International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **3**, June **2018**

 ISSN 1349-4198
pp. 1151–1157

A STUDY ON THE EFFECT OF SETUP TIME OF VARIABLE NEIGHBORHOOD SEARCH IN TWO-STAGE ASSEMBLY FLOW SHOP SCHEDULING PROBLEM

Sunwoong Jung and Byung Soo Kim*

Department of Industrial and Management Engineering
Incheon National University
119, Academy-ro, Songdo-dong, Yeonsu-gu, Incheon 22012, Korea
*Corresponding author: bskim@inu.ac.kr

Received August 2017; revised December 2017

Abstract. *In this study, we developed variable neighborhood search (VNS) algorithm for the two-stage assembly flow shop scheduling problem (TSAFSP) with parallel machining machines. The parallel machining machines produce various types of components and a setup time occurs whenever the machining machine produces a new component or produces different components in the machining stage. The types and numbers of components can be different for each product. In the assembly stage, a single assembly machine can start to assemble a product when the required components for the product must be available. The main decisions are to determine the optimal component-manufacturing and product-assembly schedules to minimize the makespan of the products. In VNS algorithm, we developed two local search heuristics with a batching rule used for a schedule of machining machine and eight types of neighborhoods used for schedules of machining machines as well as assembly machine. The effect of setup time of the VNS algorithm is tested in various problem sizes.*
Keywords: Scheduling, Two-stage assembly, Meta-heuristic, Parallel machines, Setup time

1. **Introduction.** The production scheduling is important to many companies in various aspects such as cost minimization, compliance for due date, and service quality. Due to this reason, many researches for production scheduling about various manufacturing systems have been conducted. Among these many researches, the researches for TSAFSP that was first introduced by Johnson [1] have been conducted continuously because it is applied to many real-life industrial applications. General TSAFSP consists of two consecutive stages, which are a machining stage and assembly stage. In the machining stage, there is a machining machine producing the components required to assemble the product. When the components are available, an assembly machine can assemble these components into the product in assembly stage. The TSAFSP is known as computationally very complex and difficult to find the optimal solution. Many researchers have considered various problem solving methodologies to solve the problem with various manufacturing process structures and assembly process structures. Yan et al. [2] studied a TSAFSP with identical parallel machines for the three-types of objective functions which are weighed sum of maximum makespan, earliness, and lateness. To find the near-optimal solution they proposed a hybrid variable neighborhood search-electromagnetism-like mechanism (VNS-EM) algorithm. Komaki and Kayvanfar [3] studied a TSAFSP with release time of jobs and identical parallel machines. They proposed several heuristic techniques and a meta-heuristic algorithm called Grey Wolf Optimizer. Liao et al. [4] studied a TSAFSP with

batch setup times to minimize the makespan. They developed a mixed integer linear programming (MILP) model, and found several optimality properties to enhance to find the optimal solution. Furthermore, an efficient heuristic based on these optimal properties is proposed. Komaki et al. [5] studied a two-stage hybrid flow shop scheduling problem (HFS) followed by single assembly machine. To find the near-optimal solution they proposed a local search heuristic algorithm and an artificial immune system. Deng et al. [6] studied a distributed two-stage assembly flow shop scheduling problem with makespan minimization criterion and they developed an MILP model. For the large-sized problem, they proposed a competitive memetic algorithm and local search heuristic algorithm. Jung and Kim [7] studied a TSAFSP with dynamic-component sizes, deteriorating jobs, preventive maintenance activities (PMA), and setup time. They developed an MILP model and conducted sensitivity analysis for the deterioration rate, setup time, and PMA time. Huang et al. [8] developed an MILP model for a TSAFSP with parallel batch-processing machines and re-entrant jobs. For the large-sized problem, they proposed three heuristic construction methods with polynomial complexity. Jung et al. [9] studied a TSAFSP with setup time and dynamic-component sizes. They developed an MILP model and several optimal properties. For the large-sized problem, they proposed three genetic algorithms (GA).

