SEQUENTIAL ORDERING PROBLEMS FOR CRANE SCHEDULING IN PORT TERMINALS

R. Montemanni[(a)], A.E. Rizzoli[(a)], D.H. Smith[(b)], L.M. Gambardella[(a)]

[(a)]Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland
[(b)]Division of Mathematics and Statistics, University of Glamorgan, Pontypridd, U.K.

[(a)]{roberto,andrea,luca}@idsia.ch, [(b)]dhsmith@glam.ac.uk

ABSTRACT

The sequential ordering problem is a version of the asymmetric travelling salesman problem where precedence constraints on vertices are imposed. A tour is feasible if these constraints are respected, and the objective is to find a feasible solution with minimum cost.

The sequential ordering problem models many real world applications, mainly in the fields of transportation and production planning. In particular, it can be used to optimise quay crane assignments.

In this paper we experimentally evaluate the contributions of the basic ingredients of the state-of-the-art algorithm for the sequential ordering problems: local searches, ant colony and heuristic manipulation.

Keywords: quay crane optimization, sequential ordering, heuristic algorithm.

1. THE SEQUENTIAL ORDERING PROBLEM

1.1. Introduction

Maritime transport has been constantly increasing in the past years. The Review of Maritime Transport published by UNCTAD[1] in 2007 reports an average increase of 13.52% in 2005 and 14.63% in 2006 for the top 20 ports. Such double figures are correlated to the strong development of trade with emerging countries such as China and India, and the trend is expected to continue even in the face of uncertainties in the world economic outlook. Port infrastructures are also growing, but the investments cannot keep up with the pace of the expansion of trade. Problems related to port congestion and lack of storage space are here to stay. The optimization of port terminal resources is therefore a key tool for the success of port operations. Optimization can also play a major role in minimizing the environmental impact of sea freight, since a better usage of the existing resources can also have a significant impact on the quantity of energy required to efficiently operate a large infrastructure such as a maritime port terminal.

A maritime port terminal is a complex system, composed of a number of parts, each one with its specific purpose and characteristics. Steenken et al. (2004) describe it as an open system of material flow with two interfaces, one on the quay-side, to the sea, and one on the hinterland, to inland. The management of a terminal port is obviously a complex decisional problem, due to the multiple interactions which are present among the various component processes of the port. Efforts have been made towards the systematic classification and solution of the various planning, management and control problems which are encountered in terminal systems. Of particular interest is the classification by Günther and Kim (2006) who divide the problems according to the decisional level (terminal design, operational planning, and operational control) and to the subject of the decisional problem. Thus, at the design level we encounter problems such as the design of the multi-modal interfaces, of the terminal layout, the selection of equipment, while at the planning level the focus is on the synthesis of storage and stacking policies, assigning cranes to ships, planning stowage and berth assignment. Finally, at the operational control level, the terminal manager has to solve problems such as land-side and quay-side transport and crane operations and scheduling.

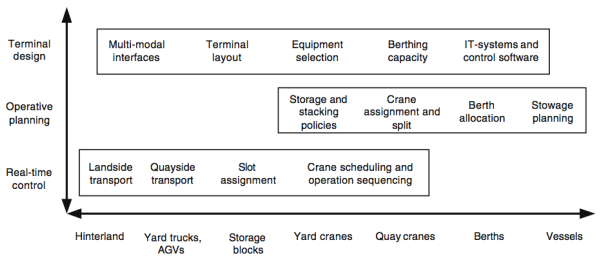

Figure 1. Design, planning and control problems with respect to the components of a maritime terminal (from Günther and Kim 2006).

Given the complexity of the structure and of the processes, may authors have attempted a "divide and conquer" approach. In this approach the global port optimization problem is decomposed into a series of simpler problems, which are solved independently, using bound-

[1] http://www.unctad.org.

ary conditions to guarantee the overall feasibility of the solution, as also shown by Gambardella et al. (2001), and also by a number of other authors, as reported in the recently updated comprehensive review by Stahlbock and Voß(2008).

In this paper we focus our attention on the problem of the optimisation of quay crane scheduling at the operational control level once the crane assignment and crane split problem have been solved at the planning level. In particular, we assume that a quay crane is dedicated to loading and unloading a specific ship section, as happens in the vast majority of cases, since two cranes serving the same ship section would need interleaving, without any clear advantage in performance. This assumption allows us to formulate the crane scheduling problem as a sequential ordering problem, since we have one machine (the crane) which must serve a number of jobs (the containers) which are subject to precedence constraints, which are given by the stacking positions of the containers on the terminal yard and by the loading/unloading lists of the ship, which are set according to the final destination of the containers.

