# Bi-objective Optimization for Reentrant Shop Scheduling Problem

**Chettha Chamnanlor[1] and Kanchana Sethanan[2*]**

*[1]Faculty of Engineering at Sriracha, Kasetsart University Sriracha Campus, Chonburi 20230, Thailand*
*[2]Research Unit on System Modeling for Industry, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand*

**Corresponding author. E-mail: ksethanan@gmail.com*

## ABSTRACT

***This paper focused on optimal scheduling solutions for the reentrant flow shop (RFS) with uniform machines and different product families in order to minimize the makespan and mean flow time in the production of hard-drive components. An RFS environment includes several workstations, each of which consists of only one machine. At each stage, the machine can process any job. Each job must be produced according to the reentrant flow. This problem can be denoted as*** $Fm \mid fmls, recrc \mid C_{max}, \sum f_j/n$***. Although the throughput maximization can be given by minimizing the maximum completion time, it does not ensure minimizing the length of the time interval between the release time and completion time of all jobs. For production quality, each job should be produced in a short flow time, since some processes must be controlled by the aging condition. Thus, the approach of a single criterion, such as minimizing the makespan, is not sufficient for handling the problem; bi-objective optimization should be considered instead. For optimal convergence, meta-heuristics, such as simulated annealing or a genetic algorithm, should be studied for practical use as a support tool for scheduling jobs. This paper applied a multi-objective genetic algorithm (MOGA) for solving the RFS problem. The results showed that MOGA could optimally solve the problem with reasonable computational effort.***

**Keywords:** Reentrant flow shop, Bi-objective, Optimization, Meta-heuristic, Genetic algorithm

## INTRODUCTION

The reentrant flow shop scheduling problem (RFSP) has become an important research topic, since this manufacturing configuration is increasingly found in high-tech industries. In particular, slider fabrication in hard-drive production consists of several manufacturing stages in series, with only one machine at each stage; furthermore, a job can visit certain machines more than once. At the same time, several products with different operational sequences should be handled with many real-industrial constraints, such as product families and time windows (Chamnanlor et al., 2013). Therefore, the problem studied here should be classified as a reentrant shop problem; it is not easy to solve the problem in terms of polynomial time.

As the most well known scheduling problem, production shop scheduling commonly uses a single objective as a criterion; however, the importance of multi-objective optimization is growing. For many real-world problems, several objectives are optimized simultaneously, instead of a single objective. For example, when considering a criterion such as minimizing makespan in hard-drive production, a single objective does not ensure minimizing the length of the time interval between the release time and the completion time of jobs, although it yields throughput maximization. For production quality, each job should be produced with a short flow time, since some processes must be controlled by an aging condition. Thus, the approach of a single criterion, such as minimizing the makespan, is not sufficient for handling the problem; bi-objective optimization should be considered instead.

This paper studied the RFSP with two objectives. The first aim was to minimize the makespan that would maximize throughput. The second aim was to minimize the mean flow time for controlling the processing quality. Consequentially, the system constraints cannot avoid considering the time window. To solve the problem, approximated solutions need to be used in practice as a support tool for scheduling jobs with reasonable computational effort. The multi-objective genetic algorithm (MOGA), an optimal convergence method, has been widely studied and been shown to apply well in this field, and it will be proposed here for solving the RFS problem.

## LITERATURE REVIEW

Recently, the Reentrant Flow Shop Scheduling Problem has become a crucial research topic. It could be considered an NP-hard problem, since this system expanded from the traditional flow shop pioneered by Johnson (Sethanan, 2001; Chen et al., 2008). Although most reentrant shop studies have looked at different objectives, such as minimizing makespan (Hwang and Sun, 1997; Pan and Chen, 2003; Chen et al., 2007; Chen et al., 2008; Jing et al., 2008), minimizing cycle time (Park et al., 2000), and minimizing total weighted tardiness (Kang et al., 2007), they have concentrated on finding an optimal solution for only a single objective, while real-world scheduling problems involve many objectives simultaneously. Thus, for decision-making, multi-objective approaches that find reasonable compromised schedules are important (Chou et al., 2014).

