PAPER – OPEN ACCESS

Perbandingan Tabu Search Dan Algoritma Genetika Dalam Menyelesaikan Masalah Penjadwalan Job Shop

Author : Rosalyn Dwi Octaviana, dkk.
DOI : 10.32734/ee.v5i2.1537
Electronic ISSN : 2654-704X
Print ISSN : 2654-7031

Volume 5 Issue 2 – 2022 TALENTA Conference Series: Energy & Engineering (EE)

EE Conference Series 05 (2022)
TALENTA Conference Series
Available online at https://talentaconfseries.usu.ac.id

PERBANDINGAN *TABU SEARCH* DAN ALGORITMA GENETIKA DALAM MENYELESAIKAN MASALAH PENJADWALAN *JOB SHOP*

Rosalyn Dwi Octaviana[a], Aditya Tirta Pratama[b*], Gembong Baskoro[c]

[a]*Rosalyn Dwi Octaviana, Master of Mechanical Engineering, Swiss German University, Tangerang 15143, Indonesia*
[b]*Aditya Tirta Pratama, Department of Industrial Engineering, Swiss German University, Tangerang 15143, Indonesia*
[c]*Gembong Baskoro, Master of Mechanical Engineering, Swiss German University, Tangerang 15143, Indonesia*

rosalyndwioctaviana@yahoo.com, aditya.pratama@sgu.ac.id*, gembong.baskoro@sgu.ac.id

Abstrak

Job shop adalah istilah yang digunakan untuk mendeskripsikan keadaan ketika produk-produk yang diproduksi oleh sebuah perusahaan dapat memiliki rute pemrosesan yang berbeda-beda. Membuat jadwal produksi di sebuah perusahaan manufaktur yang mengadopsi *job shop* dapat menjadi sebuah tantangan akibat tingginya fleksibilitas dalam proses produksi, yang menyebabkan meningkatnya kompleksitas saat membuat jadwal produksi. Pendekatan metaheuristik biasanya digunakan untuk menyelesaikan masalah penjadwalan *job shop* (JSSP). Dalam penelitian ini, dua metode metaheuristik dibandingkan, yaitu *tabu search* dan algoritma genetika, untuk menentukan metode mana yang lebih unggul. Kedua metode akan dijalankan menggunakan parameter yang paling optimal untuk masing-masing nya, sehingga dapat memberikan perbandingan yang adil. Data sampel akan digunakan untuk simulasi. Metode yang dapat menghasilkan *makespan* terendah akan dianggap lebih unggul. Berdasarkan hasil simulasi, diketahui bahwa *tabu search* lebih unggul untuk menyelesaikan JSSP, baik untuk masalah JSSP sederhana maupun kompleks, karena dapat menghasilkan *makespan* yang paling rendah untuk kedua kasus tersebut.

Kata kunci: Algoritma Genetika; *Job Shop*; *Makespan*; Metaheuristik; *Tabu Search*

1. Pendahuluan

Membuat jadwal dalam alur *job shop* biasa dikenal dengan *Job Shop Scheduling Problem* (JSSP). JSSP diklasifikasikan sebagai masalah optimasi kombinasional atau masalah *Non-deterministic Polynomial-time* (NP-*hard*) [1]. Masalah NP-hard harus diselesaikan dalam waktu polinomial karena mesin komputer tidak mampu memberikan jawaban yang pasti untuk masalah itu.

JSSP secara umum diformulasikan sebagai *n* pekerjaan ($P_1,\ldots,P_n$) yang harus diproses pada *m* mesin ($M_1,\ldots,M_n$). Pekerjaan P_i ($i = 1,\ldots,n$) terdiri atas urutan n_i dari operasi O_u [2]. Sebuah mesin hanya dapat menangani satu pekerjaan pada suatu waktu dan tanpa adanya interupsi ditengah-tengah pekerjaan. Tujuan JSSP adalah untuk menyusun jadwal pekerjaan yang meminimalkan kriteria tertentu seperti *makespan*, keterlambatan maksimum, keterlambatan tertimbang total, jumlah tertimbang pekerjaan terlambat, dan total waktu penyelesaian tertimbang dengan mempertimbangkan beberapa limitasi seperti prioritas, kapasitas, dan tanggal jatuh tempo atau kendala tambahan seperti mesin multiguna, alternatif pemrosesan, pekerjaan yang memburuk, aktivitas mesin, waktu penyiapan yang bergantung pada urutan, ketersediaan, operasi yang tumpang tindih, dan waktu pemrosesan yang dapat dikontrol [3].

