

RANCANG BANGUN APLIKASI SISTEM ABSENSI DAN JADWAL KERJA KARYAWAN PADA PERUSAHAAN MENGGUNAKAN METODE ALGORITMA GENETIKA (STUDI KASUS: SPBU 54.651.16 KOTA MALANG)

Dani Valiandra

Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang

Abstrak

Penjadwalan kerja karyawan dalam suatu perusahaan adalah hal yang rumit. Permasalahan yang dihadapi adalah jadwal antar karyawan sering bentrok dan hari libur yang tidak merata tiap karyawan, sehingga harus dipikirkan juga solusi agar libur karyawan merata dan jadwal antar karyawan perusahaan tidak bentrok.

Salah satu metode yang dapat digunakan untuk menyelesaikan permasalahan tersebut adalah dengan menggunakan pendekatan algoritma genetika. Algoritma genetika merupakan pendekatan komputasional untuk menyelesaikan masalah yang dimodelkan dengan proses biologi dari evolusi.

Diharapkan dengan digunakannya algoritma genetika akan diperoleh optimasi penjadwalan yaitu kondisi dimana terjadi kombinasi terbaik untuk pasangan mata kuliah dan dosen pengajar secara keseluruhan, tidak ada permasalahan bentrokan jadwal pada sisi mahasiswa, serta ketersediaan ruang yang cukup dan sesuai secara fasilitas untuk seluruh mata kuliah yang ada.

Kata kunci: penjadwalan, optimasi, algoritma genetika.

1. PENDAHULUAN

Penjadwalan kerja karyawan di suatu perusahaan merupakan pekerjaan yang tidak mudah. Ada beberapa aspek yang berkaitan dalam penjadwalan tersebut dan harus dilibatkan dalam pertimbangan di antaranya:

1. Dalam 1 hari kerja terdapat 3 shift, pagi, siang dan malam.
2. Karyawan Laki-laki di fokuskan pada shift malam.
3. Tidak boleh ada jadwal karyawan yang bentrok.
4. Penempatan pompa karyawan terbagi rata.

Di samping aspek-aspek di atas, dalam penyusunan jadwal kerja karyawan ini pun terdapat sangat banyak kemungkinan yang selayaknya dicoba untuk menemukan penjadwalan yang terbaik. Karena itu dibutuhkan metode optimasi yang dapat diterapkan untuk mengerjakan penjadwalan kerja karyawan ini.

2. ALGORITMA GENETIKA

2.1 Pengertian Algoritma Genetika

Algoritma ini ditemukan di Universitas Michigan, Amerika Serikat oleh John Holland (1975) melalui sebuah penelitian dan dipopulerkan oleh salah satu muridnya, David Goldberg. [1]

Algoritma genetika adalah algoritma yang berusaha menerapkan pemahaman mengenai evolusi alamiah pada tugas-tugas pemecahan-

masalah (problem solving). Pendekatan yang diambil oleh algoritma ini adalah dengan menggabungkan secara acak berbagai pilihan solusi terbaik di dalam suatu kumpulan untuk mendapatkan generasi solusi terbaik berikutnya yaitu pada suatu kondisi yang memaksimalkan kecukupannya atau lazim disebut fitness. Generasi ini akan merepresentasikan perbaikan-perbaikan pada populasi awalnya. Dengan melakukan proses ini secara berulang, algoritma ini diharapkan dapat mensimulasikan proses evolusioner. Pada akhirnya, akan didapatkan solusi-solusi yang paling tepat bagi permasalahan yang dihadapi.

Untuk menggunakan algoritma genetika, solusi permasalahan direpresentasikan sebagai khromosom. Tiga aspek yang penting untuk penggunaan algoritma genetika:

1. Definisi fitness function
2. Definisi dan implementasi representasi genetika
3. Definisi dan implementasi operasi genetika

Jika ketiga aspek di atas telah didefinisikan, algoritma genetika generik akan bekerja dengan baik.

