23rd International Congress on Modelling and Simulation, Canberra, ACT, Australia, 1 to 6 December 2019
mssanz.org.au/modsim2019

Repair priorities in repairable spare part supply systems: a simulation-optimization approach

H. H. Turan[a], **S. Elsawah**[a] **and M. J. Ryan**[a]

[a]*Capability Systems Centre, University of New South Wales, Canberra, Australia*

Email: h.turan@adfa.edu.au

Abstract: Industries employing expensive assets maintain extensive repair facility operations and keep spare stocks. This type of logistics system, where both forward and reverse logistics systems are required, in addition to repair facilities, is known as a repairable system. We study a repairable spare part supply system consisting of one repair facility and one stock point, where repairables are kept on the stock to serve expensive capital assets in order to prevent downtime.

We set up the objective of our model to minimize the expected total inventory holding costs of spare parts and costs for the downtime of assets over an infinite horizon. In this study, we particularly analyze the effect of static repair priorities on the expected total cost. To achieve this, we seek optimal values of the repairable spare parts stocks and the assignment of different repairable types into priority classes.

We model the repair facility as a multi-server multi-class queue, where failed repairable parts are repaired based on priority classes. It is generally difficult to analyze this type of queuing systems with analytical methods even for a small size problem with the limited number of priority classes and repairable types. Therefore to alleviate this difficulty, we develop a two-stage sequential simulation-optimization algorithm. In the first stage, the set of all feasible priority assignments is searched by a Genetic Algorithm (GA) meta-heuristic to find an assignment that achieves the minimum cost. In the second stage, a discrete event simulation (DES) is run for the given priority assignment provided by the GA to analyze the multi-class multi-server queueing model. The probability distribution for the number of failed spare parts in the repair facility is obtained as an output of the DES. We use probability distributions to calculate the optimal level of repairable spare part stocks to keep in the inventory.

We compare the performance of the simulation-optimization algorithm with a First-Come First-Served (FCFS) service discipline since FCFS reflects the common way of working in practice. The conducted computational experiments show that the proposed approach yields a significant amount of total cost reduction in some extreme cases reaching up to 90%.

Keywords: *Repairable parts, repair priority, simulation-optimization, genetic algorithm*

1 INTRODUCTION

A repairable spare part supply network usually consists of a repair facility, resources (e.g., workforce, spare part stocks, and service tools) needed to perform the repair operations and a stock location to store spare parts. Among these resources, repairable parts constitute the big part of the capital invested in spare parts inventories (between 70%-85%) and about 60%-70% of all repair operations for spare parts are performed by internal repair shops (Kosanoglu et al., 2018). Further, the unavailability of spare parts is the reason for 80% of all system downtime (Kosanoglu et al., 2018). For these reasons, we model and analyze a repairable part supply system that contains a single internal repair shop integrated with repairable parts inventory system as depicted in Figure 1.

The essential decisions regarding a repairable parts supply system includes: (a) the inventory policy (e.g., inventory levels and replenishment strategies), (b) the repair capacities (e.g., workforce level, amount of service machinery and tools) available to repair failed parts and (c) the scheduling method (e.g., repair priorities) to control the flow of work in the repair shop. An execution of optimal strategies regarding these decisions enables faster repairs and leads to a reduction in the downtime of capital assets (Turan et al., 2018; Sleptchenko et al., 2019). These decisions are interrelated; for example, the required amount of repairable spare parts kept on the stock depends on the repair capacity (i.e., the rate of repair), and isolated optimization of a single decision would result in suboptimal solutions. In this study, we present an optimization model and a solution algorithm that considers decisions on both the repairable spare inventory levels and the repair priorities.

