# An Efficient Heuristic for Pooled Repair Shop Designs

Hasan Hüseyin Turan[1], Shaligram Pokharel[2(✉)], Tarek Y. ElMekkawy[2], Andrei Sleptchenko[3], and Maryam Al-Khatib[2]

[1] Capability Systems Centre, School of Engineering and Information Technology, University of New South Wales, Canberra, Australia
[2] Department of Mechanical and Industrial Engineering, College of Engineering, Qatar University, Doha, Qatar
shaligram@qu.edu.qa
[3] Department of Industrial and Systems Engineering, Khalifa University of Science and Technology, Abu Dhabi, UAE

**Abstract.** An effective spare part supply system planning is essential to achieve a high capital asset availability. We investigate the design problem of a repair shop in a single echelon repairable multi-item spare parts supply system. The repair shop usually consists of several servers with different skill sets. Once a failure occurs in the system, the failed part is queued to be served by a suitable server that has the required skill. We model the repair shop as a collection of independent sub-systems, where each sub-system is responsible for repairing certain types of failed parts. The procedure of partitioning a repair shop into sub-systems is known as pooling, and the repair shop formed by the union of independent sub-systems is called a pooled repair shop. Identifying the best partition is a challenging combinatorial optimization problem. In this direction, we formulate the problem as a stochastic nonlinear integer programming model and propose a sequential solution heuristic to find the best-pooled design by considering inventory allocation and capacity level designation of the repair shop. We conduct numerical experiments to quantify the value of the pooled repair shop designs. Our analysis shows that pooled designs can yield cost reductions by 25% to 45% compared to full flexible and dedicated designs. The proposed heuristic also achieves a lower average total system cost than that generated by a Genetic Algorithm (GA)-based solution algorithm.

**Keywords:** Spare part logistics · Repair shop · Pooling · Heuristic · Genetic algorithm

## 1 Introduction

Service and manufacturing operations rely heavily on the availability of equipment and assets. High availability of assets can be achieved with effective maintenance strategies. However, maintenance can be costly. For example, a recent

© Springer Nature Switzerland AG 2019
G. H. Parlier et al. (Eds.): ICORES 2018, CCIS 966, pp. 102–118, 2019.
https://doi.org/10.1007/978-3-030-16035-7_6

report of IATA's Maintenance Cost Task Force points out that maintenance cost can be anywhere between 10% to 15% of the total operation cost of a commercial airline industry [1]. Similarly, for manufacturing firms, maintenance cost may reach up to 60% of the production cost [2]. Hence, careful planning of maintenance operations not only leads to a decrease in the total cost but also significant improvements in the reliability of systems [3]. Maintenance planning includes the determination of the maintenance strategy (e.g., failure-based/corrective, preventive and condition-based), time interval between maintenance operations, and quantity and quality of maintenance resources such as technicians, supplies and spare parts [4].

In this paper, the corrective maintenance of high-valued assets in and particular the decisions regarding the amount of spare part inventory, capacity and design of repair facilities are investigated. Corrective maintenance of assets is usually done by replacing a failed part by a repaired part available in the stock. If repairable spares are not in the stock, the asset goes down, and a downtime cost is incurred till a sufficient number of spares are supplied to the system [5–7]. A large number of spares are required to ensure a high availability of the capital asset. However, keeping a large number of repairable in inventory increases the cost [8]. The decision on repair shop design heavily influences the number of spares to be stocked. An optimal design of the repair shop can lead to a less number of spare parts that are needed to achieve the same level of availability. Thus, at the operational level, the inventory and repair shop decisions have to be coordinated together to reduce downtimes.

There are different types of repair shop design alternatives, as illustrated in Fig. 1. The two extremes are the full flexible (full cross-training) and the dedicated designs. Figure 1(a) depicts a full cross-training design scheme in which all of the servers are merged into a single cluster/sub-system. In this design scheme, all servers are considered to have necessary skills to repair any type of failed parts. On the contrary, in the dedicated design, each cluster of servers is responsible for repairing a specific type of spare part as in Fig. 1(c).

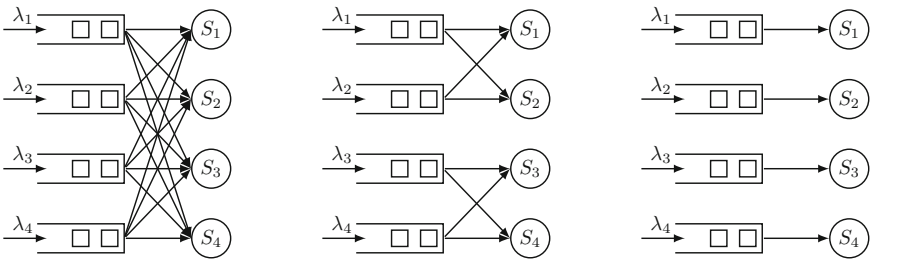


**Fig. 1.** Possible repair shop designs.

An intermediate level design scheme between the dedicated and the full flexible system is the pooled design. In the pooled design, the repair shop is a union

of independent clusters/sub-systems, where each sub-system is responsible for repairing certain types of failed parts as in Fig. 1(b). In this study, we try to find the best-pooled repair shop design that leads to minimum total system cost. Nevertheless, identifying the best-pooled repair shop design is a challenging combinatorial optimization problem. Thus, we develop an efficient solution heuristic to overcome computational complexity of the problem. The proposed solution heuristic also computes the optimal amount of spare part inventories to keep on stock and the number of servers (capacity) that have to be allocated into each cluster.

The rest of the paper is organized as follows. In Sect. 2, a review of related literature is provided. In Sect. 3, problem definition and the mathematical model are presented. The solution heuristic for the proposed model is discussed in Sect. 4. Section 5 provides a comparative computational study under input settings. Conclusions and future research directions are summarized in Sect. 6.

## 2 Literature Review

Some significant advances in optimization of spare part supply systems, capacity and inventory theory and design of flexibility service/manufacturing systems can be seen in the literature. However, research opportunities exist at the intersection of these research areas. The optimization problem analyzed in this paper exploits the intersection of the design of a flexible/cross-trained repair shop in a spare part supply system and optimization of resource capacity of the repair shop and inventory levels of spares.