Based on the aforementioned studies, no studies considered TSAFSP with identical parallel machining machines for manufacturing various components with dynamic component-sizes and a uniform setup time occurring when different components are manufactured in one of machining machines in the machining stage. In this paper, we consider TSAFSP with identical parallel machining machines requiring a uniform setup time considering dynamic component-sizes in the machining stage and a single assembly machine in the assembly stage. In the machining stage in TSAFSP, parallel machining machines produce various types of components to assemble products. During machining process, a setup time occurs whenever the machining machine produces a new component or produces different types of components. This is the same for all the machining machines. When the required components are available for the associated product, a single assembly machine can assemble these components into the product. A uniform setup time occurs whenever different components are manufactured in one of machining machines. The problem should be to determine the three main decisions, which are the optimal component-manufacturing sequence, the product-assembly sequence, and the number of setups to minimize the makespan assembled to the last product of the assembly stage. Since the machining machines with general purpose are popular in a real-life manufacturing environment, we adopted identical parallel machines to handle various components in the machining stage with dynamic component sizes. In the assembly stage, the type of the components and the number of the components being different in each product can be assembled when the required components have been ready in the machining stage. For an illustrative example of the dynamic component-sizes, three products, P_0, P_1, and P_2 with dynamic component-sizes are given in Figure 1. P_0 has 3 components with two C_1 and one C_3. P_1 has 5 components with one C_0, two C_2, and two C_3. P_2 has 3 components with two C_2 and one C_3. Thus, the component size of each product is dynamically changed depending upon the product.

The paper is organized as follows. Section 2 describes the procedure of VNSs with two local search heuristic algorithms and various neighborhood structures applied to VNSs. In Section 3, the computational experiments are executed to present the influence of setup time on makespan of VNSs by using randomly generated test problems. Finally, the conclusions and future studies are discussed in Section 4.

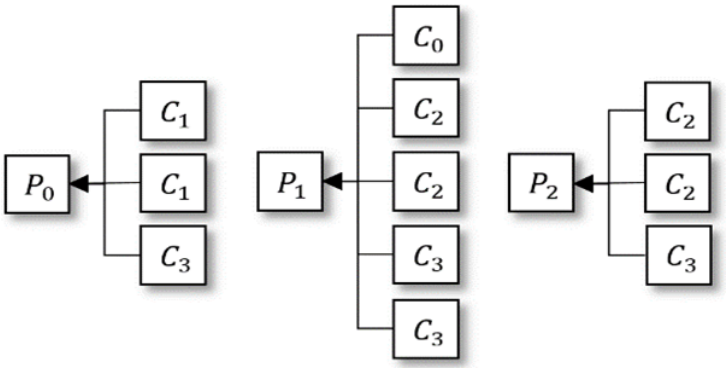

FIGURE 1. An example of dynamic component sizes

2. **Developed Heuristics and VNS.** In this section, we describe a basic procedure of VNS and propose two local search heuristics in VNS. VNS is different from a conventional local search heuristic in that it sequentially uses two or more neighborhood sets, in its structure. In particular, VNS is based on the principle of systematic change of neighborhood sets during the search procedure. The procedure of the basic VNS is presented in Figure 2. It starts with a feasible solution, x. The k is index of neighborhood sets. In each iteration, the solution is randomly determined by the shake procedure obtaining a new solution, x'. And then a local search procedure is applied to x', obtaining a better solution x''. If x'' is better than x, x is updated into x'' as the best solution found and k is set to 1. Otherwise, k is incremented for executing another local search using the next neighborhood set. This process is repeated until a termination criterion is met.

Procedure VNS

```
1: begin
2:     k ← 1;
3:     x = initial solution;
4:     repeat
5:         Shake procedure: find a random
           solution x' ∈ N_k(x);
6:         Perform a local search on N_k(x')
           to find a solution x'';
7:         if f(x'') ≤ f(x) then
8:             x ← x'';
9:             k ← 1;
10:        end if
11:        k = k + 1;
12:    until termination criterion is met
13: end
```

FIGURE 2. The pseudo-code of basic VNS

The solution representation of machining stage consisting of component manufacturing sequence and machine index is represented in 2-dimensional string array. And the assembly stage consisting of product assembly sequence is represented in 1-dimensional string array. Figure 3 shows the example of solution representation for specific complete solution.