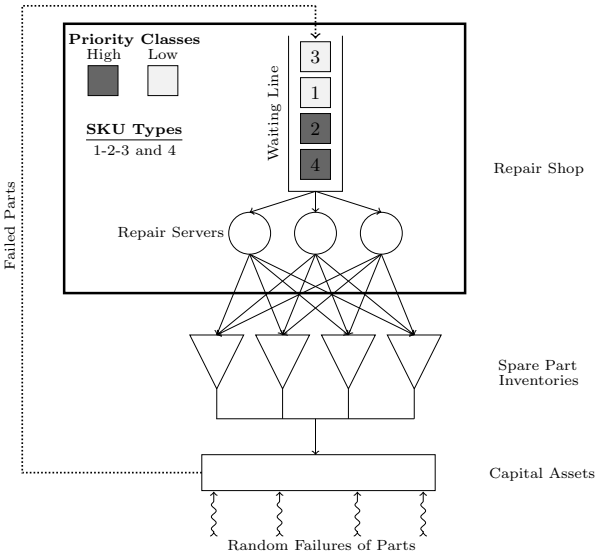

Figure 1. A repairable spare part supply network with re-pair priorities.

Researchers have developed many models regarding repairable parts inventory systems since the late 60s. Multi-Echelon Technique for Recoverable Item Control model, also known as the METRIC system (Sherbrooke, 1968), is the first practical model developed. However, the effect of repair priorities in spare part supply has been studied in a few papers only, which seems to be due to the fact that priority systems are hard to analyze in general (Adan et al., 2009). The works of Hausman and Scudder (1982) and Pyke (1990) are the earlier studies on repair priorities in systems with limited numbers of different repairable types via simulation. Both works show that dynamic priority rules outperform static rules, and utilizing priority rules may lead to significant cost savings, especially under high workloads at the repair shop. Nevertheless, these studies evaluate given priority rules with the help of simulation rather than seeking and optimizing priority assignment. Sleptchenko et al. (2005) study a two-echelon, two-indenture system with multi-server wit only two-priority classes. They develop heuristics to optimize spare parts stocks and the assignment of repair priorities. They show that static priorities may lead to significant cost reductions in comparison to FCFS. However, their computational study is limited to two priority classes, and only systems with a limited number of repairable types are analyzed. A model similar to our study is present by Adan et al. (2009), they assume failure and service rates of repairable parts obey exponential distributions and solve the model up to five priority classes with one repair server. Different from their work, any probability distribution can be used to model the arrival and service pattern of repairable parts in our approach. Further, we don't restrict the number of priority classes and the number of servers in the repair shop. The remainder of this paper is organized as follows. In Section 2, we provide a detailed description of the problem we solve. Section 3 presents our two-stage simulation-optimization algorithm, while Section 4 outlines the computational experiment design and presents obtained results. Conclusions and future research directions are discussed in Section 5.

2 PROBLEM DEFINITION AND MATHEMATICAL MODEL

We study a spare part supply system including repairable part inventories as shown in Figure 1. The modeled supply network contains a single location repair shop with fully cross-trained identical servers. That is, any

server can repair all type of failed parts. The multiple types of repairable parts; i.e., stock keeping units (SKUs), are kept as inventory to serve several high-valued capital assets.

Those assets include SKUs, and SKUs are subject to random failures. When a part fails, an order is immediately placed for a ready-for-use part of the same type at the stock point, and the failed part is sent to the repair shop (Turan et al., 2018; Sleptchenko et al., 2019).

The SKUs are grouped into different repair priority classes. Each SKU type is assigned to one of the priority class, and each priority class consists of one or more SKU type. The failed parts form a single queue such a way that parts belong to higher priority classes placed in front of the low priority classes, and parts in the same class are lined up based on their arrival times. Besides, the repair process of a low priority part is interrupted, if a high priority part arrives and finds all servers busy; if multiple low priority parts are in service, the part to be postponed is chosen randomly with equal probabilities. Meanwhile, the failed part is immediately replaced by spare inventory if there is the same type of available ready-for-use part on the stock. Otherwise, the demand is backordered and fulfilled as soon as a ready-for-use part of the demanded SKU type becomes available. In case of unavailability of the ready-for-use part, the capital asset goes down, and downtime cost starts occurring until the requested ready-for-use part is delivered (Turan et al., 2018; Sleptchenko et al., 2019).

