Performa: Media Ilmiah Teknik Industri
ISSN 1412-8624 (cetak) | ISSN 2620-6412 (online)
Vol. 20, No. 1, 2021, Hal. 15-19 doi.org/10.20961/performa.20.1.44302

Pengembangan Algoritma Programa Pembatas Untuk Penjadwalan Job-Shop

Bobby Kurniawan *
Departemen Teknik Industri, Universitas Sultan Ageng Tirtayasa
Email: b.kurniawan@untirta.ac.id

Abstrak

Penelitian ini membahas penjadwalan *job*-shop yang dilakukan di sebuah perusahaan manufaktur. Penjadwalan *job*-shop dilakukan untuk meminimasi waktu penyelesaian terbesar sekumpulan *job* (makespan). Metode programa pembatas digunakan untuk menyelesaikan masalah penjadwalan. Hasil yang didapatkan lebih baik dibandingkan dengan metode yang digunakan oleh perusahaan. Metode pemrograman pembatas juga diuji pada beberapa masalah yang dibangkitkan secara acak.

Kata kunci: *job-shop*, *makespan*, penjadwalan, programa pembatas

Abstract

This research addressed the job-shop scheduling problem in a manufacturing company. The objective is to minimize the makespan. Constraint programming approach is developed to solve the problem. The result obtained from constraint programming is better than that of currently applied by the company. The constraint programming approach is also tested on several randomly generated instances.

***Keywords**: constraint programming, job shop, makespan, scheduling*

1. Pendahuluan

Penjadwalan *job-shop* merupakan salah satu masalah optimisasi kombinatorial yang sulit untuk dipecahkan (Graham, Lawler, Lenstra, & Kan, 1979). Penjadwalan *job-shop* mendapat perhatian dari peneliti disebabkan oleh banyaknya pabrik yang menggunakan konfigurasi ini. Konfigurasi *job-shop* adalah tata letak atau susunan sebuah lantai produksi yang memiliki lebih dari satu mesin yang memiliki fungsi berbeda. Pada penjadwalan *job-shop*, sebuah *job* terdiri dari lebih dari satu operasi, di mana satu operasi diproses pada tiap mesin yang berbeda. Urutan operasi dari setiap *job* adalah tetap. Hal ini dinamakan sebagai *precedence constraint* atau *technological constraint*. Urutan operasi yang di miliki setiap *job* bisa berbeda. Sebagai contoh, *job* 1 memiliki 3 buah operasi dengan urutan pengerjaan pada mesin 1, mesin 2 dan mesin 3. Sedangkan *job* 2 memiliki 3 operasi dengan urutan mesin 2, mesin 1, dan mesin 3. Keputusan dalam penjadwalan *job-shop* adalah mencari urutan *job* (*sequence*) pada tiap mesin untuk memenuhi satu atau beberapa kriteria tertentu.

Metode-metode yang digunakan untuk memecahkan penjadwalan *job-shop* dapat dikategorikan menjadi pendekatan eksak dan heuristik/meta-heuristik. Contoh metode eksak adalah metode pemrograman bilangan bulat yang diselesaikan dengan algoritma *branch and bound* (Artigues & Feillet, 2008). Sedangkan metode heuristik/meta-heuristik antara lain *tabu search* (Bozejko, Gnatowski, Pempera, & Wodecki, 2017), algoritma genetik (Kaweegitbundit & Eguchi, 2016), algoritma memetik (Mencía, Sierra, Mencía, & Varela, 2015), dan *particle swarm optimization* (Huang, Tian, Wang, & Ji, 2016).

Penelitian ini dilakukan pada sebuah perusahaan manufaktur yang memiliki karakteristik sebuah *job-shop*. Masalah yang dihadapi oleh perusahaan ini adalah tidak adanya suatu sistem penjadwalan yang cukup baik untuk mempersingkat waktu pengerjaan seluruh *job* yang ada. Penjadwalan yang dilakukan oleh perusahaan saat ini adalah menggunakan aturan *first in first out* (FIFO). Oleh karena itu, masalah yang dihadapi merupakan suatu penjadwalan *job-shop* untuk meminimasi *makespan*. Metode programa pembatas (*constraint programming*) digunakan untuk menyelesaikan masalah penjadwalan *job-shop*.

