Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 39 (2019) 1681–1690

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

25th International Conference on Production Research Manufacturing Innovation: Cyber Physical Manufacturing
August 9-14, 2019 | Chicago, Illinois (USA)

Optimizing Retrieving Performance of an Automated Warehouse for Unconventional Stock Keeping Units

Massimo BERTOLINI, Giovanni ESPOSITO, Davide MEZZOGORI, Mattia NERONI*

University of Parma, Parco Area delle Scienze 181/A, 43100 – Parma, Italy

Abstract

In recent years, the diffusion of automated warehouses in different industrial sectors has fostered the design of more complex automated storages and handling solutions. These circumstances, from a technological point of view, have led to the development of automated warehouses that are very different from the classic pallet Automated Storage and Retrieval Systems (AS/RS), both in terms of design and operating logic. A context in which these solutions have spread is the steel sector. Warehouses with innovative layouts and operating logics have been designed to move metal bundles of different sizes, weights and quality levels, instead of standard, interchangeable stock keeping units. Moreover, picking is often not allowed in these warehouses, due to the configuration of the loading units. In this work we propose a meta-heuristic algorithm based on the Simulated Annealing (SA) procedure, which aims to optimize performance during the retrieving phase of an automated warehouse for metal bundles. The algorithm translates the customers' requests, expressed in terms of item code, quality and weight into a list of jobs. The goal is to optimize the retrieving performance, measured in missions per hour, minimizing the deviations in quality and weight between customer request and the material retrieved. For the validation, a simulation model of an existing warehouse has been created and the performance of the algorithm tested on the simulation model has been compared with the current performance of the warehouse.

Keywords: Automated Storage and Retrieval System; Unconventional Stock Keeping Units; Simulated Annealing; Digital Twin.

* Mattia Neroni. Tel.: +390521905873
E-mail address: mattia.neroni@unipr.it

10.1016/j.promfg.2020.01.272

1. Introduction

In the last years, big exchanges have taken place concerning automated warehouses both in terms of design and application contexts. Many companies, operating in several sectors, such as the steel [1] or automotive [2], decided to opt for automation and automated warehouses have spread into new application contexts. Because of this, there are now Automated Storage and Retrieval Systems (AS/RSs) [3] completely different from classic pallet warehouses on which past research has focused on. These new kinds of automated warehouses are designed for handling unconventional load units, such as cars, incandescent metal bars or metal bar's bundles up to 6 meters long. Furthermore, the diffusion of lean thinking had a big impact, not only in manufacturing, but in logistic too [4]. Because of one-piece-flow production and the concept of stock as inefficiency, purchasing and shipping batches, became increasingly smaller and more varied [5], requiring for the suppliers more shipments per day, a shorter throughput time, and, in general, higher performances [6]. In complex automated warehouses, the possibilities to improve the performance are numerous and can consider one or more aspects together. An aspect which has always been considered in the literature is the design, which has recently been studied by Yang *et al.* [7], in order to optimize the performance for multi-deep AS/RSs under full turnover-based storage policy. Other authors such as Boysen *et al.* [8] focused on sorting and order consolidation processes. However, most of solution proposed concern the policies of routing and scheduling. A recent routing solution based on Travel Salesman Problem (TSP) formulation is presented by Gharehgozli *et al.* [9], while Qing *et al.* [10] adopted a classic Dijkstra algorithm. Finally, Cinar *et al.* [11] described a scheduling approach that models the warehouse as a job shop where loads are considered as jobs, pallets are equivalent to operations and skulls are machines in a shop floor. Other models presented in literature focus on the assignment of the item or location and, recently, Bortolini *et al.* [12] study the assignment optimization in a classic pallet warehouse making a trade-off between retrieval time and energy consumed. This article falls into the category of those works in which the objective is to try to optimize performance by improving the retrieval location assignment, and does it in a context, where picking and sorting are not allowed, and the storage units are long metal bundles.

The remainder of this paper is structured as follows. In section 2 is reported a contextualization to better describe the problem approached in this paper. In section 3 is described the formalization of the problem. The model and the algorithm developed are deeply described in section 4. Case of study and relative results are reported in section 5 and conclusions and future research topics are described in section 6.

