Evolutionary Algorithm for the Minimum Cost Hybrid Berth Allocation Problem

Nataša Kovač
Maritime Faculty
University of Montenegro
Kotor, Montenegro
Email: knatasa@ac.me

Tatjana Davidović
Mathematical Institute
Serbian Academy of Sciences and Arts
Belgrade, Serbia
Email: tanjad@mi.sanu.ac.rs

Zorica Stanimirović
Faculty of Mathematics
University of Belgrade
Belgrade, Serbia
Email: zoricast@matf.bg.ac.rs

***Abstract*—A new optimization method based on the Evolutionary Algorithm (EA) is developed for solving the Minimum Cost Hybrid Berth Allocation Problem (MCHBAP) with fixed handling times of vessels. The goal of the MCHBAP is to minimize the total costs of waiting and handling, as well as earliness or tardiness of completion, for all vessels. It is well known that this kind of problem is NP hard. The main problem one faces when dealing with the MCHBAP is a large number of infeasible solutions. In order to overcome this problem, we propose an EA implementation adapted to the problem that involves four types of mutation operator and two additional improvement strategies, but no crossover operator. The proposed EA implementation is benchmarked on real life test instances. Our computational results show that the proposed EA method is able to find optimal solutions for real life test instances within relatively short running time, having in mind the nature of the considered problem.**

I. Introduction

Berth Allocation Problem (BAP) involves assigning a berthing position and a berthing time to each incoming vessel to be served within a given time horizon with an aim to minimize some objective [2]. Vessels are represented by a set of data containing the expected time of arrival, the size, anticipated handling time, a preferred berth in the port, and penalties. In [16], it was proven that BAP is a NP-hard problem.

In the recent literature BAP has been widely studied [8], [14], [21], [25]. Different authors considered diverse objectives to be optimized within the BAP solution: the total waiting and handling times were minimized in [1], [12], the total costs for waiting and handling as well as earliness or delay in completion was minimized in [11]. The solution approaches could also be distinguished: while minority of authors developed exact methods [15], [20], and some heuristic approaches, e.g., Lagrangean relaxation [12], Branch-and-Bound-based heuristic [1], in most of the papers meta-heuristic methods were used (genetic algorithm [19], [24], tabu search [4], variable neighborhood search [11]).

In our work, the minimization of berthing cost as well as the costs of earliness and delay of each vessel in the case of static hybrid BAP is considered. We refer to this variant as Minimum Cost Hybrid Berth Allocation Problem, MCHBAP. The Mixed Integer Linear Programming formulation (MILP) for MCHBAP was proposed in [5] and used with CPLEX 11.2 and MIP-based meta-heuristics that were used to deal with larger test instances. However, due to the formulation complexity it was not possible to obtain good results for the examples with more than 20 vessels. These results showed that meta-heuristic approach would be much more convenient and we suggest to use an Evolutionary Algorithm (EA).

Evolutionary-based methods, especially Genetic algorithms (GA), are very popular as solution approaches to different variants of BAP. One of the first papers in the literature proposes applying GA to BAP is [7]. The authors considered dynamic vessel arrivals and examined several variants of the Randomized Local Search, Tabu Search and Genetic Algorithms. Nishimura *et al.* [19] presented a GA heuristic for the discrete space and dynamic vessel arrival time for berth scheduling problem. Imai *et al.* [13] presented a formulation for the discrete dynamic BAP at a terminal with indented berths. Authors extended GA from [19] with a procedure to obtain feasible solutions. Han *et al.* [10] combined GA with simulated annealing in case of the discrete dynamic berth scheduling problem with the objective of minimizing the total service time of all the vessels. Theofanis *et al.* [24] were the first to present an optimization based GA heuristic for the dynamic berth scheduling problem.

We propose an EA based approach that uses four mutation techniques: Swap, Scramble, Insert and Inversion. We also incorporate two types of improvement techniques on a certain number of individuals. Chromosomes in our EA have integer representation proposed in [24]. For the experimental evaluation we used real life test instances proposed in [3]. The proposed EA is compared against CPLEX and VNDS-MIP (the best performing MIP-based meta-heuristic according to [5]). Our computational results show that the proposed EA outperforms both CPLEX and VNDS-MIP with respect to the solution quality and CPU time.

