A genetic algorithm for berth allocation and quay crane assignment

Mario Rodriguez-Molins[1], Federico Barber[1], María R. Sierra[2], Jorge Puente[2], and Miguel A. Salido[1]

[1] Instituto de Automática e Informática Indsutrial, Universidad Politécnica de Valencia (Spain)
{mrodriguez,msalido,fbarber}@dsic.upv.es
[2] Department of Computer Science, University of Oviedo (Spain)
{puente,sierramaria}@uniovi.es

Abstract. Container terminals are facilities where cargo containers are transshipped between different transport vehicles, for onward transportation. They are open systems that carry out a large number of different combinatorial problems that can be solved by means of Artificial Intelligence techniques. In this work, we focus our attention on scheduling a number of incoming vessels by assigning to each a berthing position, a mooring time and a number of Quay Cranes. This problem is known as the Berthing Allocation and Quay Crane Assignment problem. To formulate the problem, we first propose a mixed integer linear programming model to minimize the total weighted service time of the incoming vessels. Then, a meta-heuristic algorithm (Genetic Algorithm (GA)) is presented for solving the proposed problem. Computational experiments are performed to evaluate the effectiveness and efficiency of the proposed method.

Keywords: Scheduling, Planning, Genetic Algorithms, Metaheuristics, Berthing Allocation, Quay Crane Assignment

1 Introduction

A container terminal is an open system with three distinguishible areas (berth, container yard and lanside areas) where there exist different complex optimization problems. For instance, berthing allocation or stowage planning problems are related to the berth area [?]; remarshalling problem or transport optimization in the yard area; and, planning and scheduling hinterland operations related to trains and trucks in the landside area [?].

Two planning and scheduling problems are studied in this paper, the Berth Allocation Problem (BAP) and the Quay Crane Assignment Problem (QCAP). The former is a well-known combinatorial optimization problem [?], which consists in assigning incoming vessels to berthing positions. The QCAP deals with assigning a certain number of QCs to each vessel that is waiting at the roadstead such that all required movements of containers can be fulfilled [?]. Once a vessel arrives at the port, it waits at the roadstead until it has permission to moor at the quay. The locations where mooring can take place are called berths. These are equipped with giant cranes, known as Quay Cranes (QC), that are used to load and unload containers which are transferred to and from the yard by a fleet of vehicles. These QCs are mounted on the same track (or rail) and, therefore they cannot pass each other. In a transshipment terminal, the yard allows temporary storage before containers are transferred to another ship or to another transportation mode (e.g., rail or road).

A comprehensive survey of BAP and QCAP is given by [?]. These problems have been mostly considered separately and with an interest mainly focused on BAP. However, there are some studies on the combined BAP+QCAP considering different characteristics of the berths and cranes ([?], [?], [?], [?], [?]). In this paper, we present a formal mixed integer lineal programming for the combined BAP+QCAP that extends the model presented in [?], by managing a continuous quay line. In order to obtain optimized solutions in an efficient way, we develop a metaheuristic GA, so that compared with mathematical solvers obtains near-optimal solutions in competitive computational times.

The rest of the paper is organized as follows. In the next two sections we give a thorough description and a mathematical formulation of the problem. In Section 4 we give the details of the GA designed for the BAP+QCAP. Section 5 reports the results of the experimental study. Finally, in Section 6 we give the main conclusions of this work.

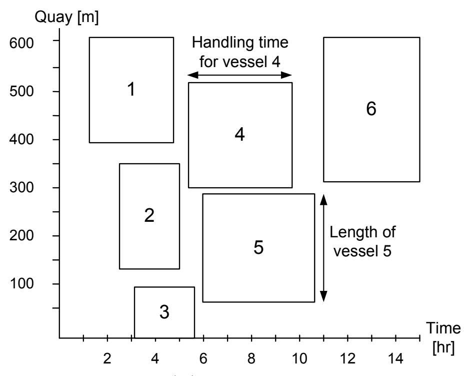
2 Problem description

The objective in BAP+QCAP is to obtain an schedule of the incoming vessels with an optimum order of vessels mooring and a distribution of the docks and QCs for these vessels. Figure 1(b) shows an example of the graphical space-time representation of a berth plan with 6 vessels. Each rectangle represents a vessel with its handling time and length.