1.2. Problem description

The *Sequential Ordering Problem* (*SOP*), also referred to as the *Asymmetric Travelling Salesman Problem with Precedence Constraints*, can be modelled in graph theoretical terms as follows. A complete directed graph $D = (V, A)$ is given, where V is the set of nodes and $A = \{(i, j)|i, j \in V\}$ is the set of arcs. A cost $c_{ij} \in \mathcal{N}$ is associated with each arc $(i, j) \in A$. Without loss of generality it can be assumed that a fixed starting node $1 \in V$ is given. It has to precede all the other nodes. The tour is also closed at node 1, after all the other nodes have been visited ($c_{i1} = 0\ \forall i \in V$ by definition). This artifact creates an analogy with the asymmetric travelling salesman problem. Such an analogy is exploited by many known algorithms. Furthermore an additional precedence digraph $P = (V, R)$ is given, defined on the same node set V as D. An arc $(i, j) \in R$, represents a precedence relationship, i.e. i has to precede j in every feasible tour. Such a relation will be denoted as $i \prec j$ in the remainder of the paper. The precedence digraph P must be acyclic in order for a feasible solution to exist. It is also assumed to be transitively closed, since $i \prec k$ can be inferred from $i \prec j$ and $j \prec k$. Note that for the last arc traversed by a tour (entering node 1), precedence constraints do not apply. A tour that satisfies precedence relationships is called *feasible*. The objective of the *SOP* is to find a feasible tour with the minimal total cost.

It is interesting to observe that *SOP* reduces to the classical asymmetric travelling salesman problem (*ATSP*) in the case where no precedence constraint is given. This observation implies that *SOP* is $\mathcal{NP}$-hard, being a generalization of the *ATSP*.

1.3. Literature review

The *SOP* can model real-world problems such as production planning (Escudero 1988 and Seo and Moon 2003), single vehicle routing problems with pick-up and delivery constraints (Pulleyblank and Timlin 1991, Savelsbergh 1990) and transportation problems in flexible manufacturing systems (Ascheuer 1995).

Sequential ordering problems were initially solved as constrained versions of the *ATSP*, especially for the development of exact algorithms. The main effort has been put into extending the mathematical definition of the *ATSP* by introducing new classes of valid inequalities to model the additional constraints. The first mathematical model for the *SOP* was introduced by Ascheuer et al. (1993), where a cutting plane approach was proposed to compute lower bounds on the optimal solution. Escudero et al (1994), a Lagrangean relaxation method was described and embedded into a branch and cut algorithm. Ascheuer (1995) has proposed a new class of valid inequalities and has described a new branch-and-cut method for a broad class of *SOP* instances. This is based on the polyhedral investigation carried out on *ATSP* problems with precedence constraints by Balas et al. (1995). The approach by Ascheuer (1995) also investigates the possibility of computing and improving sub-optimal feasible solutions starting from the upper bound provided by the polyhedral investigation. The upper bound is the initial solution of a heuristic phase based on well-known *ATSP* heuristics that are iteratively applied in order to improve feasible solutions. These heuristics do not handle constraints directly; infeasible solutions are simply rejected. A branch and bound algorithm with lower bounds obtained from homomorphic abstractions of the original search space has been presented by Hernàdvölgyi (2003, 2004). A genetic algorithm has been proposed in Chen and Smith (1996). The method works in the space of feasible solutions by introducing a sophisticated crossover operator that preserves the common schemata of two parents by identifying their maximum partial order through matrix operations. The new solution is completed using constructive heuristics. A hybrid genetic algorithm based on complete graph representation has been discussed by Seo and Moon (2003). A parallelised roll-out algorithm has been described by Guerriero and Mancini (2003). Gambardella and Dorigo (2000) presented an approach based on Ant Colony System (ACS) enriched with sophisticated Local Search (LS) procedures. Montemanni et al. (2007, 2008) built on top of this method, adding a Heuristic Manipulation Technique (HMT) on top of the original algorithm. This resulting method can be classified as state-of-the-art for the sequential ordering problem.

The contribution of the present article is an experimental study aiming at understanding the contribution of the basic ingredients (LS, ACO, HMT) in the economy of the composite method described by Montemanni et al. (2008).

The paper is organized as follows: Sections 2, 3 and 4 describe local searches, ant colony optimization and heuristic manipulation for the SOP, respectively. These are the basic ingredients of the composite algorithm de-

scribed by Montemanni et al. (2008). Extensive computational experiments, aiming at understanding how the algorithm performs when only a subset of the ingredients is put in operation, are presented in Section 5. Conclusions are drawn in Section 6.

2. LOCAL SEARCH (LS)

The extremely efficient *SOP-3-exchange* local search routine (we will refer to it as LS in the reminder of the paper) has been introduced by Gambardella and Dorigo (2000). This local search routine is a specialization to the sequential ordering problem of a known local search method for the asymmetric travelling salesman problem (Savelsbergh 1990). It is able to directly handle multiple constraints without increasing the computational complexity of the original local search.

