"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XVIII – 2015 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in:
PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

DIVIDE AND CONQUER APPROACH TO DISCRETE BERTH ALLOCATION PROBLEM

S. KORDIĆ[1]
N. KOVAČ[2]
T. DAVIDOVIĆ[3]
[1] University of Montenegro/Maritime Faculty, Kotor, Montenegro
[2] University of Montenegro/Maritime Faculty, Kotor, Montenegro
[3] Serbian Academy of Sciences and Arts/Mathematical Institute, Belgrade, Serbia

Abstract: *The paper presents an application of the divide and conquers strategy to exactly solve the Discrete Berth Allocation Problem (DBAP). After partitioning a problem by divide and conquer strategy to sub-problems Sedimentation Algorithm solver is used to exactly solve these sub-problems. Experimental evaluation we performed completely justifies application of the proposed divide and conquer strategy for exactly solving DBAP. Computational results on two classes of DBAP and discussion of the most difficult problems entirely demonstrate the superiority of the divide and conquer strategy over the approach to solve DBAP without it. Its application considerably reduces average and maximal runtimes for all test instances. Efficient C implementation enabled us to solve instances of DBAP with up to 120 vessels, which is more than enough for big ports.*
Keywords: *Combinatorial optimization, Scheduling problems, Cost minimization, exact algorithms, optimal solution.*

INTRODUCTION

The *Berth Allocation Problem* (BAP) consists of allocating berths to a set of vessels that need to be served within a given time horizon in a container port. Vessels are, among other information, represented by a set of data that includes the expected time of arrival, size, projected handling time, preferred berth in the port, and penalties. BAP can be defined as follows: for each vessel in the set, the berth index and the time interval are allocated in the manner that the given objective function is minimized. In [1], BAP was proven to be a NP-hard problem.

BAPs can be classified as discrete, continuous or hybrid. In discrete BAP (DBAP) quay is partitioned into a number of sections, called berths, and each berth can serve one vessel at a time. Time is also partitioned into discrete units, which allows the use of integer arithmetic for the calculation of the objective function value. Another possible classification distinguishes static and dynamic BAPs. In the static BAP, it is assumed that all vessels arrive at the container terminal in advance, namely before any berth becomes available. If the vessels can arrive at any time during the planning horizon (although we still have an a priori knowledge of their arrivals), then BAP is dynamic. A detailed BAP classification can be found in [2].

In recent literature, exact approaches addressing BAP are rare because they can solve problems with small number of vessels. Therefore, the majority of studies use heuristic or meta-heuristic methods to obtain suboptimal solutions of BAP. According to the recent survey of BAP by [3] exact methods are applied in 24% of approaches, while the rest of 76% approaches belongs to the heuristic and meta-heuristic methods.

An exact method for solving BAP can be found in [4]. The authors proposed an exact algorithm for solving the Tactical Berth Allocation Problem (TBAP) defined by [5]. The model for TBAP is based on an exponential number of variables, and it is solved via column generation. To obtain an integer solution, a branch-and-price scheme was applied along with several accelerating techniques specifically designed for solving TBAP.

Divide and conquer strategy is also rarely used for solving BAP. In this paper we present one general algorithm for solving BAP based on divide and conquer strategy. This algorithm needs a standalone procedure for solving BAP to be complete. We used BAP solver based on *Sedimentation Algorithm* with *Estimate and Rearrange Heuristic* (SA+ERH) develop in [6]. Numerical experiments are conducted only for DBAP. Other cases of BAP: hybrid and continuous are omitted because they would extend this paper beyond the predefined limit. Two sets of test examples involve 5 berths with one week planning horizon and 8 berths with two week planning horizon. We compare SA+ERH with or without *divide and conquer* strategy. Our computational results clearly prove the superiority of the proposed divide and conquer strategy. Model for BAP in this paper is based on Rashidi and Tsang model in [7].

The rest of this paper is organized as follows. First, we introduce our notation and explain the problem formulation in Section II. In Section III, divide and conquer strategy is described. Computational results are presented in Section IV. Finally, Section V contains concluding remarks and directions for future research.

PROBLEM FORMULATION

The process of assigning vessels to berths is very complex; it consists of several problems, such as the *Berth Planning Problem, Berth Allocation Problem, Quay Crane Assignment Problem,* and the *Quay Crane Scheduling Problem*.

In this work, we concentrate on minimum-cost BAP. The model for this problem is a sub-model of the model in [7]. We use only the part of that model that is referring to the berth allocation and reproduce it in the rest of this section. We omit the part of the model related to the crane assignment, assuming that each berth is equipped with exactly one crane.

A. Assumptions

We made the following assumptions about berthing of vessel in container port:

Assumption 1. Each vessel has a pre-determined berthing time period. A cost penalty applies if the vessel berths early, tardy or departs late.

Assumption 2. Each vessel has a preferred berthing location. Preferred location of the vessel depends on some preference like: location of the storage area where inbound/outbound containers of the vessel are stacked, depth of water, strength and direction of currents or some other preference.

Assumption 3. At each time period only one container can be loaded/unloaded on a berth. Moreover, we assume that at each berth there is exactly one crane available for loading/unloading of the vessel.

The **Assumption 3** is necessary to establish compatibility with the model for BAP in [7], since here we will deal only with the BAP without crane assignment.

