Proceedings of the 5th Faculty of Industrial Technology
International Congress Bandung, Indonesia,
September 17-18, 2025
ISSN 2962 – 1798

Metaheuristic and Classical Approaches in Production Scheduling: Campbell Dudek Smith, Dannenbring, and Variable Neighborhood Descent

Hendro Prassetiyo [a,1], *Azizah Faudina* Desvarayanti [a], *and Arif Imran* [a]

[a] Departement of Industrial Engineering, Institut Teknologi Nasional, 40124, Bandung, Indonesia

Abstract. The flow shop scheduling problem is well known to be NP-hard when involving m machines and n jobs. To address such combinatorial optimization problems, both classical heuristics and metaheuristic algorithms are often employed to generate near-optimal solutions within a reasonable computational time. This study investigates the performance of two classical heuristic methods—Campbell Dudek Smith (CDS) and Dannenbring—alongside a metaheuristic approach, Variable Neighborhood Descent (VND), in solving the flow shop scheduling problem. The objective is to determine an effective job sequence that minimizes the total completion time (makespan). A set of benchmark case studies is utilized to evaluate the methods, and the resulting makespan values are compared to assess their efficiency. The findings highlight the trade-offs between classical and metaheuristic approaches, providing insights into their applicability for practical production scheduling problems.

1 Introduction

Flow-shop scheduling is a fundamental problem in operations research and production management. It involves sequencing n jobs across m machines to minimize objectives such as makespan. The problem is known to be NP-hard for $m > 2$, making exact methods impractical for large instances and encouraging the development of heuristic and metaheuristic approaches. Among classical heuristics, the Campbell–Dudek–Smith (CDS) method and Dannenbring heuristic are still widely respected for their simplicity and computational efficiency. They remain pertinent in industrial settings despite the ascendancy of more contemporary methods.

[1] Corresponding author: prasshendro@itenas.ac.id

ISSN 2962 – 1798

In recent years, metaheuristics have further advanced. For instance, a robust genetic algorithm incorporating Smith's convexity criterion and pyramid structure was shown to outperform NEH, pair-insert, and iterated local search methods in reducing makespan for ordered flow-shop problems [1]. Similarly, a hybrid algorithm using Differential Evolution and Variable Neighborhood Search (VNS) demonstrated superior makespan performance compared to standalone GA and DE in benchmark problems [2]. Additionally, deep reinforcement learning–enhanced GA has emerged, dynamically adjusting parent selection and mutation strategies to yield better scheduling solutions by adapting to population diversity over time [3].

An important member of the VNS family is Variable Neighborhood Descent (VND), a deterministic local search method that sequentially explores multiple neighborhood structures to improve solution quality and escape local optima [4]. VND and its variants have gained traction in scheduling contexts due to their systematic structure and effectiveness in refining solutions.

Despite these advances, few studies have conducted direct comparisons between classical heuristics (e.g., CDS and Dannenbring) and metaheuristics like VND under a unified framework, especially with real-world industrial datasets from emerging economies such as Indonesia. This study aims to fill that gap by comparing CDS, Dannenbring, and Variable Neighborhood Descent (VND)—a structured local search method known for escaping local optima through neighborhood switching—with real case data from a manufacturing facility in Bandung. The novelty lies in: (i) a head-to-head experimental comparison between classical and metaheuristic methods, (ii) the use of authentic industrial data rather than synthetic benchmarks, and (iii) analysis of trade-offs between makespan optimization and computational effort to guide practitioners in selecting the most suitable method for real-world production scheduling.

Accordingly, this paper investigates the effectiveness of classical heuristics (CDS and Dannenbring) and the metaheuristic VND in solving flow-shop scheduling problems. The performance of these methods is evaluated through case study data, where the makespan values obtained by each method are compared. The dataset was collected from a manufacturing industry located in Bandung, Indonesia, ensuring that the findings are grounded in practical industrial applications.

