adalah suatu metode yang berguna untuk mempertahankan nilai best fitness suatu generasi agar tidak turun di generasi berikutnya. Dalam AG caranya adalah dengan mengcopykan individu terbaik (maxfitness) sebanyak yang dibutuhkan.

- Crossover

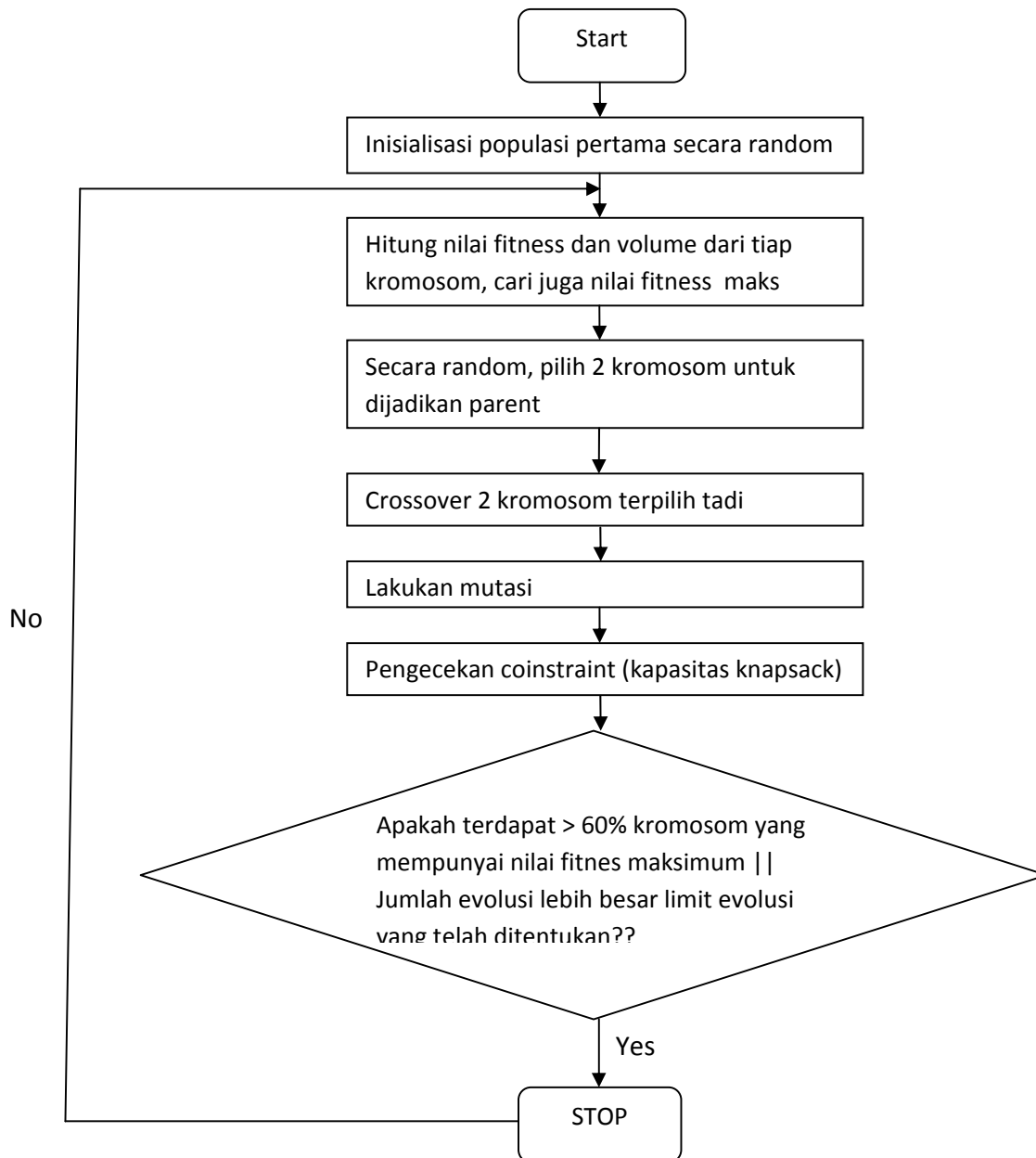
Crossover merupakan proses mengkombinasikan bit-bit dalam satu kromosom dengan kromosom lain yang terpilih sebagai parent. Jumlah kromosom yang mengalami crossover ditentukan oleh parameter $P_{crossover}$. Dimana $P_{crossover}$ ini kami assign sebesar 80%, karena kami mengharapkan 80% dari populasi mengalami crossover agar populasi individu menjadi lebih variatif.

- Mutation

Mutation diperlukan untuk mengembalikan informasi bit yang hilang akibat crossover. Mutasi ini dilakukan pada tingkat gen, dan jumlah gen yang dimutasi kami batasi dalam suatu variabel P_{mutasi} sebesar 5%. Nilai ini kami rasa cukup karena semakin banyak gen yang dimutasi maka kualitas dari suatu individu bisa mengalami penurunan.

Setelah dilakukan mutasi, kembali dicek untuk tiap kromosomnya apakah melanggar constraint atau tidak. Jika ada kromosom yang total beratnya melebihi kapasitas Knapsack, maka secara random, gen yang bernilai 1 akan diganti dengan 0 sampai kromosom tersebut tidak melanggar constraint. Jadi dapat disimpulkan, aplikasi kami akan selalu menemukan solusi.

FLOWCHART



4. KESIMPULAN

Adapun kesimpulan yang dapat kami ambil adalah :

- Penerapan Algoritma Genetika dalam penyelesaian Knapsack Problem ini memiliki kelemahan yaitu ketidakpastian untuk menghasilkan solusi optimum global. Hal ini berlaku untuk semua kasus karena sebagian besar dari Algoritma Genetika ini berhubungan dengan bilangan random yang bersifat probabilistik.

- Aplikasi ini akan selalu menemukan solusi, karena pengecekan apakah kromosom dalam suatu populasi dilakukan dua kali, yakni ketika inisialisasi populasi awal dan ketika kromosom-kromosom telah dimutasi.

5. DAFTAR PUSTAKA

- [1] Suyanto. 2005. *Algoritma Genetika Dalam Matlab*. Yogyakarta : Penerbit Andi.
- [2] Suyanto. 2007. *Artificial Intellegence*. Bandung : Penerbit Informatika.
- [3] Shrestha, Dipti dan Maya Hristakeva. *Solving the 0-1 Knapsack Problem with Genetic Algorithms*. USA : Computer Science Department, Simpson College.
- [4] Permata, Anggi Shena. *Pemecahan Masalah Knapsack dengan Algoritma Branch And Bound*. Bandung : Institut Teknologi Bandung.

LAMPIRAN

Beberapa print screen aplikasi yang kami buat:

