Journal of Advanced Marine Engineering and Technology, Vol. 45, No. 2, pp. 51~59, 2021
J. Advanced Marine Engineering and Technology (JAMET)
https://doi.org/10.5916/jamet.2021.45.2.51

ISSN 2234-7925 (Print)
ISSN 2765-4796 (Online)
Original Paper

A hybrid method for the determination of the minimum number of transport vehicles in a shipyard

Minjoo Choi†

(Received January 5, 2021 ; Revised February 2, 2021 ; Accepted March 10, 2021)

Abstract: Shipbuilding companies use transport vehicles and often sign a rental contract with third-party vendors. To avoid including unnecessary vehicles in the contract, it is important to determine the proper number of transport vehicles when the contract is made. This paper presents a hybrid method for the determination of the minimum number of transport vehicles using a given task scenario in a shipyard. The method uses a genetic algorithm for making task-allocation decisions and uses an exact method for making the task-sequence decisions for individual vehicles. The exact method ensures optimality, and thus the hybrid method could have better performance than the genetic algorithm that alone determines both task-allocation and task-sequence decisions. This paper compares the performance of the hybrid method with that of the genetic algorithm using real industry data. The comparison shows that the hybrid method has improved performance, which tends to be more significant as the problem becomes more difficult to solve.

Keywords: Hybrid genetic algorithm; Exact method; Shipbuilding; Vehicle scheduling; Optimization

1. Introduction

Shipyards build massive products that comprises a large number of units of different sizes and weights. One important facility of the shipyards is the transport vehicles that move blocks or assembly units from one place to another. The number of such vehicles that a shipyard needs is dependent on the volume of material flows of the shipyard. This volume can vary due to the varying needs of newly built ships, which are highly dependent on the markets. Many shipyards tend to operate the transport vehicles and sign rental contracts with third-party vendors instead of having their own vehicles. To reduce material handling costs, it is therefore important to determine the proper number of transport vehicles when the contracts are made. If the shipyards reserve more vehicles than that they need, then the cost can increase for the extra (and unused) equipment. If they reserve fewer vehicles, then the transport needs are often delayed. The ideal number can be determined based on the optimal scheduling of the vehicles. The decision-makers can determine the number considering some safety margins on the minimum number.

There have been research papers associated with the optimal scheduling of transport vehicles in shipyards. Ju *et al.* **[1]** present an iterative approach that uses the lower bound of the minimum number of transporters in the determination of the optimal schedule. The scheduling optimization continues to increase until a feasible schedule is found. Lee *et al.* **[2]** use a heuristic algorithm to determine transporters' optimal schedule that has the minimum weighted sum in the delays of picking blocks and the delay times of transporters. The algorithm is based on a network flow model and can work in a dynamic operating environment such as the change of block transportation requirements. Roh and Cha **[3]** propose a hybrid method for a block transportation problem in which the objective is to minimize the travel time without loading or interference of transporters on the road. The hybrid method comprises two stages: the first stage uses an ant colony optimization to make block allocation decisions for transporters. The second stage uses a GA that determines the transportation sequence of the blocks of each transporter. Heo *et al.* **[4]** further develop the method of Roh and Cha **[3]** considering a damaged path in the scheduling problem. Cha *et al.* **[5]** use the method **[4]** in an optimal planning system. Park and Seo **[6]** used a heuristic referred to as Greedy Randomized Adaptive Search Procedures (GRASPs) for the transporter scheduling and routing problem at a shipyard, in which the objective is to maximize the workload balance of transports making sure that all the transport tasks are completed in a predetermined period. Joo and Kim **[7]** present a self-evolution algorithm (SEA) in the optimal scheduling of

† Corresponding Author (ORCID: http://orcid.org/0000-0001-6797-0210): Assistant Professor, Division of Naval Architecture and Ocean Systems Engineering, Korea Maritime & Ocean University, 727, Taejong-ro, Yeongdo-gu, Busan 49112, Korea, E-mail: minjoo.choi@g.kmou.ac.kr, Tel: +82-51-410-4304

transporters. The SEA is similar to a GA but involves self-reproduction operators that use only a single parent in creating a new chromosome. The authors use a dispatching rule in decoding a chromosome to a transporters' schedule that prevents infeasible solutions from occurring.

