COMPARISON OF PARALLEL PROCESSORS SCHEDULING ALGORITHMS TO MINIMISE MAKESPAN IN A GALVANISING PLANT

Mendon Dewa
Durban University of Technology, South Africa/mendond@dut.ac.za

Bakhe Nleya
Durban University of Technology, South Africa/bakhen@dut.ac.za

Abstract
Galvanising lines consist of load/loading stations and a series of processing tanks that are generally energy-intensive. Each raw workpart needs to go through a number of processing stages sequentially. Job sizes and processing time vary from part to part, hence the need to derive an optimal schedule to minimise total processing time in a batch. The problem of minimizing the makespan on parallel processing machines using different scheduling algorithms is studied in this paper. A set of 50 independent tasks were scheduled on parallel processors in order to minimize schedule length using Integer Linear Programming, Shortest Processing Time, Longest Processing Time, and Greedy Genetic algorithms. The experimental results demonstrated that our Greedy Genetic algorithm outperformed other algorithms on minimizing makespan on parallel processing machines.

Keywords: Makespan Minimisation**,** Greedy Genetic Algorithm, Integer Linear Programming, Shortest Processing Time, Longest Processing Time

Introduction
Industries play a vital role in the economic development of any nation, and scheduling process may influence productivity of these industries. Production scheduling is commonly viewed as one of the most important issues in the planning and operation of manufacturing system [1]. Scheduling is generally defined as the allocation of resources to perform tasks over some specified time. It is a decision making process that is executed regularly in many manufacturing and services organisations. Performance criteria such as manufacturing lead times, machine utilization, meeting due dates, inventory costs, quality of products, and customer satisfaction, are all dependent on efficiency in scheduling jobs in a system [2]. Therefore, it is imperative that organisations develop effective scheduling methodologies that ensure that the desired objectives are achieved. The case study galvaniser has been facing challenges with regard to scheduling ungalvanised raw workparts, characterised by using disorganised approaches to load 4 parallel lines, which led to missed delivery dates and excessive idle times and energy consumption. The problem of minimizing the makespan on parallel processing machines using different scheduling algorithms is studied in this paper.

Literature Review
Scheduling is about assigning products and processes to available production equipment and time is key in scheduling since decision has to be made at what time the product is processed

[3]. Sequencing also complements scheduling since it actually leads to optimization where decision on the order to be taken is made, based on the algorithm designed for the optimisation considering various constraints such as cost, energy consumption and other parameters.
Verdejo et al. [4] solved a production sequencing problem of a continuous galvanizing using an algorithm based on the Tabu Search, primarily through grouping and sequencing. They successively divided an unfinished cold coils pool into groups of coils (production campaigns) mainly according to their due dates and their required galvanizing types within stated campaign sizes. The continuous galvanising line would process the coils of a production campaign continuously one after another with each new campaign requiring major set-up changes of the line. The sequence in which the coils were to be processed was then determined after ascertaining the coil composition of a campaign and this phase is a complex exercise taking consideration of a myriad of constraints.
Weinert et al. [5] posited that scheduling would influence the energy consumption behaviour of the whole system, and by integrating energy efficiency criteria into scheduling, a reduction of energy costs is to be expected. A requirement for the integration of energy efficiency benchmarks in planning undertakings is a comprehensive prediction of the energy consumption which should be executed at machine level.
Liu et al. [6] applied Non-dominant Sorting Genetic Algorithm (NDSGA) to develop a model with the objective of minimising total non-processing electricity and consumption as well as the total weighted tardiness for a job shop scheduling problem. NDSGA performance was then tested on an extended version of Fisher and Thompson job shop that integrated the electrical consumption profiles for the equipment. The result demonstrated that the total non-processing electrical energy consumption in the job shop was decreasing considerably, however at the detriment of its performance on the total weighted tardiness objective up to a certain level. On the other hand, Fernandez et al. [7] utilised ant colony optimization to schedule a galvanizing line. Given a combinatorial non-deterministic polynomial-time hard (NP-hard) problem, it was critical to develop an intelligent algorithm for scheduling able to optimisation by translating the scheduling rules and prevailing operational criteria into technical constraints and cost functions, which guaranteed a satisfactory solution within a reasonably short computation time.
Zhang et al. [8] designed and applied a self-adaptive differential evolution (DE) algorithm to solve production scheduling concerned with energy consumption optimization for process industry by introducing self-adaptive parameter mechanism into basic DE algorithm. The simulation results from the algorithm demonstrated that the designed self-adaptive DE algorithm had gains of reduced solution time and faster operation, coupled with reduced energy consumption.
Pugazhenthi et al. [9] analysed the flow shop scheduling with machines arranged in series and jobs processed in the same order. A novel BAT heuristic for achieving minimal makespan by reaching the lower bound through a reverse engineering method was proposed for the flow shop problems. The heuristic was applied with Genetic Algorithm (GA) in a MATLAB environment. The results were compared with traditional heuristics and it was found that the GA applied BAT heuristic yielded better results. However, multi-objective studies for more complex scheduling problems with additional features such as parallel machines and setup time are uncommon and new algorithms for such problems are desirable in practice [10].
Multi-objective flow shop scheduling with sequence dependent setup time can be regarded as NP hard since it is characterised by greater complexity toward optimality in a reasonable time. Mohammadi et al. [11] discussed the application of Robust Genetic Algorithm to solve a flowshop scheduling problem. Garen [12] presented a multi-objective GA for job-shop scheduling

