A Math-heuristic Algorithm for Concurrent Assignment and Sequence Scheduling in Multi-Shuttle Shared Location Automated Storage and Retrieval Systems

Morteza Kazemi*
Shiraz University of Technology, Shiraz, Iran

Ardavan Asef-Vaziri
California State University, Northridge, Northridge, California, USA

Tahereh Shojaei
Shiraz University of Technology, Shiraz, Iran

Seyyed Amir Hossein Salehi Amiri
Shiraz University of Technology, Shiraz, Iran

Multi-shuttle automated storage and retrieval systems are warehousing systems that are enabled to carry more than one load at a time. They can achieve several storage and retrievals in each cycle and can store a load in the same cell as they retrieve. These capabilities lead to lower flow time, higher utilization, and throughput. In this paper, we address concurrent storage cell assignment and storage and retrieval sequence scheduling under the shared storage policy and the modified 2n-command cycle pattern. A hybrid solution procedure formed by an Ant Colony Algorithm and Adaptive Large Neighborhood Search is proposed. A mathematical assignment model is also integrated into the solution procedure to improve the quality of the heuristics.

* Corresponding Author. E-mail address: kaazemi@sutech.ac.ir

I. INTRODUCTION

The costs of warehousing and maintaining inventory are estimated to be more than 40% of total logistics costs (Guo, Yu, and De Koster, 2016). According to Lee and Schafer (Lee and Schaefer, 1997), the total storage and retrieval time can be reduced by 70% by automating the process.

Automated Storage and Retrieval Systems (ASRS) have been widely implemented in manufacturing, distribution, and service enterprises since the early 1960s (Shell, Hall and Parsley, 2000). They have

been implemented or studied in the range from handling forty-foot containers in the Port of Los Angeles (Asef-Vaziri, Khoshnevis, and Rahimi, 2008) to handling books in the California State University, Northridge (Heinrich and Willis, 2014). The latter was the first-ever ASRS for libraries constructed in 1989-91 at a $2 million cost. During the Northridge Earthquake in 1993, almost 100% of the library's open shelf collection had fallen on the floor. Not one book in the ASRS was damaged, nor was any bin in danger of falling. ASRS has been implemented to provide the ability to handle loads without operator intervention, reduce labor costs, decrease human errors, provide high uptime reliability, increase storage capacity, shorten storage/retrieval times, increase accuracy levels, improve material flow and inventory control, and increase floor-space utilization (Cinar, Oliveira, Topcu, and Pardalos, 2017). Latest market research study estimates that the Global Warehouse Automation Market will grow more than two times from $13 billion in 2018 to $27 billion by 2025, at a compound annual growth rate of 11.7% between 2019 and 2025 (*Warehouse Automation Market - Road to $27B by 2025*).

Shuttle-based storage and retrieval systems are one of the most growing systems in the warehouse automation market. The four essential components of these systems are (i) storage racks, (ii) storage and retrieval crane (Crane), (iii) storage and retrieval shuttles (Shuttle), and (iv) input/output (I/O) stations. The rack structure is composed of a set of columns, known as bays, and a set of rows, known as tires. The intersection of a bay and a tire is referred to as a storage cell. The crane moves simultaneously in both horizontal and vertical directions. The in-aisle trip time is defined by the maximum of the horizontal time and vertical time of the crane trips. The shuttle is installed on the crane to drop-off a load in a storage cell and to pick-up a load from a cell. Typically, each crane has only a single shuttle. Two or three shuttles per crane are also a quite common system configuration, whereas four or more shuttles are technically possible, but rarely applied (Meller and Mungwattana, 1997; Popović, Vidović and Bjelić, 2014a). In the most widespread system configuration, there is just a single I/O-point located at the front end of the rack.

ASRSs have been studied in different aspects, including system analysis, design optimization, and operations planning and control (Azadeh, De Koster, and Roy, 2019). In this paper, we focus on operations planning and control. The tasks of a planning horizon are partitioned into operation cycles. At the beginning of each cycle, the crane starts picking loads at the I/O station, visiting empty cells to store the loads, picking up loads at retrieval cells to deposit at the I/O station, and then starting the next cycle. In Multi-Shuttle ASRS (MS-ASRS), Fig. 1, the crane is usually equipped with two or three shuttles carrying two or three loads per trip. The MS-ASRS design can benefit from multi-aisle and multi-tire scalability as well as improving storage capacity, throughput, and cycle times.

FIGURE 1. A TWIN-SHUTTLE AUTOMATED STORAGE/RETRIEVAL SYSTEM (YANG, PENG, YE, AND MIAO, 2017)

In each storage operation, a crane picks up a load, usually from a conveyor, and stores it in the 30- to 40-m-high racks. Driving and lifting in the aisle take place simultaneously. The process sequence is reversed for a retrieval operation. In the dual command cycle, storage and retrieval job are combined. A shuttle accomplishes both storage and retrieval during a cycle so that on the way from the I/O-point to the respective shelf, a storage request is executed and retrieved on the way back. This would save one movement per dual command cycle; however, there may be an additional wait for pairing a storage transaction with a retrieval. This attribute refers to every single shuttle so that it also covers what is called a quadruple or sextuple command if two or three shuttles exist, respectively. In general, where there are *n* shuttles, it is referred to as 2*n*-command. One intuition to improve both throughput and flow time is to alternate the MS-ASRS tasks within a cycle, which can be accomplished by having retrieval and storage tasks at the same cell in the storage racks. This idea is recognized as a modified 2*n*-command cycle with a shared storage policy where a retrieval cell is used for storage immediately after its item is retrieved (Yang, Miao, Xue and Ye, 2015). In Fig. 2, three storage and three retrieval tasks can be processed in a single cycle by a crane with three shuttles through visiting one empty and three full cells. In each cycle, an empty cell is chosen to assign one load to it, and therefore the capacity of one of the shuttles is released. In the following, after reaching the retrieval cell and placing its item in the released shuttle, another item can be placed in the retrieved item cell, and the capacity of the other shuttle is released. This process is repeated in the next cell. At the fourth cell, the last retrieved item is lifted, and the crane returns to the I/O point. Thus, three storage and three retrieval requests can be processed in a single cycle by a crane with three shuttles through visiting one empty and three full cells.