Tentu saja, algoritma genetika bukanlah solusi terbaik untuk memecahkan segala masalah. Sebagai contoh, metode tradisional telah diatur untuk mencari penyelesaian dari fungsi analitis konveks yang “berperilaku baik” yang variabelnya sedikit. Pada kasus-kasus ini, metode berbasis kalkulus lebih unggul dari algoritma genetika karena metode ini dengan cepat menemukan solusi minimum ketika algoritma

genetik masih menganalisa bobot dari populasi awal. Untuk problem-problem ini pengguna harus mengakui fakta dari pengalaman ini dan memakai metode tradisional yang lebih cepat tersebut. Akan tetapi, banyak persoalan realistik yang berada di luar golongan ini. Selain itu, untuk persoalan yang tidak terlalu rumit, banyak cara yang lebih cepat dari algoritma genetik. Jumlah besar dari populasi solusi, yang merupakan keunggulan dari algoritma genetik, juga harus mengakui kekurangannya dalam kecepatan pada sekumpulan komputer yang dipasang secara seri –fitness function dari tiap solusi harus dievaluasi. Namun, bila tersedia komputer-komputer yang paralel, tiap prosesor dapat mengevaluasi fungsi yang terpisah pada saat yang bersamaan. Karena itulah, algoritma genetik sangat cocok untuk perhitungan yang paralel.

2.2 Prosedur Umum Algoritma Genetika

Prosedur umum algoritma genetika adalah sebagai berikut :

1. Pengkodean (encoding) calon solusi dan set-up beberapa parameter awal jumlah individu, probabilitas, penyilangan dan mutasi, dan jumlah generasi maksimum.
2. Pembangkitan acak sejumlah n kromosom pada generasi ke-0.
3. Evaluasi masing-masing kromosom dengan menghitung nilai fitness-nya.
4. Seleksi beberapa kromosom dari sejumlah n individu yang memiliki nilai fitness terbaik.
5. Rekombinasikan kromosom terpilih dengan cara melakukan penyilangan (crossover) dan mutasi (mutation).
6. Update jumlah generasi dan kembali ke langkah 2 sampai jumlah generasi maksimum tercapai.

Diagram alir algoritma genetika dapat dilihat pada gambar 1.

Representasi atau Inisialisasi Kromosom

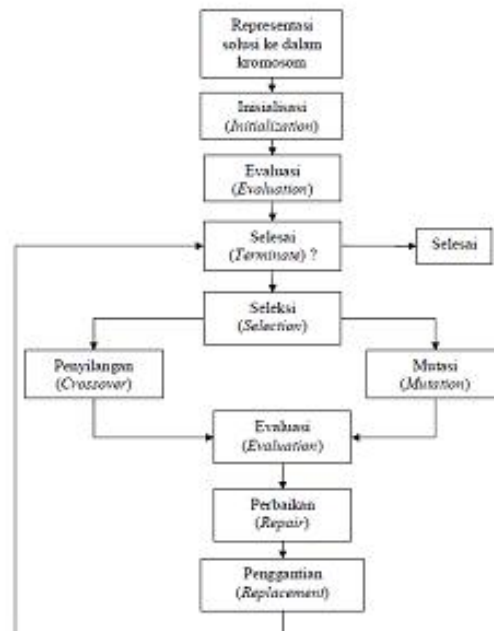
Inisialisasi kromosom direpresentasikan dalam bentuk larik dengan tipe data record yang berisi data yang mendukung proses penjadwalan. Panjang dari kromosom adalah sebanyak gen yang ada, dalam hal ini setiap gen mewakili mata kuliah yang ditawarkan. Setiap kromosom adalah barisan gen yang terdiri dari nilai hari, jam dan ruang.

Fungsi Fitness

Fungsi fitness adalah fungsi yang akan mengukur tingkat kebugaran suatu kromosom dalam populasi. Semakin besar nilai fitness, semakin bugar pula kromosom dalam populasi sehingga semakin besar kemungkinan kromosom tersebut dapat tetap bertahan pada generasi berikutnya.

Individu-individu dalam populasi telah terbentuk, maka langkah selanjutnya adalah menghitung nilai fitness setiap individu. Penghitungan dilakukan dengan memberikan pinalti untuk setiap aturan yang digunakan dalam penjadwalan. Semakin wajib aturan dilaksanakan,

maka akan semakin besar nilai pinalti yang diberikan.