The modeled repairable inventory system contains N different SKUs; i.e., repairable part types. The time to failure for all types of in-use parts is assumed to be exponentially distributed with a constant rate $\lambda_i \ \ (i = 1, \ldots, N)$, and independent of each other. Even though our approach is capable of handling any type of probability distribution, this assumption is consistent with the behavior of assets during their useful life phase in the bathtub curve (Iravani and Krishnamurthy, 2007). The repair rate for SKU type i is μ_i $(i = 1, \ldots, N)$, and independent of the server performing the repair. An exponentially distributed repair time indicates cases where shorter repair times are more probable than longer repair times.

An inventory holding cost h_i occurs for SKU type i per part per unit time $(i = 1, \ldots, N)$. We assume a backorder cost b occurs when the required part is not available and is paid per unit time per backordered part. The expected backorders are determined using steady-state probabilities on an infinite time planning horizon since the average lifetime of the high-valued assets such as military vessels is usually long (over 10 years).

The objective of the model is to minimize the average total inventory holding costs of repairable spare parts and downtime costs of assets occurring due to backorders over an infinite horizon. In the model, the optimization variables are the amount of spare parts stocks for each SKU type i, S_i $(\ i = 1, \ldots, N)$, and the priority assignment $\mathbf{P}$; i.e., the assignment of SKUs to priority classes. The objective function is given in Eq.(1) .

$$\min_{S_i,\, \mathbf{P}} \sum_{i=1}^{N} \left(h_i S_i + b \mathbb{EBO}_i \left[S_i, \mathbf{P} \right] \right) \tag{1}$$

where $\mathbb{EBO}_i \left[S_i, \mathbf{P} \right]$ denotes the expected number of backorders for SKU type i for the chosen values of S_i and $\mathbf{P}$. A feasible priority assignment $\mathbf{P}$ has to satisfy constraint sets in Eq.(2). The binary decision variable $x_{i,j}$ indicates whether SKU type i is assigned to the priority class j $(j = 1, \ldots, J)$, where J denotes the total number of priority classes in the repair shop. These constraint sets ensure that each SKU type is assigned to only one priority class and each priority class contains at least one SKU type. Further, the total number of priority class J cannot exceed the number of distinct SKU types N. The repair shop depicted in Figure 1 has two priority classes ($J = 2$) labeled as low and high, and each class contains two types of SKUs. When J is set as one, the FCFS queuing discipline is applied. The last constraint in Eq.(3) ensures that the amount of spare parts stocks kept on inventory are none negative integers.

$$\mathbf{P} \in \left\{ \begin{array}{ll} \displaystyle\sum_{j=1}^{J} x_{i,j} = 1, & i = 1, \ldots, N \\ \displaystyle\sum_{i=1}^{N} x_{i,j} \geq 1, & j = 1, \ldots, J \\ x_{i,j} \in \{0, 1\}, & i = 1, \ldots, N \; j = 1, \ldots, J \\ 2 \leq J \leq N & \end{array} \right\} \tag{2}$$

$$S_i \in \mathbb{N}_0 \qquad\qquad i = 1, \ldots, N \tag{3}$$

The total number of feasible priority assignments P for a repair shop containing N SKU is calculated by

Eq.(4), where J denotes the number of priority class.

$$\text{Total \# of feasible } \mathbf{P} = \sum_{J=2}^{N} \left(\sum_{k=0}^{J-1} (-1)^k \binom{J}{k} (J-k)^N \right) \tag{4}$$

Figure 2 shows how the problem size (i.e., the search space) increases for the increasing number of N and J. Due to the large size of the search space for priority assignments $\mathbf{P}$, it is not efficient (and often not possible) to find the optimal assignment with traditional optimization methods. Even for a small problem size, due to the intensive procedures used in obtaining steady-state probabilities, the evaluation of $\mathbb{EBO}_i[S_i, \mathbf{P}]$ becomes cumbersome. Therefore to alleviate these difficulties, we couple a meta-heuristic algorithm that systematically checks the candidate assignments with a discrete-event simulation that helps to evaluate $\mathbb{EBO}_i[S_i, \mathbf{P}]$.

