A sorting based heuristic for the Job Scheduling Problem.

Akshay Khandelwal, Grace Matson, Omkar Chaudhari

***Abstract*—This paper explores a novel method to solve the classic Job Shop Scheduling Problem, which removes the start times of orders from the decision variables, allowing the metaheuristic techniques used to further explore the variable space of machine numbers, the start times are then deterministically calculated by various sorting mechanisms.**

I. Introduction

A. The Job Shop Scheduling Problem

Scheduling is concerned with allocation of resources over time so as to execute the processing tasks required to manufacture a given set of products [1]. Job Shop Scheduling (JSP) is one of the most popular combinatorial optimization problems and was the first problem to be presented in competitive analysis [2]. In JSSP, there are N jobs to be processed on M dissimilar machines in a predefined succession which is distinct for every job. Each machine can only handle one job at once, and a job cannot be interrupted once begun. There are hard limits on the bounds of the orders/jobs; release and due dates for each operation/job are specified. The objective is to minimize the processing time, or makespan, the total time it takes to finish all operations. Other objectives include minimizing the total processing cost of all the operations, which is machine dependent. Scheduling problems vary in complexity widely according to real world tasks but are mostly NP-hard problems. Since the 1980s, scheduling problems have garnered a lot of interest from researchers who have lavished resources on problems such as JSP and a similar NP-Hard problem known as the Travelling Salesman Problem (TSP). Extensive work and development of meta-heuristic techniques have pushed researchers to benchmark their techniques on this combinatorial problem.

Previous researchers primarily used exact mathematical models like Branch Bound [3], historical rules and shifting bottleneck problems [4]. However, with increasing complexity in the problem due to increase in orders or increase in the number of machines, it was found that the above problems failed to find optimal solutions. Hence, the use of approximation techniques was considered as an alternative, which brought on the era of using NIAs or nature inspired techniques to tackle combinatorial problems. Genetic Algorithm or GA, has been recognised as a popular search algorithm for combinatorial problems. Classical GAs use a binary string to represent a potential solution, which is not suitable for ordering problems such as the job scheduling problem. Hence, many variations have been proposed, from new chromosome representations [5][6], to variations on standard GA operators [7][8].

B. Paper Contribution

We have proposed a 2 step algorithm that removes the set of start times as decision variables in the classical JSP formulation and only runs metaheuristic techniques with machine number as decision variables. Doing so, we reduce the computational and time complexity of the process by a large margin. The metaheuristic techniques output machine numbers, after which, start times are deterministically calculated using various sorting techniques such as ordering by release dates, due dates, and a greedy algorithm. We have used 6 techniques, namely sTLBO, PSO, ABC, ALO, GA, and DE and paired them with our deterministic algorithm and compared our results against [9], who have also implemented heuristic assisted metaheuristic techniques.

We have expanded the problem to solve a multiobjective problem minimizing makespan, processing time and the processing cost. We have also included a setup time after every order. We generated the pareto front for all the problems and have shown the data in the appendix. We have also included the data for the setup time in the appendix.

This paper follows the following structure, section II presents a review of the past methods used to tackle scheduling problems, section III describes the problem in detail while section IV presents our approach to the problem. Section V summarises our results, we conclude in section VI and in section VII address the possibilities for improving the algorithm.

II. Literature Review

The classic job scheduling problem is an NP-Complete problem when the number of machines is greater than 2 [10]. There have been many variations over the last 6 decades, with different constraint relaxations, [11] has presented a fantastic overview of the different types of job scheduling problems for various applications, which include supply chain problems, electronics or machine manufacturing problems and more. [12][13] both present a genetic algorithm for the flexible job shop problem, both employing different strategies for the generation of the initial population. [14] presents a hybrid particle swarm optimization algorithm. [15] presents a heuristic approach for large scale job scheduling problems, [16] provides large scale benchmarks for JSP problems, more akin to the number of jobs and machines found in the industry. Evolutionary algorithms like genetic algorithm have been used extensively for scheduling problems, [17] used a modified genetic algorithm by combining critical block and DG distance in crossover and mutation, [18] proposes local search genetic

algorithms, [19] combines the previous method by adding intelligent search agents.