Nomenclature

$i = 1, \dots, N$	Retrieved items
$j = 1, \dots, M$	Order lines
p_j	Nominal weight of code required in line j
d_i	Distance between item i and output point which required it
D_j	Total distance to run to fulfil order line j
$D_{max,j}$	Maximum distance to run to fulfil order line j
L	Distance between the output point which requires the retrieve and the furthest item
R	Number of machines required in parallel for retrieving
w_i	Weight of item i
q_i	Quality of item i
m_j	Quantity required by customer in order line j
n_j	Quality required by customer for order line j
g_j	Quantity retrieved to fulfil order line j
k_j	Quality retrieved to fulfil order line j
Δ_{weight}	Acceptable deviation between quantity required and quantity retrieved
$\Delta_{quality}$	Acceptable deviation between quality required and quality retrieved
α	Acceptance coefficient inherent to quantity
β	Acceptance coefficient inherent to quality
t	Iteration of algorithm's era
s	Initial iteration

f	Final iteration
θ	Length of current solution
T_t	Temperature value during iteration t
E	Current era
F_{max}	Maximum value of objective function and fitness of solutions
LB_{weight}, UB_{weight}	Minimum and maximum acceptable quantity
$LB_{quality}$, $UB_{quality}$	Minimum and maximum acceptable quality

2. Contextualization

The proposed algorithm improves the retrieving of metal bundles in an automated warehouse. These bundles are composed of metal bars of different shape and size, but always homogeneous within the same bundle. They are often placed inside the warehouse without the use of containers, but simply placed side by side on shelves. Each bundle is characterized by a unique code (bundles consisting of identical bars, but of different lengths, have different codes, because cutting is not contemplated). A quality index, a nominal weight in relation one-to-one with the code, and a real weight measured during the storage phase, also characterize each bundle. During the retrieving phase, picking and sorting are not allowed, and, once an output point has taken charge of an order, this must be entirely satisfied without observing the next order assigned to the same output point.

Each customer order consists of a list of order lines, each of which holds the following information:

- Item's code required;
- Quantity required expressed in kilograms;
- Quality level of retrieved items;
- Acceptable deviation between required and retrieved quantity;
- Acceptable deviation between required quality and average retrieved quality.

Since picking and sorting are not allowed, the possibilities to satisfy an order line are constrained to the following scenarios:

- Retrieving of a bundle whose code is equal to the one requested and satisfies both the quality and the quantity range;
- Retrieving of more than one bundle, whose codes are equal to the one requested, whose quality levels are constrained within the accepted quality range, and which have the sum of quantities lying within the range of the requested quantity.

If neither of these two scenarios is possible, the order line is then marked as "not-accepted".

3. Formalization

The proposed algorithm is a metaheuristic called Simulated Annealing (SA), originally proposed by Kirkpatrick [13], [14], which is also well known in literature and has already been applied for solving warehouses retrieving problems [15][16][17].

3.1. Quantity and quality bounds

In our case study the quality is expressed with a value ranging from 1 to 10 (1 is the minimum and 10 is the maximum). Typically, acceptable quality and quantity deviations are defined a priori with an agreement between customer and supplier. In our case, they are based on the required value and two parameters, α and β, fixed according to customers' requirements. The acceptable quantity deviation between requested and retrieve values depends on parameter α and bigger is the quantity required, greater is the acceptable margin of error. If the quantity required by customer in order line j is m_j, the Δ_{weight} which defines the acceptable deviation is calculated as follows:

$$\Delta_{weight} = m_j \cdot e^{-\alpha} \tag{1}$$

The upper and lower bounds are calculated as follows:

$$LB_{weight} = m_j - \Delta_{weight} \tag{2}$$

$$UB_{weight} = m_j + \Delta_{weight} \tag{3}$$

Even for the quality, being n_j the quality level required by customer, the acceptable quality deviation $\Delta_{quality}$ is calculated as follows:

$$\Delta_{quality} = n_j \cdot e^{-\alpha} \tag{4}$$

In our application case, it is not possible to supply goods with a quality level lower than the required one, therefore, while upper bound $UB_{quality}$ is calculated exactly as for the quantity, the lower bound $LB_{quality}$ is equal to quality required n_j. Figure 1 shows the trend of lower and upper bound for quantity (a) and quality (b) in a specific case in which α=β=1.5 are presented. In both the graphs the upper bound is represented in orange and the lower bound in blue.

