A BEAM SEARCH ALGORITHM FOR MINIMIZING RESHUFFLE OPERATIONS AT CONTAINER YARDS

Kun-Chih Wu[1*] and Ching-Jung Ting[2]

[1] Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road
Chung-Li, Taiwan, R.O.C.
e-mail: s968907@mail.yzu.edu.tw

[2] Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road
Chung-Li, Taiwan, R.O.C.
e-mail: ietingcj@saturn.yzu.edu.tw

Abstract: The container relocation at container yards will influence the yard operator efficiency. The problem we tackled is to retrieval all containers from an initial configuration of given number of bays and tiers with minimum number of relocations. There is no new arriving container and only the containers in the same stack above the retrieved container can be relocated. We propose a beam search (BS) algorithm that is similar to the breadth-first branch-and-bound scheme, but only few permitted nodes in each level are kept for the further search. In order to select permitted nodes in each level, a look-ahead heuristic is applied for evaluation. We randomly generate 1920 instances with different configurations to test our BS algorithm. The results show that our BS algorithm can obtain optimal solutions in small size instances and near optimal solutions in larger size instances within effective computation time.

1. INTRODUCTION

With the increasing globalization, the demand of container transportation grows steadily. Terminals have to improve their performance efficiency to compete with others. Many issues are involved in affecting the terminal performance, such as labor utilization, traffic congestion in terminal, transfer time of yard cranes, etc. (Murty *et al*., 2005). Container relocation is a critical issue with respect to the terminal performance, which affects the turn-around time of trucks and of vessels and further impacts the performance of a terminal (Kim and Kim, 1999). The relocation of a container is performed when this container is above another container that is the target to be retrieved immediately. Since relocation is not a value-added but time consuming activity, it should be reduced as many as possible.

Containers are piled up at the container yard in such a manner to increase the space utilization. Block stacking is the most common way for container stowage at container yard (Kim and Hong, 2006), as shown in Figure 1. The numbered block indicates the retrieval order of each container, i.e. the smaller number indicates the higher priority. Though, the block stacking also indicates that only the top container is accessible. On the other hand, for export containers, loading sequence of containers is usually determined before loading to vessel (Lee and Chao, 2009). According to loading sequence, each export container has its own priority to be retrieved. It also implies the relocation occurs when a container with higher priority is underneath another container with lower priority.

Stowage planning is considered to decide the most suitable locations of containers at container yard (Avriel *et al*, 1998). It can be decomposed to two types, the static problem and the dynamic problem, in terms of the arrival/departure manner of containers. In the static problem, the sequence of containers is known in advance, while in the dynamic problem, the containers arrived or departed in a random manner.

For the dynamic problem, stochastic concept is usually adapted to estimate the expected number of relocations. Kim and Kim (1999) dealt with a dynamic relocation problem and presented a formula to interpret the relation between the stack height and the expected number of relocations. Kim *et al*. (2000) considered minimizing expected numbers of relocation at the export container yard. They proposed a dynamic programming model to provide an optimal decision of stack for minimizing expected number of relocations and a decision tree to make a fast decision.

In the real time cases, the exact location for relocating container has to be determined immediately. Therefore, Avriel *et al*. (1998) considered the problem is to minimize the total cost of reshuffles when containers on a vessel have to be loaded and unloaded from/to several ports. The authors proposed a binary programming model as well as a heuristic procedure..

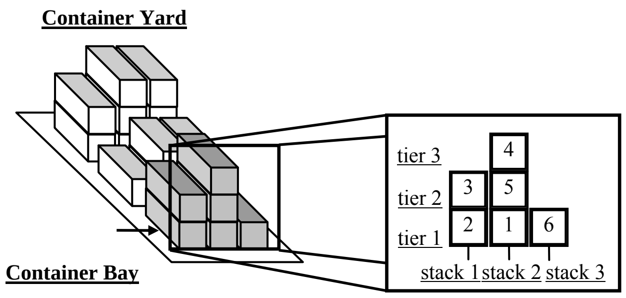

Figure 1. The Illustration of Containers Stacking at the Container Yard

Murty *et al.* (2005) provided a Decision-Support System (DSS) for Hong Kong International Terminals (HIT). The system achieved elastic capacity successfully, almost 50 percent improvement in capacity was obtained. They divided the real world problem into five related subproblems. Reshuffle problem is also an important issue considered in their study, in which a rule called the reshuffle index (RI) was performed to relocate containers. Yang and Kim (2006) addressed a general concept of relocation problem. They proposed both dynamic programming and genetic approach for a static grouped location problem, as well as three rules for the dynamic case.

Kim and Hong (2006) considered a container retrieval problem at the container yard. The objective is to minimize relocations when all containers at a yard-bay are retrieved according to a predetermined precedence. They proposed a branch-and-bound algorithm and a heuristic with respect to reducing the expected number of relocations during retrieval process. The empirical results showed that the error rate between the branch-and-bound and the heuristic is 7.3% in the first case and 4.7% in the later case.Caserta *et al* (2009) continued the study of Kim and Hong (2006). They also handled the container retrieval problem but applied an inspired metaheuristic, called corridor method (CM). The results indicated that CM can reach optimal solution in the small size problem within very short computational time. In this study, we also consider the same problem proposed by Kim and Hong (2006), but we use beam search algorithm to solve this problem.