A keyword search of 'Job Shop Scheduling' on google scholar yields a total of 24,200 publications in the last 10 years alone.

III. Problem Formulation

The problem considered here is a specific job scheduling problem [20]. The problem contains a set of operations *I* using a set of parallel dissimilar machines *J*. Each order $i \in I$ can only begin processing after its release date r_i and has to be completed before its due date d_i. Order i can be solved on any machines, however, it cannot be split across multiple machines, and once an order starts processing, it cannot be interrupted. Each order $i \in I$ on machine $j \in J$ has a processing time p_{ij} and processing cost c_{ij} associated with it. Each ordered pair of orders $(i,k) \in I$xI has a setup time st_{ik} associated with it, which constrains the start time of order k (s_k) to be atleast st_{ik} greater than the start time of order i (s_i). The objective of the problem is to find a least cost schedule to process all the orders within their time frames. We have also expanded the problem to minimize total processing time, and makespan, which is the time between the processing of the first order and the finishing of the last order. The problem for MHT is defined as,

A. Decision Variables

$$x_{ij} = \begin{cases} 1, & \text{if order i is assigned to machine j} \\ 0, & \text{otherwise} \end{cases}$$

B. Problem Structure

$$p_{cost} = \min \sum_{i\in I} \sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} P_i$$

$$p_{time} = \min \sum_{i\in I} \sum_{j\in J} p_{ij}x_{ij} + \sum_{i\in I} P_i$$

$$makespan = \min\left(\max\left(f_i\right) - \min\left(s_i\right)\right) + \sum_{i\in I} P_i$$

s.t.
$$s_i \geq r_i \quad \forall i \in I$$
$$s_i \leq d_i - \sum_{j\in J} p_{ij}x_{ij} \quad \forall i \in I$$

$$s_i \leq s_k - st_{ik} - \sum_{j\in J} p_{ij}x_{ij},$$
if order i precedes order k on the same machine

$$P_i = \begin{cases} 1000, & \text{if } s_i \geq d_i - \sum_{j\in J} p_{ij}x_{ij} \quad \forall i \in I \\ 0, & otherwise \end{cases}$$

where,
P_i = the penalty incurred by order i exceeding its due date.
s_i = start time of order i
$f_i = s_i + \sum_{j\in J} p_{ij}x_{ij}$

There are some assumptions to be made in the JSP Problem, they have been summarised in Table I:

TABLE I
Assumptions for JSP Problem

Description
The number of jobs is pre-known and fixed.
All machines are available at time zero.
Each operation is given a unique and predetermined processing time.
Jobs are independent from one another, i.e., there are no precedence constraints among different jobs.
Each job can be processed on one machine only.
Each machine can process only one job at a time.
Machines are independent of each other.
Once a job is started, it must be completed without interruption.
All machines can perform with full efficiency constantly without the need of maintenance.

IV. Our Approach

A. Metaheuristic Techniques used

We used 6 metaheuristic techniques combined with our sorting based heuristic on 12 datasets. Inbuilt matlab function gamultiobj was used to solve for pareto fronts minimizing processing time, cost and makespan.

1) sTLBO: Sanitised TLBO or sTLBO as proposed in [21] is a modification to the original TLBO algorithm where the the step of identification and removing of duplicates is avoided, as it is a computationally expensive step for a computationally expensive algorithm.

2) PSO: Particle swarm optimization [22] is a algorithm that mimics the social behaviour of a bird flock or a school of fish, each entity has a position which it updates using a velocity to avoid predators, or to find food. Each entity remembers the best location it has found, along with the best location that all the entities have found. Thus, the velocity of all entities are influenced by their individual experiences and the collective experience of the group. The particle velocity (v_i) of entity i is given by:

$$v_i = wv_i + c_1r_1(p_{best,i} - X_i) + c_2r_2(g_{best,i} - X_i)$$

The particle location is updated as:

$$X_i = X_i + v_i$$

p_{best} and g_{best} are updated as follows:

$$\begin{matrix} p_{best,i} = p_i \\ f_{p_{best,i}} = f_i \end{matrix} \quad \text{if} \quad f_i < f_{p_{best,i}}$$

$$\begin{matrix} g_{best} = p_{best,i} \\ f_{g_{best,i}} = f_{p_{best,i}} \end{matrix} \quad \text{if} \quad f_{p_{best,i}} < f_{g_{best}}$$

v_i	velocity of i^{th} particle
w	inertia of the particles
c1 and c2	acceleration coefficients
r1 and r2	vectors of random numbers ∈ [0,1] of size *(1xD)*
$p_{best,i}$	personal best of i^{th} particle
g_{best}	global best
X_i	position of i^{th} particle

