A GRASP Algorithm for Pickup and Delivery Problem with Time Windows in Vessel Routing Problem

Received: 1 Sep 2020 Accepted: 10 Jan 2021

Shirin Mardanpour Shahrekordi[1*]., Koorush Ziarati[2].
[1] *MSc Student; Shiraz University; sh.mardanpour@shirazu.ac.ir*
[2] *Associate Professor; Shiraz University; ziarati@shirazu.ac.ir*

Abstract— *In a recent study, researchers investigated a class of vessel routing problem and a benchmark suite based on real shipping segments considering incompatibility constraints. These constraints same as pickups and deliveries, cargoes selections, travel times and costs and time windows state considerable challenges for researchers. Considering the literature review on the subject and the frequent resolving of this problem with Adaptive Large Neighborhood Search (ALNS), we proposed a Greedy Randomized Adaptive Search Procedure (GRASP) to solve this problem. The algorithm was tested on 240 available benchmarks. As shown in our experimental results the GRASP outperforms all previous heuristics and generates near-optimal solutions within minutes. These results are noteworthy since we have succeeded to improved 35 large instances of this set.*

Keywords— Vessel routing, Pickup and delivery, Maritime optimization, Adaptive large neighborhood search, Grasp.

1. INTRODUCTION

International trade depends heavily on vessel transportation, as it is the only cost-effective means for the transportation of large volumes over long distances. It is common to distinguish between three main modes of operation in maritime transportation: liner, industrial, and tramp shipping.

Liner shipping, which includes container shipping, is similar to a bus service: fixed schedules and itineraries must be followed. In industrial shipping, the operator owns the cargoes and controls the fleet, trying to minimize the cargo transportation cost. Finally, a tramp shipping often transporting a mix of mandatory and optional cargoes with the goal of maximizing profit [1]. In this work, we focus on a class of industrial and tramp ship routing and scheduling problems (ITSRSPs).This class of problems typically arise in the shipping of bulk products.

In a recent work in [2], a set of benchmark instances based on real shipping segments with 7 to 130 cargoes (pickup-and-delivery pairs) has been made available to the academic community. The authors also designed an adaptive large neighborhood search (ALNS) heuristic and subsequently investigated the impact of randomization as well as that of various search operators [2]. However, due to the lack of available lower bounds or optimal solutions, the true performance of these methods is unknown for large problems. This article contributes to fill this methodological gap, from a metaheuristic standpoint.

2. PROBLEM DESCRIPTION

The objective of ITSRSP is to form routes that minimize the sum of the total travel cost and the possible penalties in the case where charter vessels are used or some cargoes are not transported. The routes begin at their respective starting points but have no specified endpoint, since vessels operate around the clock. Every route must be feasible: vessels cannot exceed their capacity, cargoes can be serviced only within their prescribed time windows, and ships cannot transport incompatible cargoes. Furthermore, the routes must respect pairing and precedence constraints.

We consider a shipping company in which take place orders for the pickup and delivery. In the following mathematical formulation shown for the routing and scheduling of vessels. If we let *i* define a cargo, there is a node *i* (same index for p-d pairs) related to the loading port and a node i+n related to the unloading port, with *n* being the number of cargoes. The set of loading and unloading nodes shows with N^P and N^D, respectively. A set of a visited node by the vessel v is considered to be N_v, and this set contains an origin node *o(v)* and a destination node *d(v)*. The set of arcs that vessel v can pass through is A_v. We also introduce $N_v^P = N^P + N_v$, $N_v^D = N^D + N_v$ for loading and unloading nodes can be visited by vessel *v*, respectively.

Each node in the search space has a time window. The cost of shipping from *i* to *j* using vessel *v* is C_{ijv}, and the relevant travel time is T_{ijv}. The time at which service starts at node *i* using vessel v is t_{iv}, and l_{iv} is the total load on board after completing service at node *i* using vessel *v*. The variables x_{ijv} are binary flow variables, denoting whether vessel *v* moves directly from node *i* to node *j*. Binary variables y_i show whether cargo *i* is transported by the available vessel fleet. If the cargo is not transported with the available ship fleet, a cost C_i^S is incurred. The mathematical formulation for the vessel routing problem is given below [3].

Min $\sum_{v\in V}\sum_{(i,j)\in A_v} C_{ijv}\, X_{ijv} + \sum_{i\in N^P} C_i^S\, y_i$ (1)

S.t.

