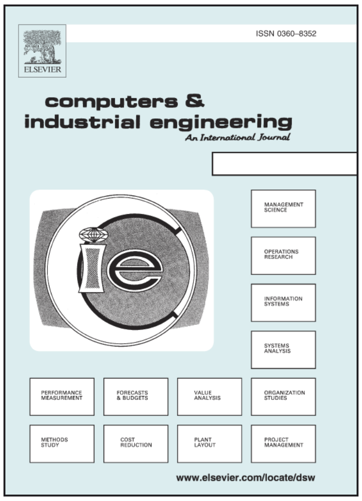

A Clustering-Based Repair Shop Design for Repairable Spare Part Supply Systems

Hasan Hüseyin Turan, Andrei Sleptchenko, Shaligram Pokharel, Tarek Y ElMekkawy

PII: S0360-8352(18)30412-1
DOI: https://doi.org/10.1016/j.cie.2018.08.032
Reference: CAIE 5381

To appear in: *Computers & Industrial Engineering*

Received Date: 30 April 2017
Accepted Date: 28 August 2018

Title Page

Type of Article: Research Article

(i) Title:

A Clustering-Based Repair Shop Design for Repairable Spare Part Supply Systems

(ii) Authors:
Hasan Hüseyin Turan[a], Andrei Sleptchenko[b], Shaligram Pokharel[c], Tarek Y ElMekkawy[c]

(iii) Affiliations:
[a] Capability Systems Centre, School of Engineering and Information Technology, University of New South Wales, Canberra, Australia

[b] Industrial Engineering, Khalifa University of Science, Technology & Research, P.O. Box 127788, Abu Dhabi, United Arab Emirates

[c] Department of Mechanical and Industrial Engineering, Qatar University, Doha 2713, Qatar

(iv) Corresponding Author:
Hasan Hüseyin Turan
Capability Systems Centre, School of Engineering and Information Technology, University of New South Wales, Canberra, Australia
E-mail: h.turan@adfa.edu.au

A Clustering-Based Repair Shop Design for Repairable Spare Part Supply Systems

Abstract

In this study, we address the design problem of a single repair shop in a repairable multi-item spare part supply system. We propose a sequential solution heuristic to solve the joint problem of resource pooling, inventory allocation, and capacity level designation of the repair shop with stochastic failure and repair time of repairables. The pooling strategies to obtain repair shop clusters/cells are handled by a K-median algorithm by taking into account the repair time and the holding cost of each repairable spare part. We find that the decomposition of the repair shop in sub-systems by clustering reduces the complexity of the problem and enables the use of queue-theoretical approximations to optimize the inventory and capacity levels. The effectiveness of the proposed approach is analyzed with several numerical experiments. The repair shop designs suggested by the approach provide around 10% and 30% cost reductions on an average when compared to fully flexible and totally dedicated designs, respectively. We also explore the impact of several input parameters and different clustering rules on the performance of the methodology and provide managerial insights.

Keywords: Spare part logistics, repair shop, pooling, K-median, queuing approximation, heuristic.

1. Introduction

Many companies face a great pressure to reduce their production costs to stay competitive. One of the main expenditure items for business is the maintenance cost which can be anywhere from 15% up to 70% of the total production cost (Wang et al., 2008). For example, Figure 1 depicts average maintenance cost and distribution of maintenance terms for critical components in the aviation industry. It is also reported that the US Navy spends more than $200 billion each year for aviation equipment maintenance, which accounts for 14.2% of all defense spending (Fang and Zhaodong, 2015).

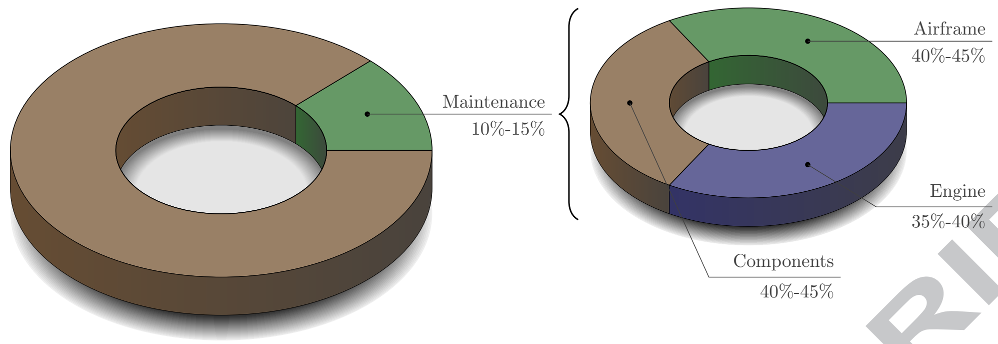

Figure 1: Airline operation and maintenance cost distributions [1].

Both planned (proactive) and unplanned/corrective (reactive) maintenance strategies have been performed for decades in order to reduce downtimes leading to the production losses. In this paper, we focus on corrective maintenance of expensive repairable capital assets, where malfunctioned parts or components of assets are immediately replaced by ready-for-use spares from the stock. The failed components are sent to the repair shop, and once the repair is completed, they are forwarded back to stock "as-good-as-new". If there is no available ready-for-use spare at the stock point, the request is back ordered and met as soon as possible. At the same time, the backorders should be avoided as much as possible (to avoid downtimes or backorder costs) and the total costs should be minimized (see ref. Sherbrooke (1968); Basten and Van Houtum (2014) and Van Houtum and Kranenburg (2015)).

The effectiveness of corrective maintenance strategies strongly depends on the design of underlying spare part supply system. The number of echelons, inventory levels, and lateral supply options are some of the decision factors considered during the design of multi-echelon spare part supply networks (Sherbrooke, 2006; Muckstadt, 2004; Van Houtum and Kranenburg, 2015). At the repair shop level, the spare part inventory levels, replenishment policies, repair capacities and the repair priorities are some of the important factors that are needed to be taken into account (van Jaarsveld et al., 2015; Sleptchenko et al., 2003, 2005). Decisions that are made about the repair shop heavily influence the number of spare parts that are required; enabling faster repairs by optimal design of the repair shop can mean less spare parts required to achieve the same service level. Hence, we present an optimization

[1] `http://ataibaviationservices.co.uk/aircraft-engine-maintenance/`

model that simultaneously takes into account both the spare inventory levels and the repair shop design.

In this paper, we consider a single-location spare part supply system consisting of a repair shop and storage facility for several types of repairable parts, or stock keeping units (SKUs). Such a facility can be a downstream facility supporting the clients directly or handling the line replaceable units (LRU). It can also be a higher echelon or a lower indenture repair facility handling shop-replaceable units (SRUs) at a base or a depot. In all these cases, the backorders should be treated differently. The backorders typically have different levels of severity depending on the structure of the installed system (e.g., due to redundancy Sleptchenko and Van der Heijden , 2016), or the location of the single-location supply point in the supply system. At the same time, the severity of the back ordering situation can often be converted into backorder costs. These costs could be direct downtime costs (e.g., production loss), as well as indirect costs that were needed to compensate the backorders at lower echelons. One can also assume that the backorder costs can be computed as the Lagrangian multipliers appearing in the problem decomposition discussed in Van Houtum and Kranenburg (2015), Section 2.5. In this paper, the main scope is the analysis of the internal design of the single location repair-inventory facility, independent of the general layout of the supply chain or the installed base.

As discussed above, the performance of the repair shop depends on a variety of factors (van Jaarsveld et al., 2015). The internal design, however, is one of the most decisive factors in the performance of the whole system. In Sleptchenko et al. (2003), the general effect of the repair capacity is analyzed, and it is indicated that the optimal utilization rate can be relatively high, especially for the expensive repair servers. However, the authors assume that all servers can handle all SKUs. In many real-life situations, however, the ability of servers to handle all SKUs may not be feasible or not efficient due to different costs arising from unavailability of servers capable of handling all of the failures types (Tekin et al., 2009; Qin et al., 2015). Therefore, in Turan et al. (2016) and Sleptchenko et al. (2017a,b), the authors extent the analysis of optimal repair shops capacity to the cases when different servers are capable (or allowed) to process different sets of SKUs, so-called partial cross-training of servers (i.e., an arbitrary design between totally dedicated and full cross-training such as designs depicted in

Figure 2b and 2c). Their presented results indicate that the partial cross-training can be much more cost-efficient in comparison to the entirely dedicated servers or full cross-training. At the same time, optimization of the system with partial cross-training is much harder due to the lack of analytical models for queuing systems with cross-training.

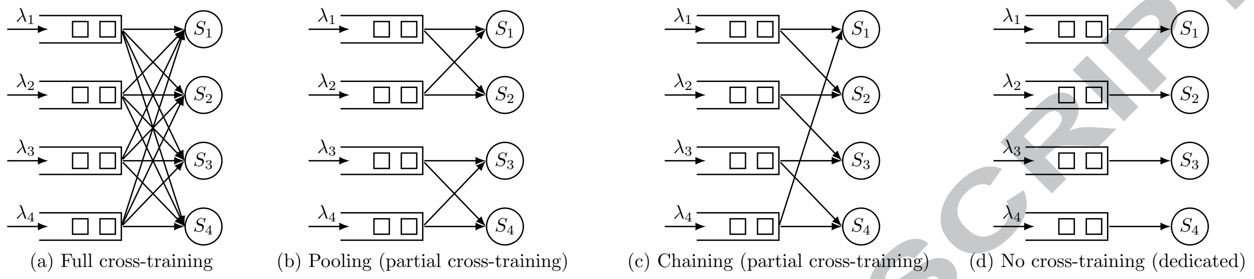
λ_1
λ_2
λ_3
λ_4
S_1
S_2
S_3
S_4
(a) Full cross-training
(b) Pooling (partial cross-training)
(c) Chaining (partial cross-training)
(d) No cross-training (dedicated)

Figure 2: Different repair shop designs.

In this paper, we analyze the performance of single-location spare part supply systems with an intermediate design scheme known as "*pooling*" (Figure 2b). The pooling in the context of this paper means clustering the SKUs in the repair shops such that each SKU is unambiguously assigned to a single cluster (cell), and every server inside a cluster has the ability to repair all SKUs that are assigned to that cluster. Nonetheless, it is both an important and tough combinatorial optimization question to determine which SKUs to pool together. This paper addresses the following fundamental issues: (i) how many clusters to pool, (ii) which SKUs to pool together, and (iii) how to assign servers (capacity) to each cluster by considering the trade-off between pooling and spare parts inventories.