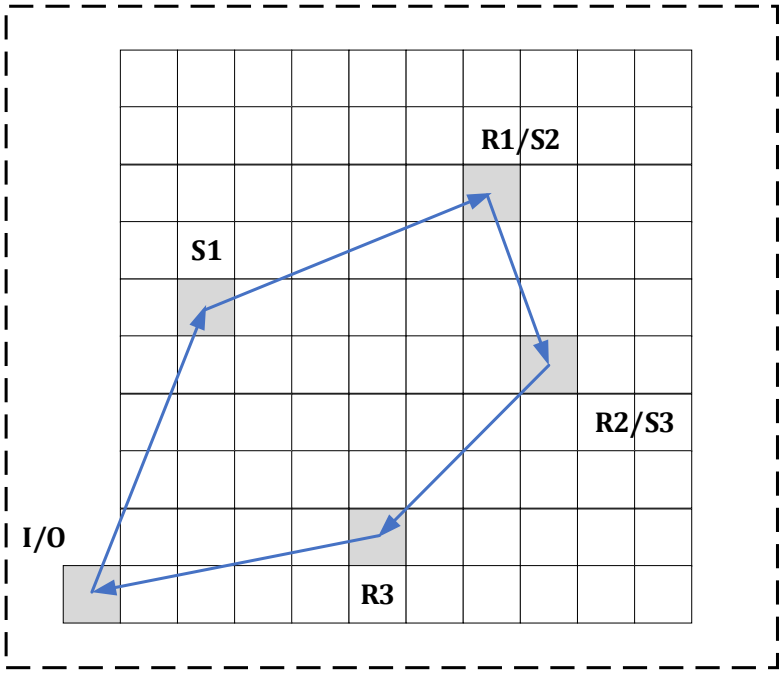

FIGURE 2. MODIFIED 2N-COMMAND CYCLE IN AN MS-ASRS.

In this paper, to reduce the total loaded and empty trip times for the MS-ASRS problem under the 2*n*-command policy, hybrid algorithms that integrate Ant Colony Optimization Algorithm and Adaptive Large Neighborhood Search Algorithm is developed. In addition, a mathematical model, based on the assignment problem, is proposed to improve the quality of the solutions.

The remainder of this paper is organized as follows. A literature review is provided in Section 2. Section 3 describes the optimization model. The proposed hybrid algorithm is discussed in Section 4. Computational considerations are covered in Section 5, and conclusions follow in Section 6.

II. LITERATURE REVIEW

ASRS's analysis covers system analysis, design optimization, and operations planning and control (Azadeh, De Koster, and Roy, 2019). Roodbergen and Vis (2009) provide an overview of ASRS's classification and research. Gagliardi, Renaud, and Ruiz (2012) review the simulation-based models for ASRSs. Boysen and Stephan (2016) survey various ASRS scheduling models. They provide a classification scheme to differentiate crane scheduling problems. We focus on the papers relating to the sequencing of storage and retrieval in a single-aisle (two racks) problem, which is part of operations planning and control.

Han, McGinnis, Shieh, and White (1987) studied retrieval sequencing methods for unit-load ASRS, in which SKUs can be stored in any available location. They present a heuristic nearest-neighbor procedure. This procedure successively pairs an open storage location for the storage move of the dual command with the closest retrieval request. The problem is extended by Chen, Langevin, and Riopel (2010). They explicitly consider the inventory duration of the unit loads, such that two-unit loads with overlapping storage

intervals cannot be assigned the same storage location. The problem is formulated in a mixed-integer model supported by construction and improvement algorithms.

Adaptions of a nearest-neighbor heuristic for class-based storage policy are presented in Eynan, Rosenblatt, and Rosenblatt (1993). They show that the nearest-neighbor policy in class-based storage led to significant savings in inter-departure time. Lee and Schaefer (1996) implement the linear assignment model in the same problem for pairing an equal number of storage and retrieval requests. However, the feasibility of the solutions is not guaranteed. They apply the ranking algorithm (Murty, 1968) to successively determine the next best assignment until either the optimal solution is obtained or the maximum number of iterations is achieved. To avoid infeasibility, they apply a repair heuristic that constructs feasible solutions in each iteration.

Lee and Schaefer (1997) presented several exact and heuristic sequencing methods under static and dynamic conditions. They found that the sequencing methods can significantly reduce the crane machine travel time and increase the throughput. Chung and Lee (2008) used a genetic algorithm for sequencing the storage and retrieval operations for a dual-cycle mode, under a random storage policy. They compare the performance of their genetic algorithm with two greedy heuristics.

MS-ASRS is widely reflected in a sequence of recent papers. Meller and Mungwattana (1997) described the characteristics of MS-ASRS. They developed an analytical model with the first-come-first-served (FCFS) policy under the nearest-neighbor and reversed nearest- neighbor sequencing policies. They compared two policies for a crane with three shuttles. They provided guidelines leading to improvement in the performance of the system. They referred to their proposed operational policies as sextuple command and modified sextuple command. The authors also concluded that ASRSs with more shuttles have more significant reductions in the total travel time in systems with higher intensity in operations. Tanaka and Araki (2007) propose a greedy heuristic procedure for the operations of MS-ASRS. Popović, Vidović, and Bjelić (2014) studied the modified sextuple commands for a crane with three shuttles in a class-based storage system. The planning horizon is formed by several successive cycles. They propose three rule-based heuristics and a genetic algorithm.

Yang, Miao, Xue, and Qin (2015) analyze an MS-ASRS with a facultative number of shuttles. They formulate an integer programming and a dynamic programming model to formulate the problem. A two-phase tabu search approach and genetic algorithm coupled with a modified nearest-neighbor heuristic are presented. Yang, Miao, Xue, and Ye (2015) add open shelf storage to this problem. In this environment, any open shelf is a potential storage location for a storage request. They also apply the modified $2n$ - command cycle. They formulate the problem as a mixed-integer model and solve it with a variable neighborhood search approach. Yang, Peng, Ye, and Miao (2017) extend this problem in block sequencing. They present an integer quadratic programming model. The authors propose two tabu search algorithms, where the first-come, first-served, and nearest-neighbor are used to generating initial solutions.

III. PROBLEM DESCRIPTION AND MATHEMATICAL MODEL

Our global optimization model is formulated in the framework of Yang, Miao,

Xue, and Ye (2015) model for the optimization of the shared storage cell assignment and storage/retrieval scheduling problem.

Assumptions:

- The set of retrieval and storage tasks are known and fixed in the planning horizon.
- The storage and retrieval tasks are assigned to a set of operation cycles. Cycles are performed immediately after each other, with no time gap.
- The number of storage and retrieval tasks are equal in all cycle.
- Crane trip time follows Chebyshev inequality.
- Each open shelf is a potential storage location for a storage request.
- The pick-up and drop-off times are constant.
- The modified 2n-command cycle pattern is implemented. The number of cells visited in each cycle is one plus the number of shuttles.