The rest of this paper is organized as follows. Sect. II contains the description of MCHBAP. Sect. III is devoted to the proposed variants of EA. Experimental evaluation is given in Sect. IV. Sect. V contains some concluding remarks and the directions for future work.

II. Problem description

The input data of MCHBAP are listed below:

l : The total number of vessels;
m : The total number of berthing positions;
T : The total number of time segments in the planning horizon;

$vessel$: The sequence of data describing vessels with the following structure:

$$vessel = \{(ETA_k, a_k, b_k, d_k, s_k, c_{1k}, c_{2k}, c_{3k}, c_{4k})\},\ k = \overline{1, l}.$$

The elements of a 9-tuple $vessel$ represent the following data for each vessel:

ETA_k : The expected time of arrival of $vessel_k$.
a_k : The processing time of $vessel_k$;
b_k : The length of $vessel_k$ (the number of berths);
d_k : The required departure time of $vessel_k$;
s_k : The least-cost berthing location for reference point of $vessel_k$;
c_{1k} : The penalty cost if $vessel_k$ cannot dock at its preferred berth;
c_{2k} : The penalty cost per unit time if $vessel_k$ must arrive before ETA_k;
c_{3k} : The penalty cost per unit time if $vessel_k$ must arrive after ETA_k;
c_{4k} : The penalty cost per unit time if $vessel_k$ is delayed beyond the departure time d_k.

A feasible solution of MCHBAP is subject to two sets of constraints: at a time t, each berth can be assigned to only one vessel (*Constraints 1*), and a berth is allocated to the vessel only between its arrival and departure times (*Constraints 2*).

Let us denote by At_k and Dt_k actual berthing and departure times for a $vessel$, respectively, and let σ_k denotes the difference from the least-cost berthing location of the reference point, $k = 1, \ldots, l$. The aim of MCHBAP is to minimize total penalty cost including: the penalty incurred as a result of missing the least-cost (preferred) berthing location of the reference point; the penalties resulted by the actual berthing earlier or later than the expected time of arrival and the penalty cost induced by the delay of the departure after the promised due time. The last three terms influence the objective function in case they are positive. More accurately, the objective function (1) can be expressed as follows:

$$\sum_{k=1}^{l}(c_{1k}\sigma_k + c_{2k}(ETA_k - At_k)^+ + c_{3k}(At_k - ETA_k)^+ + \\ + c_{4k}(Dt_k - d_k)^+), \quad (1)$$

where

$$\sigma_k = \sum_{t=1}^{T}\sum_{i=1}^{m}\{|i - s_k| : \text{vessel } k \text{ occupies position } (t, i)\}, \quad (2)$$

and

$$(a-b)^+ = \begin{cases} a - b, & \text{if } a > b, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

The detailed MILP formulation can be found in [5].

III. EA FOR MCHBAP

We have developed EA approach for solving MCHBAP that uses integer encoding of individuals. The proposed EA involves four mutation types and an additional improvement step that consists of two types of improvement techniques on a certain number of individuals. The crossover operator is omitted in order to better deal with the problem of avoiding infeasible individuals. In the following subsections all aspects of the proposed EA will be explained in details.

A. Representation of individuals

Each individual is represented as a list of m sublists, where each sublist corresponds to one berthing position. The elements of sublists are indices of the vessels allocated to a berthing position and sorted in order in which they are to be scheduled. The total number of elements through all m sublists is equal to l, which represents the length of individual (i.e., the total number of vessels to be scheduled).

Let us consider an example of the problem with $m = 8$ berths and $l = 5$ vessels to be scheduled within the time horizon of $T = 15$ units described by the data given in Table I.

