Algoritma Branch and Bound untuk Masalah Penjadwalan pada Mesin Paralel

Jeffrey Setiawan Sutanto[1], Ronny Hendrawan[2], Yosep Kurniawan[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13010@students.if.itb.ac.id[1], if13039@students.if.itb.ac.id[2], if13059@students.if.itb.ac.id[3]

Abstrak

Di dalam makalah ini, algoritma Branch and Bound dipresentasikan untuk menyelesaikan masalah minimisasi waktu penyelesaian maksimum C_{max} pada mesin paralel tidak berelasi dengan batasan kelayakan *job preemption* (interupsi yang dilakukan pada pekerjaan yang sedang diproses saat itu, untuk melakukan pekerjaan yang lain) tidak diperbolehkan. Suatu kostumisasi batas bawah (*Lower Bound*), strategi pencarian dan pencabangan dikembangkan untuk algoritma Branch and Bound ini. Faktor kelayakan mesin juga dimasukkan untuk merepresentasikan persentasi kelayakan pekerjaan pada seluruh mesin. Masalah yang muncul dengan mesin yang berbeda, pekerjaan yang berbeda, dan faktor kelayakan dipecahkan di sini. Algoritma Branch and Bound mempunyai kemampuan untuk menyelesaikan 8 buah mesin dan 40 buah pekerjaan dalam waktu yang masih masuk akal. Untuk mengevaluasi performansi dari algoritma ini, sejumlah simpul diperiksa dengan suatu ukuran performansi. Performansi dari algoritma Branch and Bound meningkat seiring dengan meningkatnya faktor kelayakan.

***Kata kunci**: Branch and Bound , Mesin bersyarat, Penjadwalan, Perentangan waktu*

1. Pendahuluan

Penjadwalan pekerjaan pada mesin paralel yang tidak berelasi adalah masalah yang umum, di mana sekumpulan mesin tersedia untuk memproses suatu pekerjaan secara simultan. Masalah pada mesin paralel yang tidak berelasi adalah generalisasi dari mesin paralel yang seragam dan identik. Ketika beberapa mesin identik, pemrosesan suatu pekerjaan tertentu memakan waktu yang sama. Dalam hal mesin-mesin yang seragam, kecepatan di mana suatu mesin memproses suatu pekerjaan adalah proporsional dengan mesin lainnya berdasarkan faktor kecepatan. Pada kasus mesin tidak berelasi yang umum, waktu pemrosesan suatu pekerjaan pada suatu mesin tidak bergantung dengan waktu pemrosesan yang dibutuhkan oleh mesin lain. Tujuan utama dari pembuatan makalah ini adalah mengembangkan algoritma Branch and Bound untuk menemukan solusi optimal pada masalah mesin paralel tidak berelasi dengan batasan faktor kelayakan mesin. Dimana tujuan utama dari algoritma ini adalah meminimisasi waktu penyelesaian maksimum. Ketika batasan kelayakan mesin diperhitungkan, beberapa pekerjaan hanya dapat diproses pada mesin dengan batasan kelayakan tertentu. Masalah dengan mesin paralel tidak berelasi ini mempunyai banyak aplikasi lainnya. Sebagai contoh, di dalam suatu manufaktur, terdapat banyak mesin yang dapat memproses suatu pekerjaan dengan kecepatan yang berbeda sesuai dengan tknologi mesin yang digunakan. Sebagai contoh lainnya, batasan kelayakan mesin dapat berada di penjadwalan pesawat terbang. Pesawat terbang dijadwalkan untuk masuk ke dalam garasi. Dalam hal ini pesawat adalah suatu pekerjaan dan garasi sebagai mesin. Bergantung apada ukuran badan pesawat terbang, beberapa garasi tidak dapat mengakomodasi pesawat terbang berbadan besar.

Makalah ini diorganisasikan sebagai berikut : Bagian 2 membicarakan formula pemrograman integer untuk menyelesaikan masalah yang ada. Bagian 3 berisi deskripsi dari Branch and Bound yang terdiri dari strategi pencarian dan pencabangan, juga penurunan rumus untuk ekspresi batas bawah (*Lower Bound*). Bagian 4 berisi hasil dari komputasional. Pada akhirnya, bagian 5 berisi kesimpulan yang diperoleh dari pembuatan makalah ini.