The dominant model for repairable items, both in the literature and in the practical applications, is METRIC (Multi-Echelon Technique for Recoverable Item Control), developed by [5]. METRIC based models assume that the repair capacity is infinite. This assumption may not be appropriate in most industrial settings. Hence, some researchers have relaxed ample repair capacity assumption by explicitly considering finite repair service capacity [9–13]. In addition to the limited repair capacity assumption, integrated optimization of repair capacity and maintenance policies are also studied extensively in the literature by [14–17]. The work of [18] relaxes assumptions of stationary failure rates of spares and finite repair capacity at the same time. Similarly, the model of [19] also optimizes spare inventory levels under finite repair capacity together with nonstationary failure rates of spares under a certain budget restriction or spending over certain availability. The impact of limited but not constant (varying repair capacity on a system for repairable items) is analyzed by [20]. We refer to the recent literature review article of [21] for integrating decisions on spare parts inventories and repair shops.

In [22] regarding the manufacturing resource flexibility, a comparison between a totally dedicated system, a totally flexible system, and several intermediate possibilities are provided. In the following years, several authors have extended the work provided by [22] and some of them have also validated robustness of *"little or limited flexibility"* being usually sufficient for optimal system performance (see Ref. [23–30]). Cross-training is one of the more widely discussed

capacity/workforce flexibility methods for complex systems [31]. Production lines [32,33], job shops [34], flow/assembly shops [35–39], manufacturing [40–43], call centers [44–47], health care [48,49], field services [50–53] and maintenance/repair [54] are some examples of systems where cross-training is applied. The more detailed discussions and classifications of flexibility and cross-training applications can be found in the recent review articles of [31,55].

A limited number of cross-training related studies also appears in the maintenance and spare part logistics literature. For example, [50–52] discuss the optimal cross-training policies of technicians/service engineers in a field service setting. Similarly, [53,54] address workforce management problems in corrective repair/maintenance environments, in which repairmen are either cross-trained or dedicated. The analysis of cross-training schemes in repair shop design is discussed in [56–58] by using simulation-optimization techniques.

Even though pooling is considered as a partial cross-training of resources, to the best of our knowledge, no results has been presented analyzing pooling in spare part supply systems other than very recent works of [59–61]. Our work fills the gap on pooled repair shops designs in spare part supply systems integrated with capacity decision in the literature.

## 3 Problem Description and Formulation

We study the design problem of a repair shop in a single echelon repairable multi-item spare part supply system. The repair shop may consist of several parallel multi-skilled servers, and storage facilities for the repaired items. Once a failed part is received from the technical system at the installed base, it is queued to be served by a suitable server with the required skills. At the same time, if a repaired (as-good-as-new) part is available in the inventory, it is sent back to the installed base. If the item is not available in the stock, the request is backordered. In this case, the technical system goes down and a downtime cost occurs till the requested ready-for-use part is delivered.

The repair shop may have pooled structure with one or more cells/clusters or an arbitrary structure. In arbitrary designs, not all servers in a cluster are fully flexible; i.e., some servers are partially cross-trained to repair only a subset of all stock keeping units (SKUs) in the cluster. In this paper, we restrict design alternatives limited to only pooled repair shops as in Fig. 1(b), and formulate a stochastic mixed-integer mathematical programming model to find the minimum cost spare part supply system.

In this paper, we proceed from commonly used assumptions in a repairable spare part supply system (see ref. [6,56] and assumption lists therein):

(a) The failures of spares occur according to a Poisson process and are mutually independent from each other with constant rates.
(b) The repair times are exponentially distributed and mutually independent. The expected repair times depend on the SKU type and are independent of the processing server.

(c) First come first served (FCFS) queuing discipline is adopted inside each and every cluster, and no priorities exist among the failed spares.
(d) For all parts $(s-1, s)$ one-for-one replenishment policy is used. That is, the stock level equals $s$ and each demand immediately generates an order for a replacement part; as a consequence, there is no batching.
(e) The total holding costs for every SKU per unit time are linear in the initial inventory levels (initially acquired inventory).
(f) Penalty costs (or backorder costs) occur when the required part is not available and are paid per time unit per not available SKU.
(g) A positive cross-training (or flexibility) cost occurs whenever an additional skill is assigned to a server. In other words, the cross-training cost is an increasing function on the number of skills per server.
(h) Each cluster inside the repair shop is modeled as a multi-class multi-server $M/M/k$ queuing system with dedicated queues; i.e., every server inside a cluster has the ability to repair all SKUs that are assigned to that cluster.
(i) The clusters inside the repair shop are mutually exclusive (disjoint) and collectively exhaustive. That is, a particular failed SKU can be repaired at exactly one cluster and each SKU is assigned to exactly one cluster.

The last two assumptions (h) and (i) restrict the repair shop design alternatives to the pooled designs. These two assumptions also limit the computational complexity of the system and enable using queue-theoretical approximations to find steady-state probability distribution of items in the system.

We use the problem formulation presented in [61]. The sets, parameters and decision variables for the developed formulations and solution procedures are presented as follows.

Decision Variables

$S_i$: Amount of initial inventory (basestock level) kept on stock for SKU type $i\,(i = 1, \ldots, N)$, where $\mathbf{S} = (S_1, \ldots, S_N)$.
$z_k$: Number of the operational servers in the cluster $k$ $(k = 1, \ldots, y)$, and where $\mathbf{Z} = (z_1, \ldots, z_y)$.
$x_{ik}$: Binary variable indicating that whether the cluster $k$ has a skill to repair SKU type $i\,(i = 1, \ldots, N)$ or not, where $\mathbf{X}_k = (x_{1k}, \ldots, x_{Nk})^T$ and $\mathbf{X} = [\mathbf{X}_1| \ldots |\mathbf{X}_y]$.
$y$: Number of clusters in the repair shop.

Problem Parameters

$N$: Number of distinct types of repairables (SKUs).
$\lambda_i$: Failure rate of SKU type $i$ $(i = 1, \ldots, N)$.
$\mu_i$: Service rate of SKU type $i$ $(i = 1, \ldots, N)$.
$h_i$: Inventory holding cost of SKU type $i$ per unit time per part $(i = 1, \ldots, N)$.
$b$: Penalty cost for each back ordered demand per unit time, which is equivalent to paying per unit time per technical system that is down because of a lack of spare parts.

$f$: Operation cost of a server per unit time (e.g., annual wage).
$c_i$: Cost of having skills to repair SKU type $i$ per unit time per server $(i = 1, \ldots, N)$ (e.g., annual qualification bonus).
$\epsilon$: Very small positive real number.