Karena metode eksak tidak dapat digunakan untuk menyelesaikan JSSP, maka metode perkiraan akan digunakan sebagai gantinya [4]. Metode perkiraan mampu menghasilkan solusi berkualitas tinggi dalam waktu yang wajar meskipun tidak menjamin optimalitas solusi yang diperoleh. Pendekatan yang paling populer untuk menyelesaikan JSSP dengan metode perkiraan adalah dengan menggunakan metaheuristik.

Metaheuristik dapat diklasifikasi berdasarkan beberapa kriteria. Salah satunya adalah pencarian berbasis lokal dan pencarian berbasis populasi. Metaheuristik dengan pencarian berbasis lokal dapat menghasilkan satu solusi pada setiap iterasi hingga mencapai solusi optimal atau mendekati optimal [5]. Pencarian dimulai dari satu titik di dalam ruang solusi dan bergerak melewati tetangganya untuk menemukan solusi yang lebih baik sampai pencarian dihentikan [6]. Beberapa contoh metaheuristik dengan pencarian berbasis lokal adalah *tabu search*, *simulated annealing*, iterasi lokal, dan *GRASP*.

Di sisi lain, metaheuristik dengan pencarian berbasis populasi dapat menghasilkan banyak solusi pada saat yang bersamaan pada setiap iterasi hingga mencapai solusi optimal atau mendekati optimal [5]. Kombinasi dari beberapa solusi dilakukan untuk menghasilkan solusi baru dengan tingkat kebugaran yang lebih baik. Beberapa contoh dari metaheuristik dengan pencarian bebasis populasi adalah algoritma genetika, optimasi koloni semut, optimasi gerombolan partikel, dan algoritma optimasi pengendara.

Karena metaheuristik dengan pencarian berbasis lokal dan populasi memiliki strategi yang sangat berbeda, maka menjadi menarik untuk membandingkan kinerja keduanya. Banyak peneliti yang sudah mengaplikasikan kedua metode tersebut untuk menyelesaikan berbagai masalah.

Selection and peer-review under responsibility of The 6th National Conference on Industrial Engineering (NCIE) 2022
p-ISSN: 2654-7031, e-ISSN: 2654-704X , DOI: 10.32734/ee.v5i2.1537

Di dalam penelitian yang dilakukan Beaty [7], mereka membandingkan kinerja *tabu search* dan algoritma genetika untuk masalah penjadwalan instruksi. Bedasarkan hasil penelitian, kedua metode mampu menemukan solusi yang optimal. Namun, *tabu search* memerlukan waktu yang lebih lama untuk menemukan solusi optimal dibandingkan algoritma genetika. Di dalam penelitian yang dilakukan oleh Chu & Fang [8], mereka membandingkan kinerja *tabu search* dan algoritma genetika untuk masalah penjadwalan kelas. Berdasarkan hasil penelitian, *tabu search* dapat menghasilkan solusi yang optimal dengan waktu komputasi yang lebih sedikit dibandingkan algoritma genetika. Di dalam penelitian yang dilakukan oleh Renman & Fristedt [9], mereka membandingkan kinerja *tabu search* dan algoritma genetika untuk masalah penjadwalan kelas. Berdasarkan dari hasil penelitian, *tabu search* memerlukan waktu komputasi yang lebih lama dibanding algoritma genetika. Hal ini disebabkan oleh banyaknya jumlah gerakan yang dievaluasi di setiap iterasi. Di dalam penelitian yang dilakukan oleh Mohan et al. [4], mereka menemukan bahwa meskipun *tabu search* dapat menjamin pencarian yang beragam untuk menemukan solusi optimal, hal itu sangat tergantung pada struktur lingkungan dan solusi awal. Selain itu, meskipun algoritma genetika menawarkan teknologi pengkodean yang matang dan operasi yang sederhana, namun membutuhkan waktu komputasi yang lebih lama dan cenderung konvergensi prematur cenderung terjadi ketika digunakan untuk menyelesaikan masalah kombinatorial yang besar.

