Paper 028_OCTAVIA_ICAMME21

by Tanti Octavia

Submission date: 03-Apr-2023 08:04PM (UTC+0700)
Submission ID: 2054581617
File name: 028_OCTAVIA_ICAMME21.pdf (471.47K)
Word count: 3989
Character count: 18360

Optimizing Travel Distance for Picking Order Problem Using Symbiotic Organism Search Algorithm Combined with Mutation Operators

Tanti Octavia [1, a)], Andreas Handojo [2, b)], and Adelyn Thungriallu [2, c)]

[1]*Industrial Engineering Department, Petra Christian University,*
Jl. Siwalankerto 121–131, Surabaya 60236, East Java, Indonesia
[2]*Informatics Department, Petra Christian University ,*
Siwalankerto 121–131, Surabaya 60236, East Java, Indonesia

[a)] Corresponding author: tanti@petra.ac.id
[b)] handojo@petra.ac.id, [c)] adelynthung@gmail.com

Abstract. This article aims to investigate the reliability of sos algorithm for solving the picking order problems considering the real environments of warehouse. We attempt to apply SOS algorithm combined with mutation operators. There are three mutation operators used, namely swap mutation operator, inversion mutation operator, and insertion operator. Simulation is carried out using a case study of warehouse's company that stores various kinds of daily products to fulfil the customer demand. The simulation is run considering the percentage decrease in the distance between initial solution and final solution and the computational time. Simulation is run for the number of iterations of 100, 500, and 1000 and the ecosystem size as 10,20, and 50. The results show the Symbiotic Organism Search (SOS) algorithm can provide a large percentage of distance reduction for a small number of consumer demand in all iterations and ecosystem sizes, with a percentage decrease in distance of more 5%. As for the large number of customer demand (500 and 1000), the percentage decrease in the total distance is below or equal to 3.71% for all iterations.

INTRODUCTION

Warehouse management is the process, control, and optimization of warehouse operations from the entry of inventory into a warehouse. In a company. Warehouse operations have a strong influence on direct operational costs of warehouse. One of operations in warehouse that causes 60-70% of the total operating process is order picking [1]. Picking involves five actions: pre-action, picking, searching, transport, and others. Among these components, the transport time is of outstanding importance, since it consumes the major proportion (at least 50%) of the total processing time [2].

Many studies have been conducted to develop optimization methods using mathematical modelling to solve picking order problems. Cano *et al*. [3] introduce mathematical programming models for the joint order batching, sequencing, and picker routing problem. meanwhile, Scholz et al developed a mathematical programming formulation for the single picker problem in block layout [4]. However, this mathematical approach is often not sufficient to solve large-scale problems while the heuristic search method often has problems with low the quality of the resulting solution due to being trapped at local optima. This encourages increased efforts to develop more accurate optimization methods and efficient.

In recent decades, there has been a growing research interest in meta-heuristics for solving picking order problems. This aims to lead picking orders in company toward an efficient and transportation time. Henn and Wascher proposed two approaches based on tabu search principle to solve order batching problem in warehouse to operate more efficiently [5]. Tsai et al. attempts to propose a batch picking model that considers not only travel cost but also an earliness and tardiness penalty to fulfil the current complex and quick-response oriented environment [6]. Hsien Pan et al developed

an order batching approach based on a group genetic algorithm to balance the workload of each picking zone and minimize the number of batches in a pick-and-pass system to improve system performance [7].

Metaheuristic algorithms have an inspired solution search scheme from developed nature acts as a source of concepts, the mechanisms, and principle. Such metaheuristics include simulated annealing, genetic algorithms, ant colony optimization and particle swarm optimization. A proposed a new routing algorithm based on Ant Colony Optimization (ACO) for two order pickers (A-TOP) with congestion consideration is developed by Chen et al. [8]. Meanwhile, De Santis et al. introduced the combination algorithm between Ant Colony Optimization Algorithm and Floyd Warshall Algorithm for optimizing the pickers' routing in warehouses [9]. Cergibozan and Tasan have developed fast and effective metaheuristic approaches to solve the order batching problem using a proposed two GA based metaheuristic approaches [10]. Botanni et al. [11] tested on twenty-five scenarios, resulting from variable length of the order pick lists and different manual storage configuration. The authors stated an adapted Harmony Search (HS) gives the better solution in term of travel distance for low level picker to part systems.