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. In this sense, contributions of this work include:

1. designing a cost-effective spare part supply system via implementing resource pooling strategies in the repair shop,
2. formulation of an integrated optimization of resource capacities at each cluster/cell (each pooled cell) and inventory levels of all spares,
3. development of an efficient sequential solution approach that integrates queuing theory

and clustering methodology together with a greedy heuristic,

4. performing numerical analysis using different scenarios where the system behavior of designed repair shop is analyzed to show the effectiveness of pooling compared to other design options, and

5. providing a rule of thumb that is implementable by the decision makers/industrial practitioners for the design of a repair shop.

The rest of the paper is organized as follows. In Section 2, a review of related literature is provided. In Section 3, the proposed stochastic mixed integer nonlinear model is presented. The solution algorithm for the proposed model is discussed in Section 4. Section 5 provides an extensive computational study under different scenarios and input settings. Conclusions and future research directions are summarized in Section 6.

2. Review of Related Research

The optimization problem analyzed in this paper lies on the intersection of the design of a flexible/cross-trained repair shop in a spare part supply system and optimization of resources mainly the capacity of the repair shop and inventory levels of spares. This section is divided into three subsections to summarize the related research. Subsection 2.1 provides a summary of studies related to capacity decisions in the spare part and maintenance networks. Subsection 2.2 summarizes the essential results from different application domains related to flexibility and cross-training. The studies on pooling of spares and capacity are summarized in Subsection 2.3.

2.1. On capacity optimization/analyses in the repair shops

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 Sherbrooke (1968). The METRIC based models assume that the repair capacity is infinite. This assumption may not be appropriate in the most industrial settings. Thus, several researchers have relaxed ample repair capacity assumption. To our knowledge, Diaz and Fu (1997) are the first to extend Sherbrooke (1968) by explicitly considering finite repair service capacity. Diaz and Fu (1997) model the repair shop as an $M/M/k$ queuing system for a single repairable item. Rappold and Van Roo (2009) also study a single item spare part supply system under limited capacity constraint and try to find the optimal locations of repair shops in the supply system. The single item finite capacity model is expanded to a multi-item model by Sleptchenko et al. (2003) and Sleptchenko et al. (2002), where the systems are analyzed as a multi-class multi-server $M/M/k$ queuing model. The multi-class multi-server $M/M/k$ queuing model is also used by Srivathsan and Viswanathan (2017) to optimize repair capacity and inventory levels in a service center by assuming that not all failed spares can be repaired.

2.2. On flexibility and cross-training of resources

The research by Jordan and Graves (1995) is the first to consider the design and effectiveness of flexibility. Their analysis shows that limited flexibility yields most of the benefits of the total flexibility. Researchers have shown that *"little or limited flexibility"* is usually sufficient for optimal system performance (see ref. Jordan et al. (2004); Bassamboo et al. (2012); Tsitsiklis et al. (2012)).

Cross-training is one of the more widely discussed capacity/workforce flexibility methods for complex systems (Qin et al., 2015). Production lines (Liu et al., 2013), job shops (Sayın and Karabatı, 2007), flow/assembly shops (Li et al., 2012; Inman et al., 2004), manufacturing (Slomp et al., 2005; Schneider et al., 2013), call centers (Wallace and Whitt, 2005; Legros et al., 2015), health care (Harper et al., 2010), field services (Simmons, 2013; Colen and Lambrecht, 2012) and maintenance/repair (Iravani and Krishnamurthy, 2007) are the some examples of systems where cross-training is applied. Even though the cross-training of resources are extensively studied in the aforementioned domains, to the best of our knowledge, no result has been presented by analyzing cross-training in spare part supply systems other than the works of Turan et al. (2016) and Sleptchenko et al. (2017a,b). These works do not restrict search space of the problem (cross-training policies) by pooled design as depicted in Figure2b. However, there are no analytical models to analyze all possible combination of cross-training policies. Thus, they use a discrete-event simulation model integrated with meta-heuristic algorithms. In addition, aforementioned articles only try to find the best cross-training policies of resources under the fixed capacity of the repair shop. In other words, capacity optimization decisions are not taken into account.

erent applications

2.3. On pooling/clustering of resources and inventories

Pooling or clustering of resources is a partial cross-training policy that is convenient for systems where tasks can be pooled in sets because of their task similarities. To determine which tasks to pool together is a significant combinatorial optimization question. Thus, different strategies from several domains are proposed. Aksin et al. (2007) discuss the design of pooling strategies in call centers. Similarly, Tekin et al. (2009) present pooling policies for a call center by cross-training. The authors show that pooling the departments with highest service time coefficients of variation reduces the expected waiting times more when mean service times are similar among the departments. Pooling of repair shop capacities is studied by Sahba et al. (2011); they investigate whether the repair shop pooling is more cost effective than having dedicated on-site repair shops to serve different fleets of machines at different locations.

In addition to the pooling of resources, it is possible to pool inventories in order to reduce the total system cost. Thus, spare parts pooling is increasingly mentioned as a plausible approach for optimizing spare parts management, more so for repairable systems (see ref. Karsten and Basten (2014); Chemweno et al. (2015) and Wong et al. (2005)). This line of research concludes that it is more cost-effective to share one stock (complete pooling of inventories) point with all parties involved. The conducted literature review on resource and inventory pooling policies reveals the fact that although there exist studies on pooling of inventories in repairable spare part systems, a very limited number of research results considers pooling of resources in spare part systems.

3. Problem Description and Optimization Model

In this section, we define the problem, discuss assumptions, formulate our optimization model and comment on its complexity.

3.1. System description

In this work, we consider a single location system with one repair facility, multiple repairable stock keeping units (SKUs), and stock points, where parts of multiple, critical repairables are kept on stock to serve a collection of technical systems installed in a certain region; this collection of technical systems is also known as the installed base (see Figure 3). The configuration of the technical systems include SKUs, and SKUs are subject to random

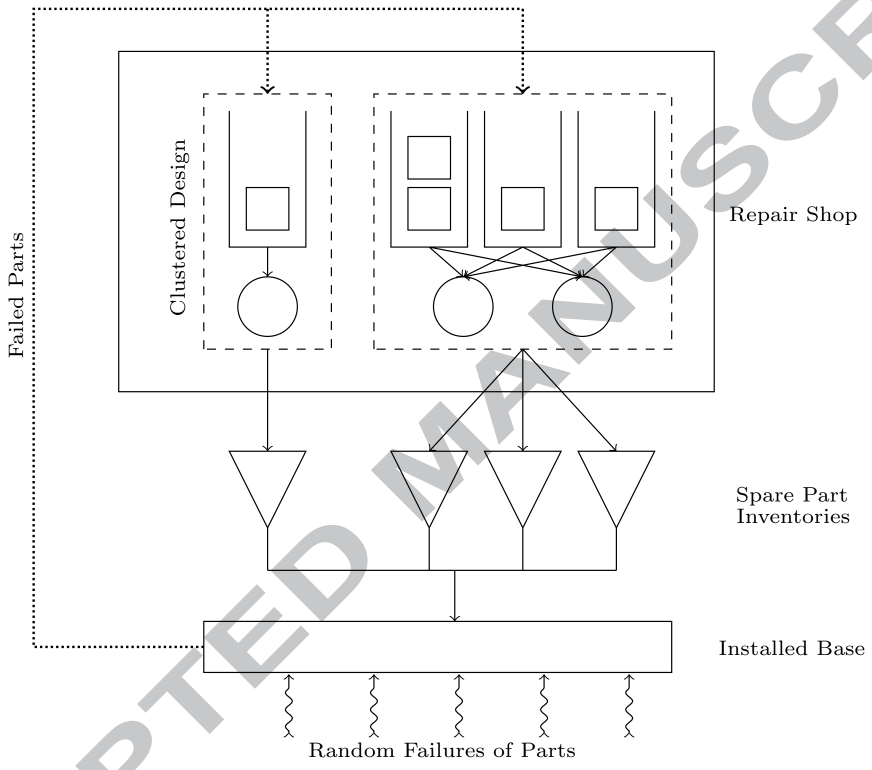

Figure 3: An example of pooled repair shop design for a repairable spare part supply system.

failures. When a part fails in a technical system, an order is immediately placed for a ready-for-use part of the same SKU at the stock point, and the failed part is sent to the repair shop. The repair shop may have pooled structure with one or more cells/clusters. The failed part is directed to its designated cluster in the repair shop, where it is repaired by either a dedicated or a cross-trained server. At the same time, the demand is immediately fulfilled if the demanded ready-for-use part is on the stock. Otherwise, the demand is backordered and fulfilled as soon as a ready-for-use part of the demanded SKU becomes available. In the latter case, the technical system goes down, and downtime cost occurs till the requested ready-for-use part is delivered.

Figure 3 depicts an example of such a pooled repair shop with four SKUs, two clusters and three servers. The first cluster has a dedicated server with an ability to serve one SKU. On the other hand, the second cluster is obtained by pooling remaining three SKUs, and this cluster has two cross-trained servers to serve three different SKUs.

3.2. Discussion of important assumptions

In this paper, we proceed from commonly used assumptions in the repairable spare part supply systems (see ref. Sherbrooke (2006); Tiemessen and van Houtum (2013); Van der Heijden et al. (2013) and Sleptchenko et al. (2003) and assumption listed therein):

- The failures of in-use (installed) parts occur with constant rates according to a Poisson process and are mutually independent from each other. This assumption is quite common and realistic for large installed bases where hazard rates are relatively stable.
- The repair times are exponentially distributed and mutually independent. The expected repair times depend on the SKUs and are independent of the processing server. The exponentially distributed service time is a common assumption in the queuing systems in general and in the maintenance system in particular. This assumption is often not realistic and using Erlang-K type distribution (or any other) with a squared coefficient of variation less than 1 would lead to lower numbers of items in repair and, consequently, lower inventory levels necessary to keep the same service levels. However, this assumption allows for easier computations in the necessary probability distributions in the underlying queuing systems, in particular, the multi-class multi-server queuing system that is discussed in subsection 4.2.1.
- First come first served (FCFS) queuing discipline is adopted inside each and every cluster, and no priorities exist among the failed spares. This discipline implies that whenever a server gets idle, it picks the failed part that has the longest waiting time in the queue, as long as the server is able to handle it (e.g., has the needed skill).
- For all parts $(s-1, s)$ one-for-one replenishment policy is used, i.e. the stock level equals s and each demand immediately generates an order for a replacement part; as a consequence, there is no batching.

