PERBANDINGAN ALGORITMA *CAMPBELL DUDEK SMITH & PARTICLE SWARM OPTIMIZATION* PADA PENGOPTIMALAN PENJADWALAN *PERMUTATION FLOWSHOP*

SKRIPSI

FAHIRA FARENSIA EDIYANTO

H011201027

PROGRAM STUDI MATEMATIKA DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

APRIL 2024

PERBANDINGAN ALGORITMA CAMPBELL DUDEK SMITH & PARTICLE SWARM OPTIMIZATION PADA PENGOPTIMALAN PENJADWALAN PERMUTATION FLOWSHOP

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains pada Program Studi Matematika Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin

FAHIRA FARENSIA EDIYANTO

H011201027

PROGRAM STUDI MATEMATIKA DEPARTEMEN MATEMAMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

APRIL 2024

LEMBAR PERNYATAAN KEOTENTKAN

saya yang bertanda tangan dibawah ini menyatakan dengan sungguh-sungguh bahwa skripsi yang saya buat dengan judul:

Perbandingan Algoritma *Campbell Dudek Smith & Particle Swarm Optimization* pada Pengoptimalan Penjadwalan *Permutation Flowshop*

Adalah benar hasil karya saya sendiri, bukan hasil plagiat dan belum pernah dipublikasikan dalam bentuk apapun.

Makassar, 23 April 2024

Fahira Farensia Ediyanto

NIM. H011201027

PERBANDINGAN ALGORITMA CAMPBELL DUDEK SMITH & PARTICLE SWARM OPTIMIZATION PADA PENGOPTIMALAN PENJADWALAN PERMUTATION FLOWSHOP

disetuji oleh:

Pembimbing Utama

Dr. Khaeruddin, M.Sc.

NIP. 196509141991031003

Pada 23 April 2024

HALAMAN PENGESAHAN

PERBANDINGAN ALGORITMA *CAMPBELL DUDEK SMITH & PARTICLE SWARM OPTIMIZATION* PADA PENGOPTIMALAN PENJADWALAN *PERMUTATION FLOWSHOP*

Disusun dan diajukan oleh

FAHIRA FARENSIA EDIYANTO

H011201027

Telah dipertahankan dihadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin pada tanggal 23 April 2024 dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui

Pembimbing Utama

Dr. Khaeruddin, M.Sc.

NIP. 196509141991031003

Ketua Program Studi

Dr. Firman, S.Si, M.Si.

NIP. 196804292002121001

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	: Fahira Farensia Ediyanto
NIM	: H011201027
Program Studi	: Matematika
Judul skripsi	: Perbandingan Algoritma *Campbell Dudek Smith & Particle Swarm Optimization* pada Pengoptimalan Penjadwalan *Permutation Flowshop*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian dari persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

DEWAN PENGUJI

1. Ketua : Dr.Khaeruddin, M.Sc. ( )
2. Anggota : Prof. Dr. Budi Nurwahyu, MS. ( )
3. Anggota : Prof. Dr. Moh. Ivan Azis, M.Sc. ( )

Ditetapkan di : Makassar

Tanggal : 23 April 2024

KATA PENGANTAR

Segala puji bagi Tuhan YME yang telah melimpahkan memberikan berkat-Nya kepada penulis sehingga skripsi dengan judul "Perbandingan Algoritma *Campbell Dudek Smith & Particle Swarm Optimization* pada Pengoptimalan Penjadwalan *Permutation Flowshop*" dapat terselesaikan. Skripsi ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Sains pada Program Studi Matematika Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

Penulis mengucapkan terima kasih khususnya kepada kedua orang tua Bapak **Heru Ediyanto** dan Ibu **Dorkas Ermina** yang telah membesarkan dan mendidik penulis dengan penuh kesabaran serta selalu mencurahkan kasih sayang yang tak pernah putus, memberikan dukungan dan doa sehingga penulis dapat menyelesaikan penulisan skripsi ini. Begitu pula kepada kedua adik penulis **Mahadna Kirana Ediyanto** dan **Mahesa Adiwangsa Ediyanto** yang telah memberikan dukungan pada penulis. Terima kasih telah menjadi sumber motivasi dan kekuatan bagi penulis selama perjalanan akademik ini.

Penulis menyadari bahwa penyelesaian skripsi ini tidak lepas dari dukungan dan bantuan banyak pihak baik secara langsung maupun tidak langsung. Untuk itu pada kesempatan ini penulis mengucapkan terima kasih dan penghargaan yang setinggi-tingginya kepada:

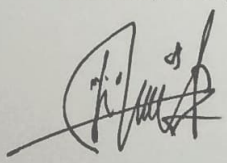
1. Bapak **Prof. Dr. Ir. Jamaluddin Jompa, M.Sc.**, selaku Rektor Universitas Hasanuddin.
2. Bapak **Dr. Eng. Amiruddin, S.Si., M.Si.**, selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.
3. Bapak **Dr. Khaeruddin., M.Sc.**, selaku pembimbing untuk segala ilmu, nasihat, dan kesabaran dalam membimbing dan mengarahkan penulis, serta bersedia meluangkan waktunya untuk mendampingi penulis sehingga skripsi ini dapat diselesaikan.
4. Bapak **Prof. Dr. Moh. Ivan Azis, M.Sc.**, dan Bapak **Prof. Dr. Budi Nurwahyu, MS.**, selaku penguji dan penasihat akademik penulis yang telah

bersedia meluangkan waktunya untuk memberikan saran dan arahan kepada penulis dalam penulisan skripsi ini.

5. Bapak/Ibu Dosen Departemen Matematika yang telah membagikan ilmu dan pengalamannya, serta Staf Departemen Matematika atas segala bantuannya.
6. Sahabat-sahabat penulis **Indah Puspita Sari**, **Anggeline Malino**, **Nur Atila Ayu**, dan **Gabriella Mega Lulun Bara** yang telah membantu, menemani, menyemangati, dan menjadi tempat berbagi keluh-kesah penulis selama masa perkuliahan.
7. Teman-teman tersayang penulis **Hilda**, **Asfi**, **Afliani**, **Esra**, **Janneth**, **Merlis**, **Rowina**, **Sisil**, **Wardalisah**, **Nurpadian**, **Nurkholisa Halim**, **Sulfina**, **Dian Fadlu** dan **Eko Sura'** yang telah membantu dan menjadi teman diskusi penulis selama perkuliahan dan proses penulisan skripsi.
8. Teman-teman Matematika 2020 atas segala dukungan, kebersamaan, dan kerjasamanya selama ini.
9. Teman-teman KKNT G-110 Posko Bonto Bontoa khususnya **Cut Alfifah Rasdeya, Muhammad Syawaluddin, A. Nur Syahdi Eka Saputra** dan **Fatimah** atas segala dukungan, kebersamaan, serta pengalaman baru yang diperoleh selama proses KKN hingga saat ini.
10. Serta segala pihak yang tidak dapat disebutkan satu persatu yang telah membantu penulis dalam menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, kritik dan saran yang membangun diharapkan oleh penulis untuk perbaikan dan pengembangan penelitian lebih lanjut. Akhir kata, semoga hasil penelitian ini dapat bermanfaat bagi kalangan akademisi, praktisi, dan semua pihak.