SOP-3-exchange starts from a feasible solution, and generates a new solution by replacing three edges (Figure 2) $(h, h+1)$, $(i, i+1)$, $(j, j+1)$ with another set of improving three edges $(h, i+1)$, $(j, h+1)$, $(i, j+1)$. This operation is iteratively executed until no additional improving 3-exchange is possible. *SOP-3-exchange* explores the set of feasible edges using a lexicographic approach. The lexicographic approach identifies step by step the set of three edges and therefore two paths, *path_left* and *path_right*, which once inverted give rise to a new feasible solution. In the lexicographic search these two paths are initially composed of one single node and are incrementally expanded adding one node at each step. This feature makes it possible to test feasibility easily because precedence conditions must be checked only for the new added node.

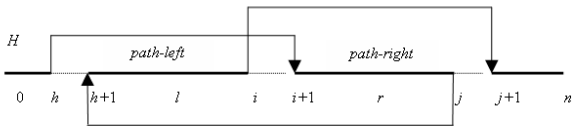

Figure 2. SOP-3-exchange local search.

In fact *SOP-3-exchange* uses a new a labeling procedure whose function is to check feasibility in constant time. The solution is to keep updated a set of global variables that indicates, each time a new node is added to *path_left* or *path_right*, if this insertion violates one of the precedence constraints.

Other important *SOP-3-exchange* features are related to the way node h is selected. The goal is to decrease the number of visited nodes introducing two heuristics that influence how node h is chosen: one is based on the *don't look bit* data structure introduced by Bentley (1992), while the other is based on a data structure called *don't push stack* introduced by Gambardella and Dorigo (2000). The *don't look bit* is a data structure in which a bit is associated with each node of the sequence. At the beginning of the search all bits are turned off. The bit associated with node h is turned on when a search for an improving move starting from h fails. The bit associated with node h is turned off again when an improving exchange involving h is executed. The use of *don't look bits* favors the exploration of nodes that have been involved in a profitable exchange.

The *don't push stack* is a data structure based on a stack, which contains the set of nodes h to be selected. At the beginning of the search the stack is initialized with all the nodes in the sequence (that is, it contains $n+1$ elements). During the search, node h is popped off the stack and feasible 3-exchanges starting from h are investigated. In case a profitable exchange is executed the six nodes involved in this exchange are pushed onto the stack (if they do not already belong to it). Using this heuristic, once a profitable exchange is executed starting from node h, the top node in the *don't push stack* remains node h. In this way the search is focused on the neighborhood of the most recent exchange: this has been experimentally shown to result in a better performance than that obtained using the *don't look bit* alone.

3. ANT COLONY SYSTEM (ACS)

In this section the basic concepts of the ACS algorithm, originally presented by Gambardella and Dorigo (2000), are discussed. The Ant Colony System algorithm is an element of the *Ant Colony Optimization* family of methods (Dorigo et al. 1999). These algorithms are based on a computational paradigm inspired by real ant colonies and the way they function.

ACS for the SOP is an adaptation to the problem of the Ant Colony System algorithm (Dorigo and Gambardella 1997). Informally, ACS works as follows. Constructive computational agents called ants (simulating real ants) are sent out sequentially. Each ant iteratively starts from node 1 and adds new nodes until all nodes have been visited. When in node i, an ant applies a so-called transition rule, that is, it probabilistically chooses the next node j from the set $F(i)$ of feasible nodes. $F(i)$ contains all the nodes j still to be visited and such that all nodes that have to precede j, according to precedence constraints, have already been inserted in the sequence.

The ant in node i chooses the next node j to visit on the basis of two factors: the heuristic *desirability* η_{ij} here defined as $1/c_{ij}$, and the *pheromone trail* τ_{ij}, that contains a measure of how good it has been in the past to include arc (i, j) into a solution. The next node to visit is chosen with probability q_0 as the node j, $j \in F(i)$, for which the product $\tau_{ij} \cdot \eta_{ij}$ is highest (deterministic rule), while with probability $1 - q_0$ the node j is chosen with a probability given by $p_{ij}{}_{j \in F(i)} = \frac{\tau_{ij} \cdot \eta_{ij}}{\sum_{l \in F(i)} (\tau_{il} \cdot \eta_{il})}$.

The value q_0 is given by $q_0 = 1 - s/|V|$. The parameter s represents the number of nodes it would be desirable to choose using the probabilistic transition rule, independently of the number of nodes of the problem, and is set to 10 in the experiments reported here.