$\sum_{v\epsilon V}\ \sum_{j\epsilon N_v} x_{ijv} + y_i = 1\,,$ $\quad i\epsilon N^P,$ (2)

$\sum_{j\in N_v} X_{0(v)jv} = 1\,,$ $\quad v\ \epsilon\ V,$ (3)

$\sum_{j\in N_v} X_{ijv} - \sum_{j\in N_v} X_{jiv} = 0\,,$ $\quad v \in V, i \in N_v \setminus \{o(v), d(v)\},$ (4)

$\sum_{j\in N_v} X_{jd(v)v} = 1\,,$ $\quad v\ \epsilon\ V,$ (5)

$l_{iv} + Q_j - l_{jv} \le K_v\big(1 - x_{ijv}\big),$ $\quad v \in V, j \in N_v^P, (i,j) \in A_v,$ (6)

$l_{iv} - Q_j - l_{(n+j)v} \leq K_v\left(1 - x_{i(j+n)v}\right),$ $v \in V, j \in N_v^P, (i, n+j) \in A_v,$ (7)

$0 \leq l_{iv} \leq K_v$, $v \in V,\ i \in N_v^P,$ (8)

$t_{iv} + T_{ijv} - t_{jv} \leq \left(\overline{T}_i + T_{ijv}\right)\left(1 - x_{ijv}\right),$ $v \in V, j \in N_v^P, (i, n+j) \in A_v,$ (9)

$\sum_{j \in N_v} X_{ijv} - \sum_{j \in N_v} X_{(n+i)jv} = 0$, $v \in V, i \in N_v^P,$ (10)

$t_{iv} + T_{i(n+i)v} - t_{(n+i)v} \leq 0,$ $v \in V, i \in N_v^P,$ (11)

$\underline{T}_i \leq t_{iv} \leq \overline{T}_i$, $v \in V, i \in N_v,$ (12)

$y_i \in \{0,1\}$, $i \in N^C,$ (13)

$X_{ijv} \in \{0,1\},$ $v \in V, (i,j) \in A_v,$ (14)

The objective function is to minimize sums up the costs from operating the fleet plus the cost of spot charters. Constraints (2) show that all cargoes must either be picked up by a vessel or transported using spot charter. Constraints (3), (5) state the movements of the vessels. The vessel's load in the loading and unloading nodes is indicated through constraints (6) and (7). Constraints (8) assure that the load does not exceed the vessel capacity. Constraints (9) make sure that the time at which service starts is possible with respect to travel times. Constraints (10) sure that if cargo is loaded, its unloading port is also visited by the same vessel. Precedence requirements are imposed through constraints (11). Time windows are shown by constraints (12), finally, binary requirements on the spot charter and flow variables are given by constraints (13) [3].

The above model is both credible for deep sea and short sea problems. It is also credible-both for the full load and the mixed load case, though more impressive formulations can be obtained for the full load case, see for example[4]. The industrial and tramp vessel cargo routing and scheduling problem is NP-hard, being more common than the TSPTW [5][3].

To our knowledge, the study of researchers [3] is the only one to report lower bounds and optimal solutions for these benchmark instances, obtained by solving a mixed-integer programming (MIP) formulation. Exact methods like MIP could solve the majority of the instances, but their CPU time becomes widely vary, even for instances with similar characteristics. Therefore, fast metaheuristic solutions remain indispensable for applications requiring a response in a guaranteed short time. For this purpose, we have examined the GRASP metaheuristic to solve this problem. Our reason for choosing the GRASP algorithm was to pick out a metaheuristic that is close to the successful heuristic previously presented for this problem so that we can use the powerful operators of the previous method[2][3]. The second reason for choosing this algorithm was to solve the vessel routing problem based on our previous experience that generally this algorithm generates good solutions to the routing problems.