B. Input Variables

Our model and algorithms use the input data listed below:

T: The total number of time periods in the planning horizon.

m: The number of berths in the port.

l: The number of vessels in the planning horizon.

vessel: The sequence of data relevant for vessels, which has the following structure:

$$vessel = \{(ETA_k, a_k, b_k, d_k, s_k, C_{1k}, C_{2k}, C_{3k}, C_{4k}) \mid k = 1,\ldots,l\}$$

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

ETA_k: The expected time of arrival of a $vessel_k$.

a_k: The processing time of the $vessel_k$.

b_k: The length of the $vessel_k$.

d_k: The required departure time for the $vessel_k$.

s_k: The least-cost berthing location of the $vessel_k$.

C_{1k}: The penalty cost for the $vessel_k$ if the vessel cannot dock at its preferred berth.

C_{2k}: The penalty cost for the $vessel_k$ per unit time for arrival before ETA_k.

C_{3k}: The penalty cost for the $vessel_k$ per unit time for arrival after ETA_k.

C_{4k}: The penalty cost for the $vessel_k$ per unit time of delayed departure after d_k.

Since, in DBAP only one berth is allocated to a vessel, then the value of the b_k parameters must be 1, for all vessels.

Set of all vessels indices we will denote by $L = \{1,...,l\}$.

C. Decision Variables and Domains

The formulation of BAP given in [7] uses following decision variables:

At_k: The arrival time of a $vessel_k$ to the corresponding berth, $At_k \in \{1,...,T\}$.

Dt_k: The departing time of the $vessel_k$ from the corresponding berth, $Dt_k \in \{1,...,T\}$.

Bi_k: The lowest berth index allocated to the $vessel_k$ from the corresponding berth, $Bi_k \in \{1,...,m\}$.

X_{itk}: If berth i at the time t is allocated to the $vessel_k$, X_{itk} takes the value 1; otherwise, its value is 0. Obviously, $X_{itk} \in \{0,1\}$.

D. Constrains

A feasible solution of BAP is subject to two sets of constraints.

Constraints1. At a time *t,* each berth can be assigned to only one vessel:

$$(\forall i \in \{1,...,m\})(\forall t \in \{1,...,T\}) \sum_{k=1}^{l} X_{itk} \le 1. \qquad (1)$$

Constraints2. A berth is allocated to the vessel only between its arrival and departure times:

$$(\forall k \in \{1,...,l\})(\forall i \in \{1,...,m\})(\forall t \in \{1,...,T\})$$
$$(At_k \le t \le Dt_k \wedge Bi_k \le i < Bi_k + b_k \Rightarrow X_{itk} = 1) \wedge \qquad (2)$$
$$(t < At_k \vee Dt_k < t \vee i < Bi_k \vee Bi_k + b_k \le i \Rightarrow X_{itk} = 0).$$

E. Objective Function

Let us first introduce the auxiliary variable Z_k, which represents the sum of the absolute distances between the preferred location of the $vessel_k$ and the berths allocated to the $vessel_k$:

$$Z_k = \sum_{t=1}^{T}\sum_{i=1}^{m}\begin{cases}|i - s_k| & : X_{itk} = 1\\ 0 & : X_{itk} = 0\end{cases}. \qquad (3)$$

The objective function for the minimization of the port penalty cost can be formulated as follows:

$$VesselCost = \sum_{k=1}^{l}\begin{pmatrix} C_{1k}Z_k + C_{2k}\left(ETA_k - At_k\right)^+ + \\ C_{3k}\left(At_k - ETA_k\right)^+ + \\ C_{4k}\left(Dt_k - d_k\right)^+ \end{pmatrix} \qquad (4)$$

The objective function minimizes the cost of vessels waiting time, speed up time, tardiness and berth position. According to [2] and the above problem formulation, our BAP can be classified as:

stat | fix | Σ(w₁ wait + w₂ speed + w₃ tard + w₄ pos).

The above model coves all three cases of BAP: discrete, continuous and hybrid.

DIVIDE AND CONQUER STRATEGY FOR SOLVING OF THE BERTH ALLOCATION PROBLEM

D&C strategy for solving BAP divides the set of input vessels into the subsets (sub-problems) and solves these sub-problems by a BAP solver. Solutions of the sub-problems are compared and then, if necessary, some of the subsets (sub-problems) are united and solved again. This procedure is repeated until there is no reason to unite subsets (sub-problems).

In this paper we selected BAP solver based on the *Sedimantation Algorithm* with *Estimate and Rearrange Heuristic* (SA+ERH) introduced in [6]. D&C strategy is independent of BAP solver, so any other BAP solver can be applied with it.

In this chapter we will introduce all the necessary prerequisites for the formulation of D&C algorithm and then algorithm for D&C algorithm will be exposed.

F. Input and Output of the D&C Algorithm

Input variables for D&C algorithm are the input variables of the BAP introduced in the previous section. Output of D&C algorithm will be the minimal (optimal) solution of the considered BAP, if it exists, and the value of the corresponding objective function. Solution of BAP is given in the form of the function:

$$s : L \to \{(t,b,dt,db) \mid t, dt \in \{1,...,T\} \wedge b, db \in \{1,...,m\}\}. \quad (5)$$

In the ordered 4-tuple (*t,b,dt,db*), *t* represents the arrival time of the vessel, *b* represents the lowest berth index allocated to the vessel, *dt* is the duration of vessel handling on the berths and *db* is the number of berths occupied by the vessel.

If function *s* is given and $s(k) = (t,b,dt,db)$, for $k \in L$, then decision variables for the $vessel_k$ can be calculated in the following way $At_k = t$ and $Bi_k = b$. Decision variables Dt_k and X_{itk} can be trivially calculated once when At_k and Bi_k are known. Therefore, it is sufficient only to determine the function *s*. Minimal value of the objective function we denote as *minimum*.