The remainder of the paper is organized as follows. In section 2, the container reshuffle problem is defined and related heuristics are introduced. The beam search and the branch-and-bound algorithms are introduced in section 3. Section 4 presents the experiment results with large number of instances. Section 5 summarizes the findings in this research.

2. CONTAINER RESHUFFLE PROBLEM

Container relocation problem considered in this paper is to retrieve all containers at a yard-bay. We assume that each container with different retrieval priority and the initial configuration of the yard-bay are known in advance. In addition, we only consider a static problem that means the new arrival container is not allowed. The objective is to retrieve all containers at the yard-bay with minimum number of relocations. The optimal retrieval procedure is performed by determining a stack to relocate a container. Formally, this problem includes following five assumptions:

1) The maximum number of stacks and maximum number of tiers at a container bay are given, and the container only can be relocated within the yard-bay.
2) Every container at the yard-bay has the same size.
3) The initial configuration of the container bay and the retrieved sequence of each container are known in advance.
4) The new arrival container is not allowed during the retrievals.
5) The reshuffle operations only occur while a target container needs to be retrieved. It implies that pre-marshalling is not allowed in this problem.

We use the following notations to describe the problem:

H	: The maximum number of tiers for every stack.
N	: The number of containers in the initial configuration.
$R(i, j)$	: The container at the top of ith stack is relocated to jth stack, where $i, j \in \{1, 2, \ldots, W\}$
S_i	: The set represents the configuration of the ith stack, where $i \in \{1, 2, \ldots, W\}$. The elements in the set indicate the retrieval order of each container, and the last element is on the top of stack.
W	: The maximum number of stacks.
Y	: The set of stacks within the yard bay, where $Y = \{S_1, S_2, \ldots, S_W\}$

An example illustrated in Figure 2 has an initial configuration $Y = \{S_1, S_2, S_3\} = \{\{2, 3\}, \{1, 5, 4\}, \{6\}\}$. The container retrieving is named *target container* henceforth, and the container that blocks the target container retrieving is named *obstructive container*. At the first stage, the target container is container 1. The container 1 cannot be retrieved, because containers 4 and 5 are above it. The relocation is therefore necessary to free the target container. In the first stage, the relocation of container 4 from stack 2 to stack 1 is denoted as *R*(2, 1). Similar procedure is applied for remainder containers until all containers are retrieved, and the total number of reshuffles in this example is four. The solution can be expressed as all reshuffle operations involved during the retrieval procedure. The solution with four reshuffles can be simply expressed as {*R*(2, 1), *R*(2, 3), *R*(1, 3), *R*(1, 2)}.

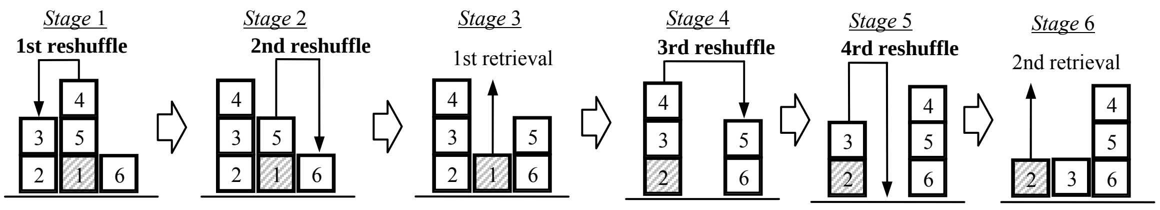

Figure 2: An Example of the Procedure for Retrieving and Reshuffling Containers (H = 3, W = 3)

2.2 Heuristic

There are three heuristics involved in this paper for solving container relocation problem. The first rule, called the lowest position (TLP), was proposed by Zhang (2000). TLP tends to force every stack with a close number of containers, so that average relocation can be reduced. This rule selects a stack that has the fewest containers. Break ties arbitrarily, if two stacks have the same lowest position. For example, in Figure 3 (a), the container 4 has to be relocated to other stacks. The stack 3 has lower position than stack 1, the container 4 then will be relocated to stack 3.

The second rule, called reshuffle index (RI) which computes the number of containers with higher priority than the obstructive container is located, was proposed by Murty *et al.* (2005). Since containers with higher priority underneath the relocated container lead to additional reshuffles. This rule chooses a stack that has lower RI to relocate container. If there are several stacks with the same RI, then break ties arbitrarily. An illustration shown as Figure 3 (b), the container 5 needs to be relocated. The stack 3 has one container with higher priority than container 5, so that stack 3 has lowest RI equal to one. The container 5 should be relocated to stack 3.