Metode pemrograman pembatas adalah suatu metode untuk memecahkan masalah optimisasi kombinatorial, di mana beberapa teknik dalam bidang kecerdasan buatan (*artificial intelligence*), ilmu komputer, dan riset operasi digabungkan. Metode ini telah digunakan pada beberapa masalah optimisasi kombinatorial, antara lain masalah penjadwalan (Sellmann, Zervoudakis, Stamatopoulos, & Fahle, 2002), perencanaan produksi (Peng, Lu, & Chen, 2014),

* Penulis korespondensi

penjadwalan proyek (Zou & Zhang, 2020), keseimbangan lini perakitan (Mehmet, Mehmet, & Mustafa, 2019), dan penugasan (Gabteni & Grönkvist, 2008).

Penjadwalan *job-shop* merupakan suatu masalah optimisasi kombinatorial yang sulit. *Domain* atau ruang solusi (*solution space*) akan bertambah secara eksponensial dengan bertambah besarnya jumlah *job* dan mesin. Selain itu, pembatas-pembatas pada penjadwalan *job-shop* juga bersifat tidak cembung (*non-convex*) sehingga banyak solusi yang bersifat *local optimal*. Fungsi tujuan juga bersifat tidak linier. Programa pembatas memiliki keuntungan dibandingkan dengan metode lain karena kemampuannya dalam eksplorasi pencarian (*search*) ruang solusi, kemampuan untuk mencari solusi apapun bentuk fungsi pembatas dan fungsi tujuan. Faktor-faktor ini yang membuat programa pembatas lebih efektif memecahkan masalah penjadwalan *job-shop* yang memiliki jumlah *job* dan mesin yang besar dibandingkan dengan pendekatan eksak seperti programa matematika dan programa bilangan bulat. Efektivitas dari programa pembatas dibandingkan dengan pendekatan metaheuristik juga telah dibuktikan dengan kemampuan programa pembatas menemukan solusi optimal dari beberapa masalah penjadwalan *benchmark* atau patokan (Laborie, Rogerie, Shaw, & Vilím, 2018).

2. Metode Penelitian

Pada bagian akan dibahas mengenai formulasi penjadwalan job-shop, metode pemrograman pembatas, serta tata cara percobaan eksperimen.

2.1 Formulasi Masalah

Penjadwalan *job-shop* untuk meminimasi *makespan* diformulasikan sebagai berikut. Sebuah perusahaan manufaktur memiliki *order* atau pesanan sebanyak n buah *job*, $j = 1,2, \dots, n$. Setiap *job* harus diproses pada m mesin, $i = 1,2, \dots, m$. Setiap mesin hanya dapat memproses satu buah *job* pada satu waktu. Setiap *job* hanya dapat diproses oleh sebuah mesin pada satu waktu. Setiap *job* terdiri dari m buah operasi yang dikerjakan secara berurutan dengan urutan yang spesifik. Operasi terakhir dan operasi ke-l dari sebuah *job* j dinyatakan sebagai o_m^j dan o_l^j.

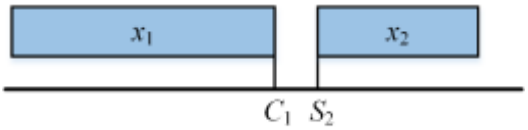
Waktu proses untuk mengerjakan *job* j pada mesin i dinotasikan sebagai p_{ij}. Mesin dan *job* siap untuk beroperasi pada waktu 0. Apabila sebuah *job* diproses pada sebuah mesin, *job* harus dikerjakan sampai selesai tanpa gangguan (*non pre-emption*).

Keputusan dari masalah penjadwalan *job-shop* adalah menentukan waktu penyelesaian dari setiap *job*, C_j, sehingga makespan, C_{max}, adalah minimum. *Makespan* dinyatakan sebagai $C_{max} = \max\{C_j\}$.

