Planning for Intra-block Remarshalling in a Container Terminal*

Jaeho Kang[1], Myung-Seob Oh[1], Eun Yeong Ahn[1], Kwang Ryel Ryu[1], and Kap Hwan Kim[2]

[1] Department of Computer Engineering, Pusan National University
San 30, Jangjeon-dong, Kumjeong-gu, Busan, 609-735, Korea
{jhkang, oms1226, tinyahn, krryu}@pusan.ac.kr
[2] Department of Industrial Engineering, Pusan National University
San 30, Jangjeon-dong, Kumjeong-gu, Busan, 609-735, Korea
kapkim@pusan.ac.kr

Abstract. Intra-block remarshalling in a container terminal refers to the task of rearranging export containers scattered around within a block into designated target bays of the same block. Since the containers must be loaded onto a ship following a predetermined order, the rearrangement should be done in a way that containers to be loaded earlier are placed on top of those to be loaded later to avoid re-handlings. To minimize time to complete a remarshalling task, re-handlings should be avoided during remarshalling. Moreover, when multiple yard cranes are used for remarshalling, interference between the cranes should be minimized. In this paper, we present a simulated annealing approach to the efficient finding of a good intra-block remarshalling plan, which is free from re-handlings at the time of loading as well as during remarshalling.

1 Introduction

In a container terminal, the efficiency of container loading onto a ship is highly dependent on the storage location of the containers in the yard. The containers must be loaded onto the ship in a predetermined loading sequence. Therefore, if the containers are located adjacently in the yard for successive loading, the travel distance of yard cranes and service time for container loading can be greatly reduced. However, gathering containers close to one another is not sufficient for efficient loading because the loading sequence must be followed. Gathering containers without considering the sequence may cause a considerable number of re-handlings at the time of loading because the container to be fetched out at each time is often found under others. In addition, time required for intra-block remarshalling must be minimized for efficient use of valuable resources such as yard cranes. In particular, when multiple yard cranes are used collaboratively for remarshalling, care must be taken to minimize crane interference.

* This work was supported by the Regional Research Centers Program (Research Center for Logistics Information Technology), granted by the Korean Ministry of Education Human Resources Development.

M. Ali and R. Dapoigny (Eds.): IEA/AIE 2006, LNAI 4031, pp. 1211–1220, 2006.

For a remarshalling task, cranes have to move tens of containers considering the sequence of container loading and interference between cranes. Therefore, the number of possible crane schedules is very large and finding the optimal one is intractable. Moreover, a crane schedule for remarshalling should be constructed in a reasonable time. In order to solve the scheduling problem efficiently, we applied simulated annealing[1]. Simulated annealing searches for a good partial order among containers to be rearranged that requires less time for remarshalling and does not make any re-handling during remarshalling and loading. A candidate solution expressed in the form of partial order graph does not regard the number of cranes to be used and their detailed movements but it can express a collection of possible re-handling-free total orders of container moves. Each candidate solution is used to construct a full crane schedule by a greedy evaluation heuristic with detailed crane simulation. By the heuristic, container moves are assigned to each crane and an appropriate order of these moves is determined under the constraint of the given partial order graph in a way of minimizing time for remarshalling, taking crane interference into account. The estimated time of the constructed full schedule is used as the quality of the partial order graph. Simulated annealing navigates the search space of partial order graphs using their evaluation results as a compass.

The performance of the evaluation heuristic is adjustable by changing the number of container assignments looked forward in selecting a container for the next assignment. Therefore, under the limited time of scheduling, there are two possible directions of improving the quality of schedule: trying more candidates or constructing a crane schedule with more container assignments looked forward. Experimental results show that a better remarshalling schedule can be obtained by carefully balancing between the two.

The next section explains the remarshalling problem in detail and reviews some related researches. The simulated annealing approach and the evaluation heuristic are presented in Section 3 and 4, respectively. In Section 5, the proposed method is tested and the results are summarized. Finally in Section 6, we give our conclusions and some topics for further research.

2 Intra-block Remarshalling

Intra-block remarshalling is a task of moving a set of target containers to designated bays within a block. We call the bays in which the target containers are located before remarshalling 'source bays' and the empty bays into which the target containers are moved 'target bays.' In Fig. 1, there are two source bays S_1 and S_2 and two target bays T_W and T_G. In this figure, each bay in the block is three rows wide and three tiers high. A bay can contain up to nine containers. Figure 1 also shows cross-sectional views of the source and target bays. A rectangle drawn with a solid line in the source bays is a container. In this example, fourteen target containers need to be moved during remarshalling. The number in each container refers to its order of loading at its target bay and an alphabetic character denotes the target bay for the container.