2. Formula Pemrograman Integer

Masalah penjadwalan pada mesin paralel tidak berelasi diformulasikan sebagai program integer campuran kosong-satu (*MIP : mixed integer program*)

Minimalisasi C_{max}
Berdasarkan pada :

$$C_{\max} - \sum_{j=1}^{n} P_{jk} X_{jk} \geq 0 \qquad \forall k \in M_j$$

$$\sum_{k \in M_j} X_{jk} = 1 \qquad 1 \leq j \leq n$$

$$X_{jk} \in \{0,1\} \qquad 1 \leq j \leq n \text{ dan } k \in M_j$$

Dimana
X_{jk} bernilai 1 jika pekerjaan j diserahkan kepada mesin k dan bernilai 0 jika sebaliknya
P_{jk} adalah waktu pemrosesan pekerjaan j pada mesin k
M_j adalah kumpulan mesin yang layak untuk melakukan proses pada pekerjaan j.
C_{max} adalah waktu maksimum yang dibutuhkan untuk melakukan suatu proses kerja

MIP ini dapat digunakan secara optimal untuk menyelesaikan masalah yang terdapat di dalam makalah ini. Akan tetapi, ketika jumlah mesin dan pekerjaannya bertambah banyak, MIP menjadi terlalu besar untuk dipecahkan dalam waktu yang terbatas. Oleh karena itu, algoritma branch and bound dikembangkan untuk menyelesaikan permasalahan ini dengan lebih efisien untuk jumlah mesin dan pekerjaan yang besar.

3. Branch and Bound

Dalam masalah yang dibahas di sini, n buah pekerjaan diproses dengan menggunakan m buah mesin tak berelasi. Setelah sekumpulan pekerjaan diserahkan kepada mesin, pencarian urutan pekerjaan tidak diperlukan lagi dalam mesin tersebut, karena yang menjadi tujuan utama adalah untuk meminimisasi C_{max} dan tidak tergantung dengan waktu pengurutan.. Oleh karena itu, algoritma branch and bound dikembangkan untuk menentukan penyerahan pekerjaan secara optimal kepada mesin.

Branch and Bound adalah suatu prosedur yang paling umum untuk mencari solusi optimal pada masalah optimasi kombinatorial seperti masalah penjadwalan. Di dalam algoritma Branch and Bound, terdapat tiga buah bagian utama, yaitu : ekspresi batas bawah (*Lower Bound (LB)*), strategi pencarian dan pencabangan (*branching*). Di dalam prosedur ini, suatu maslah dipecah menjadi beberapa submasalah yang merepresentasikan pembagian kerja secara parsial. Simpul-simpul terus bercabang lebih jauh sampai diperoleh solusi lengkap. Jika LB tidak digunakan, maka segala kemungkinan penyelesaian harus dienumerasikan satu per satu. Oleh karena itu, LB dikalkulasikan pada setiap simpul. Jika nilai LB yang dikalkulasikan lebih besar dari nilai solusi lengkap terbaik, eliminasi simpul tersebut. Prosedur ini terus diulang sampai pencarian pada pohon berakhir dan solusi optimal ditemukan.

3.1 Strategi pencabangan (*branching*)

Pencarian dimulai dengan jadwal kosong Q_0, dimana tidak ada pekerjaan yang harus dilakukan. Untuk masalah dengan n buah pekerjaan ($j_1, j_2, ..., j_n$) dan m buah mesin ($k_1, k_2, ..., k_m$), diperoleh suatu urutan [$k_1, k_1, k_2, k_1, ... k_h$], yang berarti pekerjaan (j_1, j_2, j_4) dilakukan oleh mesin k_1, pekerjaan j_3 dilakukan oleh mesin k_2, dan j_n dilakukan oleh mesin h. Pada tingkat (aras) pertama dari pohon dibuat mn buah cabang. Pada tingkat L dari pohon ini, setiap simpul mengandung L buah pekerjaan, dan dapat bercabang menjadi $m(n-L)$ buah simpul. Saat tingkat terakhir dicapai, jumlah cabang yang diperoleh adalah nol karena $n = L$. Jika prosedur ini dilakukan sepenuhnya, $n!$ buah simpul akan dihasilkan pada pohon tingkat n. Jika pada kenyataannya hal ini yang terjadi, berarti enumerasi total telah dilakukan. Oleh karena itu, prosedur pembatas (*bounding procedure*) dibutuhkan untuk mengurangi jumlah simpul yang ada di dalam pohon.