Meta-heuristic algorithms have often been used for optimal scheduling of transport vehicles in shipyards because the computation time of exact methods increases exponentially as the problem size becomes larger. However, there are variations in performance due to the stochastic nature of the meta-heuristic algorithms. These variations can be more significant when the complexity of the problem increases. The previous methods often use only a meta-heuristic algorithm or an exact method. Although there are hybrid methods, the methods combine one meta-heuristic algorithm and another. To take advantages of both meta-heuristic algorithm and exact method, this paper presents a hybrid method that combines a meta-heuristic algorithm and an exact method for the determination of the minimum number of transport vehicles in a shipyard.

2. Mathematical Description of the Problem

This section describes the minimum number determination problem for transport vehicles in a mathematical manner. The basic assumptions are 1) the vehicles are homogeneous, 2) the vehicles can perform any transport task, and 3) all tasks are known and each task has a deterministic timespan (or working hour); and the transition time between tasks is known. On this basis, two mixed integer linear programming models are made using two different assumptions: 1) fixed start time and 2) flexible start time of the tasks.

2.1 Optimization model for a fixed start time problem

This model is made assuming that each task has a specific start and finish time. This allows the conflicts of operation time between tasks to be identified before solving the problem. The model does not have subtour elimination constraints, which exponentially increase the computational time. **Equations (1)-(6)** describe the model.

Sets:

V	Vehicles, indexed by i
T	Tasks, indexed by j or k

Parameters:

M	Big number
ST_j	Start time of task j
FN_j	Finish time of task j
TR_{ij}	Transition time from task i to task j

Variables:

x_{ij}	1 if vehicle i performs task j, 0 otherwise
y_i	1 if vehicle i is used, 0 otherwise

Model:

$$\text{Max} \quad \sum_{i\in V} y_i \tag{1}$$

$$\sum_{j\in T} x_{ij} \le M y_i \qquad i \in V \tag{2}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad j \in T \tag{3}$$

$$x_{ij} + x_{ik} \le 1 \qquad i \in T,\ j \in T,\ k \in V \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad i \in V,\ j \in T \tag{5}$$

$$y_i \in \{0,1\} \qquad i \in V \tag{6}$$

Equation (1) is the objective function that minimizes the number of vehicles that are used. **Equation (2)** associates variable x_{ij} with y_i making sure that y_i has 1 if vehicle i is used in more than one task. **Equation (3)** ensures that each task is performed by a vehicle only once. **Equation (4)** allows vehicle v to perform both task i and task j only if the operation time of the tasks does not conflict to each other, in which if $ST_j + TR_{jk} < FN_k$ or $ST_k + TR_{kj} < FN_j$, task j and task k are compatible, otherwise, they are conflicted. **Equations (5)** and **(6)** define the decision variables.

2.2 Optimization model for a flexible start time problem

This model is made assuming that the start time of each task is flexible. In other words, there is no predetermined sequence between the tasks. Therefore, the model needs to include both task allocation and sequence decisions. The model is built based on a network model in which nodes indicate tasks and arcs indicate the transition from one task to another. There are two extra nodes referred to as source node and sink node. They represent the start and finish of the network, respectively. According to node combination, arcs belong to one of the subsets referred to as ‘pull-out arcs’, ‘pull-in arcs’, and ‘deadhead arcs.’ The arcs in the pull-out arcs have source-task node combination, the arcs in the deadhead arcs have task-task node combination, and the arcs in the pull-in arcs have task-sink node combination. **Figure 1** describes the network model and the arc subsets.

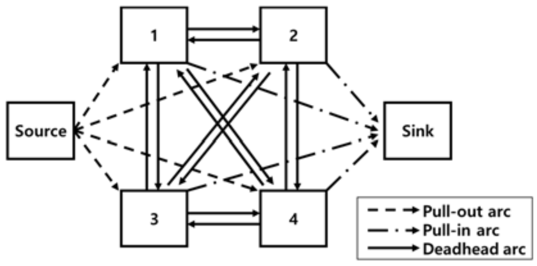


Figure 1: An example of a network-based model using three subsets of arcs

Equations (7)-(16) define the flexible start time model using the network model. **Equation (7)** is the objective function that minimizes the number of vehicles that are used. **Equation (8)** ensures that all tasks are performed by one of the vehicles once. **Equations (9)-(11)** are the flow conservation constraints of the network model. **Equation (12)** makes sure that all the vehicles can perform the allocated tasks within the given time. **Equation (13)** is the sub-tour elimination constraints. **Equation (14)** and **(15)** define the decision variables.