Definition1. Function of the form:

$$s : L \to \{(t,b,dt,db) \mid t, dt \in \{1,...,T\} \wedge b, db \in \{1,...,m\}\}, \quad (6)$$

is called *valuation*. Valuations that satisfy **Constrain 1.** and **Constrain 2.** are called *feasible solutions,* or shorter *solutions.* A 4-tuple (*t*,*b*,*dt*,*db*) is called *vessel position*.

Definition2. An algorithm *Alg* and its input variables *InVar* and output variables OutVar is denoted as:

OutVar := *Alg*(*InVar*).

If we label proposed divide and conquer strategy as D&C algorithm, then according to **Definition 2**. we write:

(*s*,*minimum*) := *D&C*(*T*,*m*,*l*,*vessel*). (7)

If the input problem does not have solution we assume that return variable *minimum* takes value $+\infty$.

G. BAP solver

BAP solver is sub procedure of the *D&C* algorithm. Therefore, the input variable for BAP solver is *ω*, a subset of the vessels indices: $\omega \subseteq L$. Similarly, as in the case of D&C algorithm, output is the optimal solution of the BAP for the subset *ω*, in the form of the function:

$$s_\omega : \omega \to \{(t,b,dt,db) \mid t, dt \in \{1,...,T\} \wedge b, db \in \{1,...,m\}\}. \quad (8)$$

Also, the value of the objective function for the solution s_ω, denoted by $minimum_\omega$, is output value as well. According to the **Definition 2**. if we label BAP solver as *BAPSolver* algorithm, we can sum the above as:

(s_ω, $minimum_\omega$) := BAPSolver(*ω*). (9)

If the input problem does not have a solution we assume that return variable *minimum* takes value $+\infty$.

As previously mentioned, *Sedimentation Algorithm* with *Estimate and Rearrange Heuristic* (SA+ERH) based solver was used as BAP solver. SA+ERH is halting algorithm and it always returns one optimal solution of the input problem. The proof of the total correctness of SA+ERH is given in [8]. In the COMPUTATION RESULT section comparison between SA+ERH and D&C is given for the case of discrete BAP (DBAP).

H. Conflicting relation

Berths and time units can be represented in a 2 dimensional discrete coordinate system. Horizontal axis represents time units, while vertical represents berths. Because of the **Constrain 1** and **Constrain 2,** vessels are rectangles that must fit in the Berth×Time plane.

Definition3. The relation ρ between the vessels with positions (t_{v1},b_{v1},dt_1,db_1) and (t_{v1},b_{v1},dt_1,db_1) is defined as follows:

$$(t_{v1}, b_{v1}, dt_1, db_1)\ \rho\ (t_{v2}, b_{v2}, dt_2, db_2) \Leftrightarrow$$
$$t_{v1} + dt_1 \le t_{v2} \vee t_{v2} + dt_2 \le t_{v1} \vee b_{v1} + db_1 \le b_{v2} \vee b_{v2} + db_2 \le b_{v1}. \quad (10)$$

If the two vessels are in relation ρ we call them *non-conflicting vessels*, otherwise we call them *conflicting vessels*. *Confliction* relation we will denote as $\overline{\rho}$.

Obviously, relations ρ and $\overline{\rho}$ are symmetrical, while only relation $\overline{\rho}$ is reflexive. In general case neither of them is transitive.

Proposition1. Valuation *s* is a feasible solution if and only if:

$$(\forall i, j \in L)\, i \ne j \Rightarrow s(i)\ \rho\ s(j). \quad (11)$$

The proof of the proposition is trivial so we omit it. In the rest of the paper we will use **Proposition 1.** very often. Stating that some valuation *s* is non-conflicting we will assume that condition (11) holds and that *s* represent a feasible solution of the input BAP. It is obvious from **Definition 4.** that vessels are non-conflicting if their rectangles in Berth×Time plane do not overlap. If their rectangles in Berth×Time plane do overlap, then they are conflicting.

Definition5. For a given valuation *s,* two disjoint subsets $\omega_1, \omega_2 \subset L$ are in relation ρ_s (called *non-conflicting*) if and only if $s(i)\,\rho\, s(j)$, for each $i \in \omega_1$ and $j \in \omega_2$. Definition of *non-conflicting* subsets can be written as:

$$\omega_1\ \rho_s\ \omega_2 \Leftrightarrow (\forall i \in \omega_1)(\forall j \in \omega_2)\, s(i)\,\rho\, s(j). \quad (12)$$

Definition6. For the given valuation *s,* two disjoint subsets $\omega_1, \omega_2 \subset L$ are in relation $\overline{\rho}_s$ (called *conflicting*) if and only if there are $i \in \omega_1$ and $j \in \omega_2$, such that $s(i)\,\overline{\rho}\, s(j)$. Definition of *conflicting* subsets can be written as:

$$\omega_1\ \overline{\rho}_s\ \omega_2 \Leftrightarrow (\exists i \in \omega_1)(\exists j \in \omega_2)\, s(i)\,\overline{\rho}\, s(j). \quad (13)$$

Definition7. For a binary relation $\sigma \subseteq L^2$, we denote its *reflexive transitive symmetric closure* by $\sigma^{\equiv}$. Relation $\sigma^{\equiv}$ is equivalence relation.