TABLE I. 9-TUPLE VALUES FOR EXAMPLE WITH $m = 8, l = 5$ AND $T = 15$

$vessel_k$	ETA_k	a_k	b_k	d_k	s_k	c_{1k}	c_{2k}	c_{3k}	c_{4k}
1	1	9	3	4	1	10	20	20	25
2	4	6	3	6	3	10	20	20	25
3	4	14	2	11	6	10	20	20	25
4	5	14	2	12	7	10	20	20	25
5	6	18	2	16	2	10	20	20	25

A typical individual corresponding to the feasible solution of this problem could be represented as:

$$\text{Ind\_1:}\{\underbrace{\{1\}}_{berth\ 1}, \underbrace{\{5,2\}}_{berth\ 2}, \underbrace{\{\}}_{berth\ 3}, \underbrace{\{\}}_{berth\ 4}, \underbrace{\{3\}}_{berth\ 5}, \underbrace{\{\}}_{berth\ 6}, \underbrace{\{4\}}_{berth\ 7}, \underbrace{\{\}}_{berth\ 8}\}$$

The meaning of this representation is that berth **1** has to serve vessel indexed by 1. Berth **2** serves vessels 5 and 2, while berths **3**, **4**, **6**, and **8** are empty. Ship 3 is served on berth **5** while berth **7** serves vessel 4.

The resulting allocation is illustrated in Fig. 1. Ship is the member of the k-th list if its reference point (lower left corner of the corresponding rectangle in the two dimensional plane) belongs to the berth with index k. Note that a vessel may also allocate some space within other adjacent berths, since we consider the hybrid case of BAP.

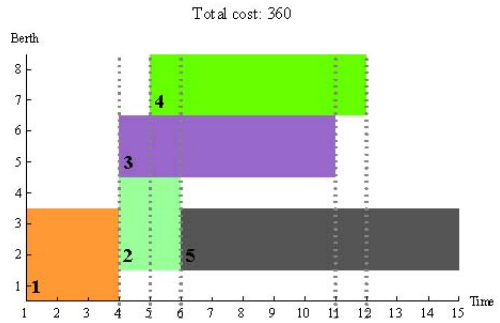


Fig. 1. Decoded solution for Ind_1

Ships 5 and 2 are served on same berth, however, the order in individual doesn't correspond to the order on time axis. Since vessel 5 is allocated to port before vessel 2, it is placed on the cheapest possible location (which is not the first empty location on berth 2). Then, vessel 2 is allocated at the cheapest location available after vessel 5 occupies its own place in the port. As it can be seen in this example, berth 3 remains empty in the representation of this individual, since

there is no reference point of any vessel assigned to berth 3. However, since we deal with HBAP, berth 3 is not free, it is occupied with some parts of vessels 1, 2 and 5. It is worthy to note that Ind_1 and the corresponding schedule presented in Fig. 1 represent the optimal solution for this example.

The individual is denoted as *infeasible* if its representation leads to the overlapping of vessels in two dimensional plane, or if the length of all vessels on the same berth exceeds the planning time horizon T. Since the length of vessels, i.e., the length of corresponding rectangles is *a priori* known, it is possible to check if a berth can handle an unallocated vessel, and the *infeasible* individual can be discarded. Feasible individuals can be decoded in the MCHBAP solution where 3-tuples (berth, time, cost) corresponds to the reference point of vessels in the two dimensional plane.

B. Generating initial EA population

Our EA starts with the procedure of forming nEA individuals for the initial population. The number of individuals in the EA population (nEA) is one of the input parameters for the EA. As a part of the initialization, for all vessels and for all possible positions of the vessel in 2-dimensional plane the list of 3-tuples elements (berth, time, penalty cost) is created.

For each of nEA individuals in the population, a list of 3-tuples elements (berth, time, penalty cost), denoted as ξ list, is created. During the EA run, ξ lists of individuals are updated after the allocation of vessels in the port. The pseudo-code of procedure for generating initial population is shown in Fig. 2.

```
Initialize(nEA)
// Generates initial population with nEA individuals
Begin
    UnusedShips ← {1, 2, ..., l};
    While UnusedShips ≠ ∅ do
        ID=Roulette (UnusedShips,selectionCriteria);
        Berths=Determine all berths with enough
          free space for vessel ID;
        berthID=TournamentForBerths(size,Berths);
        Insert vessel ID on last position on berthID;
        UnusedShips ← UnusedShips\ {ID};
    End;
End;
```

Fig. 2. Procedure for the generation of the initial population

In order to generate a new individual, the procedure needs to decide on the order of vessels selection and on which berth to place each vessel. Criterion for vessel selection is a linear combination of ETA parameter, the size of the corresponding rectangle associated to the vessel in 2-dimensional plane, and the calculated average cost of all possible ξ list elements for the observed vessel. Coefficients with the parameters in this linear combination represent the impact of a parameter on the vessel selection criterion. The values of these coefficients were determined experimentally and set to 0.12, 0.13, 0.75, respectively.