Makassar, 23 April 2024



Fahira Farensia Ediyanto

PERNYATAAN PERSETUJUAN TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universtas Hasanuddin, saya yang bertanda tangan dibawah ini:

Nama	: Fahira Farensia Ediyanto
NIM	: H011201027
Program Studi	: Matematika
Departemen	: Matematika
Fakultas	: Matematika dan Ilmu Pengetahuan Alam
Jenis Karya	: Skripsi

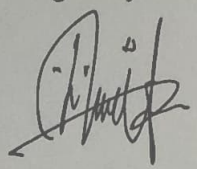
Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Hasanuddin **Hak Bebas Royalti Noneksklusif** (***Non-exclusive Royalty-Free Right***) atas karya ilmiah saya yang berjudul:

"Perbandingan Algoritma *Campbell Dudek Smith & Particle Swarm Optimization* pada Pengoptimalan Penjadwalan *Permutation Flowshop*"

beserta perangkat yang ada (jika diperlukan). Terkait dengan hal di atas, maka pihak universtas berhak menyimpan, mengalih-media/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini daa buat dengan sebenarnya.
Dibuat di Makassar pada Tanggal 23 April 2024

Yang menyatakan



Fahira Farensia ediyanto

ABSTRAK

Permutation Flowshop Problem (PFSP) merupakan salah satu variasi dari *flowshop problem* yang paling populer, dimana semua proses pekerjaan harus mengikuti urutan yang sama pada semua mesin agar dapat menemukan penjadwalan terbaik untuk n pekerjaan. Telah banyak literatur terdahulu yang fokus pada penyelesaian PFSP menggunakan algoritma *metaheuristic* dan algoritma *hybrid*. Salah satu algortima *metaheuristic* yang dapat digunakan untuk penyelesaian PFSP adalah algoritma *Particle Swarm Optimization* (PSO) yang memiliki kemampuan untuk memperbaiki solusi. Penelitian ini bertujuan untuk membandingkan algoritma PSO dan algortima *Campbell Dudek Smith* (CDS) yang merupakan satu dari banyak algoritma *heuristic* yang dapat diaplikasikan pada PFSP. CDS dikembangkan dari dasar aturan *Johnson* yang mampu meminimalkan makespan dalam penjadwalan tipe *flowshop*. PSO dan CDS memiliki perbedaan pada tipe algortima dan struktur kerja, sehingga perbandingan dalam penelitian ini dilakukan dengan melihat kinerja dan makespan optimal yang dapat diberikan oleh kedua algoritma dengan bahasa pemrograman MATLAB. Kedua algortima diaplikasikan pada delapan dataset dengan ukuran, jumlah pekerjaan yang akan dijadwalkan dan rentang nilai waktu pemrosesan yang berbeda-beda untuk melihat keunggulan dan faktor-faktor yang dapat mempengaruhi kinerja dari kedua algoritma. Pengaplikasian kedua algoritma pada *benchmark problem* menunjukkan bahwa PSO dapat memberikan makespan yang jauh lebih optimal dibandingkan CDS dengan minimumnya nilai *best relative error*, namun dalam pengaplikasiannya CDS memiliki struktur kerja yang lebih sederhana sehingga lebih mudah untuk diaplikasikan.

Kata kunci: *Permutation Flowshop Problem, PSO, CDS, MATLAB, Makespan, Penjadwalan.*

Judul : Perbandingan Algoritma *Campbell Dudek Smith & Particle Swarm Optimization* pada Pengoptimalan Penjadwalan *Permutation Flowshop*

Nama : Fahira Farensia Ediyanto

NIM : H011201027

Program Studi : Matematika

ABSTRACT

The Permutation Flowshop Problem (PFSP) is one variation of the most popular flowshop problem, where all work processes must follow the same sequence on all machines in order to find the best scheduling for n jobs. There has been a lot of previous literature that focuses on solving PFSP using metaheuristic algorithms and hybrid algorithms. One metaheuristic algorithm that can be used for PFSP solving is the Particle Swarm Optimization (PSO) algorithm which has the ability to improve solutions. This study aims to compare the PSO algorithm and Campbell Dudek Smith (CDS) algorithm which is one of many heuristic algorithms that can be applied to PFSP. CDS was developed from the foundation of Johnson's rule which is able to minimize makespan in flowshop type scheduling. PSO and CDS have differences in algorithm type and work structure, so the comparison in this study was made by looking at the optimal performance and makespan that can be provided by both algorithms with the MATLAB programming language. Both algorithms were applied to eight datasets with different sizes, number of jobs to be scheduled and different ranges of processing time values to see the advantages and factors that can affect the performance of both algorithms. The application of both algorithms to the benchmark problem shows that PSO can provide a much more optimal makespan than CDS with a minimum best relative error value, but in its application CDS has a simpler work structure so that it is easier to apply.

Keywords: *Permutation Flowshop Problem, PSO, CDS, MATLAB, Makespan, Scheduling.*

Judul : Campbell Dudek Smith & Particle Swarm Optimization Algorithm Comparison on Flowshop Permutation Scheduling Optimization

Nama : Fahira Farensia Ediyanto

NIM : H011201027

Program Studi : Mathematics

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penjadwalan merupakan salah satu aplikasi dari optimisasi yang sering dijumpai pada kehidupan sehari-hari. Contoh penjadwalan yang sering dijumpai adalah penjadwalan shift para pekerja pada sebuah perusahaan atau penjadwalan sejumlah mesin dalam industri manufaktur. Penjadwalan sangat terikat dengan waktu dan sumber daya, sehingga penjadwalan dapat digambarkan sebagai proses dalam menentukan urutan tugas dengan pengaturan waktu dan alokasi sumber daya untuk mengoptimalkan efisiensi maupun produktifitas (Pinedo, 2012).

Salah satu masalah yang sering dikaji dalam industri manufaktur dan produksi adalah penjadwalan *flowshop* atau alur produksi. Sama seperti dengan penjadwalan pada umumnya, penjadwalan alur produksi merupakan proses penentuan urutan dari beberapa tugas. Penjadwalan *flowshop* memiliki prinsip dimana tugas harus diproses secara berurutan melalui serangkaian mesin yang berbeda, sehingga setiap tugas memerlukan urutan operasi yang spesifik agar pekerjaan dapat selesai (Baker & Trietsch, 2019). Tujuan dari penjadwalan alur produksi adalah untuk mengelola produksi, mengoptimalkan penggunaan sumber daya dan meminimalkan waktu penyelesaian total atau makespan.

Permutation Flowshop Problem (PFSP) merupakan contoh masalah penjadwalan alur produksi yang dikenal dengan relevansi teknik yang luas. *Permutation Flowshop* termasuk kedalam *pure flowshop* yang merupakan variasi dari *flowshop problem* di mana terdapat i ($i \in \{1, 2, 3 \dots, n\}$) pekerjaaan yang akan diproses pada setiap j ($j \in \{1, 2, 3 \dots, m\}$) mesin dengan urutan pekerjaan pada semua mesin harus sama dan setiap mesin hanya dapat memproses satu pekerjaan pada satu waktu (Mishra dan Shrisvastava, 2020). Sebagai salah satu masalah penjadwalan, *Permutation Flowshop* mencerminkan skenario yang banyak ditemui di dunia produksi, baik dalam manufaktur, logistik hingga transportasi. Dengan berbagai skenarionya, PFSP menjadi topik yang cukup menarik dalam bidang ilmu matematika terapan.

Penyelesaian *Flowshop Problem* telah dikaji oleh banyak peneliti terdahulu dengan berbagai algoritma, diantaranya adalah algoritma CDS dan PSO. Algoritma

Campbell Dudek Smith (CDS) merupakan algoritma *heuristic* yang diperkenalkan pada tahun 1965 dengan tujuan meminimalkan makespan untuk setiap (n) job pada (m) mesin. Pada tahun 2020, Sidabutar dkk menunjukkan penggunaan algoritma *Campbell Dudek Smith* untuk penjadwalan operasi mesin pada tulisannya yang berjudul "Penjadwalan Operasi Mesin Produksi Dengan Metode CDS (*Campbell Dudek Smith*) Di Pt Tjokro Bersaudara Balikpapanindo" dengan hasil penjadwalan yang lebih optimal dibandingkan penjadwalan sebelumnya tanpa algoritma CDS, dimana makespan sebelumnya sebesar 33.46 jam yang kemudian ditekan menjadi 32.6 jam. Di sisi lain, algoritma PSO atau *Particle Swarm Optimization* dapat menjadi alternatif lain sebagai salah satu algoritma *meta-heuristic* yang mampu mengeksplorasi ruang dan solusi pencarian. Pada tahun 2007, Tasgetiren dkk berhasil menyelesaikan tulisannya mengenai penyelesaian *Permutation Flowshop* dengan algoritma *Particle Swarm Optimization*, dengan judul "*A Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in The Permutation Flowshop Sequencing Problem*". Dalam penelitiannya tersebut, Tasgetiren dkk berhasil menunjukkan penggunaan PSO dengan *local search* dalam penyelesaian *Permutation Flowshop Problem* dengan hasil yang menjanjikan pada dataset *Benchmark Taillard*. Selanjutnya di tahun 2019, Irman dkk menunjukkan hasil perbandingan dari algoritma CDS dan PSO pada *flowshop problem* untuk meminimalisasikan makespan PT KHI Pipe Industries. Pada penelitiannya tersebut yang berjudul *"Minimizing Makespan on Flow Shop Scheduling Using Campbel Dudek and Smith, Particle Swarm Optimization, And Proposed Heuristic Algorithm",* PSO mampu memberikan makespan yang lebih unggul dengan makespan sebesar 711.96 jam sedangkan CDS mampu memberikan makespan sebesar 742.87 jam dari makespan awal sebesar 746.43 jam.