Di dalam penelitian ini, untuk mengamati kinerja metaheuristik berbasis pencarian lokal dan metaheuristik berbasis populasi, *tabu search* (sebagai metaheuristik berbasis pencarian lokal) akan dibandingkan dengan algoritma genetika (sebagai metaheuristik berbasis pencarian populasi). Data sampel akan digunakan dalam penelitian untuk simulasi. Kedua metode akan dijalankan menggunakan parameter yang paling optimal untuk masing-masingnya, sehingga dapat memberikan perbandingan yang adil. Metode yang dapat menghasilkan *makespan* terendah akan dianggap lebih unggul. Hasil dari penelitian ini dapat menjadi acuan mengenai metodologi yang cocok untuk menyelesaikan JSSP.

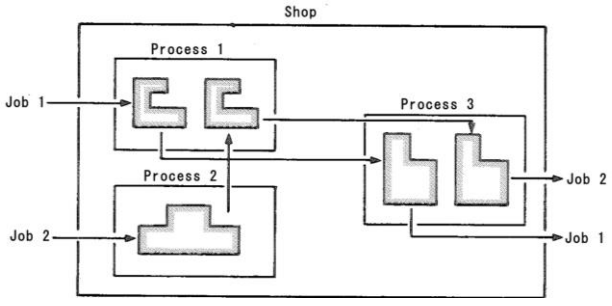
2. Tinjauan Literatur

2.1. Job Shop

Karakteristik utama dari *job shop* adalah perbedaan rute pemrosesan untuk menghasilkan produk-produk karena setiap produk mungkin memiliki persyaratan pemrosesan yang berbeda, seperti yang ditunjukkan pada Gambar 1. Menstandardisasi produk menjadi suatu pekerjaan yang sulit, karena terdapat banyak variasi produk dan masing-masing berpotensi perlu diproduksi secara berbeda dan memiliki waktu produksi yang berbeda-beda pula. Beberapa kelebihan dan kekurangan dari *job shop* seperti pada Tabel 1.

Tabel 1. Kelebihan dan Kekurangan *Job shop*

Kelebihan	Kekurangan
Mudah di *set up*	Sulit diotomatisasi
Fleksibilitas tinggi	Sulit untuk dijadwalkan
Mudah meningkatkan kapasitas	Sulit untuk diukur dan ditingkatkan



Gambar 1. *Job shop* [10]

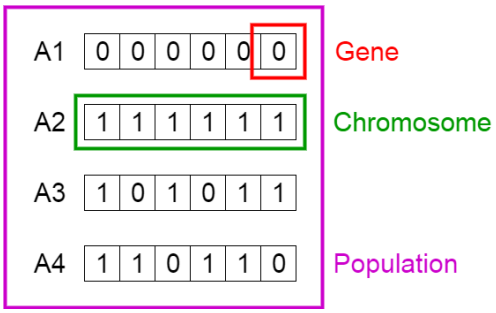
2.2. Algoritma Genetika

Algoritma Genetika (GA) adalah metode metaheuristik yang diperkenalkan oleh John Holland pada tahun 1975. Ide dasarnya didasarkan pada Prinsip Seleksi Alam Darwin, yaitu memilih yang terbaik dan membuang sisanya. Tujuan utama GA adalah mengembangkan populasi kandidat solusi (individu), di mana setiap individu memiliki kumpulan propertinya sendiri (kromosom) yang dapat dimutasi dan diubah menuju solusi yang lebih baik. GA adalah proses iteratif yang dimulai dengan memilih individu yang paling bugar dari suatu populasi. Gen dari individu yang paling bugar akan dimodifikasi untuk menghasilkan keturunan yang mewarisi karakteristik orang tuanya, lalu diwariskan kembali ke generasi berikutnya. Tingkat kebugaran setiap individu akan di