Definition8. For an equivalence relation $\sigma \subseteq L^2$, and element $j \in L$, we denote its *class of equivalence* by $[j]_\sigma = \{ i \in L \mid j\ \sigma\ i \}$.

Definition9. For an equivalence relation $\sigma \subseteq L^2$, we denote its *set of all equivalence classes* by $L_{/\sigma}$.

I. D&C Algorithm

Before describing in detail D&C algorithm let us first introduce following definition.

Definition10. Let s_0 denotes a valuation which assigns the least cost position to each vessel. In some cases function is not unique. In such cases we take any function with the feature of assigning the least cost position to a vessel.

We assume that function *CalculateObjFun*(*s*), which calculates value of the objective function for the valuation *s* is available.

As it is already mentioned, input variables for D&C are input variables of BAP. First, in the step **2**, we set initial value of the valuation *s* to be s_0. Valuation s_0 doesn't represent feasible solution, so we define relation ≈, as the reflexive transitive symmetric closure of the conflicting vessels in the valuation *s*, i.e. s_0. Obviously relation ≈ is equivalence relation.

In the step **3**, we define two sets: *U* and *S*. Set *U* contains unsolved sub-problems and set S consists of all solved sub-problems. Initial value of the set *U* is the set of all equivalent classes of the relation ≈. Elements of the sets *U* are disjoint subsets of the set of vessels indices *L* which represent sub-problems. Each two vessels which are elements of the same class of equivalence of the relation ≈ can be possibly conflicting. Vessels that are not in the same class of equivalence of the realtion ≈, are less likely to be conflicting.

Main idea of the D&C algorithm is to solve BAP separately for each sub-problem *ω* in the set *U*. If the solution of the sub-problem *ω* is not conflicting with the other sub-problems in sets *U* and *S,* then the member *ω* is moved to the set *S*. If the solution is conflicting, then all conflicting members of the sets *U* and *S* with *ω* are united, removed from the sets *U* and *S* and finally BAP for the union is solved. The described procedure is repeated until set *U* is nonempty. Therefore initial value of the set *S* is the empty set, while the final value for set *U* is the empty set.

TABLE 1
D&C ALGORITHM

1: **function** *D&C*(*T*,*m*,*l*,*vessel*)
2: $s = s_0$; $\approx = \overline{\rho_s}^{=}$
3: $U = L_{/\approx}$; $S = \{\ \}$
4: **until** $U \neq \{\ \}$ **do**
5: $\omega \in U$
6: (s_ω, *minimum*$_\omega$) := BAPSolver(*ω*)
7: **if** *minimum*$_\omega = +\infty$ **then return** $(\{\ \}, +\infty)$
8: $s = s_{|L\setminus\omega} \cup s_\omega$; $\approx = \overline{\rho_s}^{=}$
9: $\Omega = \{\varphi \in U \cup S \mid \omega \approx \varphi\ \}$
10: **if** $\Omega = \{\omega\}$ **then**
11: $S = S \cup \{\omega\}$; $U = U \setminus \{\omega\}$
12: **else**
13: $S = S \setminus \Omega$;
14: $U = U \setminus \Omega$; $U = U \cup \{\bigcup \Omega\}$
15: **endif**
16: **enduntil**
17: **return** $(s, CalculateObjFun(s))$
18: **endfunction**

In the step **4**, we enter the loop until set *U* is non empty. In loop we first select any element *ω* of the set *U* in the step **5**. In the step **6** we solve BAP for the sub-problem *ω* and save the valuation and the value of the optimal solution into varibles s_ω and *minimum*$_\omega$. If *minimum*$_\omega = +\infty$ then sub-problem *ω* does not have feasible solution. This means that the initial set of vessels *L* also does not have a solution, so in the step **7** we return empty set indicating that the initial problem does not have a feasible solution.

If the sub-problem *ω* does have a solution, then, in the step **8, we** update valuation *s* with the solutions of the sub-problem s_ω by keeping the same positions for the vessels not in *ω*, and replacing the old positions for the vessels in *ω.* Also in the step **8** we update relation ≈ so that it is defined as a reflexive transitive symmetric closure of the conflicting vessels in the updated valuation *s.*

In the step **9** we define set Ω as the set of all conflicting subsets from sets U and S with sub-problem *ω* in the valuation *s*. Set Ω is nonempty set, at least it contains only sub-problem *ω*. If that is the case, then solution of the sub-problem *ω* is not in conflict with the solutions of the solved sub-problems in *S* as well as the least costing positions of vessels of the unsolved sub-problems in *U*. So, it is safe to add *ω* to the set of solved sub-problems *S* and to remove it from the set of the unsolved sub-problems *U,* in the steps **10** and **11**.

If Ω contains more sub-problems than *ω*, then sub-problems are removed from both *U* and *S*. We unite all vessels in the set of sub-problem Ω which are in conflict with *ω* into a new sub-problem $\cup\Omega$. New sub-problem $\cup\Omega$ has to be solved, and therefore, it is added to the set *U*. Description in this paragraph covers the steps **10**, **13** and **14**.

When the set *U* is empty all the sub-problems are solved and they are mutually non-conflicting. A that point, function *s* contains valuation which is the optimal solution of the initial problem. Therefore function ends with the return of the valuation *s* and the value of the optimal solution calculated by the predefined function *CalculateObjFun*(*s*) in the step **17**.

Proposition1. *D&C* algorithm is halting.

