EVALUASI KONTRIBUSI SOLUSI AWAL HEURISTIK TERHADAP PERFORMA *SIMULATED ANNEALING* PADA *JOB SHOP SCHEDULING PROBLEM*

SKRIPSI

OLEH
ANIFATUN NISA'
NIM. 220601110011

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHI MALANG
2026

EVALUASI KONTRIBUSI SOLUSI AWAL HEURISTIK TERHADAP PERFORMA *SIMULATED ANNEALING* PADA *JOB SHOP SCHEDULING PROBLEM*

SKRIPSI

Diajukan Kepada
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Matematika (S.Mat)

Oleh
Anifatun Nisa'
NIM. 220601110011

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHI MALANG
2026

EVALUASI KONTRIBUSI SOLUSI AWAL HEURISTIK TERHADAP PERFORMA *SIMULATED ANNEALING* PADA *JOB SHOP SCHEDULING PROBLEM*

SKRIPSI

Oleh
Anifatun Nisa'
NIM. 220601110011

Telah Disetujui Untuk Diuji

Malang, 21 Mei 2026

Dosen Pembimbing I	Dosen Pembimbing II
Mohammad Nafie Jauhari, M.Si	Erna Herawati, M.Pd
NIPPPK. 19870218 202321 1 018	NIPPPK. 19760723 202321 2 006

Mengetahui,
Ketua Program Studi Matematika

Dr. Ahmad Luthfi, M.Si
NIP. 19800527 200801 1 012

EVALUASI KONTRIBUSI SOLUSI AWAL HEURISTIK TERHADAP PERFORMA *SIMULATED ANNEALING* PADA *JOB SHOP SCHEDULING PROBLEM*

SKRIPSI

Oleh
Anifatun Nisa'
NIM. 220601110011

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Matematika (S.Mat)

Tanggal 10 Juni 2026

Ketua Penguji	:	Hisyam Fahmi, M.Kom	.....
Anggota Penguji 1	:	Muhammad Khudzaifah, M.Si	.....
Anggota Penguji 2	:	Mohammad Nafie Jauhari, M.Si	.....
Anggota Penguji 3	:	Erna Herawati, M.Pd	.....

Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M.Si
NIP. 19800527 200801 1 012

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini:

Nama	: Anifatun Nisa'
NIM	: 220601110011
Program Studi	: Matematika
Fakultas	: Sains dan Teknologi
Judul Skripsi	: Evaluasi Kontribusi Solusi Awal Heuristik terhadap Performa *Simulated Annealing* pada *Job Shop Scheduling Problem*

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya sendiri, bukan merupakan pengambilan data, tulisan, pikiran orang lain yanag saaya akui sebagai hasil tulisan dan pikiran saya sendiri, kecuali dengan mencantumkan sember cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 10 Juni 2026

Yang membuat pernyataan,

Anifatun Nisa'

NIM.220601110011

MOTO

"Mustahil Allah membawaku sejauh ini hanya untuk gagal"

"*Sesungguhnya bersama kesulitan ada kemudahan*"
(QS. Al-Insyirah: 6)

PERSEMBAHAN

Bismillahirrahmanirrahim

Segala puji bagi Allah SWT yang senantiasa memeberikan pertolongan dan kemudahan kepada penulis dalam melewati segala proses penyelesaian skripsi ini. Dengan penuh rasa syukur, skripsi ini penulis persembahkan kepada:

Kedua orang tua penulis Bapak Sagi dan Ibu Rupini yang senantiasa memberikan do'a, dukungan, pengorbanan, serta nasihat terbaik pada setiap langkah penulis. Seluruh keluarga yang senantiasa memberikan semangat, do'a, dan dukungan selama proses penyusunan skripsi ini. Sahabat dan teman-teman seperjuangan yang selalu memberikan bantuan, dukungan, dan kebersamaan sehingga skripsi ini dapat terselesaikan dengan baik. Almamater tercinta yang menjadi tempat penulis menimba ilmu.

KATA PENGANTAR

Puji syukur kehadirat Allah SWT. atas segala limpahan rahmat, taufik, dan hidayah-Nya sehingga skripsi ini dapat diselesaikan dengan lancar. Shalawat dan salam tidak lupa senantiasa tercurah limpahkan kepada baginda Nabi Agung Muhammad SAW yang telah membawa kita dari zaman jahiliyyah ke zaman yang terang benderan yaitu *addinul Islam*.

Terselesaikannya skripsi ini tidak lepas dari doa, bantuan, bimbingan, serta arahan dari berbagai pihak, karena itu peneliti mengucapkan terimakasih kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si., CAHRM., CRMP, selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim.
2. Dr. Agus Mulyono, M.Kes, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim.
3. Dr. Fachrur Rozi, M.Si, selaku Ketua Program Studi Matematika, Universitas Islam Negeri Maulana Malik Ibrahim.
4. Mohammad Nafie Jauhari, M.Si, selaku Dosen Pembimbing yang senantiasa membimbing dan memberikan berbagai pengetahuan, nasehat, motivasi, dan arahan selama proses penyusunan skripsi ini.
5. Erna Herawati, M.Pd selaku Dosen Pembimbing yang senantiasa membimbing dan memberikan nasehat, motivasi, dan arahan selama proses penyusunan skripsi ini.
6. Hisyam Fahmi, M.Kom, selaku Dosen Penguji yang telah memberikan arahan dan masukan dalam menyusun skripsi ini.
7. Muhammad Khudzaifah, M.Si, selaku Dosen Penguji yang telah memberikan arahan dan masukan dalam menyusun skripsi ini.
8. Seluruh Dosen Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim yang telah memberikan ilmu, wawasan, dan pengalaman berharga selama masa perkuliahan.
9. Kedua orang tua tercinta, Ibu Rupini dan Bapak Sagi, serta seluruh keluarga besar yang senantiasa memberikan doa, kasih sayang, motivasi, dan dukungan baik moral maupun material yang tidak pernah terhenti dalam setiap perjuangan penyusunan skripsi ini.

10. Teman-teman seperjuangan mahasiswa Matematika angkatan 2022 yang selalu memberikan semangat, kebersamaan, dan kerja sama yang baik selama menempuh perjalanan akademik hingga tahap ini.
11. Seluruh pihak yang sudah terlibat secara langsung maupun tidak langsung yang tidak bisa penulis tuliskan satu per satu.

Dengan seluruh kerendahan hati, peneliti menyadari skripsi ini masih belum bisa dikatakan sempurna. Maka dari itu, peneliti berharap kritik dan saran yang membangun untuk penyempurnaan skripsi ini. Semoga karya ini mampu memberikan manfaat bagi orang banyak. *Aamin ya Rabbal 'Alamin*.

Malang, 10 Juni 2026

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SIMBOL

$C_{\max}$	: Waktu selesai operasi terakhir dari semua pekerjaan
O_{ij}	: Operasi ke-j dari pekerjaan ke-i
S_{ij}	: Waktu mulai operasi ke-j dari pekerjaan ke-i
p_{ij}	: Waktu proses operasi ke-j dari pekerjaan ke-i
M_{ij}	: Mesin yang digunakan oleh operasi ke-j dari pekerjaan ke-i
$(i,j),(k,l)$	: Dua operasi berbeda pada mesin yang sama
P	: Probabilitas menerima solusi baru yang lebih buruk
ΔE	: Perubahan nilai fungsi objektif
T	: Parameter kontrol yang menurun secara bertahap
exp	: Fungsi eksponensial

DAFTAR LAMPIRAN

ABSTRAK

Nisa', Anifatun. 2026. **Evaluasi Kontribusi Solusi Awal Heuristik terhadap Performa *Simulated Annealing* pada *Job Shop Scheduling Problem*.** Skripsi. Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Pembimbing: (I) Mohammad Nafie Jauhari, M.Si. (II) Erna Herawati, M.Pd.

Kata Kunci: *Job Shop Scheduling Problem*, *Makespan*, *Shortest Processing Time*, *Simulated Annealing*.

Penelitian ini membahas evaluasi kontribusi solusi awal heuristik berbasis *Shortest Processing Time* (SPT) terhadap performa algoritma *Simulated Annealing* (SA) dalam menyelesaikan *Job Shop Scheduling Problem* (JSSP). Permasalahan JSSP merupakan masalah optimasi penjadwalan yang bertujuan meminimalkan nilai *makespan* dengan memperhatikan urutan operasi dan keterbatasan mesin. Penelitian ini membandingkan dua pendekatan, yaitu *Simulated Annealing* dengan solusi awal acak dan *Simulated Annealing* dengan solusi awal berbasis algoritma *greedy* menggunakan aturan SPT. Dataset yang digunakan adalah dataset *benchmark* FT06 yang terdiri atas enam pekerjaan dan enam mesin. Parameter SA yang digunakan meliputi temperatur awal sebesar 100, *cooling rate* sebesar 0,95, temperatur minimum sebesar 0,1, dan iterasi per suhu sebanyak 10. Evaluasi performa dilakukan melalui pengujian sebanyak 30 percobaan dengan indikator nilai *makespan*, stabilitas hasil, konvergensi, frekuensi pencapaian solusi terbaik, serta uji statistik menggunakan uji Mann-Whitney U. Hasil penelitian menunjukkan bahwa penggunaan solusi awal berbasis SPT memberikan performa yang lebih stabil dan menghasilkan rata-rata *makespan* yang lebih baik dibandingkan solusi awal acak. Selain itu, pendekatan SA berbasis SPT juga menunjukkan proses konvergensi yang lebih cepat dan konsisten dalam mencapai solusi terbaik. Namun, penggunaan solusi awal heuristik berbasis SPT belum memberikan pengaruh yang signifikan terhadap kualitas solusi akhir yang dihasilkan oleh algoritma *Simulated Annealing*. Hal ini menunjukkan bahwa algoritma *Simulated Annealing* memiliki kemampuan eksplorasi solusi yang baik. Penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan metode optimasi penjadwalan, khususnya terkait pengaruh solusi awal heuristik terhadap performa algoritma metaheuristik.

ABSTRACT

Nisa', Anifatun. 2026. **Evaluation of Heuristic Initial Solution Contribution to the Performance of Simulated Annealing in Job Shop Scheduling Problem.** Thesis. Mathematics Study Program, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Advisors: (I) Mohammad Nafie Jauhari, M.Si. (II) Erna Herawati, M.Pd.

Keywords: Job Shop Scheduling Problem, Makespan, Shortest Processing Time, Simulated Annealing.

This study discusses the evaluation of the contribution of a heuristic initial solution based on Shortest Processing Time (SPT) to the performance of the Simulated Annealing (SA) algorithm in solving the Job Shop Scheduling Problem (JSSP). JSSP is a scheduling optimization problem that aims to minimize the makespan value while considering operation sequences and machine constraints. This research compares two approaches, namely Simulated Annealing with a random initial solution and Simulated Annealing with an initial solution generated using the greedy algorithm with the SPT rule. The dataset used in this study is the FT06 benchmark dataset consisting of six jobs and six machines. The SA parameters include an initial temperature of 100, a cooling rate of 0.95, a minimum temperature of 0.1, and 10 iterations per temperature level. Performance evaluation was carried out through 30 experimental runs using indicators such as makespan, result stability, convergence analysis, frequency of achieving the best solution, and statistical testing using the Mann-Whitney U test. The results indicate that the use of an SPT-based initial solution provides more stable performance and produces a better avera ge makespan compared to the random initial solution. However, the use of an SPT-based heuristic initial solution does not significantly affect the quality of the final solution produced by the Simulated Annealing algorithm. This finding indicates that Simulated Annealing possesses a strong capability to explore the solution space effectively. Furthermore, the SPT-based SA approach demonstrates faster and more consistent convergence in achieving the best solution. This study is expected to contribute to the development of scheduling optimization methods, particularly regarding the influence of heuristic initial solutions on the performance of metaheuristic algorithms.

مستلص البحث

النساء، أَنِيفَة. ٢٠٢٦م. **تقييم مساهمة الحل الابتدائي الاسترشادي في أداء خوارزمية التلدين المُحاكى لمسألة جدولة الورش الإنتاجية**. البحث العلمي. قسم الرياضيات، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج.

المشرف: (١) محمد نافع جوهري، الماجستير في العلوم. (٢) ايرناهيراواتي، الماجستيره في التربية.

الكلمات الأساسية: الخوارزمية الجشعة، مسألة جدولة الورش الإنتاجية، زمن الإنجاز الكلي، أقصر زمن معالجة، التلدين المحاكى.

تناقش هذه الدراسة تقييم مساهمة الحل الابتدائي الاسترشادي القائم على قاعدة **أقصر زمن معالجة** (*Shortest Processing Time – SPT*) في أداء خوارزمية **التلدين المحاكى** (*Simulated Annealing*) لحل **مشكلة جدولة الورش الإنتاجية**. (*Job Shop Scheduling Problem – JSSP*) وتُعد هذه المشكلة من مسائل تحسين الجدولة التي تهدف إلى تقليل قيمة زمن الإنجاز الكلي (*Makespan*) مع مراعاة تسلسل العمليات والقيود المتعلقة بالآلات. تقارن الدراسة بين منهجين، هما خوارزمية التلدين المحاكى ذات الحل الابتدائي العشوائي، وخوارزمية التلدين المحاكى ذات الحل الابتدائي المبني باستخدام الخوارزمية الجشعة وفق قاعدة *SPT*. وقد استُخدمت مجموعة البيانات القياسية FT06، والتي تتكون من ست وظائف وست آلات. وشملت معلمات الخوارزمية درجة حرارة ابتدائية مقدارها ١٠٠، ومعدل تبريد مقداره ٠،٩٥، ودرجة حرارة دنيا مقدارها ٠،١، وعدد ١٠ تكرارات لكل مستوى حراري. وتم تقييم الأداء من خلال إجراء ٣٠ تجربة اعتمادًا على مؤشرات زمن الإنجاز الكلي، واستقرار النتائج، وسلوك التقارب، وتكرار الوصول إلى أفضل حل، بالإضافة إلى الاختبار الإحصائي مان–ويتني U. وأظهرت النتائج أن الحل الابتدائي القائم على قاعدة *SPT* يوفر أداءً أكثر استقرارًا ويحقق متوسط زمن إنجاز كلي أفضل مقارنةً بالحل الابتدائي العشوائي. كما أظهر هذا النهج سرعة أكبر واتساقًا أفضل في التقارب نحو أفضل الحلول. ومع ذلك، فإن استخدام الحل الابتدائي الاسترشادي القائم على قاعدة *SPT* لم يُظهر تأثيرًا معنويًا على جودة الحل النهائي الناتج عن خوارزمية التلدين المحاكى. وتشير هذه النتيجة إلى أن خوارزمية التلدين المحاكى تمتلك قدرة جيدة على استكشاف فضاء الحلول بفاعلية. ومن المتوقع أن تسهم هذه الدراسة في تطوير أساليب تحسين الجدولة، ولا سيما فيما يتعلق بتأثير الحلول الابتدائية الاسترشادية على أداء الخوارزميات الميتاهيوريستية.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penjadwalan pekerjaan pada sistem manufaktur merupakan aspek penting dalam meningkatkan efisiensi produksi dan pemanfaatan sumber daya secara optimal (Pérez dkk., 2024). Salah satu permasalahan penjadwalan yang kompleks adalah *Job Shop Scheduling Problem* (JSSP), yang muncul ketika beberapa pekerjaan harus dikerjakan secara bersamaan dengan sumber daya terbatas, khususnya mesin yang harus digunakan secara bergantian sesuai urutan proses tertentu (Puspitasari, 2016). JSSP dikategorikan sebagai masalah NP-*hard*, yang berarti kompleksitasnya sangat tinggi dan sulit diselesaikan secara eksak untuk skala besar (Gromicho dkk., 2012). Oleh karena itu, diperlukan metode optimasi yang mampu menghasilkan solusi penjadwalan dengan *makespan* minimum dalam waktu komputasi yang relatif efisien.

Dalam Islam, kerja dan pengelolaan waktu adalah aspek penting yang diatur dalam Al Qur'an. Al Qur'an menekankan pentingnya memanfaatkan waktu secara produktif dan efektif demi mencapai keberhasilan dunia dan akhirat. Allah SWT berfirman dalam Surah Al-'Asr ayat 1-3:

وَالْعَصْرِۙ ﴿١﴾ اِنَّ الْاِنْسَانَ لَفِيْ خُسْرٍۙ ﴿٢﴾ اِلَّا الَّذِيْنَ اٰمَنُوْا وَعَمِلُوا الصّٰلِحٰتِ وَتَوَاصَوْا بِالْحَقِّ ۙ وَتَوَاصَوْا بِالصَّبْرِ ﴿٣﴾
(Kementrian Agama Republik Indonesia, 2022)

"*Demi masa. Sesungguhnya manusia itu benar-benar dalam kerugian, kecuali orang-orang yang beriman dan mengerjakan amal saleh serta saling menasihati dengan kebenaran dan saling menasihati dengan kesabaran.*" (QS. Al-'Asr: 1-3).