with a novel representation that enabled the use of simple recombination operators and the simulation results demostrated that the proposed approach was able to generate a set of solutions close to the Pareto-optimal front.
Saeidi et al. [13] proposed a novel mathematical linear programming model for scheduling the jobs in a parallel environment to minimize completion time and total machine cost. On the other hand, a novel mathematical model to minimize energy consumption costs for scheduling a single machine was proposed and GA was used to generate the optimal solution [14]. Madivada et al. [15] also proposed a new meta-heuristic solution approach for multi-objective job shop scheduling problems. The concept of fuzzy dominance was employed for performance evaluation of solutions in a multi-objective scenario and the results obtained from the study demostrated that the proposed algorithm can be used as a new alternative technique for scheduling complex multi-objective job shop problems.

Methodology

Data was collected for daily production quantities, customer due dates, and time studies were conducted to ascertain the duration of the pretreatment and galvanizing process. Fig.1 shows the steps for steel pre-treatment in hot dip galvanising.

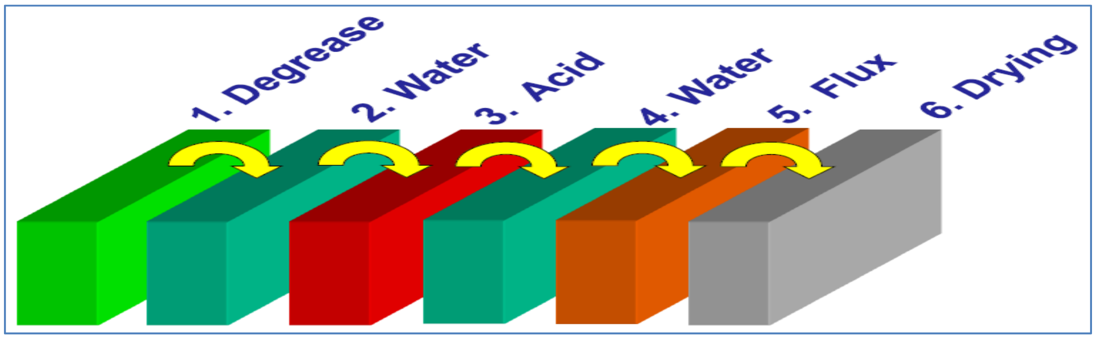

Fig.1 Steps for steel pre-treatment in hot dip galvanising

The case-in-point galvanising plant is characterised by the following problem variables:

- Number of processing lines = 4
- Average number of jobs per day = 50
- Average productive working hours per day = 6
- Total processing time from setup + pre-treatment + galvanising + transfer time

The adopted approach considered n jobs $Ji(i = 1,\dots, n)$ with processing times pi ($i = 1,\dots, n$) to be processed on m identical parallel processors $M1,\dots, Mm$. Integer Linear Programming, Shortest Processing Time Algorithm, and Longest Processing Time algorithms were executed for a set of 50 independent jobs for assignment to 4 parallel process lines in order to minimize makespan. Section 4.1, 4.2 and 4.3 will show the results from executing these algorithms.