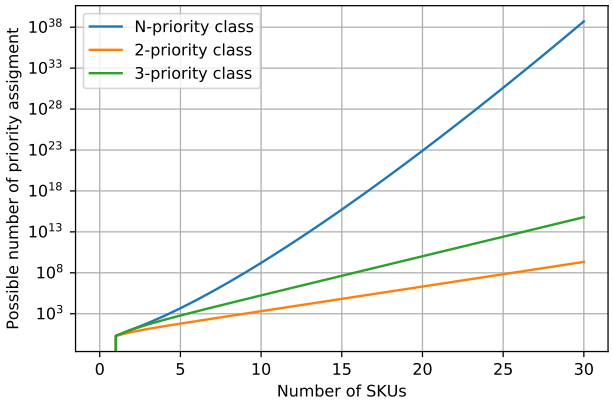

Figure 2. The total number of feasible priority assignments.

3 Solution Approach

In this section, we propose a two-stage solution approach based on a simulation-optimization technique, as described in Figure 3. At the first stage, the optimal priority assignment, $\mathbf{P}$, is searched by utilizing a genetic algorithm (GA). GA generates a set of feasible priority assignments. Afterwards these candidate feasible solutions are passed through a fitness evaluation function to find optimal inventory levels S_i for each SKU type i. In the second stage two things happen, (i) initially a discrete-event simulation (DES) is run with the priority assignment $\mathbf{P}$ produced at the first stage, and (ii) the output of DES (i.e., the probability distribution $P_i(q)$ for the number of failed parts waiting in queue for the SKU type i) are used to find the optimal value of S_i and total cost. We discuss implementation details of the GA and DES (inventory optimization) in the Subsections 3.1 and 3.2, respectively.