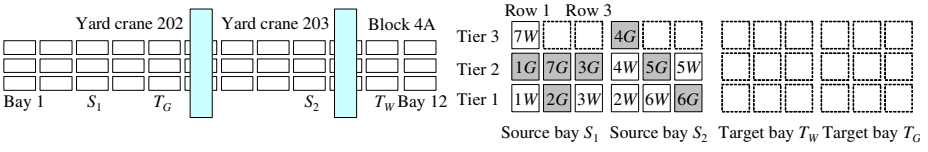

Fig. 1. An example of remarshalling problem

Because a predetermined sequence of container loading in each target bay must be followed, if a container for later loading is moved and stacked on top of an earlier one, the upper container should be moved to a temporary location during container loading. For example, in Fig. 1, if container $4G$ is moved and stacked on top of container $3G$ in one of stacks in T_G, then container $4G$ should be moved to somewhere before container $3G$ is being fetched out for loading. This temporary movement of container is called 're-handling.' Re-handlings incur extra works of yard cranes and delay the work of container loading that is one of the most expensive and time-critical services of container terminals. Therefore, target containers should be moved into their target bays in a manner of avoiding such re-handlings.

For a very simple solution of scheduling cranes for remarshalling, the moves of target containers can be sequenced in the reverse order of container loading. For example, in Fig. 1, a yard crane can move container $7W$ first into its target bay T_W, container $6W$ the next, and so on. However, this approach may cause additional work of yard cranes during remarshalling. For example, in Fig. 1, after moving container $7W$, a yard crane has to move container $5G$ temporarily to some other location before moving container $6W$. Later in the remarshalling, container $5G$ will be moved again to be stacked into its proper target bay. These temporary movements of containers during remarshalling are also called re-handlings and they delay the work of remarshalling that seizes valuable resources such as yard cranes. Therefore, this type of re-handlings should also be avoided to finish remarshalling as early as possible.

A re-handling-free schedule does not incur any re-handling situation during remarshalling as well as during loading. For a schedule to be free of re-handlings, the following two main constraints must be satisfied.

Constraint-Re-handling-Free-In-Loading: After remarshalling, each container should be loaded onto the ship without re-handling.

Constraint-Re-handling-Free-In-Remarshalling: Each container must be moved from its source bay to its target bay without re-handling.

There has been a little research on the issue of remarshalling. Kim and Bae presented a two-stage approach to planning remarshalling.[2] In the first stage, containers to be moved and the target bays of selected containers are determined in a manner of minimizing the number of containers to be moved by considering the given ship profile. In the second stage, the sequence of container moves is determined to minimize crane movements. They assumed that remarshalling is

started and finished before the loading sequence is determined and a single crane is used for remarshalling. Moreover, they did not consider the specific storage slots for containers in the bays before and after moving. In contrast, our paper assumes that multiple cranes carry out remarshalling after the loading sequence is determined. Furthermore, we consider the specific storage slots of containers in the bays and interference between cranes.

Some studies have developed strategies that determine a desirable stacking location for each container that comes into the yard. Dekker and Voogd described and compared various stacking strategies.[3] Each stacking strategy is evaluated by a few measures such as the number of re-handling occasions and workload distribution of yard cranes. Kim *et al.* derived a decision rule to locate export containers from an optimal stacking strategy to minimize the number of re-handlings in loading.[4] They assumed that containers are loaded onto a ship in the decreasing order of container weight group. These two studies tried to reduce the number of re-handlings by assigning the location of each container that comes into the yard more carefully. However, re-handlings in loading cannot be avoided completely for various reasons such as imprecise container weight information[5] and insufficient yard storage. Therefore, remarshalling may not be completely removed in most of the practical situations.

Finally, there are studies on crane scheduling particularly when multiple cranes are used in a block. Ng presented a dynamic programming-based heuristic approach to crane scheduling to minimize the waiting time of trucks.[6] Dynamic programming is used to partition the block into ranges in which each crane can move; these ranges help crane interference to be resolved. However, these studies cannot be used to schedule cranes for remarshalling directly because a primitive task of remarshalling is moving a container from one place to another in the same block rather than giving a container to or taking a container from a truck. In addition, there are some order constraints among container moves imposed by the sequence of container loading. Therefore, a crane scheduling method that can efficiently consider the characteristics of remarshalling is required.