Gambar 1. Diagram Alir Algoritma Genetika

Berikut aturan penghitungan fungsi fitness :

$$Fitness = \frac{1}{1 + Pinalti}$$

dimana :

$$Pinalti = \sum Bp \sum Np$$

Dari persamaan di atas nilai fitness ditentukan oleh nilai pinalti. Pinalti tersebut menunjukkan jumlah pelanggaran kendala pada suatu kromosom. Semakin tinggi nilai fitness akan semakin besar kemungkinan kromosom tersebut terpilih ke generasi berikutnya. Jadi nilai pinalti berbanding terbalik dengan nilai fitness, semakin kecil nilai pinalti (jumlah pelanggaran) semakin besar nilai fitness-nya.

Jadi fungsi fitness :

$$Fitness = \frac{1}{1 + \sum Bp \sum Np}$$

Keterangan :

Bp : Bobot pelanggaran

Np : Indikator pelanggaran

Seleksi (Selection) Kromosom

Setelah populasi awal terbentuk, setiap kromosom dalam populasi dievaluasi dengan menghitung nilai fungsi fitness-nya. Setelah itu proses pembentukan generasi baru diawali dengan seleksi kromosom. Seleksi adalah proses pemilihan beberapa kromosom untuk dijadikan sebagai

kromosom induk bagi generasi berikutnya. Kromosom terpilih untuk masing-masing populasi di dalam generasi yang berikutnya berdasarkan nilai fitness.

Penyilangan (Crossover) Kromosom

Setelah proses penyeleksian kromosom, langkah berikutnya adalah melakukan penyilangan terhadap pasangan-pasangan kromosom. Penyilangan (crossover) dikenal sebagai operator penggabungan ulang (recombination) yang paling utama dalam algoritma genetika. Penyilangan akan menukar informasi genetik antara dua kromosom induk yang terpilih dari proses seleksi untuk membentuk dua anak. Operator penyilangan bekerja pada sepasang kromosom induk untuk menghasilkan dua kromosom anak dengan menukarkan beberapa elemen (gen) yang dimiliki masing-masing kromosom induk.

Operator penyilangan biasanya dihubungkan dengan peluang penyilangan. Peluang penyilangan (P_c) adalah rasio antara jumlah kromosom yang diharapkan mengalami penyilangan dalam setiap generasi dengan jumlah kromosom total dalam populasi. Nilai P_c biasanya cukup tinggi (berkisar antara 0,6 – 1). Proses penyilangan akan terjadi pada sepasang kromosom jika suatu bilangan yang dibangkitkan secara acak (r), $0 < r < 1$, nilainya kurang dari atau sama dengan P_c . Bilangan acak tersebut dibangkitkan setiap kali akan menyilangan sepasang kromosom. Tingkat penyilangan yang tinggi menyebabkan semakin besar kemungkinan algoritma genetika mengeksplorasi ruang pencarian sekaligus mempercepat ditemukannya solusi optimum. Peluang penyilangan yang tepat dan efektif hanya dapat diketahui melalui pengujian (experiment) khusus terhadap masalah yang bersangkutan. Misalnya ditentukan nilai $P_c = 0.9$.

Mutasi (Mutation) Kromosom

Operasi ini akan menjadi sangat penting apabila nilai fitness kromosom dalam populasi cenderung sama atau sudah mencapai konvergen bias (premature convergen). Akibatnya, operator seleksi akan mengalami kesukaran memilih kromosom terbaik untuk dilakukan penyilangan. Dengan adanya operator mutasi, struktur suatu kromosom dapat dimodifikasi agar dihasilkan kromosom dengan struktur baru yang memiliki nilai fitness lebih baik.

Mutasi pada umumnya digunakan untuk mencegah tidak adanya kehilangan informasi sehingga dilaksanakan dengan pertukaran informasi di dalam kromosom. Operator mutasi dapat bersifat konstruktif maupun destruktif terhadap suatu kromosom, tetapi karena probabilitasnya yang kecil maka terjadinya mutasi akan sangat kecil karena didominasi oleh operator penyilangan. Peluang mutasi (P_m) adalah rasio antara jumlah gen yang diharapkan mengalami mutasi pada setiap generasi

dengan jumlah gen total dalam populasi. Nilai P_m yang digunakan biasanya sangat kecil (berkisar antara 0,001 – 0,2). Proses mutasi akan terjadi pada suatu gen, jika suatu bilangan yang dibangkitkan secara acak r , $0 < r < 1$, nilainya kurang dari atau sama dengan P_m .