Our BAP+QCAP case is classified according to the classification given by [?] as:

- *Spatial attribute: Continuous layout.* We assume that the quay is a continuous line, so there is no partitioning of the quay and the vessel can berth at arbitrary positions within the boundaries of the quay. It must be taken into account that for a continuous layout, berth planning is more complicated than for a discrete layout, but it better utilizes the quay space [?].
- *Temporal attribute: Dynamic arrival.* Fixed arrival times are given for the vessels, so that vessels cannot berth before their expected arrival times.
- *Handling time attribute: Unknown in advance.* The handling time of a vessel depends on the number of assigned QCs (*QCAP*) and the moves required.
- *Performance measure: wait and handling times* The objective is to minimize the sum of the waiting (w_i) and handling times (h_i) of all vessels.

Let V be the set of incoming vessels. Following, we introduce the notation used for each vessel $i \in V$ (Figure 1(a)). The data variables are:



(a) Vessel (b) Schedule

Fig. 1. Representation of the BAP+QCAP problem

- QC : Available QCs in the container terminal. All QCs carry out the same number of movements per time unit ($movsQC$), given by the container terminal.
- L : Total length of the berth in the container terminal.
- a_i : Arrival time of the vessel i at port.
- c_i : Number of required movements to load and unload containers of i.
- l_i : Vessel length.
- pr_i : Vessel priority.

The decision variables are:

- m_i : Mooring time of i. Thus, waiting time (w_i) of i is calculated as ($w_i = m_i - a_i$).
- p_i : Berthing position where i moors.
- q_i : Number of assigned QCs to i.
- u_{ik} : Indicates whether the QC k works (1) or not (0) on the vessel i.

The variables derived from the previous ones are:

- h_i : Loading and unloading time at quay (handling time) of vessel i. This time depends on q_i and c_i, that is : $\left(\frac{c_i}{q_i \times \texttt{movsQC}}\right)$.
- t_{ik} : Working time of the QC k that is assigned to vessel i.
- d_i : Departure time of vessel i ($d_i = m_i + h_i$).
- s_i, e_i : indexes for the first and last QC used in vessel i, respectively.

Our objective is to allocate all vessels according to several constraints minimizing the total weighted waiting and service time for all vessels:

$$T_s := \sum_{i \in V} (w_i + h_i) \times pr_i \tag{1}$$

Note that this problem is a very special case of a multi-mode resource-constrained scheduling problem, where there exist shared resources (berth length), the duration of activities (mooring time) depends on the assigned resources (QCs), and the objective function is minimizing both the waiting as the processing times of vessels.

Moreover, the following assumptions are considered:

- Number of QCs assigned to a vessel do not vary along the moored time. Once a QC starts a task in a vessel, it must complete it without any pause or shift (non-preemptive tasks). Thus, all QCs assigned to the same vessel have the same working time ($t_{ik} = h_i, \forall k \in QC, u_{ik} = 1$)
- All the information related to the waiting vessels is known in advance (arrival, priority, moves and length).
- Every vessel has a draft that is lower than or equal to the draft of the quay.
- Movements of QCs along the quay as well as berthing and departure times of vessels are not considered since it supose a constant penalty time for all vessels.
- The components of the optimization function (Equation 1) can be independently weighted without requiring changes to our proposal.
- Simultaneous berthing is allowed, subject to the length of the berth.