evaluasi pada setiap generasi untuk memastikan hanya individu yang paling bugar yang dapat bertahan. Jika orang tua memiliki tingkat kebugaran yang baik, keturunannya akan lebih baik daripada orang tuanya, sehingga memiliki peluang yang lebih baik untuk bertahan hidup. Proses terus berulang sampai hasil yang memuaskan dihasilkan atau jumlah generasi maksimum tercapai. Ada beberapa langkah yang harus diikuti dalam GA [10]:

2.2.1. Inisialisasi

Proses dimulai dengan inisialisasi populasi yang terdiri dari sekumpulan kandidat solusi (individu). Seorang individu dicirikan oleh satu set gen yang bergabung bersama untuk membentuk kromosom. Himpunan gen biasanya dikodekan menggunakan nilai biner (0 atau 1). Untuk JSSP, gen kromosom mewakili penjadwalan pekerjaan ke mesin dalam jadwal produksi. Urutan operasi yang muncul dalam kromosom mewakili urutan operasi produksi.



Gambar 2. Gen, Kromosom, Populasi [11]

Ukuran populasi dapat tergantung pada JSSP. Populasi awal dapat diinisialisasi baik secara acak untuk memungkinkan pencarian semua solusi yang mungkin atau terfokus pada area di mana solusi optimal paling mungkin dapat ditemukan. Gambar 2 menunjukkan representasi gen, kromosom, dan populasi.

2.2.2. Evaluasi

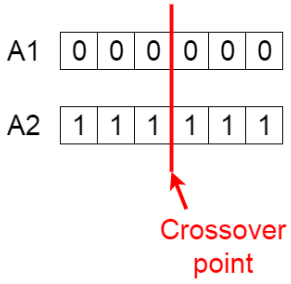
Tingkat kebugaran setiap individu yang diukur. Nilai kebugaran akan bertindak sebagai dasar dalam pemilihan individu untuk bereproduksi. Untuk JSSP, tingkat layanan, keterlambatan, dan *makespan* dapat digunakan untuk mengukur tingkat kebugaran.

2.2.3. Seleksi

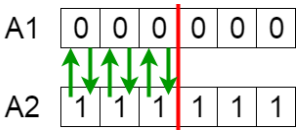
Dua individu terkuat akan dipilih sebagai orang tua untuk mewariskan gen mereka ke generasi berikutnya. Beberapa metode seleksi yang dapat digunakan adalah Turnamen, Roda Rolet, Elitisme, Peringkat dan Skala, dan Kondisi Stabil.

2.2.4. Persilangan

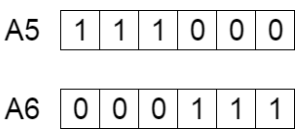
Operasi persilangan akan menghasilkan keturunan dengan cara saling menukarkan gen orang tua. Untuk melakukan persilangan, titik persilangan harus didefinisikan terlebih dahulu, seperti di Gambar 3. Gen orang tua akan dipertukarkan sampai pada titik persilangan. Contoh operasi persilangan seperti pada Gambar 4. Gambar 5 menunjukan hasil persilangan, yang juga disebut keturunan.