The objective function in Eq. (1) has four cost terms namely, server (capacity), cross-training, holding and backorder costs. Objective function considers several trade-offs between the cost terms such as the cost of holding excess inventory and the cost of downtime, and also the trade-off between the cost of having single or several clusters that include dedicated or cross-trained servers.

$$\min_{\mathbf{S},\,\mathbf{X},\,\mathbf{Z}} \sum_{k=1}^{y} f z_k + \sum_{k=1}^{y} z_k \left( \sum_{i=1}^{N} c_i x_{ik} \right) + \sum_{i=1}^{N} h_i S_i + b \sum_{i=1}^{N} \mathbb{EBO}_i \left[ S_i, \mathbf{X}, \mathbf{Z} \right] \quad (1)$$

The *penalty (backorder)* cost term is calculated using the penalty cost $b$ and the expected total number of backordered parts $\mathbb{EBO}_i [S_i, \mathbf{X}, \mathbf{Z}]$ for each SKU type $i$ in the steady-state; under the given initial inventory level $S_i$, pooling scheme of the repair shop $\mathbf{X}$ and the server assignment policy $\mathbf{Z}$. The variable $\mathbf{X}$ represents the $(N \times y)$ matrix of the binary decision variables $x_{ik}$ denoting how SKUs are pooled in the repair shop, and the variable $\mathbf{Z}$ represents a $(1 \times y)$ row matrix of integer decision variables $z_k$ denoting the number of servers in each cluster of the repair shop.

Constraints (2) and (4) ensure that pooling scheme $\mathbf{X}$ satisfies mutually exclusive and total exhaustive condition for each cluster, i.e., any SKU type being repaired by exactly one cluster. Queues (number of waiting failed spares) in each cluster have to have finite queue length at the steady-state to prevent overloading of the repair shop. Thus, the stability of the system is guaranteed by constraint (3) and (5) by assigning sufficient number of servers to each cluster. Constraints (4–7) are required for non-negativity and integrality of the variables. For a non-overloaded system, the overall utilization rate of a particular cluster $k$ $(\sum_{i=1}^{N} x_{ik} \lambda_i / \mu_i)$ must be strictly smaller than the capacity (total number of servers in the cluster $z_k$) of that cluster, which is ensured by the parameter, $\epsilon$.

$$\sum_{k=1}^{y} x_{ik} = 1 \qquad i = 1, \ldots, N \quad (2)$$

$$\sum_{i=1}^{N} x_{ik} \frac{\lambda_i}{\mu_i} \leq z_k (1 - \epsilon) \qquad k = 1, \ldots, y \quad (3)$$

$$x_{ik} \in \{0, 1\} \qquad i = 1, \ldots, N \; k = 1, \ldots, y \quad (4)$$

$$z_k \in \mathbb{Z}^+ \qquad k = 1, \ldots, y \quad (5)$$

$$S_i \in \mathbb{N}_0 \qquad i = 1, \ldots, N \quad (6)$$

$$y \in \{1, \ldots, N\} \quad (7)$$

## 4 Solution Algorithm: Pooling Heuristic

We search for the optimal values of decision variables sequentially by fixing the values of some decision variables and optimizing the remaining ones as discussed in [61]. First, feasible partitions of SKUs, i.e., pooling policies/schemes $\mathbf{X}$ are generated. Pooling schemes are generated either by pooling heuristic or by a genetic algorithm as explained in the Subsect. 5.2. Then, capacity levels $\mathbf{Z}$ and basestock inventory levels $\mathbf{S}$ are optimized under the given pooling scheme for each cluster. The visual flow of the proposed solution heuristic(s) together with its sub-routines and their interactions with each other are depicted in Fig. 2.

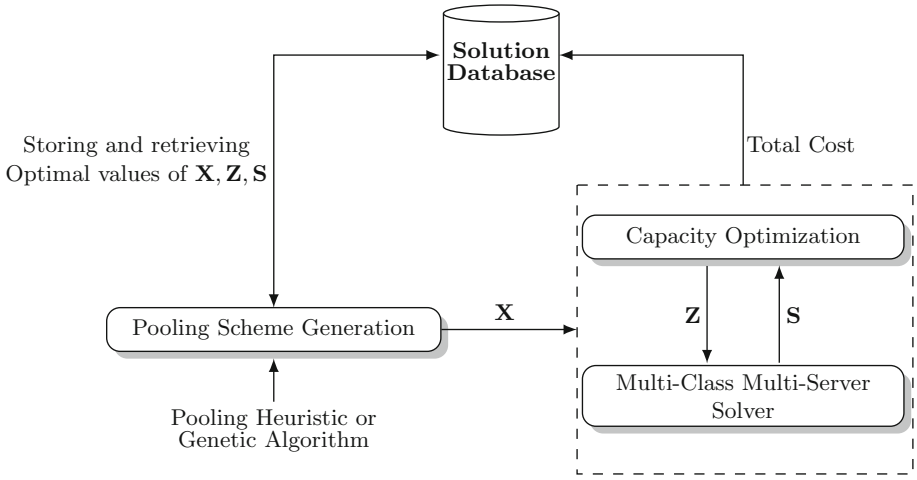


**Fig. 2.** Flow of the solution algorithms.

In the pooling heuristic, to form partitions of SKUs into clusters, all SKUs are sorted in ascending order by their service rates $\mu_i$ so that SKUs closer in service rates are likely to be in the same cluster. This is expected to decrease variations in the service times of SKUs in clusters. Decrease in the variation of service times usually results in a decrease in the number of failed parts waiting for repair in the cluster and eventually lowering the number of backorders and the total cost. Afterward, sorted list of SKUs is divided into smaller lists that have a size of $n_{max}$ or less. The trade-off between the run time of algorithm and the output solution quality are taken into account to determine the value of $n_{max}$. We set $n_{max}$ as 10 for our experimental runs. For the smaller list ($N \leq n_{max}$), the total enumeration function that is discussed in [61] is invoked. Total enumeration function takes an array of SKU indexes as an input and slices it into sub-arrays for given number of clusters $y$ from 1 to the length of the input array. Each slice/sub-array corresponds to a cluster in a pooled repair shop, and each slicing scheme corresponds to a particular pooling policy $\mathbf{X}$. For the larger sorted

SKU index sets, it is not possible to enumerate all slicing schemes with total enumeration function. Therefore, we divide the problem into sub-problems that have the maximum size of $n_{max}$ or less and call total enumeration function for each sub-problem obtained after division. Then, we generate new sub-problems by combining the last and the first elements of adjacent sub-problems. At each iteration, we insert a new SKU index to newly generated sub-problem till the size of the problem reaches $n_{max}$.