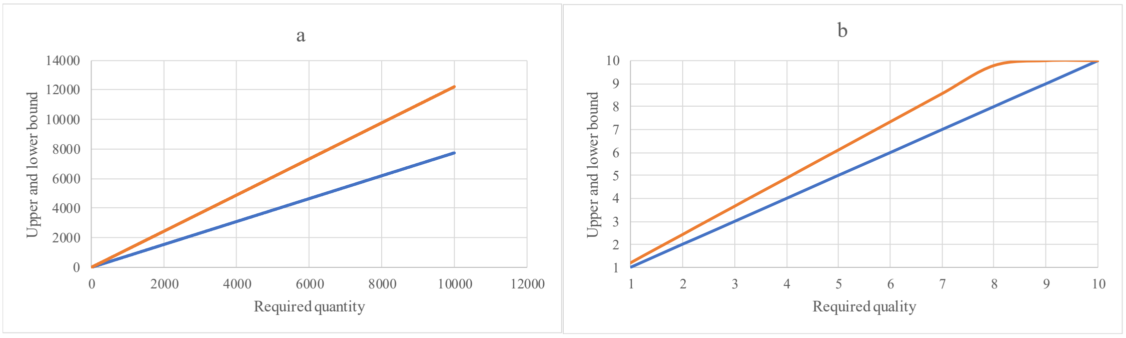

Fig. 1. (a) Quantity's upper and lower bound for increasing quantity required; (b) Quality's upper and lower bound for increasing quality required

3.2. Solution

Give an order to fulfill, expressed as a list of M order lines $[L_1 ... L_M]$, a solution is represented as a list of M elements $[E_1 ... E_M]$ named *sub-solution*, each of which satisfies the corresponding order line. Each element E_j can therefore represent of one or more bundles to retrieve.

3.3. Temperature

In SA, temperature value defines the acceptance probability of a new solution worse than the current one and indirectly defines the size of explored neighborhood. At the beginning of each era, the temperature is brought back to the initial level selected and then decreases with the increase in iterations till it reaches the final value. The temperature and its reduction define the number of solutions explored by the algorithm and acceptance threshold of worse solutions and the breadth size of neighborhood analyzed depend on its value.

3.4. Cooling schedule

The cooling schedule is the trend with which temperature decreases as the number of iterations carried out increases. Every iteration t, temperature value is calculated using the formula proposed by Lai and Chan [18]:

$$T_t = \frac{T_{t-1}}{1 + T_{t-1} \cdot B} \tag{5}$$

In Eq. (5), t-1 represents the previous iteration and B is calculated, according to Lai and Chan [18], as follows:

$$B = \frac{T_s - T_f}{T_s \cdot T_f \cdot E} \tag{6}$$

where T_s is the starting temperature, T_f is the ending temperature and E is the eras' number.

3.5. Initial and final temperature

Initial and final temperature are set accordingly to Kirkpatrick *et al.* [19]. Specifically, it is chosen to assure a high initial acceptance probability and a low ending acceptance probability. Fixed F_{max} the highest (best) of fitness that can characterize a solution and $a_{\%}$ the desired acceptance threshold, the temperature is calculated using equation proposed by Bertolini *et al.* [20]:

$$T = \frac{-F_{max}}{ln(a_{\%})} \tag{7}$$

3.6. Fitness

The fitness function, which the SA tries to optimize, takes into consideration three elements represented by three different indexes:

- Q represents the quality delta between customer order request and planned order retrieving;
- W represents the quantity delta between customer order request and planned order retrieving;
- T represents the estimated retrieving time.