Gambar 3. Titik Persilangan [11]

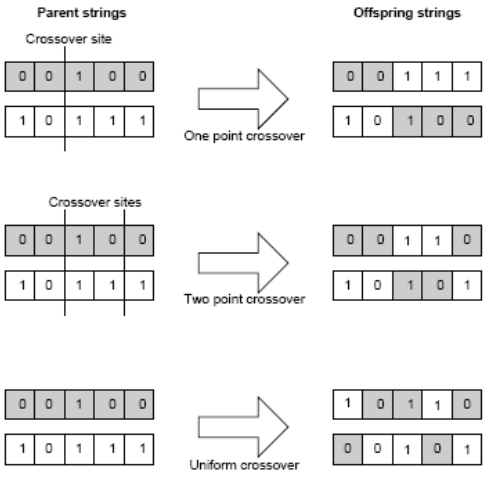


Gambar 4. Operasi Persilangan [11]



Gambar 5. Hasil Persilangan/Keturunan [11]

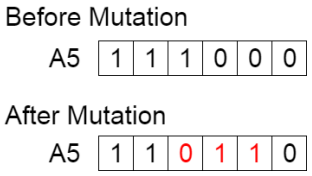
Ada beberapa metode persilangan yang dapat digunakan, seperti persilangan satu titik, persilangan dua titik, dan persilangan seragam [12]. Gambar 6 menunjukkan beberapa metode persilangan yang umum digunakan.



Gambar 6. Metode Persilangan [12]

2.2.5. Mutasi

Operasi mutasi akan memodifikasi kromosom pada individu dengan cara membalikkan gen. Mutasi bertujuan untuk menjaga keragaman dalam populasi sehingga solusi tidak terjebak pada lokal optimum. Gambar 7 menunjukkan perbandingan kromosom sebelum dan sesudah dimutasi.



Gambar 7. Mutasi [11]

Pseudocode dari GA seperti yang ditunjukkan pada Gambar 8.

Algorithm 1: Genetic Algorithm

Input: *numPop* : Number of base population, *numIter* : Number of iterations, *rateCross* : Rate of crossover, *rateMut* : Rate of mutation
Output : Global best solution

```
Generate initial population
Calculate initial population's fitness function
while (iteration ≤ numIter)
    Choose two solutions from population
    Form solutions of offspring via crossover
    if (rand(0,1) < rateMut)
        Mutate solution of offspring
    end if
    Calculate each child solution according to the fitness function
    Add a solution of offspring to the population
    Remove the rateCros x numPop least fit solutions from the population
end while
```

Gambar 8. Pseudocode Algoritma Genetika [13]

2.3. Tabu Search

Tabu search (TS) adalah metode metaheuristik yang diperkenalkan oleh Fred W. Glover pada tahun 1986. Ide dasarnya adalah untuk meningkatkan performa metode pencarian lokal yang sering terjebak dalam solusi suboptimal dengan cara melarang gerakan yang membawa solusi kembali ke ruang pencarian yang sudah pernah dikunjungi sebelumnya. Tujuan utama dari TS adalah untuk menyimpan solusi terbaik saat ini di memori sambil terus mencari solusi lain tanpa mengulangi pencarian sebelumnya

Dalam TS, beberapa istilah digunakan untuk membangun algoritma:

2.3.1. Tabu List

Tabu list adalah memori yang digunakan untuk menyimpan solusi yang sudah ditemukan sebelumnya. Solusi yang berada di *tabu list* dilarang dikunjungi lagi.

2.3.2. Tabu Tenure

Tabu tenure adalah jumlah iterasi dimana solusi yang ditemukan sebelumnya tetap berada di *tabu list*.

2.3.3. Kriteria Aspirasi

Kriteria aspirasi adalah fitur opsional dalam *tabu list*. Solusi yang merupakan bagian dari kriteria aspirasi dikeluarkan dari status *tabu*, artinya dapat dikunjungi kembali meskipun masih dalam *tabu list*. Kriteria aspirasi biasanya digunakan untuk memungkinkan mengunjungi solusi yang lebih baik daripada solusi terbaik saat ini dan untuk mencegah stagnasi ketika semua solusi masih berada dalam *tabu list*.

2.3.4. Memori Frekuensi

Memori frekuensi adalah memori yang digunakan untuk menyimpan jumlah iterasi yang diambil dari setiap solusi dari awal pencarian. Jika sebuah solusi sering dipilih, kemungkinan kecil akan dipilih lagi untuk memungkinkan pencarian yang beragam.