Data was also collected for 100 jobs that were to be executed on four process lines. The term makespan refers to the cumulative time to complete all the jobs on all process lines. It is the time taken from scheduling the first raw steel workpart until the completion of the last raw steel workpart. The objective function of the problem Greedy Genetic Algorithm (GGA) is to find a valid schedule that yields the minimum makespan.
Table 1 and Table 2 represent two chromosomes that have 4 processors and 3 jobs placed on each of the processors. The time taken by the processor which runs the longest denotes the total

makespan for the entire chromosome since all the machines are running in parallel. The algorithm then creates similar chromosomes and the makespan for each chromosome is computed. Once the makespans for all the chromosomes are computed, the least makespan amongst the chromosomes would return the best makespan for the generation while the average of all the chromosomes' makespan returns the average for that particular generation.

Table 1 Chromosome 1

Processor 1			Processor 2			Processor 3			Processor 4		
J3	J6	J10	J7	J17	J28	J4	J9	J20	J1	J16	J34

Table 2 Chromosome 2

Processor 1			Processor 2			Processor 3			Processor 4		
J2	J5	J11	J12	J18	J32	J24	J39	J44	J4	J21	J37

Tournament selection was then conducted to filter out chromosomes which had better makespan values (in this case lesser makespan value) after the makespan computation for the different chromosomes. These chromosomes were then selected to undergo crossover and mutation. A tournament size of two was considered for the GGA ad these two chromosomes were randomly selected from the population and their makespan values were compared, and the chromosome with a lesser makespan value was deemed the winner. After the parents were selected, crossover was applied to them.
A one-point crossover was adopted, whereby after constructing the parent pool, two chromosomes were chosen at random and a crossover point was selected randomly as indicated by the arrow in Table 3. From the crossover point, as shown in Table 4, the two parent chromosomes were interchanged to produce two new off-springs, and if there are any duplicates after crossover, they will be randomly replaced by unplaced jobs.

Table 3 Chromosome encoding before crossover

Processor 1			Processor 2			Processor 3			Processor 4		
J3	J6	J10	J7	J17	J28	J4	J9	J20	J1	J16	J34
J2	J5	J11	J12	J18	J32	J24	J39	J44	J4	J21	J37

Table 4 Chromosome encoding after crossover

Processor 1			Processor 2			Processor 3			Processor 4		
J3	J6	J10	J7	J17	J28	J24	J39	J44	J4	J21	J37
J2	J5	J11	J12	J18	J32	J4	J9	J20	J1	J16	J34

Two offspring are produced for the new generation and the makespan for each processor and chromosome is computed again. A crossover rate of 0.4 was used.

The mutation step was then executed after crossover, mutation operators are usually applied with small probability to stimulate small local disturbance of the individuals, giving more impact on individuals as opposed to crossover operators which transmit genetic information from parents to offsprings, with less impact on individuals. Mutation is a secondary operator which assures that the chances of searching a particular subspace of the problem space is never zero by inhibit the possibility of converging to a local optimum, rather than the global optimum [16]. Mutation can be performed in different ways such as flip bit, boundary, non-uniform, uniform, and Gaussian. A gene (job) was randomly selected from each chromosome and flipped the job placement among the chromosome. For instance, in Table 5, job J20 in Chromosome 1 and job

J32 in Chromosome 2 were randomly chosen and their positions were flipped in these two chromosomes.

Table 5 Chromosome encoding during mutation

Processor 1			Processor 2			Processor 3			Processor 4		
J3	J6	J10	J7	J17	J28	J4	J9	J20	J1	J16	J34
J2	J5	J11	J12	J18	J32	J24	J39	J44	J4	J21	J37

If there are duplicate jobs being placed on two processors, this duplication is removed by randomly selecting one of the duplicates and substituting it by an unplaced job. Mutation during evolution occurs according to a user-definable mutation probability and a mutation rate in the range 0.02-0.20 was used.

The process of recombination, mutation, decoding the individual strings, evaluation of the objective functions, and assigning of fitness values to chromosomes continues through subsequent generations. As a result, good individuals are preserved and mated with one another while the less fit individuals decease, leading to an increase in the average performance of individuals in a population. The genetic algorithm was terminated after 150 generations.