Sets:

N	Nodes, indexed by n
T	Nodes associated with tasks, indexed by t
V	Vehicles, indexed by k
A	Arcs, indexed by (i,j)
A_{po}	Pull-out arcs, indexed by (i,j)
A_{pi}	Pull-in arcs, indexed by (i,j)
A_{dh}	Deadhead arcs, indexed by (i,j)

Parameters:

MV_{ij}	Transition time of arc (i,j)
W_i	Working time of task i
TR_{ij}	Transition time from node i to node j
FN	Finish time of the timetable
M	Big number

Variables:

x_{ijk}	1 if arc (i,j,k) is selected, 0 otherwise
u_i	i-th dummy variable that for eliminating subtours

Model:

$$\text{Max} \quad \sum_{k\in V}\sum_{(i,j)\in A_{po}} x_{ijk} \tag{7}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A_{po}\cup A_{dh}} x_{ijk} = 1 \qquad j\in T,\ k\in V \tag{8}$$

$$\sum_{(i,j)\in A_{po}} x_{ijk} = \sum_{(i,j)\in A_{pi}} x_{ijk} \qquad k\in V \tag{9}$$

$$\sum_{(i,j)\in A_{po}} x_{ijk} \le 1 \qquad k\in V \tag{10}$$

$$\sum_{(i,j)\in A_{po}\cup A_{dh}} x_{ijk} = \sum_{(i,j)\in A_{po}\cup A_{dh}} x_{jik} \qquad j\in T,\ k\in V \tag{11}$$

$$\sum_{(i,j)\in A_{dh}} x_{ijk} \le M \sum_{(i,j)\in A_{po}} x_{ijk} \qquad k\in V \tag{12}$$

$$\sum_{(i,j)\in A_{po}\cup A_{dh}} W_j x_{ij} + \sum_{(i,j)\in A_{dh}} TR_{ij} x_{ij} \le FN \qquad k\in V \tag{13}$$

$$u_i - u_j + Mx_{ijk} \le M-1 \qquad (i,j)\in A,\ k\in V \tag{14}$$

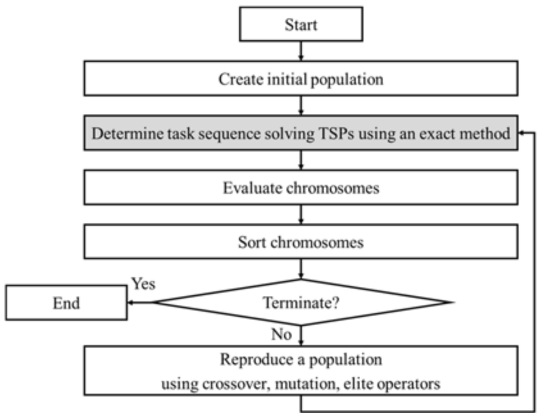
$$x_{ijk}\in\{0,1\} \qquad (i,j)\in A,\ k\in V \tag{15}$$

$$0\le u_i \qquad i\in N \tag{16}$$

3. A Hybrid Method for the Flexible Start Time Problem

As Joo and Kim **[7]** present, using an exact method alone for the flexible start time problem causes an exponential increase in computation time as the problem size increases. To avoid that problem, this paper uses a hybrid method that combines a GA and an exact method in which the GA determines task allocation, and the exact method determines task sequence, respectively. **Figure 2** describes the overall process.

Figure 3 describes the chromosome representation of the GA. The number of tasks defines the length of the chromosome. Each task is associated with a gene that comprises the vehicle identification number (ID) allocated and the priority value of the task. The priority values determine the task sequence in which a lower value task receives a higher priority.



Start
Create initial population
Determine task sequence solving TSPs using an exact method
Evaluate chromosomes
Sort chromosomes
Yes
End
Terminate?
No
Reproduce a population using crossover, mutation, elite operators

Figure 2: The overall process of the hybrid method

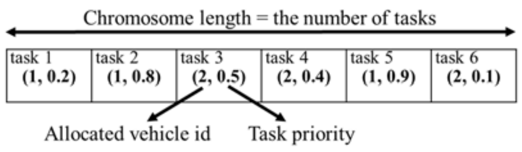


Figure 3: Chromosome representation scheme

In ‘Create initial population,’ the GA creates initial chromosomes that have random priority values. In ‘Determine task sequence’, the exact method fixes the priority values solving the traveling salesman problem (TSP), which is a well-known optimization problem in which the objective is to minimize the travel distance visiting a list of nodes only once **[8]**. Dantzig *et al.* **[9]** present the branch-and-cut method for symmetric TSPs and Eastman **[10]** and Little *et al.* **[11]** present the branch and bound method for asymmetric TSPs. Such methods have been improved further and are available in commercial optimizers such as the solvers of CPLEX **[12]** and Gurobi **[13]**.