In *ACS-SOP* only the best ant, that is the ant that built the shortest tour since the beginning of the computation, is allowed to deposit pheromone trail. If the shortest path generated since the beginning of the computation is referred to as $OptPath_{Best}$, and its cost as L_{Best}, $\forall\{i,j\} \in OptPath_{Best}$, the following formula for pheromone update is used:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \frac{\rho}{L_{best}} \quad (1)$$

Notice that in case a local search is used inside ACS, the updating rule (1) is run based on the solution obtained after the application of the local search.

Pheromone is also updated during solution building. In this case, however, it is removed from visited arcs. In other words, each ant, when moving from node i to node j, applies a pheromone updating rule that causes the amount of pheromone trail on arc (i,j) to decrease. This assures variety in the solutions generated. The rule is:

$$\tau_{ij} = (1-\psi)\cdot\tau_{ij} + \psi\cdot\tau_0 \quad (2)$$

where τ_0 is the initial value of trails. It was found that good values for the algorithm's parameters are $\tau_0 =$ (*FirstSolution* $\cdot |V|$)$^{-1}$ and $\rho = \psi = 0.1$, where *FirstSolution* is the length of the shortest solution generated by the ant colony following the *ACS-SOP* algorithm without using the pheromone trails. The number of ants in the population was set to 10. Experience has shown the chosen parameter settings to be robust.

4. HEURISTIC MANIPULATION TECHNIQUE (HMT)

It is easy to observe that adding precedence constraints to a given problem reduces its search space, making the problem potentially easier to solve. The method described in the remainder of this section is based on this observation.

Having selected an underlying heuristic method, the idea is to monitor the solutions generated by this method, and to identify precedence patterns common to solutions with a low objective value. Once such precedence patterns are identified, they can be added to the original problem as *artificial precedence constraints*. The manipulated problem is likely to be easier than the original one, as it has a reduced solution space.

Notice that any heuristic method that produces a sequence of feasible solutions to the problem (most of the known methods work in this way) can be used as the underlying method for the proposed manipulation approach.

Of course such a heuristic method may cut out all the optimal solutions of the original problem, leading to suboptimal solutions even when the best solution of the modified problem is retrieved. To overcome this side effect, during the execution of the algorithm artificial precedence constraints will not be added permanently, but will also be retracted (and substituted by other constraints).

Formally, the proposed methodology is built on top of an existing algorithm and makes use of an additional set of variables $\{m_{ij}\}$. Variable m_{ij} will be an indicator for the "quality" of the solutions in which node i is visited before node j. The method is regulated by some parameters. Parameter u regulates the number of solutions generated by the underlying heuristic method before the first artificial precedence constraints are added to the problem; parameter v regulates the number of solutions generated by the underlying heuristic method between two consecutive updates to the set of active artificial precedence constraints; parameter w regulates the (approximate) number of artificial precedence constraints active at each moment in time (after the first u solutions have been generated by the underlying heuristic approach); parameter z finally regulates the (approximate) number of artificial precedence constraints substituted after every v new solutions have been generated by the underlying heuristic algorithm (after the first u solutions have been generated).

The heuristic manipulation technique runs on top of underlying optimization algorithm, and can be summarized as follows.

Initialize $m_{ij} = 0 \ \forall(i,j) \in A$.

Each time a new solution $OptPath_k$, with cost L_k, is generated by the underlying heuristic algorithm, matrix $m = [m_{ij}]$ is updated as described in (3) and (4), where L_1 is the cost of the very first solution generated by the underlying heuristic algorithm and $\pi_k(i)$ is the index of the position occupied by node i in solution $OptPath_k$. Notice that L_1 plays here the role of a normalization factor, and is used to avoid numerical problems. The width of the window considered for updates is taken to be 5. According to Montemanni et al. (2008), some tests suggests that the method is not very sensitive to changes to this value, and that good values are however in the range $[4, 10]$.

The first update (equation (3)) reinforces the entry corresponding to a sequence which is in solution $OptPath_k$. The update is proportional to the inverse of the cost of the solution itself. Equation (4) decreases the value on arcs that are traversed in the opposite direction in the current solution. This second update has been inserted to make those pairs of nodes that do not seem to have a clear ordering relationship less attractive. Notice that only pairs with a positive entry in matrix m will be potentially transformed into artificial precedence constraints.

Notice that entries of the memory matrix m corresponding to active artificial precedence constraints are not updated. This will make a rotation of the active constraints more likely (see the remainder of this section).