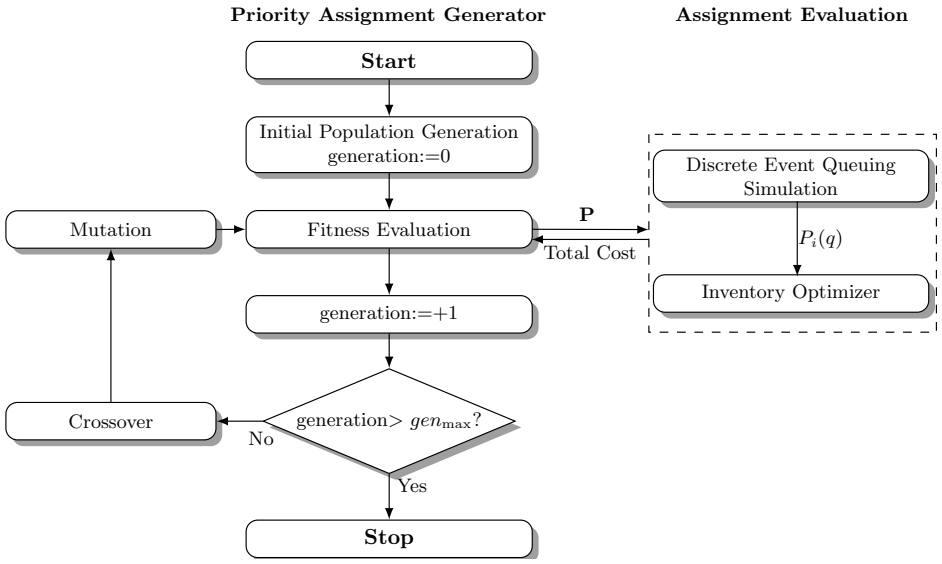

Figure 3. The flow of the proposed simulation-optimization technique

3.1 Priority assignment with a GA

GA is based on an analogy to the phenomenon of natural selection in biology (Goldberg and Holland, 1988). At the first step, a chromosome structure has to be defined (encoded) to represent the solutions of the problem. Afterwards an initial population of solutions is generated by using the chosen chromosome structure. Individuals in the population are selected, based on an evaluation function, called "fitness" that, associates a value to each individual according to its objective function such as total cost (Mahdavi et al., 2009). Usually, individuals with high fitness values are more likely to be selected in order to reproduce. That is, inferior individuals with low fitness values are replaced by more fit individuals. Genetic operators (e.g., crossover and mutation) are applied to the selected individuals to produce a new population at each generation. This procedure is repeated until a certain number of iteration (generations) is reached as shown in Figure 3.

The first stage of any GA implementation is to map solution characteristics in the format of a chromosome string. In this paper, we use a direct coding scheme as shown in Figure 4. That is, each chromosome corresponds to a particular priority assignment, $\mathbf{P}$. In this coding, each gene represents the priority class that the SKU is assigned to. For example, the chromosome in Figure 4 indicates that SKUs 1, 5 and 10 are assigned to priority class 2. The length of chromosome denotes the number of SKUs, N, in the repair shop. Further, each chromosome also carries information about the number of priority class, J, exists in the assignment $\mathbf{P}$. The total number of distinct integer in the chromosome represents the number of priority class. There are five priority classes in $\mathbf{P}$ shown in Figure 4.

In the second stage, a set of initial solutions, a population, are generated. The number of solutions to be included in the population is called population size. The initial population is generated only once at the beginning for the first generation of the GA. The initial population is generated by assigning a random integer to each gene from 1 to N with equal probability. However, this initialization does not ensure that whether the constraint $J \geq 2$ is satisfied. We check the feasibility of each individual after initial generation, crossover and mutation operations, and replace an infeasible solution with a feasible solution.

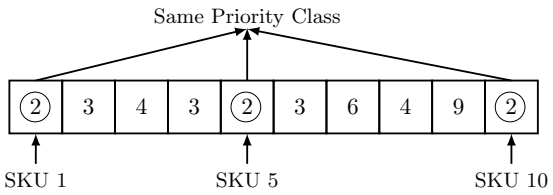

Figure 4. Chromosome (coding) structure

A fitness function is used to evaluate and reproduce new chromosomes, called offspring for the next generations. The fitness evaluation is used to measure the goodness of the candidate priority assignment $\mathbf{P}$ in the population with respect to the total cost. The fitness value of a chromosome is calculated by optimizing the inventory levels S_i that minimize the sum of holding and backorder cost in Eq.(1). The details of the fitness evaluation are provided in Subsection 3.2.

Next, some individuals from the population are selected to reproduce. The goal of selection strategy is to allow the "fittest" individuals to be considered more often to reproduce children for the next generation. We apply a tournament selection strategy. In the tournament selection, several tournaments are played among a few individuals. The individuals are chosen at random from the population. The winner of each tournament is selected for the next generation. Crossover and mutation operators are applied to the selected parents (individual solutions) to produce new offspring. We use a single point crossover operation. In this crossover scheme, two individuals, called parents, are selected from the population. Then a number between 1 and $N-1$ is chosen randomly with equal probability. Suppose this number is l. Then, the genes form $l+1$ to N of both parents are exchanged to build two new chromosomes (children).