- The total holding costs for every SKU per time unit are linear in the initial inventory levels (initially acquired inventory).

- Spare inventories are always replenished directly from the repair shop, i.e., there are no possibilities of neither emergency nor lateral shipments of spares from other locations and/or repairs shops at different echelons.

- Penalty costs (or backorder costs) occur when the required part is not available and are paid per time unit per not available SKU. In many cases, the number of backorders is directly reflecting the number of the technical systems that are down at the installed base.

- A positive cross-training (or flexibility) cost occurs whenever an additional skill is assigned to a server. In the context of the analyzed system, skills have broader definition. Typically skills mean having necessary knowledge or certification of the repairmen. At the same time, we assume, that it can mean having necessary tools or equipment upgrades in the repair shops.

- There is no different degree of cross-training; i.e., any skill assigned to a server considered as primary skill. In other words, there is no competency difference with respect to assigned skills to a server. That is, servers have full capability to repair failed parts if they have the necessary skill.

- The experience on the similar skills or assigning similar skills to a server (i.e., able to repair SKUs that have close physical properties) does not lead to either a decrease in cross-training cost or increase in the service rates of servers. In other words, the effect of learning through experience is not considered.

- The average lifetime of the expensive critical systems is assumed to be long (e.g., 20–30 years) (Tiemessen and van Houtum, 2013). Hence, the expected backorders are determined using steady-state probabilities on an infinite time planning horizon.

- 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.

- The clusters inside the repair shop are mutually exclusive (disjoint) and collectively exhaustive, i.e., a particular failed SKU can be repaired at exactly one cluster, and all SKUs are assigned to exactly one cluster.

The last two assumptions limit the computational complexity of the system and enable using queue-theoretical approximations to find the steady-state probability distribution of items in the system.

3.3. Optimization model

All used sets, parameters and decision variables for the developed formulations and solution procedures are presented as follows.

Indexes:

$\mathcal{N}$: The index set of distinct type of repairables (SKUs) numbered through 1 to N.

Decision variables:

S_i: Amount of initial inventory (basestock level) kept on stock for SKU 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 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:

λ_i Failure rate of SKU i ($i = 1, \ldots, N$),

μ_i: Service rate of SKU i ($i = 1, \ldots, N$),

h_i: Inventory holding cost of SKU i per unit time per part ($i = 1, \ldots, N$),

b: Penalty cost for each backordered 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 of a technician) ,

c_i: Cost of having the skill to repair SKU i per unit time per server ($i = 1, \ldots, N$) (e.g. annual qualification bonus of a technician),

ϵ: Very small positive real number.

The Master Problem ($\mathcal{MP}$) formulation for optimization model is presented between Eqs.(1)-(7). The objective function of $\mathcal{MP}$ is denoted by Eq.(1). The objective function has four cost terms; namely, server (capacity), training, holding, and backorder costs. Eq.(1) takes into account several trade-offs between 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 both/either dedicated and/or cross-trained servers.

$$\min_{\mathbf{S},\, \mathbf{X},\, \mathbf{Z}} \quad \underbrace{\sum_{k=1}^{y} f z_k}_{Cost\ of\ Servers} + \underbrace{\sum_{k=1}^{y} z_k \left(\sum_{i=1}^{N} c_i x_{ik} \right)}_{Cost\ of\ Training} + \underbrace{\sum_{i=1}^{N} h_i S_i}_{Holding\ Cost} + \underbrace{b \sum_{i=1}^{N} \mathbb{EBO}_i \left[S_i, \mathbf{X}, \mathbf{Z} \right]}_{Expected\ Backorder\ Cost} \qquad (1)$$

In objective function (1), all the cost terms are expressed in per unit of time (e.g., a year). The cost factors related to the *servers* and *training* (f and c_i) can represent, for example, annual wages of the technicians and bonuses for their qualifications, or annual operational costs of basic and upgraded equipment. In addition, the *inventory holding* cost term is formed from the initial stock levels, as we often see in spare parts logistics. This assumption is mainly due to the high acquisition (and depreciation) related costs in comparison to the real holding costs. One can think of the cost factors h_i as the expenses associated with having the corresponding spare part on company's balance sheets.

The *penalty (backorder)* cost term is calculated using the penalty cost b and the expected total number of backordered parts $\mathbb{EBO}_i \left[S_i, \mathbf{X}, \mathbf{Z} \right]$ for each SKU 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 **Z**. The variable **X** represents a $(N \times y)$–matrix of the binary decision variables x_{ik} denoting how SKUs are pooled in the repair shop, and the variable **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. As an illustrative example, for the system presented Figure 3, we have the following values:

$$\mathbf{X} = \begin{array}{c|c} \mathbf{X}_1 & \mathbf{X}_2 \\ \end{array} \left[\begin{array}{c|c} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{array}\right] \text{ and } \mathbf{Z} = [1,\ 2]$$

The $\mathcal{MP}$ satisfies two properties: (i) clusters are mutually exclusive and totally exhaustive, and (ii) queues in each cluster are in finite queue length at steady-state (preventing overloading of repair shop). The first is guaranteed by appropriate assignment of decision variables x_{ik}, and the second is ensured by careful selection of z_k values for the given **X**. Hence, the $\mathcal{MP}$ model has the following set of constraints:

$$\sum_{k=1}^{y} x_{ik} = 1 \qquad i = 1, \dots, N \tag{2}$$

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

$$x_{ik} \in \{0,1\} \qquad i = 1, \dots, N \;\; k = 1, \dots, y \tag{4}$$

$$z_k \in \mathbb{Z}^+ \qquad k = 1, \dots, y \tag{5}$$

$$S_i \in \mathbb{N}_0 \qquad i = 1, \dots, N \tag{6}$$

$$y \in \{1, \dots, N\} \tag{7}$$

In this system:

- constraints (2) and (4) ensure that pooling scheme **X** satisfies mutually exclusive and total exhaustive condition for each cluster, i.e., any SKU being repaired by exactly one cluster,

- constraint (3) together with constraint (5) guarantee the system stability by assigning sufficient number of servers to each cluster,
- constraints between Eqs.(4)-(7) are required for non-negativity and integrality of the variables.

For a stable 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 ϵ parameter.

3.4. On the problem complexity

Proposition 1. *An arbitrary pooling policy* $\mathbf{X}$ *is a feasible solution to the* $\mathcal{MP}$ *iff it corresponds to a partition of the index set of SKUs* ($\mathcal{N}$)*.*

Proof. A partition of SKU index set $\mathcal{N} = \{1, 2, \ldots, N\}$ is a subset division of the set into subsets that are *disjoint* and *exhaustive* i.e., every member (SKU) of $\mathcal{N}$ must belong to one and only one of the subsets. Therefore, the index subsets $\mathcal{X}_k$ in the partition corresponding to indexes of SKUs in the clusters $\mathbf{X}_k$ ($\forall k \leq y$) have to satisfy: (I) $\bigcup_{k \leq y} \mathcal{X}_k = \mathcal{N}$ and (II) $\mathcal{X}_{k_1} \bigcap_{k_1 \neq k_2} \mathcal{X}_{k_2} = \emptyset \; \forall (k_1, k_2) \leq y$. Henceforth, Eqs.(2) and (4) in $\mathcal{MP}$ are ensured, and legitimacy of pooling policy $\mathbf{X}$ is guaranteed. □

The number of ways $|\mathcal{P}(N)|$ a set of N elements can be partitioned into non-empty subsets is called a Bell number (Weisstein, 2006). Table 1 indicates the increase in the search space (numbers of pooling schemes $\mathbf{X}$) for increasing number of SKUs in the system.

Table 1: Possible numbers of pooling schemes as a function of the number of SKUs (N)

Number of SKUs (N)	$\lvert\mathcal{P}(N)\rvert$
5	52
10	11×10^4
15	13×10^8
20	51×10^{12}
25	46×10^{17}

Due to the large size of the search space for pooling policies/clustered repair shop designs $\mathbf{X}$, it is not efficient (and often not possible) to find the optimal policy with traditional optimization methods. Even for a small problem size, due to the intensive procedures used in obtaining steady-state probabilities, the evaluation of $\mathbb{EBO}_i\left[S_i, \mathbf{X}, \mathbf{Z}\right]$ becomes cumbersome. Therefore, we propose an algorithm that systematically checks a small subset of search space and uses approximations to calculate steady-state probabilities.

4. Decomposition of Problem and Sequential Solution Algorithm

We search for the optimal values of decisions variables sequentially by fixing the values of some decision variables and optimizing the remaining ones. We first find feasible partitions of SKUs, i.e., pooling policies $\mathbf{X}$ by using the K-median algorithm. Then we optimize the capacity levels $\mathbf{Z}$ and basestock inventory levels $\mathbf{S}$ for each cluster. Figure 4 presents the flow of the proposed solution approach together with its sub-routines (modules) and their interactions with each other. Solution approach consists of three sub-routines, namely, pooling policy generator, capacity optimization module and inventory level optimization module. The details of each module are provided in the following subsections.