After the generation of the pooling policy $\mathbf{X}$ via above-described pooling heuristic, capacity and inventory level optimization modules are called as shown in Fig. 2. These modules rely on the fact that for every feasible policy $\mathbf{X}$, each cluster can be analyzed and optimized separately due to the clusters being mutually exclusive and independent from each other. The decomposition of the repair shop in sub-systems by pooling reduces the complexity of the problem and enables the use of queue-theoretical approximations to optimize the inventory $\mathbf{S}$ and capacity levels $\mathbf{Z}$. Each cluster $k$ in the repair shop for given number of servers $z_k$ can be analyzed as a multi-class multi-server $M/M/z_k$ queuing system.

The probability distribution of the number of failed SKU type $i$ at the steady-state, $p_i\,(q)$, is required to evaluate $\mathbb{EBO}_i\,[S_i, \mathbf{Z}, \mathbf{X}]$ in the objective function. To calculate the probability distribution of the number of failed SKU type $i$, the approach proposed by [62] is used. Nonetheless, computational burden arises when the number of SKU types and the number of servers increases in the cluster. To overcome this issue, the queuing approximation discussed in [63,64] is used. In this approximation, marginal probability distribution (and several performance characteristics) of SKU type $i$ in cluster $k$ is derived by aggregating all other SKUs in the cluster $k$ into a single SKU type (class). The procedure is repeated to obtain the remaining distributions for other SKUs in the cluster.

Figure 3 visualizes how $N$-class $M/M/z_k$ system is decomposed into $N$ independent 3-class $M/M/z_k$ for approximation, where $\Lambda_A$ and $\Lambda_{A'}$ denote the arrival rates of aggregated classes.

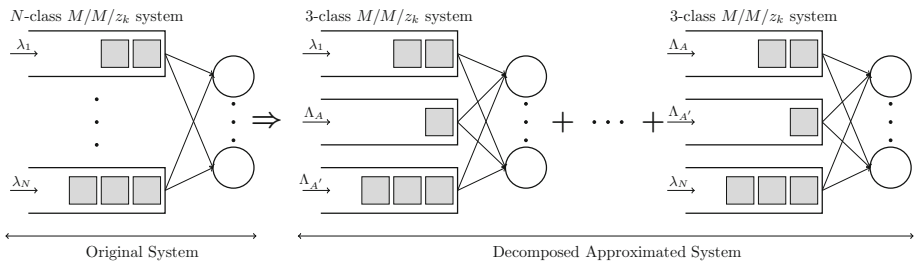


**Fig. 3.** Approximation of a $N$-class $M/M/z_k$ queuing system with decomposition into $N$ 3-class $M/M/z_k$ sub-system.

For the given pooling policy $\mathbf{X}$ and capacity levels for each cluster $\mathbf{Z}$, the problem can be reduced to the following one-dimensional optimization problem for each SKU type $i$ by using the independence of each cluster:

$$\min_{S_i \in \mathbb{N}_0} \left( h_i S_i + b\mathbb{EBO}_i \left[ S_i, \mathbf{X}, \mathbf{Z} \right] \right) \tag{8}$$

The optimization problem in Eq. (8) takes into account the trade-off between holding and backorder costs, which has similar structure as traditional newsboy problem (see [60] for a detailed discussion). By using the approximated distributions found by queuing approximation $\tilde{p}_i(q)$, the above optimization problem can be minimized by the smallest $S_i$ for which Eq. (9) holds.

$$\sum_{q=0}^{S_i} \tilde{p}_i(q) \geq \frac{b - h_i}{b} \quad i = 1, \ldots, N \tag{9}$$

## 5 Numerical Study

In this section, we present a computational study of the proposed solution algorithm. First, in Subsect. 5.1, the experiment testbed used in analysis is given and in Subsect. 5.2, details on benchmarking algorithm are provided. In Subsect. 5.3, total system cost reductions achieved by different algorithms are analyzed. Additionally, comparison of cross-training schemes are also given. In Subsect. 5.4, run times of the proposed optimization algorithms are provided.

### 5.1 Testbed

A full factorial design of experiment (DoE) with 7 factors and 2 levels per factor is used to generate the testbed with total of 128 test instances as in [56,61]. The number of SKUs, $N$, and the initial total number of servers, $M$, are the first two DoE factors with levels 10 and 20 for the numbers of SKUs, and 5 and 10 for the initial numbers of servers. The failure rates and the service rates are generated based on the system (repair shop) utilization rate with an assumption that all SKUs are processed on all servers, i.e., a repair shop design with one cluster and fully flexible servers. The system utilization rate, $\rho$, is the third design factor with levels 0.65 and 0.80. For the chosen utilization rate, we randomly generate two sets of parameters:

(a) the failure rates $\lambda_i$, such that $\sum_{i=1}^{N} \lambda_i = 1$, and
(b) workload percentages $\delta_i$, such that $\sum_{i=1}^{N} \delta_i = 1$.

Using the generated $\lambda_i$ and $\delta_i$, we produce the service rates $\mu_i$ as $\mu_i = \frac{\lambda_i}{\delta_i \rho M}$, where $\delta_i \rho M$ is the total workload of SKU type $i$. The pattern of the holding costs, $h_i$, is the fourth design factor with two variants (levels): (i) IND: completely randomly (independent) within a range $[h_{min}, h_{max}]$, and (ii) HPB: hyperbolically related to the workloads $w_i = \lambda_i / \mu_i = \delta_i \rho M$:

$$h_i = \frac{h_{max} - h_{min} + 10}{9 \frac{w_i - w_{min}}{w_{max} - w_{min}} + 1} - 10 + h_{min} + \xi_i$$

where

$$\xi_i \in U[-\frac{h_{max} - h_{min}}{20}, \frac{h_{max} - h_{min}}{20}],$$
$$w_{min} = \min_{i=1,\ldots,N} w_i \quad \text{and} \quad w_{max} = \max_{i=1,\ldots,N} w_i$$

The parameters of the hyperbolic relation are chosen such that it replicates some of the real-life scenarios where more expensive repairables are repaired less frequently. The minimum holding cost, $h_{min}$, is the fifth factor with levels 1 and 100. The maximum holding cost is fixed at 1,000. The server cost, $f$, and the skill cost, $c_i$, are the last two factors in our DoE. The server cost levels are set as 10,000 and 100,000 ($10h_{max}$ and $100h_{max}$). The skill cost is assumed as 1% or 10% of the chosen server cost for all SKUs. The penalty cost, $b$, is set as fifty-fold of the average holding cost so that about 98% of requests can be met from spare stocks. That means the probability of backorder is only 0.02. The overview of all factors and levels are presented in Table 1.