Each component is designed to take value ranging between 0 and 1, and each of them is assigned a weight coefficient ε, φ and μ whose values have been empirically defined ($\varepsilon = 0.35$, $\varphi = 0.2$, $\mu = 0.45$) to match the case study expectations. The fitness function is thus expressed as follows:

$$Fitness = \varepsilon \cdot Q + \varphi \cdot W + \mu \cdot T \tag{8}$$

Q grows for decreasing difference between quality required and quality retrieved. Fixed $j = 1 \dots M$ order lines, the whole order quality index Q is calculated as sum of lines' quality indexes Q_j. Moreover, being i=1, …, N the retrieved bundles to fulfil a generical order line j and q_i the quality level of each of them, the quality value of the sub-solution is the following:

$$k_j = \frac{\sum_{i=1}^{N} q_i}{N} \tag{9}$$

Finally, given n_j the quality required by order line j, the quality index of correspondent sub-solution Q_j is calculated as follows:

$$Q_j = 1 - \frac{|k_j - n_j|}{0.5 \cdot (UB_{quality} - LB_{quality})} \tag{10}$$

Even the quantity index W grows for decreasing difference between quantity required and quantity retrieved. Fixed j=1, …, M order lines and being W_j the quantity index of solution found for line j, the quantity index of the whole order W is calculated as sum of lines' quantity indexes. Moreover, defined i=1, …, N the retrieved bundles to fulfil order line j and w_i the weight of each of them, the quantity retrieved g_j is the following:

$$g_j = \sum_{i=1}^{N} w_i \tag{11}$$

Given m_j quantity required by order line j, the quantity index of correspondent sub-solution W_j is calculated as follows:

$$W_j = 1 - \frac{|g_j - m_j|}{0.5 \cdot (UB_{weight} - LB_{weight})} \tag{12}$$

The calculation of the time index considers the distance D_j between retrieved items and output point requiring them. It also takes into account the number of required cranes R simultaneously involved in the handling tasks (considering that each crane is able to reach just specific areas of warehouse). Concretely, the increase in the number of cranes used in parallel reduces the retrieval time by increasing the number of activities performed in parallel. The distance D_j is calculated as the distance between the output point and the item to be retrieved, multiplied by 2, to predict the worst situation in which the crane has just finished the previous mission. The travel time has not an upper bound and, in order to obtain a value between 0 and 1, the maximum distance that could be covered is calculated beforehand. Given L the distance between the output point and the furthest item, m_j the quantity required by the order line j, and p_j the nominal weight of required code, the maximum distance $D_{max,j}$ is calculated with the following function:

$$D_{max,j} = 2 \cdot L \cdot \frac{m_j}{p_j} \tag{13}$$

Where m_j/p_j is a number indicative of travels needed to fulfill the order line. This pessimistic value is compared with the real distance to be covered, that, given i = 1, …, N bundles selected to fulfill the order line and d_i the distance between bundle i and the output point, is calculated as follows:

$$D_j = \sum_{i=1}^{N} d_i \tag{14}$$

Finally, the time index T_j of a generical sub-solution is calculated as follows:

$$T_j = 1 - \frac{D_j}{D_{max,j} \cdot R} \tag{15}$$

Even in this case, the time index of the whole solution T is the sum sub-solutions' indexes.

3.7. Threshold

In the simulated annealing procedure, at each iteration a new solution is generated from the current working solution neighborhood. The procedure, while it aims to obtain at each iteration a new solution providing a better fitness function value, also considers the possibility to temporarily accept a worsening solution, so as to escape, in following iterations, a local minimum (optima). This behavior is managed using a probability threshold, which describe the probability to accept a new solution even if worse than the current one and is calculated using the following formula frequently used in literature:

$$Threshold = e^{-\Delta F / T_t} \tag{16}$$

Where T_t represents the current temperature and ΔF the difference between the fitness of two compared solutions. An example threshold's trend for ΔF equal to 1 is represented in Figure 2.

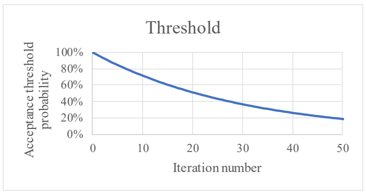

Fig. 2. Trend of threshold value when number of iterations increases

3.8. Anchor

Every time a new solution is generated adopting a neighbor search, the anchor defines the exact number of elements (bundles) to change in that iteration. Fixed θ the number of items, which compose the current solution and T_t the temperature, the anchor is calculated by using the following function proposed by Lai and Chan [18]:

$$Anchor = RoundUp\left(\theta \cdot e^{-1/T_t}\right) \quad (17)$$

4. Algorithm

4.1. Macro-procedure

The macro-procedure is represented in Figure 3. Each era a new random solution is generated, accepted and made current solution. Temperature is set equal to beginning value defined a priori. Each iteration temperature value is updated, a new solution is generated and rated: if it is accepted it becomes the new current solution. Procedure is repeated till temperature reaches the final value defined a priori. Then, if the maximum number of eras has been reached, procedure ends and return the best solution found, otherwise era's number is updated and the whole procedure is repeated.