Proof: Halting problem for D&C algorithm, described in the TABLE 1, is equivalent to the question does the until loop in steps **4.** to **16.** stops. The condition for the until loop is non emptiness of the set *U*. Notice also that cardinality of the set *S* is at most equal to the number of the loop executions.

If the $\Omega = \{\omega\}$ then then cardinality of the set U is reduced, so in that case infinite loop will not occur.

On the other hand, if $|\Omega| > 1$, then beside *ω* set Ω contains at least one more member $\varphi \in \Omega$. Member *φ* can be either from set *U* or set *S*, according to the rules for calculating set Ω (step **9**).

If $\varphi \in U$ then in the step **14**. cardinality of the set *U* is decreased at least by 2 and then increased by 1. All in all the cardinality of the set *U* is decreased at least by 1, so in that case infinite loop will not occur.

Finally, if $\varphi \in S$ cardinality of the set *S* is decreased in the step **13**, then, in the worst case, the cardinality of the set *U* remains the same in the step **14**. It appears that this case can introduce possibility of the infinite loop.

But this case can be iterated only while the set *S* is non empty. Since *S* is always finite set, this case can occur only as much times as it is cardinality of the set *S*, i.e. a finite number of times. After that, some of the cases in which cardinality of the set *U* is reduced will occur, thus preventing the infinite loop. **Q.E.D.**

Proposition2. Solution obtained by D&C algorithm is a feasible solution.

Proof: After halting of the D&C algorithm finite set *S* contains non conflicting sets of sub-problems exactly solved by the BAPSolver function. Therefore, valuation *s* is *non-conflicting* and by **Proposition 1.** it is also a feasible solution of the input BAP. **Q.E.D.**

Proposition3. Solution obtained by D&C algorithm is an exact (optimal) solution.

Proof: Let us denote by m_s the value of the objective function for the valuation *s*. Also, let us denote the members of the set $S = \{\omega_1, \omega_2, ..., \omega_k\}$. From the description of D&C algorithm it is clear that set *S* is a partition of the set *L*. Let $m_s(i)$ represents values of the objective function for the $\omega_i, i \in \{1,...k\}$ in the valuation *s*.

Let valuation *t* be any feasible solution of the input BAP. First let us denote by m_t the value of the objective function for the valuation *t* and by $m_t(i)$ values of the objective function for the $\omega_i, i \in \{1,...k\}$ in the valuation *t.*

Since $m_s(i)$ are the values of the objective function for optimal solutions of sub-problems $\omega_i, i \in \{1,...k\}$ following holds:

$$(\forall i \in \{1,...,k\})\, m_s(i) \le m_t(i). \qquad (14)$$

From (14) we easily derive that:

$$m_s = \sum_{i=1}^{k} m_s(i) \le \sum_{i=1}^{k} m_t(i) = m_t. \qquad (15)$$

So, for any feasible solution *t* its objective function value is greater than or equal to the objective function value of the feasible solution *s*, i.e. $m_s \le m_t$. Therefore, valuation *s* is one of the optimal solutions of the considered BAP. **Q.E.D.**

Note, that optimal solution of BAP does not have to be unique. Previous propositions proof the *total correctness* of the D&C algorithm for solving BAP.

COMPUTATIONAL RESULTS

In this section, we present the test instances and computational results of D&C algorithm and SA+ERH. Finally, we analyze few long running time examples in order to find answer what makes them hard for solving with D&C algorithm.

J. Test Instances

The experimental evaluation is performed on two classes of DBAP instances that are similar to those introduced in [5]. We consider test instances with 5 and 8 berths and time horizons of 1 or 2 weeks. The time horizon is divided into 3-hour time units. Thus, one week has 56 time units and two weeks are divided into 112 time units. The number of vessels ranges from 5 to 120, with an increment of 5 vessels, and it is specific for each test class we consider.

The classes of test instances are the following:

Class I: 5 berths, 1 week, and 5 to 45 vessels.

Class II: 8 berths, 2 weeks, and 5 to 120 vessels.

The information required to specify various types of vessels are presented in Table 2. The specifications resemble those used by [2]; however, here they are adjusted to DBAP. Three types of vessels are present in the test population: feeder, medium and mega. For each type, the corresponding percentage of the test population, handling time range, penalty amounts (in units of US$ 1000) are listed in TABLE 2. Let us note again that in DBAP each vessel occupies only one berth.

TABLE 2.
TEST VESSELS SPECIFICATIONS

Size, handling times & penalties for test vessels						
Vessel type	*Percentage of the test population*	*Handling time range*	C_1	C_2	C_3	C_4
Feeder	60%	1 – 3	2	3	3	9
Medium	30%	4 – 5	3	6	6	18
Mega	10%	6 – 8	4	9	9	27