**Table 1.** Problem parameter variants for test bed [61].

| Factors | Levels |
|---|---|
| No. of SKUs ($N$) | [10, 20] |
| No. of initial servers ($M$) | [5, 10] |
| Utilization rate ($\rho$) | [0.65, 0.80] |
| Minimum holding cost ($h_{min}$) | [1, 100] |
| Maximum holding cost ($h_{max}$) | 1000 |
| Holding cost/Workload relation | [IND, HPB] |
| Server cost ($f$) | [$10h_{max}$, $100h_{max}$] |
| Cross-training cost ($c_i$) | [$0.01f$, $0.10f$] |
| Penalty cost ($b$) | $50\frac{\sum_{i=1}^{N} \lambda_i h_i}{\sum_{i=1}^{N} \lambda_i}$ |

### 5.2 The Benchmarking Algorithm: A Genetic Algorithm

We compare the performance of the proposed pooling heuristic with a Genetic Algorithm (GA)-based methodology. In this method, a GA searches for the optimal pooled repair shop design policy $\mathbf{X}$ as it is depicted in Fig. 2.

The GA is a stochastic optimization technique that is inspired by natural selection and biological evolutionary philosophy. A population of individuals (solutions) is represented by a chromosome, a string of information which is randomly generated [57]. Each chromosome corresponds to a particular repair shop design policy, $\mathbf{X}$. Every chromosome in the population has $N$ genes. The value of the gene indicates the cluster that SKU is assigned into. Each chromosome also carries information about the number of clusters exist in the repair shop. The total number of distinct integer in the chromosome represents the number of clusters.

At each iteration, GA generates a set of feasible pooled repair shop design policies. Afterward, these candidate feasible solutions (policies) are passed through fitness evaluation function to find optimal values of server assignment policy **Z** and inventory levels of spares **S**. In the fitness evaluation, capacity optimization and multi-class multi-server solver sub-routines are invoked exactly the same way as described for the pooling heuristic. GA runs till it reached predefined generation number. The population size, the number of generations, crossover probability, and mutation probability are the input parameters for any GA implementation. We set the population size and the number of generations at 100 and 25, respectively. Besides, the crossover and the mutation parameters are chosen as 0.8 and 0.4, respectively.

### 5.3 Performance Comparison of Pooling Heuristic and GA

We find the optimal pooled designs together with optimal capacity and inventory levels of spares for the cases described above by using the proposed pooling heuristic. We compare the minimum total system cost achieved by pooling heuristic with the cost obtained from GA-based pooling algorithm. We define a cost-ratio metric $\Delta$, a ratio of the total minimum cost obtained from the pooling heuristic to the total minimum cost achieved by GA-based algorithm.

Table 2 presents average values of $\Delta$ under each problem factor and level. First, on an average, pooling heuristic achieves around 3% lower total cost than that of GA-based pooling algorithm. Second, the increasing size of the problem (higher number of SKUs, $N$) leads to more substantial objective function value gaps in favor of the pooling heuristic. It shows that the proposed pooling heuristic

**Table 2.** Total cost comparison of different solution algorithms under varying factors.

| Factor | Levels | Cost-ratio | # of the best cost | |
|---|---|---|---|---|
| | | $\Delta$ | GA-based pooling | Pooling heuristic |
| Number of SKUs ($N$) | 10 | 1.0009 | 17 | 47 |
| | 20 | 0.9331 | 1 | 63 |
| Number of initial servers ($M$) | 5 | 0.9629 | 10 | 54 |
| | 10 | 0.9711 | 8 | 56 |
| Utilization rate ($\rho$) | 0.65 | 0.9646 | 7 | 57 |
| | 0.80 | 0.9694 | 11 | 53 |
| Minimum holding cost ($h_{min}$) | 1 | 0.9649 | 9 | 55 |
| | 100 | 0.9691 | 9 | 55 |
| Holding cost/Work load relation | IND | 0.9695 | 8 | 56 |
| | HPB | 0.9645 | 10 | 54 |
| Server cost ($f$) | $10h_{max}$ | 0.9563 | 5 | 59 |
| | $100h_{max}$ | 0.9776 | 13 | 51 |
| Cross-training cost ($c_i$) | $0.01f$ | 0.9600 | 5 | 59 |
| | $0.1f$ | 0.9740 | 13 | 51 |
| **Overall** | | **0.9670** | **18** | **110** |

conducts a more extensive search in a larger solution space (i.e., higher number of SKUs, $N$). Lastly, the pooling heuristic outperforms the GA-based pooling optimization in 86% of the cases (110 cases out of 128) in the testbed.

We also compare the total system cost with the costs obtained from fully flexible (a single cluster where any SKU can be processed on any server) and dedicated (where the number of clusters equal to the number of SKUs) designs. Table 3 summarizes the cost reduction for both pooling heuristic and GA-based pooling optimization under different problem factors. The repair shop designs found by pooling heuristic can produce approximately 45% and 25% savings on average in comparison with dedicated and fully flexible designs, respectively. In some extreme settings, average cost reduction achieved by pooling heuristic reaches to 55% to that of a dedicated design and 40% to that of a fully flexible design. The repair shop designs suggested by GA-based pooling bring about 44% and 21% total cost reductions compared with dedicated and fully flexible designs, respectively. When the cost of having an extra skill is relatively high compared to that of having an additional server (i.e., the case of cross-training cost being equal to $0.1f$), fully flexible design becomes as good as dedicated design. However, when cross-training cost is relatively small, both of the solution algorithms exhibit worse performance with respect to fully flexible design.