3) ABC: Artificial Bee Colony Algorithm or ABC as proposed by [23] is a swarm intelligence algorithm which models the behaviour of honey bees looking for food sources. The food sources go through an employed bee phase where employed bees try to find better food sources than the one associated with it. They then undergo an onlooker bee phase probabilities are assigned to food sources representing nectar amount and new solutions are generated. In both of the phases, greedy selection is used to update the population. After completing an iteration of the 2 phases, an exhausted food

source might undergo a scout bee phase where it's discarded and a new solution is generated.

The fitness solution is evaluated as:

$$fit = \begin{cases} \frac{1}{1+f} & \text{if } f \geq 0 \\ 1 + |f| & f < 0 \end{cases}$$

Greedy Selection to update the solution:

$$\begin{matrix} X = X_{new} \\ f = f_{new} \end{matrix} \text{ if } fit_{new} > fit$$

X	Current solution
X_{new}	Newly generated solution
f	Objective function value of a solution
f_{new}	Objective function value of new solution
fit	Fitness of a solution
fit_{new}	Fitness of new solution

4) ALO: Ant lion optimizer or ALO mimics the hunting of antlions in nature [24]. Ants move across the search space using random walk as follows:

$$X(t) = [0, cumsum(2r(t_1) - 1), cumsum(2r(t_2) - 1), ..., cumsum(2r(t_n) - 1)]$$

where $cumsum$ calculates the cumulative sum, n is the maximum number of iteration, t shows the step of random walk, and r(t) is a stochastic function defined as follows:

$$fit = \begin{cases} 1 & \text{if } rand \geq 0 \\ 0 & \text{if } rand < 0 \end{cases}$$

where rand is a random number $\in [0, 1]$ generated with a uniform distribution. The position of the ants and hiding antlions are saved in matrices.

5) GA: Genetic Algorithm is an evolutionary algorithm based on natural selection. Since proposed, it has found enormous amounts of uses in both constrained and unconstrained optimization problems. The algorithm chooses parents from a randomly generated population, and produces offspring through and binomial/binary crossover. More information can be found at: https://in.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

6) DE: Differential Evolution is a stochastic population based metaheuristic technique where each solution is known as a chromosome, and the chromosome undergoes mutation to form a donor vector, which is followed by a recombination step to give the final output. The selection of better solutions is performed only after the generation of all trial vectors. Recombination is done to increase diversity, and can be done through binomial or exponential crossover.

B. Sorting Based Heuristic

In this paper, we propose a sorting based heuristic to deterministically calculate the set of start times as per the machine numbers assigned to each job by the metaheuristic technique. The decision variables used in our approach consist of machine numbers assigned to each job, $m_i \in J$ where m_i is the machine number assigned to job i. The start times are calculated through a sorting based deterministic algorithm. Later, we have also combined our heuristic with the heuristic assisted metaheuristic techniques, the problem uses 2 objective files, in succession. First the machine numbers are output by the metaheuristic techniques, and the start times are calculated using our algorithm, both machine number and start times are then passed to the next objective file. The pseudocode for the algorithm is as follows.