The solution of a TSP has a cycle that indicates the start node and the finish node are same; thus, the longest arc of the TSP solution needs to be removed for the task-sequencing problem. **Figure 4** describes the removal in the conversion of a TSP solution to the optimal task sequence of vehicle 1 in **Figure 2**.

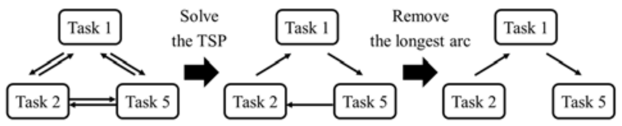


Figure 4: Conversion of a TSP solution to the optimal task sequence

In ‘Reproduce a population,’ crossover, mutation, and elite operators create new chromosomes. This paper uses a one-point cutting crossover represented in **Figure 5**. The exact method fixes the task sequences solving TSPs when all chromosomes are created.

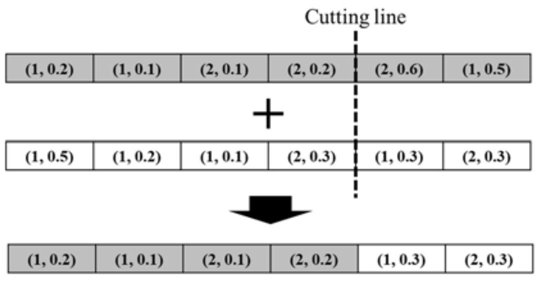


Figure 5: An example of a one-point cutting crossover

When the hybrid method determines both task allocation and sequence, the task schedule of individual vehicles can be created using the working time and transition time, which are defined as W_i and TR_{ij} in Section 2.2, respectively. **Figure 6** illustrates the conversion of a chromosome to a timetable.

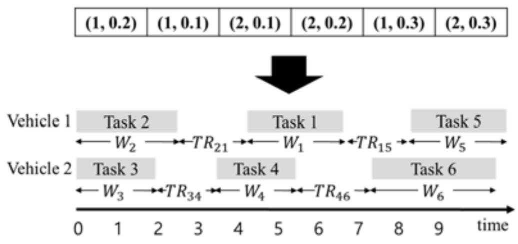


Figure 6: An example of the conversion from a chromosome to a timetable

4. Case Study

This section describes a case study in which the hybrid method was applied to a real-world problem. In the case study, a shipyard was operating with multiple types of transport vehicles signing rental contracts with third-party vendors, but the prime focus is forklifts (see **Figure 7**). The shipyard was using 61 forklifts and wanted to determine the minimum number using previous transport tasks.

The determination used the data that were collected from the forklifts using real-time sensors. The data include 1) data acquisition time, 2) the position of forklifts, and 3) status of the forklifts in which the status indicates if the forklift is working or not. **Figure 8** describes the working history of the forklifts made using the data. The horizontal axis indicates the IDs of the forklifts and the vertical axis indicates the time periods, assuming that a day comprises 10 hours between 8:00 to 18:00. The length of a time period is an hour. The white-to-red colour indicates the work density of the forklifts defined by working time per hour.