Notations, Parameters, and Decision Variables:

n : number of shuttles,
m : number of cycles in a planning horizon,
M : number of tasks to be processed in the planning horizon $(M = mn)$,
L : the set of cells in the racks,
d_{ij} : the Chebyshev trip time between cells i and j for $i, j \in L$, $i, j = 0$ stands for the I/O station,
LE : the set of initial empty cells,
LR : the set of retrieval cells defined by the retrieval tasks,
$AL = LE \cup LR$: the set of cells that may be visited in the planning horizon,
$z_{i,l,k}$: A binary decision variable which is equal to 1, if cell i is visited as the l-th point in the k-th cycle, and is equal to 0 otherwise, for $i \in AL\,; l = 1, \dots, n+1\,; k = 1, \dots, m$,

The objective function is stated as:

$$\min \sum_{k=1}^{m} \left\{ \sum_{i \in LE} (d_{0i} z_{i,1,k}) + \sum_{i \in AL} \sum_{j \in AL} \left[\sum_{l=1}^{n} (d_{ij} z_{i,l,k} z_{j,l+1,k}) \right] + \sum_{i \in LR} d_{i0} z_{i,n+1,k} \right\} \quad (1)$$

As stated earlier, the travel sequence is partitioned into three segments: $\{'0-S'\}, \{'S-R', \dots, S-R'\}$, and $\{'R-0'\}$. The first stage is the trip between the I/O point and an empty cell. The second is the trip between cells for both retrieval and storage, and the third stage is the trip from the last retrieval position to the I/O point. The objective function is the minimization of the total travel times, subject to the following constraints

$$\sum_{k=1}^{m} z_{i,1,k} \leq 1, \qquad \forall i \in LE \quad (2)$$

$$\sum_{i \in LE} z_{i,1,k} = 1, \qquad \forall\, k = 1, \dots, m \quad (3)$$

$$\sum_{l=2}^{n+1} \sum_{k=1}^{m} z_{i,l,k} = 1, \quad \forall i \in LR \quad (4)$$

$$\sum_{i \in LR} z_{i,l,k} = 1, \qquad \forall\, k = 1, \dots, m \;\; \forall l = 2, \dots, n+1 \quad (5)$$

$$z_{i,l,k} \in \{0,1\} \quad (6)$$

Constraint (2) ensures that each empty cell is visited at most once during the planning horizon. Constraint (3) implies that the first visited cell of each cycle is empty (for storage). Constraint (4) ensures that each retrieval cell is visited once in the second through n + 1th position in one of the operation cycles. Constraint (5) ensures that the full cells are visited (2 to n +1) in each operation cycle. Constraint (6) is the binary condition on the decision variables.

IV. HEURISTIC PROCEDURE

Our problem in its simplest version is converted to the case with just one cycle and one storage cell. The problem is then reduced to the Traveling Salesman Problem, which is proved to be NP-hard. Since the general form of the problem is more complicated; we provide tailored heuristic algorithms to solve real-life cases. In our heuristic procedure, a feasible solution is shown as a sequence of cycles. Each cycle is started from the I/O point and continues to visit *n+1* cells. The first cell is an empty cell to store an item. In each of the next *n-1* cells, one item is retrieved, and one item is stored. The last cell is a retrieval cell. We first develop a feasible solution based on an Ant Colony Optimization (ACO) algorithm. After generating the initial solution, we ignore the first storage place in each cycle and improve the solution by applying the Adaptive Large Neighborhood Search (ALNS) algorithm to explore the search space of the problem. In the end, we use a mathematical formulation based on the assignment problem to find the optimal storage cells of each cycle for the generated partial solution. The pseudo-code of the proposed heuristic algorithm is elaborated in Fig. 3.

```
1 Construct a feasible solution, x, by ACO ;set x* := x
2 Repeat
3    Choose a destroy neighborhood N- and
      a repair neighborhood N+ using the roulette wheel
      selection based on previously obtained weights
4   Generate a new solution x' from x using the heuristics
      corresponding to the chosen destroy and repair
      neighborhoods
5   If  x' can be accepted then
        set x: = x';
6   Update scores and weights of N- and N+
7   If f(x) < f (x*) then
    set x* :=x
8 Until stop criteria are met
9 Apply Mathematical Model for Local search on x*
10 Return x*
```

FIGURE 3. THE PSEUDO-CODE OF THE PROPOSED HEURISTIC APPROACH.

4.1. Initial Solution Generation Process

The algorithm generates an initial solution by applying an ACO construction algorithm by benefiting from Akpinar (2016). In each cycle, each artificial ant located in cell i, choose cell j based on the stochastic rule as represented in (7).

$$j = \begin{cases} j_1 = \underset{j \in A_i}{\operatorname{argmax}} \left\{ \left[\tau_{(i,j)}\right]^{\alpha} \times \left[\eta_{ij}\right]^{\beta} \right\}, & if\ r \le r_1 \\ j_2 : p(i,j_2) = \dfrac{\left[\tau_{(i,j_2)}\right]^{\alpha} \times \left[\eta_{ij_2}\right]^{\beta}}{\sum_{j \in A_i} \left(\left[\tau_{(i,j_2)}\right]^{\alpha} \times \left[\eta_{ij_2}\right]^{\beta} \right)}, & if\ r_1 < r \le r_1 + r_2 \\ j_3 : random\ selection\ of\ j \in A_i\ . & if\ \ r_1 + r_2 < r \le r_1 + r_2 + r_3 \end{cases} \quad (7)$$

where $\boldsymbol{r} \in [\boldsymbol{0}, \boldsymbol{1}]$ is a random number, and $\boldsymbol{0} \le \boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}$ are user-defined parameters, so that $\boldsymbol{r_1} + \boldsymbol{r_2} + \boldsymbol{r_3} = \boldsymbol{1}$. The intensity of pheromone between cell $\boldsymbol{i}$ and cell $\boldsymbol{j}$ is denoted by $\boldsymbol{\tau_{(i,j)}}$. The initial value of pheromone between each pair is set to $\boldsymbol{\tau_0}$. The value of $\boldsymbol{\eta_{ij}}$ represents the heuristic's information about a trip segment between the $\boldsymbol{i}$ th and $\boldsymbol{j}$ th cells and is considered as the reverse Chebyshev distance between these cells. $\boldsymbol{A_i}$ is the set of unvisited cells that can be chosen as the next visible cell $\boldsymbol{j}$ by the Crane. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are heuristic parameters that designate the relative significance of the pheromone intensity and heuristic information.

After all ants generate their solution in each cycle, the global updated rule is implemented by (8):

$$\tau(i,j) \leftarrow (1-\rho) * \tau(i,j) + \rho * \Delta\tau(i,j) \quad (8)$$

Where $\Delta\tau(i,j)$ is the inverse of the solution objective, if the edge (i,j) belongs to the best solution of this iteration, and zero otherwise. The parameter ρ represents the evaporation rate of the pheromone. In the framework of this rule, the algorithm increases the pheromone level on edges that belong to the best solution for each iteration. The process of generating a feasible solution and pheromone update is repeated until the desired number of iterations is reached.