And the following constraints must be accomplished:

- Moored time must be at least the same that its arrival time ($m_i \geq a_i$).
- It must be enough contiguous space at berth to moor a vessel of length (l_i).
- There is a safety distance (`safeDist`) between two moored ships. We assume 5% of the maximum length of two contiguous vessels.
- There must be at least one QC to assign to each vessel. The maximum number of assigned QCs by vessel depends on its length, since a safety distance is required between two contiguous QCs (`safeQC`), and the maximum number of QCs that the container terminal allows per vessel (`maxQC`). Both parameters are given by the container terminal.

3 Mathematical formulation

In this section, the mathematical formulation for BAP+QCAP is presented. The given MILP model solves the BAP+QCAP by minimizing the function given by the Equation 1, where M denotes a sufficiently large number, subject to the given constraints:

$$m_i \geq a_i \quad \forall i \in V \tag{2}$$
$$w_i = m_i - a_i \quad \forall i \in V \tag{3}$$
$$p_i + l_i \leq L \quad \forall i \in V \tag{4}$$
$$q_i = \sum_{k \in QC} u_{ik} \quad \forall i \in V \tag{5}$$
$$1 \leq q_i \leq QC_i^+ \quad \forall i \in V \tag{6}$$
$$1 \leq s_i, e_i \leq |QC| \quad \forall i \in V \tag{7}$$
$$s_i \geq e_i \quad \forall i \in V \tag{8}$$

$$q_i = e_i - s_i + 1 \quad \forall i \in V \tag{9}$$
$$\sum_{k \in QC} t_{ik} \times \texttt{movsQC} \geq c_i \quad \forall i \in V \tag{10}$$
$$h_i = \max_{k \in QC} t_{ik} \quad \forall i \in V \tag{11}$$
$$t_{ik} - u_{ik} \times M \leq 0 \quad \forall i \in V, \forall k \in QC \tag{12}$$
$$h_i - M \times (1 - u_{ik}) - t_{ik} \leq 0 \quad \forall i \in V, \forall k \in QC \tag{13}$$
$$u_{ik} + u_{jk} + z_{ij}^x \leq 2 \quad \forall i, j \in V, \forall k \in QC \tag{14}$$
$$M \times (1 - u_{ik}) + (e_i - k) \geq 0 \quad \forall i \in V, \forall k \in QC \tag{15}$$
$$M \times (1 - u_{ik}) + (k - s_i) \geq 0 \quad \forall i \in V, \forall k \in QC \tag{16}$$
$$p_i + l_i \leq p_j - sd_{ij} + M \times (1 - z_{ij}^x) \quad \forall i, j \in V,\ i \neq j \tag{17}$$
$$e_i + 1 \leq s_j + M \times (1 - z_{ij}^x) \quad \forall i, j \in V,\ i \neq j \tag{18}$$
$$m_i + h_i \leq m_j + M \times (1 - z_{ij}^y) \quad \forall i, j \in V,\ i \neq j \tag{19}$$
$$z_{ij}^x + z_{ji}^x + z_{ij}^y + z_{ji}^y \geq 1 \quad \forall i, j \in V,\ i \neq j \tag{20}$$
$$z_{ij}^x, z_{ij}^y, u_{ik} \quad 0/1 \text{ integer} \quad \forall i, j \in V,\ i \neq j, \forall k \in QC \tag{21}$$

The given formulation expands the model presented in [**?**] by adding the needed constraints to take into consideration QCs. Thereby, the handling time of vessels depends on the number of QCs and these QCs cannot pass each other when are relocated.

In the proposed model, there are two auxiliary variables: z_{ij}^x is a decision variable that indicates if vessel i is located to the left of vessel j on the berth ($z_{ij}^x = 1$); and, $z_{ij}^y = 1$ indicates that vessel i is moored before vessel j in time (see constraint 21). Moreover, Constraint 2 ensures that vessels must moor once they arrive at the terminal. Constraint 4 guarantees that a moored vessel does not exceed the length quay. Constraints 5, 6, 7, 8 and 9 assign the number of QCs to the vessel i. Constraint 10 establishes the needed handling time to load and unload their containers. Constraint 12 ensures that QCs that are not assigned QCs to i have t_{ik} zero. Constraint 13 forces all assigned QCs to i working the same number of hours. Constraint 11 assigns the handling time for vessel i.