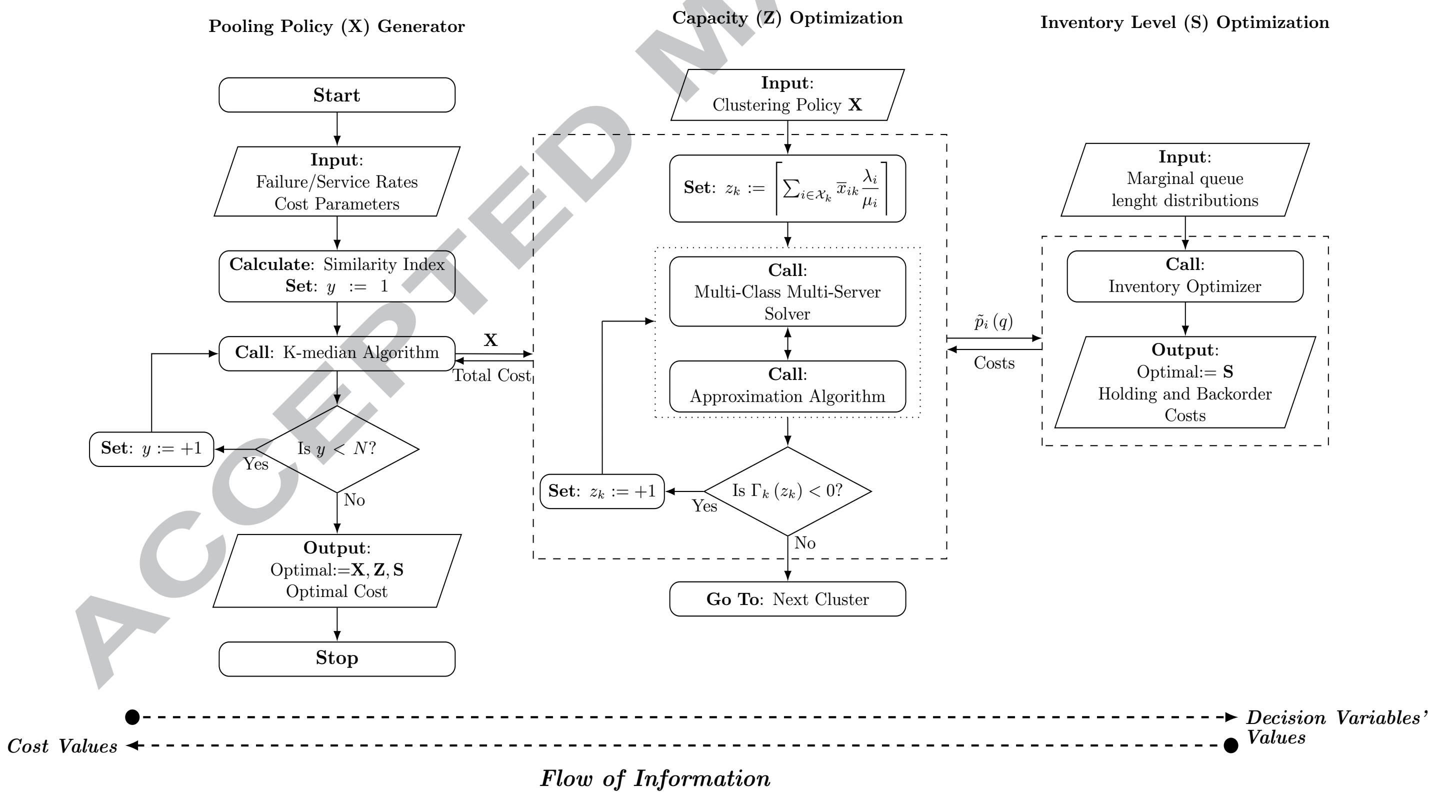
Pooling Policy (X) Generator
Capacity (Z) Optimization
Inventory Level (S) Optimization
Start
Input: Failure/Service Rates Cost Parameters
Calculate: Similarity Index Set: $y := 1$
Call: K-median Algorithm
X
Total Cost
Set: $y := +1$
Is $y < N$?
Yes
No
Output: Optimal:=$\mathbf{X}, \mathbf{Z}, \mathbf{S}$ Optimal Cost
Stop
Input: Clustering Policy X
Set: $z_k := \left\lceil \sum_{i \in \mathcal{X}_k} \bar{x}_{ik} \frac{\lambda_i}{\mu_i} \right\rceil$
Call: Multi-Class Multi-Server Solver
Call: Approximation Algorithm
Set: $z_k := +1$
Is $\Gamma_k(z_k) < 0$?
Go To: Next Cluster
$\tilde{p}_i(q)$
Costs
Input: Marginal queue lenght distributions
Call: Inventory Optimizer
Output: Optimal:= S Holding and Backorder Costs
Decision Variables' Values
Cost Values
Flow of Information

Figure 4: Flow of the proposed sequential solution algorithm.

4.1. Generation of pooling policies

In order to find feasible partitions of SKUs into clusters, we use the K-median algorithm and its integer programming formulation. The K-median requires two sets of inputs; the number of the clusters y that SKUs are partitioned into and a measure indicating similarity of SKUs. The possible number of the clusters y (from one to N) is supplied to the algorithm during the generation of clusters. We adopt and test the effects of three different similarity measures/indexes μ, $h \times \mu$ and h on the $\mathcal{MP}$. Under the similarity index μ, SKUs closer in service rates have a higher chance to be in the same cluster. This index is aimed at decreasing the variations in the service times of SKUs in a particular cluster and increase the throughput of each cluster. The decrease in the variation of service times leads to a decrease in the number of failed parts waiting for repair in the cluster and thus eventually lower the number of the backorders and total cost. The measure $h \times \mu$ takes into account both the holding costs and the service rates during the generation of clusters. This rule assigns SKUs with a closer workload (in terms of costs) to the same cluster. This rule is widely studied in stochastic scheduling problems and optimally of it under some conditions is proved (see ref. Smith (1956)). On the other hand, h rule is aimed at clustering SKUs that have closer holding cost (h), which allows allocation of more repair capacity to the most expensive SKUs. Thus, some decrease in holding costs is expected.

The K-median problem consists of identifying a subset (partition) $\mathcal{X} \subseteq \mathcal{N}$, $|\mathcal{X}| = y$ in order to minimize:

$$\sum_{i \in \mathcal{N}} \min_{j \in \mathcal{X}} \vartheta_{ij}$$

where $y\,(\leq N)$ is a positive integer (corresponding the number of clusters in the partition), and ϑ_{ij} is the *"shortest distance"* or *"similarity"* between two SKUs i and $j\,(\in \mathcal{N})$. The shortest distance/similarity between two SKUs under chosen similarity measure is defined as follows:

(i) $\vartheta_{ij} = (\mu_i - \mu_j)^2$ for μ rule

(ii) $\vartheta_{ij} = (h_i \times \mu_i - h_j \times \mu_j)^2$ for $h \times \mu$ rule[2]

(iii) $\vartheta_{ij} = (h_i - h_j)^2$ for h rule

[2]This rule known as $c\mu$-rule or Smith's-rule

The integer programming formulation of K-median is given between Eqs.(8)-(12). Integer decision variables are defined as follows:

$\nu_j = 1$, if SKU j is selected as a median of cluster, otherwise 0

$\xi_{ij} = 1$, if SKU j and SKU i pooled together(in the same cluster), otherwise 0.

$$\min \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} \vartheta_{ij}\xi_{ij} \tag{8}$$

$$\text{Subject to :} \sum_{j\in\mathcal{N}} \xi_{ij} = 1, \qquad i \in \mathcal{N} \tag{9}$$

$$\sum_{j\in\mathcal{N}} \nu_j = y \tag{10}$$

$$0 \le \xi_{ij} \le \nu_j \le 1 \qquad i,j \in \mathcal{N} \tag{11}$$

$$\xi_{ij}, \nu_j \in \{0,1\} \qquad i,j \in \mathcal{N} \tag{12}$$

K-median algorithm generates a feasible pooling policy **X** (as a result of Proposition 1) for $\mathcal{MP}$ problem under given number of cluster y in the repair shop. For every feasible policy **X**, each cluster can be analyzed and optimized separately due to the clusters being mutually exclusive and independent from each other.

4.2. Capacity and basestock inventory optimization

As seen in Figure 4, the capacity optimization module requires the output from the K-median. The capacity optimization module allocates the optimal number of server **Z** to each cluster. For given $\overline{\mathbf{X}}$[3], $\mathcal{MP}$ is reduced to a capacity and inventory optimization problem $\mathcal{CIO}\left(\overline{\mathbf{X}}\right)$ as follows:

$$\mathcal{CIO}\left(\overline{\mathbf{X}}\right) = \min_{\mathbf{S},\,\mathbf{Z}} \sum_{k=1}^{\overline{y}} z_k\Big(f + C_k\left(\overline{\mathbf{X}}\right)\Big) + \sum_{i=1}^{N} h_i S_i + b\sum_{i=1}^{N} \mathbb{EBO}_i\left[S_i, \mathbf{Z}, \overline{\mathbf{X}}\right]$$

where $C_k\left(\overline{\mathbf{X}}\right)$ indicates cross-training cost per server at cluster k for given policy $\overline{\mathbf{X}}$:

$$C_k\left(\overline{\mathbf{X}}\right) = \sum_{i\in\mathcal{X}_k} c_i \overline{x}_{ik} \qquad k = 1,\ldots,\overline{y}$$

[3]From now on fixed values of decision variables will be overlined by bar $^-$notation.

From independence of clusters, we can separate and rewrite the $\mathcal{CIO}\left(\overline{\mathbf{X}}\right)$ for each cluster k with the constraints as follows:

$$\mathcal{CIO}_k\left(\overline{\mathbf{X}}\right) = \min_{\substack{S_i,\ z_k\\ i\in\mathcal{X}_k}} \ z_k\Big(f + C_k\left(\overline{\mathbf{X}}\right)\Big) + \sum_{i\in\mathcal{X}_k} h_i S_i + b\sum_{i\in\mathcal{X}_k} \mathbb{EBO}_i\left[S_i, z_k, \overline{\mathbf{X}}\right] \tag{13}$$

$$\text{Subject to: } z_k \geq \left\lceil \sum_{i\in\mathcal{X}_k} \overline{x}_{ik}\frac{\lambda_i}{\mu_i}\right\rceil \tag{14}$$

$$z_k \in \mathbb{Z}^+ \tag{15}$$

$$S_i \in \mathbb{N}_0 \qquad\qquad i \in \mathcal{X}_k \tag{16}$$

where $\mathcal{X}_k$ is the index set of the SKUs in the cluster k $\left(\bigcup_{k=1}^{\overline{y}} \mathcal{X}_k = \mathcal{N}\right)$ and $\mathcal{CIO}\left(\overline{\mathbf{X}}\right) = \sum_{k=1}^{\overline{y}} \mathcal{CIO}_k\left(\overline{\mathbf{X}}\right)$. The objective function (13) takes into account trade-off between adding an extra server to a cluster and decreasing inventories in that cluster. In this context, Algorithm 1 performs a greedy search to find the optimal z_k by fixing server numbers at each cluster and solving following optimization subproblems:

$$\mathcal{CIO}_k\left(\overline{\mathbf{X}}, \overline{z}_k\right) = \left\{ \min_{S_i}\Big(\sum_{i\in\mathcal{X}_k} h_i S_i + b\mathbb{EBO}_i\left[S_i, \overline{z}_k, \overline{\mathbf{X}}\right]\Big) | S_i \in \mathbb{N}_0 \right\} \qquad k = 1,\ldots,\overline{y} \tag{17}$$

The Objective function of model 17 has separable structure and it can be rewritten as follows:

$$\mathcal{CIO}_k\left(\overline{\mathbf{X}}, \overline{z}_k\right) = \left\{ \sum_{i\in\mathcal{X}_k} \min_{S_i}\Big(h_i S_i + b\mathbb{EBO}_i\left[S_i, \overline{z}_k, \overline{\mathbf{X}}\right]\Big) | S_i \in \mathbb{N}_0 \right\} \qquad k = 1,\ldots,\overline{y} \tag{18}$$

Note that, the subproblems in Eq.(18) are one-dimensional optimization problems, and we will exploit their structures in the following subsection. The improvement in objective value as a function of the number of servers in the cluster is calculated as:

$$\Gamma_k\left(z_k\right) = \mathcal{CIO}_k\left(\overline{\mathbf{X}}, z_k + 1\right) - \mathcal{CIO}_k\left(\overline{\mathbf{X}}, z_k\right)$$

Algorithm 1 corresponds to Capacity and Inventory Optimization modules and their interactions with each other as it is depicted in Figure 4.