After crossover, we apply a mutation operation to randomly selected individuals. Mutation operations are used to avoid trapping in local optima and to explore new priority assignments. We develop and try four different mutation operations. When a chromosome is chosen to mutate, one of these four operators is applied with equal probability. In one-gene mutation, a gene is randomly selected and its value is changed to another value between 1 and N. We select two random genes in chromosome and swap the values of these genes in the swap mutation. In the two-gene mutation, two genes are randomly selected and their values are changed randomly to other values between 1 and N. Lastly, when shuffle mutation is applied, we shuffle and change the place of genes in the individual chromosome randomly. The latter two mutation operators are intended to explore a larger neighborhood of the mutated individual solution $\mathbf{P}$ compared to the first two mutation operations so that solution diversity is maintained. To run our GA, we set population size as 100, and set the number of generations (gen_{max}) to 50. We chose crossover and mutation probability as 0.8 and 0.4, respectively.

Moreover, the tournament size for selection procedure is decided as 10.

3.2 Fitness evaluation and inventory optimization

After the generation of priority assignments, $\mathbf{P}$, by above described GA procedure, a DES model and inventory optimizer are called to evaluate the fitness of each assignment in the population. The DES produces probability distributions $P_i(q)$ for the numbers of SKUs type i $(i = 1, \ldots, N)$ in the queue and in the repair process at the steady-state. The obtained distributions are used to compute the optimal stock levels S_i, as defined by inequities in Eq.(5).

$$\sum_{q=0}^{S_i-1} P_i(q) \leq \frac{(b-h_i)}{b} \leq \sum_{q=0}^{S_i} P_i(q)\ i = 1, \ldots, N \tag{5}$$

We refer readers to (Van Houtum and Kranenburg, 2015) for more details on the optimization of single item spare parts inventory systems under base-stock inventory policy.

The run conditions for DES model set as 10 replications for a given priority assignment, and 100,000 failure arrivals with a 5,000 arrival warm-up period length. We ensure that each simulation replication for a given priority assignment is independent of each other. Furthermore, we use the same squeeze of random numbers (common random numbers) for different priority assignments, so unbiased comparisons between assignments are achieved.

4 Computational Study

In this section, we perform two sets of computational experiments. In the first set, we compare our methodology with a brute-force approach, where it is possible to enumerate all priority assignments. In the second set of experiments, we generate larger instances as factors of the number of SKUs, N, the utilization rate of the repair facility ρ and range of minimum holding cost h_{min}. All algorithms are coded in Python programming language. We implement and run all problem experiments on a standard desktop computer with 12 cores 2.70 GHz CPU.

Table 1a and Table 1b show how the simulation-optimization approach performs compared to the brute-force and FCFS policy up to seven SKUs and five priority classes, respectively. The simulation-optimization is able to find all optimal priority assignments $\mathbf{P}$ in the first set of ex periments. We also observe that the total cost reduction compared to FCFS policy increases with the increasing number of SKUs and priority classes.

Table 1. Performance comparisons for the first set of experiments

(a) The deviation from optimal solutions

	Number of priority class			
Number of SKUs	2	3	4	5
4	0.000	0.000	0.000	
5	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	
7	0.000	0.000		

(b) The % improvement compared to FCFS policy

	Number of priority class			
Number of SKUs	2	3	4	5
4	3.00	3.00	3.00	
5	15.00	15.70	15.70	15.70
6	11.40	11.90	11.90	
7	41.00	44.10		

To generate testbed for the second set of experiments, we use the same approach used in Turan et al. (2018). That is, a full factorial design of experiment (DoE) with three factors and two levels per factor is used to generate a total of 8 instances. We choose N as 10 and 20, and set utilization rate ρ 0.65 and 0.80. The minimum holding cost, h_{min}, levels are chosen as 1 and 100. The maximum holding cost, h_{max} is fixed at 1,000. The backorder 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.

Table 2 documents both performance comparison with FCFS and algorithm runtime for each problem factor. The expected percentage improvement achieved by simulation-optimization compared to FCFS is calculated as follows:

$$\Delta = 100 \times \frac{\mathbb{E}\left[TC^{FCFS}\right] - \mathbb{E}\left[TC^{Sim-Opt}\right]}{\mathbb{E}\left[TC^{FCFS}\right]}$$

where $\mathbb{E}\left[TC^{FCFS}\right]$ and $\mathbb{E}\left[TC^{Sim-Opt}\right]$ denote the objective function values in Eq.(1) obtained when FCFS policy and simulation-optimization algorithm employed, respectively. The proposed approach pro-

duces around 70% cost reduction in reasonable computational times. Further, in the optimized assignments, the number of priority class J may reach up to 13 for the large instance with 20 SKUs.

