Journal of the Korean Institute of Industrial Engineers
Vol. 50, No. 3, pp. 157-172, June 2024.
ISSN 1225-0988 | EISSN 2234-6457

https://doi.org/10.7232/JKIIE.2024.50.3.157
© 2024 KIIE

Capacity Scalability Planning Algorithms for Job-shop-type Reconfigurable Manufacturing Systems with Dynamic Demands

Xuebin Li · Hyeon-Il Kim · Dong-Ho Lee[†]

Department of Industrial Engineering, Hanyang University

개별 공정 형태의 재구성형 제조시스템에 대한 동적 생산용량 결정 알고리즘

리학빈 · 김현일 · 이동호

한양대학교 산업공학과

This study addresses dynamic capacity scalability planning for job-shop-type reconfigurable manufacturing systems (RMSs). The problem is to determine the system components that satisfies the part demands and the minimum workstation utilization in each period of a planning horizon. For the basic case of non-decreasing demands, the previous model is extended by considering a limited number of pallets. After formulating the problem that minimizes the sum of component acquisition and configuration change costs as a nonlinear integer programming model with closed queueing network estimations of part throughputs and workstation utilizations, two backward heuristics are proposed that determine the system components from the last to the first period. Computational results show that they outperform the previous ones significantly. In addition, for the general case of fluctuating demands, two variable neighborhood search (VNS) algorithms are proposed that minimize the sum of component acquisition/removal and configuration change costs, and computational results are reported.

Keywords: Reconfigurable Manufacturing Systems, Capacity Scalability, Dynamic Demands, Backward Heuristics, Variable Neighborhood Search Algorithms

1. Introduction

Reconfigurable manufacturing is an advanced manufacturing paradigm that changes hardware and software components in order to adjust production capacity and functionality exactly in response to market or system changes (Koren *et al.*, 1999). Compared with traditional flexible manufacturing systems (FMSs) with a limited success, reconfigurable manufacturing systems (RMSs) have an intrinsic capability to change system configurations by adding or removing machine tools and other components quickly. In fact, the primary goal of RMS is to establish the exact productivity and flexibility by taking advantages of both dedicated and flexible manufacturing. For more details on the RMS concept and recent literatures, refer to Bortolini *et al.* (2018), Koren *et al.* (2018), Magnaha *et al.* (2019), Yelles-Chaouche *et al.* (2021), Lee and Ryu (2021), Napoleone *et al.* (2023) and Pansare *et al.* (2023).

A key feature of RMS is the reconfigurability that can add, remove and rearrange system components to provide the required capacity and functionality in timely and cost-effective manner. In general, the reconfigurability can be achieved by the core characteristics of modularity, integrability, scalability, convertibility, diagnosability and customization, which can reduce time and cost of reconfigurations as

This work was supported by the Ministry of Science and ICT grant funded by Korea government. (Number: 2022-0-00131)
† Corresponding author: Dong-Ho Lee, Department of Industrial Engineering, Hanyang University, Wangsimni-ro 222, Seongdong-gu, Seoul 04763, Republic of Korea, Tel: +82-2-2220-0475, E-mail: leman@hanyang.ac.kr
Received December 11, 2023; Accepted February 23, 2024.

well as increase system responsiveness. Refer to Napoleone *et al.* (2018) for the details on the core characteristics and their relationships over the manufacturing system life cycle, and Koren *et al.* (2018) for RMS design principles based on the core characteristics.

As in conventional manufacturing systems, the RMS decision problems can be classified into design, operation and control. Among them, the design problem is much different from those for conventional manufacturing systems due to the reconfigurability (Andersen *et al.*, 2017). Specifically,the RMS design can be classified into component- and system-level decisions. The component-level decisions include designs of reconfigurable machine tools, jigs/fixtures and material handling devices (Gadalla and Xue, 2016; Yang *et al.*, 2021), while the system-level decisions include reconfigurability level assessment, layout and configuration/reconfiguration selection(Koren and Shpitalni, 2010; Koren *et al.*, 2018).

Among the RMS design problems, this study addresses system-level configuration/reconfiguration selection that determines production capacity to satisfy dynamic demands by adding or removing system components. Unlike the long-term capacity planning in conventional manufacturing systems, the RMS configuration/reconfiguration selection has features of both long-term design and short-term operation due to the inherent reconfigurability (Yu *et al.*, 2014).

Most previous studies on system-level configuration/reconfiguration selection are done on the flow-shop-type with parallel machines at each stage. As an early study, Son (2000) considers a single-period static problem that determines the numbers of stages and parallel machines at each stage as well as the operation assigned to each stage for single part flow lines, and proposes an integer programming model that minimizes the total capital cost. Then, the static model is extended to a multi-period dynamic one that determines the system configuration in each period using the similarity between two consecutive configurations. See Spicer and Carlo (2007) for other multi-period model in single part flow lines. Also, Wang and Koren (2012) consider a static problem with a fixed number of stages, and propose a genetic algorithm that minimizes the total number of machines while maximizing the system throughput. Later, Koren *et al.* (2017) extend it by considering buffers between stages. Due to the limitation of a single operation assigned to each stage, the above models are extended to the ones that allow multiple operations at each stage. See Dou *et al.* (2009), Moghaddam *et al.* (2018), Zhang *et al.* (2023), Albus and Huber (2023) and Albus *et al.* (2024) for examples.

Due to the limited applications, the above studies are extended to multi-part flow lines. As an early study, Youssef and EIMaraghy (2006) extend the Son's static model and propose a genetic algorithm after an integer programming model is developed. Also, Dou *et al.* (2010) and Moghaddam *et al.* (2018) extend their previous dynamic models for multi-part flow lines. In particular, Moghaddam *et al.* (2020) show the outperformance of the multi-period dynamic approach over the single-period static approach. Besides these, some studies propose multi-criterion optimization approaches that evaluate alternative configurations and select the best one after weighting different criteria. See Goyal *et al.* (2012), Ashraf and Hasan (2018) and Kumar *et al.* (2022) for examples.

Unlike the above ones, this study considers system-level configuration/reconfiguration selection for job-shop-type RMSs with non-unidirectional flows, which is the problem of determining the system components required to satisfy dynamic demands in each period of a planning horizon, called capacity scalability planning in the literature. In fact, this study is an extension of Yu *et al.* (2014) that consider the restricted problem with non-decreasing demands. Specifically, two cases of the problem, i.e. basic case with non-decreasing demands and general case with fluctuating demands, are considered.

For the basic case, the previous model is modified to a more practical one with a limited number of pallets. For the objective of minimizing the sum of component acquisition and configuration change costs, a nonlinear integer programming model is proposed that includes closed queuing network based performance estimation. Then, two backward heuristics are proposed that determine the system configurations from the last to the first period. Computational experiments were done, and the results are reported. In addition, for the general case with fluctuating demands, the basic nonlinear integer programming model is extended for the objective of minimizing component acquisition/removal and configuration change costs. Then, two variable neighborhood search algorithms, ordinary and hybrid ones, are proposed, where the hybrid algorithm allows non-improving moves using the simulated annealing technique. To test the performance of the algorithms, computational experiments were done and the results are reported.

2. Basic case

This section describes the basic case with non-decreasing demands. After formulating the problem as a nonlinear integer programming model, solution algorithms and computational results are presented.

2.1 Problem description

The system considered in this study is a job-shop-type RMS that consists of computer numerical control machines, loading/unloading (L/U) stations, material transporters and a central buffer. Note that the RMS is fundamentally different from the conventional FMSs in that it has the capability to change system components quickly. Figure 1 shows an RMS with three machines and two L/U stations. A system-level reconfiguration from two to three machines is also illustrated in this figure.

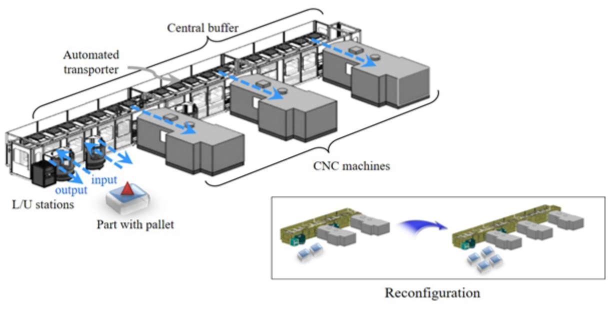


Figure 1. Reconfigurable Manufacturing System with System-level Reconfiguration: Example

In the RMS, each part is processed through one or more machines in non-unidirectional flows after loaded on a pallet at an L/U station and then released into the central buffer. Similarly, a completed part is unloaded from the pallet at an L/U station and then exits from the RMS. The central buffer, which is an automatic storage/retrieval system with a limited number of storage locations, is used to store the pallets. Each machine can process parts using the required cutting tools stored in its tool magazine with a sufficient tool slot capacity. Parts are produced by one or more operations with precedence relations, each of which can be processed on one of the machines at the pre-specified processing workstation. Finally, one or more automated transporters are used to move pallets among the system components, i.e. L/U station, processing workstations and central buffer.