Algorithm 1: Incremental Greedy Heuristic for Cluster Capacity Decision

```
**Require:** $\bar{\mathbf{X}}, \bar{y}$
**Ensure:** $z_k \geq \left\lceil \sum_{i \in \mathcal{X}_k} \overline{x}_{ik} \frac{\lambda_i}{\mu_i} \right\rceil$
1: for $k \in \{1, \ldots, \bar{y}\}$ do
2:    Calculate: $\Gamma_k(z_k)$
3:    while $\Gamma_k(z_k) < 0$ do
4:       Optimize: $\mathcal{CIO}_k\left(\overline{\mathbf{X}}, \overline{z}_k\right)$
5:       $z_k := +1$
6:    end while
7: end for
```

4.2.1. Optimization of basestock inventory levels

In general, a repair process may consist of several different stages (e.g., analysis, repair, testing). Therefore, a service time probability distribution with a lower service time variability, like Erlang-k distribution, is an appropriate choice for modeling the repair times in the cluster. Also, the use of any service time distribution with lower squared coefficient of variation (e.g., an Erlang-k type distribution) will lead to a lower number of items in repair and, consequently, to a lower inventory levels necessary to keep the same service levels. However, use of such distributions will increase the computational complexity of the analysis. Thus, based on the first two assumptions listed the in Subsection 3.2, we model each cluster k in the repair shop for the given number of servers as a multi-class multi-server $M/M/\overline{z}_k$ queuing system.

The probability distribution of the number of failed parts of type i at the steady-state

$$\lim_{t \to \infty} \mathbb{P}_i\left[Q_i(t) = q, \overline{\mathbf{X}}, \overline{\mathbf{Z}}\right] = p_i(q), \; i = 1, \ldots, N \; q = 0, 1, \ldots$$

is required to evaluate $\mathbb{EBO}_i\left[S_i, \overline{\mathbf{Z}}, \overline{\mathbf{X}}\right]$. We use the approach proposed by van Harten and Sleptchenko (2003) to calculate the marginal probability distribution of the number of failed parts os type i, $p_i(q)$ at the steady-state. The mentioned method utilizes eigenmode decomposition to find the exact solutions. However, the computational burdens arise when the number of SKUs and number of servers increases in the cluster. To overcome this issue, we use queuing approximation discussed in Altiok (1985) and Van Der Heijden et al. (2004). In this approximation, marginal probability distribution (and several performance characteristics) of the SKU i in the cluster k is derived by aggregating all other types (SKUs) in the cluster k into a single class. The procedure is repeated to obtain the remaining distributions for all SKUs in the cluster. Basically, we solve a multi-class multi-server queuing system including two-class (two SKUs) for each of the SKUs in the cluster rather than solving one multi-class multi-server with a larger number (total number of SKUs in the cluster) of SKUs.

These routines are depicted as Multi-Class Multi-Server Solver and Approximation Algorithm calls in Figure 4. Now, we can approximate the value of $\mathbb{EBO}_i\left[S_i,\overline{\mathbf{Z}},\overline{\mathbf{X}}\right]$ as:

$$\begin{aligned}\mathbb{EBO}_i\left[S_i,\overline{\mathbf{Z}},\overline{\mathbf{X}}\right] &\approxeq \sum_{q=0}^{\infty} \max\left\{0, q-S_i\right\}\tilde{p}_i\left(q\right)\\ &\approxeq \sum_{q=S_i+1}^{\infty}\left(q-S_i\right)\tilde{p}_i\left(q\right)\end{aligned}$$

where $\tilde{p}_i\left(q\right)$ is an approximation of the marginal probability distribution of the number of failed parts of type i, i.e., $p_i\left(q\right)$.
The objective function (18) is the summation of holding and backorder costs $\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right] = h_iS_i + b\mathbb{EBO}_i\left[S_i,\overline{\mathbf{Z}},\overline{\mathbf{X}}\right]$, and optimized by using the results of Proposition 2. The call of Inventory Optimizer routine in Figure 4 also uses the results of this proposition.

Proposition 2. $\mathbb{EHB}_i\left[S_i,\mathbf{X},\mathbf{Z}\right]$ *is convex on its whole domain* $S_i \in \mathbb{N}_0\ (:= \mathbb{N}\cup\{0\})$ *for given pooling policy* $\overline{\mathbf{X}}$ *and capacity levels* $\overline{\mathbf{Z}}$*, and it is minimized at the smallest* $S_i \in \mathbb{N}_0$ *for which*

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

is satisfied.

Proof. The first $\Delta\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]$ and the second $\Delta^2\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]$ order difference functions for $\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]$ are equal to Eq.(19) and Eq.(20), respectively.

$$\begin{aligned}\Delta\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right] =&\mathbb{EHB}_i\left[S_i+1,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]-\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]\\ =&h_i\left(S_i+1\right)+b\sum_{q=S_i+2}^{\infty}\left(q-\left(S_i+1\right)\right)\tilde{p}_i\left(q\right)-h_iS_i-b\sum_{q=S_i+1}^{\infty}\left(q-S_i\right)\tilde{p}_i\left(q\right)\\ =&h_i-b\sum_{q=S_i+1}^{\infty}\tilde{p}_i\left(q\right)\\ =&h_i-b\left(1-\sum_{q=0}^{S_i}\tilde{p}_i\left(q\right)\right) \qquad i=1,\ldots,N \end{aligned} \tag{19}$$

$$\begin{aligned}\Delta^2\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right] =&\Delta\mathbb{EHB}_i\left[S_i+1,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]-\Delta\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]\\ =&h_i-b\left(1-\sum_{q=0}^{S_i+1}\tilde{p}_i\left(q\right)\right)-h_i+b\left(1-\sum_{q=0}^{S_i}\tilde{p}_i\left(q\right)\right)\\ =&b\left(\sum_{q=0}^{S_i+1}\tilde{p}_i\left(q\right)-\sum_{q=0}^{S_i}\tilde{p}_i\left(q\right)\right)\\ =&b\tilde{p}_i\left(S_i+1\right) \qquad i=1,\ldots,N\end{aligned} \tag{20}$$

Eq.(20) shows that $\Delta^2\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]\geq 0$ $(\forall b\geq 0)$. Thus, $\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]$ is convex on its whole domain $S_i\in\mathbb{N}_0$, and minimized at the first point where $\Delta\mathbb{EHB}_i\left[S_i,\overline{\mathbf{X}},\overline{\mathbf{Z}}\right]\geq 0$ is satisfied. That implies,

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

$\square$

The sum $\sum_{q=0}^{S_i}\tilde{p}_i\left(q\right)$ denotes the fraction of time (probability) that there is no backordering for SKU i.

Remark. *Note that $S_i=0$ for the cases where $h_i\geq b$, and $S_i\to\infty$ if $b\gg h_i$ and/or $h_i\downarrow 0$.*

5. Computational Study

In this section, we present numerical analysis of the proposed solution algorithm. First, in Subsection 5.1, we define the experiment testbed used in our analysis. Next, in Subsection

5.2, we analyze cost savings that can be achieved by the optimized pooling schemes, and also document how the total cost is distributed among different components under different similarity measures. In Subsection 5.3, we analyze and compare how SKUs are clustered together and how capacity (number of servers) levels are affected by the input factors. Finally, in Subsection 5.4, we present the run times of the proposed optimization algorithm under different scenarios.

5.1. Problem Instance Generation

To generate the testbed, we use the full factorial design of experiment (DoE) with 7 factors and 2 levels per factor (128 test cases in total). 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 ρ with the assumption that all SKUs are processed on all servers, i.e., a repair shop design with one cluster and fully flexible servers. This system utilization rate is the third design factor with levels $\rho = 0.65$ and $\rho = 0.8$. For the chosen utilization rate, we randomly generate two sets of parameters:(1) the failure rates λ_i, such that $\sum_{i=1}^{N} \lambda_i = 1$, and

(2) workload percentages δ_i, such that $\sum_{i=1}^{N} \delta_i = 1$.

Using the generated λ_i and δ_i, we generate the service rates μ_i as $\mu_i = \frac{\lambda_i}{\delta_i \rho M}$. The denominator $\delta_i \rho M$ here 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):

(IND) completely randomly (independent) within a range $[h_{min}, h_{max}]$.

(HPB) hyperbolically related to the workloads $w_i = \lambda_i/\mu_i = \delta_i \rho M$:

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

and

$$\xi_i \in U[-\frac{h_{max} - h_{min}}{20}, \frac{h_{max} - h_{min}}{20}].$$

The minimum holding cost (h_{min}) is the fifth factor with levels 1 and 100. The maximum holding cost is fixed at 1000. 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 set as 1% or 10% of the chosen server cost for all SKUs. The penalty cost b is set to 50× the average holding cost: $b = 50\frac{\sum_{i=1}^{N}\lambda_i h_i}{\sum_{i=1}^{N}\lambda_i}$. That is, the penalty cost is such that about 98% of requests are served from stock (Probability of Backorder $\sim$ 0.02). The complete overview of the factors and levels can be found in Table 2.