Now that it has been clarified how the memory matrix m is handled, it remains to clarify how artificial precedence constraints are managed. After the first u solutions are created by the underlying heuristic algorithm, a first

$$m_{ij} = m_{ij} + \frac{L_1}{L_k} \ \forall i, j \in V, \pi_k(i) < \pi_k(j) \leq \pi_k(i) + 5, (i, j) \notin R \quad (3)$$

$$m_{ji} = m_{ji} - \frac{L_1}{L_k} \ \forall i, j \in V, \pi_k(i) < \pi_k(j) \leq \pi_k(i) + 5, (i, j) \notin R \quad (4)$$

set of (approximately) w artificial precedence constraints are added to the set R. The new constraints are selected as the ones not yet present in the precedence digraph P with the highest entries in matrix m. If there are less than w entries of m with a positive value, then only the precedence constraints corresponding to them will be added to P.

After the first artificial precedence constraints have been added to the problem, every time v new solutions are available, artificial precedence constraints are updated by dropping z constraints, that are substituted by (approximately) z new constraints. The artificial precedence constraints to be dropped are selected as those with the smallest entries in the memory matrix m. Conversely, the ones added are those with the highest entries in the same matrix m. Notice that, since entries of matrix m corresponding to active constraints are not reinforced (see equation (3)), it is likely that during the time they were active, entries corresponding to other (non active) constraints have reached higher values. Such a strategy leads to a mechanism where artificial precedence constraints are activated in turn. The mechanism should also prevent optimal solutions of the original problem from being permanently hidden by the active artificial precedence constraints.

5. EXPERIMENTAL RESULTS

The aim of the experiments reported in this section is to understand the importance of the basic components (described in Sections 2, 3 and 4) in the economy of the composite method described by Montemanni et al. (2008). Therefore, we run experiments with different combinations of active components.

5.1. Benchmark problems

Benchmark problems are those already used by Montemanni et al. (2007, 2008). They are publicly available[2], and are named *n-r-p*, where the meaning of each element is as follows:

- **n**: the number of nodes of the problem, i.e. $V = \{1, 2, \ldots, n\}$;
- **r**: the cost range, i.e. $0 \leq c_{ij} \leq r \ \forall i, j \in V$;
- **p**: the approximate percentage of precedence constraints, i.e. the number of precedence constraints of the problem will be about $\frac{p}{100} \cdot \frac{n(n-1)}{2}$.

The following values for the parameters above were used, and problems were generated for all possible combinations of them:

- $n \in \{200, 300, 400, 500, 600, 700\}$;
- $r \in \{100, 1000\}$;
- $p \in \{1, 15, 30, 60\}$.

The resulting set of problems covers a wide range of situations, with different sizes, different granularity for costs, and with radically different percentages of precedence constraints. The set appears to provide a good testbed for modern *SOP* heuristic algorithms, and are therefore particularly indicated for our purpose.

5.2. Comparison of the different combinations

All the methods considered have been coded in C++. The experiments have been run on a Dual AMD Opteron 250 2.4GHz / 4GB computer. The maximum computation time was set to 600 seconds for all the problems. This computation time is long enough to let all the methods reach a steady state, where further improvements are unlikely to be found. Ten runs are considered for each possible problem/method combination. The results of the experiments are reported in Tables 1-3. The first three columns are parameters of the problems, while the remaining columns are devoted, depending on the table, to the presentation of the average, best and worst results obtained by the following methods:

- **LS**: iterative application of the Local Search routine described in Section 2, starting from random feasible solutions;
- **LS+HMT**: the Heuristic Manipulation Technique described in Section 4 is run on top of the local search algorithm LS;
- **ACS+LS**: the local search algorithm LS is embedded in the Ant Colony System described in Section 3. Notice that in this case LS is run on the solutions built by ACS. This is the method presented by Gambardella and Dorigo (2000);
- **ACS+LS+HMT**: HMT is run on top of the ACS, which in turn embeds LS. This is the method presented by Montemanni et al. (2008).

Parameters of all the algorithms have been set according to Gambardella and Dorigo (2000) and Montemanni et al. (2008), where parameter tuning is accurately documented.

In all the tables bold text is used to highlight the best entry (entries) for each line. Accordingly, italic text is adopted for the worst entry (entries).

The results of Table 1 (average results) confirm that - apart from some sporadic cases - the best results are

[2] http://www.idsia.ch/~roberto/SOPLIB06.zip.

Table 1. Average results over ten runs.

