Metaheuristik Search dalam Penyelesaian Job Shop Scheduling

Didik Supriadi
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganeça 10, Kotamadya Bandung 40135
e-mail: 13517069@std.stei.itb.ac.id

ABSTRAK

Metaheuristik search kini dikenal sebagai salah satu teknik untuk optimalisasi yang cukup efektif setelah dilakukan beberapa percobaan komputasional. Algoritma Metaheuristik search mampu bersaing dengan teknik-teknik terkenal lainnya disebabkan fleksibilitasnya yang bisa menyaingi prosedur klasik lainnya. Karena sudah banyak terbukti bahwa Metaheuristik search adalah algoritma yang cukup efektif maka saat ini banyak sekali permasalahan dari berbagai bidang yang bisa menggunakan implementasi algoritma Metaheuristik search untuk mendapatkan solusi optimal. Salah satu contoh permasalahan yang juga dapat diselesaikan dengan algoritma Metaheuristik search adalah permasalahan job shop scheduling dimana permasalahan ini bertujuan untuk meminimasi waktu untuk menyelesaikan sederetan operasi dari job yang ada. Makalah ini akan membahas sedikit lebih mendalam mengenai algoritma Metaheuristik search dan disertai dengan contoh implementasinya dalam menyelesaikan permasalahan job shop scheduling.

Kata kunci—algoritma metaheuristic search, job-shop problem,scheduling

I. PENDAHULUAN

Pada dasarnya, job-shop problem adalah salah satu contoh dari permasalah penjadwalan yang paling dasar (basic scheduling problems) di samping beberapa contoh lain seperti open shop problem atau flow job problem.

Dalam permasalahan job-shop, terdapat beberapa pekerjaan yang memiliki operasinya sendiri-sendiri yang hanya dapat dijalankan di sebuah mesin pada suatu waktu tertentu. Objektif dari permasalahan job-shop ini adalah untuk mendapatkan sebuah jadwal operasi pekerjaan yang memakan waktu yang paling singkat yang mungkin dilakukan. Banyak pendekatan yang dapat dilakukan untuk menyelesaikan masalah ini, dan salah satu caranya adalah menggunakan pendekatan algoritma metaheuristic search. metaheuristic search ini sudah mendapatkan pengakuan luas sebagai salah satu pendekatan yang paling efektif dalam menghasilkan solusi yang berkualitas tinggi untuk masalah job-shop. Namun ternyata, masih

sedikit pihak yang mengetahui mengapa metaheuristic search ini sangat efektif untuk penyelesaian masalah job-shop, dan dalam kondisi yang bagaimana sehingga kita bisa mendapatkan hasil yang paling optimal dari metaheuristic search ini.

Makalah ini akan membahas mengenai kerangka dasar algoritma metaheuristic Search beserta sebuah contoh implementasi penggunaan algoritma metaheuristic Search dalam menyelesaikan permasalahan job shop scheduling.

II. Permasalahan Job Shop Scheduling

Permasalahan job shop scheduling problem(JSP) dapat dibentuk sebagai sebuah himpunan J, dimana J terdiri dari n jobs (pekerjaan) dan sebuah himpunan M yang terdiri dari m mesin. Setiap pekerjaan Ji memiliki ni subtasks(disebut operasi), dan setiap operasi Jij harus dijadwalkan pada mesin yang sudah ditetapkan sebelumnya, μij∈M untuk lamanya waktu yang tetap, dij, tanpa interupsi. Tidak ada mesin yang boleh memproses lebih dari satu operasi pada sebuah waktu tertentu, dan setiap operasi Jij∈Ji harus selesai sebelum operasi selanjutnya pada job tersebut (Ji(j+1)) dimulai. Suksesor dari operasi x pada jobnya dituliskan sebagai SJ[x], dan suksesor dari x pada mesinnya dituliskan sebagai SM[x]. Demikian juga dengan predecessor yang dituliskan sebagai dan PJ[x] dan PM[x]. Setiap operasi x memiliki waktu mulai yang dinotasikan sebagai r2, dan waktu tail yang dinotasikan dengan t2, dimana itu merupakan jalur terpanjang dari waktu x untuk bisa selesai sampai ke akhir.

Untuk menyelesaikan masalah ini, untuk setiap mesin, kita harus menemukan sebuah urutan dari operasi-operasi yang akan dijadwalkan
Yang bisa mengoptimalkan fungsi objektif. Ada beberapa fungsi objektif yang sering digunakan pada permasalahan ini. Sejauh ini, fungsi objektif yang paling umum adalah meminimasi waktu total untuk menyelesaikan semua pekerjaan (tasks). Objektif ini penggunaannya cukup luas karena fungsi tersebut cukup mewakili banyak persoalan di dunia industri, dan sangat mudah untuk dikomputasi secara efisien.

