

Teknik Desain Optimasi Dengan Algoritma Genetik

Arnawan Hsb

Jurusan Teknik Elektro, Fakultas Teknik
Universitas Malikussaleh, Lhokseumawe

Abstrak--Para engineer mendesain sistem pencarian melalui sejumlah besar solusi yang mungkin untuk menemukan solusi spesifik yang terbaik. Proses pencarian sering memakan waktu dan mahal. Tapi dengan pengeksploitasian proses alami sistem biologi digunakan untuk berkembang dan disesuaikan, design engineer dapat dengan cepat memecahkan kesulitan dari problem desain yang menentang solusi dengan metode optimasi tradisional. Paper ini menjelaskan teknik-teknik dasar dari algoritma genetik dan menunjukkan bagaimana design engineering dapat menggunakan algoritma genetik untuk menyelesaikan problem nyata dalam design engineering.

Paper ini mefokuskan pada penjelasan bagaimana algoritma genetik bekerja. Sebuah contoh uraian singkat pada akhir mempertunjukkan bagaimana pelatihanan tehnik dapat menggunakan tehnik yang kuat ini untuk memecahkan problem dunia nyata dalam design engineering. Contoh masalah desain struktural yang menggunakan algoritma genetik untuk meminimasi permasalahan berat dari sebuah pin-jointed, tapi algoritma genetik dapat dipakai untuk hampir di banyak jenis masalah desain. Beberapa referensi dasar teori lagi diberikan di akhir dalam sebuah penjelasan yang lebih tepat dari detail-detail algoritma genetik.

Kata kunci: design, optimization, genetic algorithm

Abstract – Engineering design system by searching through the large number of possible solutions to discover the best specific solution. The search process is often time consuming and expensive. But by exploiting the natural processes that biological systems use to evolve and adapt, design engineers can often quickly solve otherwise difficult design problems that resist solution by traditional optimization methods. This paper explains the basic technique of the genetic algorithm and show how design engineers can use a genetic algorithm to solve real design engineering problems. This paper focuses on explaining how genetic algorithms work. A brief example at the end

demonstrates how the practicing engineer can use this powerful technique to solve real world problems in engineering design. The example structural design problem uses a genetic algorithm to minimize the weight of a pin jointed frame, but the genetic algorithm can be applied to almost any type of design problem. Some more basic theoretical references are provided at the end for those interested in a more rigorous explanation of the details of genetic algorithms.

Key word : design, optimization, genetic algorithm

A. Terminologi optimasi dasar

Beberapa definisi informal untuk beberapa istilah optimasi dasar agar memastikan bahwa setiap orang memiliki pengertian umum dari diskusi ini. Ini bukanlah definisi yang tepat, tapi seharusnya cukup memadai untuk pekerjaan design engineer. Hanya istilah dasar yang ditampilkan disini untuk mendapatkan pergeseran diskusi ; istilah yang lebih spesifik lagi berhubungan dengan algoritma genetik diperkenalkan seperlunya sepanjang sisa dari paper ini.

Optimasi bertujuan untuk pencarian langsung solusi yang terbaik bagi sebuah masalah yang spesifik. Minimasi berat dengan permintaan struktur teregan adalah suatu contoh dari masalah optimasi. Masalah optimasi adalah mencari salah satu dari minimum atau maksimum dari beberapa masalah barang yang spesifik atau kumpulan barang-barang, tapi karena banyak tujuan atau objektif menjadi tidak umum. Fungsi objektif adalah model perencanaan numerik yang merepresentasikan tujuan pada masalah optimasi. Ketika objektifnya minimisasi, istilah fungsi biaya terkadang digunakan. Algoritma genetik kadangkala menggunakan istilah fungsi

Naskah ini diterima pada tanggal 2 Februari 2009, direvisi pada tanggal 20 Maret 2009 dan disetujui untuk diterbitkan pada tanggal 20 April 2009

kemampuan (*fitness*) dari fungsi objektif dan itu adalah istilah yang digunakan dalam paper ini.

Pada masalah optimasi, variabel-variabel keputusan adalah *variable independen* yang merupakan model yang bergerak selagi pencarian solusi optimum. Sebagai contoh, design engineer harus memilih daerah berpotongan dari suatu anggota struktur sebagai sebuah variabel keputusan. Suatu kekangan menunjukkan suatu batasan tempat atas salah satu *variable* keputusan atau hasil penyelesaian.