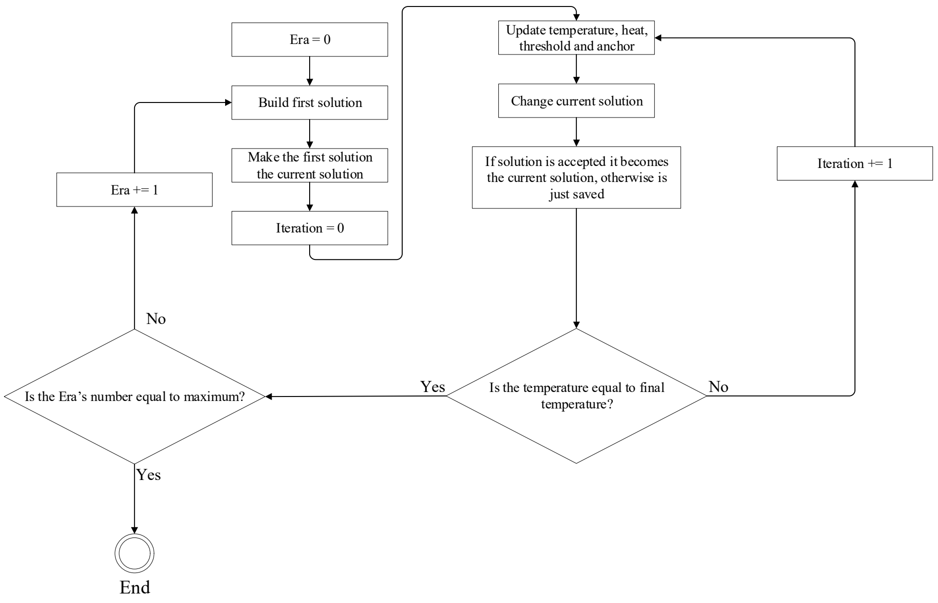
Era = 0
Build first solution
Make the first solution the current solution
Iteration = 0
Update temperature, heat, threshold and anchor
Change current solution
If solution is accepted it becomes the current solution, otherwise is just saved
Iteration += 1
Era += 1
No
Is the Era's number equal to maximum?
Yes
Is the temperature equal to final temperature?
Yes
No
End

Fig. 3. Macro-procedure of the algorithm

4.2. Initial solution's creation

The procedure for constructing the initial solution is sketched in Figure 4. Within each solution there are some sub-solutions. Each sub-solution refers to an order line and has the objective of fulfilling it as well as possible. There are two ways to fulfill an order line:

- matching: the request is fulfilled with just one bundle;
- filling: the request is fulfilled with a list of bundles.

A sub-solution and its respective order line are accepted if they have at least one item that can be used in matching, or a list of items that can be used in filling such that the sum of their weights is greater than the LB_{weight}. Matching always takes precedence. In this way, if an order line can be completed with a single item, the possibility of fulfilling it with several items is not considered, because, generally, retrieving more items reduces retrieving performances. If matching is not possible the *filling procedure* described in Figure 5 is implemented. However, it must be noted that using the filling procedure can result in a deadlock situation whenever the order line results satisfiable with the current stock, and the addition of any available item, would violate the weight upper bound allowed. In this case, an additional previous step is applied, i.e. the elimination of the heaviest item from sub-solution, before repeating the filling procedure; alternatively, the matching procedure is used, if possible. This is repeated at most a number of times equal to length of feasible items' list.

