# An Exponential Monte-Carlo Local Search Algorithm for the Berth Allocation Problem

**Nasser R. Sabar • Masri Ayob • Graham Kendall**

## 1. Introduction

Over the years, the demand for maritime transportation has rapidly increased [1], [2]. This usually leads to an increasing in competition among different ports in providing efficient services. Port managers face significant challenges in efficiently utilizing the given resources to provide a cost-effective service [1]. Among many optimization and decision making problems in the port management system, the berth allocation problem (BAP) is considered as one of the most challenging and it has a critical role in the ports effectiveness and competitiveness. The BAP seeks an allocation for a given set of vessels berthing positions and berthing times. The main objective of the BAP is to minimize the total waiting time for all vessels in a port [2].

BAP is known to be an NP-hard optimization problem [1], [2], [3]. Thus, due to the exponential growth of the computational time as instance increases, exact methods are usually only applicable for the small-sized instances; despite being able offer optimal solutions if given enough computational resource [3]. Consequently, meta-heuristic algorithms are widely adopted by the researchers to deal with BAP, as they can often return a good quality solution within a reasonable computational time. Examples of meta-heuristic algorithms are: tabu search [2], clustering search [4] and particle swarm optimization [5]. In this work, we propose an Exponential Monte-Carlo with Counter (EMCQ) local search algorithm for the BAP. EMCQ is a variant of simulated annealing, that accepts worse solutions in order to escape from local optima using a non-monotonic acceptance criterion [6], [7]. In addition, to enhance the effectiveness of the proposed EMCQ, we utilize multi-neighborhood operators to effectively explore the search space and also deal with different instance characteristics. The proposed algorithm has been tested on BAP benchmark instances that were used by other researchers and compared with the best known results in the scientific literature [2].

## 2. Problem description

The BAP has been categorized into two types, based on the berth type and the vessels arrival time [1]. The berth is called a discrete berth if the quay is divided into a set of sections (berths) and a continuous berth if the quay is not divided. The vessel's arrival time is dynamic if the vessels can arrive at any time over the planning horizon, and static if all vessels have arrived at the port before the berth planning starts. In this work, we deal with the BAP that has discrete berths and dynamic arrival times [1]. Given a set of berths and a set of vessels, each vessel is associated with an arrival time, priority and a handling time. Some vessels can be assigned to any berths while other can only be assigned to a subset of berths. The handling time of a vessel is different from one berth to another.

More formally, assign for each vessel a berth and a berthing time on the selected berth while ensuring that each vessel is assigned to exactly one berth and there is no more than one vessel assigned to the same berth at the same time. The overall goal (objective function) is to minimize the total waiting time of all vessels which is calculated as follows [2], [3], [4]:

N. R. Sabar • G. Kendall
The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih Selangor, Malaysia.
E-mail: {Nasser.Sabar, Graham.Kendall}@nottingham.edu.my

M. Ayob
Data Mining and Optimization Research Group (DMO), Centre for Artificial Intelligent Technology (CAIT), Universiti Kebangsaan Malaysia, 43600 UKM Bangi Selangor, Malaysia.
E-mail: masri@ftsm.ukm.my.

$$\min \sum_{i \in n} \sum_{k \in m} v_i \left[ T_i^k - a_i + + t_i^k \sum_{j \in n} x_{ij}^k \right] \quad (1)$$

where

- $n$ : number of vessels
- $m$ : number of berths
- $v_i$ : the priority of vessel $i$
- $T^k_i$ : the berting time of a vessel $i$ at berth $k$.
- $a_i$ : the arrival time of vessel $i$.
- $t_i^k$ : the handling time of vessel $i$ at berth $k$.
- $x_{ij}^k$ : descion variable, $x_{ij}^k = 1$ if vessel $j$ is serviced by berth $k$ after the vessel $i$.

## 3. The proposed algorithm

We propose a local search algorithm for the BAP that follows the general framework of most local search algorithms. That is, generates an initial solution and try to improve it. In the following subsections, we describe the initial solution generation method and the proposed improvement algorithm.

### 3.1 Initial solution generation method

The initial solution is generated in a random manner. For each vessel, determine the number of the available and feasible berths. Next, randomly assign vessel to a berth from the determined set of berths. If the selected berth for the current vessel is empty, the berthing time of this vessel is the same as the vessel arrival time. If selected berth has some vessels, add the current vessel, sort the assigned vessels based on their arrival time and assign to each one a berthing time based on the current vessels order. This process is repeated until all vessels have been allocated. Calculate the quality (objective function) of the generated initial solution using Equation (1).