3.2 Batas bawah (*Lower Bounds*)

LB yang digunakan dalam algoritma branch and bound ini ditentukan oleh (1):

$$LB = \frac{1}{m} \sum_{j=1}^{n} \min_{k=1,...,m} \left(P_{jk} \right) \qquad (1)$$

Pembuktian :
Untuk pembuktian yang lebih sederhana, pembuktian yang dituliskan di bawah ini tidak menyinggung kelayakan mesin. Akan tetapi pembuktian ini tetap berlaku jika kelayakan mesin diperhitungkan. Asumsikan, tanpa kehilangan hal yang sudah umum, bahwa tidak ada pekerjaan yang sudah dijadwalkan, waktu penyelesaian untuk pekerjaan terakhir pada mesin k diberikan oleh (2):

$$C_k = \sum_{i=1}^{N_k} P_{j_{k,1},k} \qquad (2)$$

N_k adalah jumlah pekerjaan yang dijadwalkan pada mesin k, P_{jk} adalah waktu pemrosesan pekerjaan j pada mesin k dan $j_{k,i}$ adalah pekerjaan ke i^{th} yang dijadwalkan pada mesin k. Untuk total m buah mesin, fungsi ini ditujukan untuk meminimalisasi C_{max}, seperti yang didefinisikan pada :

$$C_{\max} = \max\{C_1,...,C_m\} \quad (3)$$

Jika vektor $\mathbf{c}\in R^m$ didefinisikan dengan komponen C_k, maka fungsi ini dapat dituliskan sebagai :

$$C_{\max} = \|c\|_\infty \quad (4)$$

Berdasarkan *Golub and Van Loan*, rumus berikut berlaku untuk vektor $\mathbf{c}\in R^m$

$$\frac{\|C\|_1}{m} = \frac{1}{m}\sum_{k=1}^{m} C_k \le \|c\|_\infty \quad (5)$$

Pertidaksamaan (5) juga berlaku untuk minimisasi $\mathbf{c}^*$ dari C_{max} yang berarti penjadwalan optimal. Dengan menyederhanakan (2) dan (5), kita peroleh rumus berikut :

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{i=1}^{N_k} P_{(j_{k,i})^*,k} \le \|c^*\|\infty \quad (6)$$

$(j_{m,n})^*$ adalah pekerjaan ke n^{th} yang dijadwalkan pada mesin *m* pada jadwal yang optimal. Jika k_j^* berarti mesin dimana pekerjaan *j* akan diproses pada penjadwalan yang optimal, maka untuk *n* buah pekerjaan pertidaksamaan (6) dapat dituliskan sebagai :

$$\frac{1}{m}\sum_{j=1}^{n} P_{j,k_j^*} \le \|c^*\|_\infty \quad (7)$$

Akan tetapi, jika $\min\{P_{j,.}\}$ adalah masukkan minimum dari pemrosesan matrix **P** pada baris *j*, kita peroleh

$$\min\{P_{j,.}\} \le P_{j,k_j^*} \qquad j = 1..n$$

$$\xrightarrow{Eq.7} LB = \frac{1}{m}\sum_{j=1}^{n}\min\{P_{j,.}\} \le \|c^*\|_\infty \quad (8)$$

Ekspresi LB didefinisikan pada (8) dapat ditunjukkan berkorespondensi dengan nilai C_{max} optimal dari masalah penjadwalan yang terdiri dari mesin-mesin seragam dengan **P**=[$\min\{P_1,.\}$ $\min\{P_n,.\}$] ketika *preemption* dibolehkan.

3.3 Strategi pencarian

Branch and Bound dilengkapi dengan strategi DFS (*depth-first search*). Oleh karena itu, setelah melkukan pencabangan pada tingkat tertentu, sebuah simpul dipilih dan melakukan pencabangan lagi untuk semua kemungkinan simpul, simpul lain dipilih kemudain melakukan pencabangan lagih, begitu seterusnya sampai tingkat terakhir dicapai dan solusi optimal diperoleh. Solusi lengkap ini disebut sebagai "*yang terbaik sampai saat ini*", dan algoritma ini akan melakukan *backtrack* dan mengeliminasi simpul dengan LB yang lebih kecil dari solusi "*yang terbaik sampai saat ini*". Sebaliknya, algoritma ini akan bercabang ke simpul yang lain untuk melakukan proses maju. Pada setiap tingkat, simpul dengan nilai LB terkecil dipilih untuk melakukan pencabangan, kerena simpul ini mempunyai kemungkinan yang lebih besar untuk mencapai nilai fungsi objektif yang lebih kecil daripada simpul dengan nilai LB yang besar.