Table 2. Performance of simulation-optimization for larger instances.

Case ID	N	ρ	h_{min}	Δ	J	Runtime (CPU seconds)
1	20	0.80	1	84	12	28112
2	10	0.80	100	83	8	26233
3	20	0.80	100	55	12	26916
4	10	0.80	1	56	5	26295
5	10	0.65	1	91	6	26228
6	20	0.65	100	74	9	27868
7	20	0.65	1	61	13	27548
8	10	0.65	100	58	6	26086

5 Conclusions and Further Research

There are a number of industries where repair, rather than the replacement of equipment, is economically necessary or feasible. To improve the repair process and to increase the asset availability, repair facilities prefer to hold reserve stocks for repairable parts. Another way of increasing the asset availability is prioritizing SKU failures to shorten the repair lead times. In this context, we modeled a joint optimization of priority assignment and inventory levels of repairable SKUs. To solve the model, we propose a simulation-optimization approach by coupling a GA and a DES. The conducted computational experiments show that the proposed approach yields around 70% total cost reduction on average compared to the FCFS policy.

As further research, it might be beneficial to generate the initial population with a heuristic rule rather than randomly generating priority assignments. We also want to implement different crossover and mutation operators that consider characteristics of SKUs such as failure and service rates. Finally, it would be worthwhile to compare our results with other plausible scheduling heuristics in addition to the FCFS policy.

References

Adan, I. J.-B. F., A. Sleptchenko, and G.-J. van Houtum (2009). Reducing costs of spare parts supply systems via static priorities. *Asia-Pacific Journal of Operational Research 26*(04), 559–585.

Goldberg, D. E. and J. H. Holland (1988). Genetic algorithms and machine learning. *Machine learning 3*(2), 95–99.

Hausman, W. H. and G. D. Scudder (1982). Priority scheduling rules for repairable inventory systems. *Management Science 28*(11), 1215–1232.

Iravani, S. M. and V. Krishnamurthy (2007). Workforce agility in repair and maintenance environments. *Manufacturing & Service Operations Management 9*(2), 168–184.

Kosanoglu, F., H. H. Turan, and M. Atmis (2018). A simulated annealing algorithm for integrated decisions on spare part inventories and cross-training policies in repairable inventory systems. In *Proceedings of International Conference on Computers and Industrial Engineering*, pp. 1–14.

Mahdavi, I., M. M. Paydar, M. Solimanpur, and A. Heidarzade (2009). Genetic algorithm approach for solving a cell formation problem in cellular manufacturing. *Expert Systems with Applications 36*(3), 6598–6604.

Pyke, D. F. (1990). Priority repair and dispatch policies for reparable-item logistics systems. *Naval Research Logistics (NRL) 37*(1), 1–30.

Sherbrooke, C. C. (1968). Metric: A multi-echelon technique for recoverable item control. *Operations research 16*(1), 122–141.

Sleptchenko, A., H. H. Turan, S. Pokharel, and T. Y. ElMekkawy (2019). Cross-training policies for repair shops with spare part inventories. *International Journal of Production Economics 209*, 334–345.

Sleptchenko, A., M. C. van der Heijden, and A. van Harten (2005). Using repair priorities to reduce stock investment in spare part networks. *European Journal of Operational Research 163*(3), 733–750.

Turan, H. H., A. Sleptchenko, S. Pokharel, and T. Y. ElMekkawy (2018). A clustering-based repair shop design for repairable spare part supply systems. *Computers & Industrial Engineering 125*, 232 – 244.

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