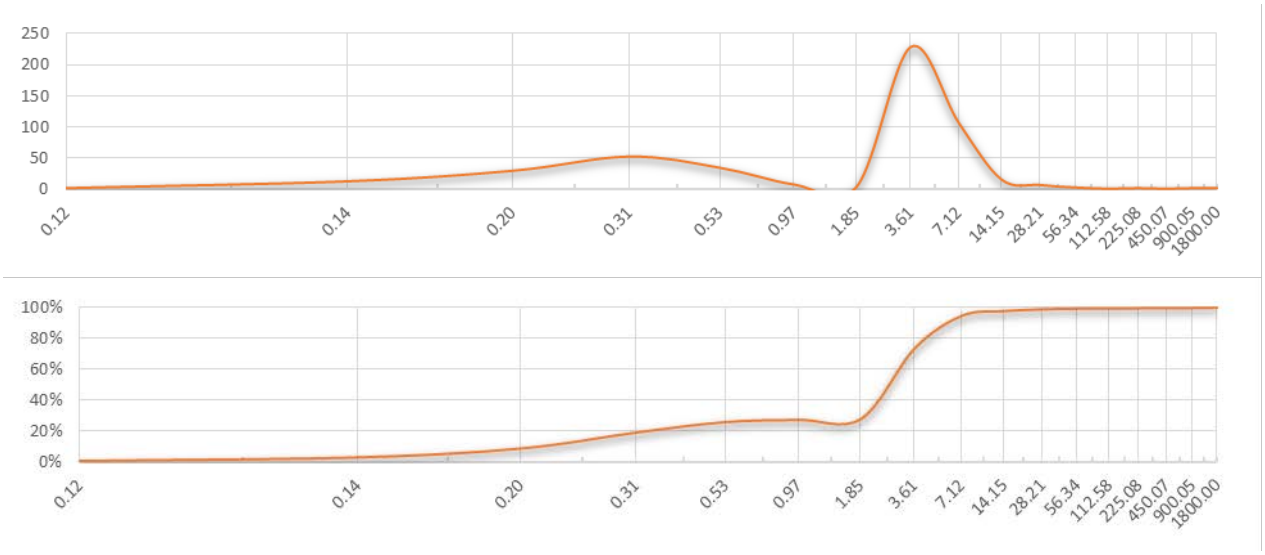

Figure 1. Distribution of runtimes for D&C and its cumulative graph of Class I for 40 vessels

The distribution of the least-cost berthing location for vessels is homogeneous. For each instance and the number of vessels, 500 tests were randomly generated. We recorded the percentage of the tests that were solved in a half-an-hour period, i.e., 1800 seconds. For all the tests that were solved in a half-an-hour period, we also recorded the minimum, average and maximum time required to find the solution.

Tests that were not solved in a half-an-hour period were interrupted and they are not included into the calculation of the minimal, average and maximal time for finding solution. All the times in the following tables are expressed in seconds.

K. Camparison Between D&C & SA+ERH

D&C algorithm and SA+ERH were coded in the *C* programming language. Code was compiled by *Microsoft C/C++ Optimizing Compiler version 18.00.31101 for x86*. The tests were conducted on a computer with an *Intel Core i7-4500U* @ 1.80GHz—2.40GHz CPU and 8 GB of RAM running the *Microsoft Windows 8.1* 64-bit operating system.

TABLE 3.
RUNTIMES FOR THE CLASS I EXAMPLES

	Class I: 5×56 | 500 samples | DBAP							
	SA+ERH				*D&C Algorithm*			
l	***t* ≤ ½*h* %**	**min**	**avg**	**max**	***t* ≤ ½*h* %**	**min**	**avg**	**max**
5	*100.0*	0.00	**0.02**	0.06	*100.0*	0.00	**0.02**	0.05
10	*100.0*	0.02	**0.02**	0.05	*100.0*	0.01	**0.04**	0.08
15	*100.0*	0.02	**0.03**	0.16	*100.0*	0.02	**0.06**	0.19
20	*100.0*	0.02	**0.06**	0.22	*100.0*	0.02	**0.08**	0.14
25	*100.0*	0.02	**0.15**	7.41	*100.0*	0.03	**0.13**	1.46
30	*100.0*	0.03	**1.33**	80.44	*100.0*	0.08	**0.39**	11.26
35	*96.6*	0.05	**17.31**	1399.49	*100.0*	0.09	**1.68**	210.37
40	–	–	–	–	*100.0*	0.11	**6.94**	1325.05
45	–	–	–	–	*99.2*	0.16	**16.60**	1077.15

TABLE3. shows the computational results for the Class I test instances of DBAP. In the cases with 5 to 25 vessels, both algorithms expectedly perform very well. In the case of 25 vessels average runtimes of D&C and SA+ERH are still almost equal, while maximal runtime of D&C algorithm is much smaller than the runtime for the SA+ERH, 1.46 seconds compared with 7.41. In the case of 30 vessels both algorithms solve all 500 examples in a half-an-hour period. In that case D&C algorithm average runtime is 3.41 times faster than SA+ERH, while maximal runtime is 7.14 times faster. In the case of 35 vessels SA+ERH cannot solve all 500 examples in a half-a-hour time period, whereas D&C algorithm solve all 500 examples not only for the cases of 35, but also for the cases of 40 vessels. It is evident that for the cases of 35 vessels and more maximal runtimes of D&C algorithm are significantly reduced compared to SA+ERH. The results of the SA+ERH execution for the cases with the number of vessel larger than or equal to 40 are not presented in TABLE 2 because the time needed for solving 500 examples was too long. Note that the average runtime of D&C algorithm for 45 vessels is still lower than the average runtime of SA+ERH for 25 vessels. Also, the percent of problems solved in a half-an-hour period by *D&C* algorithm for 45 vessels (99.2%) is larger compared with SA+ERH for 35 vessels (96.6%).

From the TABLE 3 we conclude that D&C algorithm is capable to solve all examples with 40 vessels in a half-an-hour period, while SA+ERH can solve all examples in a half-an-hour period with at most 30 vesels. Also it significantly reduces maximal runtime for any number of vessels.

Distribution of D&C algorithm runtimes for the case of 40 vessels and its cumulative function are given in the Figure 1.