4.2. Improvement Mechanism

The ALNS algorithm is developed to improve the ACO algorithm solutions. The ALNS (Ropke and Pisinger (2006)), is a meta-heuristic focused on the application of rules for consecutive partial destruction and reconstruction of the current solution to perform a search in its neighborhood. An incumbent solution consists of *m* cycles. Each cycle starts with the I/O station and a storage cell. The remaining part of the cycle consists of *n* retrieval cells. All the proposed destroy and repair operators in our ALNS algorithm are implemented on the retrieval cells. A destroy algorithm is selected to remove a certain number of cells, q, from the current solution. A repair algorithm is selected to replace them in a different form.

The destroy (repair) phases consist of choosing at random a removal (insertion) operator among a set of removal (or insertion) operators. Let w_i be the weight associated with destroying (repair) operator. The probability of choosing this operator is given by $w_i / \sum_j w_j$, where *j* belongs to the set of all destroy (repair) operators. The proposed algorithm, inspired by the algorithm introduced by Martins De Sá, Contreras, and Cordeau (2015), adaptively updates the weights. The proposed ALNS algorithm is partitioned into segments, where each segment consists of a *Reset Number* of iterations. The weight of each operator is updated based on the performance score of each operator at the end of each segment, where the score represents the contribution of

the operator to the improvement in the objective function.

At the beginning of each segment, the scores of the operators are set to zero. After choosing the pair of removal-insertion operators, the score of both is increased by a) σ1 if the solution is the new best overall solution, b) σ2 if the solution is not the best overall, but better than the best in the current iterations, and c) σ3 if the solution that is worse than the current one, but satisfies the acceptance criterion. In practice, σ1 ≥σ2 ≥σ3.

At the start of the algorithm, the initial weight of all operators is set to the same value. w_i^k is considered as the weight of the operator i in the $k-th$ segment. The relation between the weights of the operator in the consecutive steps is described by (9).

$$w_i^{k+1} = \begin{cases} w_i^k, & if\ n_i^k = 0 \\ (1-\eta)w_i^k + \eta\frac{s_i^k}{n_i^k}, & otherwise \end{cases} \quad (9)$$

where s_i^k is the score of operator $\boldsymbol{i}$, and n_i^k is the number of times when the operator $\boldsymbol{i}$ was used in the k^{th} segment. $\eta \in [0,1]$ is the reaction factor representing how fast the weights change.

At each step, the new solution S_{new}, is achieved by applying the pair of the removal-insertion operator on the current solution $S_{current}$. It is accepted as the new current solution for the next iteration of ALNS if the acceptance criteria of the simulated annealing is met. Based on these criteria, a new solution will replace the current solution for the next iteration if it is better than the current solution, i.e., $f(S_{new}) < f(S_{current})$ where f denotes the cost of the solution. Otherwise, the new solution will be accepted with a probability equivalent to $e^{-(f(S_{new})-f(S_{current}))/T}$. In this expression, $T > 0$ is the temperature parameter at an individual iteration. The value of the initial temperature for the first iteration is set to T_{start}. The temperature T_{start} is decreased at each iteration according to the constant cooling rate, $0 < k < 1$, to achieve a better solution. The algorithm terminates when a certain number of iterations are applied. The Destroy and Repair algorithms of our ALNS algorithm are discussed in the following subsection.

4.2.1. Destroy Operator

After generating an initial solution, one of several destruction operators is applied at each ALNS iteration. The idea is to destruct the incumbent solution by repeatedly removing q retrieval cells. The destruction operators employed in our execution are as follows:

(a) Random Removal Heuristic

This operator randomly selects q cells for removal. The ideas are diversification in the search space and escaping from local optima.

(b) Worst Removal Heuristic

This operator consists of removing, iteratively, q cells, yielding the most significant decrease in the objective function value.

(c) Worst Edge Removal

This operator consists of iterative removal of q cells that belong to edges with the largest length.

(d) Shaw Removal Heuristic

This operator consists of removing, iteratively, q cells that yield the smallest relatedness, according to Shaw's measure (Shaw, 1998). The purpose of this operator is to eliminate the cells that are far from each other. The relatedness measure of two cells i and j based on the Chebyshev distances is stated as:

$$R(i,j) = \varphi.d_{ij} + (1-\varphi).l_{ij} \quad (10)$$

where $\varphi \in [0,1]$ is the trip time weight parameter. If cell i and j are in the same operation cycle, l_{ij} is set to -1; otherwise, it is set to 1. The first step of this operator is performed by a random selection of a cell i and adding it to the elimination list.

(e) Cloud Heuristic

This operator consists of removing, iteratively, q cells that yield the smallest distance to the center of gravity of the set of retrieval tasks already added in the elimination list. The first step of this operator is performed by randomly choosing a cell.

Due to the deterministic nature of constraints (b)-(e), following Ropke and Pisinger (2006), we add randomness to these operators through a parameter p >1. Let H be a sorted set of cells ordered according to the removal operator's main criterion, and let r be a random number from the interval (0,1). The idea is to remove the $H\ [r^p * |H|]$ −th cell of H, instead of always removing the first one. The parameter p controls how much randomness is added to these operators, where small values of p result in more randomness.

4.2.2. Repair Operators

After the utilization of a destruction heuristic, the repair heuristic inserts each previously eliminated cell in the partial operation cycle. The following subsection expresses the repair operators employed in our algorithm.

(a) Greedy Heuristic

In this operator, the best possible retrieval cell is identified by enumerating all the potential candidates and identifying the cell leading to the best objective function value. This process is repeated for all the remaining removing cells. The cell that yields the smallest increment in the objective value is selected and inserted in its best cell. This cell is removed, and the process is repeated until all the removed retrieval cells are located. The shortcoming of this heuristic algorithm is that it often delays cells, which results in a more significant increase in the objective function. Therefore, the regret heuristic approach is introduced to solve this problem.

(b) Regret Heuristic

In this operator, the best and next best possible cell in partial solution is determined for each of the removed retrieval cells. The difference between the objective value of these two partial solutions is considered a regret measure. The cell with the highest regret value is chosen and inserted into its best cell. This cell is removed, and the process is repeated until all the removed retrieval cells are located.