Among all unused vessels, the procedure is choosing one by the roulette wheel selection based on calculated priorities of the vessels. Since the length of all vessels is known in advance, the subset of berths with enough free space to handle the selected vessel can be determined. In the next step, the procedure is resolved which berth should be associated to the selected vessel. The berth is chosen by tournament selection among all possible berths with enough free space for the examined vessel. Once the vessel and its associate berth are selected, the vessel is placed as the last one on the given berth. The index of a vessel is included as the last element into the corresponding sublist of the associate berth. The procedure runs until all vessels are allocated to the berths.

C. EA operators

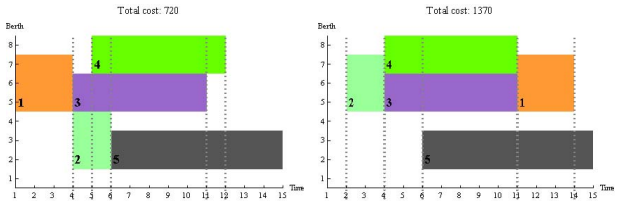
One third of individuals from the EA population directly passes to the next generation, preserving highly fitted genes of the population. In order to select remaining $2/3 * nEA$ individuals that will take part in creating new EA generation, a fine-grained tournament selection with two tournament types is implemented [9]. More precisely, $40\% * (2/3nEA)$ individuals are selected though tournaments of $size_1$, while remaining individuals are obtained as winners of tournaments of $size_2$, with $size_1 < size_2$. Applied fine grained tournament selection ensures that weaker individuals have better chance to be selected. In addition, the selection operator disables duplicated individuals to enter the next generation. This strategy helps in preserving the diversity of genetic material and in preventing the algorithm to converge to a local optimum.

In the literature, different concepts of EA are proposed with different variation of operators involved. Some of the existing EA approaches include crossover operator [17], [23], while other incorporate only mutation as variation operator [6], [22]. In our EA implementation, we do not use crossover as variation operator. The main reason for this decision was the fact that different types of crossover operators that we tried to implement produced too many infeasible individuals. Discarding infeasible individuals from the EA population, results in significant decrease of the population size, while various repair techniques that we have tried in order to restore feasibility were inefficient due to the huge CPU time they required. In addition, the quality of the final solution was worse than in the case of the EA without crossover. For these reasons, we propose EA that involves mutation only. Four types of mutation operators adapted to the problem are implemented: Insert, Inversion, Scramble and Swap. Each individual is a subject of all four mutations, producing maximally four offsprings. All types of mutations allow that the selected vessels change both berth and time coordinate. Therefore, they can produce large changes in the considered individual and the significant change in the objective function value.

Insert mutation picks two vessels (genes) at random and moves the second one to follow the first (in Fig. 3 vessels 3 and 2 are selected and vessel 2 is moved behind the vessel 3).

Inversion mutation picks two vessels at random and then inverts the substring between them. The illustration given in Fig. 4 shows the change in the individual produced by inverting part of the chromosome between vessels 2 and 1 (including these two vessels).

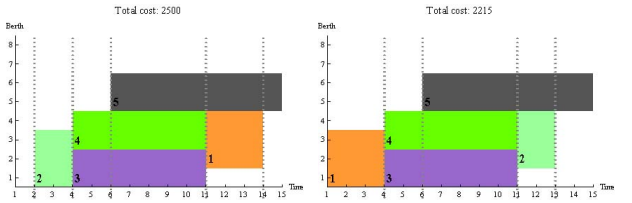
Swap mutation picks two vessels from a chromosome and swaps their positions. Fig. 5 illustrates the application of the swap mutation type to the vessels 4 and 3 in a feasible individual from the above mentioned example.