2.2 Pengembangan Algoritma Programa Pembatas

Programa pembatas adalah sebuah metode yang mengkombinasikan paradigma penelitian operasional, ilmu komputer, dan kecerdasan buatan. Programa pembatas bekerja mencari solusi layak dari pada mencari solusi optimal. Selain itu, metode ini berfokus pada pembatas dan variabel dari pada fungsi tujuan. Pemrograman pembatas dilakukan dengan mengimplementasikan strategi pencarian (*search strategy*) dan perambatan pembatas (*constraint propagation*) untuk mendapatkan solusi yang memenuhi seluruh pembatas.

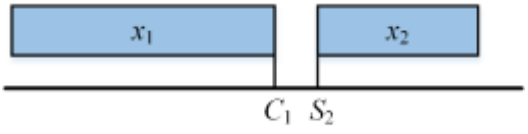
Penelitian ini menggunakan IBM ILOG CP *Optimizer* sebagai metode pemrograman pembatas. Untuk memodelkan masalah agar dapat diselesaikan oleh CP *Optimizer* digunakan OPL *modeling language*. Suatu masalah dalam pemrograman pembatas direpresentasikan oleh variabel, domain, dan pembatas. Variabel dalam pemrograman pembatas merupakan sebuah variabel *interval*. Sebuah variabel interval memiliki waktu mulai, waktu selesai, dan durasi. Dengan demikian, sebuah *job* dapat dinyatakan dinyatakan sebagai sebuah variabel *interval* yang memiliki waktu mulai, waktu selesai, dan durasi. Waktu mulai dan waktu selesai dari sebuah variabel *interval* merupakan keputusan dalam penjadwalan. Gambar 1 mengilustrasikan sebuah variabel *interval*, yaitu x_1.



Gambar 1. Pembatas urutan operasi sebuah *job* (Laborie, IBM ILOG CP Optimizer for Detailed Scheduling Illustrated on Three Problems, 2009)

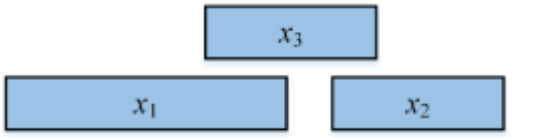
Penjelasan Gambar 1 adalah sebagai berikut. Domain dari x_1 adalah $0 \leq x_1 \leq a$. Durasi dari variabel *interval* x_1 adalah p. Waktu awal dan waktu akhir dinotasikan sebagai S_1 dan C_1.

Dalam penjadwalan *job-shop*, terdapat pembatas yang menyatakan hubungan urutan pengerjaan operasi dari sebuah *job*. Pembatas ini menyatakan dapat disebut sebagai pembatas urutan operasi (*precedence constraint*). Gambar 2 mengilustrasikan pembatas urutan operasi dari sebuah *job* yang memiliki urutan operasi x_1-x_2. Pembatas ini dinyatakan dengan *end_before_start*(x_1, x_{22}).



Gambar 2. Pembatas urutan operasi sebuah *job* (Laborie, IBM ILOG CP Optimizer for Detailed Scheduling Illustrated on Three Problems, 2009)

Selain pembatas urutan operasi, penjadwalan *job-shop* juga memiliki pembatas *no overlap*. Pembatas ini membuat sebuah mesin tidak dapat memproses dua buah operasi dalam selang waktu yang bersamaan. Gambar 3 mengilustrasikan pembatas *no overlap* di mana operasi x_1 dan x_3 saling *overlap*. Sedangkan operasi x_1 dan x_2 tidak saling *overlap*. Programa pembatas menjamin seluruh operasi tidak *overlap*.



Gambar 3. Pembatas *no overlap* (Laborie, IBM ILOG CP Optimizer for Detailed Scheduling Illustrated on Three Problems, 2009)

2.3 Percobaan Numerik

Percobaan numerik dilakukan untuk menentukan efektivitas programa pembatas dalam menyelesaikan masalah. Percobaan numerik dilakukan dengan dua cara. Cara pertama adalah dengan menyelesaikan studi kasus penjadwalan di sebuah perusahaan manufaktur. Hasil penjadwalan dari programa pembatas kemudian dibandingkan dengan hasil penjadwalan yang dilakukan oleh perusahaan manufaktur. Sedangkan cara kedua dengan menyelesaikan masalah dari data buatan yang menjadi patokan (*benchmark*).