3 Using Simulated Annealing for Remarshalling

This section describes a method of deriving a partial order graph from the locations of containers before and after moving. The derived partial order graph is used as an initial solution of simulated annealing. An efficient method of generating valid neighbors from the current partial order graph is also presented in this section.

3.1 Determining Target Locations for Containers

In this sub-section, we present a heuristic method of determining target locations of containers under the two main constraints introduced in the previous section. Figure 2 shows a step-by-step example of the heuristic. In the initial state Fig. 2(a), containers $7W$, $7G$, $3G$, $4G$, $5G$, and $5W$ are located on top of stacks

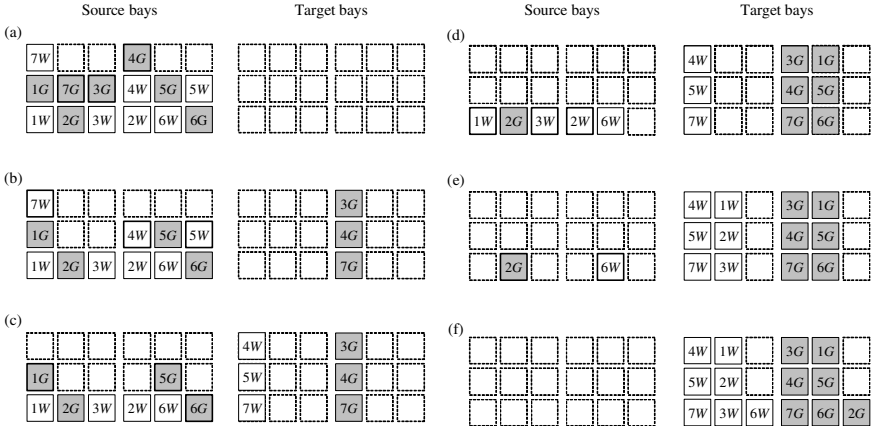

Fig. 2. Example of determining of the target locations of containers

in their source bays. These six containers can be moved to their target bays without violating **Constraint-Re-handling-Free-In-Remarshalling**. From these containers, some to the same target bay can be selected to be moved and stacked without breaking **Constraint-Re-handling-Free-In-Loading**. For example, if $7G$, $4G$, and $5G$ are selected, the order of moves for these selected containers should be $7G \rightarrow 5G \rightarrow 3G$ to satisfy **Constraint-Re-handling-Free-In-Loading**. Although there can be other container selections for a stack, each selection can have only one order of moving if the selected containers belong to the same target bay. Figure 3 shows the detailed procedure of container selection for filling a stack. By the procedure, $4G$, $7G$, and $3G$ are selected, and the order of moves is $7G \rightarrow 4G \rightarrow 3G$. The target locations of the three containers are now fixed. After removing these three containers from the source bays and placing them on a stack in their target bay, the state of bays will be shown as Fig. 2(b). The container selection and placing process is repeated until the target locations of all containers are determined. In this example, the set of containers $\{7W, 5W, 4W\}$ is the next selection, and their order is $7W \rightarrow 5W \rightarrow 4W$. When the target locations of all containers are determined, we can obtain a target configuration defined by the locations of the containers at target bays.

3.2 Deriving Partial Order Graph from the Locations of Containers

Figure 4(a) shows the source configuration defined by the initial positions of containers at the source bays for the remarshalling problem of Fig. 1. The source configuration constrains the order of container moves satisfying **Constraint-Re-handling-Free-In-Remarshalling**. If container c_1 is placed on top of another container c_2 on the same stack in a source configuration, c_1 should be moved to its target bay before c_2 to satisfy **Constraint-Re-handling-Free-In-Remarshalling**. Similarly, a target configuration, which is defined by the

1. Select target bay t that has the largest number of target containers, which can be moved without violating **Constraint-Re-handling-Free-In-Remarshalling**.
   1.1. For tie breaking, one of the target bays is chosen at random
   1.2. Let C_t be a set of containers that satisfy **Constraint-Re-handling-Free-In-Remarshalling** for selected target bay t.
2. From C_t, select a sufficient number of containers to fill one of empty stacks of t using the following priority rules. (In Fig. 2, a maximum of three containers can be stacked in a row)
   2.1. Select the container that has the largest number of target containers under it.
   2.2. If a tie is found, then select the container that has the largest number of target containers for t under it.
   2.3. If a tie still exists, then select one of these containers at random.