III. Algoritma Metaheuristik Search

Metode pencarian metaheuristic berprinsip pada penggunaan memori sebagai elemen esensial dalam pencariannya, karena pencarian metaheuristic tidak hanya menyimpan nilai sebuah solusi terbaik seperti kebanyakan metode pencarian, namun juga menyimpan informasi selama pencarian melalui solusi terakhir yang dikunjungi. Sebuah informasi akan digunakan sebagai petunjuk untuk bergerak dari i ke solusi selanjutnya dalam N(i). Penggunaan memori sebagai pembatas dalam pemilihan beberapa subset dari N(i) dengan mencegah pergerakan ke beberapa solusi tetangga.

Lebih tepatnya, kita akan menyadari bahwa struktur N(i) sebagai solusi i akan berupa variabel dari satu iterasi ke iterasi lainnya. Oleh karena itu metode pencarian metaheuristic dapat disebut sebagai kelas dari prosedur-prosedur yang disebut dynamic neighborhood search techniques.

Secara formal,kita dapat menganggap masalah optimalisasi dalam cara berikut:
Diberikan sebuah himpunan solusi S dan sebuah fungsi f : S, temukan solusi i* dalam S

sehingga f(i*)dapat diterima dengan beberapa kriteria. Secara umum kriteria untuk dapat diterima sebagai solusi i* harus memenuhi f(i*)=f(i) untuk setiap i dalam S. Dalam situasi metode pencarian metaheuristic akan menjadi sebuah algoritma minimisasi secara pasti yang menyediakan proses eksplorasi yang menjamin setelah sejumlah langkah-langkah terhingga i* dapat dicapai.

Walaupun tidak ada jaminan bahwa i* akan diperoleh, metode pencarian metaheuristic dapat secara sederhana dipandang sebagai prosedur heuristicumum secara ekstrim. Karena metode pencarian metaheuristic akan mencakup beberapa teknik heuristic dalam aturan-aturan operasi di dalamnya, maka akan lebih pantas untuk menggolongkan metode pencarian metaheuristic sebagai metaheuristic. Perannya akan sering dijadikan sebagai petunjuk dan sebagai orientasi prosedur pencarian lainnya yang lebih lokal.

Untuk mendalami lagi prinsip kerja metode pencarian metaheuristic, kita dapat merumuskan metode menurun klasik (classical descent method) dalam beberapa langkah, yaitu:

1. Memilih sebuah solusi awal i dalam S
2. Membangkitkan subset V* sebagai solusi dalam N(i)
3. Mencari j ‘terbaik’ dalam V* dan menetapkan i=j
4. Jika f(j)=f(i) maka berhenti, namun jika tidak kembali ke langkah ke-2

Dalam metode menurun secara umum akan dapat langsung ditetapkan bahwa V* = N(i).Tetapi hal ini seringkali membutuhkan waktu yang lama. Untuk itulah cara pemilihan V* yang tepat seringkali dijadikan sebagai peningkatan yang penting dalam metode pencarian.

Pada kasus yang berlawanan, akan ditetapkan |V*|=1. Hal ini akan menurunkan fase pemilihan j ‘terbaik’. Solusi j akan diterima jika f(j)=f(i), sebaliknya hal ini akan diterima dengan kemungkinan tertentu yang bergantung pada nilai-nilai f pada i dan j serta pada sebuah parameter yang disebut temperatur.

Tidak ada temperatur dalam metode pencarian metaheuristic, namun pemilihan V* akan menjadi hal yang penting guna mendefinisikannya dalam setiap langkah di mana akan terjadi penggunaan memori secara sistematis untuk memanfaatkan informasi yang ada di luar fungsi f dan lingkungan N(i).

Sebagai pengecualian pada kasus-kasus istimewa, penggunaan prosedur menurun (descent procedure) secara umum lebih rumit karena kita akan terperangkap pada sebuah minimum lokal yang mungkin masih jauh dari minimum global.

Maka untuk proses iterasi dalam eksplorasi apapun sebaiknya dalam beberapa hal juga menerima langkah-langkah yang tidak akan memberikan perkembangan dari i ke j dalam V*(misal f(j)>f(i)). Metode pencarian metaheuristic secara berbeda memilih j ‘terbaik’ dalam V*.