Results

The results were derived by using a user-friendly TORSCHE (Time Optimisation, Resources, SCHEduling) Scheduling Toolbox for MATLAB. The TORSCHE toolbox is designed for researches in operations research or industrial engineering and it focuses on scheduling, with particular attention to graphs and graph algorithms due to their large interconnection with scheduling theory. A scheduling problem for parallel machines is characterised by allocation of jobs to machines and then generating a sequence of the jobs on a machine. A minimal makespan represents a balanced load on the processing machines.

A: Integer Linear Programming Algorithm

Integer Linear Programming Algorithm solves the makespan problem, where a set of independent jobs were assigned to parallel identical processors in order to minimize schedule length. Pre-emption is not allowed and the algorithm finds optimal schedule using Integer Linear Programming (ILP). The algorithm usage is outlined as follows:

$x_{ikj} = 1$, *if job j is scheduled as the kth to last job on processor i*

The objective function is to minimize $\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n} kp_{ij}x_{ikj}$ …… *Eq. 1*

Subject to:

$\sum_{i=1}^{m}\sum_{k=1}^{n} x_{ikj} = 1 \quad j = 1,\ldots\ldots\ldots.n$

$\sum_{j=1}^{n} x_{ikj} \le 1 \quad i = 1,\ldots\ldots\ldots.m \text{ and } k = 1,\ldots\ldots\ldots.n$

$x_{ikj} \in \{0,1\} \quad i = 1,\ldots\ldots\ldots.m,\ k = 1,\ldots\ldots\ldots.n \text{ and } j = 1,\ldots\ldots\ldots.n$

Where p_{ij} *is processing time of job j on machine i*

TORSCHE command TS = algpcmax (T, problem, processors) was executed for the above algorithm and the resulting schedule is displayed in Fig.2.