$TotalVio = 0$
for $m = 1, 2, \ldots, M$ **do**
  I_m = set of orders on machine m
  $minViolation = 1e9$
  $Violation = 1$
  if $|I_m| > 0$ **then**
    $Violation = 0$
    $ctr = 0$
    Arrange the orders in I_m on machine m by their release date.
    Derive least possible start times for each order
    if there is a violation in due date of an order **then**
      $ctr = ctr + 1$
      $Violation = 1$
    end if
    if $Violation = 0$ **then**
      continue loop
    end if
    $minViolation = min(minViolation, ctr * 1000)$
    $Violation = 0$
    $ctr = 0$
    Arrange the orders in I_m on machine m by their due date.
    Derive least possible start times for each order
    if there is a violation in due date of an order **then**
      $ctr = ctr + 1$
      $Violation = 1$
    end if
    if $Violation = 0$ **then**
      continue loop
    end if
    $minViolation = min(minViolation, ctr * 1000)$
    $Violation = 0$
    $ctr = 0$
    Arrange the orders in I_m on machine m by their release date + duration on machine m.
    Derive least possible start times for each order
    if there is a violation in due date of an order **then**
      $ctr = ctr + 1$
      $Violation = 1$
    end if
    if $Violation = 0$ **then**
      continue loop
    end if
    $minViolation = min(minViolation, ctr * 1000)$
    $Violation = 0$
    $ctr = 0$
    Arrange the orders in I_m on machine m by their due date - duration on machine m.
    Derive least possible start times for each order
    if there is a violation in due date of an order **then**

$ctr = ctr + 1$
$Violation = 1$
end if
if $Violation = 0$ **then**
continue loop
end if
$minViolation = min(minViolation, ctr*1000)$
if $Violation = 1$ **then**
$TotalVio = TotalVio + minViolation$
end if
end if
end for
$totalProcessingCost = sum(x_{ij} * cij)$
$fitness = totalProcessingCost + TotalVio$

V. Results

In this section we have first presented the results of using our sorting based heuristic paired with 6 metaheuristic techniques, tested for 12 datasets. Secondly, we present the results after combining the algorithms using sTLBO on 10 datasets. The convergence curve of the fitness value is then plotted for datasets P5S1, P5S2 and P6S1. Subsequently, the next section presents the pareto fronts generated by solving a multi objective problem on P1S2, P2S2, P3S2, P4S2, P5S2. These datasets were chosen for more flexibility between the release and due dates to arrive at more solutions.

A. Sorting Based Heuristic

The statistical analysis has been presented here in II, all algorithms were run for 1000 iterations, and 15 runs were averaged to give the data. sTLBO with our heuristic is observed to perform the best, reaching or coming closest to the global optima for the first 10 datasets. Genetic Algorithm, Particle Swarm Optimization and Antlion Optimization struggled to reach even feasible solutions for the odd numbered datasets, only reaching feasible solutions in some runs. The best metric across all algorithms for each dataset has been bold-faced.

B. Combination

A combination of the algorithms is proposed in the following manner. Our algorithm is run on the problem for 100 iterations and the final population created is taken as the initial population for the CHERD heuristic algorithm, which is then run for 1000 iterations. This combination was tested on the first 10 problems. With sTLBO, the combined algorithm reached global optimality on all problems in most runs. The advantages of this combination are that we are able to reach better solutions in the problems where our deterministic algorithm did not reach global optimality on its own.

C. Mean Convergence Curve

The mean convergence curves have been plotted for problems 9-12, for problem 9, most algorithms reach optimality only after a few iterations, whereas problem 10 takes a lot of iterations for the algorithm to reach a feasible solution, possibly due the larger search space due to lower processing times.For problems 9 and 10, the mean fitness of sTLBO is much better than that of the other algorithms. For the large datasets, the algorithm struggles with all the metaheuristic techniques, with ABC and PSO doing the best in mean. However, for problem 12, our heuristic with sTLBO achieves the lowest cost schedule.

TABLE II
Statistical Analysis of Metaheuristic techniques combined with heuristic technique