Selama pergerakan yang tidak memberi perkembangan itu mungkin, resiko pengunjungan kembali sebuah solusi atau lebih umumnya disebut sebagai siklus mungkin untuk terjadi. Dalam hal inilah penggunaan memori sangat diperlukan untuk mencegah terjadinya pergerakan ke solusi yang telah dikunjungi. Jika memori seperti itu diperkenalkan, maka kita dapat menganggap struktur N(i) tergantung pada pengelilingan yang merupakan pengulangan k. Jadi kita dapat merujuk ke N(i,k) daripada N(i).

Dengan perujukan ini kita dapat mencoba untuk melakukan peningkatan algoritma menurun dalam beberapa hal untuk lebih mendekatkan metode ini ke prosedur dalam metode pencarian metaheuristic secara umum. Hal ini dapat didefinisikan dalam beberapa langkah (catatan i* adalah solusi 'terbaik' yang ditemukan dan k adalah penghitung dalam pengulangan) :

1. Memilih solusi awal I dalam S. Tetapkan i*=I dan k=0.
2. Tetapkan nilai k=k+1 dan membagkitkan subset V* sebagai solusi dalam N(i,k)
3. Pilih j 'terbaik' dalam V* dan tetapkan i=j.
4. Jika f(i) < f(i*) maka tetapkan i*=i.
5. Jika kondisi berhenti ditemukan, maka proses dihentikan, sedangkan jika belum kembali ke langkah 2.

Amati bahwa langkah-langkah pada metode penurunan klasik termasuk dalam formula ini (kondisi berhenti secara sederhana jika f(i)=f(i*)dan i* akan selalu menjadi solusi akhir). Selain itu kita juga dapat mempertimbangkan penggunaan f yang telah dimodifikasi daripada f yang dalam beberapa keadaan di deskripsikan kemudian.

Dalam metode pencarian metaheuristic, kondisi berhenti dapat berupa:

- N(i,k+1) = tidak terdefinisi
- k lebih besar daripada bilangan maksimum pada pengulangan
- banyaknya pengulangan selama peningkatan terakhir i* lebih besar dari bilangan tertentu.
- Pembuktian dapat diberikan daripada solusi optimum yang telah didapatkan.

Selama aturan-aturan berhenti ini memungkinkan untuk memiliki beberapa pengaruh dalam prosedur pencarian dan pada hasil-hasilnya, penting untuk menyadari bahwa pendefinisian N(i,k) dalam tiap pengulangan k dan pemilihan V* adalah hal yang krusial.

Definisi dari N(i,k) menyatakan secara tidak langsung bahwa beberapa solusi yang telah dikunjungi dihapus dari N(i), mereka dianggap sebagai solusi-solusi metaheuristic yang harus dihindari dalam pengulangan selanjutnya. Sebagai contoh, pemeliharaan pada pengulangan k sebuah list T(list metaheuristic) pada solusi yang telah dikunjungi terakhir |T| akan mencegah terjadinya siklus pada ukuran paling banyak sebesar |T|. Pada kasus ini, kita bisa mengambil N(i,k)=N(i)-T. Namun list T kemungkinan tidak dapat digunakan secara praktis. Oleh karena itu, kita akan mendeskripsikan proses eksplorasi pada S dalam masa pergerakan dari satu solusi ke solusi lainnya. Untuk setiap solusi I dalam S, kita dapat mendefinisikan M(i) sebagai himpunan gerak yang dapat digunakan oleh i untuk memperoleh solusi baru j (notasi: j=i∇m). Secara umum kita dapat menggunakan gerakan-gerakan yang dapat dibalik. Untuk setiap m terdapat gerakan m-1 sehingga (i∇m) ∇m-1 = i.

Jadi daripada mempertahankan list T dari solusi-solusi yang telah dikunjungi, kita dapat secara sederhana memelihara jalur gerak terakhir |T|atau gerak balik terakhir |T| yang diasosiasikan dengan gerakan-gerakan yang sebenarnya telah dilakukan. Maka dengan jelas bahwa pembatasan yang ada adalah kehilangan informasi, dan hal itu tidak menjamin tidak terjadinya siklus dengan panjang paling banyak |T|.

Untuk kepentingan efisiensi, maka diperlukan penggunaan beberapa listTγ dalam satu waktu maka beberapa unsur pokok tγ (dari i atau dari m) akan diberikan metaheuristic status

untuk mengindikasi bahwa unsur pokok ini sedang tidak diperbolehkan untuk terlibat dalam pergerakan. Secara umum pergerakan untuk metaheuristic status adalah fungsi metaheuristic status pada unsur-unsur pokoknya yang dapat diubah pada setiap pengulangan.