### 3.2 The improvement algorithm

In this work, we utilize the Exponential Monte-Carlo with counter (EMCQ) local search algorithm to further improves the generated initial solution [6], [8]. The EMCQ search strategy is similar to simulated annealing [6], which also accepts worse solutions in order to escape from a local optima but utilizes a different mechanism. EMCQ starts with an initial solution and iteratively modifies it, seeking for a better solution, for a certain number of iterations. The initial solution is modified to generate a neighborhood solution using a neighborhood operator. Then, the quality of the neighborhood solution is calculated using Equation (1) and compared with the initial one. If the quality of the neighborhood solution is better than the initial solution, it will replace the initial solution. Otherwise, the solution might be accepted based on EMCQ acceptance criterion. In EMCQ, the probability of accepting worse solution is calculated as follows: $p= e^{-\Theta/\lambda}$ where $\Theta=\delta*t$, $\lambda=q$, $\delta$ is the difference between the objective values of the initial and neighborhood solutions, $t$ is an iteration counter, and $q$ is a control parameter that represents consecutive non-improving iterations. The $q$ parameter controls the acceptance of worse solutions which controls the diversification and intensification process. In this work, the initial value of $q$ is set to 1 ($q$=1) and it will be increased by one ($q$=$q$+1) after a certain number of a consecutive non-improving iterations [9], [10]. Once a worse solution is accepted, $q$ will be reset to 1. In EMCQ, the probability of accepting a worse solution decreases as the number of iterations $t$, increases. However, if there is no improvement for a certain number of consecutive iterations, then the probability of accepting a worse solution will

increase according to the quality of the generated neighborhood solution, $q$ and $t$. The pseudo-code of the EMCQ is presented in Figure 1 [6], [11].

To improve the effectiveness of the EMCQ in dealing with various instances and also to cope with search landscape changes, we utilize three different neighbourhood operators; where at each iteration of EMCQ a random neighbourhood operator is selected. The utilized neighbourhood operators for the BAP are:

- $Nop_1$: select one vessel at random and move it to the least cost berth (least handling time).
- $Nop_2$: randomly select two vessels and swap their berths if feasible.
- $Nop_3$: select the highest cost vessel and move it to another feasible berth.

```
Generate initial solution, Sol;
Calculate the objective function for Sol, f(Sol);
Set best solution, Sol_best ← Sol; f(Sol_best) ← f(Sol);
Set maximum consecutive number of non-improvement, Max_no-improvement;
Set no-improvement counter ← 0;
Set q = 1; current iteration counter, t = 0, Max_no_iteration;
Do while (t < Max_no_iteration)
    Generate a neighborhood solution, Sol*
    Calculate the objective function of neighborhood solution, f(Sol*);
    if (f(Sol*) < f(Sol))   // better solution in term of the objective value
        Sol ← Sol*;
        f(Sol) ← f(Sol*);
        q = 1;
        no-improvement = 0;
        if (f(Sol*) < f(Sol_best)) // update the best solution
            Sol_best ← Sol*;
            f(Sol_best) ← f(Sol*);
    end if
    else  // accept worse solutions based on the acceptance probability
            Calculate δ = f(Sol*)-f(Sol);
            Generate a random number RandNum in [0,1];
            if (RandNum ≤ e^(-δ*t/q))
                Sol ← Sol*;
                f(Sol) ← f(Sol*);
                q = 1;
                no-improvement = 0;
            else
                no-improvement++;
                if (no-improvement = Max_no-improvement)
                    q++;
                    no-improvement = 0;
     end else
t++;
end while;
Return the best solution, Sol_best and f(Sol_best)
```

Figure 1. The pseudo-code of the EMCQ

### 3.3 Experiments and results

EMCQ was tested on BAP benchmark instances that have been introduced in [2] and widely used by other researchers. The benchmark has 30 different instances (denoted as $i1$ to $i30$); each instance contains 30 vessels and 13 berths. We run EMCQ 31 times for each instance. The EMCQ parameters were set based on a preliminary test as follows: $t$=1, $q$=1, *Max_no_iteration* =1,000,000 and *Max_no-improvement* =1,000. The results obtained by

EMCQ (out of 31 runs) and the current state of the art algorithms reported in the literature are presented in Table 1. For each algorithm, we report, for each instance, the best obtained results (best objective value) and the computational time (seconds). In the table, the third column (Opt.) represents the optimal value for each instance [3] and last row represents the average (Avg.). The best results obtained are shown in bold. In this work, we compare the effectiveness of EMCQ with the following algorithms that have obtained the best known results as represented in the scientific literature:

- Generalized set partition programming (GSPP) [3].
- Tabu search (TS) algorithm [2].
- Column generation (CG) algorithm [12].
- Clustering search (CS) [4].
- Particle swarm optimization (PSO) [5].