		sTLBO	DE	GA	ABC	PSO	ALO
P1S1	B	**26**	**26**	**26**	**26**	**26**	**26**
	M	**26**	**26**	**26**	**26**	**26**	**26**
	Md	**26**	**26**	**26**	**26**	**26**	**26**
P1S2	B	**18**	**18**	**18**	**18**	**18**	**18**
	M	**18**	**18**	**18**	**18**	**18**	**18**
	Md	**18**	**18**	**18**	**18**	**18**	**18**
P2S1	B	**60**	**60**	**60**	**60**	**60**	**60**
	M	**60.00**	**60.00**	854.60	**60.00**	721.53	457.07
	Md	**60**	**60**	1053	**60**	1052	**60**
P2S2	B	**44**	**44**	**44**	**44**	**44**	**44**
	M	**44**	**44**	44.13	**44**	**44**	**44**
	Md	**44**	**44**	**44**	**44**	**44**	**44**
P3S1	B	**103**	**103**	107	**103**	104	**103**
	M	103.13	**103.00**	965.53	103.20	633.73	369.07
	Md	**103**	**103**	1098	**103**	1096	105
P3S2	B	**84**	**84**	**84**	**84**	**84**	**84**
	M	**84.00**	**84.00**	84.27	**84.00**	**84.00**	84.07
	Md	**84**	**84**	**84**	**84**	**84**	**84**
P4S1	B	**115**	119	117	**115**	116	118
	M	**115.20**	120.87	120.13	116.13	118.00	120.93
	Md	**115**	121	120	116	118	121
P4S2	B	**103**	105	**103**	**103**	**103**	**103**
	M	**103.00**	106.53	105.07	**103.00**	103.60	105.87
	Md	**103**	107	105	**103**	**103**	106
P5S1	B	**160**	170	165	161	163	169
	M	**161.27**	172.13	167.47	162.47	167.40	171.93
	Md	**161**	173	167	163	167	171
P5S2	B	**141**	149	**141**	**141**	142	144
	M	**141.00**	149.87	143.87	142.27	142.67	146.53
	Md	**141**	150	144	142	143	147
P6S1	B	763	895	816	712	**709**	773
	M	796.40	907.40	832.40	**721.60**	734.20	786.20
	Md	801	908	833	**725**	743	781
P6S2	B	**681**	876	821	718	713	763
	M	755.00	914.60	837.40	**724.40**	736.40	801.60
	Md	786	915	844	**725**	744	805

Fig. 1. Mean Convergence curve for P5S1

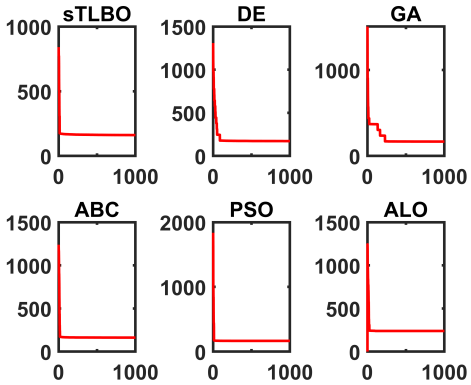
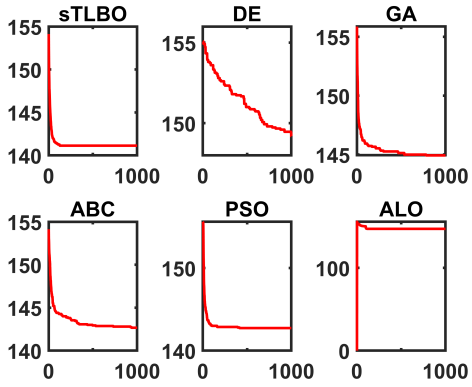


TABLE III
COMPARISON OF OUR ALGORITHM VS COMBINING IT WITH CHERD HEURISTIC

		SBH	SBHRH
P1S1	B	26	26
	M	26.00	26.00
	Md	26	26
P1S2	B	18	18
	M	18.00	18.00
	Md	18	18
P2S1	B	60	60
	M	60.00	60.00
	Md	60	60
P2S2	B	44	44
	M	44.00	44.00
	Md	44	44
P3S1	B	103	101
	M	103.13	101.80
	Md	103	102
P3S2	B	84	83
	M	84.00	83.00
	Md	84	83
P4S1	B	115	115
	M	115.20	115.73
	Md	115	116
P4S2	B	103	102
	M	103.00	102.00
	Md	103	102
P5S1	B	160	159
	M	161.27	161.13
	Md	161	161
P5S2	B	141	140
	M	141.00	140.13
	Md	141	140

Fig. 2. Mean Convergence curve for P5S2



D. Multi-Objective Optimization

The pareto fronts for even numbered datasets were generated, so as to showcase more number of solutions. A comparison was done between the datasets without setup times and with a randomly generated set of setup times. It is evident that the addition of setup times reduces the number of solutions in the pareto front.