Algoritma CDS dan PSO memliki jenis, dan prinsip kerja yang berbeda dalam menangani *Permutation Flowshop problem.* Selain keunggulan, peneliti ingin mengetahui keterbatasan dan faktor pendukung yang dimiliki oleh masing-masing algoritma sehingga dapat menentukan pendekatan yang lebih baik dalam penyelesaian PFSP. Dengan demikian penulis memilih "Perbandingan Algoritma *Campbell Dudek Smith & Paticle Swarm Optimization* pada penyelesaian *Permutation Flowshop Schedulling Problem*" sebagai judul dari penelitian ini.

1.2 Rumusan Masalah

Rumusan masalah yang akan dikaji dalam penelitian ini adalah penyelesaian *Permutation Flowshop Problem* dengan membandingkan kinerja dan hasil dari algoritma *Campbell Dudek Smith* dan *Particle Swarm Optimization*. Dengan fokus yang lebih sempit, penelitian ini akan membahas:

1. Bagaimana penerapan algoritma *Campbell Dudek Smith* dan *Paticle Swarm Optimization* pada *Permutation Flowshop Problem*?
2. Bagaimana hasil yang diperoleh dari perbandingan algoritma *Campbell Dudek Smith* dan *Paticle Swarm Optimization* pada penyelesaian *Permutation Flowshop Problem*?

1.3 Batasan Masalah

Pada penelitian ini, masalah akan dibatasi agar tidak melenceng dari fokus dan relevansi peneilitian. Masalah penjadwalan *flowshop* yang digunakan terdiri dari delapan dataset penelitian terdahulu (*Benchmark Problem*) dengan metrik evaluasi berupa total waktu penyelesaian minimum & kompleksitas algoritma.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan dari adanya penelitian ini adalah:

1. Mengimplementasikan algoritma *Campbell Dudek Smith* dan *Particle Swarm Optimization* untuk menyelesaikan *Permutation Flowshop Problem.*
2. Memperoleh hasil perbandingan algoritma *Campbell Dudek Smith* dan *Particle Swarm Optimization* dalam penyelesaian *Permutation Flowshop Problem.*

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan pemahaman kinerja dari kedua algoritma dalam penerapannya pada *Permutation Flowshop Problem*, sehingga dapat menjadi potensi dalam pengembangan penelitian dan bahan pustaka di Universitas Hasanuddin.

BAB II

TINJAUAN PUSTAKA

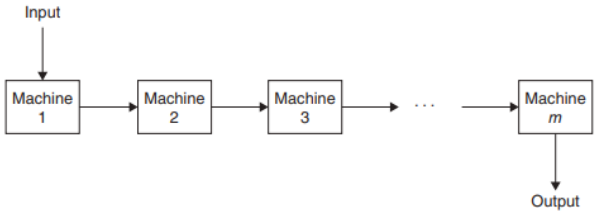
2.1 Penjadwalan

Proses penjadwalan memiliki peran yang penting bagi industri, baik dalam proses produksi, manufaktur atau bahkan transportasi. Menurut Pinedo (2012) proses penjadwalan didefinisikan sebagai pengalokasian sumber daya pada tugas-tugas atau pekerjaan dalam jangka waktu tertentu dengan tujuan untuk mengoptimalkan satu atau lebih tujuan.

Melihat peran penjadwalan yang sering terjadi pada kehidupan sehari-hari, banyak peneliti terdahulu yang memodelkan proses penjadwalan menjadi masalah penjadwalan. Salah satu masalah penjadwalan yang cukup menarik untuk dibahas adalah masalah penjadwalan dalam workshop dimana masalah penjadwalan tersebut diklasifikasi menjadi tiga, yaitu: *Job Shop Scheduling Problem, Flowshop Scheduling Problem* dan *Open Shop Scheduling Problem*. Diantara ketiga masalah penjadwalan tersebut, *Flowhop* atau alur produksi adalah masalah penjadwalan yang cukup penting, yang dikenal sebagai *flow continuous* dari beberapa pekerjaan pada beberapa mesin dengan kriteria dan kondisi yang berbeda-beda.

2.2 Flowshop

Penjadwalan alur produksi atau *flowshop* merupakan salah satu masalah yang cukup sering dibahas dalam penjadwalan. Dalam *flowshop* pekerjaan dibagi menjadi beberapa tugas yang disebut sebagai operasi atau proses, dan setiap operasi dikerjakan pada mesin yang berbeda. Dalam *Pure flowshop* setiap pekerjaan akan melalui m operasi dalam urutan yang sama dan dikerjakan dengan mesin berbeda (Baker & Trietsch, 2019). Contoh alur produksi dari *Pure Flowshop* dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Pure flowshop. Sumber: Principles of Sequencing and Scheduling Second Edition.

Dalam penjadwalan *flowshop* dilibatkan sekolompok mesin $(M_1, M_2, \cdots, M_m)$ dan sejumlah pekerjaan yang tidak memiliki kaitan satu sama lain $(J_1, J_2, \cdots, J_n)$, dengan pertimbangan akan terbatasnya jumlah mesin dan pekerjaan. Setiap pekerjaan akan terdiri dari sekelompok proses dengan jumlah mesin yang sama dengan jumlah proses pekerjaan. Waktu yang dibutuhkan pada setiap proses dinotasikan dengan P_{ij} (P_{ij} menunjukkan waktu pemrosesan dari pekerjaan j pada mesin i). Proses dari setiap pekerjaan akan melalui urutan yang sama pada setiap mesin dimana setiap pekerjaan akan diproses terlebih dahulu pada M_1 kemudian M_2 hingga urutan terakhir, yaitu M_m (Zaied dkk, 2021). Tujuan utama adanya penjadwalan *flowshop* adalah untuk mengatur serta mengoptimalkan urutan pekerjaan pada setiap mesin dalam waktu penyelesaian minimal terbaik yang dikenal sebagai kriteria waktu penyelesaian total. Hasil penjadwalan yang diperoleh dinotasikan dengan $\pi = (\pi_1, \pi_2, \cdots, \pi_n)$.

Masalah penjadwalan *flowshop* telah lama menjadi topik penilitian yang terus dibahas oleh peneliti-peneliti terdahulu, hingga sekarang telah banyak variasi dari masalah penjadwalan *flowshop* atau alur produksi yang menarik untuk dibahas. seperti *Permutation Flowshop Problem* (PFSP) yang termasuk dalam variasi dari *pure flowshop*. Dalam PFSP semua proses pekerjaan harus mengikuti urutan yang sama pada semua mesin untuk menemukan penjadwalan terbaik dengan beberapa batasan dalam penyelesaian penjadwalan *flowshop*:

1. Setiap proses dimulai pada waktu nol,
2. Pekerjaan pertama dimulai pada waktu nol di mesin pertama,
3. Semua proses harus diselesaikan,
4. Hanya satu proses yang dioperasikan oleh satu mesin pada satu waktu,
5. Setiap proses dikerjakan pada mesin tertentu dengan durasi tertentu,
6. Sebuah mesin hanya dapat mengoperasikan satu proses pada satu waktu,

7. Tidak ada gangguan saat proses sedang dikerjakan,
8. Jika sebuah mesin sedang dalam proses operasi, proses lainnya harus menunggu dalam antrian,
9. Semua proses dari sebuah pekerjaan harus dioperasikan secara berurutan pada setiap mesin.