Problem			Results			
n	r	p	LS	LS+HMT	ACS+LS	ACS+LS+HMT
200	100	1	102.5	*103.1*	90.3	**88.4**
300	100	1	*87.0*	85.3	76.4	**71.9**
400	100	1	*75.8*	72.6	64.1	**61.7**
500	100	1	*64.3*	62.0	**55.0**	55.1
600	100	1	*57.6*	57.2	49.8	**47.6**
700	100	1	*50.8*	46.6	42.6	**39.5**
200	1000	1	*1773.7*	1757.9	**1549.5**	1551.5
300	1000	1	*1878.6*	1837.7	1586.7	**1584.9**
400	1000	1	2064.8	*2065.7*	1811.1	**1773.1**
500	1000	1	*2172.1*	2151.4	1877.4	**1838.4**
600	1000	1	*2306.5*	2132.3	1986.7	**1957.6**
700	1000	1	*2272.5*	2092.7	1969.2	**1943.1**
200	100	15	*2373.2*	2337.3	2066.0	**2025.1**
300	100	15	*4108.7*	4098.9	3738.6	**3648.2**
400	100	15	*5215.7*	5183.9	5087.1	**4917.7**
500	100	15	*7005.0*	6987.3	6931.5	**6826.4**
600	100	15	7764.3	**7660.1**	*7806.6*	7733.0
700	100	15	*9640.5*	9507.1	9573.0	**9453.7**
200	1000	15	*25791.3*	25628.4	22602.9	**22419.2**
300	1000	15	*38204.1*	37939.3	34447.9	**34215.0**
400	1000	15	*51439.7*	51111.4	46638.6	**46183.2**
500	1000	15	*66344.8*	65635.0	62693.7	**61675.7**
600	1000	15	*76741.6*	76294.1	72701.1	**70136.9**
700	1000	15	*90514.0*	89973.7	85177.7	**82697.0**
200	100	30	*4454.3*	4430.8	4254.6	**4236.0**
300	100	30	*6476.2*	6469.6	6228.2	**6180.2**
400	100	30	*8747.3*	8718.3	8476.5	**8395.3**
500	100	30	*10508.2*	10432.0	10333.2	**10108.6**
600	100	30	*13277.2*	13215.0	13001.8	**12858.7**
700	100	30	15808.7	15810.0	*15905.8*	**15654.8**
200	1000	30	*42930.0*	42685.3	41371.6	**41315.2**
300	1000	30	*58099.5*	57710.0	55013.4	**54448.5**
400	1000	30	*90922.9*	90320.5	85979.6	**85922.9**
500	1000	30	105970.0	*106018.0*	101751.8	**100973.3**
600	1000	30	136792.7	*136948.8*	132314.3	**131733.7**
700	1000	30	*146869.5*	146000.0	141557.9	**141012.6**
200	100	60	***71749.0***	***71749.0***	***71749.0***	***71749.0***
300	100	60	*9732.9*	9731.1	**9726.0**	**9726.0**
400	100	60	15242.1	*15250.4*	15232.4	**15230.2**
500	100	60	*18316.5*	18312.6	18260.4	**18251.3**
600	100	60	*23427.0*	23388.6	23357.4	**23336.7**
700	100	60	*24244.5*	24218.3	24192.3	**24181.6**
200	1000	60	***71556.0***	***71556.0***	***71556.0***	***71556.0***
300	1000	60	*109796.5*	109750.3	109530.5	**109494.8**
400	1000	60	*141253.7*	141174.7	140994.9	**140936.9**
500	1000	60	*178953.5*	178803.5	**178478.1**	178500.4
600	1000	60	*215918.7*	215851.4	214970.2	**214875.1**
700	1000	60	246983.5	*247031.1*	246489.6	**246458.9**

Table 2. Best results over ten runs.

Problem			Results			
n	r	p	LS	LS+HMT	ACS+LS	ACS+LS+HMT
200	100	1	94	*96*	88	**83**
300	100	1	*79*	78	74	**65**
400	100	1	*69*	63	59	**54**
500	100	1	*57*	*57*	**51**	**51**
600	100	1	*47*	46	44	**42**
700	100	1	*41*	*41*	*41*	**32**
200	1000	1	*1750*	1722	1532	**1525**
300	1000	1	*1836*	1730	1536	**1518**
400	1000	1	*2032*	1989	1783	**1714**
500	1000	1	*2028*	2021	1840	**1770**
600	1000	1	*2248*	2055	1936	**1922**
700	1000	1	*2199*	2050	1912	**1873**
200	100	15	*2307*	2243	2002	**1926**
300	100	15	*4056*	3991	3520	**3383**
400	100	15	*5149*	5110	4838	**4735**
500	100	15	*6897*	6875	6584	**6536**
600	100	15	*7700*	7565	7610	**7217**
700	100	15	*9538*	9229	9383	**9124**
200	1000	15	*25622*	24810	21775	**21654**
300	1000	15	*37350*	37049	33533	**33148**
400	1000	15	*50890*	50614	45055	**44749**
500	1000	15	*65699*	63847	60175	**58316**
600	1000	15	74437	*74825*	70454	**68219**
700	1000	15	*89506*	88474	81439	**80887**
200	100	30	4406	*4408*	4247	**4216**
300	100	30	6406	*6428*	6151	**6148**
400	100	30	*8680*	8589	8289	**8276**
500	100	30	*10365*	10228	10047	**9974**
600	100	30	13123	*13138*	12810	**12689**
700	100	30	15695	15611	*15733*	**15180**
200	1000	30	*42300*	41835	41278	**41196**
300	1000	30	*56901*	56824	54367	**54278**
400	1000	30	*90212*	89383	85579	**85288**
500	1000	30	*105674*	105500	100453	**100112**
600	1000	30	*136386*	135077	**130244**	130541
700	1000	30	*146437*	143887	139769	**138695**
200	100	60	***71749***	***71749***	***71749***	***71749***
300	100	60	***9726***	***9726***	***9726***	***9726***
400	100	60	*15238*	**15228**	**15228**	**15228**
500	100	60	18271	*18282*	18246	**18240**
600	100	60	23370	*23374*	23342	**23259**
700	100	60	24191	*24192*	24151	**24149**
200	1000	60	***71556***	***71556***	***71556***	***71556***
300	1000	60	*109590*	109514	**109471**	**109471**
400	1000	60	*141096*	141021	140862	**140816**
500	1000	60	*178650*	178464	178323	**178212**
600	1000	60	*215650*	215469	214724	**214608**
700	1000	60	*246611*	246477	246128	**245753**