In recent years, a new metaheuristic algorithm called symbiotic organisms search (SOS) attracted the attention of because of its considerable ability good at solving some problems complex optimization, such as: time–cost–labor utilization trade off problem [12]. SOS adopts interaction pattern common symbiosis between living things life. In maintaining their survival, living things will interact with other living things In a symbiotic form. It is through this interaction that living things can improve their quality of life to be able to survive. Inspired by phenomena in this symbiosis, the SOS algorithm uses three main phases are mutualism, commensalism, and parasitism, to find the best solution effective and efficient.

In this paper, we propose SOS Algorithm combined with mutation operators for optimizing travel distance for picking order problem. The remainder of this paper is organized as follows. We briefly explain the research method. Using SOS algorithm combined with mutation operators, we conduct a simulation and discuss the result of it through computational study. Finally, we summarize the experimental results

METHODS

Symbiotic Organism Search Algorithm

SOS Algorithm adopts the symbiotic interaction of organisms to survive and reproduce in the ecosystem, namely Mutualism, Commensalism, and Parasitism. SOS Algorithm starts by initializing an ecosystem consisting of eco size organisms [13]. Next is the selection of *Xbest* from all existing organisms. Iterations of *i* are carried out by referring to the criteria for repeating the results that have been set, for example if *Xbest* does not change as much as *n* then the repetition is complete. This loop is called the outer loop. Furthermore, eco size is repeated as an inner loop to enter the stages of Mutualism, Commensalism, and Parasitism. The result of the inner-loop is the *Xbest* update, and the result of the outer-loop is the final result or route based on the last *Xbest*. The flowchart of SOS Algorithm can be seen in Fig. 1. The Mutualism phase will take X_i and X_j to compare with the results of the new X_i and X_j. X_i is the iteration of organisms in the ecosystem, while X_j is the random result of organisms in the ecosystem. Furthermore, X_i and X_j will be updated if the results of the new X_i and X_j are better than the old ones. Here are the similarities in the Mutualism phase:

$$X_{inew} = X_i + rand(0,1) * X_{best} - Mutual\_Vector * \mathrm{BF}_1 \quad (1)$$

$$X_{jnew} = X_j + rand(0,1) * X_{best} - Mutual\_Vector * \mathrm{BF}_2 \quad (2)$$

$$Mutual\_Vector = (X_i + X_j)/2 \quad (3)$$

The Commensalism phase is a continuation of the Mutualism phase. X_i which is the output of the Mutualism phase will be the input for this phase. X_j is also taken as a random result of organisms in the ecosystem. X_j will be used as a factor affecting the survival of the organism X_i. X_i will be updated if the results of the new X_i are better than the previous X_i. Here is the equation for the Commensalism phase:

$$X_{inew} = X_i + rand(-1,1) * (X_{best} - X_j) \quad (4)$$

The Parasitism phase is the last phase, where the Parasite vector will be given to X_i to try to replace X_j in the ecosystem. If the Parasite Vector has better results, then the Parasite Vector will take position X_j in the ecosystem. Whereas it is worse, the Parasite Vector will not be added to the ecosystem.

$$Parasite_vector = \begin{cases} O_{i,k} & k < rand(0,1), k = 1,2,3 \ldots, nvar \\ the\ random\ position & lainnya \end{cases} \quad (5)$$