Untuk menghitung waktu penyelesaian dari permutasi pekerjaan pada setiap mesin (C_{i,π_j}) dapat digunakan Persamaan 2.1 – 2.4, sedangkan untuk menentukan waktu penyelesaian total maksimal atau makespan digunakan Persamaan 2.5 dengan p_{i,π_j} menotasikan waktu pemrosesan dari π_1 pada mesin i (Zaied dkk, 2021).

$$C_{1,\pi_1} = p_{1,\pi_1} \tag{2.1}$$

$$C_{1,\pi_j} = C_{1,\pi_{j-1}} + p_{1,\pi_j} \qquad j = 1, 2, \cdots, n \tag{2.2}$$

$$C_{i,\pi_1} = C_{i-1,\pi_1} + p_{i,\pi_1} \qquad i = 1, 2, \cdots, m \tag{2.3}$$

$$C_{i,\pi_j} = \max\left\{C_{i,\pi_{j-1}}, C_{i-1,\pi_j}\right\} + p_{i,\pi_j} \qquad \begin{matrix} i = 1, 2, \cdots, n \\ j = 1, 2, \cdots, m \end{matrix} \tag{2.4}$$

$$C_{max} = \max\left\{C_{m,\pi_j}\right\} \tag{2.5}$$

2.3 *Campbell Dudek Smith (CDS)*

Campbell Dudek Smith merupakan salah satu algortima *heuristic* yang berhasil dikembangkan oleh Cambell, Dudek dan Smith pada tahun 1965 dengan menggunakan aturan *Johnson*. Sebagai salah satu algortima *heuristic*, *Campbell Dudek Smith* dirancang untuk menyelesaikan masalah penjadwalan dengan lebih cepat dan mudah. Solusi yang diperoleh dari algoritma *heuristic* umumnya merupakan solusi yang mendekati solusi terbaik atau bahkan berhasil menemukan solusi akurat dengan cepat dan mudah (Desale dkk, 2015).

2.3.1 Algortima *Campbell Dudek Smith*

Sebagai algortima yang menggunakan dasar dari algortima *Johnson*, *Campbell Dudek Smith* atau CDS akan membagi waktu pemrosesan menjadi dua, yaitu a_j dan b_j dengan $m-1$ iterasi dan menotasikan waktu pemrosesan sebagai p_{kj} ($k = mesin$ & $j = pekerjaan$) dimana pada iterasi 1 nilai $a_j = p_{1j}$ dan $b_j =$

p_{mj}, kemudian waktu pemrosesan pada mesin pertama dan mesin terakhir untuk iterasi 2 dirumuskan dengan $a_j = p_{1j} + p_{2j}$ dan $b_j = p_{mj} + p_{m-1,j}$. Untuk waktu pemrosesan pada dua mesin pertama dan dua mesin terakhir untuk iterasi ke-i digunakan Persamaan 2.6.

$$a_j = \sum_{k=1}^{i} p_{kj}$$
$$b_j = \sum_{k=m-i+1}^{m} p_{kj} \quad (2.6)$$

Dengan:

j = job atau pekerjaan

k = mesin

m = banyaknya mesin

i = 1,2,3,…,($m-1$)

(Baker & Trietsch, 2019)

Tabel 2. 1 Iterasi Campbell Dudek Smith

i	a_j (Mesin Pertama)	b_j(Mesin Kedua)
1	p_{1j}	p_{mj}
2	$p_{1j} + p_{2j}$	$p_{mj} + p_{m-1,j}$
$\boldsymbol{m-1}$	$p_{1j} + p_{2j} + \cdots + p_{m-1,j}$	$p_{mj} + p_{m-1,j} + \cdots + p_{2j}$

Apabila semua tahap telah selesai, algoritma akan memilih urutan pekerjaan dengan waktu penyelesaian total terkecil menggunakan aturan *johnson* sebagai jadwal terbaik.

2.3.2 Aturan *Johnson*

Untuk memperoleh urutan pekerjaan, *Campbell Dudek Smith* menerapkan aturan *Johnson* pada waktu pemrosesan dari nilai a_j dan b_j (Mashuri Dkk 2019). Gambar 2.2 menunjukkan penggunaan aturan *Johnson* dalam pengurutan atau penjadwalan sejumlah pekerjaan.

Jobs\Machines	a_j	b_j
J_1	4	3
J_2	7	5
J_3	2	12
J_4	6	9
J_5	4	1
Job Sequence	$J_3 - J_4 - J_2 - J_1 - J_5$	

Gambar 2. 2 *Johnson's Rule*. Sumber: Data diolah 2024

Dalam aturan *Johnson* akan dilakukan pendataan waktu proses tiap pekerjaan pada tiap mesin. Selanjutnya, pekerjaan dengan waktu proses terkecil akan dijadwalkan terlebih dahulu apabila pekerjaan berada didalam kolom a_j, namun jika pekerjaan dengan waktu proses terkecil berada pada kolom b_j, maka pekerjaan akan dijadwakan dari belakang. Apabila terjadi kasus dimana terdapat 2 pekerjaan dengan nilai waktu proses yang sama, maka berlaku aturan bebas mengenai posisi jadwal antar kedua pekerjaan. Aturan akan dilanjutkan hingga semua pekerjaan telah dijadwalkan.

2.3.3 Contoh Penyelesaian dengan Algoritma CDS

Akan diselesaikan contoh permasalahan alur produksi dari suatu perusahaan yang memiliki 3 pekerjaan dan 3 mesin seperti pada Tabel 2.2. Perusahaan menginginkan jadwal yang dapat memberikan waktu penyelesaian total paling minimal. Akan ditunjukkan penjadwalan menggunakan perhitungan manual algoritma CDS. Dengan jumlah 3 mesin, maka akan terdapat 2 iterasi.

Tabel 2. 2 Contoh Soal Alur produksi

Jobs\Machines	M_1	M_2	M_3
J_1	3	3	7
J_2	6	5	3
J_3	6	7	4

Sumber: Data diolah 2024

1. Iterasi 1

Dengan Persamaan 2.6, diperoleh a_j dan b_j untuk iterasi 1 dengan mengambil processing time p_{1j} dan p_{3j} seperti yang terlihat pada Tabel 2.3:

Tabel 2. 3 Nilai a_j dan b_j untuk iterasi 1 PFSP

Jobs	a_j	b_j
J_1	3	7
J_2	6	3
J_3	6	4

Sumber: Data diolah 2024

Dengan menerapkan aturan *Johnson*, pekerjaan dengan waktu terkecil pada kolom a_j akan menjadi urutan pertama, sedangkan pekerjaan dengan waktu terkecil pada kolom b_j akan berada pada urutan terakhir. Dengan menerapkan aturan tersebut, diperoleh urutan 1-3-2 (Tabel 2.4). Selanjutnya dengan Persamaan 2.1, 2.2, 2.3 dan 2.4 akan ditunjukkan waktu penyelesaian dari setiap pekerjaan pada tiap mesin.