Initial individual: {{},{**2**,5},{},{},{**3**,1},{},{4},{}}

Resulting individual: {{},{5},{},{},{3,2,1},{},{4},{}}

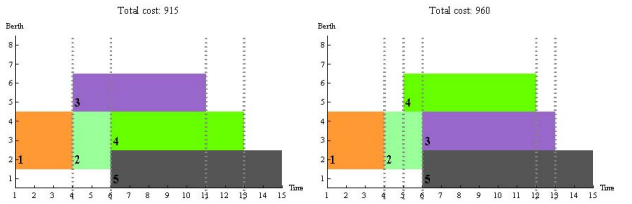
Fig. 3. Illustration of insert mutation type



Initial individual: {{**2**,3},{**1**},{4},{},{5},{},{},{}}

Resulting individual: {{1,3},{2},{4},{},{5},{},{},{}}

Fig. 4. Illustration of inversion mutation type

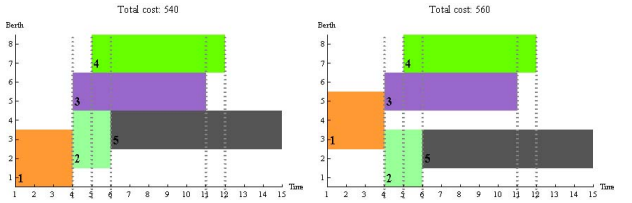


Initial individual: {{5},{1,2},{**4**},{},{**3**},{},{},{}}

Resulting individual: {{5},{1,2},{3},{},{4},{},{},{}}

Fig. 5. Illustration of swap mutation type

Scramble mutation scrambles the position of a subset of vessels in the chromosome. This actually means that a random permutation of the selected subset of vessels is generated. The subsets influenced by the mutations can be anywhere in the individual (the ending points can belong to different sublists) and sometimes, the mutations may even affect the whole individual. The illustration of scramble mutation is given in Fig. 6. The scrambled is the part between vessels 1 and 5, including these two vessels.



Initial individual: {{**1**},{2},{**5**},{},{3},{},{4},{}}

Resulting individual: {{2},{5},{1},{},{3},{},{4},{}}

Fig. 6. Illustration of scramble mutation type

Mutation probability μEA is assumed to be variable, and it changes in each generation. As suggested in [24], during the EA run the weight is shifted from the coarse type Inverse and Scramble mutations to the Insert and Swap mutations. In this way, we allow the algorithm to perform large jumps at the beginning of the search while, as the objective function improves, we concentrate on the solutions from a smaller region. The probability to apply Scramble and Inversion mutation on some gene is decreasing according to formula

$$\mu EA = 1 - 0.9 * (Index\_of\_Gener)/(Max\_Num\_Gener)$$

while probability of Insert and Swap mutations are increasing and calculated as follows

$$\mu EA = 0.9 * (Index\_of\_Gener)/(Max\_Num\_Gener).$$

The formulae above are determined through the set of preliminary experiments.

Mutation operators are performed on randomly selected pair of genes in the chromosome. Each individual is a subject of all four mutations with a given probability. Only one randomly selected gene within each individual is considered as a subject of the mutation operator with given probability μEA. After the mutation phase is completed, some individuals are selected for additional improvement strategy. The number of individuals to be improved in each generation is the input parameter of our EA. The individuals to be improved are chosen by tournament selection. The size of tournament performed for selecting individuals to be improved is also an input parameter of EA. Selection pressure is easily adjusted by changing the tournament size. If the tournament size is larger, weak individuals have a smaller chance to be selected.

Applied improvement procedure tries to optimize the allocation of vessels defined by the selected individual. Ships are examined one by one, as they are listed within individual, and they are placed on the cheapest possible position on the given berth. When all vessels are allocated within their berths, the procedure sorts vessels in the descending order of their costs in given allocation. Ships are now taken one by one according to this new order, and associated lists are examined. If there is a cheaper position, the corresponding vessel is moved there, even if that causes the change of the berth for this vessel. This procedure is applied to all l vessels. In addition, due to the selection of cheaper positions for some of the vessels, the order of service of vessels on berths can be changed. Therefore, this part of the improvement process is completed by performing the corresponding changes in the representation of the improved individual.