Optimization problems considering multiple objectives have been studied and several approaches proposed over past decades. It is likely that the vector evaluated genetic algorithm (VEGA) pioneered by Schaffer is the first of the genetic algorithms with the dominant concept to solve the problem (Gen & Lin, 2014). It contains a modified selection step, so the sub-population is selected on the basis of each objective. To maintain the progress of the algorithm, the entire population is shuffled to obtain a new population with regular crossover and mutation operators. Following on this, various single-objective approaches have been extended to become multi-objective methods that attempt to evaluate the quality of the obtained non-dominated front in the Pareto optimization. Recently, the vector method has been studied with the simulated annealing process as vector evaluated simulated annealing (VESA). It operates on sets of solutions (clouds) with parallel computation of the objective functions for all solutions (Smutnicki et al., 2015).

Furthermore, much research has focused on multi-objective manufacturing problems, such as reentrant shop scheduling problems, where the approximate solution can be constructed by heuristics and meta-heuristics. For example, Miragliotta and Perona (2005) presented a new approach relying on a heuristic algorithm built upon a decentralized architecture to schedule the multi-objective reentrant shop problem. Lin et al. (2012) proposed a novel combination of the analytical hierarchy process (AHP) and genetic algorithm (GA) to deal with the dynamic re-entrant scheduling problem considering many criteria. For scheduling problems in semiconductor final testing, Sun et al. (2012) used the non-dominated sorting genetic algorithm II (NSGA-II) to solve the reentrant flow shop in order to minimize the makespan and penalty cost of all late products in the final testing process. In addition, Rifai et al. (2015) proposed the reentrant shop scheduling problem in the flexible manufacturing system (FMS) environment. Crowding distance-based substitution was incorporated to maintain the diversity of the population in the genetic algorithm in order to reduce the makespan, mean flow time, and tardiness.

From the research mentioned, genetic algorithms are the most accepted techniques for the bi-objective RFS problem. Hybridizing appropriate heuristics has been applied in several studies, since it can increase the high performance of GA (Gao et al., 2006; Gen et al., 2009).

## MATERIALS AND METHODS

### Reentrant shop scheduling

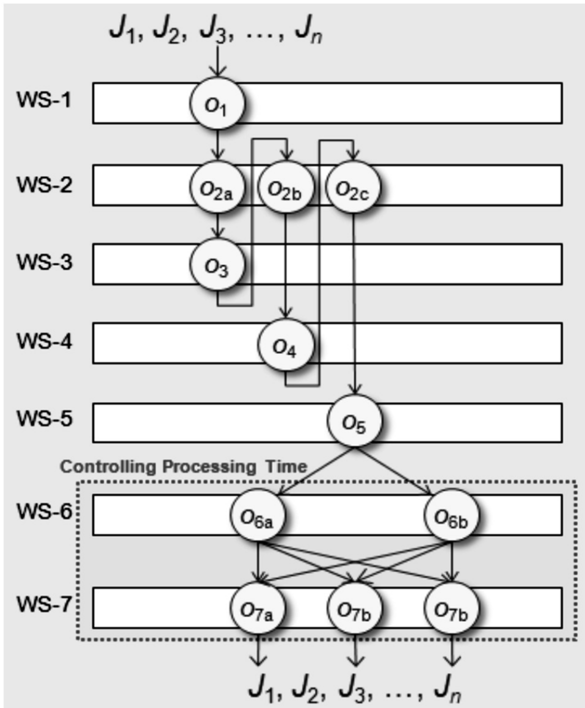


**Figure 1.** Systematic representation of RFSP in HDD manufacturing.

As shown in Figure 1, modified from Chamnanlor et al. (2013), a reentrant shop can be modeled for a HDD manufacturing shop in which n jobs need to be scheduled, minimizing both makespan and mean flow time. The reentrant flow shop scheduling problem is expanded from the traditional FSP, which consists of $m$ workstations (stages), with only one machine in each. According to the reentrant characteristic, the machine of some workstations can be used for several operations, and so the jobs can revisit the reentrant workstation several times.

In this system, each job includes different workloads (lots) exactly grouped with different families, and produced in the processing flow. Also, the flow processing time of all jobs had to be controlled to 30 min/lot for the time-window workstations (WS-6 and WS-7).

A complicated system, such as HDD manufacturing, produces many products (jobs) with several production conditions, such as job reentries, product families, machine restrictions, and even the time window constraint. Table 1 shows the different processing time depending on the operational sequence. The operations on WS-2 are used for three job reentries ($o_{2a}$, $o_{2b}$, and $o_{2c}$). WS-6 processes jobs under two-product families ($o_{6a}$ and $o_{6b}$), while WS-7 includes the restricted operations, such as $o_{7a}$, $o_{7b}$, and $o_{7c}$; all are used to calculate the objective functions.