3. PROPOSED ALGORITHM

GRASP [6] is a multi-start metaheuristic that has been widely used for finding good quality solutions in high variety to many hard combinatorial optimization problems. It relies on greedy randomized constructions and local search phase. The problem involves many decisions at different levels. The main idea is first to estimate point-to-point requests and then, based on this information, to apply the iterative GRASP phases of construction and local search. Each of these phases is described next.

a) Construction phase: Add-Node procedure

A solution consists of a set of routes followed by the number of ships and cargoes. In each route, some cargoes are transported by a specific vessel. The sequence of loading and unloading cargo is represented by using the cargo number twice, first for the pickup and then for the delivery of the cargo. The initial which ensures that the solution is feasible at each stage. In the constructive phase, we find all feasible positions for each cargo. Since the search space of a problem for generating the initial solution is very large and gradually expanding, we start with the empty solution. For this purpose, we used the idea presented in the ALNS operators [7] to perform a gradual solution phase. The sequence is that we select a list of cargoes at random. These requests represent the removal of cargoes from the problem space and their placement elsewhere in the candidate solution. The removal operator is performed with the random selector. After remove requests, the cargoes will be gradually reinserted to the candidate solution. The insertion operation is to consider all the locations that lead to a feasible solution. Note that at this point we will add to each candidate list any places that incur a lower incremental cost for pickup and delivery positions. Then we sort the candidate list in ascending order, and according to the input parameter RCL from a percentage of the list members, a solution is chosen at random. The same goes for the rest of the cargo.

b) Construction phase: Add-Node procedure

A solution generated by the constructive phase might not necessarily satisfy the capacity constraints. Therefore, the goal of the local search is to improve the quality of the solution or to repair infeasibility if needed. The proposed local search that depicted in Figure 3 is ALNS without Adaptive factor. After we implemented ALNS heuristic to apply the local search phase of GRASP, we found that the adaptive weight factor makes the algorithm to slow down so we used large neighborhood search (LNS) with 5 operators to explore and exploit the search space. In this phase, we perform a local search step with respect to the solution of the previous step. For local search, we use one of the three removal operators in ALNS[7].

The local search used in the proposed method has all the operators expressed in [7], except that the adaptive weight procedure for selecting each operator in each iteration not used. The procedure is to select a random number between [0, 1] and select one of the available operators. It should be noted that since the Shaw operator yields better results in the ALNS method, it has a higher chance of selection (1% chance of selection) and two Random and Worst operators of equal chance (1% chance of choice) for the cargo removal operation. To perform the local search phase, a variable called the neighborhood radius is used to specify our local search range. In the local search phase, we found the solutions that make a decision using the acceptance criterion in the ALNS heuristic [7], [3]. Below we discuss the acceptance criterion factor.

c) Construction phase: Add-Node procedure

Simple acceptance criteria would be to only accept solutions that are better than the current solution. We used the acceptance criterion in simulated annealing which accepts solutions that are worse than the current solutions too with probability〖 e〗^(-|f-f_new |/T), where T > 0 is the temperature. To do this in the first iterations which the difference in the cost of solutions is increased should limit the acceptance of worse solutions and in the last iteration which converges almost to the best solution, we have the chance of moving zigzag in the search space.

This criterion is set for the first 20 iterations. We calculate the average difference of the worsening of the solutions. This average as the feedback of the first 20 first repetitions along with the initial temperature parameter T can be used in the acceptance probability formula. Therefore, based on the benchmark size, we set the initial temperature as follows:

Tstart = number of cargoes × 1000 + avg (Σf_i-f_(i+1))

According to our investigations, the initial temperature T for benchmarks with a cargo size of over 50 is more significant. Because there is a possibility that there will be no feedback from the worsening of the responses in 20 iterations in large benchmarks.

4. EXPERIMENTAL RESULTS

The proposed algorithm in this paper was coded in C++ and executed on an Intel Cori7 with 2.10 GHz CPU, 8 GB RAM, and a 64-bit operating system. As described before, there are 240 standard instances of this problem. The GRASP was executed ten times for each benchmark, each run including 25,000 iterations. These instances are divided into four groups of 60, according to problem topology and cargo type¬¬. The mixed load instances have up to 130 cargoes and 40 vessels, whereas the full load instances have up to 100 cargoes and 50 vessels. So far, 123/240 instances remain open [2]. In the full-load instances, each delivery should be visited immediately after its associated pickup due to the absence of residual capacity for other loads.

This property is no longer valid for mixed load instances. Furthermore, short and deep-sea instances consider different geographical regions. The short sea instances represent shipments among European ports, whereas the deep sea instances involve long-distance ships [3]. This will indicate how difficult industrial and tramp vessel routing and scheduling problems are to solve to optimality, although we solved larger instances by using a tailored method. According to Table 1 and focusing on large-sized instances, 35 instances with 50 and 130 cargoes and between 20 and 40 vessels improved. Instances consist of both short-sea and deep-sea shipping benchmarks.