Figure 7: An example of a forklift

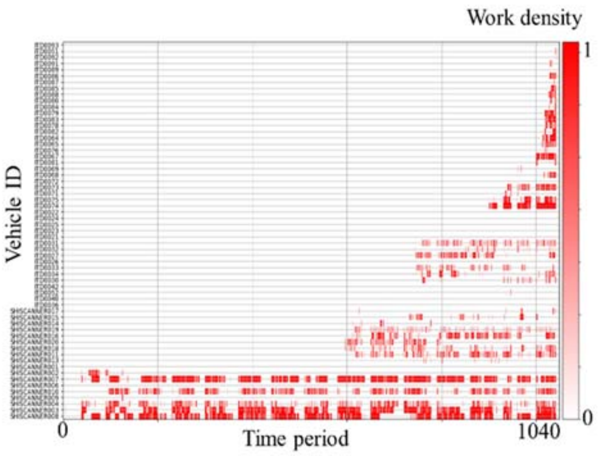


Figure 8: The density of work of forklifts

An interesting observation in **Figure 8** is that the work density in the top-left area is 0. This appears because the shipyard had installed the sensors gradually from the forklifts at the bottom in **Figure 8**. Transport tasks were defined using the work density assuming that in the same day, the continuous time periods that have work density over 0.5 were defined as a collective task. **Table 1** describes the tasks created using the definition. The task information comprises a start time, working hour, start position, and finish position. The value in the column 'start time' indicates 'year.month.day.hour.minute.second.' The unit of the column 'working hour' is an hour. The start position and finish position indicate the node IDs in **Figure 9**, in which the shipyard is represented using 48 nodes and 56 undirected edges. **Table 2** describes the information of the edges.

Table 1: An example of task information

Task (t)	Start time	Working hour	Start position	Finish position
1	2020.01.09.08.00.04	3.66	36	39
2	2020.01.09.08.05.28	3.87	26	26
3	2020.01.09.08.02.45	1.95	19	28
4	2020.01.09.09.00.13	2.90	25	19

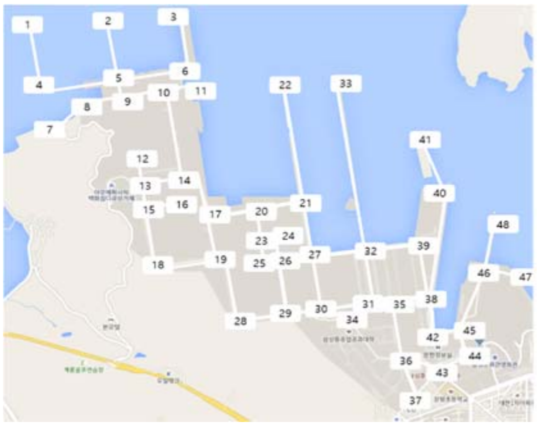


Figure 9: The graph representation of the shipyard

Table 2: Edge information

Edge	Node pair	Edge	Node pair	Edge	Node pair
1	(1, 4)	20	(17, 20)	39	(31, 32)
2	(2, 5)	21	(18, 19)	40	(31, 35)
3	(3, 6)	22	(19, 28)	41	(32, 33)
4	(4, 5)	23	(20, 21)	42	(34, 36)
5	(5, 6)	24	(20, 23)	43	(35, 36)
6	(5, 9)	25	(21, 22)	44	(35, 38)
7	(6, 10)	26	(21, 27)	45	(36, 37)
8	(7, 8)	27	(23, 24)	46	(36, 43)
9	(8, 9)	28	(23, 25)	47	(38, 39)
10	(9, 10)	29	(24, 26)	48	(38, 42)
11	(10, 11)	30	(25, 26)	49	(39, 40)
12	(10, 14)	31	(26, 27)	50	(40, 41)
13	(12, 13)	32	(26, 29)	51	(42, 45)
14	(13, 14)	33	(27, 30)	52	(43, 44)
15	(13, 15)	34	(27, 32)	53	(44, 45)
16	(14, 17)	35	(28, 29)	54	(45, 46)
17	(15, 16)	36	(29, 30)	55	(46, 47)
18	(15, 18)	37	(30, 31)	56	(46, 48)
19	(17, 19)	38	(30, 34)		

The transition time between task i and task j was computed using the finish position of task i and the start position of task j assuming that the vehicles move at 10 km/h in a constant speed, and the vehicles always use the shortest paths determined by the Dijkstra algorithm using the graph in **Figure 9**. **Tables 3-4** show examples of the task-to-task transition time TR_{ij} and shortest paths, respectively.

Table 3: Samples of task-to-task transition time (TR_{ij}).

		Task j				
		1	2	3	4	5
Task i	1	0.01	0.17	0.32	0.32	0.22
	2	0.13	0	0.18	0.18	0.12
	3	0.28	0.18	0	0	0.11
	4	0.28	0.28	0	0	0.11
	5	0.13	0.07	0.15	0.15	0.05

Note: Unit of transition time is hour.

Table 4: Samples of shortest paths between node i and node j.

node i	node j	Distance	Shortest path
1	2	1.32	1-4-5-2
1	3	1.65	1-4-5-6-3
			…
1	48	5.64	1-4-5-9-10-14-17-19-28-29-30-31-35-38-42-45-46-48
			…
48	46	0.33	48-46
48	47	0.51	48-46-47

Note: Unit of travel distance is km.