2 Methodology

2.1 Campbell Dudek Smith (CDS)

According to [5], the method proposed by Campbell, Dudek, and Smith (CDS) in 1965 is based on the rule introduced by Johnson, in which each job must pass through all machines in a sequence that yields the minimum makespan. The CDS method schedules n jobs on m machines by first constructing two hypothetical machines, where the processing times are

ISSN 2962 – 1798

defined as $t_{i,1} = t^{*}_{i,1}$ and $t^{*}_{i,2} = t_{i,m}$ representing the processing times on the first and last machines. For the second sequence, the processing times are defined as:

$$t^{*}_{i,1} = t_{i,1} + t_{i,2} \quad (1)$$

$$t^{*}_{i,2} = t_{i,m} + t_{i,m-1} \quad (2)$$

For the k sequence, the processing times on the two fictitious machines are calculated as follows:

$$t^{*}_{i,1} = \sum_{k=1}^{k} t_{i,k} \quad (3)$$

$$t^{*}_{i,2} = \sum_{k=1}^{k} t_{i,m-k+1} \quad (4)$$

In this study, the calculation is carried out under three conditions:

1. CDS based on processing times without considering parallel machines, the makespan calculation is based on processing time data, without considering parallel machines.
2. CDS based on processing times with consideration of parallel machines, the makespan calculation uses processing time data with the job sequence obtained from Condition 1, but the parallel machines have been considered
3. CDS based on equivalence, the makespan calculation is based on equivalence time data.

2.2 Dannenbring

The Dannenbring method was developed by D. G. Dannenbring. This method employs a procedure known as Rapid Access, which essentially combines the CDS method with Palmer's slope index concept. The processing time calculation for this algorithm is defined as follows:

$$P_{i1} = \sum_{j=1}^{M} (M - j + 1) \bullet t_{ij} \quad (5)$$

$$P_{i2} = \sum_{j=1}^{M} (j) \bullet t_{ij} \quad (6)$$

For i = 1, 2, …., n:

Where P_{i1} = processing time of job i on the first fictitious machine

P_{i2} = processing time of job i on the second fictitious machine

j = machine-j.

In this study, the calculation is carried out under three conditions:

1. CDS based on processing times without considering parallel machines, the makespan calculation is based on processing time data, without considering parallel machines.
2. CDS based on processing times with consideration of parallel machines, the makespan calculation uses processing time data with the job sequence obtained from Condition 1, but the parallel machines have been considered

ISSN 2962 – 1798

3. CDS based on equivalence, the makespan calculation is based on equivalence time data.

2.3 Variable Neighborhood Descent (VND)

Variable Neighborhood Descent (VND) is an extension of the Variable Neighborhood Search (VNS) method. VNS is a metaheuristic approach designed to solve complex optimization problems and is based on systematically changing the neighborhood structure [6]. In this study, the calculation is carried out under four conditions: (1) Initial solution based on the best CDS result, (2) Initial solution based on the best Dannenbring result, and (3) Initial solution based on a random job sequence.

2.4 Threshold Accepting (TA)

Threshold Accepting (TA) is a combinatorial optimization algorithm developed by [7]. This method is an alternative search approach aimed at avoiding entrapment in local minima by exploring new solutions within a certain tolerance limit. In this algorithm, a new solution is accepted only if the difference between the new solution value and the current solution value is within a predefined threshold. Thus, TA allows the acceptance of solutions that may not be better but are still within the acceptable range, thereby helping the search process escape from local optima [7].

3 Result and Discussion

3.1 Data Collection

The data used are as follows:

a. Production Process Sequence Data

The production process sequence can be seen in Fig. 1, Fig. 2, and Fig. 3.

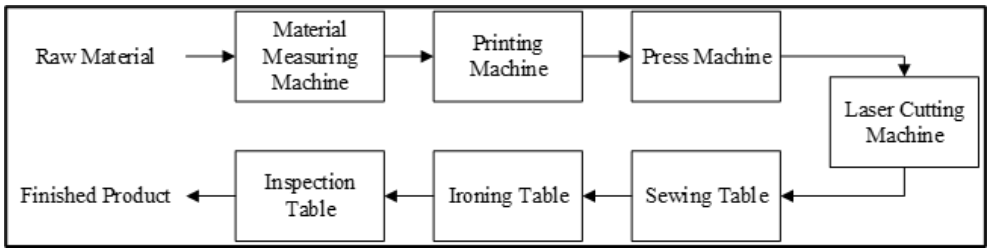

Fig. 1. Production Process Sequence of Hijab with Plate (Hijab A)

ISSN 2962 – 1798

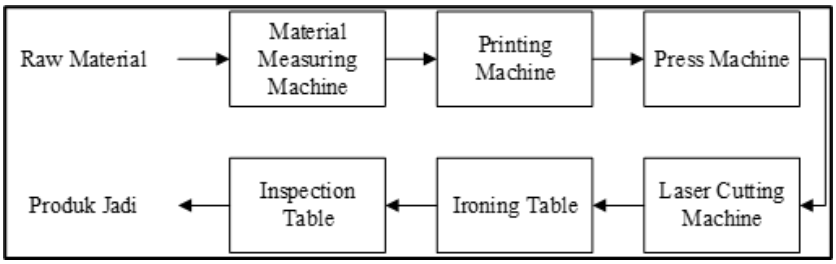

Fig. 2. Production Process Sequence of Hijab without Plate (Hijab B)