For given non-decreasing demands of multiple product types over a planning horizon, the basic case is to determine the number of additional components, i.e. machines, transporters, L/U stations and pallets, to satisfy the demands in each period of the planning horizon for the objective of minimizing the sum of component acquisition and configuration change costs. The component acquisition costs are those required to acquire components, and the configuration change costs are those required to install newly acquired components. Due to non-decreasing demands, the system components added in a period are maintained during the remaining periods. Without loss of generality, it is assumed that there are no system components at the beginning of the planning horizon. Besides the demand requirements, other constraints are the limited number of pallets and the minimum allowable station utilization. The number of pallets is limited due to the central buffer capacity, which a practical extension of Yu *et al.* (2014). Also, the minimum allowable utilization constraint implies that utilizations of processing workstations and L/U stations must not be less than a lower limit, which is needed to avoid over-installations of unnecessary components.

This study considers a deterministic version of the problem, i.e. all data such as process plans, demands and cost values are deterministic and given in advance. Among the data, the process plans, which contain the information on operations, processing workstations and processing/transportation times, is needed to estimate throughputs and station utilizations under a system configuration.

The problem can be formulated as a nonlinear integer programming model, which modifies Yu *et al.* (2014)'s by adding the limited number of pallets. The notations used are summarized below.

Indices

i	part type, $i=1, 2, ..., I$
t	periods, $t=1, 2, ..., T$
m	stations, $m=1, 2, ..., M$ $(1, 2, ... M-2$: processing workstations; $M-1$: L/U station; and M: transportation station)

Parameters

a_{mt}	acquisition cost of a component at station m in period t
c_{mt}	configuration change cost of station m in period t
h	acquisition cost of a pallet
d_{it}	demand of part type i in period t
$u_{\min}$	minimum allowable station utilization
$q_{\max}$	maximum number of pallets
G	large number

Decision variables

u_{mt}	number of newly acquired components at station m in period t
y_{mt}	= 1 if there is a configuration change at station m in period t, and 0 otherwise
z_t:	number of newly acquired pallets in period t

Now, the nonlinear integer programming model is given below. In the model, $TH_i(\mathbf{X}_t, p_t)$ and $UT_m(\mathbf{X}_t, p_t)$ denote the throughput of part type i and the utilization of station m under configuration $(\mathbf{X}_t, p_t)$ in period t, where $\mathbf{X}_t=(x_{1t}, x_{2t}, ..., x_{Mt})$ and $p_t=\sum_{l=1}^{t} z_l$. Note that $x_{mt}=\sum_{l=1}^{t} u_{ml}$ and p_t denote the numbers of components at station m and pallets in period t, respectively.

[P-NID] Minimize

$$\sum_{t=1}^{T}\left[\sum_{m=1}^{M}(a_{mt}\cdot u_{mt}+c_{mt}\cdot y_{mt})+h\cdot z_t\right]$$

subject to

$$TH_i(\mathbf{X}_t, p_t)\geq d_{it} \qquad \text{for all } i \text{ and } t \qquad (1)$$

$UT_m(\mathbf{X}_t, p_t) \geq u_{\min}$ for all $m = 1, \cdots, M-1$ and t (2)

$u_{mt} \leq G \cdot y_{mt}$ for all m and t (3)

$p_t = \sum_{l=1}^{t} z_l \leq q_{\max}$ for all t (4)

$x_{m0} = z_0 = 0$ for all m (5)

$u_{mt},\ z_t \in Z^+$ for all m and t (6)

$y_{mt} \in \{0, 1\}$ for all m and t (7)

The objective function denotes the sum of component acquisition and configuration change costs over the planning horizon. Constraints (1) and (2) represent the demand and the minimum allowable station utilization requirements, respectively. In this study, part throughputs and station utilizations are estimated using the closed queuing network model of Yu *et al.* (2014), originally proposed by Solberg (1977) for performance evaluation of FMSs. The detailed estimation methods will be explained in the next section. Constraint (3) ensures that a component acquisition occurs in a period if there is a configuration change in that period. Constraints (4) and (5) represent the limited number of pallets and there are no system components at the beginning of the planning horizon, respectively. Finally, the remaining constraints represent the conditions of decision variables.

It can be easily seen that the problem [P-NID] is difficult to solve optimally for large-sized instances due to the exponential number of possible configurations and the nonlinear closed queuing network based estimation functions $TH_i(\mathbf{X}_t, p_t)$ and $UT_m(\mathbf{X}_t, p_t)$. Therefore, instead of the optimal approach with very limited practical applications, the heuristic approach is adopted in this study.

2.2 Solution algorithms

This section presents the heuristic algorithms proposed in this study. Before explaining the heuristics, the closed queuing network (CQN) model is briefly explained.

(1) Estimating throughputs and utilizations

The CQN model represents an RMS configuration $(\mathbf{X}_t, p_t)$ as M inter-connected stations with one or more identical components. <Figure 2> shows the CQN model with processing stations $(1, 2, \ldots, M-2)$, an L/U station $(M-1)$ and a transportation station (M), where the transportation station takes over the role of a central server activated after each individual processing operation is finished on a machine at a processing station. The assumptions are: (a) exponential processing/transportation times; (b) new parts available at all times; (c) first come first served queueing discipline; (d) universal pallets that can load all part types being processed in the RMS; and (e) sufficient central buffer capacity. See Solberg (1977) for more details on the CQN topology.

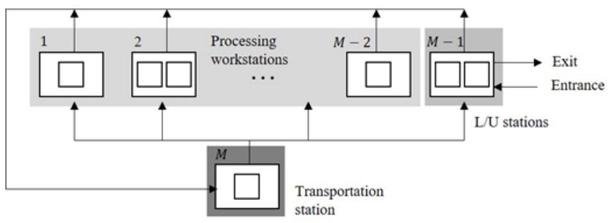


Figure 2. Closed Queuing Network (CQN) Model for an RMS Configuration

For a given RMS configuration $(\mathbf{X}_t, p_t)$, let n_{mt} be the number of pallets located at station m in period t and hence $p_t = \sum_{m=1}^{M} n_{mt}$. Also, let w_{mt} be the average workload at station m in period t, which can be calculated using the process plans. Then, the probability that the RMS is in state $\boldsymbol{n}_t = (n_{1t}, n_{2t}, \ldots, n_{Mt})$ can be represented as

$$P(\boldsymbol{n}_t) = P(n_{1t}, n_{2t}, \ldots, n_{Mt}) = \frac{1}{g(p_t, M)} \Pi_{m=1}^{M} f_m(n_{mt}) \ ,$$

where $f_m(n_{mt}) = w_{mt}^{n_{mt}} / n_{mt}!$ if $n_{mt} \leq x_{mt}$ and $w_{mt}^{n_{mt}} / x_{mt}! \cdot (x_{mt})^{n_{mt} - x_{mt}}$ otherwise. Note that $g(p_t, M)$ is the normalization constants over the state space, which can be calculated by the Buzen algorithm. See Tempelmeier and Kuhn (1993) for more details on the Buzen algorithm.

Then, the throughput of part type i can be estimated as

$$TH_i(\mathbf{X}_t, p_t) = \alpha_i \cdot \frac{[g(p_t - 1, M)/g(p_t, M)]}{v_M} \ ,$$

where α_i and v_M denote the production ratio of part type i and the average number of operations per part, respectively. Also, the utilization of station m can be estimated as

$$UT_m(\mathbf{X}_t, p_t) = b_m \cdot \frac{r_m \cdot [g(p_t - 1, M)/g(p_t, M)]}{x_m} \ ,$$

where b_m and r_m denote the average processing time per operation and the relative arrival frequency (visit ratio) of a part at station, respectively. See Tempelmeier and Kuhn (1993) for more details on the derivations of the above two formulas.