VI. Conclusion

This work proposed a sorting based heuristic in combination with a metaheuristic technique that only outputs the set of machine numbers to determine a potentially better schedule due to the reduction in the amount of variables. It was found that sTLBO combined with the heuristic outperformed all algorithms for problems 1-10, while ABC and PSO did better overall for the larger datasets. The algorithm was then combined with a repair operator, after which it reached global optima for problems 1-10 almost every run. The problem was expanded to solve a multiobjective optimization problem, including setup times in between orders. Pareto fronts with setup times had much lower amount of feasible schedules than fronts without setup times for the same problems.

Fig. 3. Mean Convergence curve for P6S1

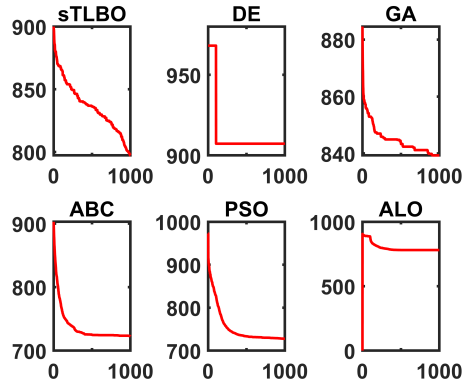
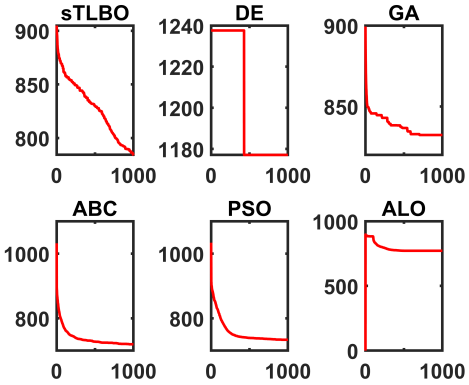


Fig. 4. Mean Convergence curve for P6S2



VII. Future Plans

The authors have considered an idea to combine the sorting algorithm with the repair operator in a single objective file, so as to avoid solving two optimization problems. The idea states that instead of one evaluation of each population member, it will be evaluated 2 times, once with each heuristic, and the lower fitness function will be selected and passed in.