3. Metodologi Penelitian

3.1 Prototyping

Prototyping merupakan salah satu metode pengembangan perangkat lunak yang banyak digunakan. Dengan metode *prototyping* ini pengembang dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem. Sering terjadi seorang pelanggan hanya mendefinisikan secara umum apa yang dikehendakinya tanpa menyebutkan secara detail *output* apa saja yang dibutuhkan, pemrosesan, dan data-data apa saja yang dibutuhkan. Sebaliknya disisi pengembang kurang memperhatikan efisiensi algoritma, kemampuan sistem operasi dan *interface* yang menghubungkan manusia dan komputer. Untuk mengatasi ketidakserasian antara pelanggan dan pengembang, maka harus dibutuhkan kerjasama yang baik diantara keduanya sehingga pengembang akan mengetahui dengan benar apa yang diinginkan pelanggan dengan tidak mengesampingkan segi-segi teknis dan pelanggan akan mengetahui proses-proses dalam menyelesaikan sistem yang diinginkan. Dengan demikian akan menghasilkan sistem sesuai dengan jadwal waktu penyelesaian yang telah ditentukan (McLeod, 1995).

Kunci agar model *Prototyping* ini berhasil dengan baik adalah dengan mendefinisikan aturan-aturan main pada saat awal, yaitu pelanggan dan pengembang harus setuju bahwa *Prototyping* dibangun untuk mendefinisikan kebutuhan. *Prototyping* akan dihilangkan sebagian atau seluruhnya dan perangkat lunak aktual direalisasikan dengan kualitas dan implementasi yang sudah ditentukan.

3.2 Tahapan Prototyping

Tahapan-tahapan prototyping model

- 1) Pengumpulan kebutuhan
Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasi semua kebutuhan, dan garis besar sistem yang akan dibuat.
- 2) Membangun prototyping
Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan format output).
- 3) Evaluasi prototyping
Evaluasi ini dilakukan oleh pelanggan apakah prototyping yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Jika sudah sesuai maka langkah 4 akan diambil. Jika tidak

prototyping direvisi dengan mengulangi langkah 1, 2, dan 3.

4) Mengkodekan sistem

Dalam tahap ini prototyping yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

5) Menguji sistem

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dicoba dahulu sebelum digunakan. Sistem tersebut diambil dari jadwal kerja dan absensi karyawan.

4. PENUTUP

4.1 Kesimpulan

Algoritma genetika dapat memberikan solusi untuk masalah penjadwalan kerja karyawan dengan menghasilkan suatu jadwal kerja karyawan yang optimal. Suatu jadwal dapat dikatakan optimal apabila tidak terdapat penalti dan mencapai nilai fitness 1.

4.2 Saran

Aplikasi optimasi penjadwalan perkuliahan ini dapat dikembangkan lagi dengan menambahkan batasan jam mengajar serta fitur baru mengenai penyusunan jadwal pengawas dan ruangan untuk Ujian Tengah Semester (UTS) dan Ujian Akhir Semester (UAS).

DAFTAR PUSTAKA

- Risky, Eka. "Algoritma Genetika, Salah Satu Algoritma Optimasi". 26 April 2014.
<http://ekarisky.com/algoritma-genetika-salah-satu-algoritma-optimasi/>
- Hermawanto, Denny. "Algoritma Genetika dan Contoh Aplikasinya". 17 Mei 2007.
<http://www.firman-its.com/2007/05/17/algoritma-genetika-dan-contoh-aplikasinya/>
- Abhique. "Metode Prototyping Dalam Pengembangan Sistem Informasi" 11 Maret 2012.
<http://abhique.blogspot.com/2012/11/metode-prototyping-dalam-pengembangan.html>
- AS, Rosa dan M. Salahuddin 2011, *Modul Pembelajaran Rekayasa Perangkat Lunak*, Modula, Bandung.
- Kadir, Abdul 2000, *Konsep dan Tuntunan Praktis Basis Data*, Andi, Yogyakarta.