The results in Table 1 illustrates that, EMCQ obtained the optimal values for all tested instances, i.e., the best results of the EMCQ are the same as those produced by the GSPP as well as CS and PSO on all tested instances. EMCQ outperforms TS on 18 and CG on 4 instances, while producing the same results as TS and CG on 18 and 24 out of 30 instances, respectively. Considering the computational time, EMCQ outperforms other algorithms on all tested instances (see Table 1) and the average computational time of EMCQ is relatively small (3.82, see last row in Table 1). Overall, the EMCQ has a fewer parameters that need to be tuned in advance compared to other algorithms which indicates that EMCQ is an effective and efficient algorithm for the BAP.

Table 1 The results of EMCQ compared to the state of the art methods

| Inst. | EMCQ | | GSPP | | TS | CG | | CS | | PSO | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Best | Time | Opt. | Time | Best | Best | Time | Best | Time | Best | Time |
| i01 | **1409** | **6.11** | **1409** | 17.92 | 1415 | **1409** | 74.61 | **1409** | 12.47 | **1409** | 11.11 |
| i02 | **1261** | **5.2** | **1261** | 15.77 | 1263 | **1261** | 60.75 | **1261** | 12.59 | **1261** | 7.89 |
| i03 | **1129** | **4.3** | **1129** | 13.54 | 1139 | **1129** | 135.45 | **1129** | 12.64 | **1129** | 7.48 |
| i04 | **1302** | **6.03** | **1302** | 14.48 | 1303 | **1302** | 110.17 | **1302** | 12.59 | **1302** | 6.03 |
| i05 | **1207** | **3.11** | **1207** | 17.21 | 1208 | **1207** | 124.7 | **1207** | 12.68 | **1207** | 5.84 |
| i06 | **1261** | **4.32** | **1261** | 13.85 | 1262 | **1261** | 78.34 | **1261** | 12.56 | **1261** | 7.67 |
| i07 | **1279** | **3.07** | **1279** | 14.6 | **1279** | **1279** | 114.2 | **1279** | 12.63 | **1279** | 7.5 |
| i08 | **1299** | **4.65** | **1299** | 14.21 | **1299** | **1299** | 57.06 | **1299** | 12.57 | **1299** | 9.94 |
| i09 | **1444** | **2.72** | **1444** | 16.51 | **1444** | **1444** | 96.47 | **1444** | 12.58 | **1444** | 4.25 |
| i10 | **1213** | **2.01** | **1213** | 14.16 | **1213** | **1213** | 99.41 | **1213** | 12.61 | **1213** | 5.2 |
| i11 | **1368** | **4.11** | **1368** | 14.13 | 1378 | 1369 | 99.34 | **1368** | 12.58 | **1368** | 10.52 |
| i12 | **1325** | **6.52** | **1325** | 15.6 | **1325** | **1325** | 80.69 | **1325** | 12.56 | **1325** | 12.92 |
| i13 | **1360** | **6.53** | **1360** | 13.87 | **1360** | **1360** | 89.94 | **1360** | 12.61 | **1360** | 11.97 |
| i14 | **1233** | **3.47** | **1233** | 15.6 | **1233** | **1233** | 73.95 | **1233** | 12.67 | **1233** | 7.11 |
| i15 | **1295** | **2.96** | **1295** | 13.52 | **1295** | **1295** | 74.19 | **1295** | 13.8 | **1295** | 8.3 |
| i16 | **1364** | **4.11** | **1364** | 13.68 | 1375 | 1365 | 170.36 | **1364** | 14.46 | **1364** | 8.48 |
| i17 | **1283** | **2.13** | **1283** | 13.37 | **1283** | **1283** | 46.58 | **1283** | 13.73 | **1283** | 5.66 |
| i18 | **1345** | **3.18** | **1345** | 13.51 | 1346 | **1345** | 84.02 | **1345** | 12.72 | **1345** | 8.02 |
| i19 | **1367** | **4.06** | **1367** | 14.59 | 1370 | **1367** | 123.19 | **1367** | 13.39 | **1367** | 11.42 |
| i20 | **1328** | **5.13** | **1328** | 16.64 | **1328** | **1328** | 82.3 | **1328** | 12.82 | **1328** | 12.28 |
| i21 | **1341** | **3.06** | **1341** | 13.37 | 1346 | **1341** | 108.08 | **1341** | 12.68 | **1341** | 7.11 |
| i22 | **1326** | **3.82** | **1326** | 15.24 | 1332 | **1326** | 105.38 | **1326** | 12.62 | **1326** | 7.94 |
| i23 | **1266** | **3.08** | **1266** | 13.65 | **1266** | **1266** | 43.72 | **1266** | 12.62 | **1266** | 7.25 |
| i24 | **1260** | **1.98** | **1260** | 15.58 | 1261 | **1260** | 78.91 | **1260** | 12.64 | **1260** | 5.67 |
| i25 | **1376** | **3.07** | **1376** | 15.8 | 1379 | **1376** | 96.58 | **1376** | 12.62 | **1376** | 7.13 |
| i26 | **1318** | **3.08** | **1318** | 15.38 | 1330 | **1318** | 101.11 | **1318** | 12.62 | **1318** | 7.44 |
| i27 | **1261** | **2.06** | **1261** | 15.52 | **1261** | **1261** | 82.86 | **1261** | 12.64 | **1261** | 6.16 |
| i28 | **1359** | **4.84** | **1359** | 16.22 | 1365 | 1360 | 52.91 | **1359** | 12.71 | **1359** | 11.52 |
| i29 | **1280** | **3.07** | **1280** | 15.3 | 1282 | **1280** | 203.36 | **1280** | 12.62 | **1280** | 8.11 |
| i30 | **1344** | **2.86** | **1344** | 16.52 | 1351 | **1344** | 71.02 | **1344** | 12.58 | **1344** | 7.13 |
| Avg | **1306.8** | **3.82** | **1306.8** | 14.98 | 1309.7 | 1306.9 | 93.99 | **1306.8** | 12.79 | **1306.8** | 8.17 |