**Table 1.** Processing time defined by different operations.

| Data defined | | |
|---|---|---|
| **Machines** | **Operations** | **Processing time (min)** |
| WS-1 | $o_1$ | 10 |
| WS-2 | $o_{2a}$ | 9 |
| WS-3 | $o_{2b}$ | 9 |
| WS-4 | $o_{2c}$ | 9 |
| WS-5 | $o_3$ | 9 |
| WS-6 | $o_4$ | 10 |
| WS-7 | $o_5$ | 9 |
| | $o_{6a}$ | 11 |
| | $o_{6b}$ | 12 |
| | $o_{7a}$ | 11 |
| | $o_{7b}$ | 10 |
| | $o_{7c}$ | 13 |

The operational sequence for six product types (A1, A2, A3, B1, B2, and B3) is shown in Table 2; these are derived from two families (A and B) and three sub-groups (1, 2, and 3). The operational sequences can be used to clarify the complex processing flows in the precedence relationships.

This paper focused on a bi-object reentrant shop scheduling problem, in which the makespan and the mean flow time are considered simultaneously. Because the maximum throughput can result from minimizing the maximum completion time of all jobs, scheduling all of the jobs in the system should be performed appropriately. However, minimizing the makespan does not ensure high-quality production. Thus, minimizing the flow time must also be considered.

**Table 2.** Operational sequence for each job in the RFSP.

| Step | WS | Product family A | | | Product family B | | |
|---|---|---|---|---|---|---|---|
| | | **A1** | **A2** | **A3** | **B1** | **B2** | **B3** |
| 1 | WS-1 | $o_1$ | $o_1$ | $o_1$ | $o_1$ | $o_1$ | $o_1$ |
| 2 | WS-2 | $o_{2a}$ | $o_{2a}$ | $o_{2a}$ | $o_{2a}$ | $o_{2a}$ | $o_{2a}$ |
| 3 | WS-3 | $o_3$ | $o_3$ | $o_3$ | $o_3$ | $o_3$ | $o_3$ |
| 4 | WS-2 | $o_{2b}$ | $o_{2b}$ | $o_{2b}$ | $o_{2b}$ | $o_{2b}$ | $o_{2b}$ |
| 5 | WS-4 | $o_4$ | $o_4$ | $o_4$ | $o_4$ | $o_4$ | $o_4$ |
| 6 | WS-2 | - | - | - | $o_{2c}$ | $o_{2c}$ | $o_{2c}$ |
| 7 | WS-5 | $o_5$ | $o_5$ | $o_5$ | - | - | - |
| 8 | WS-6 | $o_{6a}$ | $o_{6a}$ | $o_{6a}$ | $o_{6b}$ | $o_{6b}$ | $o_{6b}$ |
| 9 | WS-7 | $o_{7a}$ | $o_{7b}$ | $o_{7c}$ | $o_{7a}$ | $o_{7b}$ | $o_{7c}$ |

Figure 2 illustrates two objective values; these can be clearly calculated by equations (1) and (2). The first objective value is makespan and the second objective value is mean flow time. A scheduling solution is able to give both Gantt charts as two objectives, so they are conducted to generate the Pareto front for the bi-objective algorithm.

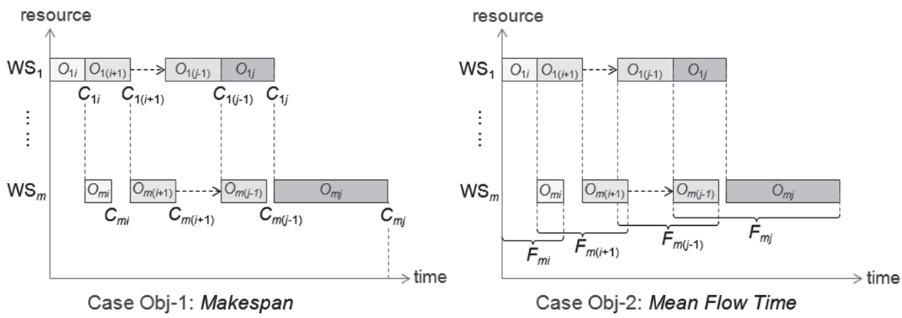


**Figure 2.** Gantt charts of bi-objective RFSP.

*Notation:*

| | |
|---|---|
| $i, j$ | : index of the job ($i \neq j$). |
| $k$ | : index of the workstation. |
| $n$ | : the number of jobs. |
| $m$ | : the maximum workstation. |
| $o_{ki}$ | : the $k^{th}$ operation of job $i$. |
| $C_{1i}$ | : completion time of operation $o_{1i}$ on workstation 1 for job $i$. |
| $C_{1(i\text{-}1)}$ | : completion time of operation $o_{1(i\text{-}1)}$ on workstation 1 for job $i$-1. |
| $C_{mj}$ | : completion time of operation $o_{mj}$ on workstation $m$ for job $j$. |