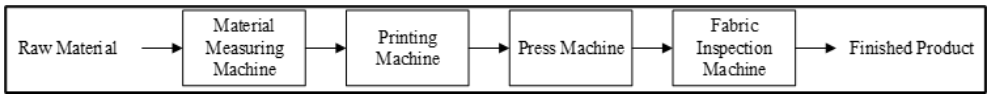

Fig. 3. Production Process Sequence of Seamless Print

b. Order Quantity Data for March 2025

The order quantity data can be seen in Table 1.

Table 1. Order Quantity Data

Job (*i*)	Product	Qty	Unit	Due Date
1	Hijab A	40	Pcs	18/03/2025
2	Hijab A	40	Pcs	19/03/2025
3	Hijab B	125	Pcs	19/03/2025
4	*Seamless Print*	96	Meter	19/03/2025
5	Hijab A	115	Pcs	17/03/2025
6	Hijab B	39	Pcs	19/03/2025
7	*Seamless Print*	120	Meter	19/03/2025
8	Hijab B	40	Pcs	19/03/2025
9	Hijab A	50	Pcs	19/03/2025
10	*Seamless Print*	48	Meter	19/03/2025
11	*Seamless Print*	72	Meter	18/03/2025
12	Hijab B	96	Pcs	18/03/2025
13	Hijab A	85	Pcs	17/03/2025
14	Hijab B	108	Pcs	18/03/2025
15	*Seamless Print*	94	Meter	18/03/2025
16	*Seamless Print*	240	Meter	17/03/2025
17	*Seamless Print*	105	Meter	19/03/2025
18	*Seamless Print*	190	Meter	18/03/2025
19	Hijab A	50	Pcs	19/03/2025
20	*Seamless Print*	63	Meter	19/03/2025
21	*Seamless Print*	63	Meter	18/03/2025
22	*Seamless Print*	63	Meter	19/03/2025
23	*Seamless Print*	101	Meter	19/03/2025
24	Hijab B	106	Pcs	18/03/2025
25	Hijab B	140	Pcs	18/03/2025

c. Machine Quantity Data

The machine quantity data can be seen in Table 2.

Table 2. Machine Quantity Data

Machine Code	Machine Name	Quantity
A	Material Measuring Machine	1
B	Printing Machine Ts 300	1
C	Printing Machine Atexco	1
D	Press Machine	2
E	Laser Cutting Machine	1

ISSN 2962 – 1798

F	Sewing Table	1
G	Ironing Table	1
H	Inspection Table	1
I	Fabric Inspection Machine	1

d. Processing Time Data

The processing time data for each machine can be seen in Table 3.

Table 3. Processing Time Data

Job (*i*)	Processing Time (Minutes)								
	Machine								
	A	B	C	D	E	F	G	H	I
1	96,54	97,94		34,35	54,84	40,48	93,13	26,09	
2	96,14	97,54		33,55	52,44	35,68	91,75	15,05	
3	286,94		112,62	90,90	141,14		292,48	63,58	
4	199,90		78,68	58,99					181,71
5	263,39	262,54		85,35	129,79	111,28	279,72	46,62	
6	93,52		48,59	32,89	53,42		44,62	16,92	
7	246,94		93,32	73,15					238,20
8	96,54		48,92	33,55	50,84		48,19	21,93	
9	118,44	120,04		40,65	63,64	48,38	55,87	21,03	
10	102,46		50,36	33,07					98,19
11	149,50		66,20	45,31					144,78
12	220,06		90,20	69,55	106,76		102,91	37,61	
13	196,49		82,17	64,95	96,94	80,78	100,29	54,76	
14	246,70		97,88	79,51	124,88		119,90	59,95	
15	187,04		78,40	58,85					185,51
16	479,74		176,12	141,55					466,35
17	214,42		87,32	63,85					205,80
18	386,94		136,02	111,65					377,45

Job (*i*)	Processing Time (Minutes) Machine								
	A	B	C	D	E	F	G	H	I
19	118,94	120,54		40,15	60,14	43,88	57,02	27,93	
20	132,69		58,55	41,17					122,22
21	128,28		59,81	41,80					126,49
22	127,90		59,81	41,17					126,00
23	206,52		84,76	63,71					199,53
24	241,20		98,56	76,05	121,76		125,81	47,10	
25	317,74		122,32	102,35	159,24		162,42	80,37	

General formula for Processing Time (PT):

PT_{Hijab} = *Setup Time* + *((Operation Time/meter* × *Hijab Size)* × *Order Quantity)* (7)

$PT_{Seamless\ Print}$ = *Setup Time* + *(Operation Time/meter* × *Order Quantity)* (8)

Because some machines have more than one unit, it is necessary to compute equivalence times to distribute jobs across all units of those machines. The equivalence time data are presented in Table 4.