**Table 3.** Average cost reductions in comparison with dedicated and fully flexible systems.

| Factor | Levels | GA-based pooling | | Pooling heuristic | |
|---|---|---|---|---|---|
| | | Dedicated | Fully Flexible | Dedicated | Fully Flexible |
| Number of SKUs ($N$) | 10 | 35.93% | 21.89% | 35.19% | 22.00% |
| | 20 | 52.40% | 21.18% | 55.65% | 28.11% |
| Number of initial servers ($M$) | 5 | 52.62% | 19.37% | 53.76% | 23.43% |
| | 10 | 35.71% | 23.70% | 37.08% | 26.68% |
| Utilization rate ($\rho$) | 0.65 | 46.73% | 23.36% | 48.01% | 24.30% |
| | 0.80 | 41.60% | 19.71% | 42.83% | 25.81% |
| Minimum holding cost ($h_{min}$) | 1 | 44.59% | 21.58% | 45.84% | 24.81% |
| | 100 | 43.75% | 21.49% | 45.00% | 25.30% |
| Holding cost/Work load relation | IND | 43.94% | 21.29% | 44.42% | 24.04% |
| | HPB | 44.39% | 21.79% | 46.42% | 26.08% |
| Server cost ($f$) | $10h_{max}$ | 38.63% | 17.19% | 39.95% | 22.44% |
| | $100h_{max}$ | 49.71% | 25.89% | 50.89% | 27.67% |
| Cross-training cost ($c_i$) | $0.01f$ | 48.81% | 9.54% | 50.22% | 9.93% |
| | $0.1f$ | 39.53% | 33.53% | 40.62% | 40.18% |
| **Average** | | **44.16**% | **21.53**% | **45.42**% | **25.06**% |

Figure 4 shows distributions of the average percentage of cross-training per server for all 128 instances investigated in this paper. We observe that, in most of the instances, the average percentage of cross-training is less than 40%, which shows that partial flexibility; i.e., partial cross-training is usually sufficient for optimal system performance.

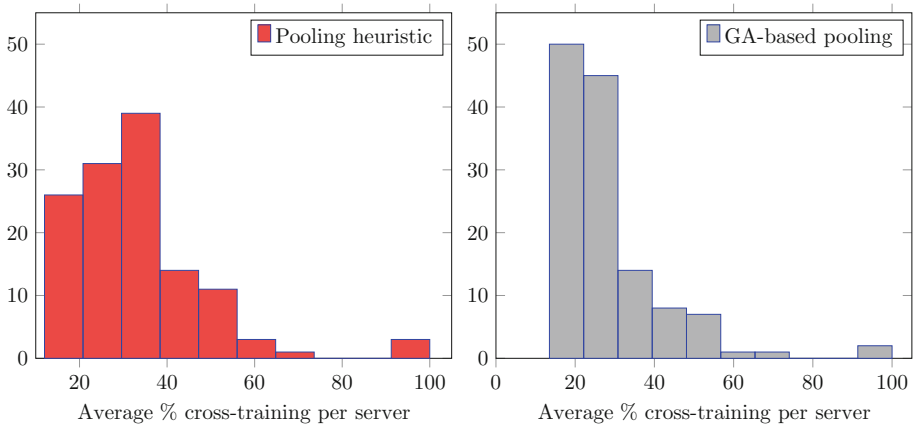


**Fig. 4.** Cross-training analysis of different solution algorithms.

### 5.4 Run Time Comparisons

All the experiments are implemented on a computer with 16 GB RAM and 2.8 GHz i7 CPU. Figure 5 shows boxplots of run time performances for both algorithms. The pooling heuristic converges quite fast in most of the cases and provides the final solution within 5000 cpu seconds with a median run time of 2000 s. GA-based pooling algorithm outperforms the pooling heuristic in terms of run times by achieving under 1000 s median run time. Even the worst run time performances of the algorithms are still acceptable for tactical and operational level decisions in real-life spare part supply systems.

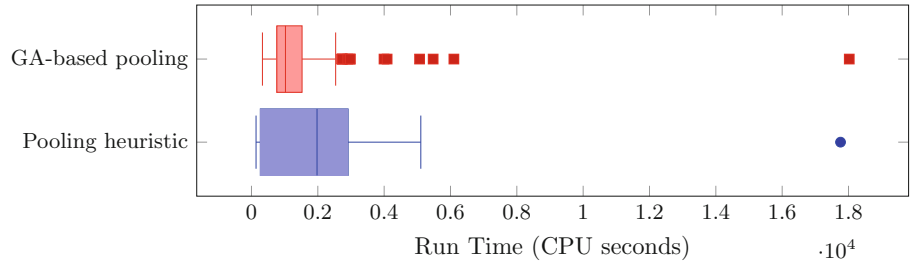


**Fig. 5.** Run time performance comparison of algorithms.

## 6 Conclusions

When designing a spare part supply network for repairable parts that balances cost efficiency with effectiveness, several questions in both strategic and tactical nature have to be answered. In this article, the joint problem of resource pooling,

inventory allocation and capacity level designation of the repair shop is analyzed, and solution heuristics are developed and compared with each other. From the numerical experiments, it can be concluded that pooled designs result in cost savings of around 45% and 25% in comparison to dedicated and fully flexible designs, respectively. Besides, we observe that the optimal repair shop designs can be achieved by partially cross-trained servers.

The results of this research are important to maintenance outsourcing companies and large firms that operate and maintain their own repair facilities. In both cases, the goal of decreasing maintenance costs and reducing the production stoppages and losses would be accomplished.

As further research possibilities, testing the applicability of the methodology with real-life cases (with larger problem sizes; i.e, a larger number of SKUs) would be an invaluable contribution. We plan to develop novel clustering heuristics or meta-heuristics that generate better pooling schemes with less computational complexity. It would be also worthwhile to integrate pooling decision with static and dynamic routing and prioritization rules in the part repair processes.

**Acknowledgement.** This research was made possible by the NPRP award [NPRP 7-308-2-128] from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the author[s].