The objective of the maximum completion time is defined by:

$$Makespan = max\ \{C_{mi}, C_{m(i+1)}, \ldots, C_{mj}\} \qquad (1)$$

The objective of mean flow time is defined by:

$$MeanFlowTime = \frac{1}{n} \sum_{i} F_i \qquad (2)$$

**Bi-objective approach**

The bi-objective scheduling problem here was studied as combinatorial optimization. The problem solution can be obtained from the proposed MOGA procedure, as detailed in Figure 3. During each generation, the chromosomes are constructed and evaluated using a fitness value. Genetic operators will solve the evolutionary process, so the best solution should be found when the procedure is terminated. The Pareto front will be a bi-objective solution for this problem, so it will be built from collected non-dominated solutions.

```
procedure: MOGA for RFS model
input: RFS problem data, GA parameters (popSize, maxGen, p_M, p_C)
output: Pareto optimal solutions
begin
    t ← 0;                                                              // t: generation number
    initialize P(t) by operation-based encoding routine;                // P(t): population
    calculate objectives f_k(P), k = 1,…, q by decoding routine;        // f_k: objective function k
    create Pareto optimal solutions E(P) by nondominated routine;
    evaluate P(t) by fitness assignment routine & keep the best compromised solution;
    while (not terminating condition) do
        create C(t) from P(t) by position-based crossover routine;      // C(t): offspring
        create C(t) from P(t) by swap mutation routine;
        check & repair C(t) by precedence constraint routine for all offspring;
        improve C(t) by left-shifts routine;
        calculate objectives f_k(C), k = 1,…, q by decoding routine;
        update Pareto optimal solutions E(P,C) by nondominated routine;
        evaluate C(t) by fitness assignment routine & update the best compromised solution;
        select P(t+1) from P(t) and C(t) by roulette wheel selection routine;
        t ← t + 1;
    end;
    output the Pareto optimal solutions;
end;
```

**Figure 3.** The MOGA procedure for the RFS model.

**Chromosome representation.** An operation vector ($v_1$) represents each candidate solution for the RFS problem. This research applied an operation-based encoding and decoding procedure. For a chromosome with a gene of one operation, the order of operational sequencing is randomly generated to become a solution string. For chromosome decoding, all ordered operations of a string would be scheduled based on the shortest completion time rule. An operation can be started whenever its predecessor has finished and a machine is available machine at the exact stage. Finally, two objectives are met when the chromosome decoding is completed.

chromosome

| id: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $v_1$ | 1 | 6 | 3 | 4 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 1 | 2 | 3 | 4 | 1 | 7 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 5 | 5 |

| id: | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $v_1$ | 4 | 4 | 4 | 4 | 5 | 3 | 5 | 5 | 5 | 3 | 5 | 5 | 6 | 7 | 6 | 7 | 6 | 7 | 6 | 7 | 1 | 7 | 6 | 7 | 7 | 6 | 2 | 6 |

**Figure 4.** Operation-based encoding.

For example, a candidate chromosome encoded by the operation-based procedure to solve problem no.1 (see Table 3) is shown in Figure 4. Next, operational decoding needs to be done according to the chromosome ($v_1$), such as $S_1$ = {($o_{1,1}$, $o_{6,1}$, $o_{3,1}$, $o_{4,1}$, $o_{1,2a}$, $o_{1,3}$, $o_{2,1}$, $o_{2,2a}$, $o_{3,2a}$, $o_{3,3}$, $o_{4,2a}$, $o_{4,3}$, $o_{1,2b}$, $o_{2,3}$, $o_{3,2b}$, $o_{4,2b}$, $o_{1,4}$, $o_{7,1}$, $o_{1,5}$, $o_{1,6a}$, $o_{2,2b}$, $o_{2,4}$, $o_{2,5}$, $o_{2,6a}$, $o_{3,4}$, $o_{3,5}$, $o_{5,1}$, $o_{5,2a}$, $o_{4,4}$, $o_{4,5}$, $o_{4,6a}$, $o_{4,7c}$, $o_{5,3}$, $o_{3,6a}$, $o_{5,2b}$, $o_{5,4}$, $o_{5,2c}$, $o_{3,7b}$, $o_{5,6b}$, $o_{5,7a}$, $o_{6,2a}$, $o_{7,2a}$, $o_{6,3}$, $o_{7,3}$, $o_{6,2b}$, $o_{7,2b}$, $o_{6,4}$, $o_{7,4}$, $o_{1,7a}$, $o_{7,6b}$, $o_{6,2c}$, $o_{7,2c}$, $o_{7,7c}$, $o_{6,6b}$, $o_{7,7b}$, $o_{6,7b}$,}; it will be used for calculating all objective functions.