Tabel 2. 4 Permutation Job Iterasi 1 PFSP

π_i^1	M_1	M_2	M_3
π_1	3	3	7
π_2	6	7	4
π_3	6	5	3

Sumber: Data diolah 2024

$C_{1,\pi_1} = p_{1,\pi_1} = 3$

$C_{1,\pi_j} = C_{1,\pi_{j-1}} + p_{1,\pi_j}$

$C_{1,\pi_2} = C_{1,\pi_1} + p_{1,\pi_2} = 3 + 3 = 6$

$C_{1,\pi_3} = C_{1,\pi_2} + p_{1,\pi_3} = 6 + 7 = 13$

$C_{i,\pi_1} = C_{i-1,\pi_1} + p_{i,\pi_1}$

$C_{2,\pi_1} = C_{1,\pi_1} + p_{2,\pi_1} = 3 + 6 = 9$

$C_{3,\pi_1} = C_{2,\pi_1} + p_{3,\pi_1} = 9 + 6 = 15$

$C_{i,\pi_j} = \max\left\{C_{i,\pi_{j-1}}, C_{i-1,\pi_j}\right\} + p_{i,\pi_j}$

$C_{2,\pi_2} = \max\{C_{2,\pi_1}, C_{1,\pi_2}\} + p_{2,\pi_2} = \max\{9, 6\} + 7 = 16$

$C_{2,\pi_3} = \max\{C_{2,\pi_2}, C_{1,\pi_3}\} + p_{2,\pi_3} = \max\{16, 13\} + 4 = 20$

$C_{3,\pi_2} = \max\{C_{3,\pi_1}, C_{2,\pi_2}\} + p_{3,\pi_2} = \max\{15, 16\} + 5 = 21$

$C_{3,\pi_3} = \max\{C_{3,\pi_2}, C_{2,\pi_3}\} + p_{3,\pi_3} = \max\{21, 20\} + 3 = 24$

Digunakan Persamaan 2.5 yang menegaskan bahwa $C_{max} = \max\{C_{m,\pi_j}\}$ dengan kata lain $C_{max} = \max\{C_{3,\pi_3}\}$ sehingga waktu penyelesaian total dari urutan pekerjaan 1-3-2 adalah sebesar 24.

2. Iterasi 2

Dengan menggunakan Persamaan 2.6 diperoleh $a_j = p_{1j} + p_{2j}$ dan $b_j = p_{3j} + p_{2j}$, dengan hasil yang terdapat pada Tabel 2.5.

Tabel 2. 5 Nilai a_j dan b_j untuk iterasi 2 PFSP

Jobs	a_j	b_j
J_1	6	10
J_2	11	8
J_3	13	11

Sumber: Data diolah 2024

Dengan menerapkan aturan *Johnson*, diperoleh urutan 1-3-2, urutan yang sama seperti pada iterasi 1. Dengan persisnya urutan yang dihasilkan pada iterasi 1 dan iterasi 2, maka makespan terbaik yang diperoleh menggunakan algoritma CDS adalah sebesar 24.

2.4 *Particle Swarm Optimization*

Ide awal dari *Particle Swarm* yang dikembangkan oleh Kennedy dan Eberhart pada tahun 1995 lahir dengan tujuan untuk menghasilkan kecerdasan komputasi dengan analogi interaksi sosial. Simulasi yang dilakukan oleh Kennedy dan Eberhart terus dikembangkan menjadi metode optimisasi yang dikenal sebagai *Particle Swarm Optimization* (Poli dkk, 2007). PSO termasuk kedalam algoritma *meta-heuristic* yang mengacu pada metode komputasional untuk mengoptimalkan suatu masalah dengan memperbaiki solusi kandidat. Algoritma *meta-heuristic* tidak selalu menjamin dapat memperoleh solusi terbaik, namun algoritma ini mampu menjelajahi ruang solusi yang besar (Desale dkk, 2015).

2.4.1 Algoritma *Particle Swarm Optimization*

Dalam karyanya, Tasgetiren dkk (2004) telah mendefinisikan beberapa komponen dari algoritma *Particle Swarm Optimization* untuk *Permutation Flowshop Problem* sebagai:

1. Partikel dan Permutasi

   Partikel dalam PSO dinotasikan sebagai X_i^t yang menunjukkan partikel ke-i pada iterasi t, dan diwakili oleh n dimensi $\left(X_i^t = \left[x_{i1}^t, x_{i2}^t, \cdots, x_{i\rho}^t\right]\right)$ dengan x_{ij}^t sebagai nilai posisi partikel i, dimensi j pada iterasi t. Himpunan ρ partikel pada iterasi t disebut populasi dan dinotasikan sebagai pop^t $\left(pop^t = \left[X_1^t, X_2^t, \cdots, X_\rho^t\right]\right)$. Dalam PFSP banyaknya pekerjaan akan dipakai sebagai banyaknya dimensi, sedangkan banyaknya partikel yang digunakan adalah 2 kali jumlah pekerjaan. Dalam pencarian jadwal terbaik dengan waktu penyelesaian total minimal, akan diperoleh permutasi penugasan pekerjaan yang dapat dinotasikan sebagai π_i^t dan himpunan permutasi pekerjaan setiap partikel $\pi_i^t = \pi_{i1}^t, \pi_{i2}^t, \cdots, \pi_{in}^t$, dimana π_{ij}^t menunjukkan penugasan pekerjaan j dari partikel i dalam permutasi π_i^t pada iterasi t.

2. Personal Best

   Personal best (P_i^t) atau Pbest dalah representasi vektor posisi terbaik untuk partikel i berdasarkan nilai *fitness* terbaik yang pernah diperoleh hingga iterasi t. Pbest hanya akan diperbarui pada setiap iterasi apabila ditemui Pbest yang lebih baik dari Pbest sebelumnya. Dalam masalah minimisasi dengan dengan fungsi objektif $f(\pi_i^t \leftarrow X_i^t)$ dengan π_i^t merupakan permutasi dari partikel X_i^t, Pbest untuk partikel i dapat diperoleh apabila $f(\pi_i^t \leftarrow P_i^t) \leq f(\pi_i^{t-1} \leftarrow P_i^{t-1})$. Fungsi *fitness* dari Pbest dinyatakan dengan $f_i^p = f(\pi_i^t \leftarrow P_i^t)$. Untuk setiap partikel, *Personal best* dinyatakan dengan $P_i^t = [P_{i1}^t, P_{i2}^t, \cdots, P_{in}^t]$ untuk setiap partikel i, dengan P_{ij}^t sebagai nilai posisi utuk Pbest ke- i pada pekerjaan j iterasi ke-t.

3. Global Best

   Global best (G^t) atau Gbest merupakan partikel terbaik yang memuat kumpulan posisi terbaik dari seluruh *personal best* pada iterasi t. $Gbest$ dapat diperoleh dengan $f(\pi^t \leftarrow G^t) \leq f(\pi_i^t \leftarrow P_i^t)$ untuk $i = 1,2, \cdots, \rho$. Fungsi *fitness* pada

Gbest dapat dipresentasikan dengan $f^g = f(\pi^t \leftarrow G^t)$. *Global best* dapat ditulis sebagai $G^t = [g_1^t, g_2^t, \cdots, g_n^t]$ dengan g_j^t merupakan nilai posisi untuk partikel yang terpilih sebagai *global best* terhadap pekerjaan j pada iterasi t.

4. Pembaruan Beban Inersia

   Beban inersia (w^t) merupakan parameter yang memiliki peran sebagai pengontrol keseimbangan dari pengaruh kecepatan sebelumnya untuk kecepatan berikutnya. Beban inersia dapat diperbaiki dengan persamaan:

$$w^t = w^{t-1} \times \alpha \tag{2.7}$$

   Dalam karya Yuhui Shi dan Russel Eberhart (1998), diperoleh rentang ideal beban inersia antara 0.9 – 1.2 dimana perbaikan signifikan dapat dicapai dengan beban inersia yang berkurang pada tahap konvergensi algoritma. Selanjutnya, untuk ukuran dataset yang lebih besar Iqbal Hayat dkk (2023) menggunakan beban inersia sebesar 1.2 – 0.4 dengan *decrement factor* sebesar 0.975.