(2) Heuristic algorithms

Modified backward-utilization (MB-UT) heuristic

The MB-UT heuristic is a modification of Yu *et al.* (2014)'s that determines the configurations from the last to the first period. Unlike the previous one that determines the last period configuration using a simple greedy heuristic, MB-UT uses a local search method. The detailed procedure is given below. In the procedure, $\mathbf{1}_m$ represents a vector with 1 in the m th place and other elements are 0.

```
Procedure 1. (MB-UT heuristic)
Step 1. Determine the last period configuration as follows.
    (a) Initialize $\mathbf{X}_T = (1,1, ..., 1)$ and $p_T = q_{max}$.
    (b) Select station $m'$ such that
        $m' = \mathrm{argmax}_{m=1,2,...,M}\{ UT_m(\mathbf{X}_T + \mathbf{1}_m, p_T)\}$
        and increase the number of components at station $m'$
        by 1. Do this until a feasible configuration is obtained.
    (c) Decrease $p_T$ to the minimum possible one.
Step 2. Determine the configurations of the preceding periods as
        follows.
    (a) Set $t = T-1$.
    (b) Initialize $\mathbf{X}_t = \mathbf{X}_{t+1}$ and set the number of pallets in
        period $t$ to 0.
    (c) For the configuration $\mathbf{X}_t$, if a feasible configuration
        can be obtained by increasing the number of pallets before reaching $p_{t+1}$, i.e. number of pallets in the directly succeeding period, set the configuration in period $t$ as $(\mathbf{X}_t, p_t)$ and go to (e). Otherwise, go to (d).
    (d) Select a station $m^*$ such that
        $m^* = \mathrm{argmin}_{m=1,2,...,M}\{ UT_m(\boldsymbol{X}_t - \mathbf{1}_m, p_t)\}$
        and set $x_{m^*t} = x_{m^*t} - 1$ and go to (c).
    (e) If $t < 2$, stop. Otherwise, set $t = t-1$ and go to (b).
```

Modified backward-throughput (MB-TH) heuristic

The MB-TH heuristic is the same as the MB-UT except for Step 1(b) of procedure 1, i.e. priority rule that selects the station for which the number of components is increased when determining the last period configuration. Specifically, the MB-TH heuristic selects the station with the maximum increase in throughput divided by the sum of unit component acquisition and configuration change costs, i.e. select station m' such that

$$m' = \mathrm{argmax}_{m=1,2,...,M}\left\{ \frac{TH(\mathbf{X}_T + \mathbf{1}_{m,p_T}) - TH(\mathbf{X}_T, p_T)}{a_{mT} + c_{mT}} \right\}.$$

2.3 Computational Results

To test the performance of the heuristics proposed in this study, computational experiments were done and the results are reported in this section. Specifically, the two heuristics, MB-UT and MB-TH, are compared with the forward-throughput (F-TH), forward-utilization (F-UT) and backward-utilization (B-UT) heuristics of Yu *et al.* (2014) after setting the limited number of pallets. The heuristics were coded in Python 3.1 and the experiments were done on a PC with an Intel core i9 processor at 3.42 GHz clock speed with 64GB RAM memory.

The first experiment was done to show the absolute performance for small sized test instances. For the test, 30 instances with three periods and five stations were generated randomly, i.e. 10 instances for each of three levels for the number of part types (10, 20 and 30) using the data of Yu *et al.* (2014) and Zhou *et al.* (2014). The data were generated as follows.

- Demands in each period $(d_{it}) \sim DU(50,70) + DU(0,10)$, where $DU(0,10)$ represents the amount of an additional demand
- Component acquisition costs $(a_{mt}) \sim DU(5000,20000)$
- Configuration changing costs $(c_{mt}) \sim DU(500,2000)$
- Pallet costs $(h) \sim DU(200,300)$
- Number of operations for a part type ~ $DU(6,16)$
- Operation processing times ~ $DU(20,100)$
- Loading/unloading times ~ $DU(5,20)$
- Transportation times ~ $DU(3,7)$

In addition, the limited number of pallets (q_{max}) was set to 60. The performance measures are: (a) percentage gaps from the optimal solution values that obtained using the full enumeration method and (b) CPU seconds.

Test results are summarized in <Table 1> (a), (b) and (c) that show the average percentage gaps from optimal solution values for three levels of the minimum allowable station utilization (0.6, 0.7 and 0.8). It can be seen from the tables that the heuristics proposed in this study outperform the previous ones in overall averages due to the better last period configurations. Of the heuristics, MB-TH was significantly better than MB-UT because it considers both throughputs and costs. In fact, the overall average gaps of MB-TH were 2.47%, 2.9% and 2.11% when the minimum allowable station utilization was 0.6, 0.7 and 0.8, respectively. To test statistical difference among the heuristics, paired t-tests were done using the average percentage gaps and the results showed that MB-TH is statistically different from the others in the significance level of 0.01. Finally, all the heuristics solve the test instances within 8.2 seconds while the full enumeration method required 2108.0 seconds in overall average.

The second experiment was done for medium-to-large sized test instances. For the test, 240 instances, i.e. 10 instances for each of 24 combinations of three levels for the number of periods (3, 5 and 10), three levels for the number of stations (5, 7 and 9) and three levels for the number of part types (10, 20 and 30), were generated using the data generation method explained earlier. The limited numbers of pallets were set to 60, 80 and 100, and the

Table 1. Basic Case: Results for the Heuristics on Small Sized Test Instances

(a) Minimum allowable station utilization = 0.6

NP[1]	NS[2]	NPT[3]	F-TH	F-UT	B-UT	MB-UT	MB-TH
3	5	10	7.16[*]	17.38	22.87	5.74	1.92
		20	5.57	12.16	11.45	5.55	3.41
		30	4.35	13.50	11.86	9.41	2.07
Average			5.69	14.35	15.39	6.90	2.47

[1]Number of periods; [2]Number of stations; [3]Number of part types

[*] Average percentage gap from optimal solution values out of 10 instances.

(b) Minimum allowable station utilization = 0.7

NP	NS	NPT	F-TH	F-UT	B-UT	MB-UT	MB-TH
3	5	10	12.19	30.11	18.96	6.33	2.61
		20	5.28	12.51	10.76	5.50	3.13
		30	5.25	14.44	14.11	10.83	2.96
Average			7.58	19.02	14.61	7.55	2.90

(c) Minimum allowable station utilization = 0.8

NP	NS	NPT	F-TH	F-UT	B-UT	MB-UT	MB-TH
3	5	10	14.78	48.02	16.25	4.53	0.71
		20	4.97	12.16	10.42	5.14	2.76
		30	5.14	14.35	14.01	10.29	2.85
Average			8.30	24.85	13.56	6.65	2.11

minimum allowable station utilizations were set to 0.6, 0.7 and 0.8 for the test instances with 5, 7 and 9 stations, respectively. The performance measures are: (a) relative performance ratio (RPR) since the optimal solutions could not be obtained; and (b) CPU seconds, where the RPR of a heuristic for an instance is calculated as

$$100 \cdot (C_a - C_{best}) / C_{best},$$

where C_a is the total cost value obtained from heuristic a and C_{best} is the best one among those obtained from all the heuristics.

Test results are summarized in <Tables 2> (a), (b) and (c) that show the average RPR values of the heuristics for the three levels of the minimum allowable station utilization. As in the test results for the small sized instances, the new heuristics perform better than the previous ones in overall averages and MB-TH performs the best. Paired *t*-tests were also done and the results showed that MB-TH is statistically better than the others in the significance level of 0.01, which shows the effectiveness of the throughput/cost ratio based local search that determines the better last period configurations. Finally, all the heuristics gave the solutions within 24.5 seconds.