Ayat ini menegaskan urgensi waktu sebagai amanah yang tidak boleh disia-siakan. Manusia dikatakan rugi jika tidak memanfaatkan waktu untuk beriman,

beramal saleh, dan saling menasihati yang secara luas dapat dimaknai sebagai usaha perbaikan diri dan lingkungan. Dalam konteks pekerjaan dan manajemen produksi, ayat ini mengajarkan pentingnya efisiensi dan produktivitas sebagai bagian dari amal yang bermanfaat.

Menurut Ibnu Katsir, ayat ini menafsirkan bahwa manusia akan rugi kecuali mereka yang memiliki empat kualifikasi: iman, amal saleh, saling menasihati dalam kebenaran, dan saling menasihati dalam kesabaran. Allah SWT bersumpah dengan "masa" (*al-'ashr*) untuk menekankan betapa berharganya waktu, yang sering kali disia-siakan oleh manusia (Riki dkk., 2024). Tafsir Al-Misbah oleh Quraish Shihab tentang ayat ini memberikan pemahaman yang mendalam tentang pentingnya waktu, bagaimana cara memanfaatkannya, dan bagaimana menghindari kerugian yang disebabkan oleh penyia-nyiaannya (Nurchoironi & Nurrohim, 2025).

Dalam konteks optimasi penjadwalan produksi seperti JSSP, prinsip-prinsip tersebut dapat menjadi landasan etika dan moral dalam mengembangkan metode ilmiah yang bertujuan meningkatkan efisiensi waktu dan sumber daya. Dengan mengintegrasikan nilai-nilai Al Qur’an dan Hadist, penelitian ini tidak hanya bertujuan menyelesaikan masalah teknis, tetapi juga mengedepankan pendekatan kerja yang bermanfaat, produktif dan sesuai dengan tuntunan agama yang mulia.

Metode heuristik dan metaheuristik menjadi pendekatan yang sering digunakan dalam mengatasi JSSP. Meskipun metode heuristik memiliki waktu komputasi yang rendah dan hasilnya tidak selalu menjamin solusi optimal. Sementara itu, metode metaheuristik mampu menghasilkan solusi yang mendekati optimal dengan waktu komputasi yang relatif singkat (Abdolrazzagh-Nezhad & Abdullah, 2024; Van Ekeris dkk., 2021). Salah satu algoritma metaheuristik yang

sering digunakan adalah *Simulated Annealing* (SA), yaitu algoritma pencarian stokastik yang mampu menerima solusi yang lebih buruk sementara waktu untuk menghindari jebakan *local optimum* dan mendekati solusi global optimum.

Di sisi lain, algoritma *greedy* merupakan salah satu metode heuristik yang sederhana dan cepat, dengan prinsip memilih solusi terbaik secara lokal pada setiap langkah untuk membangun solusi awal yang baik (Irwan, 2020). Salah satu aturan *greedy* yang umum digunakan dalam penjadwalan adalah *Shortest Processing Time* (SPT), yaitu memilih operasi dengan waktu proses paling singkat terlebih dahulu. Pendekatan ini mampu menghasilkan solusi awal dengan cepat dan sederhana, meskipun sering kali menghasilkan solusi yang bersifat lokal dan belum tentu optimal secara global.

Beberapa penelitian terdahulu menunjukkan *Simulated Annealing* (SA) terbukti mampu menghasilkan solusi mendekati optimal melalui proses eksplorasi stokastik (Chakraborty & Bhowmik, 2013; Utomo & Setiafindari, 2021). Namun, kelemahan SA terletak pada waktu konvergensi yang lama dan ketergantungan tinggi terhadap parameter awal. Di sisi lain, penelitian oleh Irwan (2020) dan Khader dkk. (2018) menunjukkan bahwa algoritma *greedy* mampu memberikan hasil penjadwalan yang cepat dan sederhana melalui pemilihan solusi lokal terbaik di setiap langkah, meskipun sering kali terjebak pada solusi suboptimal. Berdasarkan karakteristik tersebut, penggunaan solusi awal berbasis *greedy* pada *Simulated Annealing* menjadi menarik untuk dievaluasi lebih lanjut guna mengetahui sejauh mana solusi awal heuristik benar-benar berkontribusi terhadap performa pencarian solusi pada JSSP.

Penilaian performa metode dalam menyelesaikan JSSP memerlukan adanya ukuran atau indikator evaluasi yang bersifat objektif. Keberhasilan metode dapat diukur dengan metrik objektif seperti *makespan*, total waktu tunggu, atau waktu penyelesaian agar efektivitas dan perbaikan kualitas jadwal dapat terukur secara jelas (Pérez dkk., 2024). Penelitian ini menggunakan *makespan* sebagai indikator utama untuk membandingkan performa *Simulated Annealing* dengan solusi awal acak dan *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT).

Oleh karena itu, penelitian ini penting karena tidak hanya membandingkan dua pendekatan algoritma, tetapi juga menguji asumsi umum bahwa hibridisasi heuristik selalu memberikan peningkatan performa pada metaheuristik. Selain itu, penelitian ini juga bertujuan untuk menganalisis apakah solusi awal heuristik benar-benar memberikan kontribusi terhadap kualitas solusi akhir, stabilitas hasil, maupun proses konvergensi *Simulated Annealing*. Dengan demikian, penelitian ini diharapkan dapat memberikan pemahaman mengenai sejauh mana pengaruh solusi awal heuristik terhadap performa *Simulated Annealing* pada permasalahan *Job Shop Scheduling Problem*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah dalam penelitian ini adalah bagaimana perbandingan performa *Simulated Annealing* dengan solusi awal acak dan *Simulated Annealing* dengan solusi awal berbasis *Greedy Shortest Processing Time* pada *Job Shop Scheduling Problem* ditinjau dari nilai *makespan*.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah mengevaluasi kontribusi solusi awal berbasis *Shortest Processing Time* terhadap performa *Simulated Annealing* dalam menyelesaikan *Job Shop Scheduling Problem*.

1.4 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Memberikan kontribusi dalam kajian optimasi penjadwalan produksi, khususnya mengenai pengaruh solusi awal heuristik terhadap performa algoritma *Simulated Annealing* pada *Job Shop Scheduling Problem*.
2. Menyediakan referensi teknis dan ilmiah bagi peneliti maupun praktisi terkait evaluasi penggunaan solusi awal berbasis heuristik pada algoritma metaheuristik dalam permasalahan penjadwalan.
3. Memberikan gambaran mengenai performa *Simulated Annealing* dengan solusi awal acak dan solusi awal berbasis *Shortest Processing Time* (SPT) ditinjau dari *makespan*, konvergensi, stabilitas hasil, dan konsistensi solusi.
4. Menjadi bahan pertimbangan dalam pemilihan pendekatan optimasi penjadwalan yang lebih sederhana dan efisien sesuai karakteristik permasalahan yang dihadapi.

1.5 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Permasalahan yang digunakan dalam penelitian ini adalah *Job Shop Scheduling Problem* (JSSP).
2. Dataset yang digunakan adalah dataset *benchmark* FT06.

3. Algoritma yang digunakan adalah *Simulated Annealing* dengan solusi awal acak dan *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT).
4. Parameter *Simulated Annealing* yang digunakan meliputi temperatur awal (T_0) = 100, *cooling rate* (α) = 0,95, temperatur minimum (T_{min}) = 0,1, dan iterasi per suhu sebanyak 10.
5. Evaluasi performa algoritma difokuskan pada parameter *makespan*, grafik konvergensi, stabilitas hasil, frekuensi pencapaian solusi terbaik, dan uji statistik.
6. Penelitian ini tidak membahas aspek biaya produksi, kualitas produk, maupun faktor sumber daya manusia dalam proses penjadwalan.
7. Implementasi algoritma dilakukan menggunakan bahasa pemrograman *Python*.

1.6 Definisi Istilah

1. Optimasi

   Optimasi atau *optimization* merupakan kegiatan untuk mencapai hasil terbaik dalam kondisi yang telah ditentukan (Manik dkk., 2018).

2. *Job Shop Scheduling Problem*

   Job Shop Scheduling Problem (JSSP) merupakan permasalahan optimasi kombinatorial klasik yang bertujuan menjadwalkan sejumlah pekerjaan (*jobs*) pada mesin-mesin tertentu dengan urutan operasi yang telah ditentukan, guna meminimalkan waktu penyelesaian total atau *makespan* (Guzman dkk., 2022).

3. Algoritma *Greedy*

   Algoritma *greedy* metode yang digunakan untuk mendapatkan penyelesaian dari suatu permasalahan optimasi (Basriati dkk., 2020). Menurut (Khader dkk., 2018) algoritma *greedy* adalah salah satu varian algoritma *Best First Search* yang paling mudah, karena hanya memperhitungkan biaya estimasi (*estimated cost*) saja.

4. *Simulated Annealing*

   Algoritma *simulated annealing* merupakan suatu metode optimasi yang terinspirasi dari proses *annealing* pada metalurgi, yaitu prosedur pemanasan dan pendinginan material logam secara bertahap. Mekanisme tersebut diadaptasi dalam konteks komputasional untuk memungkinkan pencarian solusi yang lebih efisien dan mendekati optimal melalui pengendalian perubahan energi secara bertahap (Samana dkk., 2015).

5. *Makespan*

   Makespan merupakan total waktu penyelesaian seluruh pekerjaan, dimulai dari operasi pertama yang diproses pada mesin atau *work center* awal hingga operasi terakhir yang diselesaikan pada mesin atau *work center* akhir (Kurniawati, 2019).

6. *Gantt Chart*

   Gantt chart merupakan diagram berbentuk batang yang digunakan untuk merepresentasikan durasi setiap aktivitas dalam suatu proyek (Russel & Taylor, 2011).

BAB II

KAJIAN TEORI

2.1 Teori Pendukung

Sebuah penelitian memerlukan teori-teori sebagai bahan rujukan dalam prosesnya. Berikut adalah teori-teori yang membantu memahami jalannya penelitian ini:

2.1.1 Optimasi

Optimasi atau *optimization* merupakan kegiatan untuk mencapai hasil terbaik dalam kondisi yang telah ditentukan (Manik dkk., 2018). Menurut Devita & Wibawa (2020) optimasi adalah masalah komputasional yang bertujuan menemukan solusi optimal dari berbagai pilihan alternatif yang tersedia, dengan mematuhi sejumlah batasan (*constraints*). Pada dasarnya, tujuan optimasi adalah memperoleh respons atau tindakan yang paling tepat sesuai dengan keadaan yang diinginkan (Fauzi dkk., 2022).

2.1.2 *Job Shop Scheduling Problem*

Job Shop Scheduling Problem (JSSP) merupakan permasalahan optimasi kombinatorial klasik yang bertujuan menjadwalkan sejumlah pekerjaan (*jobs*) pada mesin-mesin tertentu dengan urutan operasi yang telah ditentukan, guna meminimalkan waktu penyelesaian total atau *makespan* (Guzman dkk., 2022). JSSP termasuk kategori masalah NP-*hard*, jadi solusi optimal sulit diperoleh untuk skala besar, sehingga diperlukan pendekatan heuristik atau metaheuristik (Gromicho dkk., 2012).

Menurut Miloš Šeda (2007), misalkan terdapat n pekerjaan dan m mesin. Setiap pekerjaan J_i terdiri atas rangkaian operasi $O_{i1}, O_{i2}, \dots, O_{i,m_i}$ yang harus

diproses secara berurutan. Setiap operasi O_{ij} wajib dijalankan pada mesin tertentu M_{ij} dan memiliki waktu proses sebesar p_{ij}. Variabel keputusan utama dalam model ini adalah waktu mulai operasi S_{ij}, sedangkan waktu selesainya didefinisikan sebagai $C_{ij} = S_{ij} + p_{ij}$. Tujuan optimasi dalam JSSP adalah meminimalkan *makespan*, yaitu waktu penyelesaian operasi terakhir dari seluruh pekerjaan. Secara matematis, fungsi objektif tersebut dituliskan sebagai (Miloš Šeda, 2007):

$$\min C_{\max} = \min\left\{\max_{i,j}\left(S_{ij} + p_{ij}\right)\right\} \tag{2.1}$$

Formulasi tersebut dibatasi oleh dua kelompok kendala utama. Pertama, kendala urutan operasi antar-operasi dalam satu pekerjaan, yang menyatakan bahwa operasi ke-$j+1$ dari suatu pekerjaan tidak dapat dimulai sebelum operasi sebelumnya selesai. Hal ini dirumuskan sebagai

$$S_{i,j+1} \geq S_{ij} + p_{ij}, \quad \forall\, i = 1, \ldots, n;\ j = 1, \ldots, m_i - 1,$$

Kedua, terdapat kendala kapasitas mesin yang menyatakan bahwa setiap mesin hanya dapat memproses satu operasi pada satu waktu. Untuk dua operasi berbeda yang dikerjakan pada mesin yang sama, yaitu $M_{ij} = M_{kl}$, salah satu dari kedua operasi tersebut harus dikerjakan lebih dahulu sehingga hubungan keduanya dinyatakan dalam bentuk disjungsi:

$$S_{ij} \geq S_{kl} + p_{kl} \text{ atau } S_{kl} \geq S_{ij} + p_{ij}\ , \forall (i,j) \neq (k,l), M_{ij} = M_{kl}.$$

Keterangan:

$C_{\max}$: Waktu selesai operasi terakhir dari semua pekerjaan.

S_{ij} : Waktu mulai operasi ke-j dari pekerjaan ke-i.

p_{ij} : Waktu proses operasi ke-j dari pekerjaan ke-i.

M_{ij} : Mesin yang digunakan operasi ke-j dari pekerjaan ke-i.

$(i,j),(k,l)$: Dua operasi berbeda pada mesin yang sama.

2.1.3 Algoritma *Greedy*

Algoritma *greedy* metode yang digunakan untuk mendapatkan penyelesaian dari suatu permasalahan optimasi (Basriati dkk., 2020). Menurut Khader dkk. (2018) algoritma *greedy* adalah salah satu varian algoritma *Best First Search* yang paling mudah, karena hanya memperhitungkan biaya estimasi (*estimated cost*) saja, yaitu $f(n) = h(n)$. Algoritma *greedy* termasuk dalam kategori algoritma yang selalu memilih solusi sementara atau lokal terbaik pada setiap tahapnya untuk mengatasi suatu masalah. Pilihan optimal akan dipilih di setiap langkah tanpa memikirkan dampaknya terhadap hasil akhir secara menyeluruh.

Algoritma *greedy* dalam konteks JSSP memilih operasi berikutnya yang dapat segera diproses berdasarkan kriteria tertentu, misalnya operasi dengan waktu proses paling singkat atau operasi yang tersedia lebih awal (Irwan, 2020). Algoritma ini membangun solusi secara bertahap dengan keputusan lokal terbaik tanpa mempertimbangkan konsekuensi jangka panjang, sehingga cepat namun berisiko terjebak pada solusi lokal. Pada penelitian ini, pendekatan *greedy* yang digunakan adalah aturan *Shortest Processing Time* (SPT). SPT adalah aturan penjadwalan yang secara *greedy* memprioritaskan operasi dengan waktu proses paling singkat untuk dikerjakan terlebih dahulu. Dengan memilih operasi berdurasi pendek pada setiap langkah, aturan ini diharapkan mampu mengurangi waktu tunggu dan mempercepat penyelesaian pekerjaan secara keseluruhan (Darnita & Toyib, 2019).

Berbeda dengan penelitian *hybrid* yang bertujuan mengembangkan metode baru, pada penelitian ini algoritma *greedy* tidak digunakan sebagai metode optimasi utama, melainkan sebagai pembentuk solusi awal bagi algoritma *Simulated Annealing*. Solusi awal berbasis SPT tersebut kemudian dievaluasi untuk mengetahui sejauh mana pengaruhnya terhadap performa pencarian solusi pada JSSP. Proses pembentukan solusi awal menggunakan algoritma *greedy* dengan aturan SPT dapat dilihat pada *pseudocode* berikut:

Algoritma 2.1 Algoritma *Greedy Shortest Processing Time* (SPT)

Input: Set pekerjaan J, set mesin M, dan waktu proses p_{ij} untuk setiap operasi O_{ij}
Output: Jadwal operasi dan nilai makespan ($C_{\max}$)

Langkah 1: Inisialisasi
foreach mesin $k \in M$ do
    $waktu\_mesin[k] \leftarrow 0$
end
foreach pekerjaan $i \in J$ do
    $waktu\_job[i] \leftarrow 0$
end
$ReadySet \leftarrow$ {operasi pertama dari setiap pekerjaan $i \in J$}
Langkah 2: Iterasi Utama
while $ReadySet \neq \emptyset$ do
    Langkah 3: Pemilihan Operasi dengan aturan SPT
    Pilih $O_{ij} \in ReadySet$ dengan nilai p_{ij} terkecil
    Langkah 4: Penjadwalan Operasi
    $S_{ij} \leftarrow \max(waktu\_job[i], waktu\_mesin[k])$
    $C_{ij} \leftarrow S_{ij} + p_{ij}$
    Langkah 5: Pembaruan Waktu
    $waktu\_job[i] \leftarrow C_{ij}$
    $waktu\_mesin[k] \leftarrow C_{ij}$
    Langkah 6: Pembaruan *Ready Set*
    $ReadySet \leftarrow ReadySet \setminus \{O_{ij}\}$
    if pekerjaan i memiliki operasi berikutnya $O_{i(j+1)}$ then
    $ReadySet \leftarrow ReadySet \cup \{O_{i(j+1)}\}$
    end
end
Langkah 7: Perhitungan *Makespan*
$C_{\max} \leftarrow \max(C_{ij})$
return $C_{\max}$

Berdasarkan *pseudocode* tersebut, proses penjadwalan dengan aturan *Shortest Processing Time* (SPT) dilakukan melalui beberapa tahapan sebagai berikut.