## 4. Conclusion

In this work, we have proposed an Exponential Monte-Carlo with Counter (EMCQ) local search algorithm for the berth allocation problem. The proposed algorithm starts with an initial solution and iteratively improves it for a certain number of iterations. At each iteration, EMCQ uses a neighborhood operator to generate a neighborhood solution. Improving neighborhood solutions are always accepted, while worse solutions are adaptively accepted based on the quality of the incumbent solution, the search time and the number of consecutive non-improving iterations. To improve the effectiveness of the proposed EMCQ, we utilized three different neighbourhood operators to deal with a different instance characteristics. The proposed algorithm has been tested on berth allocation problem benchmark instances that have been used by other researchers in the literature. Results demonstrated that the proposed algorithm is very promising and can be used to produce good quality solutions compared to state of art methods.

## References

[1] C. Bierwirth and F. Meisel, "A survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, vol. 202, pp. 615-627, 2010.

[2] J.-F. Cordeau, G. Laporte, P. Legato, and L. Moccia, "Models and tabu search heuristics for the berth-allocation problem," *Transportation science*, vol. 39, pp. 526-538, 2005.

[3] K. Buhrkal, S. Zuglian, S. Ropke, J. Larsen, and R. Lusby, "Models for the discrete berth allocation problem: a computational comparison," *Transportation Research Part E: Logistics and Transportation Review*, vol. 47, pp. 461-473, 2011.

[4] R. M. de Oliveira, G. R. Mauri, and L. A. Nogueira Lorena, "Clustering Search for the Berth Allocation Problem," *Expert Systems with Applications*, vol. 39, pp. 5499-5505, 2012.

[5] C.-J. Ting, K.-C. Wu, and H. Chou, "Particle swarm optimization algorithm for the berth allocation problem," *Expert Systems with Applications*, vol. 41, pp. 1543-1550, 2014.

[6] M. Ayob and G. Kendall, "A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine," in *Proceedings of the international conference on intelligent technologies, InTech*, 2003, pp. 132-141.

[7] N. R. Sabar, M. Ayob, and G. Kendall, "Tabu exponential Monte-Carlo with counter heuristic for examination timetabling," in *Computational Intelligence in Scheduling, 2009. CI-Sched '09. IEEE Symposium on*, 2009, pp. 90-94.

[8] N. R. Sabar and M. Ayob, "Solving Examination Timetabling Problems using Exponential Monte-Carlo with Counter Heuristics," in *Proceeding of the 2008 first conference on Data Mining and Optimization (DMO'08). 3-4 Dec, Universiti Kebangsann Malaysia, pp.16-20.*, 2008.

[9] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A honey-bee mating optimization algorithm for educational timetabling problems," *European Journal of Operational Research*, vol. 216, pp. 533-543, 2/1/ 2012.

[10] N. R. Sabar, M. Ayob, and G. Kendall, "Solving examination timetabling problems using honey-bee mating optimization (ETP-HBMO)," *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009), Dublin, Ireland*, pp. 399-408, 2009.

[11] S. Abdullah, N. R. Sabar, M. Z. Ahmad Nazri, and M. Ayob, "An Exponential Monte-Carlo algorithm for feature selection problems," *Computers & Industrial Engineering*, vol. 67, pp. 160-167, 1// 2014.

[12] G. R. Mauri, A. C. Oliveira, and L. A. N. Lorena, "A hybrid column generation approach for the berth allocation problem," in *Evolutionary Computation in Combinatorial Optimization*, ed: Springer, 2008, pp. 110-122.