Suatu gerakan m (digunakan pada solusi i) akan menjadi gerak metaheuristic jika semua kondisi telah dipenuhi. Ilustrasi dari konsep ini dapat lebih dimengerti pada penggunaan metode pencarian metaheuristic pada penjadwalan pekerjaan (Job Scheduling).

Kekurangan lain dari simplifikasi pada list metaheuristic (penggantian solusi menjadi gerak) adalah fakta bahwa kita mungkin memberikan status metaheuristic ke solusi-solusi yang belum dikunjungi. Maka kita pun dapat memaksakan peregangan dari metaheuristic status. Kita dapat menguasai metaheuristic status saat beberapa metaheuristic solusi akan terlihat menarik. Hal ini akan dilakukan dimaksudkan untuk tingkat kondisi aspirasi (aspiration level conditions)

Gerak metaheuristic m digunakan pada solusi i yang mungkin tampak menarik karena itu diberikan sebagai contoh sebuah solusi yang lebih baik dari pada yang telah ditemukan. Kita akan dapat menerima m tanpa memperhatikan statusnya. Kita akan melakukan hal tersebut jika m memiliki tingkat aspirasi (aspiration level) a(i,m) yang lebih baik daripada permulaan A(i,m).

Sekarang kita dapat mendefinisikan karakteristik dari prosedur pencarian metaheuristic dalam langkah-langkah berikut, antara lain:

1. Memilih solusi awal i dalam S. Tetapkan i*=I dan k=0.
2. 2. Tetapkan k=k+1 dan bangkitkan sebuah subset V* dari solusi dalam N(I,k) sehingga salah satu dari kondisi metaheuristic tγ yang melanggar (γ=1,2,...,t)atau setidaknya satu kondisi aspirasi aγyang memiliki (γ=1,2,...,a).
3. 3. Pilih j terbaik melalui j=i∇m dalam V*dan tetapkan i=j.
4. 4. Jika f(i)<f(i*) maka tetapkan i*=i.
5. 5. Perbaharui kondisi metaheuristic dan aspirasi.
6. 6. Jika kondisi berhenti ditemukan, maka proses dihentikan. Jika tidak, kembali ke langkah 2.

Hal-hal di atas dapat dikatakan sebagai bahan-bahan dasar dari metode pencarian metaheuristic yang nantinya akan digunakan untuk memecahkan masalah penjadwalan pekerjaan (job scheduling) pada pembahasan selanjutnya.

IV. Penerapan Algoritma metaheuristic Search dalam Job Shop Scheduling Problem

metaheuristic Search adalah sebuah meta-heuristic untuk kendali local search yang secara deterministik mencoba untuk menghindari solusi yang baru-baru saja dikunjungi. Secara spesifik, algoritma ini mengelola sebuah metaheuristic List yang berisi perpindahan yang terlarang. List ini mengikuti aturan LIFO dan biasanya sangat pendek (panjangnya biasanya sebesar O(N), dimana N adalah jumlah total dari operasi). Setiap saat ada langkah yang diambil, maka langkah itu akan ditempatkan dalam metaheuristic list.

Salah satu komponen paling penting dalam algoritma metaheuristic search adalah Neighborhood function (NC), dimana fungsi tersebut secara signifikan akan mempengaruhi running time dan kualitas dari solusi yang dihasilkan. Dalam

metaheuristic list, elemen yang ditempatkan di dalam list adalah busar yang dibalik, dan langkah dianggap metaheuristic jika ada komponen dalam busar adalah metaheuristic.

V. Kesimpulan

Setelah meneliti metode pencarian metaheuristic dapat disimpulkan bahwa walaupun metode ini belum diketahui secara pasti bekerja optimal dalam keadaan apa dan mengapa metode ini bekerja dengan baik, namun dalam melakukan pencarian untuk memecahkan beberapa masalah terutama job scheduling problem metode ini terbukti cukup efektif dibandingkan dengan metode-metode lainnya. Penelitian ini telah menunjukkan bahwa ada kemungkinan untuk memakai algoritma metaheuristic search dan melakukan beberapa penyesuaian untuk bisa menghasilkan solusi yang layak pada sebuah kelas yang lebih lebar dari sebuah permasalahan.

VI. Referensi

-

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 April 2019

ttd

Didik Supriadi

13517069