Constraint 14 avoids that one QC is assigned to two different vessels at the same time. Constraints 15 and 16 force the QCs to be assigned contiguously (from s_i up to e_i). Constraint 17 takes into account the safety distance between each two vessels. Constraint 18 avoids that one vessel uses a QC which should cross through the others QCs. Constraint 19 avoids that vessel j moors while the previous vessel i is still at the quay. Finally, constraint 20 establishes the relationship between each pair of vessels.

This mathematical model has been coded in IBM ILOG CPLEX Optimization Studio 12.3 as detailed in the Evaluation Section 5.

4 Genetic Algorithm

Algorithm 1 shows the structure of the GA we have considered herein. The core of this algorithm is taken from [**?**,**?**] and is quite similar to others generational genetic algorithms described in the literature ([**?**], [**?**] or [**?**]). In the first step, the initial population is generated and evaluated. Then, the genetic algorithm iterates over a number of steps or generations. In each iteration, a new generation is built from the previous one by applying the genetic operators of selection, reproduction and replacement. These operators can be implemented in a variety of ways and, in principle, are independent from each other. However, in practice all of them should be chosen considering their effect on the remaining ones in order to get a successful overall algorithm. The approach taken in this work is the following. In the selection phase all chromosomes are grouped into pairs, and then each one of these pairs is mated or not in accordance with a crossover probability (P_c) to obtain two offspring. Each offspring, or parent if the parents were not mated, undergoes mutation in accordance with the mutation probability (P_m). Finally, the replacement is carried out as a tournament selection (4:2) among each pair of parents and their offspring.

Algorithm 1 The genetic algorithm

```
Require: A BAP-QCAP instance $P$
Ensure: A mooring schedule for instance $P$
  1. Generate the initial population;
  2. Evaluate the population;
  while No termination criterion is satisfied do
    3. Select chromosomes from the current population;
    4. Apply the reproduction operators to the chromosomes selected at step 3. to
    generate new ones;
    5. Evaluate the chromosomes generated at step 4;
    6. Apply the replacement criterion to the set of chromosomes selected at step 3.
    together with the chromosomes generated at step 4.;
  end while
  return The schedule from the best chromosome evaluated so far;
```

The coding schema is based on permutations of vessels, each one with a given number of QCs. So a gene is a pair (i, q_i), $1 \leq q_i \leq \min(maxQC_i, maxQC)$, and a chromosome includes a gene like this for each one of the vessels. For example, for an instance with 5 vessels where the maximum number of QCs are 2, 3, 4, 3 and 2 respectively, two feasible chromosomes are the following ones:

$$c_1\text{: ((1 1) (2 1) (3 1) (4 2) (5 1))}$$

$$c_2\text{: ((3 2) (1 2) (2 2) (5 2) (4 3))}$$

Note that, the same vessel may have different number of QCs in each chromosome. In accordance with this encoding, a chromosome expresses the number of QCs that each vessel is assigned in the solution and an order for building the schedule.

The order of vessels in chromosomes is used as a dispatching rule. Hence, we use the following decoding algorithm: the genes are visited from left to right in the chromosome sequence. For each gene (i, q_i) the vessel i is scheduled at the earliest mooring time with q_i consecutive QCs available, so that none of the constraints is violated. If there are several positions available at the earliest time, that closest to one of the berth extremes is selected. Also, the QCs are chosen starting from the same extreme of the berth.