1. Inisialisasi
   a. Menentukan waktu awal untuk setiap mesin, yaitu:
   $$waktu\_mesin_k = 0, \forall k$$
   b. Menentukan waktu awal untuk setiap pekerjaan:
   $$waktu\_job_i = 0, \forall i$$
   c. Membentuk himpunan operasi siap (*Ready set*) yang berisi operasi pertama dari setiap pekerjaan.
2. Pemilihan Operasi (Aturan SPT)

   Dari himpunan *Ready set*, dipilih satu operasi dengan waktu proses terkecil (*Shortest Processing Time*).
3. Penjadwalan Operasi
   a. Untuk setiap operasi terpilih O_{ij}, waktu mulai ditentukan dengan:
   $$S_{ij} = \max (waktu\_job_i, waktu\_mesin_{M_{ij}})$$
   b. Waktu selesai operasi dihitung sebagai:
   $$C_{ij} = S_{ij} + p_{ij}$$
4. Pembaruan Waktu
   a. Memperbarui waktu pekerjaan:
   $$waktu\_job_i = C_{ij}$$
   b. Memperbarui waktu mesin:
   $$waktu\_mesin_{M_{ij}} = C_{ij}$$

5. Pembaruan *Ready set*
    a. Operasi yang telah dijadwalkan dihapus dari *Ready set*.
    b. Jika terdapat operasi berikutnya dalam pekerjaan yang sama, maka operasi tersebut ditambahkan ke dalam *Ready set*.
6. Iterasi

    Langkah 2 hingga 5 diulang hingga seluruh operasi dari semua pekerjaan telah dijadwalkan.
7. Perhitungan *Makespan*

    Nilai *makespan* dihitung sebagai waktu selesai maksimum dari seluruh operasi:

$$C_{max} = \max(C_{ij}) \tag{2.2}$$

2.1.4 *Simulated Annealing*

Algoritma *simulated annealing* merupakan suatu metode optimasi yang terinspirasi dari proses *annealing* pada metalurgi, yaitu prosedur pemanasan dan pendinginan material logam secara bertahap. Mekanisme tersebut diadaptasi dalam konteks komputasional untuk memungkinkan pencarian solusi yang lebih efisien dan mendekati optimal melalui pengendalian perubahan energi secara bertahap (Samana dkk., 2015). Dalam proses pengolahan logam, material dipanaskan hingga mencapai suhu tertentu sehingga partikel-partikel di dalamnya memiliki tingkat energi yang tinggi dan dapat bergerak secara bebas. Selanjutnya, suhu diturunkan secara bertahap dengan tujuan menurunkan energi yang ada secara perlahan. Laju pendinginan yang semakin lambat akan memungkinkan pencapaian tingkat energi yang semakin rendah, hingga partikel-partikel tersebut akhirnya mencapai posisi yang stabil dan optimal dengan energi minimum.

Menurut Chakraborty & Bhowmik (2013) prosedur *simulated annealing* dapat dijelaskan sebagai berikut:

1. Mulai dengan konfigurasi sistem yang diketahui beserta energi awal E.
2. Tetapkan suhu awal T sebagai nilai panas dan buat status pembekuan (*frozen* = *false*).
3. Selama status belum beku (*frozen* = *false*), lakukan:
   a. Lakukan gangguan kecil pada sistem (misalnya menggeser posisi satu partikel).
   b. Hitung energi baru E dan perubahan energi $\Delta E = E' - E$ akibat gangguan tersebut.
   c. Jika $\Delta E < 0$, terima gangguan tersebut sebagai konfigurasi sistem baru.
   d. Jika $\Delta E \geq 0$, terima gangguan tersebut dengan probabilitas

$$P = \exp\left(-\frac{\Delta E}{T}\right) \tag{2.3}$$

   Keterangan:

   P : Probabilitas menerima solusi baru yang lebih buruk

   ΔE : Perubahan nilai fungsi objektif

   T : Parameter kontrol yang menurun secara bertahap

   exp : Fungsi eksponensial

   e. Ulangi sampai sistem mencapai kesetimbangan termal pada suhu T.
4. Jika perubahan energi ΔE masih menurun selama beberapa suhu terakhir, kurangi suhu dengan $T = \alpha \times T$ dan ulangi langkah 3.
5. Jika tidak ada penurunan energi lagi, set $frozen = true$ untuk mengakhiri proses.

6. Kembalikan konfigurasi akhir sebagai solusi energi rendah atau solusi terbaik

Algoritma *Simulated Annealing* memiliki tiga komponen utama yang perlu diperhatikan (Tospornsampan dkk., 2007). Komponen tersebut meliputi:

1. Proses *annealing*

   Proses *annealing* merupakan mekanisme inti dalam algoritma *simulated annealing* yang berfungsi untuk mencegah sistem terjebak pada solusi minimum lokal. Tahapan ini sangat dipengaruhi oleh beberapa parameter berikut:

   a. Temperatur awal

      Temperatur awal ditetapkan cukup tinggi agar ruang pencarian lebih luas dan peluang penerimaan solusi baru meningkat. Penelitian sebelumnya menunjukkan variasi penggunaan nilai temperatur awal, seperti 200 pada penelitian (Chakraborty & Bhowmik, 2013), yang dipilih untuk memaksimalkan probabilitas penerimaan, dan 100 pada penelitian (Utomo & Setiafindari, 2021).

   b. Jumlah iterasi pada setiap temperatur

      Penurunan temperatur dilakukan setelah sejumlah iterasi tertentu tercapai. Jumlah iterasi tersebut dapat ditetapkan secara tetap maupun bervariasi sesuai kebutuhan penelitian.

   c. Parameter penurunan temperatur

      Penurunan temperatur dikendalikan oleh parameter *cooling rate* (a) dengan nilai antara 0 hingga 1. Studi terdahulu menunjukkan bahwa

nilai optimal *cooling rate* berada pada kisaran $0{,}99 \leq a$ (Chakraborty & Bhowmik, 2013).

2. Penyusunan ulang

   Penyusunan ulang atau pembentukan solusi tetangga dilakukan dengan memodifikasi konfigurasi saat ini secara acak. Metode perubahannya dapat berbeda-beda bergantung pada karakteristik permasalahan yang sedang diselesaikan.

3. Penghentian algoritma

   Kriteria penghentian ditetapkan sebelum algoritma dijalankan. Penghentian dapat didasarkan pada tercapainya temperatur minimum tertentu, atau berdasarkan batas waktu eksekusi, di mana algoritma akan berhenti ketika durasi yang telah ditentukan terpenuhi.

2.1.5 *Makespan*

Makespan merupakan total waktu penyelesaian seluruh pekerjaan, dimulai dari operasi pertama yang diproses pada mesin atau *work center* awal hingga operasi terakhir yang diselesaikan pada mesin atau *work center* akhir (Kurniawati, 2019). Menurut Puspitasari (2016) *makespan* merujuk pada total waktu yang dibutuhkan untuk menyelesaikan seluruh rangkaian pekerjaan dalam suatu sistem penjadwalan.

Dalam JSSP, *makespan* menjadi salah satu fungsi objektif yang paling umum digunakan karena merepresentasikan efisiensi keseluruhan jadwal produksi. Semakin kecil nilai *makespan*, maka semakin efisien proses penjadwalan yang dihasilkan karena seluruh pekerjaan dapat diselesaikan dalam waktu yang lebih singkat.

Secara matematis, *makespan* didefinisikan sebagai waktu selesai maksimum dari seluruh operasi, yaitu:

$$C_{max} = \max (C_{ij})$$

dengan:

C_{max} : *Makespan*

C_{ij} : Waktu selesai operasi ke-j dari pekerjaan ke-i

2.1.6 *Gantt Chart*

Gantt chart merupakan diagram berbentuk batang yang digunakan untuk merepresentasikan durasi setiap aktivitas dalam suatu proyek. Dalam *Gantt chart*, ditunjukkan hubungan ketergantungan (*precedence relationship*) antar aktivitas, yaitu suatu aktivitas hanya dapat dimulai setelah aktivitas sebelumnya selesai dilaksanakan (Russel & Taylor, 2011). Secara struktural, *Gantt chart* terdiri atas dua sumbu, yaitu sumbu horizontal (absis) dan sumbu vertikal (ordinat). Sumbu horizontal menggambarkan waktu mulai serta durasi pelaksanaan suatu aktivitas, sedangkan sumbu vertikal menunjukkan daftar aktivitas atau rincian pekerjaan yang harus diselesaikan dalam proyek tersebut (Wignjosoebroto, 2006). Dalam konteks JSSP, *Gantt chart* digunakan untuk memvisualisasikan urutan operasi pada setiap mesin sehingga hubungan antaroperasi dan penggunaan mesin dapat diamati dengan lebih jelas.

2.2 Kajian Integrasi Topik dengan Al-Quran dan Hadits

Secara umum, berbagai konsep dari beragam disiplin ilmu telah dijelaskan dalam Al-Qur'an, termasuk konsep-konsep yang berkaitan dengan matematika. Beberapa prinsip matematika beserta cabang-cabangnya yang termuat dalam Al-Qur'an meliputi logika, pemodelan, statistika, teori graf, dan lainnya. Teori graf,

sebagai salah satu cabang matematika, didefinisikan sebagai himpunan tak kosong yang terdiri atas elemen-elemen yang disebut titik, serta daftar pasangan tak terurut dari titik-titik tersebut yang disebut sisi. Dalam perspektif Islam, titik-titik tersebut dapat dimaknai sebagai Pencipta (Allah) dan hamba-hamba-Nya, sedangkan sisi yang menghubungkan titik-titik tersebut menggambarkan hubungan antara Allah SWT dan makhluk-Nya, serta hubungan antar sesama manusia yang dikenal sebagai *ḥablun min Allāh* dan *ḥablun min al-nās*.

Analoginya, keberadaan dan keterhubungan antar titik menggambarkan adanya relasi atau hubungan yang saling berkaitan. Jika dikaitkan dalam kehidupan sehari-hari, jumlah titik yang saling terhubung dalam suatu graf dapat merepresentasikan jumlah peristiwa tertentu, di mana setiap peristiwa memiliki keterkaitan dengan peristiwa berikutnya. Salah satu bentuk nyata dari aplikasi graf adalah penjadwalan, yang berkaitan erat dengan pengelolaan dan pemanfaatan waktu secara efektif.

Menurut Ibrahim Al-Faqi dalam (Mujahidin dkk., 2022) manajemen waktu diartikan sebagai proses mengatur berbagai pekerjaan ke dalam alokasi waktu yang tersedia, yakni 24 jam per hari, dengan upaya meminimalkan pengorbanan atau biaya. Manajemen waktu juga dipahami sebagai usaha untuk melatih kemampuan menguasai waktu, bukan sebaliknya dikuasai oleh waktu. Dengan demikian, esensi manajemen waktu adalah kemampuan mengelola diri secara optimal dalam berinteraksi dengan waktu guna memperoleh manfaat dan mencapai tujuan. Dalam Al-Qur'an, kata "waktu" diulang tidak kurang dari dua belas kali dalam berbagai bentuk, seperti *waqt*, *mīqāt*, *mawāqīt*, dan *mawqūt*, serta lebih dari seratus kali

dalam bentuk sinonimnya, antara lain *al-'aṣr*, *al-dahr*, *al-lail*, *al-nahār*, dan lainnya (Mujahidin dkk., 2022).

Salah satu indikasi kuat bahwa Islam memberikan perhatian besar terhadap waktu adalah bahwa banyak ibadah ditetapkan berdasarkan batasan waktu yang spesifik. Salat dilaksanakan pada waktu-waktu tertentu yang telah ditetapkan secara rinci; puasa memiliki periode pelaksanaan yang jelas; zakat terkait dengan perhitungan haul; dan haji memiliki bulan-bulan tertentu sebagai waktu pelaksanaannya (Parhan dkk., 2022). Fenomena ini menunjukkan urgensi pengelolaan waktu dalam ajaran Islam. Waktu idealnya dikelola dengan baik melalui perencanaan yang matang, penyusunan aktivitas secara sistematis, penjadwalan yang teratur, serta penetapan target beserta strategi pencapaiannya. Dalam Al-Qur`an dinyatakan:

فَإِذَا فَرَغْتَ فَانْصَبْ ۝٧

(Kementrian Agama Republik Indonesia, 2022)

"Apabila engkau telah selesai (dengan suatu kebajikan), teruslah bekerja keras (untuk kebajikan yang lain)"(QS. Al Insyirah: 7).

Menurut Tafsir Al-Misbah (Shihab, 2002a), ayat ini mengandung anjuran agar manusia tidak berhenti setelah menyelesaikan suatu pekerjaan, tetapi terus melanjutkan kegiatan yang bermanfaat secara berkesinambungan. Waktu harus dimanfaatkan secara produktif karena dalam pandangan Islam, waktu merupakan amanah yang harus dikelola dengan baik. Sementara menurut Tafsir Ibnu Katsir (Mubarakfuri, 2013), makna *"fa idza faraghta fanshab"* menunjukkan bahwa seorang mukmin hendaknya senantiasa beralih dari satu kebaikan ke kebaikan lain, dari satu tugas ke tugas berikutnya tanpa lalai dan tanpa jeda yang sia-sia.

Ayat ini sejalan dengan prinsip JSSP yang mengatur urutan pekerjaan dan waktu pengerjaan secara efisien agar tidak ada waktu yang terbuang (*idle time*). Sama seperti algoritma penjadwalan, ayat ini menekankan pentingnya perencanaan berurutan dan berkesinambungan, sehingga setiap pekerjaan dapat terselesaikan tepat waktu dengan hasil yang maksimal.

Hadist Nabi Muhammad SAW juga mengajarkan pentingnya keefisienan dalam bekerja dan manajemen waktu. Nabi bersabda, “Manfaatkan lima perkara sebelum datang lima perkara: mudamu sebelum datang tuamu, sehatmu sebelum datang sakitmu, kayamu sebelum datang miskinmu, waktumu sebelum datang matimu, dan hidupmu sebelum datang kematianmu” (HR. Al Hakim) (Syukri, 2023). Pesan ini menggarisbawahi pentingnya penggunaan waktu secara optimal dan bijaksana terutama dalam konteks pekerjaan dan pengelolaan sumber daya.

Selai itu, dalam Islam, kerja dan pengelolaan waktu adalah aspek penting yang diatur dalam Al Qur'an. Al Qur'an menekankan pentingnya memanfaatkan waktu secara produktif dan efektif demi mencapai keberhasilan dunia dan akhirat. Allah SWT berfirman dalam Surah Al-Mulk ayat 2:

الَّذِيْ خَلَقَ الْمَوْتَ وَالْحَيٰوةَ لِيَبْلُوَكُمْ اَيُّكُمْ اَحْسَنُ عَمَلًاۗ وَهُوَ الْعَزِيْزُ الْغَفُوْرُۙ ۝٢

(Kementrian Agama Republik Indonesia, 2022)

“yaitu yang menciptakan kematian dan kehidupan untuk menguji kamu, siapa di antara kamu yang lebih baik amalnya. Dia Mahaperkasa lagi Maha Pengampun”(QS. Al Mulk: 2).

Menurut Tafsir As-Sa’di, makna “*li-yabluwakum ayyukum ahsanu ‘amala*” adalah bahwa Allah menguji manusia bukan siapa yang paling banyak amalnya, tetapi siapa yang paling baik amalnya, yakni yang dilakukan dengan ikhlas dan sesuai tuntunan yang benar (Abdurrahman bin Nashir as-Sa’di, 2015). Tafsir Al-

Qurthubi juga menegaskan bahwa ayat ini mengandung perintah agar manusia selalu berusaha menghasilkan amal yang terbaik dan paling sempurna (Al-Qurthubi, 2009).

Konsep *ahsanu 'amala* tersebut sejalan dengan prinsip evaluasi dan pemilihan solusi terbaik dalam bidang optimasi. Dalam suatu permasalahan sering kali terdapat beberapa alternatif solusi yang dapat digunakan untuk mencapai tujuan yang sama, namun masing-masing memiliki tingkat efektivitas yang berbeda. Oleh karena itu, diperlukan proses analisis dan perbandingan untuk menentukan pendekatan yang memberikan hasil terbaik berdasarkan kriteria yang telah ditetapkan.