Table 2. Basic Case: Results for the Heuristics on Medium-to-Large Sized Test Instances

(a) Minimum allowable station utilization = 0.6

NP	NS	NPT	F-TH	F-UT	B-UT	MB-UT	MB-TH
3	7	10	17.70[*]	27.48	66.07	2.96	0.00
		20	7.67	17.02	19.87	10.46	0.15
		30	9.33	11.15	8.84	1.88	0.00
	9	10	63.05	52.48	144.10	2.07	0.00
		20	22.44	31.36	72.09	1.13	0.00
		30	18.45	20.78	32.46	3.00	0.00

Table 2. Basic Case: Results for the Heuristics on Medium-to-Large Sized Test Instances (Continued)

NP	NS	NPT	F-TH	F-UT	B-UT	MB-UT	MB-TH
5	5	10	9.37	17.63	17.36	3.90	0.22
		20	2.37	8.04	7.78	2.81	0.99
		30	3.45	8.69	5.02	2.26	0.46
	7	10	15.93	28.09	69.43	3.28	0.00
		20	4.53	22.02	22.14	10.43	0.23
		30	6.46	11.14	9.58	3.29	0.32
	9	10	67.84	50.72	137.91	1.61	0.00
		20	17.13	30.71	69.02	2.16	0.07
		30	16.78	21.13	37.92	0.29	0.18
10	5	10	6.29	23.47	17.35	9.92	1.26
		20	2.52	12.51	9.32	5.24	1.30
		30	0.81	7.44	5.57	2.94	0.98
	7	10	11.56	13.13	49.15	1.30	0.00
		20	3.47	11.97	9.69	2.43	0.78
		30	4.44	10.72	8.84	2.55	1.09
	9	10	51.29	50.07	125.99	0.89	0.00
		20	14.75	20.03	45.91	2.85	0.09
		30	10.09	21.68	21.63	2.94	0.54
Average			16.15	22.06	42.21	3.44	0.36

See the footnotes of <Table 1>.
[*] Average relative performance ratio out of 10 instances

(b) Minimum allowable station utilization = 0.7

NP	NS	NPT	F-TH	F-UT	B-UT	MB-UT	MB-TH
3	7	10	61.32	70.37	55.56	5.13	0.00
		20	9.35	21.94	18.06	10.66	0.15
		30	9.13	10.95	8.74	1.69	0.01
	9	10	135.20	119.46	116.27	4.60	0.00
		20	98.90	83.92	80.85	0.98	0.00
		30	44.56	40.81	35.63	2.64	0.00
5	5	10	11.78	24.64	17.96	3.87	0.24
		20	2.00	7.65	6.91	3.04	0.43
		30	3.74	8.99	5.89	2.44	0.42
	7	10	53.94	78.96	71.20	3.19	0.91
		20	6.06	26.21	19.14	9.96	0.15
		30	6.56	11.34	9.69	3.27	0.41
	9	10	139.22	131.30	125.25	0.00	0.00
		20	90.53	82.52	77.28	1.90	0.07
		30	58.71	51.92	47.21	0.29	0.15
10	5	10	5.94	21.67	11.50	8.63	1.62
		20	1.72	11.67	10.21	7.15	1.97
		30	0.70	7.34	5.93	3.38	0.98
	7	10	44.42	67.59	59.57	0.83	0.50
		20	4.23	9.99	8.90	2.62	0.88
		30	4.03	10.52	8.36	3.71	1.65
	9	10	151.01	138.38	130.32	1.91	0.18
		20	76.57	65.37	59.03	3.64	0.43
		30	22.79	26.38	19.53	3.46	0.37
Average			43.43	47.08	42.04	3.71	0.48

Table 2. Basic Case: Results for the Heuristics on Medium-to-Large Sized Test Instances (Continued)

(c) Minimum allowable station utilization = 0.8

NP	NS	NPT	F-TH	F-UT	B-UT	MB-UT	MB-TH
3	7	10	77.09	107.82	50.05	6.26	0.00
		20	21.00	33.57	17.74	10.66	0.17
		30	9.10	13.87	8.71	1.62	0.01
	9	10	225.18	176.84	94.58	0.09	0.00
		20	206.88	167.74	77.45	1.74	0.00
		30	134.31	105.22	35.51	3.07	0.00
5	5	10	15.88	36.53	17.45	4.75	0.91
		20	2.30	7.98	7.24	3.51	0.72
		30	3.30	8.53	5.44	2.69	0.49
	7	10	71.60	98.09	57.42	2.55	0.00
		20	19.32	28.29	19.37	10.87	0.00
		30	6.79	12.00	9.53	3.12	0.03
	9	10	222.33	182.72	91.03	4.43	0.00
		20	197.32	167.25	76.91	1.92	0.07
		30	158.71	118.16	47.61	0.22	0.19
10	5	10	8.42	22.79	10.95	7.48	0.52
		20	1.46	11.34	9.87	7.49	2.87
		30	0.86	7.47	6.11	3.57	1.49
	7	10	73.49	98.07	55.73	0.00	0.00
		20	6.00	13.73	8.80	1.79	0.10
		30	4.35	11.39	9.01	3.66	2.18
	9	10	244.31	187.21	112.57	2.45	0.00
		20	231.35	142.74	58.51	3.81	0.12
		30	103.97	57.12	19.06	3.32	0.50
Average			85.22	75.69	37.78	3.80	0.43

3. General Case

This section describes the general case with fluctuating demands. After the basic nonlinear integer programming model is extended, two variable neighborhood search algorithms and their test results are presented.

3.1 Problem Description

The general case is the same as the basic one except that the demands are fluctuating, i.e. increasing or decreasing, over a planning horizon. Therefore, the general case can be defined as follows. For given fluctuating demands of multiple product types, the problem is to determine the number of system components to satisfy the demands in each period of a planning horizon for the objective of minimizing the sum of component acquisition/removal and configuration change costs. As in the basic case, the constraints are demand requirements, the limited number of pallets and the minimum allowable station utilization. It is assumed that the pallets are acquired at the beginning of the planning horizon.

Let s_{mt} and g_{mt} denote the removal cost of a component and the number of removed components at station m in period t, respectively. Then, the general case can be formulated as the following nonlinear integer programming model.

[P-FD] Minimize

$$\sum_{m=1}^{M}\sum_{t=1}^{T}(a_{mt}\cdot u_{mt}+c_{mt}\cdot y_{mt}+s_{mt}\cdot g_{mt}) + h\cdot \max_{(t=1,\dots,T)}\{p_t\}$$

subject to

$TH_i(\mathbf{X}_t, p_t) \geq d_{it}$	for all i and t	(1)
$UT_m(\mathbf{X}_t, p_t) \geq u_{\min}$	for all $m=1, \dots, M-1$ and t	(2)
$u_{mt}+g_{mt} \leq G\cdot y_{mt}$	for all m and t	(3')
$0 \leq \max_{t=1,\dots,T}\{p_t\} \leq q_{\max}$		(4')
$u_{m0}=g_{m0}=0$	for all m	(5')
$u_{mt}\cdot g_{mt}=0$	for all m and t	(6')
$u_{mt}, g_{mt} \geq 0$ and integers	for all m and t	(7')
$y_{mt} \in \{0,1\}$	for all m and t	(8)

The objective function represents the sum of component acquisition/removal and configuration change costs over the planning horizon. Constraint (3') ensures that a component acquisition/removal occurs in a period if there is a configuration change in that period. Constraints (4') and (5') represent the limited number of pallets and there are no system components at the beginning of the planning horizon, respectively. Constraint (6') ensures that acquisition and removal of the same component cannot occur at the same time in a period. Finally, the remaining constraints (7') and (8) represent the conditions of decision variables.

It can be easily seen that the problem [P-FD] is harder than [P-NID] because components can be removed according to fluctuating demands and hence the solution space gets much larger. Therefore, instead of simple local search heuristics, this study adopts the meta-heuristic approach.

3.2 Solution Algorithms

This section explains the two variable neighborhood search algorithms proposed in this study, i.e. ordinary and hybrid ones. Variable neighborhood search (VNS), proposed by Mladenović and Hansen (1997), is a meta-heuristic that explores a set of predefined neighborhood structures successively to escape from local optimums. See Hansen *et al.* (2017) for more details on the VNS algorithm.

(1) Ordinary VNS algorithm

An overview of the ordinary VNS algorithm is shown in Figure 3. As can be seen in the figure, the algorithm consists of four main steps: (a) obtaining an initial solution; (b) generating shaking solutions using different neighborhood structures; (c) local search that improves the shaking solution and (d) termination. In the algorithm, a solution is represented by an $M \times T$ matrix X$=[x_{mt}]$, where x_{mt} denotes the number of components at station m in period t. Therefore, the t th column vector $\mathbf{X}_t = (x_{1t}, x_{2t}, ..., x_{Mt})^T$ represents the configuration with p_t pallets in period t. The details of each step are explained below.

Step 1. Obtaining an initial solution

An initial solution, denoted as $\mathrm{X}^i = \left[x^i_{mt}\right]$, is obtained by determining the feasible configuration in each period using the first step of the MB-UT heuristic, i.e. Step 1 of procedure 1.