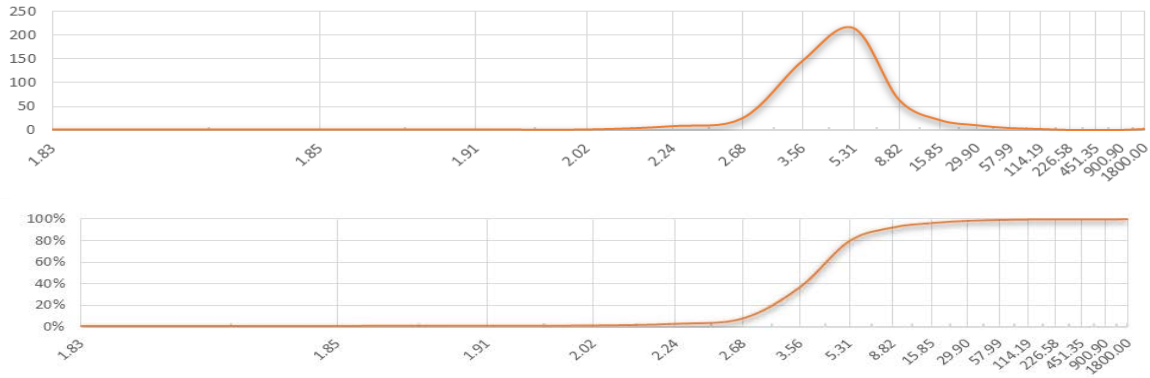

Figure 2. Distribution of runtimes for D&C and its cumulative graph of Class II for 100 vessels

On the horizontal axis time interval limits in logarithmic scale are given. On the vertical axis for distribution graph number of solved examples is represented, while for the cumulative graph percentage of the solved problems is represented. For the cumulative graph we can read that 99% of the problems were solved in less than 28.21 seconds and only 1% needed more than 28.21 seconds to be solved.

Number of unsolved examples for the case of 45 vessels is relatively small, 3 out of 500, which is 0.4%. These are examples numbered by: 100, 268 and 444, which we call *hard examples* for the case with 45 vessels. Given the unlimited time for solving *hard examples* by D&C algorithm, the corresponding runtimes are presented in TABLE 4.

TABLE 4.
CLASS I HARD CASES FOR 45 VESSELS

No	**Class I: 5×56 | 500 samples | DBAP**	
	D&C h:m:s	*Maximal length sub problem*
100	0:20:49	*31*
268	4:48:20	*33*
444	More than 12:00:00	*33*

Examples 268 and 444 are hard to solve by *D&C* algorithm in the half-an-hour time-period because of the large sub problems with 33 and more vessels. On the other hand D&C algorithm's failure to solve example 100 in a half-an-hour period is due to the bad estimations of the sub problems optimal solutions, which plays an important role in the efficiency of the BAPSolver(ω) procedure. In the second run example 100 was solved within the time limit of half-an-hour thanks to the much batter estimations during the work of the BAPSolver(ω) procedure.

TABLE 5. shows the computational results for the Class II test instances of DBAP. Superiority of the D&C algorithm over SA+ERH is evident in every aspect. Average runtimes and maximal runtimes are significantly reduced. In the case of 60 vessels average runtime of D&C algorithm is 23.25 times faster compared to SA+ERH. Also, D&C algorithm is capable to solve all examples with 100 vessels in a half-an-hour period, while SA+ERH can solve all examples in a half-an-hour period with at most 60 vessels. The percentage of examples not solved in a half-an-hour period for 105 to 115 vessels is relatively small from 0.4% or 0.8% (2 or 4 out of 500 examples). For 120 vessels it is 2.6%, which is still good result considering the number of vessels.

TABLE 5.
RUNTIMES FOR THE CLASS I EXAMPLES

l	**Class I: 8×112 | 500 samples | DBAP**							
	SA+ERH				*D&C Algorithm*			
	***t* ≤ ½*h* %**	**min**	**avg**	**max**	***t* ≤ ½*h* %**	**min**	**avg**	**max**
5	*100.0*	0.00	**0.02**	0.03	100.0	0.00	**0.02**	0.05
10	*100.0*	0.01	**0.02**	0.05	100.0	0.02	**0.02**	0.06
15	*100.0*	0.02	**0.03**	0.09	100.0	0.00	**0.01**	0.08
20	*100.0*	0.02	**0.04**	0.49	100.0	0.00	**0.01**	0.03
25	*100.0*	0.03	**0.07**	0.77	100.0	0.00	**0.01**	0.02
30	*100.0*	0.03	**0.13**	0.84	100.0	0.00	**0.02**	0.11
35	*100.0*	0.03	**0.30**	1.47	100.0	0.00	**0.02**	0.16
40	*100.0*	0.05	**0.61**	1.88	100.0	0.01	**0.03**	0.38
45	*100.0*	0.05	**0.90**	2.27	100.0	0.02	**0.04**	0.36
50	*100.0*	0.05	**1.52**	22.10	100.0	0.02	**0.08**	0.66
55	*100.0*	0.07	**4.45**	385.47	100.0	0.02	**0.16**	0.95
60	*100.0*	0.24	**6.51**	780.00	100.0	0.03	**0.28**	1.23
65	*99.4*	0.41	**41.82**	1753.07	100.0	0.03	**0.49**	1.38
70	–	–	–	–	100.0	0.03	**0.70**	3.88
75	–	–	–	–	100.0	0.05	**1.01**	22.74
80	–	–	–	–	100.0	0.05	**1.25**	12.09
85	–	–	–	–	100.0	0.06	**1.86**	26.05
90	–	–	–	–	100.0	0.58	**2.43**	19.27
95	–	–	–	–	100.0	1.23	**3.56**	38.16
100	–	–	–	–	100.0	1.81	**9.08**	984.72
105	–	–	–	–	99.6	1.91	**10.04**	1153.83
110	–	–	–	–	99.2	2.13	**11.91**	1199.67
115	–	–	–	–	99.2	2.41	**20.05**	1126.23
120	–	–	–	–	97.4	2.61	**25.30**	1566.79