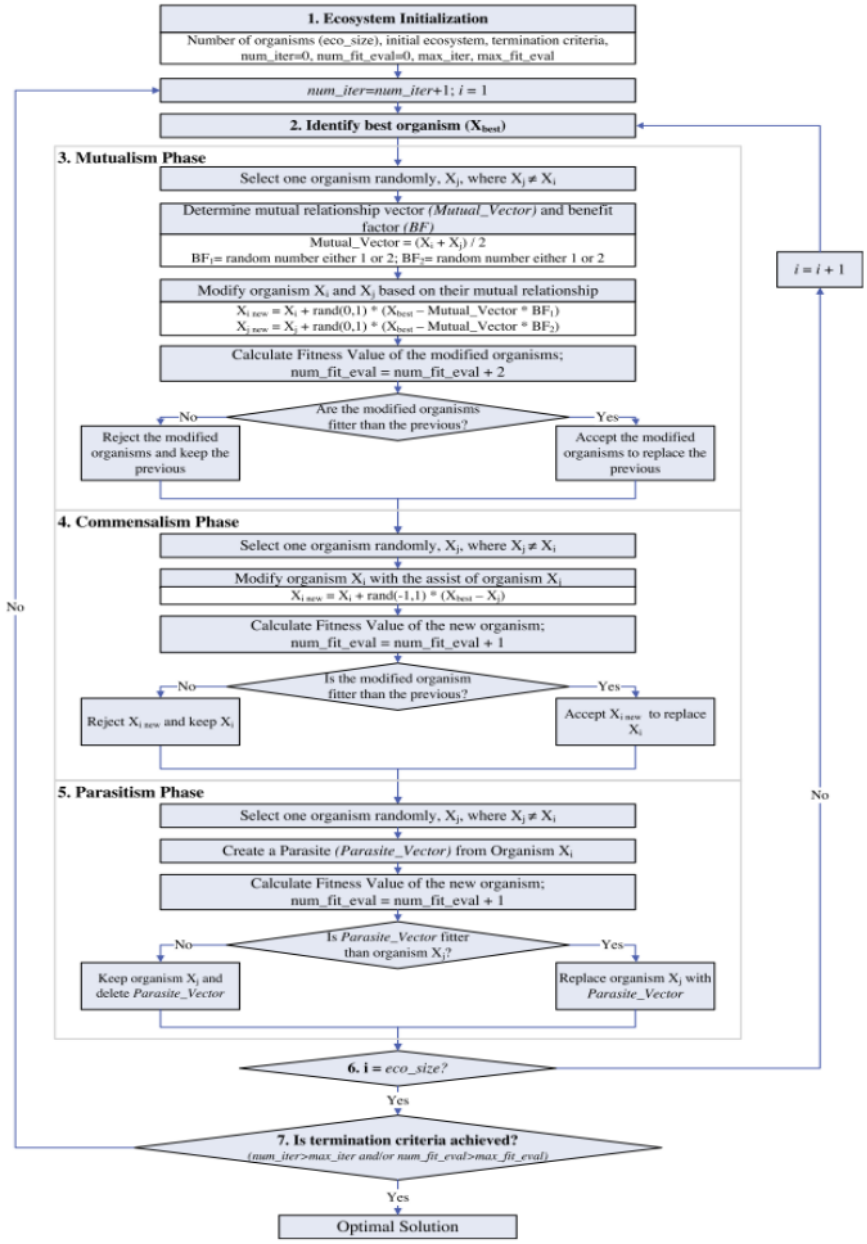

FIGURE 1. The flowchart of SOS algorithm [12]

Mutation Operators

In this research, there are three stages that need to be done before entering the phase of Mutualism, Commensalism and Parasitism. The three stages are Swap Mutation Operator, Inversion Mutation Operator, and Insertion Operator [14]. The three stages are as follows:

1. Swap Mutation Operator: At this stage (Fig. 2), the position of 2 random data is exchanged in a solution vector. Select *i* and *j* randomly, then the values of the positions *i* and *j* are exchanged to form the new solution vector which is then compared with the previous solution vector. If it turns out that the new solution vector is better than the current one, the current solution vector is replaced with a new solution vector.