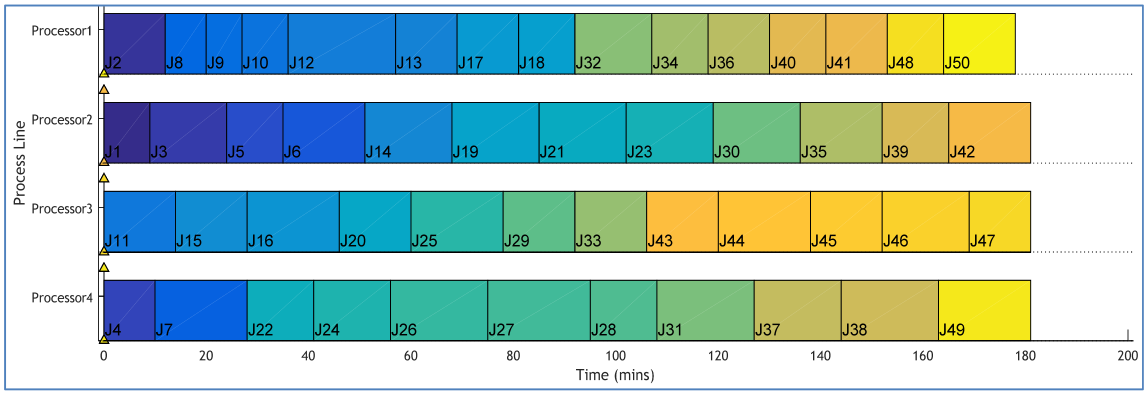

Fig.2 Optimal schedule using Integer Linear Programming

B: Shortest Processing Time

Shortest Processing Time (SPT) Algorithm was also used to solve the makespan problem, and the tasks are arranged in non-decreasing order of processing time p_j before aplying List Scheduling algorithm. According to the SPT-rule, jobs are ordered according to non-increasing processing requirements, and schedule each successive job preemptively to minimise its completion time. The procedure is to schedule job n on the fastest processor M_1 until it is completed at time $t_1 = pn/s_1$. Thereafter schedule job $n - 1$ first on processor M_2 for t_1 time units and then on processor M_1 from time t_1 to time $t_2 \geq t_1$ until it is completed. Job $n - 2$ is scheduled on M_3 for t_1 time units, on M_2 for t_2 - t_1 time units, and on processor M_1 from time t_2 to time $t_3 \geq t_2$ until it is completed, and so forth. The SPT algorithm is as follows:

```
1. $a := 0;$
2. WHILE $p1 > 0$ DO
BEGIN
3.      Find the largest index $i$ with $p_i > 0;$
4.      $\Delta t := p_i/s_1;$
5.      For $\gamma := i$ DOWN TO $k := \max\{1, i - m + 1\}$ DO BEGIN
6.      Schedule job $\gamma$ on $M_{1+i-\gamma}$ during $[a, a + \Delta t];$
7.      $p_\gamma := p_\gamma - \Delta t \cdot s_{1+i-\gamma}$
END
8. $a := a + \Delta t$
END
```

TORSCHE command TS = sptcmax (T, problem, processors) was executed for the above algorithm and the resulting schedule is displayed in Fig.3.

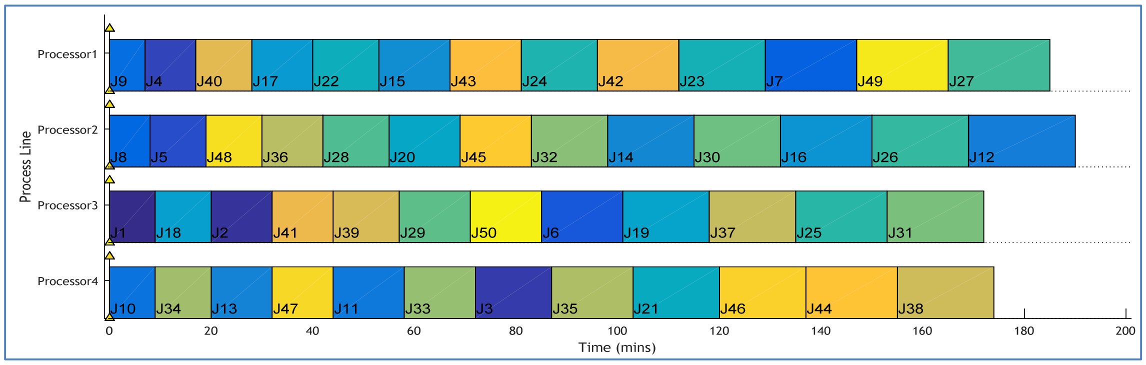

Fig.3 Optimal schedule using Shortest Processing Time

C: Longest Processing Time

Longest Processing Time (LPT) Algorithm was also used makespan problem, and the jobs were arranged in non-increasing order of processing time p_j after which List Scheduling algorithm was applied. With List Scheduling (LS) algorithm, jobs are taken from a pre-specified list and whenever a processor becomes idle, the first available job on the list is scheduled and subsequently removed from the list. TORSCHE command RS = listsch (T, problem, processors, 'LPT') was executed and the resulting schedule is displayed in Fig.4.