Finally, each improved individual is subject to a local search procedure. The applied local search algorithm examines if it is possible to reduce the total cost by rearranging the order of vessels on a single berth. In this phase, it is not allowed to create some new conflicts by the perturbation of vessels. i.e., vessels can change their positions only if that new allocation is not producing any conflict with vessels on adjacent berths.

After the improvement procedure is finished, individuals are evaluated in means of calculating their fitness values in order to enter the selection process for creating the next generation. The steps described above are iteratively applied until one of the EA stopping criteria (time limit or maximal number of generations) is met. The pseudo-code of our EA is presented in Fig. 7.

```
EAforMCHBAP()
Begin;
    Initialize(nEA);
    popID=1;
    globalBest=Infinity;
    minT=Infinity;
    While (SessionTime ≤ RunTime) && (popID ≤ maxGen) do
        For each individual do
            μGA = 0.9∗popID/maxGen;
            newindividual1← InsertMutation(noGenes,μGA);
            newindividual2← SwapMutation(noGenes,μGA);
            newindividual3← InversionMutation(noGenes,1-μGA);
            newindividual4← ScrambleMutation(noGenes,1-μGA);
        End;
        For noImprovements individuals do
            individualID = TournamentForImprovement(size1, cost);
            newindividualI= Improve(IndividualID);
            newindividualI=BerthLocalSearch(newIndividualI);
        End;
        Calculate Cost for each individual;
        If bestCost(popID)<globalBest then
            Update globalBest;
            Update minT;
        End;
        eliteNO= nEA/3;
        Copy best eliteNO individuals to the new generation;
        size1NO=Round[(nEA-elitNO)*0.4];
        Choose size1NO individuals through tournaments of size1;
        size2NO=nEA-elitNO-size1NO;
        Choose size2NO individuals through tournaments of size2;
        popID ← popID + 1;
    End;
End;
```

Fig. 7. Optimization Based EA for MCHBAP

IV. Experimental Evaluation

In our computational experiments, we used the set of real life instances for BAP proposed in [3]. These real-life instances are generated from the example that involves 21 vessels, 12 berths and the time horizon of 54 units [3]. This example is further extended by adding new vessels; up to 28 (see Fig. 8).

The proposed EA approach is coded in the Wolfram Mathematica v8.0 programming language. All tests were conducted on a computer with an Intel Pentium 4, with 3.00-GHz CPU and 512 MB of RAM running the Microsoft Windows XP Professional Version 2002 Service Pack 2 operating system. It is important to note here that Wolfram Mathematica v8.0 interprets instructions which may prolong the execution time of the algorithm. If faster execution is required, it is necessary to apply the converter that transforms program into C, Java, or some other conventional programming language.

The set of preliminary computational experiments is performed in order to determine the values of EA parameters that lead to the best results of the algorithm. Finally. the following setting is adopted. Number of individuals in population nEA is set to 20, while the maximal number of generations, i.e., max_Number_Generations is 40. Smaller tournament size, $size_1$ is equal to 3, and larger tournament size, $size_2$ has the value of 5. Probability of mutation μEA depends on the index of population as it is described in previous section. The number of individuals to be improved in each generation is set to 5.

The proposed EA approach is compared with commercial CPLEX solver version 11.2 and VNDS-MIP meta-heuristic from [5]. Time limit for EA is set to 10 minutes of CPU time, and the EA was run 10 times on each test example. The summary of EA results and comparisons with CPLEX and VNDS-MIP method are presented in Table II.

TABLE II. Computational results on real-life test examples: $m = 12, T = 54$

	EA		CPLEX		VNDS-MIP		T.Lim.
l	COST	Time	COST	Time	COST	Time	(sec.)
21	4779	196.35	24562	3698.41	15424	494.30	3600
22	4983	119.44	16334	7434.44	10108	961.13	7200
23	5193	189.85	96549	7404.73	14110	1207.77	7200
24	5643	109.25	6594	7429.48	9521	1609.55	7200
25	5953	156.82	13262	18709.60	91620	1569.54	18500
26	6298	156.12	26614	18716.10	77439	24742.42	18500
27	6478	188.95	26679	18638.50	69143	3215.10	21600
28	6980	324.74	8418	44530.70	63249	7779.35	43200
av.	5788.38	180.19	27376.5	15820.24	43826.75	5197.39	15750