Step 2. Shaking

Shaking is done to escape from the local optimums by changing neighborhood structures. More specifically, from the current incumbent solution X, a shaking solution is obtained as follows. First,

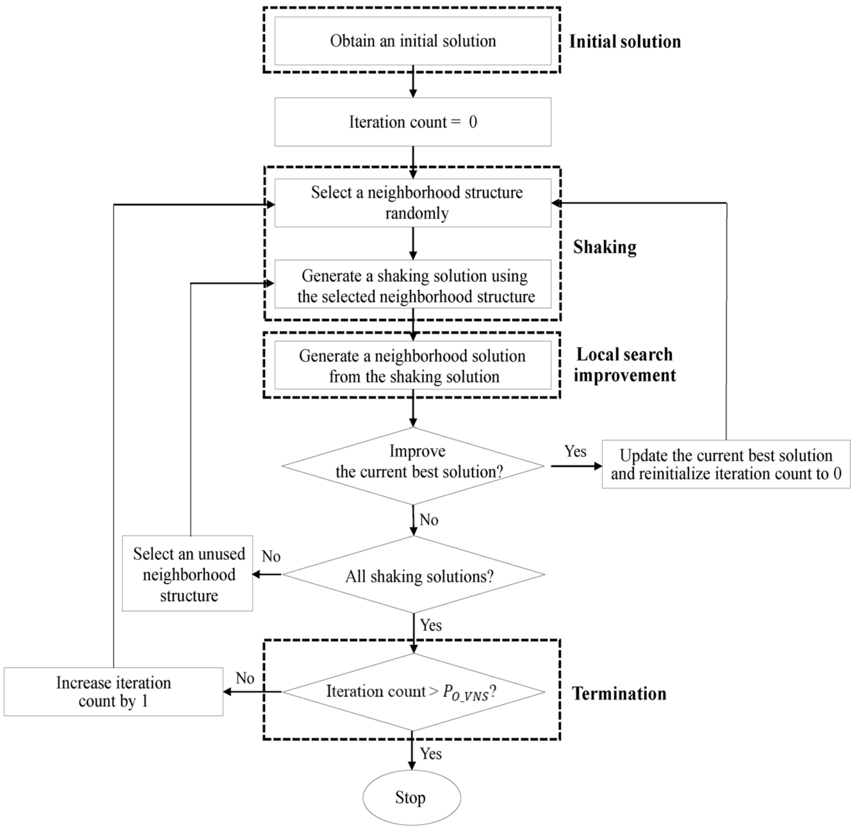

Obtain an initial solution
Initial solution
Iteration count = 0
Select a neighborhood structure randomly
Shaking
Generate a shaking solution using the selected neighborhood structure
Generate a neighborhood solution from the shaking solution
Local search improvement
Improve the current best solution?
Yes
Update the current best solution and reinitialize iteration count to 0
No
Select an unused neighborhood structure
No
All shaking solutions?
Yes
Increase iteration count by 1
No
Iteration count > P_{O_VNS}?
Termination
Yes
Stop

Figure 3. Ordinary VNS Algorithm: Overview

a neighborhood structure $\mathcal{N}'(\mathrm{X})$ is selected randomly. Then, a feasible solution X^f that satisfies the demand and the minimum allowable utilization constraints is generated randomly using the selected neighborhood structure $\mathcal{N}'(\mathrm{X})$, where $\mathrm{X}^f \in \mathcal{N}'(\mathrm{X})$. Finally, a feasible shaking solution X' is obtained by decreasing the number of pallets in X^f to the minimum possible one.

In this study, the following neighborhood structures are proposed.

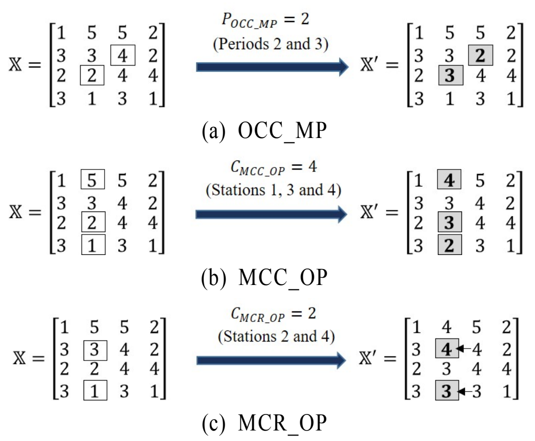
One component change in multiple periods (OCC-MP). This method generates a feasible neighborhood solution by selecting P_{OCC_MP} different periods randomly and then changing the number of components at a random station in each of the periods in such a way that the changed number does not exceed the maximum number of components in the initial solution, i.e. $\max_{m,t}\{x^i_{mt}\}$. <Figure 4 (a)> shows an example with $P_{OCC\_MP} = 2$, in which periods 2 and 3 are selected and then the number of components at station 3 (2) in period 2 (3) is changed from 2 (4) to 3 (2).

Multiple component changes in one period (MCC-OP). This method generates a feasible neighborhood solution by selecting a period randomly and then adding or removing one component randomly at each of C_{MCC_OP} random stations in that period. <Figure 4(b)> shows an example with $C_{MCC\_OP} = 3$, in which period 2 is selected and then one component is removed at station 1 while added at stations 3 and 4.

Multiple component replacements in one period (MCR-OP). This method generates a feasible neighborhood solution by selecting a period randomly and then replacing its number of components at each of C_{MCR_OP} random stations by that of the random adjacent period. <Figure 4 (c)> shows an example with $C_{MCR\_OP} = 2$, in which period 2 is selected and then the number of components at station 2 (4) is replaced by 4 (3) in the adjacent period 3.

Step 3. Local search improvement

This step improves the shaking solutions by the neighborhood structures explained earlier. Specifically, a neighborhood solution is generated from the current shaking solution X' using each of OCC_MP, MCC_OP and MCR_OP methods, and the best one X'' is selected. If X'' improves the current best solution, the best solution is updated and the next shaking is done. Otherwise, the local search is done for another shaking solution generated by an unused neighborhood structure. Note that the iteration count is reinitialized to 0 if an improved solution is obtained by the consecutive shaking and local search improvement steps, while increased by 1, otherwise.



$P_{OCC\_MP} = 2$
(Periods 2 and 3)
$\mathbb{X} = \begin{bmatrix}1&5&5&2\\3&3&\boxed{4}&2\\2&\boxed{2}&4&4\\3&1&3&1\end{bmatrix} \longrightarrow \mathbb{X}' = \begin{bmatrix}1&5&5&2\\3&3&\boxed{2}&2\\2&\boxed{3}&4&4\\3&1&3&1\end{bmatrix}$
(a) OCC_MP
$C_{MCC\_OP} = 4$
(Stations 1, 3 and 4)
$\mathbb{X} = \begin{bmatrix}1&\boxed{5}&5&2\\3&3&4&2\\2&\boxed{2}&4&4\\3&\boxed{1}&3&1\end{bmatrix} \longrightarrow \mathbb{X}' = \begin{bmatrix}1&\boxed{4}&5&2\\3&3&4&2\\2&\boxed{3}&4&4\\3&\boxed{2}&3&1\end{bmatrix}$
(b) MCC_OP
$C_{MCR\_OP} = 2$
(Stations 2 and 4)
$\mathbb{X} = \begin{bmatrix}1&5&5&2\\3&\boxed{3}&4&2\\2&2&4&4\\3&\boxed{1}&3&1\end{bmatrix} \longrightarrow \mathbb{X}' = \begin{bmatrix}1&4&5&2\\3&\boxed{4}\leftarrow&4&2\\2&3&4&4\\3&\boxed{3}\leftarrow&3&1\end{bmatrix}$
(c) MCR_OP

Figure 4. Neighborhood Structures: Examples

Step 4. Termination

The ordinary VNS algorithm is terminated when there is no improvement for a certain number P_{O_VNS} of consecutive iterations.

(2) Hybrid VNS Algorithm

The hybrid VNS algorithm is the same as the ordinary one except that the simulated annealing (SA) technique is used to allow non-improving moves in the local search improvement. Let X^* and X'' denote the current best solution and the new solution obtained by the shaking and local search methods, respectively. Then, in the local search improvement step, the new solution X'' is accepted if it improves the current best one X^*. Otherwise, it is accepted with a specified probability $\exp(-\Delta/Temp)$, where Δ and $Temp$ denote the difference in objective values of the new and the current best solutions, i.e. $\Delta = Z(\mathrm{X}'') - Z(\mathrm{X}^*)$, and the temperature, respectively. $Z(\mathrm{X})$denotes the objective value of solution X.) As in the ordinary SA algorithm, the temperature is decreased by $Temp_l = \beta \cdot Temp_{l-1}$, where β denotes a positive cooling ratio less than 1 and $Temp_l$ is the temperature during lth epoch, i.e. the number Γ of movements made with the same temperature. Note that another shaking solution is generated using an unused neighborhood structure if the new solution is not accepted and the iteration count is increased by 1 when no improved solution can be obtained by all shaking solutions. Finally, the hybrid VNS algorithm is terminated when no improvement occurs for a certain number P_{H_VNS} of consecutive iterations.