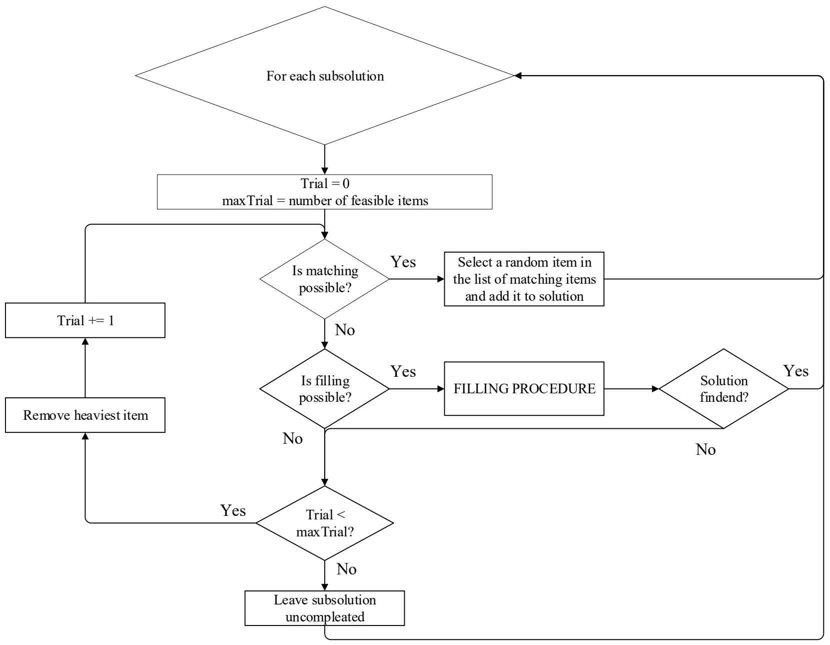

Fig. 4. Detailed procedure of the algorithm

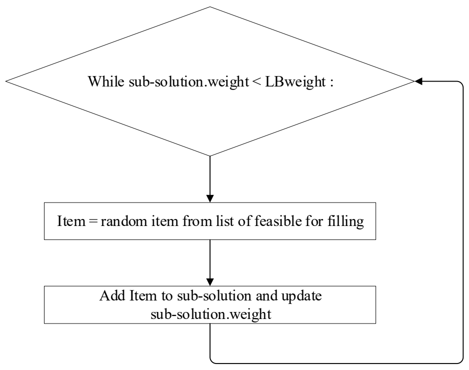
While sub-solution.weight < LBweight :
Item = random item from list of feasible for filling
Add Item to sub-solution and update sub-solution.weight

Fig. 5. Representation of filling procedure

4.3. Neighborhood search

Finally, the current solution's change is performed exactly as the initial solution's creation but is preceded by the random deletion of an anchor number of items in the current solution.

5. Results

To test the algorithm, a simulations model of an existing warehouse was created in Python 3.6 using the Simpy module, and simulation was run on an Intel-based computer with 3.60 GHz and 8 Gb of RAM. The warehouse consists of two distinct storage areas served by two different cranes, one output point and one input point connected to the cranes by shuttles and elevators. The algorithm was tested comparing the simulation model with the real system over two weeks of work characterized by high peak of requests and shifts of 8 hours per day, during which both input and output missions were performed. The warehouse's filling at the beginning of test was 63% and three different types of codes of different length and nominal weight are managed. The parameters α and β were both equal to 1.5, and the number of eras were set equal to 30. Results are reported in Table 1 and show how the proposed procedure correctly ensure better performance and less difference between quality or quantity required and retrieved.

Table 1. Results and comparison between algorithm developed and algorithm currently implemented.

Comparison value	Units of measurement	Proposed algorithm	Currently implemented algorithm
Working hours simulated	hour	80	80
Missions completed	number	2,243	2,018
Missions per hour	$^{number}/_{hour}$	28	25
Average computation time	seconds	1.526	0.039
Average quality difference	%	0.72	3.07
Average quantity difference	%	8.96	14.06

6. Conclusion

In this paper is presented an algorithm based on simulated annealing procedure for the optimization of the retrieving performances in a warehouse for metal bundles of different lengths, weight and quality indexes. Tests show promising results, outperforming the currently implemented algorithm operating in the real warehouse here simulated; moreover, this work aims to directly optimize performances with respect to quantity and quality constraints, which is little explored in literature. Future research topics will consist in the application of a similar algorithm to more complex warehouses where the storage compartments are double depth (allowing the joint removal of two bundles) or where two cranes share the same railway obstructing each other. Another development will consist in the creation of an algorithm for the scheduling of the jobs returned by the one described in this paper, to further optimize the performance finding the optimal routing.