In batching manufacturing processing, the reduction for number of setups potentially leads to a decrease in the makespan. Due to this point, we propose a knowledge-based heuristic namely batching local search heuristic (BLS) as a local search in VNS. Also, we propose popular local search heuristics in VNS for parallel-machine scheduling namely neighborhood-based local search heuristic (NLS).

To improve the exploration of VNS, we generate 8 neighborhood combinations ($N_k, k = 1, \ldots, 8$) using 4 operators, that is, *within transfer (WT)*, *within swap (WS)*, *between transfer (BT)*, and *between swap (BS)*. *WT* is an operator to transfer a component or a

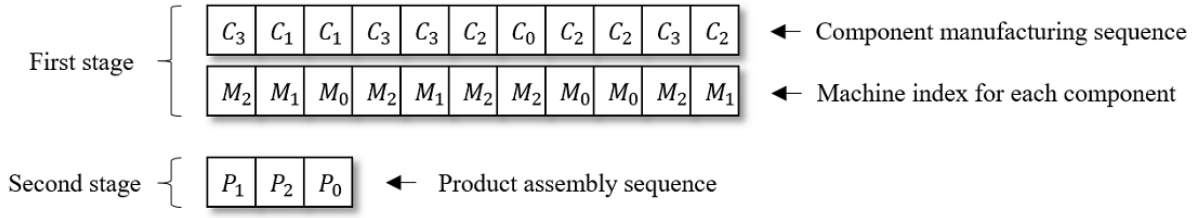

FIGURE 3. The example of solution representation for specific complete solution

product to a random location in the sequence of selected machine. WS is an operator that swaps the positions of two products in a sequence of selected machines. BT is an operator that randomly selects component from one of two randomly selected machines and then randomly transfers the selected components to another machine. BS is an operator that selects one component from each sequence of two selected machines and then interchanges the component. In this paper, N_1 applies WT to the machining stage and WT to the assembly stage called (WT, WT). The rest of neighborhood combinations from N_2 to N_8 are (WT, WS), (WS, WT), (WS, WS), (BT, WT), (BT, WS), (BS, WT), and (BS, WS). In the NLS, we repeat the generating method for each neighborhood until the objective function value is not improved. For instance, the local search for N_1 is conducted WT to the machining stage and WT to the assembly stage until all possible combinations are searched. The procedure of NLS is presented in Figure 4.

Procedure NLS

```
01: begin
02:     k ← 1;
03:     set a neighborhood N_k;
04:     x = a complete initial solution;
05:     repeat
06:         x' = local search for N_k(x);
07:         if f(x') ≤ f(x) then
08:             x ← x';
09:         end if
10:     until all possible combinations are
        searched;
11:     k ← k + 1;
12: end
```

FIGURE 4. The pseudo-code of NLS

The NLS starts with a feasible solution, x in neighborhood N_k. The k is neighborhood index and in each iteration, a local search procedure is applied to x for obtaining a better solution x'. If x' is better than x, x is updated into x' as the best solution found. When the iteration is finished, k is incremented for executing another local search using the next neighborhood combination. This process is terminated until all neighborhood combinations, N_k, $k = 1, \ldots, 8$ are processed. The NLS provides a good solution because the neighborhood combinations can explore a huge search space. However, the computation time is longer due to the neighborhood combinations in NLS. Furthermore, we found that NLS is difficult to escape a local search once the search falls into the local optima. Due to this reason, we propose BLS to be able to reduce the computation time and to search the global optimal solution by focusing on the idea of aggregating the same components as many as possible to reduce the setup time in machining stage because the setup time has a significant effect on the makespan of assembly stage. To apply the BLS, we propose two neighborhoods (SN_k, $k = 1, 2$) for effectively aggregating components in the machining stage. The neighborhoods, SN_k are generated by applying BLS to machining stage and applying *within transfer* (WT) and *within swap* (WS) to assembly stage. Like the NLS,

```
Procedure BLS
01: begin
02:     k ← 1;
03:     set a neighborhood SN_k;
04:     x = a complete initial solution;
05:     select one of machines randomly;
06:     m = a partial solution of
        component scheduling for the
        selected machine;
07:     count the number of each
        component (m);
08:     repeat
09:         select a component having the
            largest requirement among
            unscheduled components;
10:         if there is component that
            has the same requirement then
            select a component having
            bigger processing time;
11:         end if
12:         schedule as many as required
            # selected component;
13:         remove the selected component
            from unscheduled components;
14:     until all components are rescheduled;
15:     m' = a partial solution of component
        scheduling after BLS;
16:     x' = a complete initial solution
        after BLS;
17:     k ← k + 1;
18: end
```

FIGURE 5. The pseudo-code of BLS

the local search for each neighborhood is conducted until the objective function value is not improved. The procedure of BLS for all neighborhood SN_k is presented in Figure 5.