3.3 Computational Results

To test the performance of the VNS algorithms, computational experiments were done and the results are reported in this section. The VNS algorithms were coded in Python 3.1 and the tests were done on a PC with an Intel core i9 processor at 3.42 GHz clock speed with 64GB RAM memory.

Before the main tests, a preliminary test was done to set the parameters of the hybrid VNS algorithm. Specifically, the best one was selected using RPR values after comparing six combinations of three levels for cooling ratio (0.6, 0.7 and 0.8) and two levels for epoch length (6 and 8) when the initial temperature and the termination parameter were set 10000 and 30, respectively. For the test, 10 small sized (3 periods, 5 stations and 10 part types), 10 medium sized (5 periods, 7 stations and 20 part types)and 10 large sized instances (10 periods, 9 stations and 30 part types) were generated randomly. The data were generated as follows.

- Demands in each period (d_{it}) ~ $DU(10,150)$
- Component acquisition costs (a_{mt}) ~ $DU(10000,20000)$
- Component removal costs (s_{mt}) ~ $DU(5000,10000)$
- Pallet costs (h) ~ $DU(200,300)$
- Configuration changing costs (c_{mt}) ~ $DU(500,2000)$
- Number of operations for a part type ~ $DU(6,16)$
- Operation processing times ~ $DU(20,100)$
- Loading/unloading times ~ $DU(5,20)$
- Transportation times ~ $DU(3,7)$

In addition, the limited numbers of pallets (q_{max}) were set to 60, 80 and 100 for the instances with 5, 7 and 9 stations, respectively, and the minimum allowable station utilization was set to 0.7. Test results are summarized in Table 3 that shows the average RPR values for all parameter combinations. From the test results, the cooling ratio (β) and the epoch length (Γ) were set to 0.7 and 8, respectively.

The first test was done to show the absolute performance of the VNS algorithms for small sized test instances. The performance measures are: (a) percentage gaps from the optimal solution values; and (b) CPU seconds, where the optimal solutions were obtained using the full enumeration method. For the test, 30 instances with 3 periods and 5 stations were generated randomly, i.e. 10 instances for each of three levels for the number of part types (10, 20 and 30). The data were generated using the method for the preliminary test and the algorithms were terminated when there was no improvement for 30 consecutive iterations, i.e. $P_{O\_VNS} = P_{H\_VNS} = 30$.

Test results are summarized in <Table 4(a), (b) and (c)> that show the average percentage gaps from the optimal solution values for the three levels of the minimum allowable station utilization (0.6, 0.7 and 0.8). It can be seen from the tables that the VNS algorithms give optimal or near optimal solutions for all test instances. Of the two algorithms, the hybrid VNS algorithm outperformed the other in overall average and gave more optimal solutions as the minimum allowable station utilization increases. A paired t-test was done and the results showed that the hybrid algorithm is statistically better than the other in the significance level of 0.01. The overall average gaps of the hybrid algorithm were 0.09%, 0.06% and 0.05% for utilization levels 0.6, 0.7 and 0.8, respectively. Also, it gave the 77 optimal solutions out of 90 test instances. This implies that the SA based solution acceptance criterion is an effective method to escape from the local optimums in the local search improvement step. Finally, as can be seen in <Table 5>, the hybrid VNS algorithm required more computation times, but gave the solutions within 854.2 seconds. Note that the full enumeration method required 3203.4 seconds in the overall average.

Table 3. Test results for Setting Parameters of hybrid VNS Algorithm

NP	NS	NPT	$\beta = 0.5$ $\Gamma = 6$	$\beta = 0.5$ $\Gamma = 8$	$\beta = 0.7$ $\Gamma = 6$	$\beta = 0.7$ $\Gamma = 8$	$\beta = 0.9$ $\Gamma = 6$	$\beta = 0.9$ $\Gamma = 8$
3	5	10	0.37*	0.00	0.04	0.04	0.27	0.35
5	7	20	0.74	1.21	0.72	0.48	1.38	2.11
10	9	30	1.89	2.26	2.18	1.68	1.83	1.16
Average			1.00	1.16	0.98	0.74	1.16	1.20

See the footnotes of <Table 1>.
* Average relative performance ratio out of 10 instances.

Table 4. General Case: Optimality Gaps of VNS Algorithms for Small Sized Test Instances

(a) Minimum allowable station utilization = 0.6

NP	NS	NPT	Ordinary VNS		Hybrid VNS	
			APG	N_{opt}	APG	N_{opt}
3	5	10	0.09*	8**	0.04	9
		20	0.48	7	0.10	7
		30	0.53	4	0.12	8
Average			0.37		0.09	

See the footnotes of <Table 1>.
* Average percentage gap from optimal solution values out of 10 instances.
** Number of optimal solutions obtained from the algorithm out of 10 instances.

Table 4. General Case: Optimality Gaps of VNS Algorithms for Small Sized Test Instances (Continued)

(b) Minimum allowable station utilization = 0.7

NP	NS	NPT	Ordinary VNS		Hybrid VNS	
			APG	N_{opt}	APG	N_{opt}
3	5	10	0.04	9	0.04	9
		20	0.15	8	0.09	8
		30	0.38	5	0.04	8
Average			0.19		0.06	

(c) Minimum allowable station utilization = 0.8

NP	NS	NPT	Ordinary VNS		Hybrid VNS	
			APG	N_{opt}	APG	N_{opt}
3	5	10	0.00	10	0.00	10
		20	0.23	6	0.14	8
		30	0.53	3	0.00	10
Average			0.25		0.05	

Table 5. General Case: CPU Seconds of VNS Algorithms for Small Sized Test Instances

NP	NS	NPT	Ordinary VNS	Hybrid VNS
3	5	10	44.9*	63.0
		20	92.6	126.5
		30	306.0	492.6

See the footnotes of <Table 1>.

* Average CPU second out of 30 instances for 3 levels of allowable utilization.

The second main test was done to compare the relative performance of the two VNS algorithms for medium-to-large sized test instances. For the test, 240 instances, i.e. 10 instances for each of 24 combinations of three levels for the number of periods (3, 5 and 10), three levels for the number of stations (5, 7 and 9) and three levels for the number of part types (10, 20 and 30), were generated using the method explained earlier. <Table 6(a), (b) and (c)> summarize the average, minimum and maximum RPR values of the two algorithms for the three levels of the minimum allowable station utilization. As in the results for small sized test instances, the hybrid algorithm outperformed the ordinary one in overall averages and gave smaller RPR values as the minimum allowable station utilization increases, which also implies that the SA based solution acceptance is an effective method. In fact, the result of paired t-test showed that the two algorithms are statistically different in the significance level 0.01. Finally, as can be seen in <Table 7>, there was not much difference in computation times. In fact, the hybrid VNS algorithm required 1443.3 seconds in overall average, which is acceptable for practical applications because capacity scalability planning is a system design problem.