Design enginner dapat membatasi tekanan atau pembelokan (*defleksi*) sebagai contohnya. Suatu solusi yang mudah adalah solusi yang tidak melanggar kekangan design engineer yang telah ditempatkan pada masalah optimasi. Wilayah yang memungkinkan terdiri dari semua solusi diambil keseluruhannya. Selebihnya batasan yang bersifat membatasi ditempatkan pada permasalahan dengan design engineer, wilayah yang memungkinkan yang lebih kecil. Masalah yang terlalu dibatasi bisa menghasilkan sebuah wilayah yang memungkinkan kosong, dimana dengan jelas tidak mempunyai solusi yang memungkinkan.

Ruang pencarian meliputi wilayah yang akan dicari selama proses optimasi. Ruang pencarian memasukan semua nilai kemungkinan dari variabel keputusan. Tapi sejak kebanyakan masalah mempunyai kekangan pada penyelesaian masalah, ruang pencarian akan memasukan beberapa penyelesaian yang tidak mudah. Proses pencarian optimasi akan membuang penyelesaian yang di luar wilayah yang memungkinkan.

B. Jenis-jenis masalah optimasi

Jenis-jenis masalah optimasi dapat dikategorikan menurut strategi yang berbeda. Sebagai tujuan dari diskusi ini,

saya menyusun permasalahan dalam sebuah rangkaian kesatuan dari angka numerik yang teratur ke keadaan yang.

Fungsi *fitness* dari sebuah permasalahan dalam angka numerik adalah terus menerus dan dapat didiferensialkan dimanapun dalam ruang pencarian. Fungsi *fitness* juga adalah *mono-modal* yang memiliki suatu titik minimum atau maksimum tunggal yang menggambarkan solusi yang terbaik. Sebuah masalah seperti itu mudah untuk dipecahkan dengan banyak metode optimasi klasik yang diarahkan sebagai metode *hill-climbing*. Metode *hill-climbing* dirancang untuk eksploitasi maksimal dari bentuk *fitness landscape*.

Ada banyak variasi dari metode *hill-climbing*, tapi umumnya terdiri dari pencarian awal solusi tunggal yang memungkinkan dan kemudian mengevaluasi arah kejadiannya dalam setiap variabel keputusan. Selanjutnya pencarian dihubungkan sepanjang garis kurva tercuram untuk menemukan solusi yang relatif terbaik dalam arahan itu. Proses ini diulang dari titik awal hingga tak ada perbaikan yang diperoleh karena titik solusi ada pada *hill-climbing*. Pendekatan secara harfiah dengan *hill-climbing* ini selalu mengambil langkah tercuram (*tinggi*). Pendekatan dengan cara yang sama mengambil yang tercuram dan layak ketika mendekati minimum.

Kamu beruntung jika masalah designmu dapat dipecahkan dengan metode *hill-climbing* karena tak ada yang lebih efisien dalam mengeksplorasi model dari fungsi *fitness* atau *fitness landscape* seperti yang umum disebutkan. Banyak masalah dunia nyata, akan tetapi tidak cocok benar ke dalam solusi jenis ini.

Ketika fungsi *fitness* adalah *multi-modal* pada metode *hill-climbing*, akan diletakkan pada puncak pertama yang ditemukan.

Untuk memecahkan masalah itu design engineer secara khusus memulai proses penyelesaian dari beberapa solusi yang berbeda. Solusi yang terbaik diperoleh dalam rangkaian percobaan itu kemudian menjadi optimum global bagi ruang solusi. Jika design engineer memiliki pengetahuan yang baik mengenai bidang ini berarti masih menjadi pendekatan terbaik bagi optimasi masalah desain.

Pada masalah ekstrim lainnya dari kesatuan rangkaian masalah adalah fungsi objektif yang muncul secara acak. Penampilan dari pengacakan ini bisa disebabkan oleh ilmu pengetahuan yang tidak lengkap dari parameter yang mengatur fungsi fitness atau masalah yang benar-benar dapat secara acak. Dalam satu kasus fungsi fitness muncul agar tidak mempunyai hubungan yang konsisten dengan variabel keputusan. Masalah pengacakan dipecahkan dengan sebuah eksplorasi yang lengkap dari ruang solusi. Metode itu dinyatakan sebagai penghitungan yang lengkap.

Metode penghitungan yang lengkap, walaupun efektif tetapi jarang dilaksanakan. Bahkan computer-komputer modern yang murahpun, biaya perhitungan untuk pencarian yang mendalam adalah sangat tinggi kecuali masalah yang paling sederhana. Kebanyakan masalah dunia yang nyata mempunyai banyak variabel keputusan. Sejak itu ruang pencarian merupakan produk kartesian dari semua variabel keputusan, ruang pencarian menjadi hebat bagi banyak penyelidikan mendalam yang tidak bisa disepelekan.