Table 4. Equivalence Time Data

Job (*i*)	Equivalence Time (Minutes) Machine								
	A	B	C	D	E	F	G	H	I
1	96,54	97,94		17,18	54,84	40,48	93,13	26,09	
2	96,14	97,54		16,78	52,44	35,68	91,75	15,05	
3	286,94		112,62	45,45	141,14		292,48	63,58	
4	199,90		78,68	29,50					181,71
5	263,39	262,54		42,68	129,79	111,28	279,72	46,62	
6	93,52		48,59	16,45	53,42		44,62	16,92	
7	246,94		93,32	36,58					238,20
8	96,54		48,92	16,78	50,84		48,19	21,93	
9	118,44	120,04		20,33	63,64	48,38	55,87	21,03	
10	102,46		50,36	16,54					98,19
11	149,50		66,20	22,66					144,78

ISSN 2962 – 1798

Job (*i*)	Equivalence Time (Minutes)								
	Machine								
	A	B	C	D	E	F	G	H	I
12	220,06		90,20	34,78	106,76		102,91	37,61	
13	196,49		82,17	32,48	96,94	80,78	100,29	54,76	
14	246,70		97,88	39,76	124,88		119,90	59,95	
15	187,04		78,40	29,43					185,51
16	479,74		176,12	70,78					466,35
17	214,42		87,32	31,93					205,80
18	386,94		136,02	55,83					377,45
19	118,94	120,54		20,08	60,14	43,88	57,02	27,93	
20	132,69		58,55	20,59					122,22
21	128,28		59,81	20,90					126,49
22	127,90		59,81	20,59					126,00
23	206,52		84,76	31,86					199,53
24	241,20		98,56	38,03	121,76		125,81	47,10	
25	317,74		122,32	102,35	159,24		162,42	80,37	

General formula for Equivalence Time:

$$Equivalence\ Time = \frac{Processing\ Time}{Quantity\ Machine} \quad (10)$$

.

3.2 Calculation of Makespan Using the Campbell Dudek Smith (CDS) Method

Based on the calculation results using the CDS method, eight alternatives were obtained for each condition. The recap of the best alternative for each condition can be seen in Table 8. The calculation results show that the smallest makespan is obtained in Condition 3, which is based on equivalence time, where the processing time is distributed according to the number of available parallel machines. In this condition, three alternative job sequences were selected (highlighted in Table 8). The best total makespan obtained is 5,114.31 minutes, equivalent to 5.33 days. These results indicate that balancing the workload among parallel machines is able to reduce makespan.

ISSN 2962 – 1798

Table 8. The Recap of the Best Job Sequences and Makespan for Each Condition Using CDS Method

Condi-tions	Alt.	Job Sequences	Makespan	
			Minutes	Days
1	4	16 – 5 – 18 – 3 – 7 – 13 – 17 – 25 – 23 – 15 – 4 – 14 – 24 – 11 – 1 – 2 – 12 - 21 - 22 – 20 – 19 – 9 – 10 – 8 – 6	5130,34	5,34
	7	16 – 3 – 18 – 5 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 15 – 4 – 11 – 1 – 21 – 22 – 20 – 9 – 19 – 2 – 10 – 8 – 6	5130,34	5,34
	8	5 – 16 – 3 – 18 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 9 – 19 – 1 – 15 – 4 – 2 – 11 – 21 – 22 – 20 – 10 – 8 – 6	5130,34	5,34
2	4	16 – 5 – 18 – 3 – 7 – 13 – 17 – 25 – 23 – 15 – 4 – 14 – 24 – 11 – 1 – 2 – 12 – 21 – 22 – 20 – 19 – 9 – 10 - 8 – 6	5130,34	5,34
	7	16 – 3 – 18 – 5 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 15 – 4 – 11 – 1 – 21 – 22 – 20 – 9 – 19 – 2 – 10 – 8 – 6	5130,34	5,34
	8	5 – 16 – 3 – 18 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 9 – 19 – 1 – 15 – 4 – 2 – 11 – 21 – 22 – 20 – 10 – 8 – 6	5130,34	5,34
3	4	16 – 5 – 18 – 3 – 7 – 13 – 17 – 25 – 23 – 15 – 4 – 14 – 24 – 11 – 1 – 2 – 12 – 21 – 22 – 20 – 19 – 9 – 10 - 8 – 6	5114,31	5,33
	7	16 – 3 – 18 – 5 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 15 – 4 – 11 – 1 – 21 – 22 – 20 – 9 – 19 – 2 – 10 – 8 – 6	5114,31	5,33
	8	5 – 16 – 3 – 18 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 9 – 19 – 1 – 2 – 15 – 4 – 11 – 21 – 22 – 20 – 10 – 8 – 6	5114,31	5,33