Fig. 3. Procedure for selecting containers to fill a stack

positions of containers at the target bays, also restricts the order of moves to fulfill the two main constraints. In a stack of a target configuration, if container c_1 is prior to container c_2 in loading, c_2 must be moved before c_1 so that c_1 should be placed on top of c_2. By combining these two types of ordering constraints imposed by the source and target configurations, a partial order graph can be derived to express all the possible re-handling-free sequences of moves under the two configurations. The derived partial order graph is shown in Fig. 4(c). A solid arrow in the figure represents an order constraint issued by the source configuration, and a dotted arrow is a constraint derived from the target configuration. In our experiment, a derived partial order graph is used for the initial solution of simulated annealing.

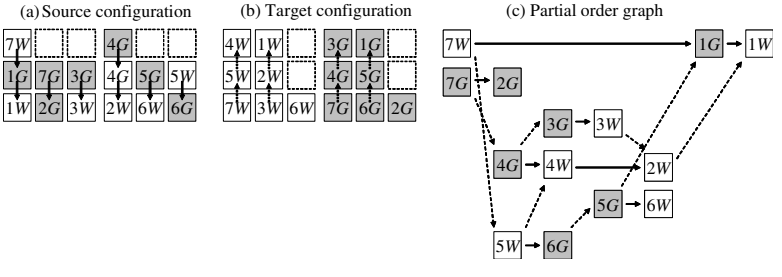

Fig. 4. Partial ordering of container moves

3.3 Generating Neighborhood Solutions

A different partial order graph can be obtained by slightly modifying the current one. Figure 5 shows an example of generating neighbors from the partial order graph in Fig. 4(c). In the graph, two containers on different stacks of the same target bay are selected at random for swapping. After swapping the

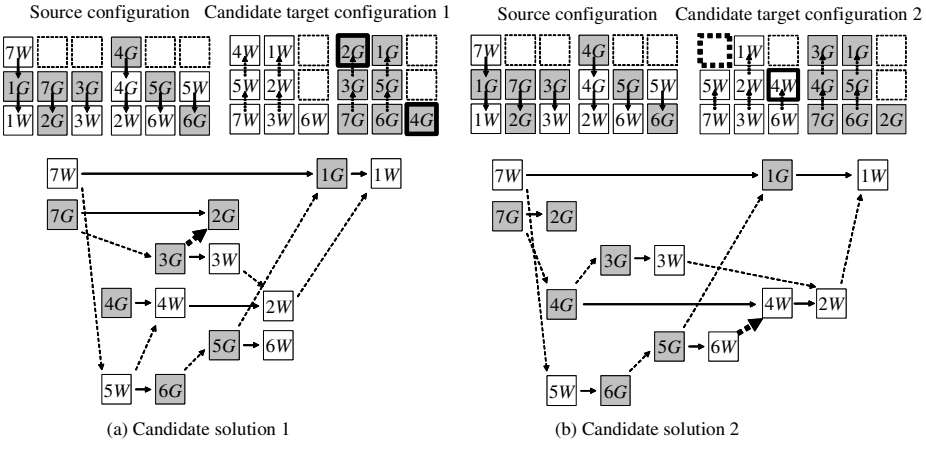

Fig. 5. An example of generating neighbors from the partial order graph in Fig. 4(c)

selected containers, a new partial order graph can be generated. For example, in Fig. 5(a), container $2G$ and $4G$ are selected and a neighbor, which is slightly different with the partial order graph of Fig. 4(c), is obtained. Note that some containers should be reordered in their stacks according to their turns of loading to satisfy **Constraint-Re-handling-Free-In-Loading**. A single container can be transferred to another non-full stack of the same target bay by swapping it with a virtual container as shown in Fig. 5(b).