Table 3. Worst results over ten runs.

Problem			Results			
n	r	p	LS	LS+HMT	ACS+LS	ACS+LS+HMT
200	100	1	*109*	107	**93**	94
300	100	1	*93*	89	79	**76**
400	100	1	*82*	79	**67**	**67**
500	100	1	*70*	67	62	**58**
600	100	1	*64*	62	54	**53**
700	100	1	*61*	48	**44**	**44**
200	1000	1	*1809*	1798	**1572**	1576
300	1000	1	*1940*	1856	**1604**	1633
400	1000	1	*2096*	2094	1864	**1816**
500	1000	1	*2238*	2217	1921	**1888**
600	1000	1	*2419*	2181	2025	**2001**
700	1000	1	*2348*	2139	**2013**	**2013**
200	100	15	*2426*	2378	2186	**2182**
300	100	15	*4156*	4126	4013	**3883**
400	100	15	5287	5231	*5292*	**5135**
500	100	15	7074	7074	*7146*	**7013**
600	100	15	7877	**7715**	7992	*8030*
700	100	15	9704	**9628**	*9949*	9762
200	1000	15	*25944*	25877	24211	**22931**
300	1000	15	*38632*	38608	36808	**35687**
400	1000	15	*51999*	51598	51051	**49220**
500	1000	15	*66774*	66637	65177	**64765**
600	1000	15	*77807*	76582	75082	**72542**
700	1000	15	*91377*	91037	90808	**84321**
200	100	30	*4488*	4451	4269	**4256**
300	100	30	*6536*	6497	6338	**6257**
400	100	30	8783	*8787*	8697	**8639**
500	100	30	10574	10506	*10845*	**10362**
600	100	30	*13381*	13297	13311	**13084**
700	100	30	15920	**15889**	*16135*	15968
200	1000	30	*43178*	42831	41544	**41516**
300	1000	30	*59283*	58338	56606	**54725**
400	1000	30	*91210*	90931	86918	**86487**
500	1000	30	106507	*106568*	104420	**102118**
600	1000	30	137492	*137513*	135642	**133236**
700	1000	30	147098	146918	*148310*	**143460**
200	100	60	***71749***	***71749***	***71749***	***71749***
300	100	60	*9745*	9737	**9726**	**9726**
400	100	60	15253	*15262*	**15250**	**15250**
500	100	60	*18332*	18329	18303	**18278**
600	100	60	*23475*	23402	**23370**	23440
700	100	60	*24271*	**24230**	24249	24238
200	1000	60	***71556***	***71556***	***71556***	***71556***
300	1000	60	*109940*	109849	**109590**	**109590**
400	1000	60	*141418*	141251	**141118**	**141118**
500	1000	60	*179219*	179057	**178679**	178840
600	1000	60	*216294*	216093	215389	**215348**
700	1000	60	*247296*	**246902**	247045	247045

obtained when all the components are active (last column). As already observed by Montemanni et al. (2008), the improvements guaranteed by the use of HMT on top of ACS+LS are marginal, but exist. Analogously, we can observe how ACS+LS obtain far better results than LS alone. This phenomenon had already been observed by Gambardella and Dorigo (2000), but in our case the results obtained by the two methods appear much further apart than in the paper by Gambardella and Dorigo (2000). This suggests that the use of more difficult benchmarks amplifies the phenomenon.