4.3. Local improvement

The solution obtained from the ALNS phase is to try on retrieval cells and the first storage cell in each cycle is temporarily removed. To obtain an optimal storage location for the partial solution the following assignment problem-based integer programming model is implemented.

$$\min z = \sum_{i=1}^{m} \sum_{j \in S} c_{ij} x_{ij} \tag{11}$$

$$s.t \quad \sum_{j \in S} x_{ij} = 1, \qquad \forall\, i = 1, \dots, m \tag{12}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall\, j \in S \tag{13}$$

$$x_{ij} = 0, \text{or } 1. \tag{14}$$

where S is the set of all storage cells, x_{ij} is a binary decision variable which is equal to 1, if cell j is allocated to the i^{th} operating cycle and is 0 otherwise. The parameter c_{ij} is the cost of assigning the storage cell j to the i^{th} operating cycle. It is equal to the sum of the trip times of the crane from the I/O station to the cell j, and from this cell to the first retrieval cell in the i^{th} operating cycle. Constraint (12) determines the storage cell for each cycle. Constraint (13) ensures that each storage cell must be visited at most once in all operation cycles.

4.4. The complexity of the Proposed Heuristic

The complexity of each iteration of the proposed ACO algorithm is O (AntNo.m.n), where AntNo is the number of ants. Therefore the ACO's complexity is O ("ACO iteration No." . "AntNo.m.n"). In each iteration of the ALNS algorithm, 2q operation should be done to construct a new solution. As q<m therefore the complexity of the ALNS can be considered as O (m. "ALNS iteration No."). The proposed assignment problem is in order of O(m^3) (Tomizawa, 1971).

As these heuristics are performed sequentially, the complexity of the proposed algorithm will be O(AntNo.m.n) + O(m. ALNS iteration No.) + O(m^3). As the fixed value of the iteration number in all of the test problems, the iteration number value can be omitted from the complexity and conclude that the overall complexity of the heuristic algorithm is O(m.n+m^3).

V. COMPUTATIONAL CONSIDERATIONS

All algorithms are coded in C# and run on a computer with a 2.6 GHz processor and 4 GB RAM. Two categories of medium and large instances are generated. Each instance is solved ten times, and the average objective function value is reported. The Taguchi experiments design is utilized to identify the most efficient level for algorithms' parameters.

5.1. Test Problems

The performance of the algorithms is evaluated by generating different classes of the test problem. Each class of instances is represented as (*T, L, m, n*) combination, where *T* is the number of tires, *L* is the number of cells, *m* is the number of cycles in a time horizon, and *n* is the shuttle's capacity. The values associated with these parameters are shown in Table 1. It is assumed that twenty percent of the storage space is available at the beginning of the planning horizon.

TABLE 1: VALUES OF PARAMETERS

T	Number of tires	10,20,30, 50
L	Number of cells	60,70,80, 81,90, 100, 200, 400, 600, 1000
m	Operational cycles number	2,3,4
n	Shuttle's capacity	3,4

5.2. Parameter Tuning

The parameters of our models are partitioned into two categories. The values of the first category, based on the values reported in the relevant references, mentioned in Table 2.

TABLE 2 VALUES OF PARAMETERS.

Parameters	Description	Value
φ	Shaw's trip time weight	0.75
τ_0	Primary pheromone intensity	1
r_1	Random number in ACO transition rule	0.5
r_2	Random number in ACO transition rule	0.3
r_3	Random number in ACO transition rule	0.2
k	Cooling rate	0.99
q	Number of removable places	3
p	degree of randomness	3
ACO iteration No.	Total Iteration No. of ACO algorithm	10
AntNo.	Number of ants	10

The values of the second category of parameters are determined by applying the Taguchi method (Taguchi, 1989). This method is used to tune the parameters of metaheuristic algorithms (Nalbant, Gökkaya, and Sur, 2007). The Taguchi method provides a unique design of orthogonal arrays to study the entire parameter space with a small number of experiments. By computing the objective function value for each array, the signal-to-noise (S/N) ratio is computed for parameters. This ratio is used to measure the deviation of quality characteristics from the desired values. The larger the S/N ratio, the better the performance characteristic (Nalbant, Gökkaya, and Sur, 2007).

For each of the parameters, as shown in Table 3, three levels are considered. These levels were identified based on the values reported in the relevant references. Based on these levels and using the Minitab software, the $L27$ orthogonal array is used to set the parameter. The best level of each parameter is reported in Table 4.

TABLE 3: FACTORS AND THEIR BEST LEVEL.

Factor	Factors levels		
	Level 1	Level 2	Level 3
T_{start}	50	100	1000000
σ1	37	42	10
σ2	32	31	5
σ3	9	22	2
η	0.2	0.15	0.5
Reset number	200	100	450
ALNS iteration No.	250000	50000	25000
α	1	2	3
β	2	4	6
ρ	0.01	0.05	0.1

TABLE 4: BEST LEVEL OF PARAMETERS

Factor	Best Level
T_{start}	1000000
σ1	10
σ2	5
σ3	2
η	0.15
Reset number	450
ALNS iteration No.	25000
α	1
β	4
ρ	0.01

5.3. Computational Results

The average objective function values are considered as the measure of effectiveness. Following the tuning process and obtaining the best parameter values, the problem is solved by the proposed hybrid algorithm. The ACO algorithm creates initial solutions. The combined Assignment-ALNS algorithm then improves the quality of the initial solutions. The objective function values of ACO, ALNS, and Assignment-ALNS and the improvement percentage values for the average of ten instances of each of the medium and large class problems are shown in Table 5 and Table 6, respectively.

- Numerical results on medium instances

As shown in Table 5, in the medium instances, the ALNS improves the solution of ACO on average by 11.13%. The Assignment-ALNS algorithm was able to improve the solution of the ACO algorithm in a range of 15.03% to 32.15%, with an average of 23.39%.

TABLE 5: ACO, ALNS, ASSIGNMENT-ALNS OBJECTIVE FUNCTION VALUES, AND IMPROVEMENT PERCENTAGES FOR THE AVERAGE OF ALL TEN MEDIUM -SCALE SAMPLES.