BAB III

METODE PENELITIAN

3.1 Jenis Penelitian

Jenis penelitian yang digunakan dalam penelitian ini merupakan penelitian kuantitatif dengan pendekatan eksperimental. Penelitian ini bertujuan untuk menganalisis dan mengevaluasi performa algoritma *Simulated Annealing* dalam menyelesaikan *Job Shop Scheduling Problem* (JSSP) menggunakan dua jenis solusi awal, yaitu solusi awal acak dan solusi awal berbasis *Shortest Processing Time* (SPT). Evaluasi performa dilakukan melalui simulasi penjadwalan dan perhitungan nilai *makespan* sebagai indikator utama.

3.2 Data dan Sumber Data

Data yang digunakan dalam penelitian ini berupa dataset *benchmark Job Shop Scheduling Problem* (JSSP), yaitu dataset FT06. Dataset FT06 pasti terdiri atas 6 pekerjaan (*jobs*) dan 6 mesin (*machines*), di mana setiap pekerjaan memiliki urutan operasi yang harus diproses pada mesin tertentu dengan waktu proses yang telah ditentukan. Dataset *benchmark* tersebut diperoleh dari repositori JSPLIB yang tersedia pada GitHub (Tamy0612, n.d.), melalui tautan berikut: https://github.com/tamy0612/JSPLIB/blob/master/instances/ft06.

Data pada dataset FT06 meliputi informasi mengenai urutan mesin yang digunakan pada setiap operasi serta durasi proses masing-masing operasi. Dataset ini banyak digunakan sebagai *benchmark* dalam penelitian *Job Shop Scheduling Problem* karena memiliki kompleksitas yang cukup representatif untuk pengujian algoritma optimasi. Tabel 3.1 menunjukkan dataset FT06 yang digunakan dalam penelitian ini.

Tabel 3.1 Dataset FT06

Job	O1 (M,p)	O2	O3	O4	O5	O6
J1	(2,1)	(0,3)	(1,6)	(3,7)	(5,3)	(4,6)
J2	(1,8)	(2,5)	(4,10)	(5,10)	(0,10)	(3,4)
J3	(2,5)	(3,4)	(5,8)	(0,9)	(1,1)	(4,7)
J4	(1,5)	(0,5)	(2,5)	(3,3)	(4,8)	(5,9)
J5	(2,9)	(1,3)	(4,5)	(5,4)	(0,3)	(3,1)
J6	(1,3)	(3,3)	(5,9)	(0,10)	(4,4)	(2,1)

Keterangan:

$$(M, p)$$

dengan:

M : Mesin yang digunakan

p : Waktu proses operasi

3.3 Tahapan Penelitian

Dalam penelitian ini, tahapan penelitian yang dilakukan adalah sebagai berikut:

1. Mengumpulkan data JSSP dari dataset *benchmark*.
2. Menyusun model matematis JSSP dengan fungsi tujuan untuk meminimalkan nilai *makespan* ($C_{\max}$).
3. Membentuk solusi awal acak untuk *Simulated Annealing* murni.
4. Membentuk solusi awal menggunakan algoritma *greedy*, dengan aturan pemilihan operasi *Shortest Processing Time (SPT)*.
    a. Menentukan waktu awal mesin dan pekerjaan sama dengan nol, serta membentuk *Ready set* dari operasi pertama tiap *job*.

b. Operasi dalam *Ready set* dikelompokkan berdasarkan mesin, kemudian dipilih operasi dengan waktu proses terkecil.

c. Waktu mulai dihitung dengan

$$S_{ij} = \max\left(waktu\_job_i, waktu\_mesin_{M_{ij}}\right)$$

dan waktu selesai

$$C_{ij} = S_{ij} + p_{ij}$$

d. Memperbarui waktu mesin dan pekerjaan, serta menambahkan operasi berikutnya ke *Ready set*.

e. Proses diulang hingga seluruh operasi selesai dijadwalkan.

5. Menghitung nilai *makespan* hasil algoritma *Greedy* sebagai solusi awal (*initial solution*).

6. Menginisialisasi parameter SA, meliputi temperatur awal (T_0), *cooling rate* (α), jumlah iterasi per suhu, dan batas temperatur minimum (T_{min}).

7. Proses iterasi SA:

a. Membuat solusi tetangga dengan menukar dua posisi job (*sequence*).

b. Menghitung perubahan nilai objektif (ΔE = *makespan* baru − *makespan* lama).

c. Jika $\Delta E < 0$, terima solusi baru (lebih baik).

d. Jika $\Delta E \geq 0$, terima dengan probabilitas $e^{-\Delta E/T}$.

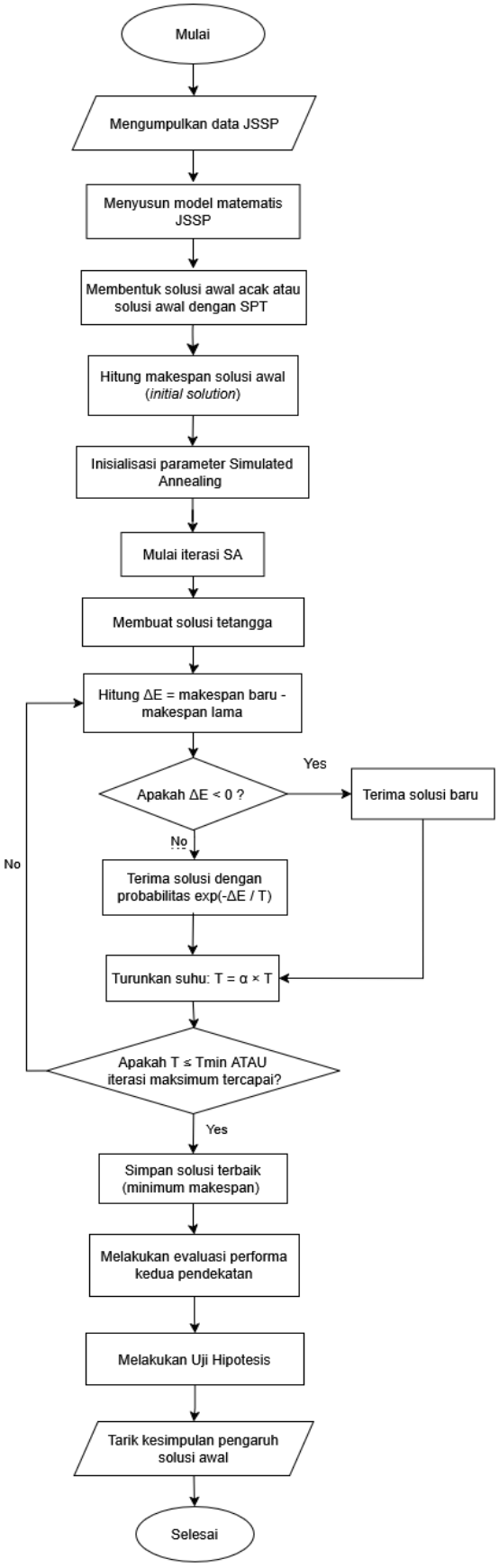
e. Menurunkan suhu $T = \alpha \times T$.

8. Mengulangi proses iterasi hingga temperatur mencapai batas T_{min} atau jumlah iterasi maksimum tercapai.

9. Menyimpan solusi terbaik (*makespan* minimum) yang diperoleh selama proses SA.

10. Melakukan pengujian dengan banyak percobaan untuk masing-masing pendekatan:
    a. *Simulated Annealing* dengan solusi awal acak.
    b. *Simulated Annealing* dengan solusi awal berbasis SPT.
11. Melakukan evaluasi performa kedua pendekatan.
12. Melakukan uji hipotesis menggunakan uji statistik untuk mengetahui apakah terdapat perbedaan performa yang signifikan antara kedua pendekatan.
13. Menarik kesimpulan mengenai pengaruh solusi awal berbasis *Shortest Processing Time* (SPT) terhadap performa algoritma *Simulated Annealing* dalam menyelesaikan *Job Shop Scheduling Problem*.

Flowchart dari tahapan penelitian yang dilakukan tersebut, dapat dilihat pada Gambar 3.1.



Gambar 3.1 *Flowchart* Tahapan Penelitian

BAB IV

HASIL DAN PEMBAHASAN

4.1 Deskripsi Dataset Penelitian

Data yang digunakan dalam penelitian ini adalah dataset *benchmark Job Shop Scheduling Problem* (JSSP) yaitu FT06, yang merupakan salah satu dataset standar yang sering digunakan dalam pengujian algoritma optimasi penjadwalan. Dataset FT06 terdiri atas 6 pekerjaan (*jobs*), yaitu $J1$ hingga $J6$, dan 6 mesin (*machines*), yaitu $M0$ hingga $M5$. Setiap pekerjaan memiliki 6 operasi yang harus diproses secara berurutan pada mesin tertentu dengan waktu proses yang telah ditentukan. Setiap operasi dinyatakan dalam pasangan (M, p) dengan:

M : Mesin yang digunakan

p : Waktu proses (*processing time*)

Berdasarkan Tabel 3.1, setiap pekerjaan memiliki urutan operasi yang harus diproses secara berurutan pada mesin tertentu. Sebagai contoh, pekerjaan $J1$ memiliki urutan operasi:

Operasi 1 → Mesin 2 selama 1 waktu

Operasi 2 → Mesin 0 selama 3 waktu

Operasi 3 → Mesin 1 selama 6 waktu

dan seterusnya.

Hal tersebut menunjukkan bahwa setiap operasi pada suatu pekerjaan harus diproses secara berurutan sesuai urutan yang telah ditentukan. Selain itu, setiap mesin hanya dapat memproses satu operasi pada satu waktu sehingga dapat terjadi antrean penggunaan mesin antar pekerjaan.

4.2 Simulasi Perhitungan Algoritma

Pada bagian ini dilakukan simulasi penyelesaian JSSP secara manual menggunakan algoritma SA dengan solusi awal berbasis SPT. Simulasi perhitungan ini bertujuan untuk memberikan gambaran mengenai mekanisme kerja algoritma sebelum diimplementasikan dalam bentuk program. Pada penelitian ini, aturan SPT digunakan sebagai pembentuk solusi awal heuristik, sedangkan SA digunakan sebagai metode optimasi utama untuk mengeksplorasi ruang solusi dan mencari nilai *makespan* minimum.

Mengingat kompleksitas dataset FT06 yang cukup besar, maka pada simulasi manual digunakan sebagian data, yaitu 3 pekerjaan (*job*) dan 3 mesin sebagai ilustrasi proses perhitungan. Data yang digunakan dalam simulasi perhitungan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Subdata FT06 3x3

Job	Operasi	Mesin	Waktu
J1	O1	M2	1
J1	O2	M0	3
J1	O3	M1	6
J2	O1	M1	8
J2	O2	M2	5
J2	O3	M0	10
J3	O1	M2	5
J3	O2	M0	9
J3	O3	M1	1

Berdasarkan data pada Tabel 4.1, setiap pekerjaan memiliki urutan operasi yang harus diproses secara berurutan pada mesin tertentu. Sebagai contoh, pekerjaan $J1$ harus diproses pada mesin $M2$, kemudian $M0$, dan terakhir $M1$. Selain itu, setiap mesin hanya dapat memproses satu operasi pada satu waktu sehingga

proses penjadwalan harus memperhatikan urutan operasi pekerjaan dan ketersediaan mesin secara bersamaan.

Tujuan dari simulasi ini adalah memperoleh nilai *makespan* minimum, yaitu total waktu penyelesaian seluruh pekerjaan. Dalam permasalahan JSSP, *makespan* didefinisikan sebagai waktu selesai terbesar dari seluruh operasi yang dijadwalkan. Oleh karena itu, nilai *makespan* diperoleh dengan mencari waktu selesai maksimum (max) dari seluruh operasi. Secara matematis, fungsi objektif yang digunakan dalam penelitian ini dinyatakan sebagai berikut:

$$C_{max} = \max (C_{ij})$$

dengan:

C_{max} : *Makespan*

C_{ij} : Waktu selesai operasi ke-j dari pekerjaan ke-i

4.2.1 Pembentukan Solusi Awal dengan *Shortest Processing Time*

Proses pembentukan solusi awal dilakukan menggunakan algoritma *Greedy* dengan aturan *Shortest Processing Time* (SPT). Pada aturan ini, operasi dengan waktu proses paling singkat diprioritaskan untuk dijadwalkan terlebih dahulu. Proses diawali dengan inisialisasi waktu setiap pekerjaan (*job*) dan mesin sebesar nol. Selanjutnya dibentuk himpunan *ready set* yang berisi operasi pertama dari setiap pekerjaan. Berdasarkan dataset Tabel 4.1, operasi awal yang tersedia adalah $J1$ pada mesin $M2$ dengan waktu proses 1, $J2$ pada mesin $M1$ dengan waktu proses 8, $J3$ pada mesin $M2$ dengan waktu proses 5.

Iterasi 1

1. Pembentukan *Ready set*

$$R = \{(J1, M2,1), (J2, M1,8), (J3, M2,5)\}$$

2. Pemilihan operasi (SPT)

$$\min\{1,8,5\} = 1 \Rightarrow J1$$

3. Perhitungan waktu

$$S_{11} = \max(0,0) = 0$$

$$C_{11} = 0 + 1 = 1$$

4. Pembaruan

   Waktu *job* $J1 = 1$

   Waktu mesin $M2 = 1$

5. Update *ready set*

   Hapus $J1 \rightarrow (M2,1)$

   Tambahkan $J1 \rightarrow (M0,3)$

Iterasi 2

1. *Ready set*

$$R = \{(J1, M0,3), (J2, M1,8), (J3, M2,5)\}$$

2. Pemilihan operasi

$$\min\{3,8,5\} = 3 \Rightarrow J1$$

4. Perhitungan waktu

$$S_{12} = \max(1,0) = 1$$

$$C_{12} = 1 + 3 = 4$$

5. Pembaruan

   Waktu *job* $J1 = 4$

   Waktu mesin $M0 = 4$

6. Update *ready set*

   Hapus $J1 \rightarrow (M0,3)$

Tambahkan $J1 \rightarrow (M1,6)$

Iterasi 3

1. *Ready set*

$$R = \{(J1, M1,6), (J2, M1,8), (J3, M2,5)\}$$

2. Pemilihan operasi

$$\min \{6,8,5\} = 5 \Rightarrow J3$$

3. Perhitungan waktu

$$S_{31} = \max (0,1) = 1$$

$$C_{31} = 1 + 5 = 6$$

4. Pembaruan

Waktu *job* $J3 = 6$

Waktu mesin $M2 = 6$

5. Update *ready set*

Hapus $J3 \rightarrow (M2,5)$

Tambahkan $J3 \rightarrow (M0,9)$

Iterasi 4

1. *Ready set*

$$R = \{(J1, M1,6), (J2, M1,8), (J3, M0,9)\}$$

2. Pemilihan operasi

$$\min \{6,8,9\} = 6 \Rightarrow J1$$

3. Perhitungan waktu

$$S_{13} = \max (4,0) = 4$$

$$C_{13} = 4 + 6 = 10$$

4. Pembaruan

   Waktu *job* $J1 = 10$

   Waktu mesin $M1 = 10$

5. Update *ready set*

   Hapus $J1 \rightarrow (M1,6)$

Iterasi 5

1. *Ready set*

$$R = \{(J2, M1,8), (J3, M0,9)\}$$

2. Pemilihan operasi

$$\min\{8,9\} = 8 \Rightarrow J2$$

3. Perhitungan waktu

$$S_{21} = \max(0,10) = 10$$

$$C_{21} = 10 + 8 = 18$$

4. Pembaruan

   Waktu job $J2 = 18$

   Waktu mesin $M1 = 18$

5. Update *ready set*

   Hapus $J2 \rightarrow (M1,8)$

   Tambahkan $J2 \rightarrow (M2,5)$

Iterasi 6

1. *Ready set*

$$R = \{(J2, M2,5), (J3, M0,9)\}$$

2. Pemilihan operasi

$$\min\{5,9\} = 5 \Rightarrow J2$$

3. Perhitungan waktu

$$S_{22} = \max(18,6) = 18$$

$$C_{22} = 18 + 5 = 23$$

4. Pembaruan

   Waktu *job* $J2 = 23$

   Waktu mesin $M2 = 23$

5. Update *ready set*

   Hapus $J2 \rightarrow (M2,5)$

   Tambahkan $J2 \rightarrow (M0,10)$

Iterasi 7

1. *Ready set*

$$R = \{(J2, M0,10), (J3, M0,9)\}$$

2. Pemilihan operasi

$$\min\{10,9\} = 9 \Rightarrow J3$$

3. Perhitungan waktu

$$S_{32} = \max(6,4) = 6$$

$$C_{32} = 6 + 9 = 15$$

4. Pembaruan

   Waktu *job* $J3 = 15$

   Waktu mesin $M0 = 15$

5. Update *ready set*

   Hapus $J3 \rightarrow (M0,9)$

   Tambahkan $J3 \rightarrow (M1,1)$

Iterasi 8

1. *Ready set*

$$R = \{(J2, M0,10), (J3, M1,1)\}$$

2. Pemilihan operasi

$$\min\{10,1\} = 1 \Rightarrow J3$$

3. Perhitungan waktu

$$S_{33} = \max(15,18) = 18$$

$$C_{33} = 18 + 1 = 19$$

4. Pembaruan

   Waktu *job* $J3 = 19$

   Waktu mesin $M1 = 19$

5. Update *ready set*

   Hapus $J3 \rightarrow (M1,19)$

Iterasi 9

1. *Ready set*

$$R = \{(J2, M0,10)\}$$

2. Pemilihan operasi

$$10 \Rightarrow J2$$

3. Perhitungan waktu

$$S_{23} = \max(23,15) = 23$$

$$C_{23} = 23 + 10 = 33$$

4. Pembaruan

   Waktu *job* $J2 = 33$

   Waktu mesin $M0 = 33$

Berdasarkan hasil perhitungan tersebut, didapatkan jadwal solusi awal dengan Algoritma *Greedy* pada Tabel 4.2.