Dan jika masalah secara dimensi tidak terlalu buruk, kebanyakan masalah nyata dunia adalah bukan masalah nilai integer. Untuk pencarian yang mendalam bagi angka nyata yang terbaik berarti design engineer harus memutuskan bagaimana menghaluskan sebuah butiran untuk

digunakan dalam pencarian. Sebagai contoh untuk mencari antara 20-25, haruskah kenaikan percobaan dengan 0.5, 0.1 atau 0.01 atau bahkan gradasi yang lebih halus dibutuhkan? Membuat ukuran langkah pencarian yang terlalu besar bisa membuatmu melewati solusi yang lebih baik, akan tetapi ukuran langkah yang terlalu kecil bisa menghabiskan sumber-sumber perhitungan dengan memperpanjang pencarian yang tiada gunanya. Design engineer harus membuat semua keputusan ini sebelum pencarian dikerjakan.

Dua perbedaan besar antara keadaan baik dan fungsi fitness yang acak membohongi masalah dunia nyata. Masalah optimasi yang nyata bisa terputus pada beberapa titik dan sering terputus pada optimum global karena ada kekangan. Masalah optimasi nyata tidak boleh dapat berlainan, boleh memiliki beberapa nilai maksimum relatif, dan boleh dibatasi dengan rapat sekali pada kondisi kekangan yang banyak. Tapi kondisi ini tidak diacak; pencarian dengan arah yang benar biasanya menuntun menuju perbaikan pada fungsi fitness. Ini adalah jenis masalah yaitu algoritma genetik yang mudah dan sering dapat dipecahkan.

C. Kerja Algoritma Genetik

Algoritma genetik kira-kira meniru bagaimana sistem biologis alami berkembang dan beradaptasi melalui proses seleksi alami. Banyak yang menganalogikan sebagai sistem biologis yang digunakan untuk menjelaskan operasi dari algoritma genetik.

Pendekatan algoritma genetik berbeda dengan hill-climbing yang lebih tradisional dan metode penghitungan yang lengkap dalam 4 cara dasar.

Pertama, algoritma genetik klasik menyandikan nilai dari variabel keputusan

dalam sebuah benang yang disebut kromosom. Penyandian dan penterjemahan dari benang dilengkapi dengan design engineer, tapi komputer menafsirkan benang dengan sederhana dari angka biner. Tiap bit dapat dipikirkan sebagai sebuah gen dalam kromosom.

Kedua, algoritma genetik menggunakan sebuah populasi dari individu untuk melaksanakan pencarian. Tiap individu mewakili satu solusi yang mungkin pada masalah itu. Kromosom individu menyandikan satu kumpulan dari variabel keputusan dan juga hasil dalam sebuah titik tunggal dalam ruang solusi.

Ketiga, evaluasi dari proses pencarian didasarkan pada kecocokan sendiri, yang menjaga pencarian tetap fokus pada objek yang sebenarnya dan dicari dengan design engineer. Permulaannya tidak digunakan jadi biaya dari perhitungan awal tidak dilakukan. Sejak itu awalnya tidak digunakan, metode itu tidak dipengaruhi oleh fungsi yang terputus.

Keempat, algoritma genetik bukanlah sebuah metode pencarian secara acak, tapi dia menggunakan proses acak untuk transisi dari satu keadaan pencarian menuju keadaan lainnya. Proses secara acak memberikan algoritma genetik pada sebuah keseimbangan yang bagus antara eksplorasi yang luas dari ruang pencarian dan eksploitasi keistimewaan dari fitness landscape. Mengingat bahwa tehnik penghitungan adalah sangat bagus pada eksploitasi tetapi sama sekali tidak memanfaatkan ruang pencarian. Metode hill-climbing walaupun sangat bagus pada eksploitasi tapi tidak mengeksplorasi seluruh ruang pencarian.

Algoritma genetik menggunakan operator seleksi, rekombinasi dan mutasi pada populasi dari individu untuk melakukan pencarian. Populasi secara acak dibentuk

pada awal pencarian. Fitness digunakan untuk memilih individu dari generasi sekarang menjadi yang terdepan pada generasi selanjutnya. Individu-individu ini direkombinasikan dan boleh dimutasikan ke bentuk generasi selanjutnya. Proses ini dilanjutkan sampai tidak ada lagi perubahan individu yang terbaik dari populasi.