For chromosome mating we have considered a classical crossover operator such as Generalized Position Crossover (GPX) which is commonly used in permutation based encodings. This is a two points crossover operator which work as follows. Let us consider two parents like:

$$p_1\text{: ((1 1) | (2 1) (3 1) | (4 2) (5 1))}$$

$$p_2\text{: ((3 2) | (1 2) (2 2) | (5 2) (4 3))}$$

Symbols "|" represent crossover positions, 1 and 3 respectively in this example, which are selected at random for each mating. Then two offsprings are built taking the substrings between positions 1 and 3 in each parent and then filling the remaining positions with the genes representing the remaining vessels taken from the other parent keeping their relative order. So in this case the two offsprings are:

$$o_1\text{: ((1 2) | (2 1) (3 1) | (5 2) (4 3))}$$

$$o_2\text{: ((3 1) | (1 2) (2 2) | (4 2) (5 1))}$$

For mutation we have implemented an operator that shuffles a random substring of the chromosome and at the same time changes the number of QCs assigned to each one of the shuffled genes at random, provided that the number of QCs is kept in between the proper limits for the vessel.

The initial population in generated at random, i.e. a random order for the vessels is chosen and each vessel i is assigned a number of QCs chosen uniformly in $[1, \min(maxQC_i, maxQC)]$. The termination condition is given in one of these three forms: (1) a number of generations, (2) a time limit or (3) a number of evaluations.

5 Evaluation

The experiments were performed in a corpus of 100 instances generated randomly, each one is composed of a queue from 5 to 20 vessels. These instances follow an exponential distribution for the inter-arrival times of the vessels ($\lambda = \frac{1}{20}$). The number of required movements and length of vessels are generated uniformly in $[100, 1000]$ and $[100, 500]$ respectively. In all cases, the berth length (L) is fixed to 700 meters; the number of QCs is 7 (corresponding to a determined MSC berth line) and the maximum number of QCs per vessel is 5 (`maxQC`); the safety distance between QCs (`safeQC`) is 35 meters and the number of movements that QCs carry out is 2.5 (`movsQC`) per time unit.

The two approaches developed in this paper, the GA and the MILP model, were coded using C++ and the IBM ILOG CPLEX Optimization Studio 12.3, respectively. They were solved on a Linux PC 2.26Ghz.

In the GA, the population size is 200. Mutation and crossover probabilities are $P_m = 0.1$ and $P_c = 0.8$, respectively. Due to the stochastic nature of the GA process, each one of the instances were solved 30 times and the results show the average obtained values.

Table 1 shows the results form CPLEX and GA averaged for each group of 100 instances with the same number of vessels (5 to 20). The timeout was 10 seconds. For CPLEX, the reported values are the average value of T_s for the solutions reached, the number of instances solved to optimality (#Opt), the number of instances solved without certify optimality (#NOpt) and the number of instances for which no solution is reached by the timeout (#NSol) The last two columns show the best and the average values of the solutions obtained by the GA in 30 runs. Obviously, in all cases, the objective function (T_s) increases as the number of incoming vessels increases from 5 up to 20.

Table 1. Comparision CPLEX with GA (timeout 10 secs)

|V|	**CPLEX**				**GA**	
	Avg T_s	**#Opt**	**#NOpt**	**#NSol**	**Best T_s**	**Avg T_s**
5	1723.75	98	2	0	1723.75	1723.75
6	2193.06	88	12	0	2189.63	2189.63
7	2702.46	66	34	0	2681.14	2681.67
8	3287.66	41	59	0	3219.80	3222.13
9	3891.09	24	76	0	3729.78	3734.72
10	4642.23	14	86	0	4337.10	4350.23
11	5453.31	6	94	0	4946.66	4971.86
12	6557.60	3	97	0	5552.09	5589.16
13	7944.50	2	98	0	6181.67	6236.60
14	9332.26	1	98	1	6854.33	6931.59
15	11578.40	0	98	2	7526.27	7631.98
16	13518.00	0	97	3	8290.95	8438.06
17	15105.80	0	94	6	8972.13	9163.65
18	17253.80	0	85	15	9694.16	9927.06
19	18390.40	0	65	35	10506.20	10787.79
20	20410.50	0	46	54	11395.52	11725.64