Tabel 4.2 Jadwal Solusi Awal Dataset 3x3

Job	Operasi	Mesin	Start	End
J1	O1	M2	0	1
J1	O2	M0	1	4
J3	O1	M2	1	6
J1	O3	M1	4	10
J2	O1	M1	10	18
J2	O2	M2	18	23
J3	O2	M0	6	15
J3	O3	M1	18	19
J2	O3	M0	23	33

Sehingga diperoleh nilai makespan awal sebesar:

$$C_{max}^{Greedy} = 33$$

4.2.2 Inisiasi Parameter *Simulated Annealing*

Pada penelitian ini, algoritma *Simulated Annealing* diimplementasikan dengan beberapa parameter yang telah ditentukan sebelumnya. Parameter-parameter tersebut meliputi temperatur awal (T_0), *cooling rate* (α), jumlah iterasi pada setiap temperatur, temperatur minimum (T_{min}), serta jumlah siklus pendinginan (*max iteration*). Penentuan parameter ini bertujuan untuk mengatur proses eksplorasi solusi selama algoritma berjalan. Nilai parameter yang digunakan dalam penelitian ini disajikan pada Tabel 4.3.

Tabel 4.3 Parameter *Simulated Annealing*

Parameter	Nilai
T_0 (Temperatur awal)	100
α (Cooling rate)	0,95
Iterasi per suhu	10
T_min (Temperatur minimum)	0,1
max_iter (Jumlah siklus pendinginan)	100

Temperatur awal digunakan untuk menentukan tingkat probabilitas penerimaan solusi baru pada tahap awal pencarian. Nilai temperatur awal yang cukup tinggi memungkinkan algoritma menerima solusi yang lebih buruk sehingga proses eksplorasi ruang solusi menjadi lebih luas dan algoritma tidak mudah terjebak pada *local optimum*. Pada penelitian ini digunakan temperatur awal sebesar 100.

Parameter *cooling rate* (α) digunakan untuk mengatur laju penurunan temperatur pada setiap siklus iterasi. Nilai α yang mendekati 1 menyebabkan proses pendinginan berlangsung lebih lambat sehingga pencarian solusi menjadi lebih stabil. Pada penelitian ini digunakan nilai *cooling rate* sebesar 0,95.

Jumlah iterasi pada setiap temperatur menunjukkan banyaknya proses pembentukan solusi tetangga yang dilakukan sebelum temperatur diturunkan. Sementara itu, temperatur minimum (T_{min})digunakan sebagai salah satu kondisi penghentian algoritma. Proses iterasi akan dihentikan apabila temperatur telah mencapai batas minimum atau jumlah siklus pendinginan maksimum telah terpenuhi.

4.2.3 Proses *Simulated Annealing*

Setelah diperoleh solusi awal berbasis *Shortest Processing Time* (SPT), langkah selanjutnya adalah melakukan perhitungan dengan algoritma *Simulated Annealing*. Berdasarkan hasil perhitungan sebelumnya, diperoleh nilai *makespan* awal sebesar $C_{max} = 33$.

Nilai tersebut digunakan sebagai solusi awal (*initial solution*) dalam proses *Simulated Annealing* dengan parameter yang telah ditentukan pada Tabel 4.3. Pada bagian ini hanya ditunjukkan satu iterasi sebagai ilustrasi mekanisme algoritma *Simulated Annealing*. Pada implementasi program, proses iterasi dilakukan secara berulang hingga memenuhi kondisi penghentian algoritma.

Langkah-langkah pada iterasi pertama sebagai berikut:

1. Solusi awal

   Solusi awal dinyatakan sebagai urutan pekerjaan (*sequence*):

   $$S = [J1, J1, J3, J1, J2, J2, J3, J3, J2]$$

   Urutan tersebut menunjukkan urutan operasi yang diproses berdasarkan representasi solusi pada JSSP. Berdasarkan hasil perhitungan sebelumnya, diperoleh:

   $$C_{\max}(S) = 33$$

2. Pembentukan solusi tetangga

   Solusi tetangga dibentuk menggunakan operator *insertion* (*shift*), yaitu memindahkan satu elemen ke posisi lain dalam urutan solusi. Misalkan operasi milik $J2$ pada posisi ke-5 dipindahkan ke posisi ke-2, sehingga diperoleh solusi baru:

   $$S' = [J1, J2, J1, J3, J1, J2, J3, J3, J2]$$

Proses ini bertujuan untuk mengeksplorasi kemungkinan solusi lain yang mungkin menghasilkan nilai *makespan* yang lebih baik. Pergeseran dilakukan tanpa mengubah jumlah operasi masing-masing pekerjaan, sehingga *precedence constraint* pada setiap *job* tetap terpenuhi.

3. Evaluasi solusi baru

Untuk solusi baru S', dilakukan perhitungan ulang waktu mulai (*start time*) dan waktu selesai (*completion time*) setiap operasi berdasarkan aturan:

$$S_{ij} = \max \text{ (waktu } job \text{ tersedia,waktu mesin tersedia)}$$

Nilai waktu job diperoleh dari waktu selesai operasi terakhir pada pekerjaan yang sama, sedangkan waktu mesin diperoleh dari waktu selesai operasi terakhir yang menggunakan mesin tersebut. Inisiasi semua mesin dan *job* mulai dari 0:

Mesin:

$$M0 = 0,\ M1 = 0,\ M2 = 0$$

Job:

$$J1 = 0,\ J2 = 0,\ J3 = 0$$

a. $J1 \rightarrow O1\ (M2,1)$

$$S = \max (0,0) = 0$$

$$C = 0 + 1 = 1$$

Update:

$$M2 = 1$$

$$J1 = 1$$

b. $J2 \rightarrow O1\ (M1,8)$

$$S = \max (0,0) = 0$$

$$C = 0 + 8 = 8$$

Update:

$$M1 = 8$$

$$J2 = 8$$

c. $J1 \rightarrow O2$ $(M0,3)$

$$S = \max(1,0) = 1$$

$$C = 1 + 3 = 4$$

Update:

$$M0 = 4$$

$$J1 = 4$$

d. $J3 \rightarrow O1$ $(M2,5)$

$$S = \max(0,1) = 1$$

$$C = 1 + 5 = 6$$

Update:

$$M2 = 6$$

$$J3 = 6$$

e. $J1 \rightarrow O3$ $(M1,6)$

$$S = \max(4,8) = 8$$

$$C = 8 + 6 = 14$$

Update:

$$M1 = 14$$

$$J1 = 14$$

f. $J2 \rightarrow O2$ $(M2,5)$

$$S = \max(8,6) = 8$$

$$C = 8 + 5 = 13$$

Update:

$$M2 = 13$$

$$J2 = 13$$

g. $J3 \rightarrow O2$ $(M0,9)$

$$S = \max(6,4) = 6$$

$$C = 6 + 9 = 15$$

Update:

$$M0 = 15$$

$$J3 = 15$$

h. $J3 \rightarrow O3$ $(M1,1)$

$$S = \max(15,14) = 15$$

$$C = 15 + 1 = 16$$

Update:

$$M1 = 16$$

$$J3 = 16$$

i. $J2 \rightarrow O3$ $(M0,10)$

$$S = \max(13,15) = 15$$

$$C = 15 + 10 = 25$$

Update:

$$M0 = 25$$

$$J2 = 25$$

Hasil perhitungan jadwal ditunjukkan pada Tabel 4.4.

Tabel 4.4 Jadwal Solusi Baru Dataset 3x3

Job	Operasi	Mesin	Start	End
J1	O1	M2	0	1
J2	O1	M1	0	8
J1	O2	M0	1	4
J3	O1	M2	1	6
J1	O3	M1	8	14
J2	O2	M2	8	13
J3	O2	M0	6	15
J3	O3	M1	15	16
J2	O3	M0	15	25

Dari Tabel 4.4 tersebut diperoleh nilai *makespan* baru:

$$C_{\max}(S') = 25$$

2. Perhitungan perubahan energi

   Perubahan nilai objektif dihitung menggunakan:

$$\Delta E = C_{\max}(S') - C_{\max}(S)$$

$$\Delta E = 25 - 33 = -8$$

   Nilai ΔE menunjukkan bahwa solusi baru memiliki nilai *makespan* yang lebih kecil dibandingkan solusi sebelumnya

3. Kriteria penerimaan

   Berdasarkan aturan dalam algoritma *Simulated Annealing* keputusan penerimaan solusi dilakukan berdasarkan nilai ΔE:

   Jika $\Delta E < 0$, maka solusi baru lebih baik dan langsung diterima.

   Jika $\Delta E \geq 0$, maka solusi diterima dengan probabilitas $P = e^{-\Delta E/T}$.

   Karena:

$$\Delta E = -8 < 0$$

   maka solusi baru S' diterima dan menggantikan solusi sebelumnya.

4. Update solusi

   Nilai solusi saat ini diperbarui menjadi:

$$C_{current} = 25$$

5. Penurunan temperatur

   Temperatur diperbarui menggunakan:

$$T = \alpha \times T = 0{,}95 \times 100 = 95$$

   Penurunan temperatur dilakukan secara bertahap agar probabilitas penerimaan solusi yang lebih buruk semakin kecil seiring bertambahnya iterasi.

Proses *Simulated Annealing* selanjutnya dilakukan secara berulang dengan tahapan yang sama, yaitu pembangkitan solusi tetangga, evaluasi solusi, perhitungan perubahan energi, dan penentuan penerimaan solusi, hingga memenuhi kondisi penghentian (*stopping criteria*). Algoritma akan berhenti apabila salah satu dari kondisi berikut terpenuhi:

1. Temperatur telah mencapai batas minimum ($T \leq T_{min}$), sehingga probabilitas penerimaan solusi yang lebih buruk menjadi sangat kecil.
2. Jumlah siklus pendinginan telah mencapai batas maksimum (max_iter).

4.3 Implementasi Algoritma

4.3.1 Implementasi *Simulated Annealing* dengan Solusi Awal Acak

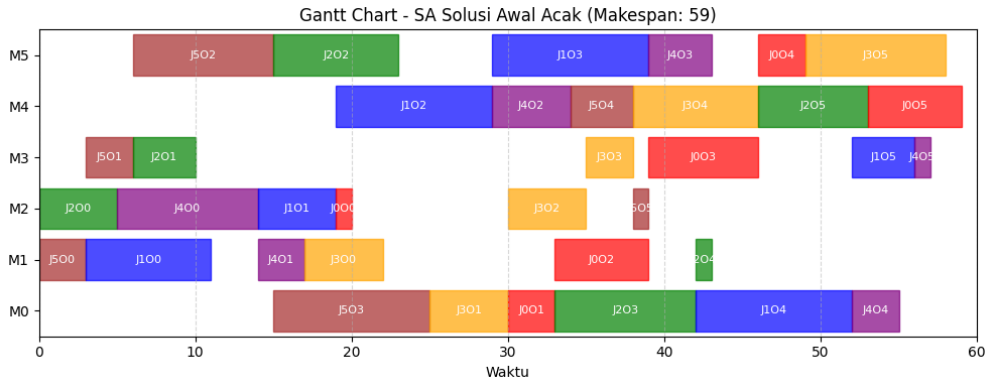
Pada dataset FT06, implementasi algoritma *Simulated Annealing* dilakukan menggunakan bahasa pemrograman *Python* untuk keseluruhan data. Pada pendekatan ini, solusi awal dibangkitkan secara acak sebelum dilakukan proses optimasi menggunakan *Simulated Annealing*. Berdasarkan hasil satu kali percobaan, diperoleh nilai makespan awal sebesar 73 dan nilai makespan terbaik

sebesar 58. Hasil tersebut menunjukkan bahwa algoritma *Simulated Annealing* mampu menurunkan nilai *makespan* secara signifikan melalui proses iterasi yang melibatkan pembangkitan solusi tetangga dan mekanisme penerimaan solusi.

Adapun urutan solusi terbaik yang diperoleh adalah

$$s = [J5, J2, J5, J1, J4, J2, J5, J4, J1, J3, J5, J3, J1, J0, J0, J2, J3, J3, J1,$$
$$J4, J5, J2, J0, J4, J1, J2, J0, J4, J3, J5, J0, J1, J4, J2, J3, J0]$$

Berdasarkan solusi tersebut, dibuat visualisasi dalam bentuk *Gantt chart* pada Gambar 4.1 yang menunjukkan bahwa setiap mesin hanya memproses satu operasi pada satu waktu, tidak terdapat konflik antar operasi, serta urutan operasi pada setiap *job* tetap terpenuhi. Nilai *makespan* yang dihasilkan dari solusi tersebut adalah sebesar $C_{\max} = 59$.



Gambar 4.1 *Gantt Chart Simulated Annealing* Solusi Awal Acak

Berdasarkan satu kali percobaan, algoritma *Simulated Annealing* mampu menurunkan nilai *makespan* dari 73 menjadi 59. Hal ini menunjukkan bahwa mekanisme eksplorasi solusi pada *Simulated Annealing* berjalan dengan baik dalam memperbaiki kualitas solusi penjadwalan. Namun demikian, karena solusi awal dibangkitkan secara acak, hasil yang diperoleh pada satu kali percobaan belum dapat digunakan untuk menarik kesimpulan umum mengenai performa

algoritma. Oleh karena itu, diperlukan pengujian berulang untuk mengevaluasi kestabilan hasil dan performa algoritma secara lebih menyeluruh.

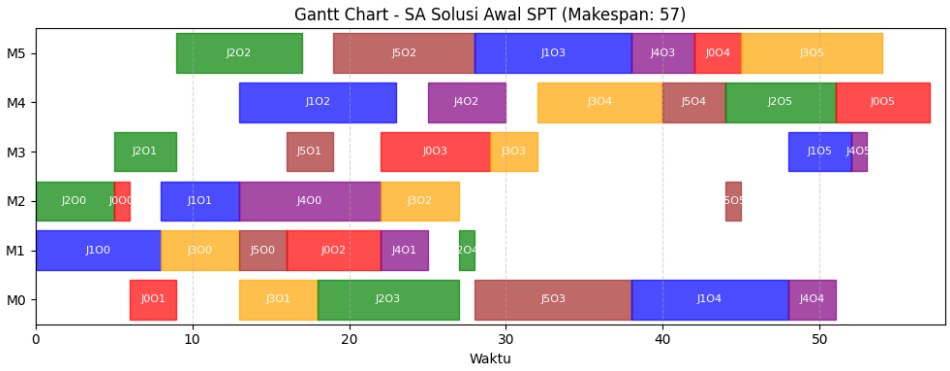
4.3.2 Implementasi *Simulated Annealing* dengan Solusi Awal Berbasis SPT

Pada dataset FT06, implementasi *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT) dilakukan menggunakan bahasa pemrograman *Python* untuk keseluruhan data. Pada pendekatan ini, solusi awal dibentuk terlebih dahulu menggunakan aturan *Shortest Processing Time* (SPT), kemudian dilakukan proses algoritma *Simulated Annealing*. Berdasarkan hasil satu kali percobaan, diperoleh nilai makespan awal sebesar 88 dan nilai makespan terbaik sebesar 57. Hasil tersebut menunjukkan bahwa algoritma *Simulated Annealing* mampu memperbaiki solusi awal yang dibentuk menggunakan aturan SPT melalui proses iterasi dan eksplorasi solusi.