Seleksi dimulai dengan penentuan fitness relatif dari setiap individu dengan memperhitungkan fitness individu dibagi dengan jumlah total fitness dari seluruh populasi. Kemudian fitness kumulatif dijumlahkan pada setiap individu sebagai jumlah dari fitness relatif bagi semua anggota yang naik dihitung. Fitness kumulatif dengan cara demikian dinormalisir pada seluruh populasi menuju maksimum 1.0 bagi individu terakhir.

Populasi dapat dipikirkan sebagai bentuk sebuah roda roulette dengan celah proporsional kepada fitness individu yang relatif ke sisa populasi. Sebuah angka acak di antara 0-1.0 selanjutnya dibangkitkan dan individu dengan fitness kumulatif yang terbatas dengan nilai acak akan terpilih. Proses pemilihan ini berlanjut sampai sebuah populasi baru terbentuk. Umumnya, individu-individu itu dengan nilai fitness yang lebih tinggi lebih mungkin untuk dipilih, tetapi ada juga suatu elemen dari pilihan acak.

Dengan cara yang sama, perkalian individu yang mempunyai kromosom yang sama dan kecocokan yang sama juga akan mempunyai sebuah kesempatan yang lebih baik untuk dipilih.

Pertama-tama populasi dari individu yang baru dipilih untuk memulai rekombinasi. Algoritma genetik bergerak melalui populasi dengan berpasangan dan dapat menentukan secara acak apabila setiap pasangan individu akan direkombinasi.

Jika itu adalah kasus sebuah titik acak sepanjang pasangan kromosom yang dipilih dan merupakan sisa dari setiap kromosom ke titik seleksi yang benar maka antara dua kromosom akan ditukar. Dua individu yang baru dibentuk yang merupakan sebuah rekombinasi (penggabungan kembali) gen-gen dalam dua kromosom yang asli.

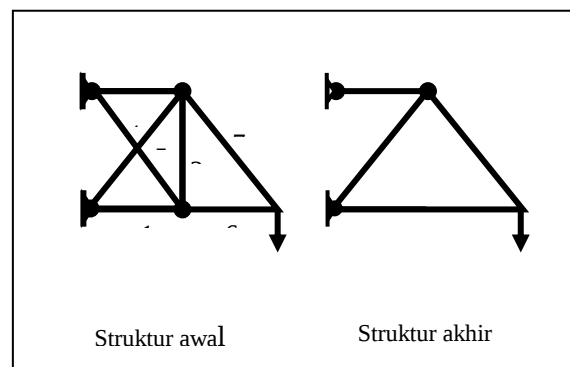
Akhirnya, masing-masing gen dari tiap kromosom pada tiap individu secara acak disa dimutasi agar menunjukkan perbedaan tambahan dalam populasi. Probabilitas sebuah mutasi biasanya rendah, tapi design engineer dapat mengendalikan ini dan semua probabilitas lainnya untuk menyelaraskan dengan baik proses pencarian.

Sejak itu kebanyakan masalah dunia nyata mempunyai kekangan, algoritma genetik memerlukan sebuah mekanisme untuk penggunaan masalah kekangan. Sebuah fungsi pinalti adalah suatu cara mudah untuk membatasi keadaan dari fungsi fitness ke wilayah yang memungkinkan dengan penggunaan sebuah hukuman bagi pelanggaran sebuah pembatas masalah. Sebuah fungsi hukuman mengurangi nilai fungsi kecocokan ketika sebuah pembatas dilanggar. Sebuah fungsi hukuman yang baik menurunkan nilai fitness secara tajam pada batas pembatas pembentuk sebuah jurang dalam landscape fitness. Mengingat bahwa terputusnya hal itu tidak mengganggu algoritma genetik jadi tepi jurang yang lebih tajam lebih baik.

Hasil yang bagus dapat diperoleh dengan mengurangi nilai fitness yang tidak dibatasi dengan pinalti yang meningkat secara eksponensial sebagai peningkatan pelanggaran dari kekangan. Pelanggaran yang dikuadratkan dikurangi dengan nilai fungsi fitness dapat dikalikan dengan :

$$e^{-I_{hukumani}}$$

Sejak itu algoritma genetik menggunakan proses acak untuk transisi dari satu generasi ke generasi berikutnya, algoritma genetik tidak dapat ditetapkan. Untuk menyatakannya tidak mungkin akan dihasilkan jawaban yang sama dalam dua usaha lainnya pada masalah yang sama. Jawaban itu akan menjadi yang terbaik yang dapat ditemukan oleh algoritma genetik. Algoritma genetik melakukan penjelajahan secara luas dan mengeksplotasi fitness landscape untuk menemukan sebuah solusi yang terbaik.