A container swapping does not always lead to a valid partial order graph. When a cycle exists in a modified graph, no re-handling-free total order can be derived from it. Another swapping is tried to the original graph if the previous swapping introduces a cycle. Cycle detection can be performed in a linear time proportional to the number of containers.[7]

4 Constructing a Schedule from Partial Order Graph

This section presents a detailed description of the evaluation heuristic. The evaluation heuristic constructs an executable crane schedule by assigning each move of container to one of cranes and by determining the order of moves archived by each crane. It also detects crane interference by simulation based on the state-transition-graph model. Figure 6 shows a snapshot of constructing a crane schedule using the heuristic. In this figure, some container moves have been already assigned and currently crane A is moving container $2W$. A new container assignment is required to the idle crane B that just finished the work of moving container $3W$. Under the constraints of the given partial order graph, container $1G$ and $6W$ are two possible containers for the assignment to crane B. The heuristic simulates all the possible container assignments and selects the best-looking one to construct the partially built crane schedule incrementally.

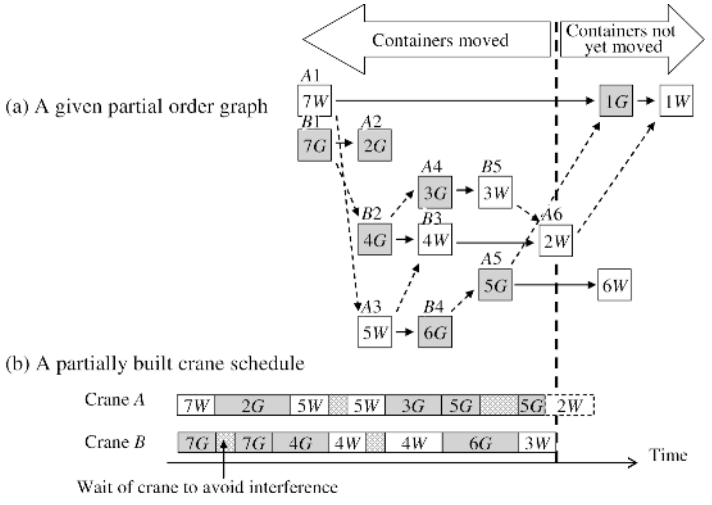

Fig. 6. An example of partially built crane schedule by the evaluation heuristic

The heuristic uses a depth-limited branch-and-bound search[8] to find the best-looking container. Figure 7 shows a tree that was constructed by the heuristic with the maximum depth of four. For each of the containers, there are two possible scenarios of avoiding crane interference when two cranes are used for remarshalling and interference is detected by simulation. Interference between the cranes can be resolved by giving the priority of working to one of the two cranes. The crane that is given the priority works without waiting. In contrast, the other should wait at the location nearest to the crane with priority to avoid interference. Therefore, a node in the tree represents both of crane priority and container assignment. The crane that has priority in each node is expressed in underline face. A node is evaluated by delay time, which is denoted on the right of the node, during the work of moving the assigned container. In Fig. 7, container $1G$ is selected for the assignment of a new container to crane B, and crane A will take the priority in moving container $2W$. The partially built crane schedule of Fig. 6 is expanded by adding the selected container $1G$ to crane B. Another tree search will be performed for the next new container assignment for crane A because crane A will finish its work of moving container $2W$ earlier than crane B. This process is repeated until all the containers are assigned to one of the cranes.

The maximum depth L of the depth-limited search is adjustable. Usually a better crane schedule can be obtained with higher L. However, higher L requires longer time in evaluation and results less trials in the level of simulated annealing if the time for crane scheduling is limited. The results of the experiment show that a better remarshalling schedule can be obtained by adjusting the balance between trying more partial order graphs and evaluating each graph with more forward looking.