4. Studi komputasional

Terdapat tiga buah faktor yang harus dipertimbangkan, yaitu :jumlah pekerjaan *n*, jumlah mesin *m*, dan faktor kelayakan ρ. Waktu pemrosesan digambarkan dengan distribusi seragam [10,100] dan direpresentasikan dengan matriks **P** berukuran $m \times n$. untuk menghitung kelayakan mesin, matriks biner **P*e*** berukuran $m \times n$ dengan masukkan biner dari elemen ke a_{11} sampai a_{mn} dibentuk. Jika $a_{kj} = 1$, maka pekerjaan *j* adalah layak untuk diproses pada mesin k.; sebaliknya, menjadi tidak layak jika $a_{kj} = 0$. Faktor ρ adalah persentase dari pekerjaan yang layak pada satu atau lebih buah mesin dan sama dengan jumlah angka 1 dibagi dengan *mn*. Faktor-faktor dan tingkatnya dituliskan sebagai berikut :

n: 10, 20, 30, 40
m: 2, 4, 6, 8
ρ: 0.2, 0.5, 0.8

4.1 Hasil

Ukuran performansi yang dipilih untuk mengevaluasi performansi dari Branch and Bound adalah jumlah simpul yang ditelaah. Ukuran ini dipilih daripada berdasarkan *CPU time* karena *CPU* time sangat bergantung dengan konfigurasi perangkat keras yang digunakan.

Hasil dari komputasional dituliskan pada Tabel 1. Beberapa nilai ρ tidak dapat diaplikasikan pada nilai *m* tertentu. Secara lebih spesifik, untuk memperoleh hasil yang valid, masalah non-trivial maka faktor ρ harus lebih besar dari *1/m*. Gambar 1-a sampai Gambar 1-d adalah kesimpulan dari hasil komputasional.

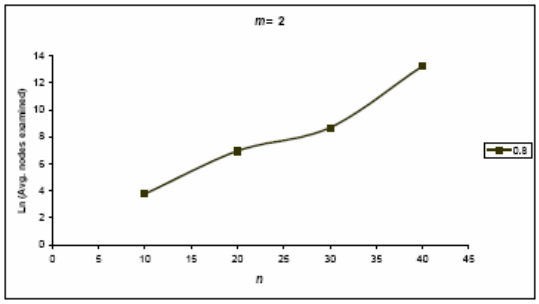
No.	*m*	*n*	Ln (rata-rata)		
			ρ = 0.2	ρ = 0.5	ρ = 0.8
1	2	10	-	-	3.756
2	2	20	-	-	6.939
3	2	30	-	-	8.682
4	2	40	-	-	13.22
5	4	10	-	5.167	5.999
6	4	20	-	9.811	12.39
7	4	30	-	13.34	17.30
8	4	40	-	16.50	17.18
9	6	10	3.034	6.463	7.175

10	6	20	4.320	11.34	12.60
11	6	30	5.835	17.23	18.54
12	6	40	6.528	19.61	21.11
13	8	10	4.158	6.923	7.319
14	8	20	6.700	13.96	14.21
15	8	30	10.59	18.67	19.41
16	8	40	13.75	21.73	23.77

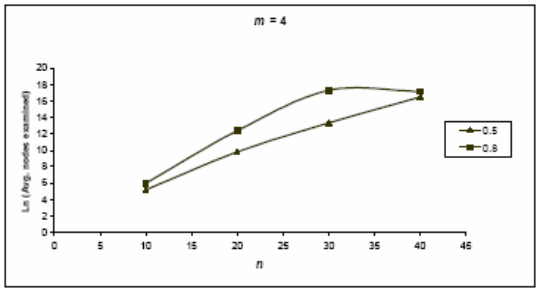
Tabel 1. Hasil komputasional

Logaritma natural (Ln) untuk jumlah simpul rata-rata dikalkulasikan untuk kemudahan dalam pengamatan. Gambar 1-a sampai 1-d menunjukkan relasi antara jumlah pekerjaan dan jumlah simpul rata-rata untuk nilai ρ yang berbeda. Lebih jauh lagi, untuk mempelajari sifat dari algoritma seiring dengan meningkatnya jumlah mesin, jumlah simpul rata-rata untuk semua ukuran pekerjaan dikalkulasikan dengan nilai ρ yang berbeda seperti ditunjukkan pada Gambar 2.