Table 2: Problem parameter variants for test bed.

Factors	Levels	
No. of SKUs (N)	[10,	20]
No. of Initial servers (M)	[5,	10]
Utilization Rate (ρ)	[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. Cost savings and cost performance comparisons

Using the proposed clustering-based heuristic, we found the optimal pooling scheme together with optimal capacity and inventory levels of spares for the cases described above. Under each similarity measure, we compared 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 reductions for different problem factors under each similarity measure. On an average, the proposed pooled designs can produce from $\sim$34% to $\sim$36% and from $\sim$11% to $\sim$13% savings in comparison with dedicated and fully flexible designs, respectively. In some extreme settings, average cost reductions reach to 46% and 25% compared with dedicated and fully flexible systems, respectively.

The analysis shows that when the cost of having an extra skill is relatively small compared to that for having an extra server (i.e., the case of cross-training cost being equivalent $0.01f$), fully flexible design is as good as a pooled system. On the contrary, dedicated and fully flexible systems perform similarly where there is a high cost of cross-training.

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

		μ rule		$h \times \mu$ rule		h rule	
Factor	Levels	Dedicated	Fully Flexible	Dedicated	Fully Flexible	Dedicated	Fully Flexible
Number of SKUs (N)	10	25.21%	10.88%	23.88%	9.39%	25.09%	11.01%
	20	44.96%	13.56%	43.96%	12.53%	46.53%	15.93%
Number of Initial Servers (M)	5	44.03%	9.49%	42.99%	8.27%	44.84%	11.18%
	10	26.13%	14.94%	24.84%	13.65%	26.78%	15.76%
Utilization Rate (ρ)	0.65	37.93%	10.90%	36.86%	9.69%	39.12%	12.83%
	0.80	32.24%	13.54%	30.98%	12.23%	32.50%	14.11%
Minimum Holding Cost (h_{min})	1	36.51%	12.96%	34.69%	11.04%	36.70%	13.59%
	100	33.66%	11.48%	33.15%	10.87%	34.92%	13.35%
Holding Cost/Work Load Relation	IND	34.26 %	11.62%	34.13%	11.47%	35.19%	13.00%
	HPB	35.90%	12.82%	33.70%	10.45%	36.43%	13.95%
Server Cost (f)	$10h_{max}$	31.14%	12.48%	29.95%	11.23%	30.64%	12.19%
	$100h_{max}$	39.02%	11.96%	37.89%	10.68%	40.98%	14.75%
Cross-Training Cost (c_i)	$0.01f$	45.17%	1.08%	44.77%	0.55%	45.43%	1.64%
	$0.1f$	24.99%	23.36%	23.07%	21.37%	26.19%	25.30%
Average		**35.08%**	**12.22%**	**33.92%**	**10.96%**	**35.81%**	**13.47%**

Table 4 presents a performance comparison of μ, $h \times \mu$ and h rules based on the number of cases in which minimum total cost achieved.

In 69% (88 cases out of 128) of the cases, h rule obtains the lowest total cost among three similarity measures. We observe that when the same best objective function value is achieved by all of the rules (in 35 cases out of 128), all SKUs are pooled together as a single cluster (i.e. fully flexible). When the cost of cross-training increases (i.e., the case of cross-training cost being equivalent $0.1f$), a drastic decrease occurs in the number of cases minimum total cost achieved by $h \times \mu$ rule. On the other hand, the highest number of minimum cost achievement for all clustering rules is observed when the cost of cross-training is equal to 1% ($0.01f$) of the server cost. This occurs due to the fact that under a relatively low cross-training cost, all similarity measures tend to pool all SKUs together as a single cluster. Finally, Table 4 together with Table 3 show that on an average, h rule-based K-median clustering slightly surpasses the μ based clustering and $h \times \mu$ rule is dominated by other two clustering measures.

Next, we analyze and compare how the total system cost is distributed between different terms and how this distribution is affected by the input factors under different similarity measures (see Table 5). We observe that the distributions of cost term are almost identical under both similarity measure, and server and cross-training costs constitute the large portion of the total system cost.

Table 4: Cost performance comparison of two clustering rules under different scenarios.

Factor	Levels	# of cases μ rule achieves min.	# of cases $h \times \mu$ rule achieves min.	# of cases h rule achieves min.
Number of SKUs (N)	10	31	25	38
	20	33	25	50
Number of Initial Servers (M)	5	36	28	49
	10	28	22	39
Utilization Rate (ρ)	0.65	31	26	45
	0.80	33	24	43
Minimum Holding Cost (h_{min})	1	37	25	42
	100	27	25	46
Holding Cost/Work Load Relation	IND	32	31	41
	HPB	32	19	47
Server Cost (f)	$10h_{max}$	39	32	39
	$100h_{max}$	25	18	49
Cross-Training Cost (c_i)	$0.01f$	43	41	53
	$0.1f$	21	9	35
Overall		**64**	**50**	**88**

Table 5: Cost term distributions under different scenarios.

		μ rule				$h \times \mu$ rule				h rule			
Factor	Levels	Holding	Backorder	Server	Training	Holding	Backorder	Server	Training	Holding	Backorder	Server	Training
Number of SKUs (N)	10	10.53	3.49	73.05	12.91	10.46	3.71	72.14	13.66	11.77	4.00	72.40	11.81
	20	9.52	3.69	66.28	20.49	9.69	3.68	67.27	19.33	11.43	4.78	64.01	19.75
Number of Initial Servers (M)	5	10.94	4.17	66.98	17.90	10.59	4.07	67.43	17.89	12.09	4.77	65.48	17.65
	10	9.12	3.01	72.35	15.50	9.57	3.32	71.99	15.11	11.11	4.02	70.94	13.91
Utilization Rate (ρ)	0.65	9.99	3.64	69.97	16.38	10.38	3.89	68.55	17.15	11.72	4.41	67.75	16.10
	0.80	10.06	3.54	69.36	17.02	9.77	3.50	70.86	15.85	11.47	4.37	68.67	15.46
Minimum Holding Cost (h_{min})	1	9.34	3.40	70.25	17.00	9.86	3.65	69.4	17.06	10.63	4.15	69.20	16.00
	100	10.72	3.78	69.08	16.41	10.30	3.74	70.01	15.94	12.57	4.64	67.22	15.56
Holding Cost/Work Load Relation	IND	11.04	3.82	69.15	15.97	11.21	3.92	68.77	16.08	13.28	4.80	66.04	15.86
	HPB	9.02	3.35	70.18	17.43	8.95	3.47	70.65	16.91	9.92	3.98	70.37	15.70
Server Cost (f)	$10h_{max}$	14.36	5.27	65.11	15.25	14.67	5.45	65.10	14.75	15.97	6.28	62.96	14.77
	$100h_{max}$	5.69	1.91	74.23	18.15	5.49	1.94	74.32	18.24	7.23	2.51	73.46	16.79
Cross-Training Cost (c_i)	$0.01f$	10.71	4.32	74.91	10.04	10.78	4.54	74.20	10.47	11.38	4.52	75.06	9.01
	$0.1f$	9.34	2.85	64.42	23.36	9.38	2.85	65.22	22.53	11.81	4.27	61.35	22.55
Average		**10.03**	**3.59**	**69.67**	**16.70**	**10.08**	**3.69**	**69.71**	**16.50**	**11.60**	**4.39**	**68.21**	**15.78**

5.3. Capacity and pooling analyses

Table 6 shows the average number of used servers, clusters generated and SKUs assigned to each cluster under different factors and clustering rules. The number of initial servers M factor sets a lower bound on the minimum required servers to achieve a non-overloaded system (see Subsection 5.1). Thus, it is not interesting to observe a nearly doubled increase in the average number of used servers when the number of initial servers increased from 5 to 10.

We observe that except for the cross-training cost factor, the average number of clusters varies between 3 to 4 and the average number of SKUs per cluster varies between 3 to 5 all clustering rules. The high fluctuation in the number of clusters formed is observed under varying cross-training cost factor levels. This is consistent with the discussions presented in Subsection 5.2, i.e., when the cost of having an extra skill is relatively small compared to have an extra server (i.e., the case of cross-training cost being equivalent $0.01f$), fully flexible design (forming just one cluster) is as good as a pooled system.

Table 6: Average # of server, # of cluster and # of SKUs per cluster under each scenario.

		μ rule			$h \times \mu$ rule			h rule		
Factor	Levels	# server	# cluster	# skill per cluster	# server	# cluster	# skill per cluster	# server	# cluster	# skill per cluster
Number of SKUs (N)	10	8.00	3.29	3.03	8.04	3.06	3.26	7.92	3.32	3.00
	20	8.73	3.85	5.18	9.04	4.18	4.77	8.12	3.15	6.33
Number of Initial Servers (M)	5	5.98	2.90	5.16	6.18	3.00	5.00	5.68	2.56	5.85
	10	10.75	4.25	3.52	10.90	4.25	3.52	10.35	3.92	3.82
Utilization Rate (ρ)	0.65	7.81	3.57	4.19	7.84	3.43	4.36	7.34	3.00	5.00
	0.80	8.92	3.57	4.19	9.25	3.81	3.93	8.70	3.48	4.30
Minimum Holding Cost (h_{min})	1	8.12	3.42	4.38	8.28	3.43	4.36	7.96	3.21	4.66
	100	8.60	3.73	4.01	8.81	3.81	3.93	8.07	3.26	4.59
Holding Cost/Work Load Relation	IND	8.56	3.67	4.08	8.59	3.64	4.12	8.04	3.18	4.70
	HPB	8.17	3.48	4.30	8.50	3.60	4.15	8.00	3.29	4.54
Server Cost (f)	$10h_{max}$	8.92	3.60	4.15	9.12	3.73	4.01	8.67	3.18	4.70
	$100h_{max}$	7.81	3.54	4.22	7.96	3.51	4.26	7.37	3.29	4.54
Cross-Training Cost (c_i)	$0.01f$	7.32	1.43	10.43	7.29	1.18	12.63	7.26	1.56	9.60
	$0.1f$	9.40	5.71	2.62	9.79	6.06	2.47	8.78	4.92	3.04
Average		**8.36**	**3.57**	**4.19**	**8.54**	**3.62**	**4.13**	**8.02**	**3.24**	**4.62**

Next, we analyze the cost improvements achieved by the increasing number of servers used in each cluster (see Algorithm 1). Figure 5 presents the gap between the utilized number of servers at optimal design z_k and the minimum number of servers required $\left\lceil \sum_{i=1}^{N} x_{ik} \frac{\lambda_i}{\mu_i} \right\rceil$ for a particular cluster k. At the optimal pooled designs, it is interesting to note that over 80% of the formed clusters uses an only minimum number of required servers that is enough to ensure the system stability (not overloading).