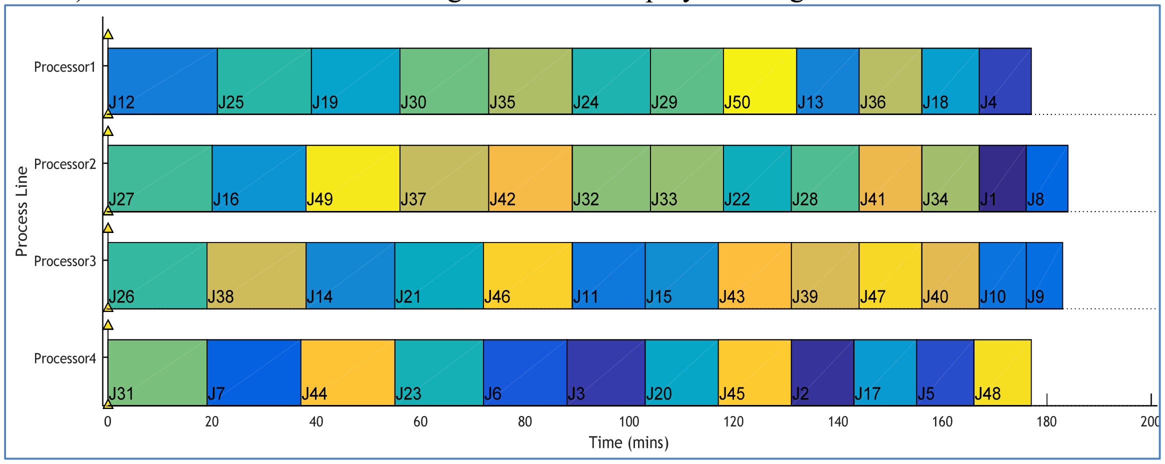

Fig.4 Optimal schedule using Longest Processing Time

D: Greedy Genetic Algorithm

We then developed a Greedy Genetic Algorithm to minimize the makespan of placing 50 jobs on 4 galvanising processing lines. A sum of 100 jobs were randomly placed on the 4 processing lines in the form of a chromosome and generated 50 chromosomes. The makespan for each chromosome was then calculated and we selected the best chromosomes using tournament selection. Thereafter, crossover and mutation were performed, with specific parameters and replaced the newly generated chromosomes with the previous ones in the population. The process was replicated for 150 generations. This procedure can be summarised as follows:

```
PROCEDURE cGA.
Determine a population size popsize.
Begin
t := 0;
initialize P(t);
evaluate P(t);
while (not termination-condition)
t := t + 1;
select P(t) from P(t − 1);
crossover P(t);
mutate P(t);
evaluate P(t);
end while
end begin
End
```

The GGA search for a one-way based on the base point and that way it would the ability of global optimization and increase speed, and speed up the generation of good population. In addition, the genetic algorithm in the early evolution of population was conducted to ensure an overall search and avoiding premature convergence. The following points summarise the GGA; Z is the solution it builds.

```
Compute job densities;
Sort the jobs by their value densities;
$Z \leftarrow \emptyset$;
$Weight \leftarrow 0$;
For the job in sorted order
$i \leftarrow$ next job;
if ( $weight + w_i \leq C$ )
$Z = Z \cup \{i\}$;
$Weight \leftarrow weight + w_i$;
Evaluate and report Z;
```

A series of 10 experiments was conducted and average values of the outcome were taken each time. The results for GGA demostrated that although the value of the resulting makespan was varying from one experiment to the other, when balancing the makespan with the running time, the optimal combination of parameter values was a crossover rate of 0.4, a mutation rate of 0.2, from a population size of 50. The resulting schedule is displayed in Fig.5.

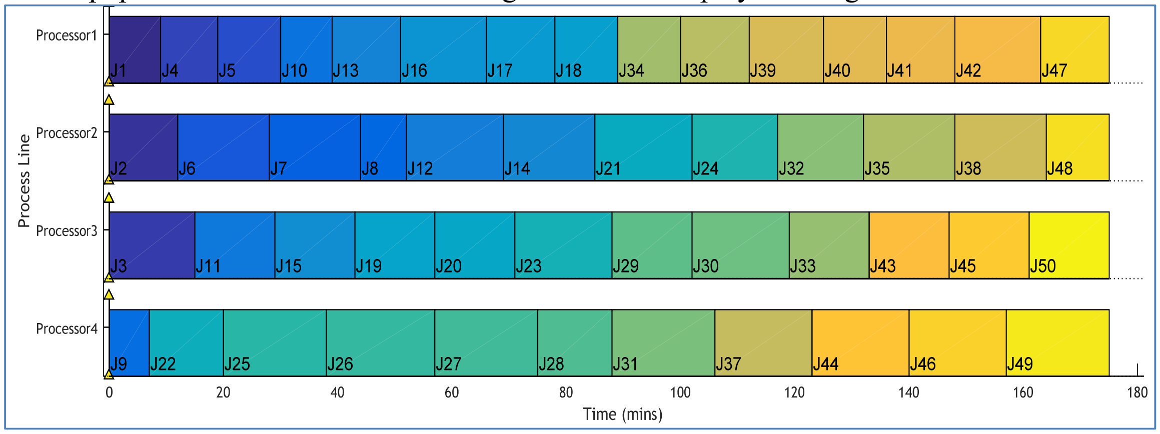

Fig.5 Optimal schedule using Longest Processing Time

Discussion

The scheduling problem discussed in this study is an assignment problem of assigning galvanising jobs to the galvanising lines in such a manner that minimises the makespan and enhances the performance of a production system. The results shown in Table 6 demonstrated that GGA outperformed other algorithms on minimizing makespan on parallel processing machines while Shortest Processing Time algorithm performed the worst.