Pseudocode dari TS seperti yang ditunjukkan pada Gambar 9.

Algorithm 4: Tabu Search Algorithm

Input: - *tabuList*: Tabu list, - *tabuLen*: Length of tabuList
Output : tabuList

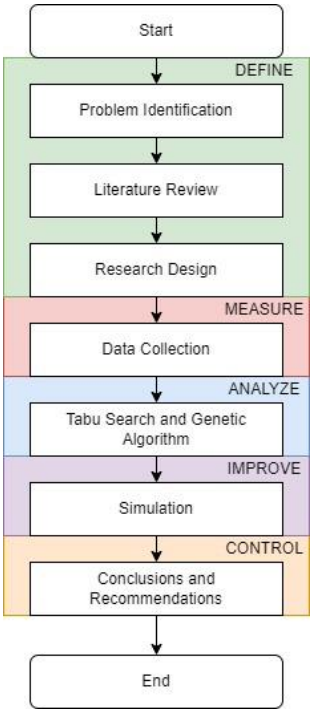
```
1. Initialize tabuList
2. Set tabuLen = 0
3. while (tabuLen < tabuLenMax)
4. |  tabuLen = tabuLen + 1
5. |  Search neighborhood
6. |  Calculate candidate solutions
7. |  Update tabuList
8. end while
```

Gambar 9. *Pseudocode Tabu Search* [13]

3. Metodologi Penelitian

Metodologi penelitian yang digunakan didasarkan pada kerangka kerja DMAIC. Pertama-tama, masalah yang ingin dipecahkan diidentifikasi. Selanjutnya, studi penelitian dan teori-teori dilakukan untuk membangun kerangka teoritis dengan mengeksplorasi pengetahuan dan dasar keilmuan yang sudah ada. Referensi seperti jurnal digunakan untuk memperluas kedalaman penelitian. Kemudian, desain penelitian dibangun untuk menunjukkan langkah-demi-langkah tentang bagaimana penelitian akan dilakukan dalam bentuk *flowchart*. Pada langkah ini, metode penelitian yang cocok untuk memecahkan masalah juga dipilih. Setelah itu, data sampel dikumpulkan untuk keperluan simulasi. Kemudian, data sampel digunakan untuk mencari parameter optimal untuk *tabu search* dan algoritma genetika. Setelah itu, simulasi dilakukan untuk membandingkan kinerja *tabu search* dan algoritma genetika. Akhirnya, kesimpulan dan rekomendasi dibuat di akhir penelitian.

Kerangka penelitian seperti yang ditunjukan Gambar 10.



Gambar 10. Kerangka Penelitian

4. Hasil dan Diskusi

4.1. Penyetelan Parameter Tabu Search

Data sampel digunakan dalam penyetelan parameter *tabu search*. Ada beberapa parameter *tabu search* yang harus disesuaikan, yaitu:

- Ukuran tabu list
- Ukuran tetangga

Yang pertama adalah ukuran *tabu list*. Untuk mendapatkan ukuran *tabu list* yang optimal, algoritma dijalankan selama 5 detik dengan ukuran tetangga 10 dan berbagai macam ukuran *tabu list*. Hasil terbaik didapatkan ketika ukuran *tabu list* 10 digunakan.

Yang kedua adalah ukuran tetangga. Untuk mendapatkan ukuran tetangga yang optimal, algoritma dijalankan selama 5 detik dengan ukuran *tabu list* 10 dan berbagai macam ukuran tetangga. Hasil terbaik didapatkan ketika ukuran tentangga 5 digunakan.