5. Pembaruan Kecepatan dan Posisi

   Velocity atau kecepatan (V_i^t) merupakan parameter yang digunakan untuk menetukan arah perpindahan partikel untuk memperbarui posisi sebelumnya, sehingga partikel dapat bergerak menuju ruang solusi yang lebih baik. $V_i^t = [v_{i1}^t, v_{i2}^t, \cdots, v_{in}^t]$, dengan v_{ij}^t merupakan kecepatan dari partikel i, dimensi j pada iterasi t. Kecepatan diperbarui pada tiap iterasi dengan memanfaatkan kecepatan, *personal best* dan *global best* pada iterasi sebelumnya menggunakan Persamaan 2.8.

$$v_{ij}^t = w^{t-1} v_{ij}^{t-1} + c_1 r_1 \left(p_{ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2 \left(g_j^{t-1} - x_{ij}^{t-1}\right) \tag{2.8}$$

   Dengan $c_1 \& c_2$ merupakan parameter kognitif dan parameter social yang berperan untuk mengontrol ukuran perpindahan maksimum yang dapat dilakukan paertikel (Tasgetitiren dkk, 2007), sedangkan $r_1 \& r_2$ merupakan bilangan acak antara (0,1). Sama seperti kecepatan, posisi akan diperbaiki dengan memperbarui nilai posisinya dengan menjumlahkan posisi sebelumnya dengan kecepatan yang telah diperbarui seperti yang diberikan pada Persamaan 2.9.

$$x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t \qquad i = 1,2,\cdots,n \tag{2.9}$$

   Dimana n menunjukkan banyaknya partikel.

Dalam pengaplikasiannya, *Particle Swarm Optimization* memiliki prosedur standar (Poli dkk, 2007):

1. Inisialisasi populasi dari partikel-partikel dengan posisi dan kecepatan secara random pada dimensi D dalam ruang penelusuran. Dalam penelitianya, Tasgetiren dkk (2007) melakukan inisialisasi posisi dengan $x_{min} = 0.0$ dan $x_{max} = 4.0$, sedangkan untuk kecepatan digunakan $v_{min} = -4.0$ dan $v_{max} = 4.0$. Hal ini telah dibahas dalam Poli (2007) yang menyatakan bahwa kecepatan dari PSO akan berada pada rentang $[-V_{max}, +V_{max}]$.
2. Evaluasi fungsi *fitness* optimisasi yang diinginkan dalam variabel D pada setiap partikel.
3. Membandingkan evaluasi *fitness* partikel dengan Pbestnya. Apabila nilai yang ada lebih baik dibandingkan dengan nilai Pbestnya, maka Pbest diset sama dengan nilai tersebut pada Pi sama dengan lokasi partikel yang ada xi dalam ruang dimensional D.
4. Identifikasi partikel dalam lingkungan dengan hasil terbaik sejauh ini, dan set sebagai Gbest.
5. *Update* kecepatan dan posisi partikel.
6. Kembali ke step dua sampai kriteria berhenti terpenuhi.

Terdapat beberapa kondisi yang dapat menjadi kriteria berhenti dari PSO, seperti jumlah iterasi maksimum atau evaluasi fungsi (FEs) telah terlampaui. Kondisi lain yang dapat menjadi kriteria berhenti adalah ketika tidak adanya peningkatan yang signifikan selama sejumlah iterasi. Sebagai contoh, proses dapat dianggap berhenti atau telah berakhir jika perubahan rata-rata posisi partikel sangat kecil atau kecepatan rata-rata partikel hampir 0 selama sejumlah iterasi (Engelbrecht, A. P. 2007).

2.4.2 Aturan *Smallest Position Value* (SPV)

Smallest Position Value atau aturan SPV merupakan aturan yang kerap kali digunakan bersama dengan *Particle Swarm Optimization*. Aturan SPV bekerja dengan mengurutkan nilai mulai dari yang terkecil hingga yang terbesar. Dalam *Particle Swarm Optimization*, nilai yang akan diurutkan menggunakan aturan SPV adalah nilai posisi partikel (x_{ij}^t).

Dimensi (j)	1	2	3	4	5
Posisi (X_{ij})	1.7	-0.9	2.1	-0.5	3
Job (π_{ij})	3	2	4	1	5

Gambar 2. 3 Aturan SPV. Sumber: Data diolah 2024

Pada contoh representasi aturan SPV di atas dapat terlihat bahwa -0.9 merupakan nilai posisi terkecil, sehingga *job* 2 akan dipindahkan ke urutan pertama, menggantikan *job* 1. Selanjutnya, *job* 4 dengan nilai posisi terkecil sesudah *job* 2 akan maju menjadi urutan berikutnya, hingga pada urutan terakhir, *job* 5 dengan nilai posisi sebesar 3.

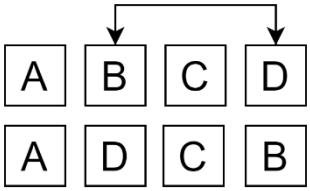
2.4.3 *Local Search*

Local Search atau pencarian lokal untuk optimisasi kombinatorial dilakukan dengan melakukan serangkaian perubahan lokal pada solusi awal. Perubahan ini bertujuan untuk meningkatkan nilai fungsi tujuan hingga mencapai optimum lokal (Mladenovic dan Hansen, 1997).

Salah satu metode *local search* yang kerap dijumpai untuk algoritma *Particle Swarm Optimization* dalam penyelesaian *Permutation Flowshop Problem* adalah *Variable Neighborhood Search* (VNS). *Variable Neighborhood Search* merupakan sebuah metode yang diperkenalkan untuk menunjukkan bahwa sebuah metode *meta-heuristic* sederhana dan efektif dapat diperoleh dengan melakukan perubahan sistematis pada *neighborhood* kedalam algoritma *local search*. VNS bekerja dengan menjelajahi lebih jauh *neighborhood* dari solusi saat ini, dan akan memperbarui solusi hanya jika bisa memberikan hasil yang lebih baik.

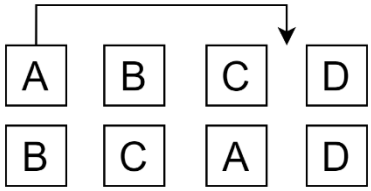
Local Search untuk PFSP diterapkan pada π^t dari solusi terbaik G^t pada setiap iterasi. Menurut Tasgetiren dkk (2007), kinerja dari *local search* bergantung pada operasi *neighborhood* yang digunakan. Dalam PFSP dua operasi *neighborhood* yang dapat digunakan adalah:

1. *Interchange*, menukar dua pekerjaan atau operator dari dimensi η dan κ dengan $\eta \neq \kappa$.



Gambar 2. 4 Operasi Interchange. Sumber: Data diolah 2024

2. *Insert*, memindahkan pekerjaan atau operator yang berada di dimensi η kemudian menyisipkannya pada dimensi κ dengan $\eta \neq \kappa$.



Gambar 2. 5 Operasi Insert. Sumber:Data diolah 2024

η dan κ merupakan bilangan bulat acak antara 1 dan n. Selanjutnya untuk memperoleh s akan dilakukan operasi *insert*, $s = insert(s_0, \eta, \kappa)$ atau memindahkan pekerjaan atau operator yang berada di dimensi η dari solusi s_0 kemudian menyisipkannya pada dimensi κ pada solusi s_0. Berdasarkan dari *pseudo-code* yang telah diberikan oleh Tasgetiren dkk (2007) diperoleh prosedur dari VNS sebagai berikut:

1. Mengasumsikan permutasi *job* atau π^t dari *global best* sebagai s_0.
2. Memilih η dan κ secara random pada integer $[1, n]$ dengan $\eta \neq \kappa$.
3. Memperoleh permutasi *job* baru untuk s dengan operasi *insert* pada permutasi job s_0 dengan syarat:
   a. Jika $\eta < \kappa$ maka pindahkan *job* yang berada di dimensi η kemudian sisipkan pada *job* yang berada di depan *job* dimensi κ.
   b. Jika $\eta > \kappa$ maka pindahkan *job* yang berada di dimensi η kemudian sisipkan pada *job* yang berada di belakang *job* dimensi κ.
4. Mengatur loop = 0, kcount = 0 dan maxmethod = 2.
5. Memilih η dan κ secara random untuk integer $[1, n]$ dengan $\eta \neq \kappa$. Selanjutnya lakukan operasi *neighborhood* dengan syarat:

a. Apabila kcount = 0 maka akan diperoleh permutasi *job* s_1 dengan operasi *insert* pada permutasi *job* s dengan syarat:
   - Jika $\eta < \kappa$ maka pindahkan *job* yang berada di dimensi η kemudian sisipkan pada *job* yang berada di depan *job* dimensi κ.
   - Jika $\eta > \kappa$ maka pindahkan *job* yang berada di dimensi η kemudian sisipkan pada *job* yang berada di belakang *job* dimensi κ.

b. Apabila kcount = 1 maka akan diperoleh permutasi *job* s_1 dengan operasi *interchange* pada permutasi *job* s, yaitu dengan menukar *job* dari dimensi η dengan *job* yang berada pada dimensi κ.