Table 6 Summary of algorithms' makespan

No	Algorithm	Makespan
1	Integer Linear Programming	180
2	Shortest Processing Time	192
3	Longest Processing Time	182
4	Greedy Genetic Algorithm	176

Conclusion

The objective considered in this study is minimization of makespan using Integer Linear Programming, Shortest Processing Time, Longest Processing Time, and Greedy Genetic algorithms. This paper solved parallel line job shop scheduling problem and the experimental results demonstrated that the Greedy Genetic algorithm outperformed other algorithms on minimizing makespan on parallel processing machines.

References

[1] **Giret, A., Trentesaux, D. and Prabhu, V. 2015.** Sustainability in manufacturing operations scheduling: A state of the art review. *Journal of Manufacturing Systems*, *37*, pp.126-140**.**

[2] **Hitomi, K., 2017.** *Manufacturing systems engineering: A unified approach to manufacturing technology, production management and industrial economics*. Routledge.

[3] **Kallrath, J. 2002**. Planning and scheduling in the process industry. *OR spectrum*, *24*(3), pp.219-250.

[4] **Verdejo, V. V., Alarcó, M. A. P. and Sorlí, M. P. L. 2009.** Scheduling in a continuous galvanizing line. *Computers & Operations Research*, 36 (1): 280-296.

[5] **Weinert, N., Chiotellis, S. and Seliger, G. 2011**. Methodology for planning and operating energy-efficient production systems. *CIRP Annals - Manufacturing Technology*, 60 (1): 41-44.

[6] **Liu, Y., Dong, H., Lohse, N., Petrovic, S. and Gindy, N. 2014.** An investigation into minimising total energy consumption and total weighted tardiness in job shops. *Journal of Cleaner Production*, 65 (0): 87-96.

[7] **Fernandez, S., Alvarez, S., Díaz, D., Iglesias, M. and Ena, B. 2014.** *Scheduling a galvanizing line by ant colony optimization*. In: Swarm Intelligence. Springer, 146-157.

[8] **Zhang, L., Luo, Y., Zhang, Y. and Song, G. 2015.** Production Scheduling Oriented to Energy Consumption Optimization for Process Industry Based on Self-adaptive DE Algorithm. *International Journal of Control and Automation*, 8 (2): 31-42.

[9] Pugazhenthi, R., 2014. A Genetic Algorithm Applied Heuristic to Minimize the Makespan in a Flow Shop. *Procedia Engineering*, *97*, pp.1735-1744.

[10] **Parveen, S. and Ullah, H. 2010.** Review On Job-Shop And Flow-Shop Scheduling Using Multi Criteria Decision Making. *Journal of Mechanical Engineering*, *41*(2), pp.130-146.

[11] **Mohammadi, G. 2015.** Multi-objective flow shop production scheduling via robust genetic algorithms optimization technique. *International Journal of Service Science, Management and Engineering*, *2*(1), p.1.

[12] **Garen, J. 2002.** Multi objective job-shop scheduling with genetic algorithms using a new representation and standard uniform crossover. In *MOMH Workshop, Paris*(pp. 4-5).

[13] **Saeidi, S. 2016.** A Multi-objective Mathematical Model for Job Scheduling on Parallel Machines Using NSGA-II. *International Journal of Information Technology and Computer Science (IJITCS)*, *8*(8), pp.43-49.

[14] **Shrouf, F., Ordieres-Meré, J., García-Sánchez, A. and Ortega-Mier, M. 2014.** Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *Journal of Cleaner Production*, *67*, pp.197-207.

[15] **Madivada, H. and Rao, C.S.P. 2012.** An invasive weed optimization (IWO) approach for multi-objective job shop scheduling problems (JSSPs). *International Journal of Mechanical Engineering and Technology (IJMET)*, *3*(3), pp.627-637.

[16] **Eldos, T. 2013.** Mutative Genetic Algorithms. *Journal of Computations & Modelling*, 3 (2): 111-124.