3.3 Calculation of Makespan Using the Dannenbring Method

The recap of the job sequence results and makespan for each condition using the Dannenbring method can be seen in Table 9. The calculation results show that the smallest makespan is obtained in Condition 3, which is based on equivalence time, where the processing time is distributed according to the number of available parallel machines. The selected job sequence is highlighted in red font in Table 9. The best total makespan obtained is 5,114.31 minutes, equivalent to 5.33 days. These results indicate that balancing the workload among parallel machines is able to reduce makespan.

Table 9. The Recap of the Best Job Sequences and Makespan for Each Condition Using Dannenbring Method

ISSN 2962 – 1798

Condi-tions	Job Sequences	Makespan	
		Minutes	Hari
1	16 – 18 – 5 – 3 – 25 – 7 – 14 – 17 – 13 – 23 – 24 – 15 – 4 – 12 – 11 – 1 – 19 – 9 – 21 – 22 – 20 – 2 – 10 – 8 – 6	5130,34	5,34
2	16 – 18 – 5 – 3 – 25 – 7 – 14 – 17 – 13 – 23 – 24 – 15 – 4 – 12 – 11 – 1 – 19 – 9 – 21 – 22 – 20 – 2 – 10 – 8 – 6	5130,34	5,34
3	16 – 18 – 5 – 3 – 25 – 7 – 17 – 13 – 14 – 23 – 24 – 15 – 4 – 12 – 11 – 1 – 19 – 9 – 21 – 22 – 2 – 20 – 10 – 8 – 6	5114,31	5,33

3.4 Calculation of Makespan Using the VND Method with Threshold

In this calculation, the initial solutions will be based on the best job sequence CDS result, the best job sequence Dannenbring result, and a random job sequence. The initial solutions obtained from the CDS and Dannenbring methods will be selected based on the smallest makespan value that has been previously calculated (Table 8 and Table 9). The initial random solution is generated using Python. The following notations will be used in the VND method:

i = job index (i = 1, 2, 3, …, n)
l = machine index (l = A, B, C, …, m)
t_i = processing time of job i
M = total processing time (makespan)
M' = makespan of the new solution after movement
M_{Th} = makespan threshold
T = threshold (1%)
S = job sequence
S' = new solution after movement
N_k = neighborhood structure (1, 2, …, k)
HN_1 = neighborhood structure list 1
HN_2 = neighborhood structure list 2
N_i = position of the job to be moved
N_j = new position where the job is inserted

The steps of the VND method are as follows:

Initial Solution

Step 1 : Input the processing time data for each job on each machine.
Step 2 : Schedule S as the initial solution using CDS, Dannenbring, and random, then save it as the current solution (CS).
Step 3 : Calculate the makespan value of the job sequence.

Local Search

Step 4 : Calculate the makespan threshold
$M_{Th} = CS * (1 + T)$
Step 5 : Initialize neighborhood list (N_1 = insertion, N_2 = 2-opt)
Step 6 : Set $k = 1$
Step 7 : Randomly select job i to be removed from N_i

Step 8 : Randomly select N_j as the new position to insert job i with $j \neq i$
Step 9 : Remove job i from N_i and insert it into N_j, then save into HN_1
Step 10: Repeat Step 6 to Step 8 for all possible S'
Step 11: Calculate M' of S' and select the best solution
Step 12: Check if M' < M_{Th}
If Yes: Set CS = S' and M = M', then return to Step 6
If No: Set $k = k+1$ and continue to the next neighborhood (Step 13)
Step 13: Select two random positions i and j with $j \geq i + 2$
Step 14: Reverse the order of the segment $(i + 1: j + 1)$ and save into HN_2
$(i + 1: j + 1)$ = Reversed $(i + 1: j + 1)$
Step 15: Repeat Step 11 to Step 14 for all possible S'
Step 16: Calculate M' of S' and select the best solution
Step 17: Check if M' < M_{Th}
If Yes: Set CS = S' and M = M', then return to Step 6
If No: proceed to Step 18
Step 18: Stop and display the best solution S and the best makespan M.
The recap of the makespan values using the VND method can be seen in Table 10.