When the ratio of cross-training cost/server cost is not too small ($\nleq 0.01$), practitioner may come up with decent pooled designs by easily implementing rule of thumb: (i) generating clusters such that each cluster includes 20%-30% of the total SKUs (result of Table 6) that are close in service rates and (ii) assigning minimum number of required servers to each cluster (result of Figure 5).

5.4. Runtime performance

All the experiments are implemented on a computer with 16 GB RAM and 2.8 GHz i7 CPU. The presented solution algorithm under both similarity measures converges quite fast

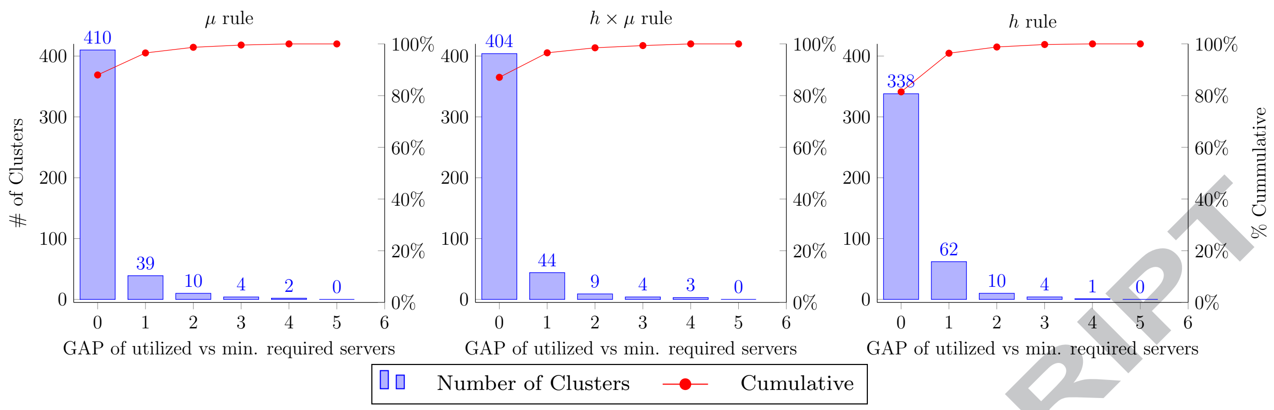

Figure 5: Difference between used and minimum required servers per cluster.

in most of the cases and provides us the final solution within 400 seconds as shown in Figure 6.

The runtime distributions of the implemented methodology for all similarity measures and for all different factors are also provided and compared in Figure 6. The analysis shows that the only factors affecting run time of the proposed algorithm are the number of SKUs, the number of initial servers and utilization rate. Other DoE factors do not affect the run times significantly. The increasing number of SKUs results in higher run times due to the increased number of required calls needed for the K-median subroutine and increased size of the clustering problem. Similarly, the number of initial servers indirectly affects the size of the multi-class multi-server queuing problem that has to be solved several times.

Apart from our initial testbed, we generated 40 different test instances with varying sizes from one SKU to 40 SKUs to investigate the effect of problem size on running time. Figure 7 presents an exponential trend in running time due to the $\mathcal{NP}$-hard complexity of K-median problem. However, even for the larger instances, run time performance is acceptable for tactical/operational levels decisions in real-life spare part supply systems.

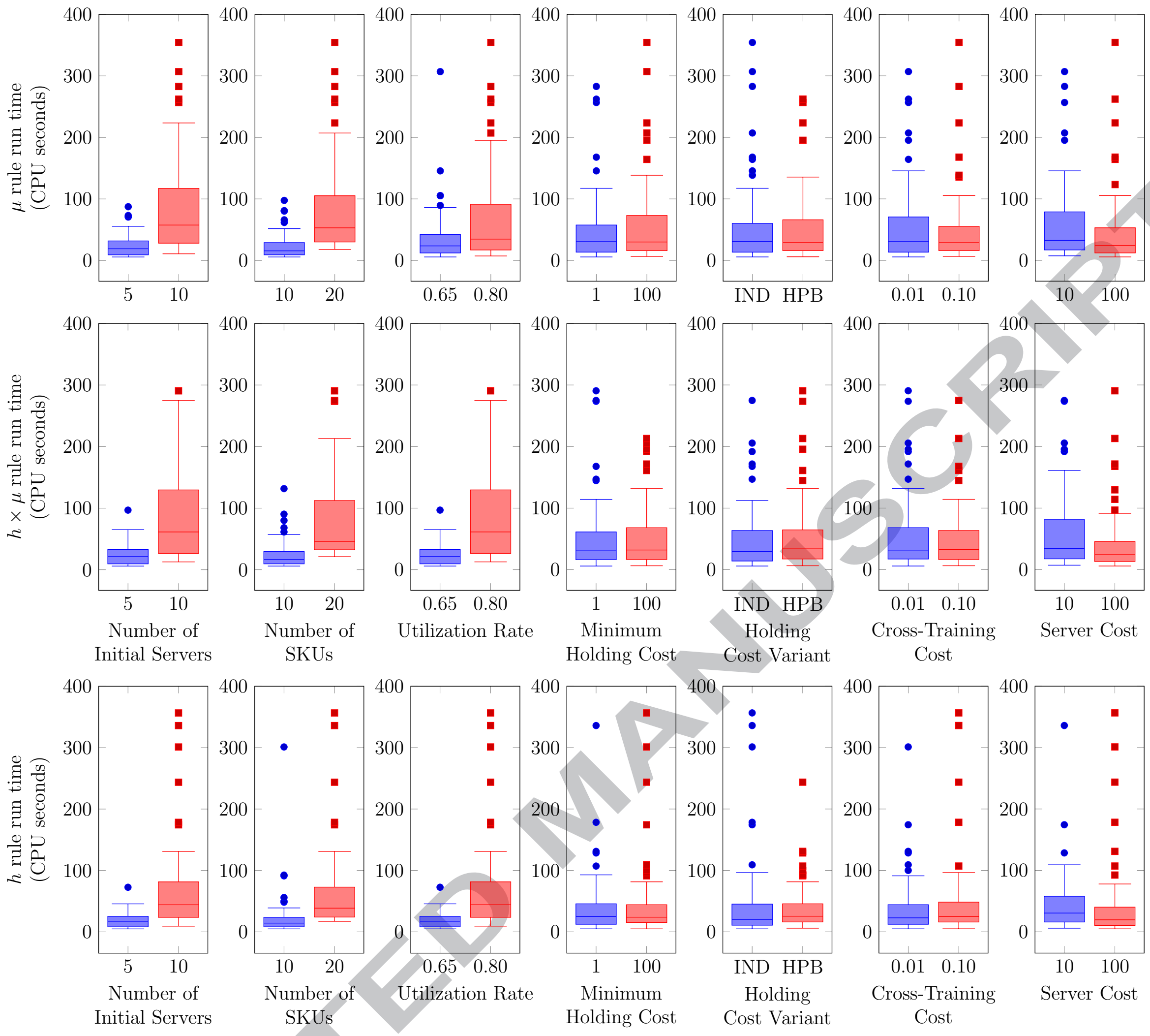

Figure 6: Runtime performance under various scenarios.

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. Different strategical actions will lead to different tactical operating consequences in terms of cost and responsiveness. The inextricably linked structure of strategic and tactical decisions forces the decision maker to take into account all plausible actions simultaneously. In this direction, we analyzed the joint problem of resource pooling, inventory allocation and capacity level designation of the repair shop, and addressed the following issues: (i) how many clusters to pool; (ii) which spare part types to pool together; and (iii) how to assign servers (capacity) to each cluster by considering trade-off between pooling and spare inventories.

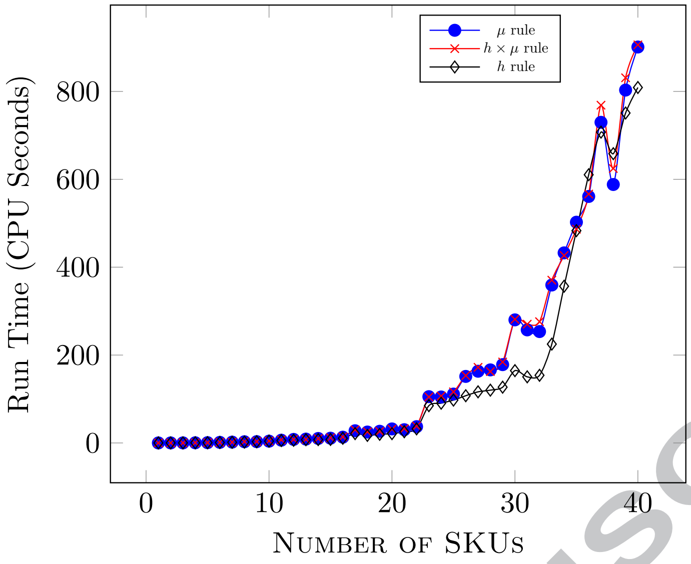

Figure 7: Effect of problem size on runtime.

Our solution approach divided the problem into three stages: first, the pooling problem was solved; the trade-off of capacity versus the allocation of spare inventories was considered in the second and the third stages simultaneously. To find the optimal pooling schemes, we proposed a K-median clustering-based sequential algorithm and performed extensive numerical studies, using different sets of generated cases. The main results and conclusions include:

1. The pooled designs can result in average cost savings up to ∼36% and ∼13% in comparison to dedicated and fully flexible designs, respectively.

2. The advantages of clustered repair shop designs diminish as decreasing ratio of cross-training cost/server cost.

3. Chosen similarity measures do not significantly outperform each other, when both total system cost and running times are considered.