This section reports our computational experiments with the two ALNS algorithms and the GRASP[3], [2]. Our aim is twofold:

• We compare the proposed algorithms, and evaluate their ability to solve practical-size vessel routing problems to optimality in limited time

• In situations where a faster response is sought, we evaluate the quality of the solutions produced by the GRASP metaheuristic. Obtaining a good sequence of cargoes during constructing a route is the main difference between this algorithm and its priors. In Table 1, the first column shows the number of benchmarks and the second one presents the name of instances in each group. The old and new best-known solutions and the average gap in each group can be seen in the next column repeatedly. The following formula is used for calculating the gap between the optimal solution and the obtained solution for each benchmark:

$$Gap = \frac{(old\ best-known)-(new\ best-known)}{(old\ best-known)} \times 100$$

Furthermore, among 2400 runs (10 runs of 240 instances) of each benchmark, the best performance with is 0.64 % where the performance is calculated based on the gap to the best-known solutions. The results of the tests also led to finding some new best results for the data instances used.

Table 1- Experimental results

Number of Instances	**#Instance**	**Old Best Cost**	**New Best Cost**	**Average GA**
		ALNS	***GRASP***	**(%)**
1	SHORTSEA_MUN_C23_V13_HE_2	2255870	**2255469**	-0.018
2	SHORTSEA_MUN_C80_V20_HE_3	9763401	**9726826**	-0.375
3	SHORTSEA_MUN_C80_V20_HE_5	10983117	**10948325**	-0.317
4	SHORTSEA_MUN_C100_V30_HE_1	12845591	**12780320**	-0.508
5	SHORTSEA_MUN_C100_V30_HE_2	13057536	**13000084**	-0.440
6	SHORTSEA_MUN_C100_V30_HE_3	12088444	**12082339**	-0.051
7	SHORTSEA_MUN_C100_V30_HE_4	13791917	**13765993**	-0.188
8	SHORTSEA_MUN_C130_V40_HE_1	16524192	**16505513**	-0.113
9	SHORTSEA_MUN_C130_V40_HE_2	16713067	**16651942**	-0.366
10	SHORTSEA_MUN_C130_V40_HE_3	15862154	**15786198**	-0.479
11	SHORTSEA_MUN_C130_V40_HE_4	17305841	**17194544**	-0.643
12	SHORTSEA_FUN_C70_V30_HE_1	10088768	**10074896**	-0.137
13	SHORTSEA_FUN_C70_V30_HE_3	10314521	**10309352**	-0.050
14	SHORTSEA_FUN_C70_V30_HE_4	10910832	**10909957**	-0.008
15	SHORTSEA_FUN_C90_V40_HE_3	12767716	**12703025**	-0.507
16	SHORTSEA_FUN_C90_V40_HE_5	13720466	**13654943**	-0.478
17	SHORTSEA_FUN_C100_V50_HE_1	13893237	**13864911**	-0.204
18	SHORTSEA_FUN_C100_V50_HE_3	13206559	**13189976**	-0.126
19	SHORTSEA_FUN_C100_V50_HE_4	14936198	**14921380**	-0.099
20	SHORTSEA_FUN_C100_V50_HE_5	14106741	**14059978**	-0.331
21	DEEPSEA_MUN_C60_V13_HE_1	81709586	**81518899**	-0.233
22	DEEPSEA_MUN_C80_V20_HE_3	78918099	**78687219**	-0.293
23	DEEPSEA_MUN_C100_V30_HE_2	154533570	**153998958**	-0.346
24	DEEPSEA_MUN_C100_V30_HE_4	157017186	**157010893**	-0.004
25	DEEPSEA_MUN_C130_V40_HE_1	239877031	**239326604**	-0.229
26	DEEPSEA_FUN_C50_V20_HE_1	41398100	**41377030**	-0.051
27	DEEPSEA_FUN_C50_V20_HE_2	37872273	**37870753**	-0.004
28	DEEPSEA_FUN_C50_V20_HE_3	39916853	**39910778**	-0.015
29	DEEPSEA_FUN_C70_V30_HE_1	142923793	**142813482**	-0.077
30	DEEPSEA_FUN_C70_V30_HE_4	156541043	**156449624**	-0.058
31	DEEPSEA_FUN_C90_V40_HE_4	211046180	**210828532**	-0.103
32	DEEPSEA_FUN_C100_V50_HE_1	207105715	**206778407**	-0.158
33	DEEPSEA_FUN_C100_V50_HE_3	218438412	**217835593**	-0.276
34	DEEPSEA_FUN_C100_V50_HE_4	221248187	**221112912**	-0.061
35	DEEPSEA_FUN_C100_V50_HE_5	224430601	**223341048**	-0.485