Table 10. The Recap of Job Sequences and Makespan for Initial Solutions Using VND Method

Initial Solutions	Alt	Initial Job Sequences	*Makespan*		Final Job Sequences	*Makespan*	
			Minutes	Days		Minutes	Days
The Best CDS	4	16 – 5 – 18 – 3 – 7 – 13 – 17 – 25 – 23 – 15 – 4 – 14 – 24 – 11 – 1 – 2 – 12 – 21 – 22 – 20 – 19 – 9 – 10 – 8 – 6	5114,31	5,33	16 – 5 – 18 – 3 – 7 – 13 – 17 – 25 – 23 – 15 – 4 – 14 – 24 – 11 – 1 – 2 – 12 – 21 – 22 – 20 – 19 – 9 – 10 – 8 – 6	5114,31	5,33
	7	16 – 3 – 18 – 5 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 15 – 4 – 11 – 1 – 21 – 22 – 20 – 9 – 19 – 2 – 10 – 8 – 6	5114,31	5,33	16 – 3 – 18 – 5 – 25 – 14 – 13 – 24 – 7 – 12 – 17 – 23 – 15 – 4 – 11 – 1 – 21 – 22 – 20 – 9 – 19 – 2 – 10 – 8 – 6	5114,31	5,33
	8	5 - 16 - 3 - 18 - 25 - 14 - 13 - 24 - 7 - 12 - 17 - 23 - 9 - 19 – 1 – 2 – 15 – 4 – 11 – 21 – 22 – 20 – 10 – 8 – 6	5114,31	5,33	5 - 16 - 3 - 18 - 25 - 14 - 13 - 24 - 7 - 12 - 17 - 23 - 9 - 19 – 1 – 2 – 15 – 4 – 11 – 21 – 22 – 20 – 10 – 8 – 6	5114,31	5,33
The Best Dannenbring	1	16 - 18 - 5 - 3 - 25 - 7 - 17 - 13 - 14 - 23 - 24 - 15 - 4 - 12 - 11 - 1 - 19 - 9 - 21 - 22 - 2 - 20 - 10 - 8 - 6	5114,31	5,33	16 - 18 - 5 - 3 - 25 - 7 - 17 – 13 – 14 – 23 – 24 – 15 – 4 – 12 – 11 – 1 – 19 – 9 – 21 – 22 – 2 – 20 – 10 – 8 – 6	5114,31	5,33
Random Job Sequences	1	10 – 24 – 21 – 1 – 20 – 13 – 12 – 5 – 22 – 7 – 15 – 6 – 19 – 16 – 8 – 18 – 9 – 23 – 4 – 3 – 14 – 11 – 2 – 17 – 25	5478,52	5,71	10 – 24 – 21 – 1 – 20 – 13 – 12 – 5 – 22 – 7 – 15 – 19 – 16 – 18 – 9 – 23 – 4 – 3 – 25 – 14 – 11 – 2 – 17 – 8 – 6	5114,31	5,33
Company's Job Sequences	1	1 – 2 – 3 – 4 – 6 – 8 – 10 – 11 – 12 – 9 – 7 – 15 – 5 – 14 – 17 – 13 – 16 – 18 – 20 – 23 – 24 – 21 – 19 – 25 – 22	5401,80	5,62	1 – 2 – 3 – 4 – 10 – 11 – 12 – 9 – 7 – 15 – 5 – 14 – 17 – 13 – 16 – 25 – 18 – 20 – 23 – 24 – 21 – 19 – 22 – 8 – 6	5114,31	5,33

3.5 Calculation of Flow Time

The makespan results obtained from the three methods are identical, as presented in Table 8, Table 9, and Table 10. To further determine the best proposed schedule, flow time is used as an additional criterion, with the formulas defined as follows:

$$Flow\ Time\ Job\ (i) = Completion\ Time\ of\ Job(i) - Starting\ Time\ of\ Job\ (i) \quad (11)$$

$$Mean\ Flow\ Time = \frac{Total\ Flow\ Time}{Number\ of\ Job} \quad (12)$$

The recap of the flow time and mean flow time calculations can be seen in Table 11. Based on the table, it can be observed that the same makespan value may result in different mean flow time values. The smallest mean flow time is obtained from the CDS method alternative 4; therefore, this job sequence is proposed as the best schedule. The Gantt Chart of the selected schedule is presented in Figure 5. Furthermore, as shown in Table 12, the selected job sequence does not result in any delays.