Number of Tires	Number of shelf cells	Number of Shuttles	Number of cycles	Z_ACO	Z_ALNS	Z_ALNS-Assignment	Gap1	Gap2
9	81	3	2	49.4	41.7	34	15.59%	31.17%
9	81	3	3	62.4	57.5	46.1	7.85%	26.12%
9	81	3	4	76.9	74.3	58	3.38%	24.58%
9	81	4	2	47.7	42.8	36	10.27%	24.53%
9	81	4	3	66.1	60.8	50.6	8.02%	23.45%
9	81	4	4	92.4	83.5	70.7	9.63%	23.48%
10	60	3	2	43.6	38.8	37	11.01%	15.14%
10	60	3	3	63.9	55.9	47.6	12.52%	25.51%
10	60	3	4	79.1	67.4	55.8	14.79%	29.46%
10	60	4	2	44.4	37	34	16.67%	23.42%
10	60	4	3	71.2	63	57.3	11.52%	19.52%
10	60	4	4	80.9	73.2	62.8	9.52%	22.37%
10	70	3	2	39.8	35.8	29..5	10.05%	25.88%
10	70	3	3	64.8	57.7	53.5	10.96%	17.44%
10	70	3	4	83.7	73.6	68.9	12.07%	17.68%
10	70	4	2	51.1	43.8	38.7	14.29%	24.27%
10	70	4	3	74	61	54.4	17.57%	26.49%
10	70	4	4	81.2	73.2	61.1	9.85%	24.75%
10	80	3	2	50.7	40	34.4	21.10%	32.15%
10	80	3	3	60.7	56.1	46.3	7.58%	23.72%
10	80	4	2	46.9	42.1	33.3	10.23%	29.00%
10	80	4	3	68.4	61.4	54.7	10.23%	20.03%
10	80	4	4	92.8	87.5	75.2	5.71%	18.97%
10	80	4	4	95.1	81.5	71.8	14.30%	24.50%
10	90	3	3	64.7	58.9	50.6	8.96%	21.79%
10	90	4	2	56.4	48.7	44.5	13.65%	21.10%
10	90	4	3	67.2	64.4	57.1	4.17%	15.03%
10	90	4	4	97.9	87.9	75	10.21%	23.39%

The percentage of improvement by the Assignment-ALNS in comparison to the ACO algorithm, based on the number of cells and cycles for medium instances are shown In Fig. 4 and Fig. 5.

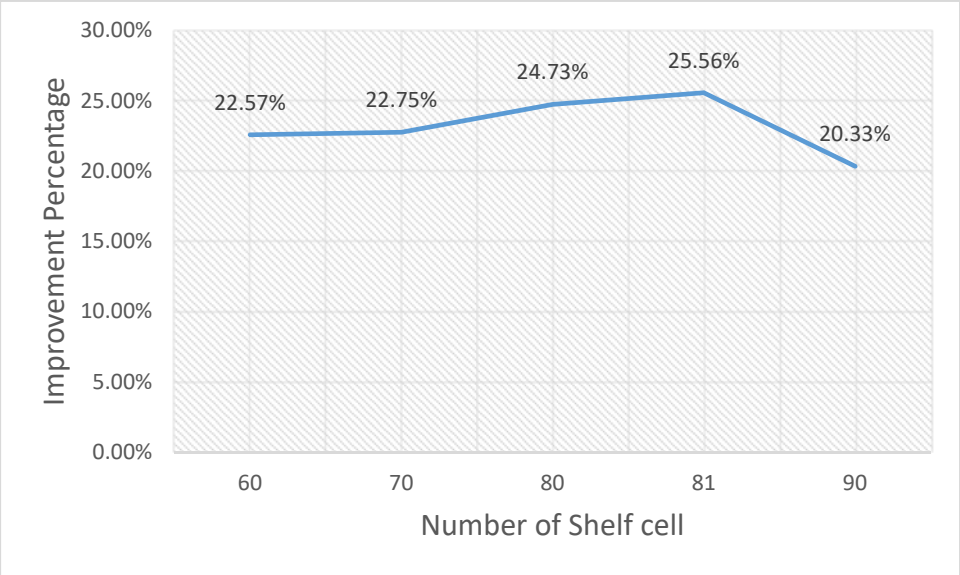

FIGURE 4. IMPROVEMENT PERCENTAGE VALUES BASED ON THE NUMBER OF SHELF CELLS IN MEDIUM SIZE PROBLEMS.

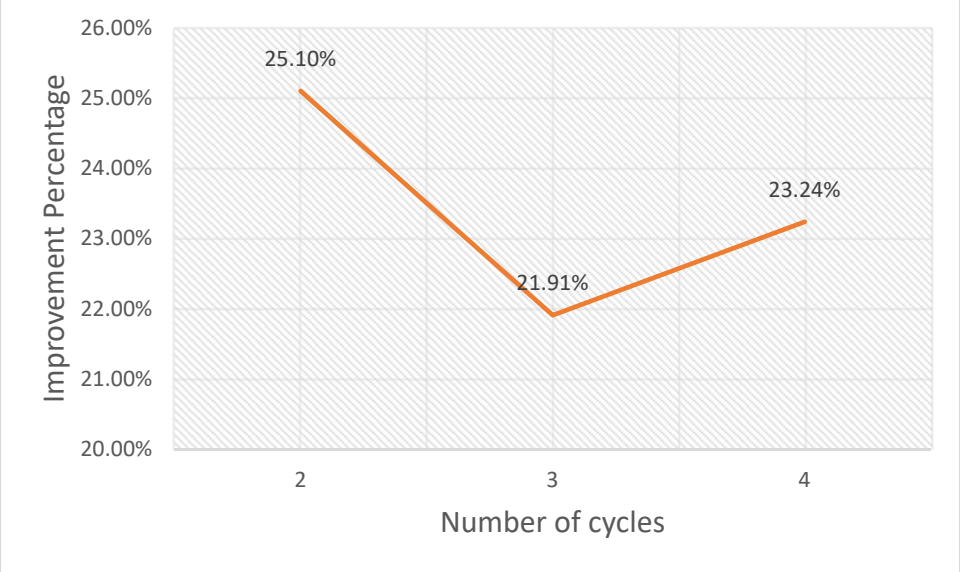

FIGURE 5. IMPROVEMENT PERCENTAGE VALUES BASED ON THE NUMBER OF CYCLES IN MEDIUM SIZE PROBLEMS.

- Numerical results on large instances

Computational results for large instances are shown in Table 6. The ALNS improves the solution of ACO on average by 10.11%. The Assignment-ALNS algorithm was able to improve the solution of the ACO algorithm in a range of 17.08% to 22.96% with an average of 20.45%.

TABLE 6: ACO, ALNS, ASSIGNMENT-ALNS OBJECTIVE FUNCTION VALUES, AND IMPROVEMENT PERCENTAGES FOR THE AVERAGE OF ALL TEN LARGE-SCALE SAMPLES.