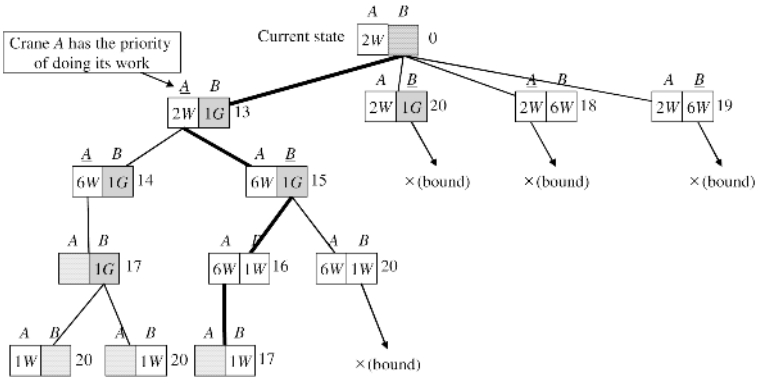

Fig. 7. Depth-limited branch-and-bound for assigning a container to idle crane B

5 Results of Experiment

For the experiment, we chose an environment with 33 bays in a block, each bay consisting of nine rows by six tiers. Two non-crossing cranes were used for remarshalling and the cranes required at least a five-bay gap for suitable working clearance. Four empty target bays are used to stack 196 target containers. Initially, target containers were randomly scattered within the block. Five different scenarios were used to test our approach to problems of various difficulties by changing the crowdness of target containers in the source bays. The time of searching for the remarshalling schedule was limited to 10 minutes. About 15,800 candidate solutions were evaluated in this time limit by depth-limited branch-and-bound with $L = 1$. About 14 candidates were evaluated with $L = 8$ in the same time limit. We also had tried another local-search algorithm, hill-climbing search, which did not show a better performance than simulated annealing. A Pentium PC with 3.2GHz was used for the experiments and each experiment was repeated ten times.

Figure 8 shows the performance of search with different L for each scenario. The graph depicts the relative estimated time of generated remarshalling schedules compared to those generated by search with $L = 1$. It is easily noticeable that higher L does not always give a better remarshalling schedule when the running time is limited. There is a point of balance between the maximum depth in evaluation and the number of evaluations for each scenario.

6 Conclusions

This paper presented a method of generating a remarshalling schedule for multiple cranes. Partial order graph is used to make a schedule to be free of rehandlings, and simulated annealing is applied to minimize time required for remarshalling at the level of partial order graphs. We also presented an evaluation heuristic for constructing a full crane schedule from a partial order graph.

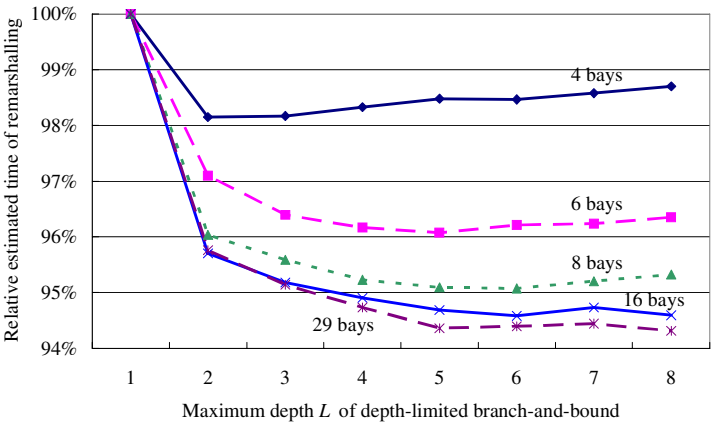

Fig. 8. The relative estimated time of remarshalling with schedules generated by simulated annealing with various maximum depths of evaluation heuristics

Our proposed method can generate an efficient crane schedule in a reasonable time. The results of the experiment show that a better remarshalling schedule can be obtained by carefully adjusting the running time of the evaluation heuristic under a limited time of computation. For further research, we are interested in finding a smart way of generating neighbors by selecting containers to be swapped considering feedback information that can be obtained in the process of applying the evaluation heuristic.

References

1. Aarts, E., Korst, J.: Simulated Annealing. Local Search in Combinatorial Optimization. John Wiley & Sons (1997) 91-120
2. Kim, K. H., Bae, J.-W.: Re-Marshaling Export Containers. Computer and Industrial Engineering **35(3-4)** (1998) 655–658
3. Dekker, R. and Voogd, P.: Advanced methods for container stacking. The Proceeding of International Workshop on Intelligent Logistics Systems (2005) 3–29
4. Kim, K. H., Park, Y. M., Ryu, K. R.: Deriving decision rules to locate export containers in container yards. European Journal of Operational Research, **124** (2000) 89–10
5. Kang, J., Ryu, K. R., Kim, K. H.: Determination of Storage Locations for Incoming Containers of Uncertain Weight. Proceedings of the 19th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent System (2006) 1159–1168
6. Ng, W. C.: Crane scheduling in container yards with inter-crane interference. European Journal of Operational Research **164** (2005) 64–78
7. Nivasch, G.: Cycle Detection Using a Stack. Information Processing Letters **90(3)** (2004) 135–140
8. Russell, S. J., Novrig, P.: Artificial Intelligence: A Modern Approach (second edition) Prentice Hall (2002)