Adapun urutan solusi terbaik (*best sequence*) yang diperoleh adalah

$$s = [J2, J1, J2, J0, J3, J1, J2, J4, J0, J5, J0, J5, J1, J3, J5, J0, J4, J3, J1,$$
$$J4, J4, J3, J3, J2, J2, J5, J1, J0, J1, J4, J5, J3, J2, J0, J4, J5]$$

Berdasarkan solusi tersebut, dibuat visualisasi dalam bentuk *Gantt chart* pada Gambar 4.2 yang menunjukkan bahwa jadwal yang dihasilkan telah memenuhi seluruh *constraint* pada permasalahan *Job Shop Scheduling Problem*. Setiap mesin hanya memproses satu operasi pada satu waktu dan urutan operasi pada setiap pekerjaan tetap terpenuhi. Nilai makespan yang dihasilkan dari solusi ini adalah sebesar $C_{\max} = 57$.



Gambar 4.2 *Gantt Chart Simulated Annealing* Solusi Awal dengan SPT

Berdasarkan satu kali percobaan, pendekatan *Simulated Annealing* dengan solusi awal berbasis SPT menghasilkan nilai *makespan* yang lebih kecil dibandingkan *Simulated Annealing* dengan solusi awal acak. Hal ini menunjukkan bahwa solusi awal berbasis heuristik dapat membantu proses pencarian solusi pada *Simulated Annealing*. Namun demikian, hasil ini masih bersifat deskriptif dan belum dapat digunakan untuk menarik kesimpulan umum mengenai pengaruh solusi awal terhadap performa algoritma. Oleh karena itu, diperlukan pengujian berulang dan analisis statistik untuk mengevaluasi kestabilan hasil serta mengetahui apakah perbedaan performa kedua pendekatan signifikan secara statistik.

4.4 Evaluasi Performa Algoritma

4.4.1 Hasil Pengujian 30 Percobaan

Pengujian algoritma *Simulated Annealing* (SA) dengan solusi awal acak dan *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT) dilakukan sebanyak 30 kali percobaan pada dataset FT06. Pengujian ini bertujuan untuk mengevaluasi performa dan konsistensi kedua pendekatan dalam menghasilkan nilai *makespan*. Karena algoritma *Simulated Annealing* bersifat stokastik, hasil yang diperoleh pada setiap percobaan dapat berbeda meskipun

menggunakan parameter yang sama. Oleh karena itu, pengujian berulang diperlukan untuk memperoleh gambaran performa algoritma secara lebih objektif.

Pada setiap percobaan dicatat nilai *makespan* awal (*initial makespan*) dan nilai *makespan* terbaik (*best makespan*) yang diperoleh setelah proses optimasi menggunakan *Simulated Annealing*. Hasil pengujian untuk masing-masing percobaan dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Pengujian Nilai *Makespan* 30 Percobaan

Run	Makespan Awal SA Acak	Makespan Terbaik SA Acak	Makespan Awal SA SPT	Makespan Terbaik SA SPT
1	104	57	88	55
2	100	59	88	57
3	75	59	88	59
4	94	58	88	55
5	85	55	88	58
6	106	59	88	58
7	92	55	88	59
8	68	55	88	57
9	76	59	88	57
10	78	57	88	58
11	92	59	88	59
12	84	59	88	57
13	73	59	88	55
14	74	58	88	55
15	84	55	88	59
16	82	58	88	58
17	72	59	88	55
18	98	58	88	58
19	81	59	88	59
20	86	59	88	58
21	85	55	88	55
22	71	59	88	55
23	105	59	88	59
24	102	58	88	55
25	92	55	88	58
26	88	55	88	57
27	86	59	88	57
28	92	59	88	59
29	99	59	88	59
30	88	55	88	59

Berdasarkan Tabel 4.5, terlihat bahwa metode *Simulated Annealing* murni menghasilkan nilai *makespan* awal yang bervariasi karena solusi awal dibangkitkan secara acak. Nilai *makespan* awal yang diperoleh berada pada rentang 68 hingga 104. Setelah dilakukan proses optimasi menggunakan *Simulated Annealing*, nilai *makespan* terbaik yang dihasilkan berada pada rentang 55 hingga 59.

Sementara itu, pada pendekatan *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT), nilai *makespan* awal selalu bernilai 88 karena solusi awal dibentuk dengan cara mengulang aturan heuristik yang sama pada setiap percobaan. Setelah dilakukan proses optimasi menggunakan *Simulated Annealing*, nilai *makespan* terbaik yang diperoleh berada pada rentang 55 hingga 59.

Secara umum, kedua pendekatan mampu menghasilkan solusi dengan kualitas yang relatif serupa. Hal tersebut menunjukkan bahwa algoritma *Simulated Annealing* memiliki kemampuan eksplorasi solusi yang cukup baik, baik menggunakan solusi awal acak maupun solusi awal berbasis heuristik SPT. Namun demikian, untuk mengetahui perbedaan performa kedua pendekatan secara lebih objektif, diperlukan analisis statistik dan evaluasi performa lebih lanjut.

4.4.2 Ringkasan Statistik Hasil Pengujian

Untuk mempermudah analisis, ringkasan statistik dari hasil pengujian disajikan pada Tabel 4.6.

Tabel 4.6 Ringkasan Statistik Hasil Pengujian

Metode	Rata-rata Makespan	Standar Deviasi
SA Solusi Awal Acak	57,63	1,68
SA Solusi Awal SPT	57,30	1,55

Berdasarkan Tabel 4.6, metode *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT) menghasilkan rata-rata *makespan* sebesar 57,30, sedikit lebih kecil dibandingkan metode *Simulated Annealing* murni yang menghasilkan rata-rata *makespan* sebesar 57,63. Hal ini menunjukkan bahwa penggunaan solusi awal berbasis *Shortest Processing Time* (SPT) mampu membantu proses pencarian solusi sehingga menghasilkan kualitas solusi yang sedikit lebih baik secara rata-rata.

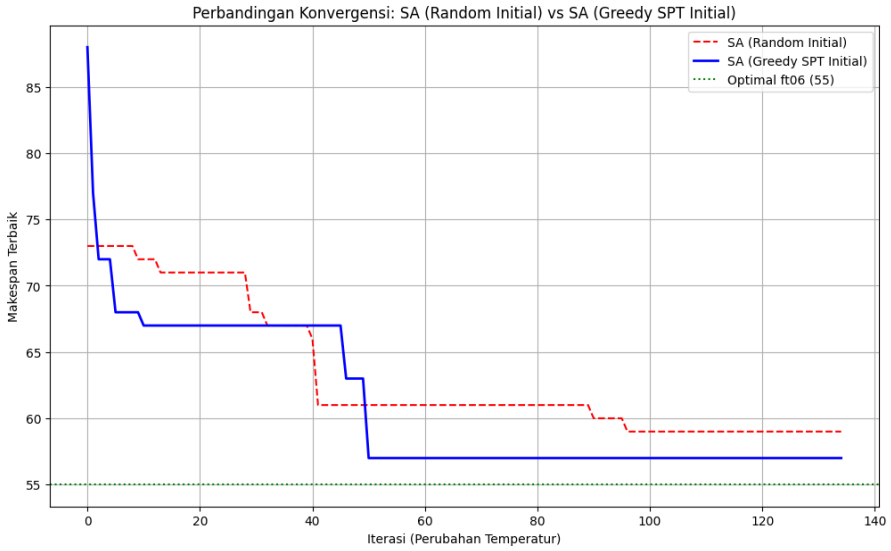
Selain itu, metode *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT) juga memiliki standar deviasi yang lebih kecil dibandingkan *Simulated Annealing* dengan solusi awal acak, yaitu sebesar 1,55 dibandingkan 1,68. Nilai standar deviasi yang lebih kecil menunjukkan bahwa hasil yang diperoleh metode *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT) cenderung lebih stabil dan konsisten pada berbagai percobaan.

Meskipun demikian, selisih rata-rata *makespan* dan standar deviasi antara kedua metode relatif kecil sehingga diperlukan analisis statistik lebih lanjut untuk mengetahui apakah perbedaan tersebut signifikan secara statistik.

4.4.3 Analisis Konvergensi

Untuk melihat perkembangan kualitas solusi selama proses iterasi, dilakukan analisis konvergensi terhadap algoritma *Simulated Annealing* dengan solusi awal acak dan *Simulated Annealing* dengan solusi awal berbasis *Shortest*

Processing Time (SPT). Grafik konvergensi menunjukkan perubahan nilai *makespan* terbaik pada setiap iterasi penurunan temperatur.



Gambar 4.3 Grafik Perbandingan Konvergensi

Berdasarkan Gambar 4.3, kedua pendekatan mengalami penurunan nilai *makespan* secara bertahap seiring bertambahnya iterasi. Pada tahap awal iterasi, pendekatan dengan solusi awal berbasis SPT memiliki nilai *makespan* awal yang lebih besar dibandingkan solusi awal acak. Namun demikian, penurunan nilai *makespan* pada pendekatan berbasis SPT berlangsung lebih cepat hingga mencapai nilai 57 pada iterasi yang relatif lebih sedikit, yaitu sekitar iterasi ke 50.

Sementara itu, pendekatan *Simulated Annealing* dengan solusi awal acak menunjukkan penurunan *makespan* yang lebih lambat, dan mencapai nilai 59 pada iterasi ke 95 dan cenderung stabil pada nilai 59 hingga akhir iterasi. Hal ini menunjukkan bahwa solusi awal berbasis heuristik dapat membantu proses pencarian solusi menjadi lebih terarah pada tahap awal optimasi.

Selain itu, terlihat bahwa kedua pendekatan belum mencapai nilai optimal FT06 sebesar 55 pada percobaan tersebut, meskipun keduanya mampu menghasilkan solusi yang mendekati optimum. Hasil ini menunjukkan bahwa

algoritma *Simulated Annealing* memiliki kemampuan eksplorasi solusi yang baik, baik menggunakan solusi awal acak maupun solusi awal berbasis heuristik SPT.

4.4.4 Frekuensi Pencapaian Solusi Terbaik

Selain menggunakan rata-rata *makespan*, evaluasi performa algoritma juga dilakukan berdasarkan frekuensi pencapaian solusi terbaik, yaitu nilai *makespan* 55 yang merupakan solusi optimal pada dataset FT06. Hasil pengujian menunjukkan bahwa metode *Simulated Annealing* dengan solusi awal acak dan metode *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT) sama-sama berhasil mencapai nilai *makespan* 55 sebanyak 8 kali dari 30 percobaan. Persentase pencapaian solusi optimal pada kedua metode juga sama, yaitu sebesar 26,67%.

Tabel 4.7 Frekuensi Pencapaian Makespan 55

Metode	Frekuensi	Persentase
SA Solusi Awal Acak	8	26,67%
SA Solusi Awal SPT	8	26,67%

Hasil tersebut menunjukkan bahwa penggunaan solusi awal berbasis heuristik SPT belum memberikan peningkatan kemampuan yang signifikan dalam mencapai solusi optimal dibandingkan solusi awal acak. Kedua metode memiliki tingkat keberhasilan yang relatif sama dalam menemukan nilai *makespan* terbaik pada dataset FT06.

Selain itu, hasil ini juga memperlihatkan bahwa algoritma *Simulated Annealing* memiliki kemampuan eksplorasi solusi yang cukup baik sehingga tetap mampu menemukan solusi optimal meskipun menggunakan solusi awal yang dibangkitkan secara acak. Dengan demikian, kualitas solusi akhir yang dihasilkan

lebih dipengaruhi oleh proses eksplorasi pada algoritma *Simulated Annealing* dibandingkan jenis solusi awal yang digunakan.

4.5 Analisis Uji Statistik

Analisis uji statistik dilakukan untuk mengetahui apakah terdapat perbedaan performa yang signifikan antara metode *Simulated Annealing* (SA) dengan solusi awal acak dan metode *Simulated Annealing* (SA) dengan solusi awal berbasis SPT berdasarkan hasil *makespan* yang diperoleh dari 30 kali percobaan.Sebelum dilakukan uji hipotesis, terlebih dahulu dilakukan uji normalitas menggunakan metode *Shapiro-Wilk*. Uji ini bertujuan untuk mengetahui apakah data hasil *makespan* dari kedua metode berdistribusi normal atau tidak. Hasil uji normalitas ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil Uji Normalitas Shapiro-Wilk

Metode	Statistik	P-value	Kesimpulan
SA Solusi Awal Acak	0,714	0,000	Data tidak berdistribusi normal
SA Solusi Awal SPT	0,819	0,000	Data tidak berdistribusi normal

Berdasarkan Tabel 4.8, diperoleh nilai *p-value* pada kedua metode lebih kecil dari 0,05. Dengan demikian, hipotesis nol (H_0) ditolak sehingga data hasil *makespan* pada kedua metode tidak berdistribusi normal. Karena data tidak memenuhi asumsi normalitas, maka pengujian hipotesis dilanjutkan menggunakan uji nonparametrik *Mann-Whitney U*.

Uji *Mann-Whitney U* digunakan untuk mengetahui apakah terdapat perbedaan performa yang signifikan antara metode *Simulated Annealing* (SA) dengan solusi awal acak dan metode *Simulated Annealing* (SA) dengan solusi awal berbasis SPT. Hasil pengujian ditunjukkan pada Tabel 4.9.

Tabel 4.9 Hasil Uji Mann-Whitney U

Statistik	Nilai
U-statistik	524,000
P-value	0,254

Berdasarkan hasil uji *Mann-Whitney U*, diperoleh nilai *p-value* sebesar 0,254, yaitu lebih besar dari taraf signifikansi 0,05. Dengan demikian, hipotesis nol (H_0) gagal ditolak sehingga tidak terdapat perbedaan yang signifikan secara statistik antara metode *Simulated Annealing* (SA) dengan solusi awal acak dan metode *Simulated Annealing* (SA) dengan solusi awal berbasis SPT.

Hasil tersebut menunjukkan bahwa penggunaan solusi awal heuristik berbasis SPT belum memberikan peningkatan performa yang signifikan terhadap kualitas solusi akhir yang dihasilkan oleh algoritma *Simulated Annealing*. Meskipun metode *Simulated Annealing* (SA) dengan solusi awal berbasis SPT menghasilkan rata-rata *makespan* yang sedikit lebih baik dan standar deviasi yang lebih kecil, perbedaan tersebut belum cukup besar untuk dinyatakan signifikan secara statistik.

Hal ini menunjukkan bahwa algoritma *Simulated Annealing* memiliki kemampuan eksplorasi solusi yang cukup kuat sehingga tetap mampu menemukan solusi dengan kualitas yang relatif serupa meskipun menggunakan solusi awal yang berbeda. Dengan demikian, solusi awal berbasis SPT dapat membantu proses pencarian solusi pada algoritma *Simulated Annealing*, namun pada dataset FT06 pengaruhnya terhadap kualitas solusi akhir belum menunjukkan perbedaan yang signifikan dibandingkan solusi awal acak.

4.6 Perbandingan dalam Perspektif Islam

Allah SWT dalam Al-Qur'an telah mengajarkan manusia untuk menggunakan akal dan pertimbangan yang baik dalam memilih suatu pilihan. Hal ini dijelaskan dalam QS. Az-Zumar ayat 18:

الَّذِيْنَ يَسْتَمِعُوْنَ الْقَوْلَ فَيَتَّبِعُوْنَ اَحْسَنَهٗ ۗاُولٰۤىِٕكَ الَّذِيْنَ هَدٰىهُمُ اللّٰهُ وَاُولٰۤىِٕكَ هُمْ اُولُوا الْاَلْبَابِ ﴿١٨﴾

(Kementrian Agama Republik Indonesia, 2022)

"*(Yaitu) mereka yang mendengarkan perkataan lalu mengikuti apa yang paling baik di antaranya. Mereka itulah orang-orang yang telah diberi petunjuk oleh Allah dan mereka itulah ulul-albab (orang-orang yang mempunyai akal sehat).''* (QS. Az-Zumar: 18).

Menurut Tafsir Ibnu Katsir, *ulul-albab* adalah orang yang memiliki akal sehat dan fitrah yang lurus sehingga mampu membedakan antara berbagai pilihan, kemudian memilih jalan yang paling baik dan paling benar. Orang yang berpaling dari pilihan yang lebih baik seakan-akan telah kehilangan fungsi akalnya dalam menerima petunjuk Allah SWT (Katsir, 2000a). Sementara itu, Tafsir Al-Misbah menjelaskan bahwa orang-orang yang mengikuti yang terbaik merupakan pribadi yang memiliki pemikiran cemerlang, kreatif, dan senantiasa berusaha mencari alternatif yang lebih baik dalam setiap urusannya. Mereka tidak berhenti pada satu pilihan saja, tetapi melakukan pertimbangan sebelum menentukan keputusan yang paling tepat (Shihab, 2002b).

Nilai yang terkandung dalam ayat tersebut sejalan dengan proses yang dilakukan dalam penelitian ini. Penelitian ini tidak hanya menerapkan algoritma untuk memperoleh solusi penjadwalan, tetapi juga melakukan perbandingan antara dua pendekatan yang berbeda, yaitu SA dengan solusi awal acak dan SA dengan solusi awal berbasis SPT. Kedua pendekatan tersebut dievaluasi berdasarkan nilai

makespan, stabilitas hasil, konvergensi, frekuensi pencapaian solusi terbaik, serta pengujian statistik untuk mengetahui performa yang dihasilkan.