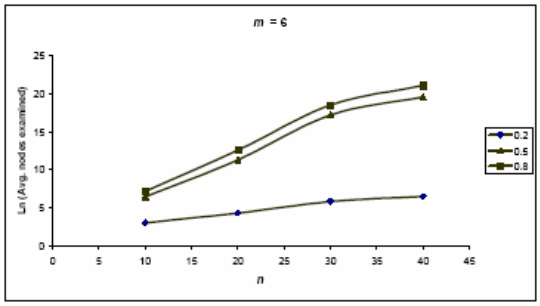
Dengan mengobservasi Gambar 1 dan 2, sangat jelas bahwa seiring dengan meningkatnya jumlah pekerjaan dan/atau jumlah mesin, maka jumlah simpul rata-rata yang dibentuk juga meningkat. Sebagai tambahan, semakin kecil nilai ρ, semakin kecil pula jumlah simpul rata-rata. Semakin besar nilai persentase pekerjaan yang layak, semakin banyak pula kombinasi yang terbentuk, semakin banyak pula simpul yang harus diperiksa.



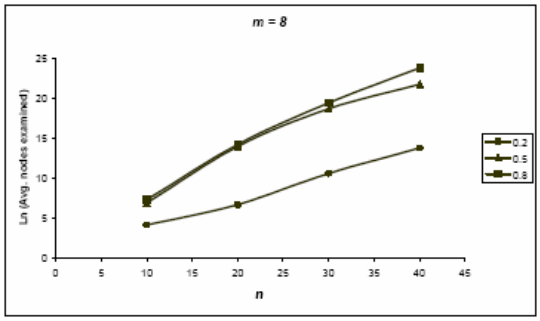
Gambar 1a



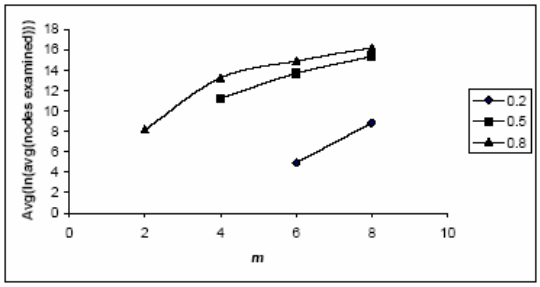
Gambar 1b



Gambar 1c



Gambar 1d



Gambar 2

5. Kesimpulan

Masalah mesin paralel tidak berelasi dengan batasan kelayakan dibahas dalam makalah ini. Oleh karena itu, algoritma Branch and Bound dikembangkan untuk meminimisasi waktu pada mesin paralel tidak berelasi dengan barasan kelayakan mesin. Performansi dari algoritma ini diekspresikan dengan jumlah simpul rata-rata yang diperiksa. Hal ini diuji dengan berbagai konfigurasi permasalahan yang berbeda (*m*, *n*, dan ρ). Performansi Branch and Bound meningkat secara signifikan ketika faktor kelayakan berada dibawah 0.5. Penelitian ke depan dapat difokuskan untuk memperoleh kualitas batas bawah (LB) yang lebih baik untuk meningkatkan performansi algoritma Branch and Bound ini secara keseluruhan.

6. Referensi

Cheng, T. and Sin, C. 1990. *"A State-of-the-Art Review of Parallel-Machine Scheduling Research***"**. European Journal of Operation Research, Vol. 47: 271-292.

Davis, E., and Jaffe, J. 1980. "*Algorithms for Scheduling Tasks on Unrelated Processors*". Journal of the Association for Computing Machinery, Vol. 28: 721-736.

De, P., and Morton, T. 1980. "*Scheduling to Minimize Makespan on Unrelated Processors*". Decision Sciences, Vol. 11: 586-602.

W. H. Freeman and Company. Golub, G.H. and Van Loan, C.F. 1996. "*Matrix Computations*", 3rd Ed., The Johns Hopkins University Press.

Martello, S.; Soumis, F.; and Toth, P. 1997. "*Exact and Approximation Algorithm for Makespan Minimization on Unrelated Parallel Machines*". Discrete Applied Mathematics, Vol. 75: 169-188.

Salhi, S. 1976. "*Algorithms for Scheduling with Independent Tasks*". Journal of the Association for Computing Machinery, Vol. 23: 116-127.