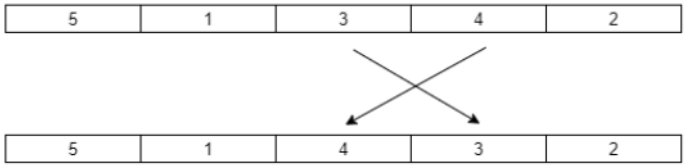

FIGURE 2. Swab mutation operator

2. Inversion Mutation Operator: At this stage (Fig. 3), the solution vector data is inverted. choose *i* and *j* at random, then a new solution vector is created by inverting from index to *i* to *j*. After that the new solution vector is compared with the old one. If the new solution vector is better, then the current solution vector is replaced with the new one.

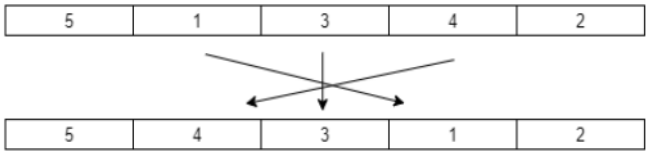

FIGURE 3. Inversion mutation operator

3. exhaustion Insertion Operator: At this stage (Fig. 4), the solution vector data is inserted into the new index. First, *i* and *j* are selected randomly. After that, a new solution vector is created by taking data from index *i* and inserting it into index *j*.

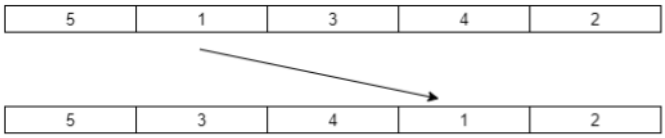

FIGURE 4. Insertion operator

RESULTS AND DISCUSSION

This study will investigate the ability of SOS in solving order picking problems. It will be investigated using a case study of warehouse's company that stores various kinds of daily products to fulfil the customer. The warehouse located in Manado, North Sulawesi-Indonesia. Some of the data needed to solve this picking order problem are product type, demand data for each product, warehouse area, number of available shelves, location of each product on the shelf, the distance between shelves, and the location of each shelf. Warehouse layout can be seen in Fig. 5.

Several stages are carried out in conducting the simulation, namely:

1. Create a picking order scenario with reference to the activities and problems of picking up goods in the warehouse. In this simulation, Data for number of customer demand that will be used are 100, 500, and 1000 to evaluate the reliability of the SOS algorithm.

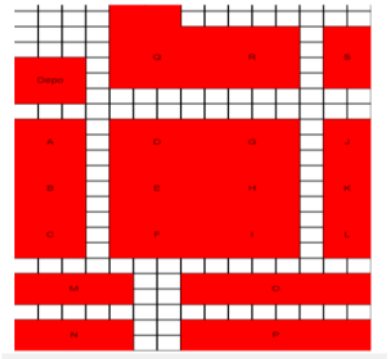

FIGURE 5. Warehouse layout

2. Create the simulation model and verify it with manual calculations before simulation is run. In this model there are 5 consumer orders that need to be fulfilled by the warehouse. The product item data, the number of items, the total weight, the distance between the shelves, as well as the shelf code of the product are known.
   2.1. The route sequence will initially be generated using random numbers. The random numbers are then converted into a sequence of shelves as a route called the solution vector. The number of route sequences is the specified eco_size. In this case, the eco_size used is 5 (Table 1). Each solution vector has a mileage starting at the depot and ending at the depot (Table 2). In the Table 3, we get the random number of X_i is 4-3-2-1-5. It means the picker will take the sequence of demand and the route will be Depot-shelf H-shelf F-shel C-shelf B-shelf K-Depot with the total distance 90.