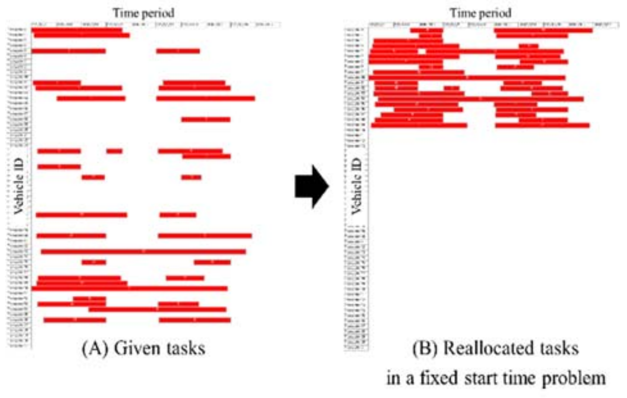
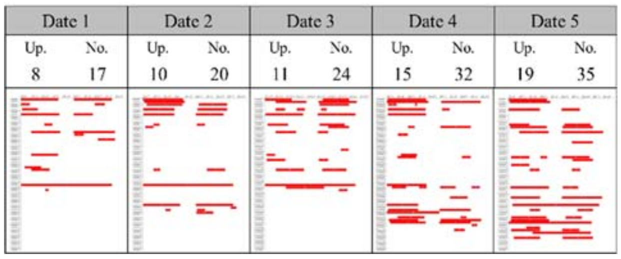


Figure 10: The minimum number of forklifts; the vertical axis indicates vehicle IDs and the horizontal axis indicates time periods

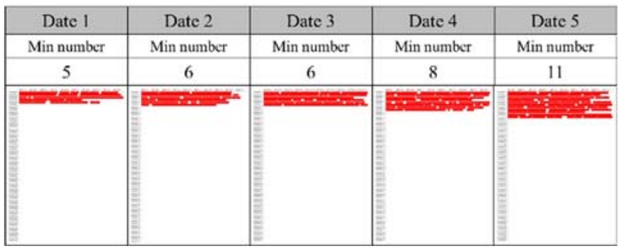
Calculating the minimum number of the forklifts is the flexible start time problem because the shipyard considers the start time to be flexible. In applying the hybrid method, the level of difficulty of the problem is proportional to the number of tasks, which determines the length of chromosomes, and the maximum number of vehicles. This in turn determines the value range of genes. To reduce the level of difficulty, the upper bound was used to determine the maximum number of forklifts instead of considering all given 61 forklifts. The minimum number of the fixed start time problem can be used for the upper bound because the optimal solution of the fixed start time problem is always greater than or equal to that of the flexible start time problem, i.e., the constraints of the flexible start time problem are a subset of the fixed start time problem. For instance, **Figure 10** describes the optimal solution of a fixed start time problem in which the minimum number of forklifts is 19. This can be used to determine the maximum value of allocated forklift IDs in creating the genes of the hybrid method for the flexible start time problem.

Although the data were collected from 28th September 2019 to 9th January 2020, it is difficult to use all data because many of the sensors were installed late. The determination thus used the last five days, in which most of the forklifts have sensors. **Figure 11** describes the upper bound denoted by 'Up.' and the number of tasks of the selected dates denoted by 'No.', respectively. **Figure 12** describes the optimal schedules that use the minimum number of forklifts for the dates. The vertical and horizontal axes of **Figures 11-12** are same with that of **Figure 10**, in which the axes indicate vehicle IDs and time periods, respectively.



Date 1		Date 2		Date 3		Date 4		Date 5	
Up.	No.	Up.	No.	Up.	No.	Up.	No.	Up.	No.
8	17	10	20	11	24	15	32	19	35

Figure 11: The upper bound and the number of tasks



Date 1	Date 2	Date 3	Date 4	Date 5
Min number	Min number	Min number	Min number	Min number
5	6	6	8	11

Figure 12: The minimum number of forklifts with a flexible start time problem

In addition, the performance of the hybrid method was compared with that of the GA using the dates. Both the hybrid method and the GA use 50 chromosomes, and the termination criterion reaches the 2000th iteration. The fraction of crossover, mutation, and elite is 80%, 12%, and 8%, respectively. These parameters were determined based on numerous trials. The TSPs were solved using CPLEX's solver **[12]** to make task-sequence decisions. **Figure 13** describes an example of the convergence of the objective value over the 2000 iterations.