Number of shelves	Number of shelf cells	Number of Shuttles	Number of cycles	Z_ACO	Z_ALNS	Z_ALNS-Assignment	Gap1	Gap2
10	100	3	2	58	52.2	47.1	10.00%	18.79%
10	100	3	3	55.7	51.4	45.6	7.72%	7.72%
10	100	3	4	58.9	53	46.3	10.02%	21.39%
10	100	4	2	60.1	52.4	46.3	12.81%	12.81%
10	100	4	3	59.5	52.5	46.8	11.76%	21.34%
10	100	4	4	58.6	51.7	47.3	11.77%	11.77%
20	200	3	2	58.3	53.6	46.5	8.06%	20.24%
20	200	3	3	56.8	52	46.1	8.45%	8.45%
20	200	3	4	59.4	51	45.8	14.14%	22.90%
20	200	4	2	59.5	52.6	46.2	11.60%	11.60%
20	200	4	3	58.3	52.7	46.4	9.61%	20.41%
20	200	4	4	58.3	50.4	47	13.55%	13.55%
20	400	3	2	56.6	51.5	46.6	5.68%	14.65%
20	400	3	3	56.4	49.5	46.2	12.23%	12.23%
20	400	3	4	58.8	52.3	46.5	11.05%	20.92%
20	400	4	2	59.5	54.6	46	8.24%	8.24%
20	400	4	3	58.5	53.7	47	8.21%	19.66%
20	400	4	4	59.5	52.2	46.7	12.27%	12.27%
30	600	3	2	58.6	53.8	46.5	8.19%	20.56%
30	600	3	3	56.2	49.6	46.6	11.74%	11.74%
30	600	3	4	59.8	54.5	46.3	8.86%	22.58%
30	600	4	2	60.2	53.9	46.7	10.47%	10.47%
30	600	4	3	57.2	51.9	45.8	9.27%	19.93%
30	600	4	4	60	53.6	46.5	10.67%	10.67%
50	1000	3	2	58.5	53.5	46	8.55%	21.37%
50	1000	3	3	56.8	53.3	46.3	6.16%	6.16%
50	1000	3	4	58.5	51.3	46.3	12.31%	20.85%
50	1000	4	2	59.4	51.4	45.8	13.47%	13.47%
50	1000	4	3	57.9	52.1	46.1	10.02%	20.38%
50	1000	4	4	56.7	53.1	46.6	6.35%	6.35%

The percentage of improvement by the Assignment-ALNS in comparison to the ACO algorithm, based on the number of cells and cycles for large instances are shown in Fig. 6 and Fig. 7.

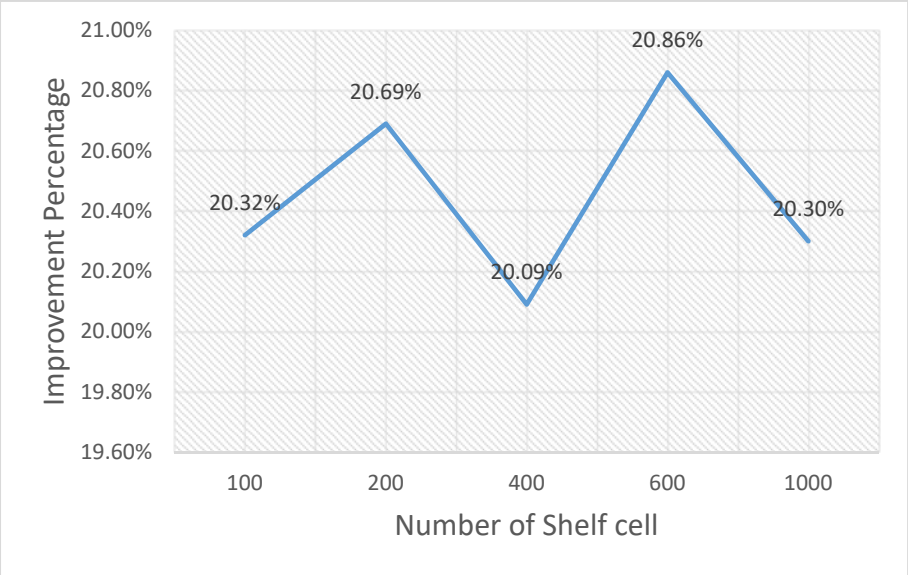

FIGURE 6. IMPROVEMENT PERCENTAGE VALUES BASED ON THE NUMBER OF SHELF CELLS IN LARGE PROBLEMS.

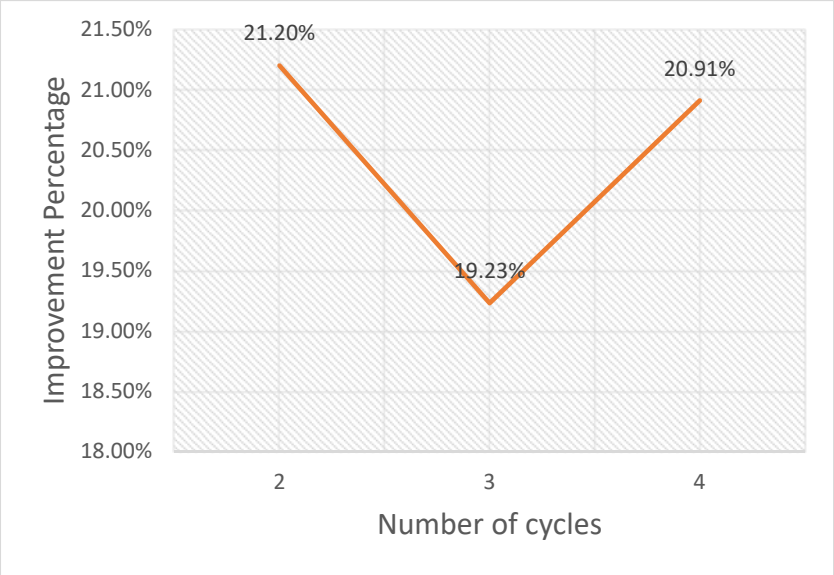

FIGURE 7. IMPROVEMENT PERCENTAGE VALUES BASED ON THE NUMBER OF CYCLES IN LARGE PROBLEMS.

VI. CONCLUSIONS

New companies are looking for new ways to improve procurement, production, and distribution operations to meet complex customer requirements. ASRSs bring flexibility to adapt to rapidly changing demands. In this study, the issues of space allocation and operation sequencing have been studied concurrently. Shared storage and modified $2n$ -command operating cycle pattern modes are considered in the planning

horizon, which includes the implementation of successive operating cycles for the crane. Due to the complexity of the problem, a combined heuristic algorithm consisting of an Ant Colony Algorithm and the Adaptive Large Neighborhood Search algorithm was proposed. The mathematical programming model was then used as a local search algorithm. The target function values of the ACO, ALNS, and Assignment-ALNS algorithms and the percentage improvements were reported for the average of all ten runs for each problem instance. The results show that the Assignment-ALNS algorithm significantly improves the initial solution obtained by the ACO algorithm.

REFERENCES

Akpinar, S. (2016) 'Hybrid large neighborhood search algorithm for capacitated vehicle routing problem', *Expert Systems with Applications*. Elsevier Ltd, 61, pp. 28–38. doi: 10.1016/j.eswa.2016.05.023.