According to Table 1 and Figure 2, not only our proposed method decreased the average gaps especially in large instances but also improved convergence time for the optimal solution. As you can see, as the number of cargoes increases, the convergence time for the optimal solution increases exponentially. In this environment, the use of operators to generate solutions gradually can lead to a prolonged implementation process and slow convergence. We also observed the effect of exploring problem space with insertion operators in the present study. This was done by the initial phase of the GRASP algorithm in the form of inserting cargoes into the routes. Figure 2 displays the number of instances solved by GRASP and ALNS as a function of the CPU time limit. GRASP visibly produces superior results, its performance depends on the ability to do a complete route enumeration at the root node within the optimality gap. The difference is that in large size cargo instances are easier for GRASP to solve, finding optimal solutions of instances with up to 60 cargoes. Among the other instances with 7-35 cargoes, the ALNS and GRASP have equal execution time.

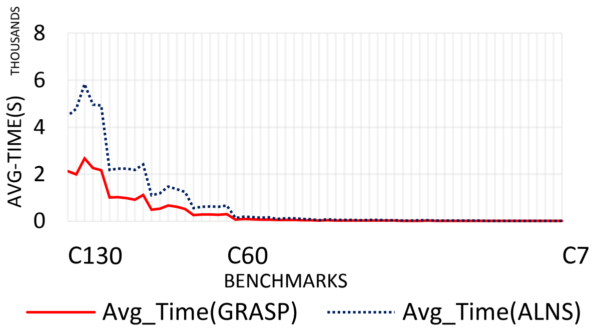

Figure 2. Convergence time comparison between GRASP and ALNS.

GRASP has been widely used for finding good quality solutions in high variety to many hard combinatorial optimization problems. As the runtime exponentially increases as the instance size grows, GRASP not only succeeded in producing 35 quality samples but also reduced the time to find the optimal solution compared with the ALNS.

5. CONCLUSION

As demonstrated in this paper, the best method that has been published for solving the vessel routing problem with pickup and delivery (VRPPDPs) was an ALNS heuristic. Our proposed method was the GRASP algorithm which could improve the results in 35 instances. In the ALNS method, removal operators played the role of randomization and insertion operators were used for greediness. The adaptive weight factor was used to make a tradeoff between the randomization and greediness. These properties are clearly present in the GRASP algorithm, which performs both randomization and greediness using both phases and performs adaptive weigh with a strict candidate list. According to the literature in [4], the authors underlined the importance of the acceptance criterion in finding optimal solutions. Also, after fine-tuning the parameters in the acceptance criterion in ALNS, we used this factor to accept solutions in the GRASP algorithm. Statistical analysis revealed that our proposed method improved the quality and convergence of the solutions compared to the ALNS algorithm [4]. The results indicate that the proposed method improved the 35 instances from the large size of the cargoes.

Future works include using the novel metaheuristics for improving the sequence of cargoes in other problems that are almost similar to the VRPPDPs.

REFERENCES

[1]. I. Review, “Cargo ships routing and scheduling : Survey of models and problems,” no. 4, 1983.

[2]. A. Hemmati and L. Magnus, “Evaluating the importance of randomization in adaptive large neighborhood search,” vol. 24, pp. 929–942, 2017.

[3]. A. Hemmati, L. M. Hvattum, K. Fagerholt, and I. Norstad, “Benchmark Suite for Industrial and Tramp Ship Routing and Scheduling Problems,” vol. 52, no. 1, pp. 28–38, 2014.

[4]. G. Br, M. Christiansen, K. Fagerholt, and Z. Bj, “A multi-start local search heuristic for ship scheduling — a computational study,” vol. 34, pp. 900–917, 2007.

[5]. C. Science, “Local search in routing problems with time windows,” vol. 4, pp. 285–305, 1985.

[6]. M. G. C. Resende, “Greedy Randomized Adaptive Search Procedures,” no. March 1995, 2014.

[7]. S. Ropke, D. Pisinger, S. Ropke, and D. Pisinger, “An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows,” no. October 2014, 2006.