TABLE 1. Demand information

Item	Qty (In pieces)	Weighted (Grams)	Shelf
1	36	4320	B
2	36	4320	C
3	36	2736	F
4	36	1980	H
5	6	5400	K

TABLE 2. The distance among Shelves and Depot

Shelf	Depot	B	C	F	H	K
Depot	0	7	10	10	26	26
B	7	0	3	3	19	19
C	10	3	0	0	16	16
F	10	3	0	0	16	16
H	26	19	16	16	0	0
K	26	19	16	16	0	0

TABLE 3. The distance among Shelves and Depot

	Vector Solution					Dist
x1	0.5962 4	0.3353 3	0.1590 2	0.0551 1	0.9074 5	90
x2	0.7937 4	0.4876 2	0.5961 3	0.3280 1	0.8835 5	90
x3	0.1752 1	0.9231 5	0.4538 2	0.4785 3	0.9103 4	84
x4	0.1545 1	0.6392 4	0.5259 3	0.7792 5	0.3591 2	84
x5	0.568 5	0.4341 4	0.3767 3	0.1856 1	0.2628 2	58

2.2. Find the *Xbest* that gives the smallest distance from solution vector and run the number of iterations and performs the Swap Mutation Operator, Inversion Mutation Operator, and Insertion Operator stages on the *Xbest* solution vector.

- In the Swap Mutation Operator stage, select *i* and *j* randomly. Then, the values of index *i* and *j* are swapped to form a new solution vector. If the new solution vector is better, the old solution vector will be replaced with the new one. In the calculation, *i* and *j* obtained are 2 and 3. *Xbest* and the new solution vector can be seen in Table 4. Since the new solution vector is the same as the current one, the *Xbest* solution vector is not replaced.
- In the Inversion Mutation Operator stage, choose *i* and *j* at random. Then, the values from index *i* to *j* are inverted to form a new solution vector. If the new solution vector is better, the current solution vector will be replaced with the new one. In the calculation, *i* and *j* obtained are 0 and 2. *Xbest* and the new solution vector can be seen in Table 5. Since the new solution vector turned out to be better, the *Xbest* solution vector was replaced with a new one.

TABLE 4. The swap mutation operator result of vector solution

	Vector Solution					Dist
x5	0.5688 5	0.4341 4	0.1856 1	0.3767 3	0.2628 2	58
x5 new	0.5688 5	0.4341 4	0.3767 3	0.1856 1	0.2628 2	58

TABLE 5. The inversion mutation operator result of vector solution

	Vector Solution					Dist
x5	0.1856 1	0.4341 4	0.5688 5	0.1856 3	0.2628 2	58
x5 new	0.5688 5	0.4341 4	0.1856 1	0.3767 3	0.2628 2	52

- In the Insertion Operator stage, choose *i* and *j* randomly. Then, the value of index *i* is taken and inserted into index *j* to form a new solution vector. If the new solution vector is better, the current solution vector will be replaced with the new one. In the calculation, *i* and *j* obtained are 0 and 4. *Xbest* and the new solution vector can be seen in Table 6. Since the new solution vector is found to be equal to *Xbest*, the *Xbest* solution vector is not replaced.

TABLE 6. The inversion mutation operator result of vector solution

	Vector Solution					Dist
x5	0.1856 1	0.4341 4	0.5688 5	0.1856 3	0.2628 2	52
x5 new	0.4341 4	0.5688 5	0.1856 3	0.2628 2	0.1856 1	52

2.3. In a number of iterations, the iteration operation is repeated for the number of vectors or eco_size specified through the stages of Mutualism, Commensalism, and Parasitism.

- In the Mutualism stage (Table 7), there is a vector xi according to the iteration and one random solution vector is chosen as X_j to be operated. In this calculation, X_i is X_1 and the selected X_j is X_4.

TABLE 7. The result of mutualism stage

	Vector Solution					Dist
x1	0.5962 4	0.3353 3	0.1590 2	0.0551 1	0.9074 5	90
x4	0.0551 1	0.5962 4	0.3353 3	0.9074 5	0.1590 2	84
x1 new	0.1590 2	9074 5	0.3353 3	0.5962 4	0.0551 1	71
x4 new	0.0551 1	0.3353 3	0.5962 4	0.9074 5	0.1590 2	52

- In commensalism stage, komensalisme stage, which is expressed as the following equation:

$$X_{1new} = X_1 + rand(-1,1) * (X_{best} - X_3)$$

In this calculation, X_i *is* X_1 and X_j is X_3. From the above equation, the calculation results are obtained as in Table 8. If *the new* X_i is better than X_i, then X_i is replaced with *the new* X_i. Since the calculation above is not better than X_i, the solution vector X_i is not replaced.