**Selection.** Because of the significant effect of an optimal convergence, roulette wheel selection is often found in an efficient genetic algorithm, and in this proposed method. This method assumes that the probability of selection is proportional to the fitness. The wheel will be generated with the cumulative probability of each. A surviving chromosome is selected by randomly choosing a point on the wheel.

**Crossover and mutation.** To produce new chromosomes for the next generation, the crossover operation will act to exchange genes between the parent chromosomes, with the good properties of the parent retained by the offspring; escaping the local minima is the aim of the mutation operator.

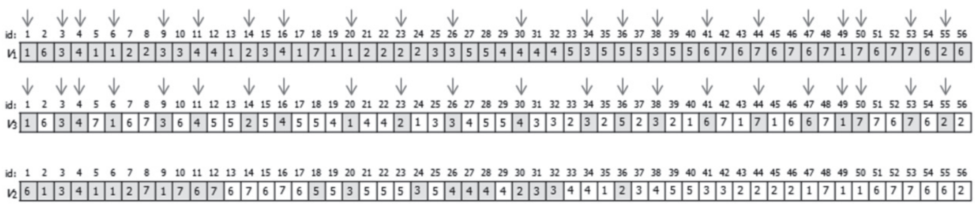

**Figure 5.** Position-based crossover operator.

This study adopted the position-based crossover operator proposed by Syswerda (Gen and Cheng, 2000). It can be used for integer permutation representation, together with a repair procedure. As in Figure 5, some positions of genes from one parent will be randomized and assigned into the new chromosome, and then the remaining positions must be filled with the positions of genes from the other parent by a left-to-right scan.

After the crossover operator, this research used the swap mutation operator. It selects two positions at random, and swaps their genes for the mutant offspring.

**Fitness function.** Solving the RFS problem to minimize the makespan ($f_1$) and mean flow time ($f_2$) simultaneously is a bi-objective optimization, so the fitness function must focus on the two criteria. For each generation, a non-dominated solution will be found and maintained in the set of Pareto.

## RESULTS

**Table 3.** Problem data of product types and number of lots.

| No. | Jobs | Product family A | | | Product family B | | | Total lots |
|---|---|---|---|---|---|---|---|---|
| | | A1 | A2 | A3 | B1 | B2 | B3 | |
| 1. | 7 jobs | $J_1 = 27$ | $J_2 = 36$<br>$J_3 = 33$ | $J_4 = 12$ | $J_5 = 10$ | $J_6 = 43$ | $J_7 = 10$ | 171 |
| 2. | 7 jobs | $J_1 = 16$ | $J_2 = 19$ | $J_3 = 15$ | $J_4 = 30$<br>$J_5 = 26$ | $J_6 = 36$ | $J_7 = 18$ | 160 |
| 3. | 7 jobs | $J_1 = 22$ | $J_2 = 31$ | $J_3 = 26$ | $J_4 = 21$ | $J_5 = 15$ | $J_6 = 23$<br>$J_7 = 28$ | 166 |
| 4. | 7 jobs | $J_1 = 41$ | $J_2 = 30$ | $J_3 = 11$<br>$J_4 = 16$ | - | $J_5 = 15$ | $J_6 = 38$<br>$J_7 = 17$ | 168 |
| 5. | 7 jobs | $J_1 = 15$ | $J_2 = 18$<br>$J_3 = 22$ | - | $J_4 = 34$ | $J_5 = 14$<br>$J_6 = 25$ | $J_7 = 32$ | 160 |
| 6. | 11 jobs | $J_1 = 3$<br>$J_2 = 5$<br>$J_3 = 21$ | $J_4 = 30$<br>$J_5 = 28$ | $J_6 = 6$ | $J_7 = 6$ | $J_8 = 13$<br>$J_9 = 37$ | $J_{10} = 21$<br>$J_{11} = 4$ | 174 |
| 7. | 11 jobs | $J_1 = 10$ | $J_2 = 13$<br>$J_3 = 21$<br>$J_4 = 7$ | $J_5 = 8$<br>$J_6 = 5$ | $J_7 = 24$<br>$J_8 = 20$ | $J_9 = 30$<br>$J_{10} = 12$ | $J_{11} = 12$ | 162 |
| 8. | 11 jobs | $J_1 = 11$ | $J_2 = 20$ | $J_3 = 19$<br>$J_4 = 19$<br>$J_5 = 15$ | $J_6 = 10$<br>$J_7 = 18$ | $J_8 = 4$ | $J_9 = 12$<br>$J_{10} = 17$<br>$J_{11} = 21$ | 166 |
| 9. | 11 jobs | $J_1 = 33$ | $J_2 = 22$<br>$J_3 = 10$ | $J_4 = 3$<br>$J_5 = 8$ | - | $J_6 = 6$<br>$J_7 = 10$<br>$J_8 = 21$ | $J_9 = 29$<br>$J_{10} = 9$<br>$J_{11} = 19$ | 170 |
| 10. | 11 jobs | $J_1 = 4$<br>$J_2 = 7$ | $J_3 = 10$<br>$J_4 = 15$ | - | $J_5 = 26$ | $J_6 = 19$<br>$J_7 = 6$<br>$J_8 = 17$ | $J_9 = 6$<br>$J_{10} = 24$<br>$J_{11} = 26$ | 160 |