Cara pertama adalah membandingkan hasil penjadwalan menggunakan metode pemrograman numerik dengan hasil penjadwalan yang dilakukan saat ini.

a. Data perusahaan manufaktur

Data untuk menyelesaikan studi kasus di sebuah perusahaan manufaktur diambil dalam pada September 2019. Data-data tersebut adalah data teknis pembuatan produk yang diambil di bagian produksi dan pengendalian kualitas. Adapun data yang diambil adalah sebagai berikut. Jumlah produk adalah 3 buah. Jumlah mesin adalah 3 buah. Untuk setiap produk, waktu proses serta alur proses pengerjaan (routing) ditampilkan pada Tabel 1 dan Tabel 2.

b. Data buatan dari patokan

Data yang digunakan berasal dari data Orlib (Beasley, 2020) dan Talliard (Taillard, 2020).

c. Prosedur eksperimen

Pemrograman pembatas dijalankan pada komputer 6 *cores* dengan memori 8 GB. Pemrograman pembatas diimplementasikan menggunakan bahasa pemrograman IBM OPL dan *software* optimisasi IBM CPLEX CP *Optimizer* versi 12.9. Seluruh masalah penjadwalan dijalankan paling lama selama 1 jam (3.600 detik). Apabila CPLEX belum menghasilkan solusi setelah 1 jam, maka proses dihentikan.

Tabel 1. Waktu proses produk (jam)

Produk	Op1	Op2	Op3
1	3	3	3
2	2	2	3
3	2	2	2

Tabel 2. Waktu proses produk (jam)

Produk	Op1	Op2	Op3
1	1	3	2
2	1	2	3
3	3	1	2

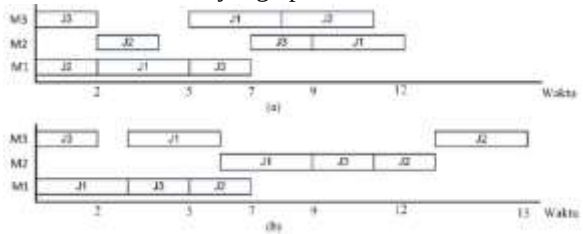
3. Hasil dan Pembahasan

Percobaan numerik dibagi menjadi dua bagian. Bagian pertama membandingkan hasil dari pemrograman pembatas dengan hasil penjadwalan yang dilakukan oleh perusahaan manufaktur. Bagian kedua mengukur efektivitas pemrograman pembatas dalam menyelesaikan masalah penjadwalan *job-shop* dengan jumlah *job* dan mesin yang besar.

a. Perbandingan pemrograman pembatas dengan penjadwalan oleh perusahaan

Gambar 4 menunjukan perbandingan hasil penjadwalan pemrograman pembatas dengan metode *first in first out* (FIFO) yang dilakukan oleh perusahaan manufaktur. Hasil penjadwalan menghasilkan *makespan* sebesar 12 jam. Sedangkan apabila menggunakan metode FIFO, *makespan* yang didapat adalah sebesar 15 jam.

Hasil ini menunjukan bahwa walaupun penjadwalan metode FIFO sangat mudah dan menghasilkan hasil yang baik, akan tetapi hasil yang didapat belum menjamin keoptimalan suatu solusi. Berbeda dengan pemrograman pembatas yang melakukan *branching* dan penjalaran (*propagation*) untuk mencari solusi yang optimal.



Gambar 4. Jadwal hasil dari: (a) pemrograman pembatas, (b) metode FIFO

b. Efektivitas pemrograman pembatas

Untuk menunjukan efektivitas pemrograman pembatas dalam menyelesaikan masalah penjadwalan *job-shop* yang memiliki jumlah *job* dan mesin yang besar, pemrograman pembatas dijalankan pada masalah-masalah *job-shop* yang telah dijadikan patokan (*benchmark*). CPLEX dijalankan selama maksimal 3.600 detik untuk setiap masalah. Hasil yang didapat dari CPLEX berupa batas bawah (*lower bound*) dan solusi terbaik yang didapat (*best integer solution*).