Table 6. General Case: RPR Values of VNS Algorithms for Medium-to-Large Sized Test Instances

(a) Minimum allowable station utilization = 0.6

NP	NS	NPT	Ordinary VNS	Hybrid VNS
3	7	10	0.17 (0.00, 0.91)*	0.00 (0.00, 0.00)
		20	0.29 (0.00, 1.17)	0.04 (0.00, 0.27)
		30	0.50 (0.00, 2.04)	0.14 (0.00, 0.87)
	9	10	0.03 (0.00, 0.18)	0.00 (0.00, 0.00)
		20	0.38 (0.00, 1.82)	0.05 (0.00, 0.24)
		30	0.31 (0.00, 1.44)	0.08 (0.00, 0.46)

Table 6. General Case: RPR Values of VNS Algorithms for Medium-to-Large Sized Test Instances (Continued)

NP	NS	NPT	Ordinary VNS	Hybrid VNS
5	5	10	0.69 (0.00, 3.50)	0.33 (0.00, 1.86)
		20	0.82 (0.00, 3.11)	0.14 (0.00, 1.01)
		30	0.35 (0.00, 1.25)	0.00 (0.00, 0.02)
	7	10	1.02 (0.00, 7.41)	0.14 (0.00, 1.43)
		20	0.62 (0.00, 2.29)	0.31 (0.00, 1.34)
		30	1.11 (0.00, 3.60)	0.21 (0.00, 1.83)
	9	10	1.21 (0.00, 5.04)	0.33 (0.00, 3.08)
		20	1.21 (0.00, 5.99)	0.15 (0.00, 0.78)
		30	0.92 (0.00, 2.39)	0.04 (0.00, 0.37)
10	5	10	2.46 (0.00, 7.17)	0.69 (0.00, 5.45)
		20	1.46 (0.00, 4.50)	0.26 (0.00, 1.26)
		30	1.37 (0.00, 3.56)	0.21 (0.00, 1.32)
	7	10	0.99 (0.00, 5.39)	0.00 (0.00, 0.00)
		20	1.98 (0.00, 5.69)	0.17 (0.00, 0.84)
		30	1.15 (0.00, 2.47)	0.06 (0.00, 0.62)
	9	10	1.97 (0.00, 6.60)	0.43 (0.00, 2.77)
		20	1.34 (0.00, 5.01)	0.29 (0.00, 1.08)
		30	1.02 (0.00, 2.36)	0.18 (0.00, 1.78)
Average			0.97	0.18

See the footnotes of <Table 1>.
* Average relative performance ratio out of 10 instances (minimum and maximum in parenthesis)

(b) Minimum allowable station utilization = 0.7

NP	NS	NPT	Ordinary VNS	Hybrid VNS
3	7	10	0.15 (0.00, 1.06)	0.00 (0.00, 0.00)
		20	0.35 (0.00, 2.86)	0.07 (0.00, 0.69)
		30	0.31 (0.00, 2.02)	0.06 (0.00, 0.58)
	9	10	0.17 (0.00, 1.70)	0.00 (0.00, 0.01)
		20	0.13 (0.00, 0.91)	0.03 (0.00, 0.24)
		30	0.27 (0.00, 1.07)	0.06 (0.00, 0.36)
5	5	10	0.84 (0.00, 7.53)	0.25 (0.00, 1.36)
		20	1.11 (0.00, 4.19)	0.07 (0.00, 0.44)
		30	1.07 (0.00, 2.97)	0.01 (0.00, 0.15)
	7	10	1.77 (0.00, 4.64)	0.00 (0.00, 0.00)
		20	1.28 (0.00, 4.71)	0.13 (0.00, 0.93)
		30	1.08 (0.00, 4.92)	0.47 (0.00, 2.70)
	9	10	0.63 (0.00, 2.69)	0.00 (0.00, 0.00)
		20	0.61 (0.00, 1.51)	0.00 (0.00, 0.00)
		30	1.13 (0.00, 2.44)	0.01 (0.00, 0.09)
10	5	10	3.14 (0.00, 9.04)	0.66 (0.00, 3.88)
		20	1.41 (0.00, 5.63)	0.20 (0.00, 1.02)
		30	1.16 (0.00, 4.95)	0.54 (0.00, 1.92)
	7	10	2.37 (0.00, 9.74)	0.45 (0.00, 3.54)
		20	2.90 (0.00, 8.51)	0.04 (0.00, 0.41)
		30	1.08 (0.00, 5.20)	0.31 (0.00, 1.52)
	9	10	3.25 (0.00, 13.95)	0.00 (0.00, 0.00)
		20	1.61 (0.00, 5.43)	0.47 (0.00, 3.87)
		30	1.58 (0.00, 3.80)	0.04 (0.00, 0.41)
Average			1.23	0.16

Table 6. General Case: RPR Values of VNS Algorithms for Medium-to-Large Sized Test Instances (Continued)

(c) Minimum allowable station utilization = 0.8

NP	NS	NPT	Ordinary VNS	Hybrid VNS
3	7	10	0.12 (0.00, 1.16)	0.00 (0.00, 0.00)
		20	0.37 (0.00, 2.00)	0.00 (0.00, 0.00)
		30	0.26 (0.00, 1.49)	0.04 (0.00, 0.26)
	9	10	0.00 (0.00, 0.00)	0.00 (0.00, 0.00)
		20	0.02 (0.00, 0.21)	0.00 (0.00, 0.00)
		30	0.23 (0.00, 1.27)	0.00 (0.00, 0.00)
5	5	10	1.67 (0.00, 7.41)	0.00 (0.00, 0.00)
		20	0.36 (0.00, 1.37)	0.12 (0.00, 1.05)
		30	0.49 (0.00, 1.90)	0.13 (0.00, 0.74)
	7	10	0.90 (0.00, 3.72)	0.00 (0.00, 0.00)
		20	0.14 (0.00, 0.74)	0.05 (0.00, 0.44)
		30	0.97 (0.00, 2.49)	0.07 (0.00, 0.71)
	9	10	0.19 (0.00, 1.30)	0.00 (0.00, 0.00)
		20	1.56 (0.00, 7.61)	0.00 (0.00, 0.00)
		30	0.64 (0.00, 4.08)	0.11 (0.00, 0.59)
10	5	10	2.58 (0.00, 9.59)	0.06 (0.00, 0.58)
		20	1.90 (0.26, 4.33)	0.00 (0.00, 0.00)
		30	1.70 (0.00, 3.35)	0.01 (0.00, 0.06)
	7	10	0.77 (0.00, 5.48)	0.07 (0.00, 0.61)
		20	2.43 (0.00, 9.45)	0.17 (0.00, 1.13)
		30	1.15 (0.00, 3.71)	0.02 (0.00, 0.10)
	9	10	3.65 (0.00, 11.91)	0.02 (0.00, 0.16)
		20	1.74 (0.00, 6.19)	0.16 (0.00, 1.13)
		30	2.14 (0.00, 3.52)	0.04 (0.00, 0.38)
Average			1.08	0.04

Table 7. General Case: CPU Seconds of VNS Algorithms for Medium-to-Large Sized Test Instances

NP	NS	NPT	Ordinary VNS	Hybrid VNS
3	7	10	293.0*	329.3
		20	335.4	433.0
		30	634.4	804.2
	9	10	1112.4	1529.7
		20	1312.0	1781.6
		30	1797.9	2064.6
5	5	10	71.7	78.4
		20	136.6	156.5
		30	404.8	587.8
	7	10	359.9	494.1
		20	559.8	741.1
		30	884.6	1187.6
	9	10	1469.5	1885.4
		20	1783.0	2489.5
		30	2379.7	3100.6
10	5	10	104.3	121.0
		20	225.0	260.7
		30	884.8	928.9
	7	10	539.0	658.9
		20	1050.8	1177.5
		30	1695.4	2099.0
	9	10	1959.6	2624.3
		20	2833.2	3784.1
		30	4640.7	5322.1
Average			1144.5	1443.3

See the footnotes of <Table 1>.

* Average CPU second out of 30 instances for 3 levels of allowable utilization.

4. Concluding Remarks

This study addressed multi-period capacity scalability planning for job-shop-type RMSs with dynamic demands over a planning horizon. Two cases of the problem, basic case with non-decreasing demands and general case with fluctuating demands, were considered with the practical constraints of the limited number of pallets and the minimum allowable station utilization. For the basic case that determines the system components to be added in each period of the planning horizon, a nonlinear integer programming model was proposed that includes a closed queueing network based estimations of throughputs and utilizations for the objective of minimizing the sum of component acquisition and configuration change costs. Then, two backward heuristics, MB-UT and MB-TH, were proposed and computation results showed that they outperform the existing ones significantly because they start with better last period configurations. Of the two heuristics, MB-TH that uses the throughput/cost ratio when selecting the components to be added in the last period configurations performed better than the other. For the general case with an additional decision of removing system components, the basic nonlinear programming model was extended, and then two VNS algorithms were proposed for the objective of minimizing the sum of component acquisition/removal and configuration change costs. Computational results showed that the hybrid VNS algorithm with the SA technique to allow non-improving moves outperforms the ordinary one.

This study can be extended in several directions. First, the optimal solution approach is worth to be developed in the theoretical aspect. Second, the current problem can be generalized into stochastic ones for which various simulation optimization methods, such as sample average approximation and robust optimization, can be used. Finally, the digital twin technology can be used to evaluate the performance of RMS configurations in real-time.

References

Albus, M. and Huber, M. F. (2023), Resource Reconfiguration and Optimization in Brownfield Constrained Robotic Assembly Line Balancing Problem, *Journal of Manufacturing Systems*, **67**, 132-142.