## References

1. IATA's Maintenance Cost Task Force: Airline maintenance cost: executive commentary (2015). https://www.iata.org/whatwedo/workgroups/Documents/MCTF/AMC-Exec-Comment-FY14.pdf. Accessed 30 Aug 2017
2. Keizer, M.C.O., Teunter, R.H., Veldman, J.: Clustering condition-based maintenance for systems with redundancy and economic dependencies. Eur. J. Oper. Res. **251**(2), 531–540 (2016)
3. López-Santana, E., Akhavan-Tabatabaei, R., Dieulle, L., Labadie, N., Medaglia, A.L.: On the combined maintenance and routing optimization problem. Reliab. Eng. Syst. Saf. **145**, 199–214 (2016)
4. Duffuaa, S.O.: Mathematical models in maintenance planning and scheduling. In: Ben-Daya, M., Duffuaa, S.O., Raouf, A. (eds.) Maintenance, Modeling and Optimization, pp. 39–53. Springer, Boston (2000). https://doi.org/10.1007/978-1-4615-4329-9_2
5. Sherbrooke, C.C.: Metric: a multi-echelon technique for recoverable item control. Oper. Res. **16**(1), 122–141 (1968)
6. Sherbrooke, C.C.: Optimal Inventory Modeling of Systems: Multi-echelon Techniques, vol. 72. Springer, New York (2004). https://doi.org/10.1007/b109856
7. Basten, R., Van Houtum, G.: System-oriented inventory models for spare parts. Surv. Oper. Res. Manag. Sci. **19**(1), 34–55 (2014)
8. Arts, J.: A multi-item approach to repairable stocking and expediting in a fluctuating demand environment. Eur. J. Oper. Res. **256**(1), 102–115 (2017)
9. Diaz, A., Fu, M.C.: Models for multi-echelon repairable item inventory systems with limited repair capacity. Eur. J. Oper. Res. **97**(3), 480–492 (1997)
10. Rappold, J.A., Van Roo, B.D.: Designing multi-echelon service parts networks with finite repair capacity. Eur. J. Oper. Res. **199**(3), 781–792 (2009)

11. Sleptchenko, A., Van der Heijden, M., Van Harten, A.: Trade-off between inventory and repair capacity in spare part networks. J. Oper. Res. Soc. **54**(3), 263–272 (2003)
12. Srivathsan, S., Viswanathan, S.: A queueing-based optimization model for planning inventory of repaired components in a service center. Comput. Ind. Eng. **106**, 373–385 (2017)
13. Sleptchenko, A., Van der Heijden, M., Van Harten, A.: Effects of finite repair capacity in multi-echelon, multi-indenture service part supply systems. Int. J. Prod. Econ. **79**(3), 209–230 (2002)
14. de Smidt-Destombes, K.S., van der Heijden, M.C., van Harten, A.: Joint optimisation of spare part inventory, maintenance frequency and repair capacity for k-out-of-n systems. Int. J. Prod. Econ. **118**(1), 260–268 (2009)
15. de Smidt-Destombes, K.S., van der Heijden, M.C., van Harten, A.: Availability of k-out-of-n systems under block replacement sharing limited spares and repair capacity. Int. J. Prod. Econ. **107**(2), 404–421 (2007)
16. de Smidt-Destombes, K.S., van der Heijden, M.C., Van Harten, A.: On the interaction between maintenance, spare part inventories and repair capacity for a k-out-of-n system with wear-out. Eur. J. Oper. Res. **174**(1), 182–200 (2006)
17. de Smidt-Destombes, K.S., van der Heijden, M.C., van Harten, A.: On the availability of a k-out-of-n system given limited spares and repair capacity under a condition based maintenance strategy. Reliab. Eng. Syst. Saf. **83**(3), 287–300 (2004)
18. Lau, H.C., Song, H.: Multi-echelon repairable item inventory system with limited repair capacity under nonstationary demands. Int. J. Inventory Res. **1**(1), 67–92 (2008)
19. Yoon, H., Jung, S., Lee, S.: The effect analysis of multi-echelon inventory models considering demand rate uncertainty and limited maintenance capacity. Int. J. Oper. Res. **24**(1), 38–58 (2015)
20. Tracht, K., Funke, L., Schneider, D.: Varying repair capacity in a repairable item system. Procedia CIRP **17**, 446–450 (2014)
21. Driessen, M.A., Rustenburg, J.W., van Houtum, G.J., Wiers, V.C.S.: Connecting inventory and repair shop control for repairable items. In: Zijm, H., Klumpp, M., Clausen, U., Hompel, M. (eds.) Logistics and Supply Chain Innovation, pp. 199–221. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-22288-2_12
22. Jordan, W.C., Graves, S.C.: Principles on the benefits of manufacturing process flexibility. Manage. Sci. **41**(4), 577–594 (1995)
23. Jordan, W.C., Inman, R.R., Blumenfeld, D.E.: Chained cross-training of workers for robust performance. IIE Trans. **36**(10), 953–967 (2004)
24. Bassamboo, A., Randhawa, R.S., Mieghem, J.A.V.: A little flexibility is all you need: on the asymptotic value of flexible capacity in parallel queuing systems. Oper. Res. **60**(6), 1423–1435 (2012)
25. Brusco, M.J., Johns, T.R.: Staffing a multiskilled workforce with varying levels of productivity: an analysis of cross-training policies*. Decis. Sci. **29**(2), 499–515 (1998)
26. Brusco, M.J.: An exact algorithm for a workforce allocation problem with application to an analysis of cross-training policies. IIE Trans. **40**(5), 495–508 (2008)
27. Chou, M.C., Chua, G.A., Teo, C.P., Zheng, H.: Design for process flexibility: efficiency of the long chain and sparse structure. Oper. Res. **58**(1), 43–58 (2010)
28. Pinker, E.J., Shumsky, R.A.: The efficiency-quality trade-off of cross-trained workers. Manuf. Serv. Oper. Manag. **2**(1), 32–48 (2000)
29. Tsitsiklis, J.N., Xu, K., et al.: On the power of (even a little) resource pooling. Stoch. Syst. **2**(1), 1–66 (2012)