Dari hasil tersebut, dapat dihitung celah (*gap*) antara solusi terbaik dengan batas bawah. Suatu solusi disebut optimal apabila gap yang dihasilkan adalah nol. Dengan demikian, semakin kecil gap semakin baik solusi. *Gap* dihitung berdasarkan persamaan berikut.

$$Gap = \frac{BS - LB}{LB} \quad (1)$$

di mana BS dan LB adalah solusi terbaik dan batas bawah.

Tabel 3 menampilkan hasil dari pemrograman pembatas dalam menyelesaikan masalah penjadwalan *job-shop* skala besar. Sebanyak 10 masalah penjadwalan *job-shop* berasal dari Orlib dan Taillard. Metode pemrograman pembatas menemukan solusi optimal pada 8 buah masalah (*ft*06, *la*02, *la*07, *la*14, *abz*5, *abz*6, *la*22, dan *la*28). Gap yang dihasilkan oleh metode pemrograman pembatas pada 8 buah masalah tersebut adalah 0. Selain itu, metode pemrograman pembatas mampu menemukan solusi optimal kurang dari 11 detik. Hal ini menunjukan bahwa metode pemrograman pembatas cukup efisien dalam menyelesaikan masalah penjadwalan *job-shop* skala kecil dan menengah, dari ukuran *job* 6 sampai 20 buah, dan ukuran mesin 6 sampai 10 buah.

Tabel 3. Hasil percobaan numerik

No	Masalah	Ukuran (n×m)	Gap (%)	Waktu (detik)
1	*ft*06[a]	6×6	0	0,57
2	*la*02[a]	10×5	0	0,94
3	*la*07[a]	15×5	0	0,22
4	*la*14[a]	20×5	0	0,12
5	*abz*5[a]	10×10	0	5,14
6	*abz*6[a]	10×10	0	1,81
7	*la*22[a]	15×10	0	10,55
8	*la*28[a]	20×10	0	5,97
9	*ta*41[b]	30×20	8,87	3.600
10	*ta*62[b]	50×20	0,45	3.600

[a]: Data berasal dari Orlib (Beasley, 2020)
[b]: Data berasal dari Taillard (Taillard, 2020)

Untuk masalah penjadwalan *job-shop* ukuran besar (lebih dari 30 *job* dan 20 mesin), pemrograman pembatas tidak dapat menemukan solusi optimal setelah 3600 detik. Gap yang dihasilkan adalah 8,87% (untuk *ta*41) dan 0,45% (untuk *ta*62). Dapat disimpulkan bahwa walaupun pemrograman pembatas tidak dapat menemukan solusi optimal, akan tetapi gap yang dihasilkan sangat kecil.

Dari hasil percobaan numerik ini, dapat disimpulkan bahwa metode pemrograman pembatas merupakan metode yang cukup potensial untuk dapat menyeledaikan masalah optimisasi kombinatorial. Metode ini dapat digunakan pada perusahaan manufaktur bertipe *job-shop* yang memiliki *order* dalam jumlah menengah.

4. Kesimpulan

Penjadwalan *job-shop* untuk meminimasi makespan dibahas pada penelitian ini. Metode pemrograman pembatas digunakan untuk memecahkan masalah *job-shop* pada suatu perusahaan manufaktur. Berdasarkan hasil percobaan numerik, metode pemrograman pembatas lebih baik dibandingkan dengan metode FIFO yang digunakan oleh perusahaan. Efektivitas pemrograman pembatas dalam memecahkan penjadwalan *job-shop* dievaluasi melalui percobaan numerik pada masalah-masalah yang menjadi patokan.

Hasil menunjukan bahwa walaupun metode pemrograman pembatas cukup baik dalam memecahkan masalah penjadwalan *job-shop* dalam skala besar, akan tetapi solusi optimal belum dapat dicapai. Oleh karena itu, penelitian lanjutan membahas mengenai metode lain, seperti *tabu search*, algoritma genetik, algoritma memetik, dan *particle swarm optimization*. Selain itu, diperlukan perbandingan dari metode pemrograman pembatas dengan metode-metode lain.