Selain itu, prinsip memilih alternatif yang lebih baik juga dijelaskan dalam hadits Arbain Nawawi nomor 11 yang diriwayatkan dari Abu Muhammad Hasan bin Ali bin Abi Thalib radhiyallahu 'anhuma. Rasulullah SAW bersabda:

حَدَّثَنَا أَبُو مُوسَى الأَنْصَارِيُّ، حَدَّثَنَا عَبْدُ اللَّهِ بْنُ إِدْرِيسَ، حَدَّثَنَا شُعْبَةُ، عَنْ بُرَيْدِ بْنِ أَبِي مَرْيَمَ، عَنْ أَبِي الْحَوْرَاءِ السَّعْدِيِّ، قَالَ قُلْتُ لِلْحَسَنِ بْنِ عَلِيٍّ مَا حَفِظْتَ مِنْ رَسُولِ اللَّهِ صلى الله عليه وسلم قَالَ حَفِظْتُ مِنْ رَسُولِ اللَّهِ صلى الله عليه وسلم: "دَعْ مَا يَرِيبُكَ إِلَى مَا لاَ يَرِيبُكَ فَإِنَّ الصِّدْقَ طُمَأْنِينَةٌ وَإِنَّ الْكَذِبَ رِيبَةٌ"
(At-Tirmidzi, n.d.)

"Telah menceritakan kepada kami Abu Musa al-Anshari, telah menceritakan kepada kami Abdullah bin Idris, telah menceritakan kepada kami Syu'bah, dari Buraid bin Abi Maryam, dari Abu al-Hawra' al-Sa'di, ia berkata: Aku bertanya kepada al-Hasan bin 'Ali: 'Apa yang engkau hafal dari Rasulullah?' Ia menjawab: 'Aku menghafal dari Rasulullah: Tinggalkanlah sesuatu yang meragukanmu menuju sesuatu yang tidak meragukanmu. Sesungguhnya kejujuran mendatangkan ketenangan dan sesungguhnya kebohongan mendatangkan keraguan." (HR. At-Tirmidzi No. 2518 dan An-Nasa'i No. 5711).

Imam Nawawi dalam *Al-Arba'in An-Nawawiyah* menjelaskan bahwa hadits ini mengajarkan agar seseorang meninggalkan perkara yang masih menimbulkan keraguan dan beralih kepada perkara yang lebih jelas serta lebih meyakinkan. Sikap tersebut merupakan bentuk kehati-hatian dalam mengambil keputusan dan termasuk salah satu jalan untuk mencapai ketakwaan (An-Nawawi, 2017). Senada dengan itu, Yazid bin Abdul Qadir Jawas menjelaskan bahwa kata *ya'ribuka* bermakna sesuatu yang menimbulkan keraguan, kebimbangan, atau was-was. Oleh karena itu, hadits ini memberikan tuntunan agar seorang muslim memilih sesuatu yang lebih jelas dan lebih dapat dipertanggungjawabkan dibandingkan pilihan yang masih mengandung ketidakpastian (Jawas, 2018).

Nilai yang terkandung dalam hadits tersebut sejalan dengan proses evaluasi yang dilakukan dalam penelitian ini. Penelitian ini membandingkan dua pendekatan, yaitu SA dengan solusi awal acak dan SA dengan solusi awal berbasis SPT. Melalui pengujian sebanyak 30 percobaan, analisis konvergensi, frekuensi pencapaian solusi terbaik, serta uji statistik Mann-Whitney U, diperoleh bukti bahwa solusi awal berbasis SPT menghasilkan rata-rata *makespan* yang lebih baik dan performa yang lebih stabil dibandingkan solusi awal acak. Dengan demikian, proses perbandingan yang dilakukan dalam penelitian ini merupakan upaya untuk memperoleh pendekatan yang lebih meyakinkan berdasarkan data dan hasil pengujian yang objektif.

Namun demikian, hasil akhir tetap tidak sepenuhnya dapat dipastikan, karena terdapat unsur probabilitas dalam algoritma tersebut. Hal ini dapat dikaitkan dengan konsep *tawakkal*, yaitu menyerahkan hasil akhir kepada Allah SWT setelah melakukan usaha secara maksimal.

Hal ini sejalan dengan surah Ali Imran ayat 159 yang menjelaskan bahwa dalam Islam tidak hanya mengandalkan kemampuan manusia, tetapi juga melibatkan tawakkal, yaitu berserah diri kepada Allah setelah berusaha maksimal. Allah SWT berfirman:

فَبِمَا رَحْمَةٍ مِّنَ اللّٰهِ لِنْتَ لَهُمْۚ وَلَوْ كُنْتَ فَظًّا غَلِيْظَ الْقَلْبِ لَانْفَضُّوْا مِنْ حَوْلِكَۖ فَاعْفُ عَنْهُمْ وَاسْتَغْفِرْ لَهُمْ وَشَاوِرْهُمْ فِى الْاَمْرِۚ فَاِذَا عَزَمْتَ فَتَوَكَّلْ عَلَى اللّٰهِۗ اِنَّ اللّٰهَ يُحِبُّ الْمُتَوَكِّلِيْنَ ﴿١٥٩﴾

(Kementrian Agama Republik Indonesia, 2022)

" Maka, berkat rahmat Allah engkau (Nabi Muhammad) berlaku lemah lembut terhadap mereka. Seandainya engkau bersikap keras dan berhati kasar, tentulah mereka menjauhkan diri dari sekitarmu. Karena itu, maafkanlah mereka, mohonkanlah ampunan untuk mereka, dan bermusyawarah-lah dengan mereka dalam urusan itu. Kemudian, apabila engkau telah membulatkan tekad, bertawakallah kepada Allah. Sesungguhnya Allah mencintai orang-orang yang bertawakal." (QS. Ali Imran: 159).

Ibnu Katsir menjelaskan bahwa bagian akhir ayat ini, memerintahkan Nabi SAW setelah bermusyawarah dan membulatkan tekad (ʿazamta), hendaklah bertawakal kepada Allah dalam urusan tersebut. Tawakal di sini adalah penyerahan sepenuhnya setelah usaha maksimal, karena Allah SWT menyukai hamba yang bertawakal (Katsir, 2000c). Quraish Shihab dalam Tafsir Al-Misbah menafsirkan frasa yang sama sebagai instruksi: setelah musyawarah dan tekad bulat, serahkan hasilnya kepada Allah SWT (tawakal), karena penilaian benar-salah mutlak milik-Nya. Tawakal bukan pasrah malas, melainkan usaha sungguh-sungguh diikuti penyerahan ikhlas, mencakup sikap akhlak kepada Allah pasca-akhlak sosial (lemah lembut, pemaaf, musyawarah) (Choirunisa', 2021). Hal ini menunjukkan bahwa Islam tidak mengajarkan sikap pasif, melainkan menekankan pentingnya usaha maksimal sebelum menyerahkan hasil kepada Allah SWT.

Dengan demikian, penelitian ini tidak hanya memberikan kontribusi dalam bidang optimasi dan komputasi, tetapi juga sejalan dengan nilai-nilai Islam dalam pengelolaan waktu, profesionalisme dalam bekerja, serta keseimbangan antara usaha dan tawakkal.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa algoritma *Simulated Annealing* (SA) mampu digunakan untuk menyelesaikan *Job Shop Scheduling Problem* (JSSP) pada dataset FT06 dengan menghasilkan solusi penjadwalan yang mendekati optimal. Proses optimasi dilakukan melalui mekanisme pembentukan solusi tetangga, evaluasi perubahan nilai *makespan*, serta penerimaan solusi berdasarkan probabilitas tertentu sehingga algoritma mampu mengeksplorasi ruang solusi secara lebih luas.

Hasil implementasi menunjukkan bahwa metode *Simulated Annealing* dengan solusi awal acak maupun *Simulated Annealing* dengan solusi awal berbasis *Shortest Processing Time* (SPT) sama-sama mampu menghasilkan nilai *makespan* yang baik. Berdasarkan 30 kali percobaan, metode *Simulated Annealing* dengan solusi awal berbasis SPT menghasilkan rata-rata *makespan* sebesar 57,30 dengan standar deviasi 1,55, sedangkan metode *Simulated Annealing* murni menghasilkan rata-rata *makespan* sebesar 57,63 dengan standar deviasi 1,68. Selain itu, kedua metode memiliki frekuensi pencapaian nilai *makespan* optimal 55 yang sama yaitu 8 kali dari 30 percobaan.

Namun demikian, berdasarkan hasil uji statistik menggunakan *Mann-Whitney U*, diperoleh nilai *p-value* sebesar 0,254 $(p > 0{,}05)$, sehingga tidak terdapat perbedaan yang signifikan secara statistik antara kedua metode. Hasil tersebut menunjukkan bahwa penggunaan solusi awal heuristik berbasis SPT belum

memberikan pengaruh yang signifikan terhadap kualitas solusi akhir yang dihasilkan oleh algoritma *Simulated Annealing* pada dataset FT06.

Penelitian ini menunjukkan bahwa algoritma *Simulated Annealing* memiliki kemampuan eksplorasi solusi yang cukup baik sehingga tetap mampu menghasilkan solusi dengan kualitas yang relatif serupa meskipun menggunakan solusi awal yang berbeda. Dengan demikian, solusi awal heuristik berbasis SPT dapat membantu proses pencarian solusi, namun pengaruhnya terhadap kualitas solusi akhir belum menunjukkan perbedaan yang signifikan dibandingkan solusi awal acak.

5.2 Saran untuk Penelitian Lanjutan

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran yang dapat diberikan untuk pengembangan penelitian selanjutnya.

1. Penelitian selanjutnya dapat menggunakan dataset JSSP dengan ukuran yang lebih besar dan tingkat kompleksitas yang lebih tinggi untuk mengetahui pengaruh solusi awal heuristik terhadap performa *Simulated Annealing* secara lebih luas.
2. Penelitian berikutnya dapat membandingkan berbagai metode pembentukan solusi awal selain aturan *Shortest Processing Time* (SPT), seperti *Longest Processing Time* (LPT), *Most Work Remaining* (MWR), atau heuristik lainnya untuk mengetahui pengaruh masing-masing solusi awal terhadap performa algoritma.
3. Pengembangan penelitian juga dapat dilakukan dengan melakukan analisis terhadap waktu komputasi, kecepatan konvergensi, dan stabilitas algoritma sehingga evaluasi performa tidak hanya berdasarkan nilai *makespan* akhir.

4. Penelitian selanjutnya dapat mengimplementasikan struktur data *Priority Queue* pada proses pemilihan operasi dalam algoritma SPT untuk meningkatkan efisiensi proses penjadwalan, khususnya pada dataset dengan jumlah *job* dan mesin yang lebih besar.
5. Penelitian berikutnya juga dapat melakukan proses *tuning parameter* secara lebih mendalam terhadap parameter *Simulated Annealing*, seperti temperatur awal, *cooling rate*, dan jumlah iterasi, karena parameter tersebut sangat memengaruhi performa algoritma dalam proses pencarian solusi.

DAFTAR PUSTAKA

Abdolrazzagh-Nezhad, M., & Abdullah, S. (2024). A Review on Metaheuristic Approaches for Job-Shop Scheduling Problems. *Data Science: Journal of Computing and Applied Informatics*, *8*(1), 45–63. https://doi.org/10.32734/jocai.v8.i1-17138

Abdurrahman bin Nashir as-Sa'di, S. (2015). *Tafsir Al-Qur'an jilid 7*. Darul Haq.

Al-Qurthubi, I. (2009). *Tafsir Al-Qur'an (19). Diterjemahkan oleh Muhyiddin Mas Rida dan M. Rana Mengala. Ed. Mukhlis B Mukti.* Pustaka Azzam.

An-Nawawi, I. (2017). *Syarah Hadits Arba'in An-Nawawiyah*. Jakarta: Pustaka Imam Asy-Syafi'i.

An-Nasa'i, A. S. (n.d.). *Al-Sunan al-Kubra (Sunan an-Nasa'i)* (Vol. 5, No. 5711).

At-Tirmidzi, M. bin I. (n.d.). *Al-Jami' al-Kabir (Sunan at-Tirmidzi)* (Vol. 4, No. 2518).

Basriati, S., Safitri, E., & Ermanita, M. (2020). Aplikasi Algoritma Greedy Terhadap Permasalahan Integer Knapsack pada Toko Surya Muda Pekanbaru. *Jurnal Sains Matematika Dan Statistika*, *6*(2).

Chakraborty, S., & Bhowmik, S. (2013). Job Shop Scheduling using Simulated Annealing. *First International Conference on Computation and Comunication Advancement (IC3A),* JIS, Vol. 1. https://www.researchgate.net/publication/316988567

Choirunisa', F. O. (2021). *Relevansi Nilai Nilai Pendidikan Akhlak dalam Surah Ali Imran : 159-160 Perspektif Tafsir Al Misbah dengan Masyarakat Modern* [Universitas Islam Negeri Maulana Malik Ibrahim]. http://etheses.uin-malang.ac.id/id/eprint/28727

Darnita, Y., & Toyib, R. (2019). Penerapan Algoritma Greedy Dalam Pencarian Jalur Terpendek Pada Instansi-Instasi Penting Di Kota Argamakmur Kabupaten Bengkulu Utara. In *Jurnal Media Infotama* (Vol. 15, Number 2).

Devita, R. N., & Wibawa, A. P. (2020). Teknik-teknik optimasi knapsack problem. *Sains, Aplikasi, Komputasi Dan Teknologi Informasi*, *2*(1), 35.

Fauzi, R. A., Widyasanti, A., Dwiratna, S., Perwitasari, N., & Nurhasanah, S. (2022). Optimization of Drying Process on Antioxidant Activity of Butterfly Pea (Clitoria ternatea) by Using Response Surface Methodology. In *Jurnal Teknologi Pertanian* (Vol. 23, Number 1).

Gromicho, J. A. S., Van Hoorn, J. J., Saldanha-Da-Gama, F., & Timmer, G. T. (2012). Solving the job-shop scheduling problem optimally by dynamic programming. *Computers and Operations Research*, *39*(12), 2968–2977. https://doi.org/10.1016/j.cor.2012.02.024

Guzman, E., Andres, B., & Poler, R. (2022). Matheuristic Algorithm for Job-Shop Scheduling Problem Using a Disjunctive Mathematical Model. *Computers*, *11*(1). https://doi.org/10.3390/computers11010001

Irwan, H. (2020). Optimasi Penjadwalan Job Shop dengan Metode Algoritma Greedy. *Profisiensi*, *8*(2), 164–176.

Jawas, Y. A. Q. (2018). *42 Faidah Hadits Arbain*. Bogor: Pustaka At-Taqwa.

Katsir, I. (2000a). *Tafsir al-Qur'an al-Azhim. Jilid 7*. Dar al-Tayyibah.

Katsir, I. (2000b). *Tafsir al-Qur'an al-Azhim. Jilid 8*. Dar al-Tayyibah.

Kementrian Agama Republik Indonesia. (2022). *Al-Qur'an dan Terjemahannya*. Lajnah Pentashihan Mushaf Al-Qur'an.

Khader, Y. M., Nurhasanah, Y. I., & Kartika, A. D. (2018). Penjadwalan Matakuliah Menggunakan Algoritma Greedy (Studi Kasus Penjadwalan Semester Ganjil 2017-2018 Informatika ITENAS). In *Jurnal Ilmiah Teknologi Informasi Terapan: IV* (Number 3).

Kurniawati, D. A. (2019). Penjadwalan Produksi Flow Shop untuk Meminimalkan Makespan dengan Metode Pour, Pemrograman Dinamis dan Branch and Bound di CV. Bonjor Jaya. *Jurnal Teknik Industri*, *9*, 63.

Manik, T. M., Gultom, P., & Nababan, E. (2018). Analisis Karakteristik Fungsi Lagrange Dalam Menyelesaikan Permasalahan Optimasi Berkendala. *Talenta Conference Series: Science and Technology (ST)*, *1*(1), 037–043. https://doi.org/10.32734/st.v1i1.187

Miloš Šeda. (2007). Mathematical Models of Flow Shop and Job Shop Scheduling Problems. *International Scholarly and Scientific Research & Innovation*, *1*(7).

Mubarakfuri, S. A. (2013). *Shahih Tafsir Ibnu Katsir Juz Amma (Edisi Bahasa Indonesia)*. Pustaka Ibnu Katsir.