4.2. Penyetelan Parameter Algoritma Genetika

Data sampel yang digunakan dalam penyetelan parameter *tabu search* juga digunakan untuk penyetelan parameter algoritma genetika. Ada beberapa parameter algoritma genetika yang harus disesuaikan yaitu:

- Metode seleksi
- Ukuran seleksi (untuk metode turnamen)
- Probabilitas mutasi
- Ukuran populasi

Yang pertama adalah metode seleksi dan ukuran seleksi (untuk metode turnamen). *Library* JSSP yang dibuat oleh McFadden [14] menyediakan 3 metode seleksi yang dapat digunakan, seperti Roda Rolet, Turnamen, and Acak. Untuk menemukan metode dan ukuran seleksi yang optimal, algoritma dijalankan selama 5 detik dengan probabilitas mutasi 0.2, ukuran populasi 100, dan berbagai macam metode dan ukuran seleksi. Hasil terbaik didapatkan ketika metode turnamen dengan ukuran seleksi 2 digunakan.

Yang kedua adalah ukuran populasi. Untuk menemukan ukuran populasi yang optimal, algoritma dijalankan selama 5 detik dengan metode seleksi turnamen, ukuran seleksi 2, probabilitas mutasi 0.2, dan berbagai macam ukuran populasi. Hasil terbaik didapatkan ketika ukuran populasi 80 atau 100 atau 120 digunakan.

Yang ketiga adalah probabilitas mutasi. Untuk menemukan probabilitas mutasi yang optimal, algoritma dijalankan selama 5 detik dengan metode seleksi turnamen, ukuran seleksi 2, ukuran populasi 100, dan berbagai macam probabilitas mutasi. Hasil terbaik didapatkan ketika probabilitas mutasi 0.2 digunakan.

4.3. Komparasi Performa Tabu Search dan Algoritma Genetika

Dengan menggunakan parameter optimal untuk *tabu search* dan genetika algoritma yang didapatkan dari 4.1 dan 4.2, kedua metode di jalankan selama 10 detik untuk JSSP sederhana dan 30 detik untuk JSSP kompleks untuk membandingkan hasil *makespan*nya.

Hasil dari penyelesaian JSSP sederhana seperti di Tabel 2.

Tabel 2. Hasil Penyelesaian JSSP Sederhana

No.	***Makespan* Berdasarkan Metode Metaheuristik**	
	Tabu search	**Algoritma Generika**
1	1311	1338
2	1297	1379
3	1310	1379
4	1314	1401
5	1299	1343
6	1304	1383
7	1309	1343
8	1301	1350
9	1308	1340
10	1303	1365
Rata-rata	**1305,6**	**1362,1**

Tabel 2 menunjukan bahwa pada rata-rata, *makespan* terendah dihasilkan pada saat *tabu search* digunakan untuk menyelesaikan JSSP sederhana.

Hasil penyelesaian JSSP kompleks seperti di Tabel 3.

Tabel 3. Hasil Penyelesaian JSSP Kompleks

No.	***Makespan* Berdasarkan Metode Metaheuristik**	
	Tabu search	**Algoritma Genetika**
1	4111	4468
2	4112	4359
3	4131	4676
4	4118	4727
5	4123	4613
6	4119	4401
7	4115	4580
8	4116	4341
9	4116	4539
10	4112	4583
Rata-rata	**4117,3**	**4528,7**

Tabel 3 menunjukan bahwa pada rata-rata, *makespan* terendah dihasilkan pada saat *tabu search* digunakan untuk menyelesaikan JSSP kompleks.

5. Kesimpulan

Berdasarkan perbandingan kinerja *tabu search* sebagai metaheuristik dengan pencarian berbasis lokal dan algoritma genetika sebagai metaheuristik dengan pencarian berbasis populasi untuk menyelesaikan JSSP, hasil simulasi menunjukkan bahwa *makespan* terendah dicapai ketika *tabu search* digunakan untuk menyelesaikan JSSP sederhana dan kompleks. Oleh karena itu, dapat disimpulkan bahwa *tabu search* lebih unggul daripada algoritma genetika untuk menyelesaikan JSSP. Selain itu, untuk JSSP yang terkait dengan penelitian ini, parameter yang paling optimal untuk *tabu search* adalah ukuran *tabu list* 10 dan ukuran tetangga 5. Parameter yang paling optimal untuk algoritma genetika adalah metode seleksi turnamen dengan ukuran seleksi 2, ukuran populasi 100, dan probabilitas mutasi 0.2. Untuk kedepannya, akan menarik untuk membandingkan metaheuristik berbasis pencarian lokal dan metaheuristik berbasis pencarian populasi lainnya untuk menyelesaikan JSSP serta mengamati kinerja kombinasi *tabu search* dan algoritma genetika.