Table II is organized as follows. Number of vessels is presented in the first column. Columns 2 and 3 contain the average best objective function value and the corresponding average CPU time obtained by 10 EA runs each lasting 10 minutes. The best found cost obtained by CPLEX solver within the given time limit and the required CPLEX running time are given in columns 4 and 5. The best objective value and the average required CPU time of the VNDS-MIP heuristic are presented in columns 6 and 7. The specified CPLEX and VNDS-MIP time limit for each example is presented in the last column of Table II.

From the presented results it can be seen that for all tested instances, the proposed EA approach showed to be superior comparing to VNDS-MIP and CPLEX with respect to both solution quality and CPU time. The EA produced solutions with significantly lower objective values in each of the 10 runs. In average, the best cost obtained by EA was around 4.7 times lower compared to the cost produced by CPLEX, and 7.6 lower than the best cost reached by VNDS-MIP heuristic.

The average CPU time of the proposed EA was shorter compared to corresponding average CPU times of both CPLEX and VNDS-MIP method. Note that EA is coded in Wolfram Mathematica v8.0 programming language and tested on slower computer under Microsoft Windows operating system with interpreter, while CPLEX and VNDS-MIP are written in C++ with optimized code and compiled and executed under Linux operating system on much faster computer. Even then, the EA was approximately 87.8 times faster than CPLEX and approximately 28.8 times faster then VNDS-MIP. From these preliminary results it can be concluded that EA produces promising results in testing on real life BAP data sets.

It is worth noting that EA actually produced optimal solution in all 10 runs. Contrary to the MIP based solvers, combinatorial exact algorithm developed in [15] is able to solve these test examples within reasonable CPU time. In fact, it is faster for instances with up to 27 vessels, but EA required less CPU time to find optimal solution for the largest example. This conclusion supports the need to develop heuristic approaches for MCHBAP.

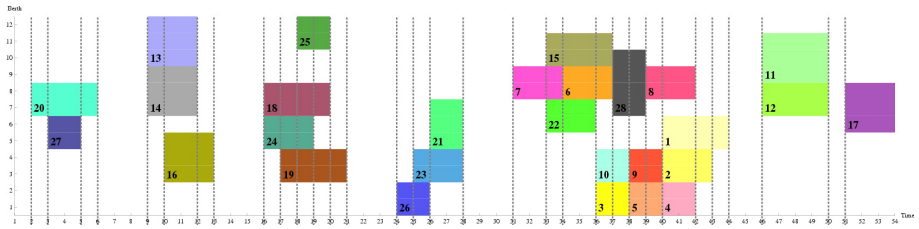


Fig. 8. An example of real–life MCHBAP solution

V. Conclusion

Evolutionary based algorithm (EA) to solve Minimum Cost Hybrid Berth Allocation Problem with fixed handling times of vessels (MCHBAP) is presented. We proposed EA based only on mutation of genes in the individuals, involving four types of mutations. In addition, we proposed optimization steps after mutations in order to improve few individuals in each generation. Two different optimizations where developed for chosen individuals: the first one allows changing the associated berth of vessel while the second performs local search within berth, and allows only perturbations of vessels order within the chosen berth. As future work, more exhaustive experimental evaluation on both real and artificially generated random examples are to be performed, in order to additionally evaluate the proposed EA. Moreover, the comparison with some other meta-heuristic methods, especially with the neighborhood based ones, is also needed.

References

[1] C. Bierwirth, F. Meisel, A fast heuristic for quay crane scheduling with interference constraints, *Journal of Scheduling*, 12 (4), 2009, pp. 345–360.

[2] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, European Journal of Operational Research 202 (2010) 615–627.

[3] D. Changa, Z. Jiang, W. Yan, J. He, Integrating berth allocation and quay crane assignments, *Transportation Research Part E*, 46 (6), 2010, pp. 975–990.

[4] J.-F. Cordeau, G. Laporte, P. Legato, L. Moccia, Models and tabu search heuristics for the berth-allocation problem, *Transportation Science*, 39 (4), 2005, pp. 526–538.