All data related to the computational experiment was simplified from a real industry situation (Chamnanlor et al., 2013). A seven-stages flow shop, including the reentrant workstation and controlling-processing workstation, was conducted to test the proposed algorithm. The data set of the number of jobs (including 7 jobs and 11 jobs) is shown in Table 3. They are presented with different lot sizes for each job, and grouped by two families and six sub-groups (A1, A2, A3, B1, B2, and B3).

For solving the bi-objective RFS problem above, this study proposed MOGA. The associated parameters included are the crossover rate ($p_C$ = 0.8), mutation rate ($p_M$ = 0.2), population size (*popSize* = 10), and maximum generation (*maxGen* = 1,000). To obtain the computational results, ten repetitions were conducted for each combination.

In the experimental design, the effectiveness of the proposed algorithm was compared by VESA (Smutnicki et al., 2015) and VEGA (Schaffer, 1985) with two objectives – makespan and mean flow time (see Table 4). All tests were implemented using MATLAB; the program runs on Intel Core i5 CPU 2.27 GHz with 2 GB of RAM.

**Table 4.** Computational results of all algorithms.

| No. | VESA | | | VEGA | | | MOGA | | |
|---|---|---|---|---|---|---|---|---|---|
| | $C_{max}$ [m] | *MFT* [m] | CPU [m] | $C_{max}$ [m] | *MFT* [m] | CPU [m] | $C_{max}$ [m] | *MFT* [m] | CPU [m] |
| 1 | 5,597 | 3,691 | 0.13 | 5,598 | 3,732 | **0.46** | **5,563** | 3,382 | 1.01 |
| 2 | 5,380 | 3,615 | 0.13 | 5,309 | 3,571 | **0.82** | **5,245** | 3,158 | 0.82 |
| 3 | 5,331 | 3,457 | 0.14 | 5,338 | 3,534 | **0.76** | **5,308** | 3,444 | 0.54 |
| 4 | 5,686 | 3,505 | 0.13 | 5,474 | 3,669 | **0.65** | **5,418** | 3,425 | 0.73 |
| 5 | 5,355 | 3,509 | 0.08 | 5,334 | 3,552 | **0.72** | **5,254** | 3,223 | 0.55 |
| 6 | 5,790 | 3,880 | 0.11 | 5,560 | 3,476 | **1.05** | **5,416** | 3,340 | 1.02 |
| 7 | 5,356 | 3,444 | 0.12 | 5,317 | 3,295 | **1.02** | **5,257** | 3,246 | 0.82 |
| 8 | 5,347 | 3,432 | 0.12 | 5,280 | 3,474 | **0.83** | **5,068** | 3,031 | 0.94 |
| 9 | 5,671 | 3,661 | 0.18 | **5,575** | 3,677 | **1.04** | **5,575** | 3,526 | 0.82 |
| 10 | 5,444 | 3,644 | 0.14 | 5,362 | 3,269 | **1.33** | **5,303** | 3,247 | 1.32 |