Asef-Vaziri, A., Khoshnevis, B. and Rahimi, M. (2008) 'Design and analysis of an automated container handling system in seaports', *International Journal of Agile Systems and Management*. Inderscience Publishers, 3(1–2), pp. 112–126. doi: 10.1504/IJASM.2008.019602.

Azadeh, K., De Koster, R. and Roy, D. (2019) 'Robotized and automated warehouse systems: Review and recent developments', *Transportation Science*, 53(4), pp. 917–945. doi: 10.1287/trsc.2018.0873.

Boysen, N. and Stephan, K. (2016) 'A survey on single crane scheduling in automated storage/retrieval systems', *European Journal of Operational Research*. Elsevier B.V., pp. 691–704. doi: 10.1016/j.ejor.2016.04.008.

Chen, L., Langevin, A., and Riopel, D. (2010) 'The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage', *International Journal of Production Research*. Taylor & Francis Group, 48(4), pp. 991–1011. doi: 10.1080/00207540802506218.

Chung, E. and Lee, H. F. (2008) 'A genetic algorithm for the generalized sequencing problem for automated storage and retrieval systems', *International Journal of Services Operations and Informatics*. Inderscience Publishers, 3(1), pp. 90–106. doi: 10.1504/IJSOI.2008.017707.

Cinar, D. *et al.* (2017) 'Scheduling the truckload operations in automated warehouses with alternative aisles for pallets', *Applied Soft Computing Journal*. Elsevier Ltd, 52, pp. 566–574. doi: 10.1016/j.asoc.2016.10.013.

Eynan, A., Rosenblatt, M. J. and Rosenblatt, M. J. (1993) 'An interleaving policy in automated storage/retrieval systems', *International Journal of Production Research*. Taylor & Francis Group, 31(1), pp. 1–18. doi: 10.1080/00207549308956710.

Gagliardi, J. P., Renaud, J. and Ruiz, A. (2012) 'Models for automated storage and retrieval systems: A literature review', *International Journal of Production Research*. Taylor & Francis Group, 50(24), pp. 7110–7125. doi: 10.1080/00207543.2011.633234.

Guo, X., Yu, Y. and De Koster, R. B. M.

(2016) 'Impact of required storage space on storage policy performance in a unit-load warehouse', *International Journal of Production Research*. Taylor and Francis Ltd., 54(8), pp. 2405–2418. doi: 10.1080/00207543.2015.1083624.

Han, M.-H. *et al.* (1987) 'On Sequencing Retrievals In An Automated Storage/Retrieval System', *IIE Transactions*. Taylor & Francis Group, 19(1), pp. 56–66. doi: 10.1080/07408178708975370.

Heinrich, H. and Willis, E. (2014) 'Automated storage and retrieval system: A time-tested innovation', *Library Management*. Emerald Group Publishing Ltd., 35(6–7), pp. 444–453. doi: 10.1108/LM-09-2013-0086.

Lee, H. F., and Schaefer, S. K. (1996) 'Retrieval sequencing for unit-load automated storage and retrieval systems with multiple openings', *International Journal of Production Research*. Taylor & Francis Group, 34(10), pp. 2943–2962. doi: 10.1080/00207549608905067.

Lee, H. F. and Schaefer, S. K. (1997) 'Sequencing methods for automated storage and retrieval systems with dedicated storage', *Computers & Industrial Engineering*. Pergamon, 32(2), pp. 351–362. doi: 10.1016/S0360-8352(96)00298-7.

Martins De Sá, E., Contreras, I. and Cordeau, J. F. (2015) 'Exact and heuristic algorithms for the design of hub networks with multiple lines', *European Journal of Operational Research*, 246(1), pp. 186–198. doi: 10.1016/j.ejor.2015.04.017.

MELLER, R. D. and MUNGWATTANA, A. (1997) 'Multi-shuttle automated storage/retrieval systems', *IIE Transactions*. Springer, 29(10), pp. 925–938. doi: 10.1023/A:1018592017528.

Murty, K. G. (1968) 'An Algorithm for Ranking all the Assignments in Order of Increasing Cost', *Operations Research*, 16(3), pp. 682–687. doi: 10.1287/opre.16.3.682.

Nalbant, M., Gökkaya, H. and Sur, G. (2007) 'Application of Taguchi method in the optimization of cutting parameters for surface roughness in turning', *Materials and Design*. Elsevier Ltd, 28(4), pp. 1379–1385. doi: 10.1016/j.matdes.2006.01.008.

Popović, D., Vidović, M. and Bjelić, N. (2014) 'Application of genetic algorithms for sequencing of AS/RS with a triple-shuttle module in class-based storage', *Flexible Services and Manufacturing Journal*. Springer New York LLC, 26(3), pp. 432–453. doi: 10.1007/s10696-012-9139-2.

Roodbergen, K. J. and Vis, I. F. A. (2009) 'A survey of literature on automated storage and retrieval systems', *European Journal of Operational Research*. Elsevier, 194(2), pp. 343–362. doi: 10.1016/j.ejor.2008.01.038.

Ropke, S. and Pisinger, D. (2006) 'An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows', *Transportation Science*. INFORMS Inst.for Operations Res.and the Management Sciences, 40(4), pp. 455–472. doi: 10.1287/trsc.1050.0135.

Shaw, P. (1998) 'Using constraint programming and local search methods to solve vehicle routing problems', in *Lecture Notes in*

Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, pp. 417–431. doi: 10.1007/3-540-49481-2_30.

Shell, R., Hall, E. and Parsley, S. (2000) 'Automated Storage and Retrieval Systems', in *Handbook Of Industrial Automation*. doi: 10.1201/9780203908587.ch7.2.

Taguchi, G. (1989) 'Introduction to Quality Engineering: Designing Quality into Products and Processes', *Technometrics*, 31(2), p. 255. doi: 10.2307/1268824.

Tomizawa, N. (1971) 'On some techniques useful for solution of transportation network problems', *Networks*. John Wiley & Sons, Ltd, 1(2), pp. 173–194. doi: 10.1002/net.3230010206.

Warehouse Automation Market - Road to $27B by 2025 (no date). Available at: https://www.thelogisticsiq.com/research/warehouse-automation-market/ (Accessed: 17 July 2020).

Yang, P., Miao, L., Xue, Z. and Qin, L. (2015) 'An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems', *Journal of Intelligent Manufacturing*. Springer New York LLC, 26(6), pp. 1145–1159. doi: 10.1007/s10845-013-0846-7.

Yang, P., Miao, L., Xue, Z. and Ye, B. (2015) 'Variable neighborhood search heuristic for storage location assignment and storage/retrieval scheduling under shared storage in multi-shuttle automated storage/retrieval systems', *Transportation Research Part E: Logistics and Transportation Review*. Elsevier Ltd, 79, pp. 164–177. doi: 10.1016/j.tre.2015.04.009.