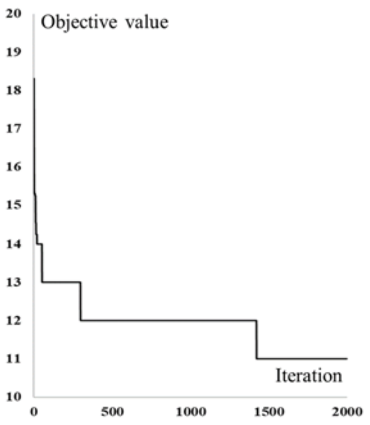


Figure 13: The convergence of objective values of the hybrid GA

There can be variation in performance because of the stochastic nature of the GA. The comparison thus uses 20 replicates for the hybrid method and the GA at each date. **Table 5** presents comparison results in which $\bar{\rho}$ indicates the mean of the relative percentage deviations, and each ρ was computed by $\frac{x-x^*}{x^*}$. Herein, x^* and $x^\sim$ are the optimal solution and the worst solution respectively, and they indicate the minimum and the maximum numbers found in the 20 replicates; x is the minimum number found in each replicate. Both the hybrid method and the GA could determine the minimum number more than several times out of the 20 replicates in all the dates. Although the solution found in a replicate was not optimal, the absolute gap is only 1 (see all gaps between the best and worst solution for all dates). However, the $\bar{\rho}$ shows that the hybrid method has more stable performance than the GA with a higher success rate of determining the optimal solution.

Table 5: The performance comparison between the hybrid method and the GA

	GA			Hybrid method		
	$\bar{\rho}$	x^*	$x^\sim$	$\bar{\rho}$	x^*	$x^\sim$
Date 1	0%	5	5	0%	5	5
Date 2	0%	6	6	0%	6	6
Date 3	16%	6	7	9%	6	7
Date 4	10%	8	9	6%	8	9
Date 5	8%	11	12	3%	11	12

Solving the TSP of the hybrid method could increase the computation time because it is an additional process that GA does not have. Thus, one needs to compare computation time between the hybrid method and the GA. **Figure 14** shows the average computation time in each date. As the upper bound and the number of tasks increase, the computation time of both the hybrid method and the GA tend to increase; the computation time of the GA is always less than the hybrid method. However, the average computation time in the worst case was about 10 seconds: this computation time is not significant when it come to the purpose of the problem, in which real-time scheduling is not the primary concern. Thus, the computation time of the hybrid method seems to be reasonable.

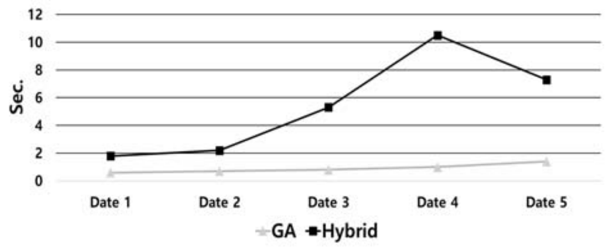


Figure 14: Comparison between the hybrid method and GA in average computation time for each date

It is also worth investigating how the computation time of TSPs increases in different number of nodes. For this, some numerical experiments were carried out, in which the number of nodes increases from 4 to 18. **Figure 15** describes the results of the experiments. When the number of nodes is less than 14, the computation time is less than one second. The computation time then increases exponentially. When considering that the average number of tasks assigned to a vehicle is around 6, the problem-solving time for TSPs is not significant in the minimum number determination problem.

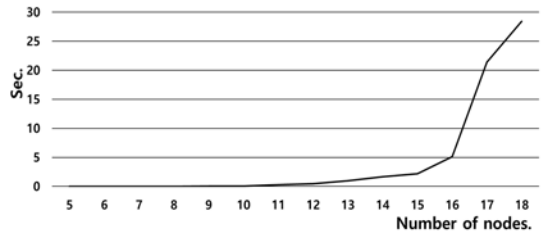


Figure 15: Experiments on the computation time of TSPs in different number of nodes

5. Conclusion

In this paper, a hybrid method was presented for the determination of the minimum number of transport vehicles in a shipyard. The hybrid method uses a GA for making task allocation decisions and an exact method for making task-sequence decisions. The exact method determines the optimal task sequence by solving TSPs. The objective is to determine the shortest round-trip path that visits all given nodes only once. Because the exact method ensures the optimality of the task sequence, the hybrid method could have better performance than the GA that alone makes both task allocation and task sequence decisions. The hybrid method was presented in a case study in which previous tasks in a shipyard were analyzed using working data from forklifts collected using real-time sensors. Using the tasks, the performance of the hybrid method was compared with that of a GA. The comparison shows that the hybrid method has better performance than the GA; this superiority can be more significant as the problem becomes more difficult to determine the optimal solution.