6. Melakukan evaluasi dari s_1 dan membandingkan nilai *fitness* dari s_1 $\left(f(s_1)\right)$ dengan nilai *fitness* dari s $\left(f(s)\right)$, dimana:
   a. Jika $f(s_1) < f(s)$ maka kcount = 0 dan s = s_1.
   b. Jika $f(s_1) > f(s)$ maka kcount = kcount+1.
7. Mengulangi poin 5 selama kcount < maxmethod, kemudian atur loop = loop+1 jika kcount $\geq$ maxmethod. Selanjutnya ulangi poin 4 hingga loop $\leq n * (n-1)$.
8. membandingkan nilai *fitness* dari s $\left(f(s)\right)$ dengan nilai *fitness* dari π^t $\left(f(\pi^t)\right)$, dimana:
   a. Jika $f(s) < f(\pi^t)$ maka *global best* akan diperbaiki dengan $\pi^t = s$.
   b. Jika $f(s) > f(\pi^t)$ maka $\pi^t = s_0$.

2.4.4 Contoh Penyelesaian PFSP dengan Algoritma PSO

Contoh permasalahan yang akan diambil adalah *permutation flowshop* dengan 3 pekerjaan dan 3 mesin, seperti pada Tabel 2.2. Pada contoh permasalahan yang akan diselesaikan, iterasi hanya akan dilakukan hingga iterasi satu, dengan prosedur PSO dan *Local Search*.

Nilai posisi dan kecepatan partikel akan dibangkitkan secara acak dengan rentang [-4.0,4.0] untuk kecepatan dan [0.0,4.0] untuk posisi. Jumlah partikel akan diambil sebanyak 2 kali jumlah pekerjaan berdasarkan saran yang telah diberikan oleh Tasgetiren pada tahun 2004 dan 2007. Nilai posisi dan kecepatan awal untuk setiap partikel i dan dimensi j ditunjukkan pada Tabel 2.6 dan 2.7.

Tabel 2. 6 Nilai Posisi PFSP

Partikel	Nilai Posisi		
	X_{i1}	X_{i2}	X_{i3}
X_1	0.25	3.02	3.30
X_2	1.05	3.60	2.25
X_3	3.66	1.37	2.61
X_4	0.45	2.87	0.28
X_5	1.13	2.39	2.21
X_6	3.10	3.22	1.05

Sumber: Data diolah 2024

Tabel 2. 7 Nilai Kecepatan PFSP

Partikel	Nilai Kecepatan		
	V_{i1}	V_{i2}	V_{i3}
V_1	2.49	-1.65	0.23
V_2	1.24	3.22	-2.46
V_3	0.36	-1.8	2.21
V_4	-3.08	2.46	-0.97
V_5	0.14	-2.01	1.9
V_6	-1.72	0.81	-3.12

Sumber: Data diolah 2024

Parameter yang akan digunakan untuk memperbarui kecepatan pada iterasi selanjutnya adalah sebesar 1.2 untuk beban inersia, $c_1 = c_2 = 2$, $r_1 = 0.2$ dan $r_2 = 0.4$.

Setelah memperoleh nilai posisi, berikutnya akan diperoleh permutasi pekerjaan dengan aturan SPV untuk semua partikel, seperti pada Tabel 2.8.

Tabel 2. 8 Permutasi Pekerjaan Iterasi 0 PFSP

Partikel (i)	Permutasi Pekerjaan (π_i^0)		
X_1	1	2	3
X_2	1	3	2

Partikel (i)	Permutasi Pekerjaan (π_i^0)		
X_3	3	1	2
X_4	2	3	1
X_5	1	3	2
X_6	2	3	1

Sumber: Data diolah 2024

Selanjutnya urutan pekerjaan yang telah diperoleh untuk tiap partikel dievaluasi menggunakan Persamaan 2.1, 2.2, 2.3, 2.4 dan 2.5. Sehingga akan diperoleh waktu penyelesaian total dari tiap partikel. Akan ditunjukkan penggunaan Persamaan 2.1 hingga 2.5 pada partikel 3 pada tabel 2.9.

Tabel 2. 9 Contoh Pencarian Waktu Penyelesaian dari PFSP

π_i^1	M1	M2	M3
π_1	6	7	4
π_2	6	3	7
π_3	3	5	3

Sumber: Data diolah 2024

Persamaan 2.1, 2.2, 2.3 dan 2.4 digunakan untuk menunjukkan waktu penyelesaian proses dari setiap pekerjaan pada tiap mesin.

$C_{1,\pi_1} = p_{1,\pi_1} = 6$

$C_{1,\pi_j} = C_{1,\pi_{j-1}} + p_{1,\pi_j}$

$C_{1,\pi_2} = C_{1,\pi_1} + p_{1,\pi_2} = 6 + 7 = 13$

$C_{1,\pi_3} = C_{1,\pi_2} + p_{1,\pi_3} = 13 + 4 = 17$

$C_{i,\pi_1} = C_{i-1,\pi_1} + p_{i,\pi_1}$

$C_{2,\pi_1} = C_{1,\pi_1} + p_{2,\pi_1} = 6 + 6 = 12$

$C_{3,\pi_1} = C_{2,\pi_1} + p_{3,\pi_1} = 12 + 3 = 15$

$C_{i,\pi_j} = \max\left\{C_{i,\pi_{j-1}}, C_{i-1,\pi_j}\right\} + p_{i,\pi_j}$

$C_{2,\pi_2} = \max\{C_{2,\pi_1}, C_{1,\pi_2}\} + p_{2,\pi_2} = \max\{12, 13\} + 3 = 16$

$C_{2,\pi_3} = \max\{C_{2,\pi_2}, C_{1,\pi_3}\} + p_{2,\pi_3} = \max\{16, 17\} + 7 = 24$

$C_{3,\pi_2} = \max\{C_{3,\pi_1}, C_{2,\pi_2}\} + p_{3,\pi_2} = \max\{15, 16\} + 5 = 21$

$C_{3,\pi_3} = \max\{C_{3,\pi_2}, C_{2,\pi_3}\} + p_{3,\pi_3} = \max\{21, 24\} + 3 = 27$

Selanjutnya untuk melihat waktu penyelesaian total dari partikel 3, akan digunakan Persamaan 2.5 yang telah menegaskan bahwa $C_{max} = \max\left\{C_{m,\pi_j}\right\}$ atau $C_{max} = \max\{C_{3,\pi_3}\}$ sehingga waktu penyelesaian total dari partikel 3 adalah 27. Dengan cara yang sama, akan diperoleh waktu penyelesaian total untuk setiap partikel yang dapat dilihat pada Tabel 2.10.