Referensi

[1] M. R. Garey and D. S. Johnson, *Computers and intractability. A guide to the theory of NP-completeness*. Bell Telephone Laboratories, Incorporated, 1979.

[2] B. . Lageweg, J. . Lenstra, and A. H. G. Rinnooy Kan, "Job-Shop Scheduling by Implicit Enumeration," *Manage. Sci.*, vol. 24, no. 4, pp. 441–450, 1997.

[3] M. Abdolrazzagh-Nezhad and S. Abdullah, "Job Shop Scheduling: Classification, Constraints and Objective Functions," *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 11, no. 4, pp. 429–434, 2017.

[4] J. Mohan, K. Lanka, and A. Neelakanteswara Rao, "A Review of Dynamic Job Shop Scheduling Techniques," *Procedia Manuf.*, vol. 30, pp. 34–39, 2019, doi: 10.1016/j.promfg.2019.02.006.

[5] A. Lala, V. Kolici, F. Xhafa, X. Herrero, and A. Barolli, "On Local vs. Population-Based Heuristics for Ground Station Scheduling," *Proc. - 2015 9th Int. Conf. Complex, Intelligent, Softw. Intensive Syst. CISIS 2015*, pp. 267–275, 2015, doi: 10.1109/CISIS.2015.40.

[6] M. A. Bozorgirad and R. Logendran, "A comparison of local search algorithms with population-based algorithms in hybrid flow shop scheduling problems with realistic characteristics," *Int. J. Adv. Manuf. Technol.*, vol. 83, no. 5–8, pp. 1135–1151, 2016, doi: 10.1007/s00170-015-7650-9.

[7] S. J. Beaty, "Genetic Algorithms versus Tabu Search for Instruction Scheduling," *Artif. Neural Nets Genet. Algorithms*, no. Ddd, pp. 496–501, 1993, doi: 10.1007/978-3-7091-7533-0_72.

[8] S. C. Chu and H. L. Fang, "Genetic algorithms vs. Tabu search in timetable scheduling," *Int. Conf. Knowledge-Based Intell. Electron. Syst. Proceedings, KES*, pp. 492–495, 1999, doi: 10.1109/kes.1999.820230.

[9] C. Renman and H. Fristedt, "A comparative analysis of a Tabu Search and a Genetic Algorithm for solving a University Course Timetabling Problem," 2015, [Online]. Available: https://www.diva-portal.org/smash/get/diva2:810264/FULLTEXT01.pdf.

[10] S. E. Ramadhania and S. Rani, "Implementasi Kombinasi Algoritma Genetika dan Tabu Search untuk Penyelesaian Travelling Salesman Problem," *Automata*, 2021.

[11] V. Mallawaarachchi, "Introduction to Genetic Algorithms — Including Example Code," 2017. www.towardsdatascience.com.

[12] M. F. Sobeih, M. I. Doma, and A. F. El Shoney, "Mixture-Order Design of Gps Networks Based on Genetic Algorithms," *ERJ. Eng. Res. J.*, vol. 33, no. 4, pp. 431–439, 2010, doi: 10.21608/erjm.2010.67343.

[13] C. Cebi, E. Atac, and O. K. Sahingoz, "Job Shop Scheduling Problem and Solution Algorithms: A Review," *2020 11th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2020*, 2020, doi: 10.1109/ICCCNT49239.2020.9225581.

[14] M. McFadden, "Job Shop Schedule Problem," 2019. https://github.com/mcfadd/Job_Shop_Schedule_Problem.