TABLE 5. shows the computational results for the Class II test instances of DBAP. Superiority of the D&C algorithm over SA+ERH is evident in every aspect. Average runtimes and maximal runtimes are significantly reduced. In the case of 60 vessels average runtime of D&C algorithm is 23.25 times faster compared to SA+ERH. Also, D&C algorithm is capable to solve all examples with 100 vessels in a half-an-hour period, while SA+ERH can solve all examples in a half-an-hour period with at most 60 vessels. The percentage of examples not solved in a half-an-hour period for 105 to 115 vessels is relatively small from 0.4% or 0.8% (2 or 4 out of 500 examples). For 120 vessels it is 2.6%, which is still good result considering the number of vessels.

Distribution of D&C algorithm runtimes for the case of 100 vessels and its cumulative function are given in the Figure 2. For the cumulative graph we can read that 98.4% of the problems were solved in less than 29.90 seconds and only 1.6% needed more than 29.90 seconds to be solved.

Number of unsolved examples for the case of 105 vessels is relatively low, 2 out of 500, which is, as mentioned, 0.2%. These are examples numbered by: 299 and 377, which we call *hard examples* for 105 vessels. Given the unlimited time for solving *hard examples* by D&C algorithm, the corresponding runtimes are given in TABLE 6.

TABLE 6.
CLASS II HARD CASES FOR 105 VESSELS

No	Class II: 8×112 | 500 samples | DBAP	
	D&C h:m:s	*Maximal length sub problem*
299	4:22:32	*38*
377	0:11:32	*27*

Example 299 is hard to solve by D&C algorithm in the half-an-hour time-period because of the large sub problem with 38 vessels. Example 377, similarly as example 100 in Class I, was not solved in a half-an-hour period due to the bad estimations of the sub problems optimal solutions by BAPSolver(*ω*) procedure. In the second run example 377 was solved within the time limit of half-an-hour thanks to the much batter estimations during the work of the BAPSolver(*ω*) procedure.

CONCLUSION

We considered divide and conquer strategy for solving (minimum-cost) Berth Allocation Problem (BAP) with the static arrival of vessels and fixed vessel handling times. Proposed strategy implemented as *D&C* algorithm is not stand alone method for solving BAP. It is additional technique which can significantly reduce solving time of any standalone algorithm for solving BAP. *Total correctness* of the D&C algorithm for solving BAP is proved. The computational experiments performed only for discrete BAP (DBAP) fully justify the design and further development of D&C algorithm for solving BAP. For more than 98% of our test instances optimal solution was found in less than 30 seconds. Average runtime for finding optimal solution is 6.97 seconds for the cases with up to 40 vessels, 5 berths and 56 time units and 9.65 seconds for the cases with up to 100 vessels, 8 berths and 112 time units. The most difficult problems were solved within the time limit of 1800 seconds. These results indicate that this method can be used for solving real-life (big sized) instances of BAP, depending on container port layout.

The differences between the minimum and maximum solving times for a large number of vessels in the test instances indicate that D&C algorithm is worth further development. We find it especially worthwhile to investigate the implementation of parallel variant of the *D&C* algorithm.

Moreover, the generalization of the algorithm to the hybrid and continuous BAP and the inclusion of crane assignment are natural avenues for further development of the algorithms presented here.

BIBLIOGRAPHY:

[1] A. Lim, "The Berth planning problem," *Operational Research Letters,* vol. 22, no. 2-3, pp. 105-110, 1998.

[2] F. Meisel, Seaside Operations Planning in Container Terminals, Berlin Heidelberg: Physica-Verlag, 2009.

[3] C. Bierwirth and F. Meisel, "A Follow-up Survey of Berth Allocation and Quay Crane Scheduling Problems in Container Terminals," *European Journal of Operational Research,* vol. 244, no. 3, 2015.

[4] I. Vacca, M. Salani and M. Bierlaire, "An Exact Algorithm for the Integrated Planning of Berth Allocation and Quay Crane Assignment," Ecole Polytechnique Fédéralede Lausanne, Lausanne, 2011.

[5] G. Giallombardo, L. Moccia and M. Salani, "Modeling and Solving the Tactical Berth Allocation Problem," *Transportation Research Part B Methodological,* vol. 44, no. 2, pp. 232-245, 02 2009.

[6] S. Kordić, B. Dragović, T. Davidović and N. Kovač, "A Combinatorial Algorithm for Berth Allocation Problem in Container Port," in *Proceedings of International Association of Maritime Economists Conference*, Taipei, 2012.

[7] H. Rashidi and E. P. Tsang, "Novel Constrains Satisfaction Models for Optimization Problems in Container Terminals," *Applied Mathematical Modelling,* vol. 37, no. 6, pp. 3601-3634, 2013.

[8] S. Kordić, B. Dragović, T. Davidović and N. Kovač, "Combinatorial Approach to Exactly Solvig Discrete and Hybrid Berth Allocation Problem," *unpublished.*