The case study in this paper deals with a single type of vehicles. However, the shipyard operates multiple types of vehicles with different levels of capacities. Future studies should study heterogeneous vehicles considering the difference of each vehicle type, i.e., operating costs and compatibility between the tasks and the vehicles.

There are potential risks in solving the TSPs using commercial optimizers such as the solvers of CPLEX **[12]** and Gurobi **[13]**. Although there have been significant improvements in the performance of the commercial optimizers, the computation time of solving the TSPs can increase exponentially as the number of tasks increase. In this case study, the maximum number of tasks allocated to a vehicle was 12, which is not problematic in terms of computation time. However, there may need to be a strategy for cases in which the maximum number could be problematic with commercial solvers because of the increased computation time.

Acknowledgement

This work was supported by the Korea Maritime & Ocean University Research Fund.

Author Contributions

Conceptualization, M. J. Choi; Methodology, M. J. Choi; Software, M. J. Choi; Formal Analysis, M. J. Choi; Investigation, M. J. Choi; Data Curation, M. J. Choi; Writing-Original Draft Preparation, M. J. Choi; Writing-Review & Editing, M. J. Choi; Visualization, M. J. Choi; Supervision, M. J. Choi; Funding Acquisition, M. J. Choi.

References

[1] C. -M. Ju, U. -S. Lee, and G. -B. Lee, "Transporter scheduling for block transportation in the shipyard," Proceedings of the Korean Operations and Management Science Society Conference, pp. 348-352, 2005 (in Korean).

[2] W. -S. Lee, W. -I. Lim, and P. -H. Koo, "Transporter scheduling based on a network flow model under a dynamic block transportation environment," Proceedings of International Conference on Computers & Industrial Engineering, pp. 311-316, 2009. https://doi.org/10.1109/ICCIE.2009.5223874.

[3] M. -I. Roh and J. -H. Cha, "A block transportation scheduling system considering a minimisation of travel distance without loading of and interference between multiple transporters," International Journal of Production Research, vol. 49, no. 11, pp. 3231-3250, 2011. https://doi.org/10.1080/00207543.2010.484427.

[4] Y. -J. Heo, J. -H. Cha, D. -Y. Cho, and H. -C. Song, "Optimal block transportation path planning of transporters considering the damaged path," Journal of the Society of Naval Architects of Korea, vol. 50, no. 5, pp. 298-306, 2013 (in Korean). https://doi.org/10.3744/SNAK.2013.50.5.298.

[5] J. -H. Cha, D. -Y. Cho, W. -S. Ruy, and H. -J. Hwang, "Development of optimal planning system for operating transporters in shipyard," Journal of Society for Computational Design and Engineering, vol. 21, no. 2, pp. 177-185, 2016 (in Korean).

[6] C. -K. Park and J. -Y. Seo, "A GRASP approach to transporter scheduling and routing at a shipyard," Computers & Industrial Engineering, vol. 63, no. 2, pp. 390-399, 2012.

[7] C. -M. Joo and B. -S. Kim, "Block transportation scheduling under delivery restriction in shipyard using meta-heuristic algorithms," Expert Systems with Applications, vol. 41, no. 6, pp. 2851-2858, 2014. https://doi.org/10.1016/j.eswa.2013.10.020.

[8] G. Laporte, "A short history of the traveling salesman problem," Canada Research Chair in Distribution Management, Centre for Research on Transportation (CRT) and GERAD HEC Montréal, Canada, 2006.

[9] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," Journal of the Operations Research Society of America, vol. 2, no. 4, pp. 393-410, 1954.

[10] W. L. Eastman, "Linear programming with pattern constraints," Ph.D. Dissertation, Department of Economics, Harvard University, USA, 1958.

[11] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel, “An algorithm for the traveling salesman problem,” Operations research, vol. 11, no. 6, pp. 972-989, 1963.

[12] I. I. Cplex. V12. 1: User’s Manual for CPLEX, NY, USA: International Business Machines Corporation, 2009.

[13] Gurobi optimizer reference manual, https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf, Accessed March 31, 2021.