From these results, we can observe that CPLEX is not able to reach any optimal solution by the given timeout in at least 30% of the instances with 7 vessels or more. In addition, it can not get any optimal solution from 15 up to 20 vessels with this timeout. Moreover, for a number of instances with more than 14 vessels CPLEX is not able to reach a feasible solution. Regarding GA, all instances are solved and we can observe that the average values are better than those from CPLEX, the differences being in direct ratio with the number of vessels. Here, it is important to remark that GA reaches 1063 generations in 10 seconds. However, the GA is able to converge in lower times. Figure 2 shows the GA convergence for one representative instance of 20 vessels, so that near-optimal values are obtained after 100 generations, taking 0.94 seconds. Furthermore, Figure 3 shows how the average T_s for 10 vessels decreases as more computation time is allowed. In this experiment, the timeout was set to 5, 10, 20, and 60 seconds. As it can be observed, the GA approach does not require a large timeout (the improvement is lower than 1% beyond 5 seconds).

We remark that we have not been able to use previous test cases proposed in the literature because we assume a continuous berthing and non-preemtitive tasks ([**?**], [**?**]). However, even considering this more complex case, we can see that the results achieved are highly competitive against these previous approaches.

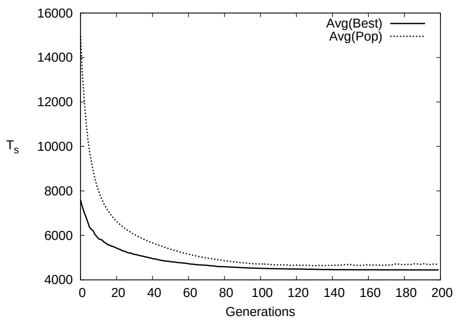


Fig. 2. Convergence of the Genetic Algorithm

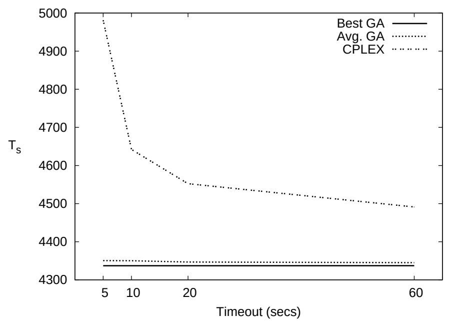


Fig. 3. Average T_s for 10 vessels setting different timeouts

6 Conclusions

The competitiveness among container terminals causes the need to improve the efficiency of each one of the subprocesses that are performed within them. This paper focuses on two of the main related problems, the Berth Allocation and Quay Crane Assignment Problems, in an integrated way. To this end, a mixed integer lineal programming model and a Genetic Algorithm were developed. The MILP model was unable to get optimal solutions when a reasonable timeout is set or when the problem becomes harder (more than 10 vessels). Moreover, many of the instances were solved but without any guarantees of being the optimal ones since the timeout was reached. However, the GA approach is able to obtain near-optimal solutions in lower computational times and it also maintains a rapid convergence of the results even with large vessel queues. From these results, it is concluded the adequacy of a metaheuristic approach based on GA for solving the BAP+QCAP problem. This approach also extends the previous approaches given in the literature by adding features (as continuous quay line and non-preemptitive QC assignments) and it gives near-optimal solutions in a very competitive computational time. For future research, we propose devising some local search strategy that can be then combined with the GA or other metaheuristics such as GRASP, Tabu or Scatter Search.

Acknowledgments. This research has been supported by the Spanish Government under research project MICINN TIN2010-20976-C02-01 and TIN2010-20976-C02-02, and the predoctoral FPU fellowship (AP2010-4405).