## DISCUSSION

The production environment in HDD manufacturing is very complicated. To support modern markets, non-traditional flow shops are the current design of choice. The machine configuration often used is a reentrant shop in order to maximize throughput under crucial restrictions, such as controlling-processing time and various product families. Consequently, an efficient scheduling method is an important issue in academic and industrial research. Chamnanlor et al. (2013) studied the reentrant flow shop scheduling problem. A genetic algorithm including local search techniques was used to minimize makespan. Although a hybridized GA can produce an efficient solution without lost lots, it has been studied with only a single objective. However, for some real-world industrial situations, many criteria need to be solved simultaneously. Therefore, the production scheduling approach focuses on multiple objectives.

For reentrant shop scheduling with minimizing bi-objective, several approaches have been considered, including a decentralization heuristic used for multi-objective reentrant shop scheduling (Miragliotta and Perona, 2005) that outperformed practical methods and the crowding distance-based substitution (Rifai et al., 2015) that exhibited better performance and robustness than conventional methods for both small and large datasets. Meanwhile, in genetic algorithm developments, NSGA-II has performed much better at minimizing the makespan and penalty cost in the reentrant flow shop (Sun et al., 2012). In addition, GA-AHP has performed best among the practical methods and the single GA for scheduling problems in semiconductor final testing (Lin et al, 2012). During the last two decades, the meta-heuristic, especially GA, has been famous for considering in the multi-objective solution. VEGA has been well accepted for its robust and reliable performance, but it takes longer to locate the good regions of complex search spaces (Schaffer, 1985). However, the test results of the algorithm clearly show that VESA is comparatively faster, as it evaluates around 31% more solutions per unit of time than the basic algorithm. Moreover, previously performed tests showed that it achieves better results than the classic SA (Smutnicki et al., 2015).

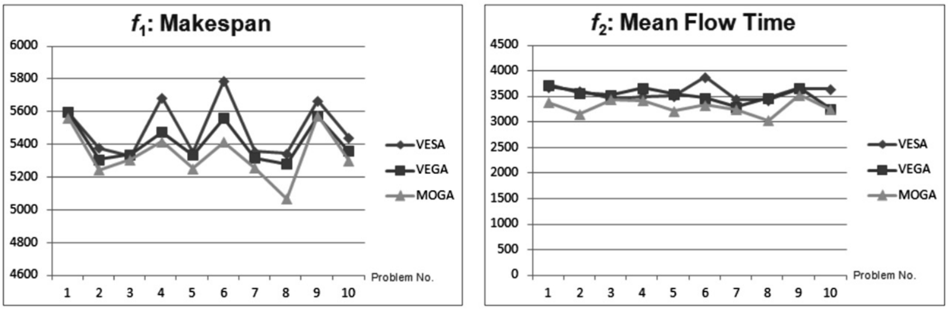


**Figure 6.** Comparison of algorithms with makespan and mean flow time.

Therefore, this study used the extended genetic algorithm for bi-objective optimizations to schedule n jobs in a manufacturing system; the performance was compared with two well-known algorithms – VESA and VEGA. Two objective values in the different methods are shown in the Figure 6. From the results, MOGA efficiently gave a mean flow time on average that was better than VESA and VEGA by 8% and 6%, respectively. Although only slightly different, the averaged makespan obtained from the MOGA was still better than that of 3% and 1% for the VESA and VEGA comparisons. The proposed method in this study provided a quality solution, even though VESA had a shorter computational time (there was no difference with VEGA).

## CONCLUSION

This study presented an application of the multi-objective genetic algorithm (MOGA) for solving the reentrant flow shop (RFS) problem considering the production time control with the dual objectives of minimizing makespan and mean flow time simultaneously. Based on a multiple criteria genetic mechanism, good solutions would be selected appropriately, so non-dominated solutions obtained from each evolution are adopted to construct a Pareto front. A better Pareto front would be kept until the procedure termination was met.

Dimplified test problems were experimented with to evaluate the efficiency of the approaches in HDD manufacturing. The performance of the MOGA approach was compared to other evolutionary algorithms. The experimental results showed that MOGA could solve all problems and deliver a quality solution in a suitable timeframe. The proposed method solved RFS with the best makespan and mean flow time among the three methods tested here. Moreover, the proposed MOGA is a competitive approach that is capable of finding a high-quality Pareto front. In the near future, the expanded reentrant shop considering realistic constraints will be studied further; the best solution will be refined by efficient algorithms, which will be designed in future research.