Mujahidin, E., Rachmat, R., Tamam, A. M., & Alim, A. (2022). Konsep Manajemen Waktu dalam Perspektif Pendidikan Islam. *Edukasi Islami: Jurnal Pendidikan Islam*, *11*(01), 129. https://doi.org/10.30868/ei.v11i01.2203

Nurchoironi, M. H., & Nurrohim, A. (2025). Memahami Implementasi Manfaat Waktu pada Aktivitas Sehari-Hari dalam Surat Al-'Ashr Menurut Tafsir Al-Mishbah. *Jurnal Pendidikan Dan Kebudayaan (JURDIKBUD)*, *5*(2), 425–435. https://doi.org/10.55606/jurdikbud.v5i2.7309

Parhan, M., Maharani, A. J., Haqqu, O. A., Karima, Q. S., & Nurfaujiah, R. (2022). Orang Indonesia dan Jam Karet: Budaya Tidak Tepat Waktu dalam Pandangan Islam. *SOSIETAS: Jurnal Pendidikan Sosiologi*, *12*(1), 25–34. https://doi.org/10.17509/sosietas.v12i1.48065

Pérez, C., March, C., & Salido, M. (2024). Instance Configuration for Sustainable Job Shop Scheduling. *CEUR Workshop Proceedings*. https://gps.blogs.upv.es/

Puspitasari, I. (2016). Job Shop Scheduling Problem Modelling Using Petri Net for Making The Application of Scheduling Production Simulation. *JURNAL ITSMART*, *5*(1).

Riki, Nurtofik, Sultan, U., Muchtarom, & Mulyati, M. (2024). Analisis Nilai-Nilai Pendidikan dalam Al-Quran Surah Al-Ashr Ayat 1-3. *JIIP (Jurnal Ilmiah Ilmu Pendidikan)*, *7*(4), 3577–3586. http://Jiip.stkipyapisdompu.ac.id

Russel, R. S., & Taylor, B. W. (2011). *Operation Management*. John Wiley and Sons.

Samana, E., Prihandono, B., & Intisari, E. N. (2015). Aplikasi Simulated Annealing untuk Menyelesaikan Traveling Salesmen Problem. In *Buletin Ilmiah Mat. Stat. dan Terapannya (Bimaster)* (Vol. 03, Number 1).

Shihab, M. Q. (2002a). *Tafsir Al Misbah : Pesan, Kesan dan Keserasian Al-Qur'an* (Vol. 15). Lentera Hati.

Shihab, M. Q. (2002b). *Tafsir Al-Misbah: Pesan, Kesan, dan Keserasian Al-Qur'an* (Vol. 12). Jakarta: Lentera Hati.

Syukri, A. N. (2023). *Gunakan Setiap Kesempatan untuk Berbuat Kebajikan*. NU Online Jateng. https://jateng.nu.or.id/taushiyah/gunakan-setiap-kesempatan-untuk-berbuat-kebajikan-Sjfj4

Tamy0612. (n.d.). *JSPLIB: Benchmark instances for the job-shop scheduling problem*. GitHub Repository. Diakses dari https://github.com/tamy0612/JSPLIB

Tospornsampan, J., Kita, I., Ishii, M., & Kitamura, Y. (2007). Split-Pipe Design of Water Distribution Network Using Simulated Annealing. *World Academy of Science, Engineering and Technology, International Journal of Environmental, Chemical, Ecological, Geological and Geophysical Engineering*, *1*, 28–38. https://api.semanticscholar.org/CorpusID:14449230

Utomo, P. B., & Setiafindari, W. (2021). Optimasi Penjadwalan Produksi Menggunakan Metode Simulated Annealing di Industri XYZ. *Borobudur Engineering Review*, *1*(1), 49–55. https://doi.org/10.31603/benr.4610

Van Ekeris, T., Meyes, R., & Meisen, T. (2021). Discovering Heuristics And Metaheuristics For Job Shop Scheduling From Scratch Via Deep Reinforcement Learning. *Proceedings of the Conference on Production Systems and Logistics*, 709–718. https://doi.org/10.15488/11231

Wignjosoebroto, S. (2006). *Pengantar Teknik dan Manajemen Industri*. Guna Widya.

LAMPIRAN

Lampiran 1. Coding Implementasi Algoritma

```
import random
import math
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# ============================================================
# 1. DATASET ft06 JSSP (6 JOB, 6 MESIN) - BENCHMARK ASLI
# ============================================================
jobs = [
    [[2, 1], [0, 3], [1, 6], [3, 7], [5, 3], [4, 6]],  # Job 0
    [[1, 8], [2, 5], [4, 10], [5, 10], [0, 10], [3, 4]], # Job 1
    [[2, 5], [3, 4], [5, 8], [0, 9], [1, 1], [4, 7]],  # Job 2
    [[1, 5], [0, 5], [2, 5], [3, 3], [4, 8], [5, 9]],  # Job 3
    [[2, 9], [1, 3], [4, 5], [5, 4], [0, 3], [3, 1]],  # Job 4
    [[1, 3], [3, 3], [5, 9], [0, 10], [4, 4], [2, 1]]  # Job 5
]

num_jobs = 6
num_machines = 6
ops_per_job = 6

# ============================================================
# 2. FUNGSI INTI (MAKESPAN & NEIGHBOR)
# ============================================================
```

```
def calculate_makespan(sequence):
    m_avail = [0] * num_machines
    j_avail = [0] * num_jobs
    j_op_ptr = [0] * num_jobs

    for job_id in sequence:
        op_idx = j_op_ptr[job_id]
        # Ensure we don't go out of bounds for ops_per_job in case of malformed sequence
        if op_idx >= ops_per_job:
            continue # Skip if all operations for this job are already scheduled

        machine, duration = jobs[job_id][op_idx]
        start_time = max(m_avail[machine], j_avail[job_id])
        end_time = start_time + duration
        m_avail[machine] = end_time
        j_avail[job_id] = end_time
        j_op_ptr[job_id] += 1
    return max(m_avail) if m_avail else 0 # Return 0 if m_avail is empty

def generate_neighbor(sequence):
    seq = sequence.copy()
    idx1, idx2 = random.sample(range(len(seq)), 2)
    seq[idx1], seq[idx2] = seq[idx2], seq[idx1]
    return seq
```

```
# ============================================================
# 3. STRATEGI INITIAL SOLUTION
# ============================================================
def get_random_initial():
    seq = []
    for j in range(num_jobs):
        seq.extend([j] * ops_per_job)
    random.shuffle(seq)
    return seq

def get_spt_initial():
    machine_times = [0] * num_machines
    job_times = [0] * num_jobs
    job_op_ptr = [0] * num_jobs

    ready_set = [(j, 0) for j in range(num_jobs)]
    sequence = []

    while ready_set:
        candidates = []
```

```
        for j, op_idx in ready_set:
            m_id, duration = jobs[j][op_idx]

            start_time = max(machine_times[m_id], job_times[j])
            end_time = start_time + duration

            candidates.append({
                'start': start_time,
                'duration': duration,
                'job_id': j,
                'm_id': m_id,
                'end': end_time
            })

        candidates.sort(key=lambda x: (x['start'], x['duration']))

        best = candidates[0]

        sequence.append(best['job_id'])
        machine_times[best['m_id']] = best['end']
        job_times[best['job_id']] = best['end']

        ready_set.remove((best['job_id'], job_op_ptr[best['job_id']]))
        job_op_ptr[best['job_id']] += 1
```

```
        if job_op_ptr[best['job_id']] < ops_per_job:
            next_op = (best['job_id'], job_op_ptr[best['job_id']])
            ready_set.append(next_op)

    return sequence

# ============================================================
# GANTT CHART
# ============================================================
def plot_gantt(sequence, title):
    machine_available = [0] * num_machines
    job_available = [0] * num_jobs
    job_op_ptr = [0] * num_jobs

    fig, ax = plt.subplots(figsize=(10, 4))
    colors = ['red', 'blue', 'green', 'orange', 'purple',
              'brown', 'pink', 'gray', 'olive', 'cyan']

    y_labels = [f'M{m}' for m in range(num_machines)]
    ax.set_yticks(range(num_machines))
    ax.set_yticklabels(y_labels)
    ax.set_xlabel('Waktu')
    ax.set_title(title)

    max_time = 0
```

```
    for job_id in sequence:
        op_idx = job_op_ptr[job_id]
        if op_idx < ops_per_job:
            machine, proc_time = jobs[job_id][op_idx]
            start_time = max(machine_available[machine], job_available[job_id])
            end_time = start_time + proc_time

            machine_available[machine] = end_time
            job_available[job_id] = end_time
            job_op_ptr[job_id] += 1

            ax.add_patch(
                patches.Rectangle(
                    (start_time, machine - 0.4),
                    proc_time, 0.8,
                    color=colors[job_id % len(colors)],
                    alpha=0.7
                )
            )
            ax.text(start_time + proc_time / 2,
                    machine,
                    f'J{job_id}O{op_idx}',
                    ha='center', va='center', fontsize=8, color='white')
            max_time = max(max_time, end_time)
```

```
            else:
                pass

        ax.set_xlim(0, max_time + 1)
        ax.set_ylim(-0.5, num_machines - 0.5)
        ax.grid(True, axis='x', linestyle='--', alpha=0.5)
        plt.tight_layout()
        plt.show()

# ============================================================
# SIMULATED ANNEALING ALGORITHM
# ============================================================
# SA parameters, can be adjusted
SA_T0 = 100.0
SA_ALPHA = 0.95
SA_T_MIN = 0.1
SA_ITER_PER_TEMP = 10

def simulated_annealing_algorithm(initial_sequence, T_initial=SA_T0, alpha=SA_ALPHA, T_min=SA_T_MIN, iter_per_temp=SA_ITER_PER_TEMP):
    current_seq = initial_sequence[:]
    current_makespan = calculate_makespan(current_seq)
    best_seq = current_seq[:]
    best_makespan = current_makespan
    history = [] # To store the best makespan found so far at each temperature step
```

```
    T = T_initial
    while T > T_min:
        for _ in range(iter_per_temp):
            neighbor = generate_neighbor(current_seq)
            neighbor_makespan = calculate_makespan(neighbor)
            delta = neighbor_makespan - current_makespan

            if delta < 0:
                current_seq = neighbor[:]
                current_makespan = neighbor_makespan
            else:
                acceptance_prob = math.exp(-delta / T)
                if random.random() < acceptance_prob:
                    current_seq = neighbor[:]
                    current_makespan = neighbor_makespan

            if current_makespan < best_makespan:
                best_makespan = current_makespan
                best_seq = current_seq[:]
        history.append(best_makespan) # Record the best makespan for the current temperature
        T *= alpha
    return best_seq, best_makespan, history
```

```
# --- RUN THE SA PROCESS FOR BOTH INITIAL SOLUTIONS TO DEFINE VARIABLES ---
# SA Murni (Random Initial Solution)
init_random = get_random_initial()
murni_best_seq, murni_best_val, murni_history = simulated_annealing_algorithm(init_random)

# Hybrid SPT+SA (SPT Initial Solution)
init_spt = get_spt_initial()
hybrid_best_seq, hybrid_best_val, hybrid_history = simulated_annealing_algorithm(init_spt)

# --- OUTPUT HASIL ---
print("\n" + "="*30)
print(f"{'METODE':<15} | {'AWAL':<8} | {'TERBAIK':<8}")
print("-" * 30)
print(f"{'SA Solusi Awal Acak':<15} | {calculate_makespan(init_random):<8} | {murni_best_val:<8}")
print(f"{'SA Solusi Awal SPT':<15} | {calculate_makespan(init_spt):<8} | {hybrid_best_val:<8}")
print("="*30)

# --- Display Best Sequences & Gantt Charts ---
print("\n" + "="*50)
print("HASIL TERBAIK MAKESPAN DAN SEQUENCE")
print("="*50)

print(f"SA Solusi Awal Acak Terbaik: Makespan = {murni_best_val}")
print(f"Sequence SA Solusi Awal Acak: {murni_best_seq}")
```

```
print(f"\nSA Solusi Awal SPT: Makespan = {hybrid_best_val}")
print(f"Sequence SA Solusi Awal SPT: {hybrid_best_seq}")

# Gantt Charts (fungsi plot_gantt didefinisikan di cell lain dan harus sudah dijalankan)
print("\n" + "="*50)
print("VISUALISASI JADWAL (GANTT CHARTS)")
print("="*50)
plot_gantt(murni_best_seq, f"Gantt Chart - SA Solusi Awal Acak (Makespan: {murni_best_val})")
plot_gantt(hybrid_best_seq, f"Gantt Chart - SA Solusi Awal SPT (Makespan: {hybrid_best_val})")
```

```python
import pandas as pd

# Pastikan kedua history memiliki panjang yang sama, ambil yang terpendek jika berbeda
min_len = min(len(murni_history), len(hybrid_history))

convergence_data = {
    'Iterasi': range(1, min_len + 1),
    'Makespan SA Murni': murni_history[:min_len],
    'Makespan Hybrid SPT+SA': hybrid_history[:min_len]
}

convergence_df = pd.DataFrame(convergence_data)

print("\n" + "="*60)
print(f"{'Tabel Konvergensi Makespan per Iterasi':^60}")
print("="*60)
display(convergence_df)
print("="*60)
```

```python
plt.figure(figsize=(12, 7))
plt.plot(murni_history, label='SA (Random Initial)', color='red', linestyle='--')
plt.plot(hybrid_history, label='SA (Greedy SPT Initial)', color='blue', linewidth=2)
plt.axhline(y=55, color='green', linestyle=':', label='Optimal ft06 (55)')
plt.title('Perbandingan Konvergensi: SA (Random Initial) vs SA (Greedy SPT Initial)')
plt.xlabel('Iterasi (Perubahan Temperatur)')
plt.ylabel('Makespan Terbaik')
plt.legend()
plt.grid(True)
plt.show()
```

RIWAYAT HIDUP

Anifatun Nisa', lahir di Bojonegoro pada tanggal 27 Februari 2004. Penulis merupakan anak kedua dari dua bersaudara. Anak dari Bapak Sagi dan Ibu Rupini. Penulis telah menempuh pendidikan mulai dari TK ABA II yang lulus pada tahun 2010, dilanjutkan menempuh pendidikan sekolah dasar di SD Negeri Krondonan II dan lulus pada tahun 2016. Kemudian penulis melanjutkan pendidikan sekolah menengah pertama di SMP Negeri 2 Gondang dan lulus pada tahun 2019. Setelah itu, melanjutkan pendidikan madrasah aliyah di MA Ma'arif 7 Sunan Drajat dan lulus pada tahun 2022. Pada tahun yang sama, penulis melanjutkan pendidikan di Universitas Islam Negeri Maulana Malik Ibrahim Malang pada Program Studi Matematika, Fakultas Sains dan Teknologi. Selama menempuh pendidikan tinggi, penulis turut berkontribusi aktif dalam berbagai kegiatan baik internal maupun eksternal kampus. Penulis juga aktif dalam kepanitiaan internal maupun eksternal kampus serta mengikuti kegiatan di luar kampus seperti pelatihan dan seminar.

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No.50 Dinoyo Malang Telp. / Fax. (0341)558933

BUKTI KONSULTASI SKRIPSI

Nama : Anifatun Nisa'
NIM : 220601110011
Fakultas / Jurusan : Sains dan Teknologi / Matematika
Judul Skripsi : Evaluasi Kontribusi Solusi Awal Heuristik terhadap Performa *Simulated Annealing* pada *Job Shop Scheduling Problem*
Pembimbing I : Mohammad Nafie Jauhari, M.Si.
Pembimbing II : Erna Herawati, M.Pd.

No	Tanggal	Hal	Tanda Tangan
1.	27 September 2025	Konsultasi Topik	1.
2.	28 Oktober 2025	Konsultasi Bab I, II, dan III	2.
3.	29 Oktober 2025	Konsultasi Bab I, II, dan III	3.
4.	29 Oktober 2025	ACC Bab I, II, dan III	4.
5.	30 Oktober 2025	Konsultasi Kajian Agama Bab I dan II	5.
6.	7 November 2025	Konsultasi Kajian Agama Bab I dan II	6.
7.	13 November 2025	ACC Kajian Agama Bab I dan II	7.
8.	19 November 2025	ACC Seminar Proposal	8.
9.	7 Januari 2026	Konsultasi Revisi Seminar Proposal	9.
10.	16 April 2026	Konsultasi Bab IV dan V	10.
11.	5 Mei 2026	Konsultasi Kajian Agama Bab IV	11.
12.	6 Mei 2026	Konsultasi Bab IV dan V	12.
13.	7 Mei 2026	Konsultasi Bab IV dan V	13.
14.	7 Mei 2026	Konsultasi Kajian Agama Bab IV	14.
15.	11 Mei 2026	Konsultasi Bab IV dan V	15.

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No.50 Dinoyo Malang Telp. / Fax. (0341)558933

16.	18 Mei 2026	ACC Bab IV dan V	16.
17.	18 Mei 2026	ACC Kajian Agama Bab IV	17.
18.	18 Mei 2026	ACC Seminar Hasil	18.
19.	26 Mei 2026	Konsultasi Revisi Seminar Hasil	19.
20.	2 Juni 2026	ACC Sidang Skripsi	20.
21.	10 Juni 2026	ACC Keseluruhan	21.

Malang, 10 Juni 2026
Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M.Si.
NIP. 19800527 200801 1 012