[5] T. Davidović, J. Lazić, N. Mladenović, S. Kordić, N. Kovač, B. Dragović, Mip-heuristics for minimum cost berth allocation problem, *In: Proc. International Conference on Traffic and Transport Engineering, ICTTE 2012, Belgrade, Serbia*, 2012, pp. 21—28.

[6] I. De Falco, A. Della Cioppa, E. Tarantino, Mutation-based genetic algorithm: performance evaluation, *Applied Soft Computing*, 1 (4), 2002, pp. 285—299.

[7] K.-S. Goh, A. Lim, Combining various algorithms to solve the ship berthing problem, *In: 12th IEEE Int. Conf. on Tools with Artificial Intelligence, IEEE, 2000, pp. ICTAI 2000*, 2000, pp. 370—375.

[8] Y. Guan, R. K. Cheung, The berth allocation problem: models and solution methods, *OR Spectrum*, 26 (1), 2004, pp. 75–92.

[9] V. Filipović, Fine-Grained Tournament Selection Operator in Genetic Algorithms, *Computing and Informatics*, 22, 2003, pp. 143—161.

[10] M. Han, P. Li, J. Sun, The algorithm for berth scheduling problem by the hybrid optimization strategy gasa, *In: 9th Int. Conf. Control, Automation, Robotics and Vision, IEEE, 2006, pp. ICARCV'06*, 2006, pp. 1—4.

[11] P. Hansen, C. Oğuz, N. Mladenović, Variable neighborhood search for minimum cost berth allocation, *European Journal of Operational Research*, 191 (3), 2008, pp. 636–649.

[12] A. Imai, E. Nishimura, S. Papadimitriou, The dynamic berth allocation problem for a container port, *Transportation Research Part B*, 35, 2001, pp. 401–417.

[13] A. Imai, E. Nishimura, M. Hattori, S. Papadimitriou, Berth allocation at indented berths for mega-containerships, *European Journal of Operational Research*, 179 (2), 2007, pp. 579—593.

[14] K. H. Kim, K. C. Moon, Berth scheduling by simulated annealing, *Transportation Research Part B*, 37 (6), 2003, pp. 541–560.

[15] S. Kordić, B. Dragović, T. Davidović, N. Kovač, Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, (submitted for publication).

[16] A. Lim, The berth planning problem, *Operations Research Letters*, 22 (2), 1998, pp. 105–110.

[17] M. Marić, Z. Stanimirović, Z. P. Stanojević, An efficient memetic algorithm for the uncapacitated single allocation hub location problem, *Soft Computing*, 17 (3), 2013, pp. 445—466.

[18] F. Meisel, C. Bierwirth, Heuristics for the integration of crane productivity in the berth allocation problem, *Transportation Research Part E*, 45, 2009, pp. 196–209.

[19] E. Nishimura, A. Imai, S. Papadimitriou, Berth allocation planning in the public berth system by genetic algorithms, *European Journal of Operational Research*, 131, 2001, pp. 282–292.

[20] C. Oğuz, Ö. Narin, Solving berth allocation problem with column generation, *In: Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009), Dublin, Ireland*, 2009, pp. 744—747.

[21] Y. M. Park, K. H. Kim, A scheduling method for berth and quay cranes, *OR Spectrum*, 25 (1), 2003, pp. 1–23.

[22] C. Prodhon, A hybrid evolutionary algorithm for the periodic location-routing problem, *European Journal of Operational Research*, 210, 2011, pp. 204–212.

[23] Z. Stanimirović, M. Marić, S. Božović, P. Stanojević, An Efficient Evolutionary Algorithm for Locating Long-Term Care Facilities, *Information Technology and Control*, 41 (1), 2012, pp. 77—89.

[24] S. Theofanis, M. Boile, M. Golias, An optimization based genetic algorithm heuristic for the berth allocation problem, *In: IEEE Congress on Evolutionary Computation, pp. CEC 2007*, pp. 4439—4445.

[25] F. Wang, A. Lim, A stochastic beam search for the berth allocation problem, *Decision Support Systems*, 42, 2007, pp. 2186–2196.