- In Parasitism stage, firstly choose X_j from the solution vector. After that, a Parasite Vector was created based on the solution vector X_i and compared with X_j. If Parasite Vector is better, then X_j will be replaced by Parasite Vector. In this calculation, X_i is X_1 and X_j is X_2, and the results can be seen in Table 9.

TABLE 8. The result of commensalism stage

	Vector Solution					Dist
x1	0.15902	0.90745	0.33533	0.59624	0.05511	71
x3	0.17521	0.92315	0.45382	0.47853	0.91034	84
x1 new	0.59624	0.05511	0.15902	0.59623	0.90745	98

TABLE 9. The result of parasitism stage

	Vector Solution					Dist
Parasite Vector	0.03112	0.33535	0.04293	0.05514	-0.0961	71
x2	0.79374	0.48762	0.59613	0.32801	0.88355	90

2.4. After doing all the steps, update *Xbest* according to the best Solution Vector which has the shortest distance. Repeat until iterations are reached. The final results obtained can be seen in Table 10.

TABLE 10. The final result

	Vector Solution					Dist
x1	0.15902	0.90745	0.33533	0.59624	0.05511	71
x2	0.15902	0.90745	0.33533	0.59624	0.05511	71
x3	0.17521	0.92315	0.45382	0.47853	0.91034	84
x4	-0.03511	0.63923	0.75234	0.77925	-0.01132	52
x5	0.18561	0.43414	0.56885	0.37673	0.26282	58

3. Run a simulation considering the number of iterations and the ecosystem size. The iterations used are 100, 500, and 1000 iterations, while the ecosystem sizes used are 10, 20, and 50. The distance decreasing based on iterations 100, 500 and 1000 SOS algorithm for each number of customer demand (100, 500, 1000) with eco_size= 50 can be seen in Table 11.

TABLE 11. The distance reduction based on iterations

Number of Iterations	Number of Customer Demand	Distance (First Iteration)	The lowest Distance (Final Iteration)	Reduction (%)	Average Reduction (%)
100	100	1688	1596	5.45	2.16
	500	9154	9132	0.24	
	1000	18742	18594	0.79	
500	100	1718	1612	6.17	3.77
	500	9338	8992	3.71	
	1000	18692	18424	1.43	
1000	100	1694	1588	6.26	3.01
	500	9232	9058	1.99	
	1000	18622	18454	0.92	

The result (Table 12) shows the largest percentage of distance reduction is in 100 iterations. This means that for small number of customer demand, the SOS algorithm can provide a good solution. The percentage reduction in

distance for a small number of customer demand (100) by more than 5%. As for the large number of customer demand (500 and 1000), the percentage decrease in the total distance is below or equal to 3.71% for all iterations. It also shows that the more iterations performed for the same amount of data demand, the greater the percentage decrease in distance.

TABLE 12. The distance reduction based on ecosystem size

Eco-Size	Number of Customer Demand	Distance (First Iteration)	The Lowest Distance (Final Iteration)	Reduction (%)	Average Reduction (%)
10	100	1744	1634	6.31	3.97
	500	9376	9042	3.56	
	1000	18810	18428	2.03	
20	100	1730	1592	7.98	4.18
	500	9332	8930	4.31	
	1000	18562	18510	0.28	
50	100	1694	1588	6.26	3.01
	500	9232	9058	1.89	
	1000	18622	18454	0.90	

From the Table 13, the largest percentage of decline in distance is in the small number of customer demand. Meanwhile, for the large number of customer demands (500 and 1000), the percentage decrease in distance is below or equal to 4.31%. SOS algorithm computation time will increase along with the increase in the amount of data, eco_size, and iterations. In addition, it can be seen that with a large number of iterations and eco_size, it does not guarantee that the results obtained by SOS will be better. It is interesting to note that for a large number of customer demands, it would be better to use eco-size 50 with 500 iterations because it gives a high percentage of distance reduction with a computational time as 1158 seconds (19.3 minutes).