It is also interesting to observe that HMT is able to improve the results of LS (column HMT+LS vs column LS) even though the results are worse than those achieved by method ACS+LS. Therefore there is a clear indication that HMT does its job, but it needs a strong underlying method in order to achieve really good results. Another indication is that most of the work done by the whole composite algorithm (last column) is hidden in the combination of ACS and LS methods. Notice, finally, that the worst results are consistently obtained by method LS.

The study of Tables 2 (best results) and 3 (worst results) leads to conclusions that are analogous to those in the comment in Table 1. It is however interesting to observe that the advantages of the use of HMT seems to be amplified in Table 2 and reduced in Table 3. This means that the results of HMT have a higher standard deviation over the ten runs considered. This is due to the heuristic nature of the rules used to select the active artificial constraints: it can lead to very good results in case of a happy choice, or to worse results in case of an unlucky selection.

6. CONCLUSIONS

The sequential ordering problem, which is faced in many practical applications, has been studied. The state-of-the-art algorithm for this problem is composed of different components, namely local searches, an ant colony system and a heuristic manipulation technique.

In this paper we have studied - from an experimental point of view - the impact of the different components on the performance of the full method. Specifically, we have run experiments with different combinations of components switched off, and we have analyzed how the results of the method were affected.

The conclusion was that the heuristic manipulation technique is able to slightly improve the results achieved by both the local searches alone, and the combination of local searches and the ant colony system. However, what appears to be crucial is the interaction between the ant colony system and the local search routines. This results was expected. Heuristic manipulation can be seen as a tool useful to further refine already good results, but here it is shown that it needs to be run on a very efficient underlying heuristic method (ant colony system plus local search, in our case) in order to obtain good results.

REFERENCES

Ascheuer N., 1995. *Hamiltonian path problems in the on-line optimization of flexible manufacturing systems*. PhD thesis, Technische Universität Berlin.

Ascheuer N., Escudero L.F., Grötschel M., Stoer M., 1993. A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). *SIAM Journal on Optimization*, 3:25–42.

Balas E., Fischetti M., Pulleyblank W.R., 1995. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 65:241–265.

Bentley J.L., 1992. Fast algorithms for geometric traveling salesman problem. *ORSA Journal on Computing*, 4:387–411.

Chen S., Smith S., 1996. Commonality and genetic algorithms. Technical Report CMU-RI-TR-96-27, The Robotic Institute, Carnagie Mellon University.

Dorigo M., Di Caro G., Gambardella L.M., 1999. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172.

Dorigo M., Gambardella L.M., 1997. Ant Colony System: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66.

Escudero L.F., 1988. An inexact algorithm for the sequential ordering problem. *European Journal of Operational Research*, 37:232–253.

Escudero L.F., Guignard M., Malik K., 1994. A Lagrangean relax-and-cut approach for the sequential ordering problem with precedence relationships. *Annals of Operations Research*, 50:219–237.

Gambardella L.M., Dorigo M., 2000. An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255.

Gambardella L.M., Mastrolilli M., Rizzoli A.E., Zaffalon M., 2001. An optimization methodology for intermodal terminal management. *Journal of Intelligent Manufacturing*, 12:521–534.

Guerriero F., Mancini M., 2003. A cooperative parallel rollout algorithm for the sequential ordering problem. *Parallel Computing*, 29(5):663–677.

Günther H.-O., Kim K.-H., 2006. Container terminals and terminal operations. *OR Spectrum*, 28:437–445.

Hernàdvölgyi I.T., 2003. Solving the sequential ordering problem with automatically generated lower bounds. In *Proceedings of Operations Research 2003*, pages 355–362.

Hernàdvölgyi I.T., 2004. *Automatically Generated Lower Bounds for Search*. PhD thesis, University of Ottawa.

Montemanni R., Smith D.H., Gambardella L.M., 2007. Ant Colony Systems for large Sequential Ordering Problems. In *Proceedings of IEEE SIS 2007*.

Montemanni R., Smith D.H., Gambardella L.M., 2008. A Heuristic Manipulation Technique for the Sequential Ordering Problem. *Computers and Operations Research*, to appear.

Pullyblank W., Timlin M., 1991. Precedence constrained routing and helicopter scheduling: heuristic design. Technical Report RC17154 (#76032), IBM T.J. Watson Research Center.

Savelsbergh M.W.P., 1990. An efficient implementation of local search algorithms for constrained routing problems. *European Journal of Operational Research*, 47:75–85.

Seo D.-I., Moon B.R., 2003. A hybrid genetic algorithm based on complete graph representation for the sequential ordering problem. In *Proceedings of GECCO 2003*, pages 69–680.

Stahlbock R., Voß S., 2008. Operations research at container terminals: a literature update. *OR Spectrum*, 30:1–52.

Steenken D., Voß S., Stahlbock R., 2004. Container terminal operation and operations research - a classication and literature review. *OR Spectrum*, 26:3–49.