**ACKNOWLEDGEMENTS**

This work was supported by the Research Unit on System Modeling for Industry (SMI), Khon Kaen University, Thailand.

**REFERENCES**

Chamnanlor, C., K. Sethanan, C.F. Chien, and M. Gen. 2013. Hybrid genetic algorithms for solving reentrant flow-shop scheduling with time windows. Industrial Engineering & Management Systems 12(4): 306-316. doi:10.7232/iems.2013.12.4.306

Chen, J.S., J.C.H. Pan, and C.M. Lin. 2008. A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. Expert Systems with Applications 34: 570-577. doi:10.1016/j.eswa.2006.09.021

Chen, J.S.C., J.C.H. Pan, and C.K.W. Wu. 2007. Minimizing makespan in reentrant flow-shops using hybrid tabu search. International Journal of Advanced Manufacturing Technology 34: 353-361. doi:10.1007/s00170-006-0607-2

Chou, C.W., C.F. Chien, and M. Gen. 2014. A multiobjective hybrid genetic algorithm for TFT-LCD module assembly scheduling. IEEE Transactions on Automation Science and Engineering 11(3): 692-705. doi:10.1109/TASE.2014.2316193

Gen, M., and R. Cheng. 2000. Genetic algorithms and engineering optimization. John Wiley & Sons, New York.

Gen, M., and L. Lin. 2014. Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. Journal of Intelligent Manufacturing 25(5): 849-866. doi:10.1007/s10845-013-0804-4

Gen, M., L. Lin, and J. Gao. 2009. Multistage-based genetic algorithm for flexible job-shop scheduling problem. Intelligent and Evolutionary Systems 187: 183-196. doi:10.1007/978-3-540-95978-6_13

Gao, J., M. Gen, and L. Sun. 2006. Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. Journal of Intelligent Manufacturing 17: 493-507. doi:10.1007/s10845-005-0021-x

Hwang, H., and J.U. Sun. 1997. Production sequencing problem with reentrant work flows and sequence dependent setup times. Computers & Industrial Engineering 33: 773-776. doi:10.1016/S0360-8352(97)00250-7

Jing, C., G. Tang, and X. Qian. 2008. Heuristic algorithms for two machine re-entrant flow shop. Theoretical Computer Science 400: 137-143. doi:10.1016/j.tcs.2008.02.046

Kang, Y.H., S.S. Kim, and H.J. Shin. 2007. A scheduling algorithm for the reentrant shop: an application in semiconductor manufacture. International Journal of Advanced Manufacturing Technology 35: 566-574. doi:10.1007/s00170-006-0736-7

Lin, D., C.K.M. Lee, and Z. Wu. 2012. Integrating analytical hierarchy process to genetic algorithm for re-entrant flow shop scheduling problem. International Journal of Production Research 50(7): 1813-1824. doi:10.1080/00207543.2011.561884

Miragliotta, G., and M. Perona. 2005. Decentralised, multi-objective driven scheduling for reentrant shops: A conceptual development and a test case. European Journal of Operational Research 167: 644-662. doi:10.1016/j.ejor.2004.07.013

Pan, J.C.H., and J.S. Chen. 2003. Minimizing Makespan in Re-Entrant Permutation Flow-Shops. Journal of the Operational Research Society 54: 642-653. doi:10.1016/j.tcs.2011.08.030

Park, Y., S. Kim, and C.H. Jun. 2000. Performance analysis of re-entrant flow shop with single-job and batch machines using mean value analysis. Production Planning & Control 11: 537-546. doi:10.1080/095372800414098

Schaffer, J.D. 1985. Multiple objective optimization with vector evaluated genetic algorithms. In Proceedings of 1st international conference on Gas, pp. 93-100.

Sethanan, K. 2001. Scheduling flexible flowshops with sequence dependent setup times. Ph.D. Dissertation. West Virginia University, West Virginia, USA. doi:10.1109/70.864235

Smutnicki, C., J. Pempera, J. Rudy, and D. Zelazny. 2015. A new approach for multi-criteria scheduling. Computers & Industrial Engineering 90: 212-220. doi:10.1016/j.cie.2015.09.003

Sun, Y., X. Wei, S. Fujimura, and G.K. Yang. 2012. Multi-objective optimization approach with job-based encoding method for semiconductor final testing scheduling problem. Advanced Materials Research 622-623: 152-157. doi: 10.4028/www.scientific.net/AMR.622-623.152

none