30. Andradóttir, S., Ayhan, H., Down, D.G.: Design principles for flexible systems. Prod. Oper. Manag. **22**(5), 1144–1156 (2013)
31. Qin, R., Nembhard, D.A., Barnes II, W.L.: Workforce flexibility in operations management. Surv. Oper. Res. Manag. Sci. **20**(1), 19–33 (2015)
32. Hopp, W.J., Tekin, E., Van Oyen, M.P.: Benefits of skill chaining in serial production lines with cross-trained workers. Manage. Sci. **50**(1), 83–98 (2004)
33. Liu, C., Yang, N., Li, W., Lian, J., Evans, S., Yin, Y.: Training and assignment of multi-skilled workers for implementing seru production systems. Int. J. Adv. Manuf. Technol. **69**(5–8), 937–959 (2013)
34. Sayın, S., Karabatı, S.: Assigning cross-trained workers to departments: a two-stage optimization model to maximize utility and skill improvement. Eur. J. Oper. Res. **176**(3), 1643–1658 (2007)
35. Hopp, W.J., Oyen, M.P.: Agile workforce evaluation: a framework for cross-training and coordination. IIE Trans. **36**(10), 919–940 (2004)
36. Li, Q., Gong, J., Fung, R.Y., Tang, J.: Multi-objective optimal cross-training configuration models for an assembly cell using non-dominated sorting genetic algorithm-II. Int. J. Comput. Integr. Manuf. **25**(11), 981–995 (2012)
37. Inman, R.R., Jordan, W.C., Blumenfeld, D.E.: Chained cross-training of assembly line workers. Int. J. Prod. Res. **42**(10), 1899–1910 (2004)
38. Tiwari, M., Roy, D.: Application of an evolutionary fuzzy system for the estimation of workforce deployment and cross-training in an assembly environment. Int. J. Prod. Res. **40**(18), 4651–4674 (2002)
39. Vairaktarakis, G., Winch, J.K.: Worker cross-training in paced assembly lines. Manuf. Serv. Oper. Manag. **1**(2), 112–131 (1999)
40. Slomp, J., Bokhorst, J.A., Molleman, E.: Cross-training in a cellular manufacturing environment. Comput. Ind. Eng. **48**(3), 609–624 (2005)
41. Iravani, S.M., Van Oyen, M.P., Sims, K.T.: Structural flexibility: a new perspective on the design of manufacturing and service operations. Manage. Sci. **51**(2), 151–166 (2005)
42. Bokhorst, J.A., Slomp, J., Molleman, E.: Development and evaluation of cross-training policies for manufacturing teams. IIE Trans. **36**(10), 969–984 (2004)
43. Schneider, M., Grahl, J., Francas, D., Vigo, D.: A problem-adjusted genetic algorithm for flexibility design. Int. J. Prod. Econ. **141**(1), 56–65 (2013)
44. Wallace, R.B., Whitt, W.: A staffing algorithm for call centers with skill-based routing. Manuf. Serv. Oper. Manag. **7**(4), 276–294 (2005)
45. Ahghari, M., Balcioglu, B.: Benefits of cross-training in a skill-based routing contact center with priority queues and impatient customers. IIE Trans. **41**(6), 524–536 (2009)
46. Legros, B., Jouini, O., Dallery, Y.: A flexible architecture for call centers with skill-based routing. Int. J. Prod. Econ. **159**, 192–207 (2015)
47. Tekin, E., Hopp, W.J., Van Oyen, M.P.: Pooling strategies for call center agent cross-training. IIE Trans. **41**(6), 546–561 (2009)
48. Harper, P.R., Powell, N., Williams, J.E.: Modelling the size and skill-mix of hospital nursing teams. J. Oper. Res. Soc. **61**(5), 768–779 (2010)
49. Li, L.L.X., King, B.E.: A healthcare staff decision model considering the effects of staff cross-training. Health Care Manag. Sci. **2**(1), 53–61 (1999)
50. Simmons, D.: The effect of non-linear delay costs on workforce mix. J. Oper. Res. Soc. **64**(11), 1622–1629 (2013)
51. Agnihothri, S.R., Mishra, A.K.: Cross-training decisions in field services with three job types and server-job mismatch. Decis. Sci. **35**(2), 239–257 (2004)

52. Agnihothri, S., Mishra, A., Simmons, D.: Workforce cross-training decisions in field service systems with two job types. J. Oper. Res. Soc. **54**, 410–418 (2003)
53. Colen, P., Lambrecht, M.: Cross-training policies in field services. Int. J. Prod. Econ. **138**(1), 76–88 (2012)
54. Iravani, S.M., Krishnamurthy, V.: Workforce agility in repair and maintenance environments. Manuf. Serv. Oper. Manag. **9**(2), 168–184 (2007)
55. De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E.: Workforce planning incorporating skills: state of the art. Eur. J. Oper. Res. **243**(1), 1–16 (2015)
56. Sleptchenko, A., Turan, H.H., Pokharel, S., ElMekkawy, T.Y.: Cross training policies for repair shops with spare part inventories. Int. J. Prod. Econ. (2018). https://doi.org/10.1016/j.ijpe.2017.12.018
57. Turan, H.H., Pokharel, S., Sleptchenko, A., ElMekkawy, T.Y.: Integrated optimization for stock levels and cross-training schemes with simulation-based genetic algorithm. In: International Conference on Computational Science and Computational Intelligence, pp. 1158–1163 (2016)
58. Sleptchenko, A., Elmekkawy, T.Y., Turan, H.H., Pokharel, S.: Simulation based particle swarm optimization of cross-training policies in spare parts supply systems. In: The Ninth International Conference on Advanced Computational Intelligence (ICACI 2017), pp. 60–65 (2017)
59. Al-Khatib, M., Turan, H.H., Sleptchenko, A.: Optimal skill assignment with modular architecture in spare parts supply systems. In: 4th International Conference on Industrial Engineering and Applications (ICIEA), pp. 136–140. IEEE (2017)
60. Turan, H.H., Sleptchenko, A., Pokharel, S., ElMekkawy, T.Y.: A clustering-based repair shop design for repairable spare part supply systems. Comput. Ind. Eng. **125**, 232–244 (2018)
61. Turan, H.H., Pokharel, S., Sleptchenko, A., ElMekkawy, T.Y., Al-Khatib, M.: A pooling strategy for flexible repair shop designs. In: Proceedings of the 7th International Conference on Operations Research and Enterprise Systems, pp. 272–278 (2018)
62. Van Harten, A., Sleptchenko, A.: On Markovian multi-class, multi-server queueing. Queueing Syst. **43**(4), 307–328 (2003)
63. Altiok, T.: On the phase-type approximations of general distributions. IIE Trans. **17**(2), 110–116 (1985)
64. Van Der Heijden, M., Van Harten, A., Sleptchenko, A.: Approximations for Markovian multi-class queues with preemptive priorities. Oper. Res. Lett. **32**(3), 273–282 (2004)