3. **Computational Result.** In order to compare the performance of NLS to that of BLS and identify the effect of setups on NLS and BLS, two computational experiments are conducted using randomly generated test instances. All experiments were executed on a PC with Intel core i7-4770 CPU, 12GB RAM and Windows 10 operating system. We executed computational experiments dividing into two instance cases, which are 100 setup time as small-setup case and 1000 setup time as large-setup time case. For comparing small-setup case with large-setup case, the manufacturing time of components and assembly time of products are randomly generated from $U[10, 30]$ and $U[50, 100]$. The complexity of this problem depends on the schedule of the machining stage due to the number of parallel machines and products. As the number of products increases, the complexity of the machining stage is increased. Thus, the number of types for component, the number of machines, and required number of components for product were gradually increased as the number of products is increased. All the experiments were executed with 30 replications. The test results of developed algorithm are summarized in Tables 1 and 2. It shows the mean of the objective function value (*Mean*), mean absolute deviation (*MAD*), relative percentage deviation (*RPD*) calculated by Equation (1), and computation time (*Time*) according to changing the number of products (P).

$$RPD(\%) = \frac{obj - G\_best}{G\_best} \times 100, \tag{1}$$

where *obj* is the objective function value obtained by developed algorithms and *G_best* is the best solution of all the experiments for each test problem.

The *RPD* is a measure of the deviations between *G_best* and objective function value for each instance and the *MAD* is a measure of the deviation between *G_best* and *Mean*.

In Table 1, the computation time of NLS is dramatically increased by increasing in number of products. This result indicates that the NLS is not suitable for solving the large-sized problems. Also, *RPD* and *MAD* of NLS are lower than BLS.

TABLE 1. The test result of NLS and BLS

	NLS				BLS			
P	*Mean*	*MAD*	*RPD*	*Time*	*Mean*	*MAD*	*RPD*	*Time*
7	985.20	6.84	7.57	106.78	780.50	1.67	4.78	0.01
8	974.00	3.37	13.65	268.17	812.13	2.55	4.71	0.03
9	1227.97	3.44	10.33	648.98	850.07	5.8	8.61	0.04
10	1148.40	5.68	6.73	1097.46	822.23	2.66	3.69	0.06
11	1461.13	5.8	6.65	1912.42	977.80	3.7	7.57	0.12
Avg.		5.03	8.99	806.76	848.55	3.28	5.87	0.05

TABLE 2. The test result of developed algorithm

	Small-setup				Large-setup			
P	*Mean*	*MAD*	*RPD (%)*	*Time (sec.)*	*Mean*	*MAD*	*RPD (%)*	*Time (sec.)*
21	1589.00	4.66	6.86	1.62	5673.37	0.86	2.54	2.47
22	1673.77	4.18	7.71	1.66	5776.37	0.84	2.33	3.06
23	1670.57	3.38	6.07	2.16	5799.53	0.93	1.78	3.70
24	1883.70	4.99	10.29	2.97	5861.90	0.90	1.73	5.34
25	1849.27	3.93	8.40	3.04	5724.73	1.40	2.12	4.61
26	1675.90	2.40	3.32	3.65	5720.20	0.80	1.46	6.42
27	2046.13	3.17	4.18	4.41	5849.87	0.91	2.02	8.59
28	2040.57	4.51	7.12	4.94	5845.07	0.68	1.39	8.35
29	2026.73	3.17	4.96	5.84	5708.80	1.00	1.80	10.87
30	2145.30	1.78	1.77	6.09	5755.13	1.11	1.63	13.43
31	2186.97	1.86	1.62	7.77	5729.03	1.04	1.51	12.82
32	2298.70	1.12	0.73	9.44	5787.43	1.00	1.89	17.25
33	2339.00	4.47	6.90	10.83	5754.13	1.95	2.88	18.87
34	2362.70	3.38	3.81	11.97	5957.87	0.87	2.21	23.69
35	2555.17	3.21	5.19	13.45	5946.23	1.79	2.61	20.37
36	2492.87	3.30	3.91	18.40	5812.33	1.07	1.99	31.22
37	2687.37	3.11	5.30	18.93	5874.57	0.92	2.40	33.95
38	2666.03	1.65	1.45	19.86	5806.67	1.00	2.86	39.45
39	2651.87	2.64	2.23	23.39	6043.63	0.85	1.71	41.83
40	2711.90	1.70	1.53	24.96	5948.83	1.04	1.97	41.79
Avg.		3.13	4.67	9.77		1.05	2.04	17.40