Dari sisi model penjadwalan, masalah dapat dikembangkan dengan menyertakan unsur konsumsi energi dalam fungsi tujuan. Dengan demikian, masalah penjadwalan *job-shop* akan menjadi masalah multi-obyektif yang memerlukan metode baru dalam penentuan solusinya.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada para penelaah yang telah memberikan saran yang membangun untuk artikel ini.

Daftar Pustaka

Artigues, C., & Feillet, D. (2008). A branch and bound method for the job-shop problem with sequence-dependent setup times. *Annals of Operations Research, 159*, 135–-159. Retrieved from https://doi.org/10.1007/s10479-007-0283-0

Beasley, J. E. (2020). OR-Library. *OR-Library*.

Bozejko, W., Gnatowski, A., Pempera, J., & Wodecki, M. (2017, 11). Parallel tabu search for the cyclic job shop scheduling problem. *Computers & Industrial Engineering, 113*, 512–524. doi:10.1016/j.cie.2017.09.042

Gabteni, S., & Grönkvist, M. (2008, 6). Combining column generation and constraint programming to solve the tail assignment problem. *Annals of Operations Research, 171*, 61. doi:10.1007/s10479-008-0379-1

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. (1979, 1). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In P. L. Hammer, E. L. Johnson, & B. H. Korte (Eds.). Elsevier. doi:10.1016/S0167-5060(08)70356-X

Huang, S., Tian, N., Wang, Y., & Ji, Z. (2016, 11). An Improved Version of Discrete Particle Swarm Optimization for Flexible Job Shop Scheduling Problem with Fuzzy Processing Time. *An Improved Version of Discrete Particle Swarm Optimization for Flexible Job Shop Scheduling Problem with Fuzzy Processing Time, 2016*, e5958640. Hindawi. doi:https://doi.org/10.1155/2016/5958640

Kaweegitbundit, P., & Eguchi, T. (2016). Flexible job shop scheduling using genetic algorithm and

heuristic rules. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, *10*, JAMDSM0010-JAMDSM0010. doi:10.1299/jamdsm.2016jamdsm0010

Laborie, P. (2009). IBM ILOG CP Optimizer for Detailed Scheduling Illustrated on Three Problems. In W.-J. van Hoeve, & J. N. Hooker (Ed.), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (pp. 148–162). Berlin: Springer. doi:10.1007/978-3-642-01929-6_12

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018, 4). IBM ILOG CP optimizer for scheduling. *Constraints*, *23*, 210–250. doi:10.1007/s10601-018-9281-x

Mehmet, P., Mehmet, A. H., & Mustafa, Y. (2019, 1). A constraint programming approach to type-2 assembly line balancing problem with assignment restrictions. *Assembly Automation*, *39*, 813–826. Retrieved from https://doi.org/10.1108/AA-12-2018-0262

Mencía, R., Sierra, M. R., Mencía, C., & Varela, R. (2015, 9). Memetic algorithms for the job shop scheduling problem with operators. *Applied Soft Computing*, *34*, 94–105. doi:10.1016/j.asoc.2015.05.004

Peng, Y., Lu, D., & Chen, Y. (2014, 1). A Constraint Programming Method for Advanced Planning and Scheduling System with Multilevel Structured Products. *A Constraint Programming Method for Advanced Planning and Scheduling System with Multilevel Structured Products*, *2014*, e917685. Hindawi. doi:https://doi.org/10.1155/2014/917685

Sellmann, M., Zervoudakis, K., Stamatopoulos, P., & Fahle, T. (2002, 9). Crew Assignment via Constraint Programming: Integrating Column Generation and Heuristic Tree Search. *Annals of Operations Research*, *115*, 207–225. doi:10.1023/A:1021105422248

Taillard, E. (2020). Scheduling instances. *Scheduling instances*.

Zou, X., & Zhang, L. (2020, 1). A constraint programming approach for scheduling repetitive projects with atypical activities considering soft logic. *Automation in Construction*, *109*, 102990. doi:10.1016/j.autcon.2019.102990