Albus, M., Hornek, T., Kraus, W., and Huber, M. F. (2024), Towards Scalability for Resource Reconfiguration in Robotic Assembly Line Balancing Problem using aModified Genetic Algorithm, *Journal of Intelligent Manufacturing*, DOI: 10.1007/s10845-023-02292-0.

Andersen, A. L., Brunoe, T. D., Nielsen, K., and Rösiö, C. (2017), Towards a Generic Design Method for Reconfigurable Manufacturing Systems: Analysis and Synthesis of Current Design Methods and Evaluation of Supportive Tools, *Journal of Manufacturing Systems*, **42**, 179-195.

Ashraf, M. and Hasan, F. (2018), Configuration Selection for a Reconfigurable Manufacturing Flow Line Involving Part Production with Operation Constraints, *The International Journal of Advanced Manufacturing Technology*, **98**(5), 2137-2156.

Bortolini, M., Galizia, F. G., and Mora, C. (2018), Reconfigurable Manufacturing Systems: Literature Review and Research Trend, *Journal of Manufacturing Systems*, **49**, 93-106.

Dou, J., Dai, X., and Meng, Z. (2009), Graph Theory-based Approach to Optimize Single-product Flow-line Configurations of RMS, *The International Journal of Advanced Manufacturing Technology*, **41**(9), 916-931.

Dou, J., Dai, X., and Meng, Z. (2010), Optimisation for Multi-part Flow-line Configuration of Reconfigurable Manufacturing System Using GA, *International Journal of Production Research*, **48**(14), 4071-4100.

Hansen, P., Mladenović, N., Todosijević, R., and Hanafi, S. (2017), Variable Neighborhood Search: Basics and Variants, *EURO Journal on Computational Optimization*, **5**(3), 423-454.

Gadalla, M. and Xue, D. (2016), Recent Advances in Research on Reconfigurable Machine Tools: A Literature Review, *International Journal of Production Research*, **55**(5), 1440-1454.

Goyal, K. K., Jain, P. K., and Jain, M. (2012), Optimal Configuration Selection for Reconfigurable Manufacturing System using NSGA II and TOPSIS, *International Journal of Production Research*, **50**(15), 4157-4191.

Koren, Y., Gu, X., and Guo, W. (2018), Reconfigurable Manufacturing Systems: Principles, Design, and Future Trends, *Frontiers of Mechanical Engineering*, **13**(2), 121-136.

Koren, Y., Heisel U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., and Brussel, H. (1999), Reconfigurable Manufacturing Systems, *Annals of the CIRP*, **48**(2), 527-540.

Koren, Y. and Shpitalni, M. (2010), Design of Reconfigurable Manufacturing Systems, *Journal of Manufacturing Systems*, **29**(4), 130-141.

Koren, Y., Wang, W., and Gu, X. (2017), Value Creation through Design for Scalability of Reconfigurable Manufacturing Systems, *International Journal of Production Research*, **55**(5), 1227-1242.

Kumar, G., Goyal, K. K., Batra, N. K., and Rani, D. (2022), Single Part Reconfigurable Flow Line Design using Fuzzy Best Worst Method, *OPSEARCH*, **59**(2), 603-631.

Lee, S. and Ryu, K. (2021), Development of a Goal Model for Self-Reconfigurable Manufacturing Systems, *Journal of the Korean Institute of Industrial Engineers*, **47**(2), 160-173.

Maganha, I., Silva, C., and Ferreira, L. M. D. (2019), The Layout Design in Reconfigurable Manufacturing Systems: A Literature Review, *The International Journal of Advanced Manufacturing Technology*, **105**, 683-700.

Mladenović, N. and Hansen, P. (1997), Variable Neighborhood Search, *Computers & Operations Research*, **24**(11), 1097-1100.

Moghaddam, S. K., Houshmand, M., and Valilai, O. F. (2018), Configuration Design in Scalable Reconfigurable Manufacturing Systems (RMS): A Case of Single-product Flow Line (SPFL), *International Journal of Production Research*, **56**(11), 3932-3954.

Moghaddam, S. K., Houshmand, M., Saitou, K., and Valilai, O. F. (2020), Configuration Design of Scalable Reconfigurable Manufacturing Systems for Part Family, *International Journal of Production Research*, **58**(10), 2974-2996.

Napoleone, A., Anderson, A. L., Brunoe, T. D., and Nielsen, K. (2023), Towards Human-centric Reconfigurable Manufacturing Systems: Literature Review of Reconfigurability Enablers for Reduced Reconfiguration Effort and Classification Frameworks, *Journal of Manufacturing Systems*, **67**, 23-34.

Napoleone, A., Pozzetti, A., and Macchi, M. (2018), A Framework to Manage Reconfigurability in Manufacturing, *International Journal of Production Research*, **56**(11), 3815-3837.

Pansare, R., Yadav, G., and Nagare, M. R. (2023), Reconfigurable Manufacturing Systems: A Systematic Review, Meta-analysis and Future Research Directions, *Journal of Engineering, Design and Technology*, **21**(1), 228-265.

Solberg, J. J. (1977), A Mathematical Model of Computerized Manufacturing Systems, *Proceedings of the 4th International Conference on Production Research*, Tokyo, Japan.

Son, S. Y. (2000), *Design Principles and Methodologies for Reconfigurable Machining Systems*, PhD Thesis. University of Michigan.

Spicer, P., and Carlo, H. J. (2007), Integrating Reconfiguration Cost into the Design of Multi-period Scalable Reconfigurable Manufacturing Systems, *Journal of Manufacturing Science and Engineering*, **129**(1), 202-210.

Tempelmeier, H., and Kuhn, H. (1993), *Flexible Manufacturing Systems: Decision Support for Design and Operation*, John Wiley and Sons, New York.

Wang, W., and Koren, Y. (2012), Scalability Planning for Reconfigurable Manufacturing Systems, *Journal of Manufacturing Systems*, **31**(2), 83-91.

Yang J. Son, Y. H., Lee D., Noh, S. D., Kim, H., Lee, J. S., Kim, Y. S., Won, Y. (2021), A Flexible Jig for Reconfigurable and Flexible Assembly of Smart Factory, *Journal of the Korean Institute of Industrial Engineers*, **47**(1), 102-116.

Yelles-Chaouche, A. R., Gurevsky, E., Brahimi, N., and Dolgui, A. (2021), Reconfigurable Manufacturing Systems from an Optimisation Perspective: A Focused Review of Literature, *International Journal of Production Research*, **59**(21), 6400-6418.

Youssef, A. M. A. and ElMaraghy, H. A. (2006), Modelling and Optimization of Multiple-aspect RMS Configurations, *International Journal of Production Research*, **44**(22), 4929-4958.

Yu, Z.-J., Shin, J.-H., and Lee, D.-H. (2014), An Operational-level Dynamic Capacity Scalability Model for Reconfigurable Manufacturing Systems, *The International Journal of Industrial Engineering: Theory, Applications and Practice*, **21**(6), 317-326.

Zhang, C., Dou, J., and Wang, P. (2023), Configuration Design of Reconfigurable Single-product Robotic Assembly Line for Capacity Scalability, *Computers & Industrial Engineering*, **185**, 109682.

Zhou, Y.-D., Shin, J.-H., and Lee, D.-H. (2019), Loading and Scheduling for Flexible Manufacturing Systems with Controllable Processing Times, *Engineering Optimization*, **51**(3), 412-426.

Author Profile

Xuebin Li received the B.S. degree in Mechanical Engineering from Beijing Institute of Technology and the M.S. degree in Industrial Engineering from Hanyang University. His research interests include manufacturing planning/scheduling and optimization.

Hyeon-Il Kim is a Ph.D. student at the Department of Industrial Engineering, Hanyang University, Seoul, Republic of Korea. He received the B.S. degree from Kyungsung University and M.S. degree from Hanyang University, all in Industrial Engineering. His research interests include design and operation of manufacturing/service systems, environmental conscious manufacturing, reverse logistics and operations research applications.

Dong-Ho Lee is a professor at the Department of Industrial Engineering, Hanyang University, Seoul, Republic of Korea. He received the B.S. degree from Seoul National University and M.S./Ph.D. degrees from Korea Advanced Institute of Science and Technology (KAIST), all in Industrial Engineering. After earning the Ph.D. degree, he worked as a post-doctoral research fellow at the Department of Mechanical Engineering, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. His research interests include design and operation of manufacturing/service systems, optimization theory and applications, environmental conscious manufacturing and reverse logistics.