Gambar 1. Proses acak algoritma genetik

D. Contoh struktur sederhana

Algoritma genetika telah dicoba untuk minimisasi berat dari beberapa tiang penyangga sederhana sebagaimana tujuh anggota tiang penunjang yang ditunjukkan dalam gambar. Algoritma genetika memperoleh hasil yang sama untuk setiap kasus. Hasil dari tujuh anggota dijelaskan disini secara mendetail (rinci).

Analisis program sebuah elemen sederhana yang terbatas dijalankan dengan algoritma genetik untuk menganalisa contoh struktur. Barang-barang material yang sama digunakan untuk semua anggota, tapi sebuah tekanan beban anggota yang diperkenankan dibatasi menjadi setengah dari sebuah tegangan anggota. Daerah potongan melintang dari tujuh anggota dipilih sebagai masalah desain variabel keputusan. Struktur dianggap menjadi pin-

jointed, jadi belokan tidak dipertimbangkan dalam masalah uji. Beban anggota yang diizinkan menjadi pembatas pada masalah uji. Ketika kebanyakan beban bagian dalam anggota melebihi tiap nilai tegangan atau tekanan yang diizinkan, fitness dikurangi dengan sebuah fungsi pinalti eksponen.

Sebuah populasi dari 50 individu digunakan. Probabilitas dari rekombinasi adalah 0.8 dan probabilitas dari mutasi adalah 0.06. Daerah anggota potongan melintang dikodekan kepada jarak yang cukup tinggi untuk menjamin bahwa sedikitnya ada satu solusi yang mudah.

Solusi pertama menyatakan bahwa anggota 4 dalam gambar tidak diperlukan dan dia dapat dihilangkan dari struktur. Ini ditandai dengan keadaan daerah potongan melintang hampir menuju nol. Anggota ini kemudian dipindahkan dari model struktur dan masalahpun berjalan lagi.

Analisis berikutnya menyatakan bahwa anggota 2 dapat dihilangkan untuk alasan yang sama seperti anggota 4 yang dipindahkan lebih cepat. Penghapusan anggota 2 harus dilakukan meskipun panjang pasangan yang terbatas dari tali yang lebih rendah dan dapat menyebabkan penghindaran masalah. Tali yang lebih pendek dari tiang penopang tidak dianalisis untuk tekuk kolom pada model ini.

Jika anggota 2 dan 4 dihilangkan, sebuah jumlah maksimal yang tepat dapat ditemukan secara analisis, sejak itu tidak berlebihan kalau anggota-anggota itu ada. Solusi pertama dengan ketujuh anggota dalam tiang penompang adalah dengan 10% dari jumlah maksimal yang benar. Hasil yang sama ini diulang dengan struktur yang lebih kompleks. Walaupun algoritma genetik tidak menjamin sebuah solusi yang optimal, tentu saja dia

melakukan sebuah hal untuk mendekati nilai optimal.

E. Kesimpulan

Algoritma genetik mempunyai beberapa keuntungan yang sangat kuat disamping kedua cara hill-climbing klasik dan cara penjumlahan yang lengkap. Pertama algoritma genetik memberikan sebuah keseimbangan yang bagus dari eksploitasi dan eksplorasi pada ruang pencarian. Itu artinya solusi-solusi yang efisien belum dijelajahi dari seluruh ruang yang disediakan, jadi pemecah masalah mungkin kurang mampu untuk memutuskan pada sebuah jumlah maksimum lokal yang relatif.

Kedua algoritma genetik tidak bermasalah pada sebuah keadaan terputus dalam solusi ruang. Masalah didapatkan secara matematis dari masalah dunia nyata yang tidak menentu. Design engineer tidak perlu menciptakan fiksi matematis yang rumit, biarpun untuk membodohi pemecah masalah ke dalam pemikiran masalah yang sebenarnya adalah dalam keadaan baik.

Akhirnya algoritma genetik cukup mencari solusi yang sangat baik daripada solusi yang sangat paling baik. Ini sebenarnya adalah sebuah kekuatan yang mencegah algoritma genetik jatuh ke lubang dengan pikiran dangkal secara matematika atau mengikuti puncak sebuah bintik panas lokal.

Daftar Pustaka

- [1] Goldberg, D. E., Genetic Algorithm in search, Optimization, and Machine Learning, Addison Wesley, 1989.
- [2] Michalewicz, Z., Genetic Algorithm + Data Structures = Evolution Programs Third Edition, Springer, 1996.
- [3] Dinger, R. H., Engineering Design Optimization with Genetic Algorithm, IEEE Northcon Conference 21-23 October 1998, Amerika, 1998