Tabel 2. 10 Waktu Penyelesaian Total Setiap Partikel

Partikel (i)	**Permutasi Pekerjaan** (π_i^0)			**Waktu Penyelesaian Total** (f_i^0)
X_1	1	2	3	26
X_2	1	3	2	24
X_3	3	1	2	27
X_4	2	3	1	30
X_5	1	3	2	24
X_6	2	3	1	30

Sumber: Data diolah 2024

Pada iterasi 0 posisi awal akan diasumsikan sebagai posisi-posisi terbaik dari tiap partikel dengan waktu penyelesaian total sebagai nilai *fitness.* Partikel dengan nilai *fitness* atau waktu peyelesaian total terkecil akan dianggap sebagai *global best.*

Tabel 2. 11 Pbest Setiap Partikel dan Gbest Iterasi 0

Partikel	**Personal Best**			**Fitness** (f_i^p)
	X_{i1}	X_{i2}	X_{i3}	
X_1	0.25	3.02	3.30	26
X_2	1.05	3.60	2.25	24
X_3	3.66	1.37	2.61	27
X_4	0.45	2.87	0.28	30
X_5	1.13	2.39	2.21	24
X_6	3.10	3.22	1.05	30

Sumber: Data diolah 2024

Dengan Persamaan 2.8 dan 2.9, kecepatan dan posisi tiap partikel akan diperbaiki untuk iterasi 1. Penggunaan Persamaan 2.8 dan 2.9 pada pembaruan v_{11}^{1} dan x_{11}^{1} akan ditunjukan sebagai berikut:

- Pembaharuan Kecepatan

  $v_{ij}^{t} = w^{t-1}v_{ij}^{t-1} + c_1 r_1\left(p_{ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2\left(g_j^{t-1} - x_{ij}^{t-1}\right)$

  $v_{11}^{1} = w^{0}v_{11}^{0} + c_1 r_1(p_{11}^{0} - x_{11}^{0}) + c_2 r_2(g_1^{0} - x_{11}^{0})$

  $v_{11}^{1} = 1.2 * (2.49) + 2 * 0.2(0.25 - 0.25) + 2 * 0.4(1.13 - 0.25)$

  $v_{11}^{1} = 3{,}692$

- Pembaharuan Posisi

  $x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}$

  $x_{11}^{1} = x_{11}^{0} + v_{11}^{1}$

  $x_{11}^{1} = 0.25 + (3.692)$

  $x_{11}^{1} = 3.942$

  Dengan cara yang sama, akan diperoleh pembaharuan kecepatan dan posisi untuk seluruh populasi seperti pada Tabel 2.12 dan 2.13.

Tabel 2. 12 Pembaharuan Nilai Kecepatan PFSP

Partikel	**Pembahruan Nilai Kecepatan**		
	$\boldsymbol{V_{i1}}$	$\boldsymbol{V_{i2}}$	$\boldsymbol{V_{i3}}$
$\boldsymbol{V_1}$	3.692	-2.484	-0.596
$\boldsymbol{V_2}$	1.552	2.896	-2.984
$\boldsymbol{V_3}$	-1.592	-1.344	2.332
$\boldsymbol{V_4}$	-3.152	2.568	0.38
$\boldsymbol{V_5}$	0.168	-2.412	2.28
$\boldsymbol{V_6}$	-3.64	0.308	-2.816

Sumber: Data diolah 2024

Tabel 2. 13 Pembaharuan Nilai Posisi PFSP

Partikel	**Pembaharuan Nilai Posisi**		
	$\boldsymbol{X_{i1}}$	$\boldsymbol{X_{i2}}$	$\boldsymbol{X_{i3}}$
$\boldsymbol{X_1}$	3.942	0.536	2.704

Partikel	Pembaharuan Nilai Posisi		
	X_{i1}	X_{i2}	X_{i3}
X_2	2.602	6.496	-0.734
X_3	2.068	0.026	4.942
X_4	-2.702	5.438	0.66
X_5	1.298	-0.022	4.49
X_6	-0.54	3.528	-1.766

Sumber: Data diolah 2024

Dengan menerapkan aturan SPV pada posisi yang telah diperbaiki, akan diperoleh urutan pekerjaan atau penjadwalan yang baru, dengan nilai waktu penyelesaian untuk tiap partikelnya dapat diperoleh dengan mengulang Persamaan 2.1 hingga 2.5. Dengan melihat Tabel 2.14, diketahui bahwa penjadwala terbaik diperoleh dengan urutan pekerjaan 1-3-2, yang memiliki nilai waktu penyelesaian total sebesar 24.

Tabel 2. 14 Permutasi Pekerjaan dan Waktu Penyelesaia Total Iterasi 1 PFSP

Particle (i)	Permutasi Job(π_i^1)			Makespan (f_i^1)
	X_{i1}	X_{i2}	X_{i3}	
X_1	3	1	2	27
X_2	2	3	1	30
X_3	2	1	3	26
X_4	1	3	2	24
X_5	2	1	3	26
X_6	2	3	1	30

Sumber: Data diolah 2024

Setelah diperoleh waktu penyelesaian total atau nilai *fitness* dari permutasi pekerjaan iterasi 1, akan dilakukan perbandingan nilai antara (f_i^1) dengan (f_i^0) untuk menghasilkan (f_i^p) yang baru, dengan melihat nilai yang lebih kecil seperti pada Tabel 2.15. Selanjutnya Pbest untuk iterasi 1 akan diperbaiki dengan melihat nilai fitness Pbest-nya. Sedangkan (f_i^p) dengan nilai terkecil akan dijadikan sebagai Gbest untuk iterasi 1 seperti pada Tabel 2.16.

Tabel 2. 15 Penentuan Nilai Fitness Baru

Partikel (i)	(f_i^p)	(f_i^1)	(f_i^p) **Baru**
1	26	27	26
2	24	30	24
3	27	26	26
4	30	24	24
5	24	26	24
6	30	30	30

Sumber: Data diolah 2024

Tabel 2. 16 Pbest Setiap Particle dan Gbest Iterasi 1

Partikel	**Pbest**			**Fitness**
	X_{i1}	X_{i2}	X_{i3}	(f_i^p)
X_1	0.25	3.02	3.30	26
X_2	1.05	3.60	2.25	24
X_3	2.068	0.026	4.942	26
X_4	-2.702	5.438	0.66	24
X_5	1.13	2.39	2.21	24
X_6	3.10	3.22	1.05	30

Sumber: Data diolah 2024

Melihat dari Gbest yang telah diperoleh pada iterasi 1, jadwal dengan waktu penyelesaian total terbaik diberikan oleh urutan pekerjaan 1-3-2 dengan waktu penyelesaian total sebesar 24.

Dengan diperoleh urutan pekerjaan dari *global best*, yaitu 1-3-2 maka akan diambil s_0 dengan urutan pekerjaan 1-3-2. Selanjutnya, dipilih nilai η dan κ dari integer $[1, n]$ dengan $\eta \neq \kappa$, misalkan dipilih $[1, n]$ dengan $\eta = 2$ dan $\kappa = 3$, akan dilakukan operasi *insert* dengan memindahkan *job* yang berada di dimensi 2, kemudian sisipkan pada *job* yang berada di depan *job* dimensi 3. Sehingga diperoleh permutasi *job* dari s dengan urutan pekerjaan 1-3-2. Dilakukan evaluasi waktu penyelesaian total dari permutasi *job* s, dimana diperoleh waktu

penyelesaian total untuk s adalah 24. Selanjutnya diatur loop = 0 dan kcount = 0 untuk melakukan operasi *neighborhood* dengan algoritma VNS:

Untuk loop = 0

1. kcount = 0

   dipilih $\eta = 1$ dan $\kappa = 3$

   $s_0 = insert(s, \eta, \kappa)$

   $s_1 = 3 - 1 - 2$

   Dengan urutan pekerjaan 3-1-2, maka waktu penyelesaian total dari s_1 adalah 27. Karena $f(s_1) > f(s)$ maka ketentuan yang berlaku adalah kcount = kcount+1.

2. Kcount = 1

   dipilih $\eta = 1$ dan $\kappa = 2$

   $s_0 = interchange(s, \eta, \kappa)$

   $s_1 = 1 - 3 - 2$

   Dengan urutan pekerjaan 1-3-2, maka waktu penyelesaian total dari s_1 adalah 24. Karena $f(s_1) = f(s)$ maka ketentuan yang berlaku adalah kcount = kcount+1. Sehingga loop akan dilanjutkan hingga loop $\leq n * (n - 1)$.

Apabila waktu penyelesaian total tidak memperoleh hasil yang lebih baik, maka urutan pekerjaan 1-3-2 dianggap sebagai urutan paling optimal dengan hasil sebesar 24.