4. A simple heuristic rule leading to effective clustered repair shops designs is proposed as below:

   "When the ratio of cross-training cost/server cost is not too small ($\nleq 0.01$)*, (i) generate clusters such that each cluster includes 20%-30% of the total SKUs that are close in service rates and (ii) assign a minimum number of required servers to each cluster."*

Like most of the combinatorial optimization problems, solving $\mathcal{MP}$ becomes more difficult in terms of running time when the number of SKUs increases. Therefore, further research can be on the design of clustering heuristics or meta-heuristics that can generate better pooling schemes with less computational complexity compared to the proposed K-median algorithm. Another research direction would be to investigate the use of more realistic service time distributions (e.g. Erlang-k) in the proposed optimization model. Finally, incorporating effects of learning by experience during the assignment of skills to servers as discussed by Valeva et al. (2017b,a) and Corominas et al. (2010) would be interesting extension for the modeling of cross-training.

References

Aksin, Z., Armony, M., & Mehrotra, V. (2007). The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management*, *16*, 665–688.

Altiok, T. (1985). On the phase-type approximations of general distributions. *IIE Transactions*, *17*, 110–116.

Bassamboo, A., Randhawa, R. S., & Mieghem, J. A. V. (2012). A little flexibility is all you need: on the asymptotic value of flexible capacity in parallel queuing systems. *Operations Research*, *60*, 1423–1435.

Basten, R., & Van Houtum, G. (2014). System-oriented inventory models for spare parts. *Surveys in Operations Research and Management Science*, *19*, 34–55.

Chemweno, P. K., Pintelon, L., & Muchiri, P. N. (2015). Evaluating the impact of spare parts pooling strategy on the maintenance of unreliable repairable systems. *IFAC-PapersOnLine*, *48*, 989–994.

Colen, P., & Lambrecht, M. (2012). Cross-training policies in field services. *International Journal of Production Economics*, *138*, 76–88.

Corominas, A., Olivella, J., & Pastor, R. (2010). A model for the assignment of a set of tasks when work performance depends on experience of all tasks involved. *International Journal of Production Economics*, *126*, 335–340.

Diaz, A., & Fu, M. C. (1997). Models for multi-echelon repairable item inventory systems with limited repair capacity. *European Journal of Operational Research*, *97*, 480–492.

Fang, L., & Zhaodong, H. (2015). System dynamics based simulation approach on corrective maintenance cost of aviation equipments. *Procedia Engineering*, *99*, 150–155.

Harper, P. R., Powell, N., & Williams, J. E. (2010). Modelling the size and skill-mix of hospital nursing teams. *Journal of the Operational Research Society*, *61*, 768–779.

van Harten, A., & Sleptchenko, A. (2003). On markovian multi-class, multi-server queueing. *Queueing systems*, *43*, 307–328.

Van der Heijden, M., Alvarez, E., & Schutten, J. (2013). Inventory reduction in spare part networks by selective throughput time reduction. *International Journal of Production Economics*, *143*, 509–517.

Inman, R. R., Jordan, W. C., & Blumenfeld, D. E. (2004). Chained cross-training of assembly line workers. *International Journal of Production Research*, *42*, 1899–1910.

Iravani, S. M., & Krishnamurthy, V. (2007). Workforce agility in repair and maintenance environments. *Manufacturing & Service Operations Management*, *9*, 168–184.

van Jaarsveld, W., Dollevoet, T., & Dekker, R. (2015). Improving spare parts inventory control at a repair shop. *Omega*, *57*, 217–229.

Jordan, W. C., & Graves, S. C. (1995). Principles on the benefits of manufacturing process flexibility. *Management Science*, *41*, 577–594.

Jordan, W. C., Inman, R. R., & Blumenfeld, D. E. (2004). Chained cross-training of workers for robust performance. *IIE Transactions*, *36*, 953–967.

Karsten, F., & Basten, R. J. (2014). Pooling of spare parts between multiple users: How to share the benefits? *European Journal of Operational Research*, *233*, 94–104.

Legros, B., Jouini, O., & Dallery, Y. (2015). A flexible architecture for call centers with skill-based routing. *International Journal of Production Economics*, *159*, 192–207.

Li, Q., Gong, J., Fung, R. Y., & Tang, J. (2012). Multi-objective optimal cross-training configuration models for an assembly cell using non-dominated sorting genetic algorithm-II. *International Journal of Computer Integrated Manufacturing*, *25*, 981–995.

Liu, C., Yang, N., Li, W., Lian, J., Evans, S., & Yin, Y. (2013). Training and assignment of multi-skilled workers for implementing seru production systems. *The International Journal of Advanced Manufacturing Technology*, *69*, 937–959.

Muckstadt, J. A. (2004). *Analysis and algorithms for service parts supply chains*. Springer Science & Business Media.

Qin, R., Nembhard, D. A., & Barnes II, W. L. (2015). Workforce flexibility in operations management. *Surveys in Operations Research and Management Science*, *20*, 19–33.

Rappold, J. A., & Van Roo, B. D. (2009). Designing multi-echelon service parts networks with finite repair capacity. *European Journal of Operational Research*, *199*, 781–792.

Sahba, P., Balcıog, B. et al. (2011). The impact of transportation delays on repairshop capacity pooling and spare part inventories. *European Journal of Operational Research*, *214*, 674–682.

Sayın, S., & Karabatı, S. (2007). Assigning cross-trained workers to departments: A two-stage optimization model to maximize utility and skill improvement. *European Journal of Operational Research*, *176*, 1643–1658.

Schneider, M., Grahl, J., Francas, D., & Vigo, D. (2013). A problem-adjusted genetic algorithm for flexibility design. *International Journal of Production Economics*, *141*, 56–65.

Sherbrooke, C. C. (1968). Metric: A multi-echelon technique for recoverable item control. *Operations Research*, *16*, 122–141.

Sherbrooke, C. C. (2006). *Optimal inventory modeling of systems: multi-echelon techniques* volume 72. Springer Science & Business Media.

Simmons, D. (2013). The effect of non-linear delay costs on workforce mix. *Journal of the Operational Research Society*, *64*, 1622–1629.

Sleptchenko, A., ElMekkawy, T. Y., Turan, H. H., & Pokharel, S. (2017a). 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).

Sleptchenko, A.,& Van der Heijden, M. (2016). Joint optimization of redundancy level and spare part inventories. *Reliability engineering & system safety*, *153*, 64–74.

Sleptchenko, A., Van der Heijden, M., & Van Harten, A. (2002). Effects of finite repair capacity in multi-echelon, multi-indenture service part supply systems. *International Journal of Production Economics*, *79*, 209–230.

Sleptchenko, A., Van der Heijden, M., & Van Harten, A. (2003). Trade-off between inventory and repair capacity in spare part networks. *Journal of the Operational Research Society*, *54*, 263–272.

Sleptchenko, A., van der Heijden, M. C., & van Harten, A. (2005). Using repair priorities to reduce stock investment in spare part networks. *European Journal of Operational Research*, *163*, 733–750.

Sleptchenko, A., Turan, H. H., Pokharel, S., & ElMekkawy, T. Y. (2017b). Cross-training policies for repair shops with spare part inventories. *International Journal of Production Economics*, https://doi.org/10.1016/j.ijpe.2017.12.018. Accepted for publication.

Slomp, J., Bokhorst, J. A., & Molleman, E. (2005). Cross-training in a cellular manufacturing environment. *Computers & Industrial Engineering*, *48*, 609–624.

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, *3*, 59–66.

Srivathsan, S., & Viswanathan, S. (2017). A queueing-based optimization model for planning inventory of repaired components in a service center. *Computers & Industrial Engineering*,*106*, 373–385.

Tekin, E., Hopp, W. J., & Van Oyen, M. P. (2009). Pooling strategies for call center agent cross-training. *IIE Transactions*, *41*, 546–561.

Tiemessen, H., & van Houtum, G.-J. (2013). Reducing costs of repairable inventory supply systems via dynamic scheduling. *International Journal of Production Economics*, *143*, 478–488.

Tsitsiklis, J. N., Xu, K. et al. (2012). On the power of (even a little) resource pooling. *Stochastic Systems*, *2*, 1–66.

Turan, H. H., Pokharel, S., Sleptchenko, A., & ElMekkawy, T. Y. (2016). Integrated optimization for stock levels and cross-training schemes with simulation-based genetic algorithm. In *2016 International Conference on Computational Science and Computational Intelligence* (pp. 1158–1163).

Valeva, S., Hewitt, M., & Thomas, B. W. (2017a). A matheuristic for workforce planning with employee learning and stochastic demand. *International Journal of Production Research*, (pp. 1–18).

Valeva, S., Hewitt, M., Thomas, B. W., & Brown, K. G. (2017b). Balancing flexibility and inventory in workforce planning with learning. *International Journal of Production Economics*, *183*, 194–207.

Van Der Heijden, M., Van Harten, A., & Sleptchenko, A. (2004). Approximations for markovian multi-class queues with preemptive priorities. *Operations Research Letters*, *32*, 273–282.

Van Horenbeek, A., Buré, J., Cattrysse, D., Pintelon, L., & Vansteenwegen, P. (2013). Joint maintenance and inventory optimization systems: A review. *International Journal of Production Economics*, *143*, 499–508.

Van Houtum, G.-J., & Kranenburg, B. (2015). *Spare parts inventory control under system availability constraints* volume 227. Springer.

Wallace, R. B., & Whitt, W. (2005). A staffing algorithm for call centers with skill-based routing. *Manufacturing & Service Operations Management*, *7*, 276–294.

Wang, L., Chu, J., & Mao, W. (2008). An optimum condition-based replacement and spare provisioning policy based on markov chains. *Journal of Quality in Maintenance Engineering*, *14*, 387–401.

Weisstein, E. W. (2006). Bell number. from mathworld—a wolfram web resource `http://mathworld.wolfram.com/BellNumber.html`.

Wong, H., Cattrysse, D., & Van Oudheusden, D. (2005). Inventory pooling of repairable spare parts with non-zero lateral transshipment time and delayed lateral transshipments. *European Journal of Operational Research*, *165*, 207–218.

- The design problem of a single repair shop in a spare part supply system is studied.
- The joint problem of pooling, inventory allocation and capacity level designation is solved.
- A novel sequential heuristic integrating queuing theory and K-median is developed.
- Approach provides ~11% and ~34% cost reductions compared to fully flexible and dedicated designs, respectively.
- The advantages of clustered design diminish with decreasing cross-training/server cost ratio.