Table 11. The Recap of Total Flow Time and Mean Flow Time

Methods	Alt	Total Flow Time (Minutes)	Mean Flow Time (Minutes)
CDS	4	13945,00	557,80
	7	14303,42	572,14
	8	14341,62	573,66
Dannenbring	1	14452,29	578,09
VND	Random Job Sequence	14770,49	590,82

Table 12. Job Scheduling Using the CDS Method Alternative 4 (Condition 3)

Job (*i*)	PO Date	Production Date	*Due Date*	Completion Date	Remark
16	10/03/2025	13/03/2025	17/03/2025	14/03/2025	On Time
5	08/03/2025	13/03/2025	17/03/2025	14/03/2025	On Time
18	11/03/2025	13/03/2025	18/03/2025	14/03/2025	On Time
3	07/03/2025	14/03/2025	19/03/2025	15/03/2025	On Time
7	08/03/2025	14/03/2025	19/03/2025	15/03/2025	On Time
13	10/03/2025	14/03/2025	17/03/2025	15/03/2025	On Time
17	11/03/2025	14/03/2025	19/03/2025	15/03/2025	On Time
25	13/03/2025	15/03/2025	18/03/2025	15/03/2025	On Time
23	13/03/2025	15/03/2025	19/03/2025	15/03/2025	On Time
15	10/03/2025	15/03/2025	18/03/2025	17/03/2025	On Time

ISSN 2962 – 1798

Job (*i*)	PO Date	Production Date	*Due Date*	Completion Date	Remark
4	07/03/2025	15/03/2025	19/03/2025	17/03/2025	On Time
14	10/03/2025	17/03/2025	18/03/2025	17/03/2025	On Time
24	13/03/2025	17/03/2025	18/03/2025	17/03/2025	On Time
11	10/03/2025	17/03/2025	18/03/2025	17/03/2025	On Time
1	07/03/2025	17/03/2025	18/03/2025	18/03/2025	On Time
2	07/03/2025	17/03/2025	19/03/2025	18/03/2025	On Time
12	10/03/2025	17/03/2025	18/03/2025	18/03/2025	On Time
21	13/03/2025	18/03/2025	18/03/2025	18/03/2025	On Time
22	13/03/2025	18/03/2025	19/03/2025	18/03/2025	On Time
20	12/03/2025	18/03/2025	19/03/2025	18/03/2025	On Time
19	12/03/2025	18/03/2025	19/03/2025	19/03/2025	On Time
9	08/03/2025	18/03/2025	19/03/2025	19/03/2025	On Time
10	08/03/2025	18/03/2025	19/03/2025	19/03/2025	On Time
8	08/03/2025	18/03/2025	19/03/2025	19/03/2025	On Time
6	08/03/2025	19/03/2025	19/03/2025	19/03/2025	On Time