References

[1] Salamov, Warehouse logistics, Russ. J. Non-Ferrous Met., 9 (2019).
[2] Spassov, Automated warehouses and parkings for cars with stacker cranes, International Body Engineering Conference and Exhibition and Automotive and Transportation Technology Conference, (2002).
[3] Jan and Vis, A survey of literature on automated storage, Eur. J. Oper. Res., 194 (2009) 343–362.
[4] Aslam, Farruckh, Gardezi, and Hayat, Design, development and analysis of automated storage and retrieval system with single and dual command dispatching using MATLAB, World Acad. Sci. Eng. Technol., 36 (2009) 91–95.
[5] P. Hines, M. Holweg, and N. Rich, Learning to evolve A review of contemporary lean thinking, Int. J. Oper. Prod. Manag., (2004).
[6] N. Boysen, R. De Koster, and F. Weidinger, Warehousing in the e-commerce era : A survey, Eur. J. Oper. Res., vol. 1 (2018) 1–16.
[7] P. Yang, K. Yang, M. Qi, L. Miao, and B. Ye, Designing the optimal multi-deep AS / RS storage rack under full turnover-based storage policy based on non-approximate speed model of S / R machine, Transp. Res. Part E, 104 (2017) 113–130.
[8] N. Boysen, S. Fedtke, and F. Weidinger, Optimizing automated sorting in warehouses : The minimum order spread sequencing problem, Eur. J. Oper. Res., 270 (2018) 386–400.
[9] Gharehgozli, Yu, Zhang, and D. Koster, Polynomial time algorithms to minimize total travel time in a two-depot automated storage/retrieval system, Transp. Sci., 51 (2017) 19–33.
[10] Qing, Zheng, and Yue, Path-planning of automated guided vehicle based on improved Dijkstra algorithm, 29th Chinese Control and Decision Conference, (2017) 7138–7143.
[11] D. Cinar, J. António, Y. I. Topcu, and P. M. Pardalos, Scheduling the truckload operations in automated warehouses with alternative aisles for pallets, Appl. Soft Comput. J., 52 (2017) 566–574.
[12] M. Bortolini, M. Faccio, E. Ferrari, M. Gamberi, and F. Pilati, Time and energy optimal unit-load assignment for automatic S / R warehouses, Intern. J. Prod. Econ., 190 (2017) 133–143.
[13] S. S. Heragu, B. Mazacioglu, and K. Fuerst, Meta-heuristic algorithms for the order picking problem, Int. J. Ind. Eng., 1 (1994) 67–76.
[14] E. Atmaca and A. Ozturk, Defining order picking policy. A storage assignment model and a simulated annealing solution in AS / RS systems, Appl. Math. Model., 37 (2013) 5069–5079.
[15] H. Nadir, B. Zaki, and S. Latéfa, Metaheuristic based control of a flow rack automated storage retrieval system, J. Intell. Manuf., 23 (2012) 1157–1166.
[16] Bian, Dai, Cao, and Chang, Research on path planning of stacker based on improved simulated annealing algorithm, Pap. Asia, 1 (2018) 106–110.
[17] Yang, Miao, and Qin, Job scheduling optimization in multi-shuttle automated storage and retrieval system, Jisuanji Jicheng Zhizao Xitong/Computer Integr. Manuf. Syst. CIMS, 19 (2013) 1626–1623.
[18] K. K. Lai and J. W. M. Chan, Developing a simulated annealing algorithm for the cutting stock problem, Comput. Ind. Eng., 32 (1997) 115–127.
[19] Kirkpatrick, Gelatt, and Vecchi, Optimization by simulated annealing, Science, 220, (1983) 671–680.
[20] M. Bertolini, D. Mezzogori, and F. Zammori, Comparison of new metaheuristics, for the solution of an integrated jobs-maintenance scheduling problem, 122 (2019) 118–136.