Fig. 5. Pareto Front for P4S2 without Setup Times

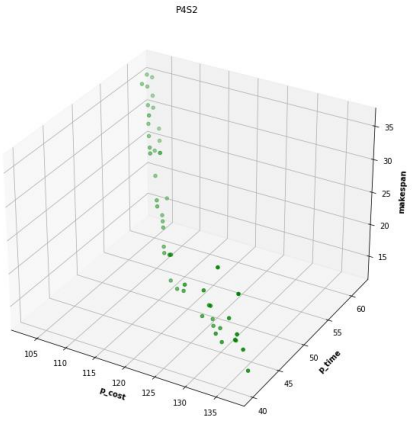


Fig. 6. Pareto Front for P4S2 with Setup Times

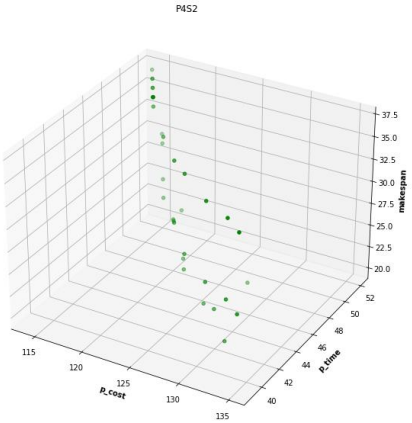


Fig. 7. Pareto Front for P5S2 without Setup Times

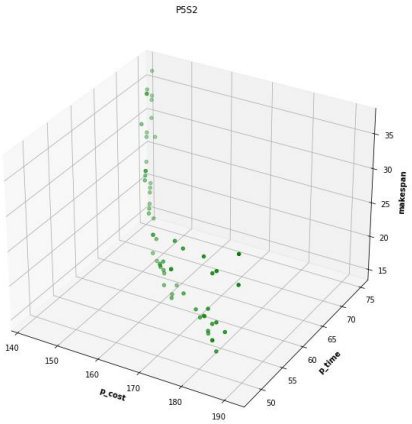
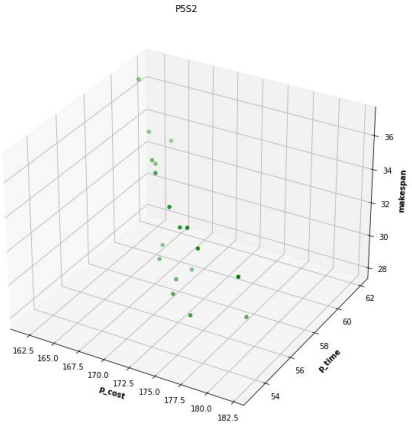


Fig. 8. Pareto Front for P5S2 with Setup Times



REFERENCES

[1] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer Publishing Company, Incorporated, 2001.

[2] R. L. Graham, "Bounds for certain multiprocessing anomalies," *The Bell System Technical Journal*, vol. 45, no. 9, pp. 1563–1581, 1966.

[3] J. Carlier and É. Pinson, "An algorithm for solving the job-shop problem," *Management science*, vol. 35, no. 2, pp. 164–176, 1989.

[4] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management science*, vol. 34, no. 3, pp. 391–401, 1988.

[5] C. Bierwirth, "A generalized permutation approach to job shop scheduling with genetic algorithms," *Operations-Research-Spektrum*, vol. 17, no. 2, pp. 87–92, 1995.

[6] G. Syswerda, "Scheduling optimization using genetic algorithms," *Handbook of genetic algorithms*, 1991.

[7] T. Yamada and R. Nakano, "A genetic algorithm applicable to large-scale job-shop problems." in *PPSN*, vol. 2, 1992, pp. 281–290.

[8] U. Dorndorf and E. Pesch, "Combining genetic and local search for solving the job shop scheduling problem," in *APMOD93 proc. Preprints*, 1993, pp. 142–149.

[9] R. Kommadath, D. Maharana, and P. Kotecha, "Efficient scheduling of jobs on dissimilar parallel machines using heuristic assisted metaheuristic techniques," *Chemical Engineering Research and Design*, vol. 188, pp. 916–934, 2022.

[10] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[11] H. Xiong, S. Shi, D. Ren, and J. Hu, "A survey of job shop scheduling problem: The types and models," *Computers Operations Research*, vol. 142, p. 105731, 2022.

[12] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & operations research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[13] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563–3573, 2011.

[14] D. Sha and C.-Y. Hsu, "A hybrid particle swarm optimization for job shop scheduling problem," *Computers & Industrial Engineering*, vol. 51, no. 4, pp. 791–808, 2006.

[15] M. K. Gavareshki and M. F. Zarandi., "A heuristic approach for large scale job shop scheduling problems," *Journal of Applied Sciences*, vol. 8, no. 6, pp. 992–999, 2008.

[16] G. Da Col and E. Teppan, "Large-scale benchmarks for the job shop scheduling problem," 2021.

[17] M. Omar, A. Baharum, and Y. A. Hasan, "A job-shop scheduling problem (jssp) using genetic algorithm (ga)," 2006.

[18] B. M. Ombuki and M. Ventresca, "Local search genetic algorithms for the job shop scheduling problem," *Applied Intelligence*, vol. 21, no. 1, p. 99 – 109, 2004, cited by: 103.

[19] L. Asadzadeh, "A local search genetic algorithm for the job shop scheduling problem with intelligent agents," *Computers Industrial Engineering*, vol. 85, pp. 376–383, 2015.

[20] J. Hooker, G. Ottosson, E. Thorsteinsson, and H.-J. Kim, "On integrating constraint propagation and linear programming for combinatorial optimization," 01 1999, pp. 136–141.

[21] R. Kommadath, D. Maharana, C. Sivadurgaprasad, and P. Kotecha, "Parallel computing strategies for sanitized teaching learning based optimization," *Journal of Computational Science*, vol. 63, p. 101766, 2022.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[23] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[24] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.