Table 13. Computational time of SOS algorithm's simulation

n-data	100			500			10000		
Iteration	Eco Size	Best Distance	CT (sec)	Eco Size	Best Distance	CT (sec)	Eco Size	Best Distance	CT (sec)
100	10	1666	3.58161	10	9110	22.9202	10	18680	51.9361
	20	1654	7.69532	20	9126	43.0239	20	18524	97.2948
	50	1596	17.8083	50	9132	108.86	50	18594	238.374
500	10	1644	16.853	10	9128	104.42	10	18486	239.101
	20	1594	34.0005	20	9046	203.152	20	18366	467.705
	50	1612	87.9	50	8992	492.564	50	18424	1158.89
1000	10	1634	33.5705	10	9042	198.784	10	18428	497.439
	20	1592	68.2468	20	8930	409.154	20	18510	928.561
	50	1588	167.591	50	9058	979.436	50	18454	2298.59

CONCLUSION

This article aims to investigate the reliability of sos algorithm for solving the picking order problems considering the real environments of warehouse. We attempt to apply SOS algorithm combined with mutation operator. From the results, it can be concluded that the SOS algorithm can provide a large percentage of decline for a small number of consumer demand in all iterations and ecosystem sizes, with a percentage decrease in distance of more 5%. As for the large number of customer demand (500 and 1000), the percentage decrease in the total distance is below or equal to 3.71% for all iterations. Moreover, the result shows number of iterations and ecosystem size affect computational time(CT) significantly, the higher the amount of data, ecosystem size and iteration, the higher the computation time required.

REFERENCES

1. T. L. Chen, C. Y. Cheng, Y. Y. Chen, and L. K. Chan, International Journal Production Economics **159**, 158-167 (2015).
2. J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco, *Facilities Planning*, 4th Ed. (Wiley, Chichester, 2010).
3. J. A. Cano, A. A. C. Espinal, and R. A. G. Montoya, Journal of King Saud University- Engineering Science **32**(3), 219-228 (2020).

4. A. Scholz, S. Henn, M. Stuhlmann, and G. Wäscher, European Journal of Operation Research **253**(1), 68-84 (2016).
5. S. Henn and G. Wäscher, European Journal of Operational Research **222**(3), 484–494 (2012).
6. C. Y. Tsai, J. J. H. Liou, and T. M. Huang, International Journal of Production Research **46**(22), 6533–6555 (2008).
7. J. C. H. Pan, P. H. Shih, H. Po, and M. H. Wu, Omega, **57** (Part B), 238-248 (2015).
8. F. Chen, H. Wang, C. Qi, and Y. Xie, Computer and Industrial Engineering **66**(1), 77-85 (2013).
9. R. De Santis, R. G Montanari, G. Vignali, and E. Bottani, European Journal of Operational Research **267**(1), 120–137 (2018).
10. Ç. Cergibozan and A. S. Tasan, Journal of Intelligent Manufacturing **30**(1), 335-349 (2019).
11. E. Bottani, G. Casella, and T. Murino, Computers & Industrial Engineering **160** 107540, 1-11 (2021).
12. D. H. Tran, M. Y. Cheng, and D. Prayogo, Knowledge-Based Systems **94**, 132-145 (2021).
13. M. Y. Cheng and D. Prayogo, Computers and Structures **139**, 98–112 (2014).
14. E. S Ezugwu, A. O. Adewumi, and M. E. Frîncu, Expert Systems with Applications **77**, 189-210 (2017).

Paper 028_OCTAVIA_ICAMME21

ORIGINALITY REPORT

0%	0%	0%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	On	Exclude matches	< 3%
Exclude bibliography	On		