ISSN 2962 – 1798

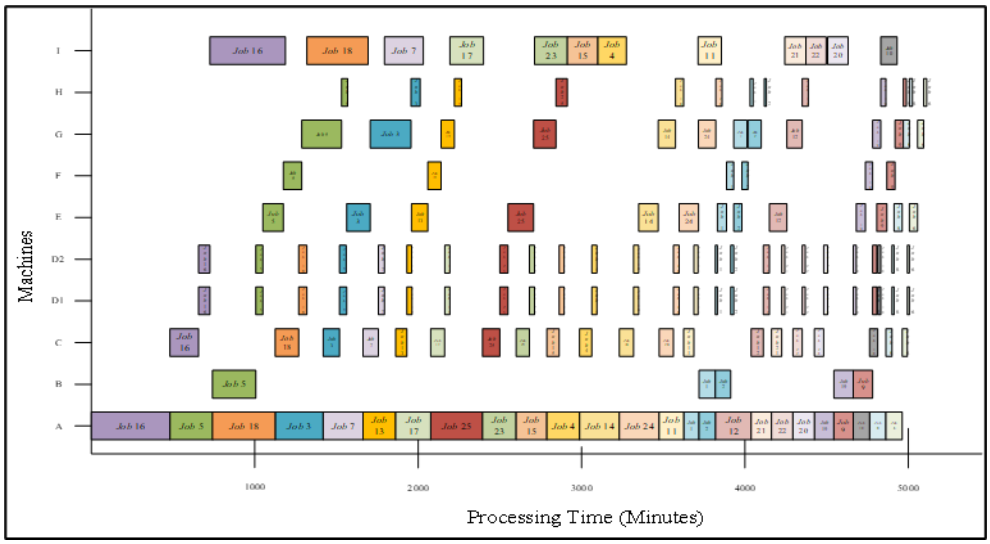

Fig. 5. Gantt Chart of the CDS Alternative 4 (Condition 3)

3.6 Analysis

The results show that the scheduling outcomes obtained from the three methods are identical due to the problem size being not particularly large and the low variability of processing times across machines and jobs. This indicates that the job sequences tend to be quite similar. The exploration order and number of neighborhoods in VND may affect solution quality [8]. However, in this case, additional neighborhood structures did not significantly improve results, consistent with [9], who emphasized that the effectiveness of VND strongly depends on problem characteristics. From the calculations, the CDS method (alternative 4) achieved the best performance with a makespan of 5114.31 minutes (5.33 days) and the smallest mean flow time of 557.80 minutes. Conversely, by applying the CDS method, scheduling considers the processing times of each job comprehensively, thereby minimizing both makespan and mean flow time.

4 Conclusion

Based on the research results, all three methods yield the same makespan value of 5114.31 minutes (5.33 days). To further distinguish the best sequence, mean flow time was calculated as an additional criterion. The CDS method produced a smaller mean flow time (577.80 minutes) compared to Dannenbring, and VND. Using the proposed CDS alternative 4 resulted in a 5.32% reduction in makespan and a 16.77% reduction in mean flow time.

References

[1] Y. Zhu, X. Wu, and C. Zhang, "A robust genetic algorithm with convexity criterion for ordered flow shop scheduling," Processes, vol. 11, no. 5, p. 1583, 2023, doi: 10.3390/pr11051583.

[2] R. Kumar, A. Singh, and P. Gupta, "Hybrid differential evolution and variable neighborhood search for permutation flow shop scheduling," Applied Mathematics and Computation, vol. 421, p. 126957, 2022, doi: 10.1016/j.amc.2022.126957.

[3] H. Liu, J. Chen, and X. Li, "A deep reinforcement learning–enhanced genetic algorithm for complex scheduling problems," Expert Systems with Applications, vol. 223, p. 119965, 2023, doi: 10.1016/j.eswa.2023.119965.

[4] J. de Armas, & J. A. Moreno-Pérez, (2025). A Survey on Variable Neighborhood Search for Sustainable Logistics. *Algorithms*, *18*(1), 38. https://doi.org/10.3390/a18010038

[5] D. D. Bedworth and J. E. Bailey, Integrated Production Control Systems: Management, Analysis, Design. New York, NY, USA: John Wiley & Sons, 1987.

[6] A. Duarte, N. Mladenovic, J. Sanchez-Oro, and R. Todosijevic, "Handbook of Heuristics," in *Handbook of Heuristics*, 2016, pp. 1–27. doi: 10.1007/978-3-319-07153-4.

[7] G. Dueck and T. Scheuer, "Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing," *J. Comput. Phys.*, vol. 90, no. 1, pp. 161–175, 1990, doi: 10.1016/0021-9991(90)90201-B.

[8] R. O. M. Diana and S. R. de Souza, "Analysis of variable neighborhood descent as a local search operator for total weighted tardiness problem on unrelated parallel machines," *Comput. Oper. Research*, vol. 117, no. 5, pp. 1–15, 2020, [Online]. Available: http://dx.doi.org/10.1016/j.bpj.2015.06.056%0Ahttps://academic.oup.com/bioinformatics/article-abstract/34/13/2201/4852827%0Ainternal-pdf://semisupervised-3254828305/semisupervised.ppt%0Ahttp://dx.doi.org/10.1016/j.str.2013.02.005%0Ahttp://dx.doi.org/10.10

[9] A. Lamghari, R. Dimitrakopoulos, and J. A. Ferland, "A variable neighbourhood descent algorithm for the open-pit mine production scheduling problem with metal uncertainty," *J. Oper. Res. Soc.*, vol. 65, no. 9, pp. 1305–1314, 2014, doi: 10.1057/jors.2013.81.