In Table 2, the differences in RPD values of the proposed algorithm for each case are interesting and statistically significant. We found three interesting insights during the experiments. Firstly, *RPD* and *MAD* in all cases were generally low. These results indicate VNS with BLS is robust in various problem environments. Secondly, the *RPD* and *MAD* of large-setup (1.05 and 2.04) case are lower than those of small-setup case (3.13 and 4.67). This phenomenon indicates that the VNS with BLS is more effective in large-setup case so that the effectiveness of the VNS with BLS can be expected in manufacturing systems with very high setup times. Finally, VNS with BLS expects to efficiently apply in manufacturing systems, because the computation times of all cases are less than 20.

4. **Conclusions.** In this paper, TSAFSP with identical parallel machining machines requiring a uniform setup time considering dynamic component-sizes in the machining stage

and a single assembly machine in the assembly stage is considered. We proposed VNSs with two local searches: NLS and BLS. The test results showed that VNS with BLS shows effective and efficient. Furthermore, the algorithm provides robust results in various problem environments in TSAFSP. In the future studies, we will propose a meta-heuristic as a local search heuristic to compare with BLS. Also, we have to develop a mixed integer linear programming (MILP) model for developed problem. Finally, we can extend our current study into sequence-dependent setup and parallel assembly machines.

Acknowledgment. This work was supported by the Incheon National University Research Grant in 2017.

REFERENCES

[1] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, vol.1, pp.61-68, 1954.

[2] H. S. Yan, X. Q. Wan and F. L. Xiong, A hybrid electromagnetism-like algorithm for two-stage assembly flow shop scheduling problem, *International Journal of Production Research*, vol.52, no.19, pp.5626-5639, 2014.

[3] G. M. Komaki and V. Kayvanfar, Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time, *Journal of Computational Science*, vol.8, pp.109-120, 2015.

[4] C. J. Liao, C. H. Lee and H. C. Lee, An efficient heuristic for a two-stage assembly scheduling problem with batch setup times to minimize makespan, *Computers & Industrial Engineering*, vol.88, pp.317-325, 2015.

[5] G. M. Komaki, E. Teymourian and V. Kayvanfar, Minimising makespan in the two-stage assembly hybrid flow shop scheduling problem using artificial immune systems, *International Journal of Production Research*, vol.54, no.4, pp.963-983, 2016.

[6] J. Deng, L. Wang, S. Wang and X. Zheng, A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem, *International Journal of Production Research*, vol.54, no.12, pp.3561-3577, 2016.

[7] S. Jung and B. S. Kim, Novel mathematical models for two-stage assembly flow shop scheduling problem with deterioration and preventive maintenance activities, *ICIC Express Letters, Part B: Applications*, vol.7, no.11, pp.2477-2482, 2016.

[8] J. D. Huang, J. J. Liu, Q. X. Chen and N. Mao, Minimizing makespan in a two-stage flow shop with parallel batch-processing machines and re-entrant jobs, *Engineering Optimization*, vol.49, no.6, pp.1010-1023, 2017.

[9] S. Jung, W. B. Woo and B. S. Kim, Two-stage assembly scheduling problem for processing products with dynamic component-sizes and a setup time, *Computers & Industrial Engineering*, vol.104, pp.98-113, 2017.