The last one is reshuffle index with look-ahead (RIL) that is an extension of reshuffle index rule. The difference between RI and RIL is the rule for breaking ties. In the RIL, the stack for relocating stack is first determined according to RI, and the RIL breaks ties by look-ahead mechanism. The look-ahead rule selects a stack, in which the highest priority of the containers is the lowest. The concept of the look-ahead rule is to avoid additional relocations in near future steps. If a relocated container is above a container that will be retrieved just after the target container, then an extra relocation will occurs in retrieving next container. In Figure 3 (c), the container 4 has to be retrieved, and stack 1 and stack 3 both have equal RI. If container 4 is relocated to stack 1, then one more relocation would be needed when retrieving container 2. According to RIL rule, stack 3 is chosen because the priority of container 3 is lower than the priority of container 2.

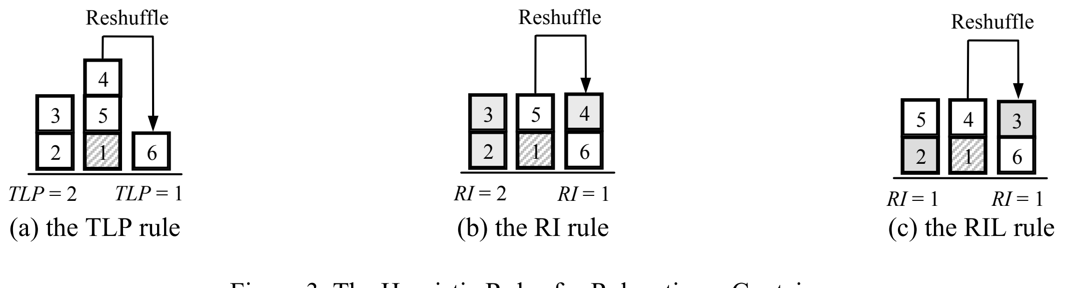

Figure 3. The Heuristic Rules for Relocating a Container

3. BEAM SEARCH

Beam Search (BS) is a heuristic based on breadth-first branch-and-bound algorithm. It was earliest applied in artificial intelligent to deal with the speech recognition (Lowerre, 1976). It has been then employed on the scheduling problem, like the studies conducted by Fox (1983) and Ow and Smith (1988). Ow and Morton (1988) later proposed a novel mechanism for the beam search, called Filter Beam Search, which gives a compromise between computational time and solution quality. They suggested that a rough evaluation resembles a filter is applied first to prune out the nodes, and only *α* nodes are retained. These *α* nodes are evaluated again by a caution estimated heuristic and only *ß* nodes are chosen. Sabuncuoglu and Bayiz (1999) applied filter beam search for scheduling problem with makespan and tardiness criteria. A tiny revision for their beam search is that nodes in the first level are evaluated without filter, and a filter is applied in remainder levels.

3.1 Standard Beam Search

BS follows breath-first search strategy in its searching scheme, in which the nodes are searched level by level. Unlike the general tree search scheme that generates every possible node to seek optimal solution, BS only keeps some promised nodes at each level for sprouting the descendants. In the beam search, only *ß* promised nodes at each level are required for the further search. The *ß* is so-called the beam width. The BS reduces a great deal of searching space via limiting the nodes retained at each level. Suppose *n* is the depth of a tree, the searching space of the BS includes *ßn* solutions against *n*! solutions contained in the original solution space. However, the optimality can not be confirmed by the beam search, since the optimal solution may be ignored during searching process. The beam search is hence a constructive heuristic.

The promised nodes can be determined simply by estimated rules or heuristics. The nodes with better estimated values have more possibility to gain better results. The concept of beam search is to keep the nodes with higher probability achieving optimal solution during the searching process. The search manner is different from the branch-and-bound that keeps the nodes via a strict restriction, but the beam search realize the concept in an opportunistic manner.

3.2 Beam Search for Container Reshuffle Problem

The decision involved in the container reshuffle problem is to decide which stack to relocate a container. The best sequence of stacks for relocations results in the minimum total number of reshuffles. A node in our beam search represents a partial solution that includes the stacks for relocations from root to current node.

Generally, BS chooses the best *ß* nodes at each level as beam nodes. As illustration in Figure 4, there are four nodes in the level three generated by pervious beam nodes (N_3 and N_4), and two of them generated from the same node (N_3) are chosen as next beam nodes. The search manner concentrates on the partial beams that have better performance. It can increase the possibility to gain better solution via exploiting the partial beams, however it also increases the possibility to fall into the local optimal solution.

In this study, we apply a standard beam search to solve container relocation problem, in which the filter mechanism is not considered. The search begins from an initial configuration, and then the nodes at first level are sprouted out. Every node is evaluated by a given global heuristic to estimate its upper bound. The global heuristic can use one of those three mentioned in section 2.2 to estimate an upper-bound. Only *ß* nodes with lower upper-bound will be selected as beam nodes. The selected beam nodes then sprout out their descendants for the next level. Repeating the procedure until a feasible solution is obtained. An illustration shown as Figure 4, the *ß* of the beam search is set as two. The evaluation function is performed to estimate the upper bound and the search scheme selects the best two descendants to be beam nodes. Only the beam nodes have to generate the descendants at each level, remaining nodes are ignored. The process is terminated at level four, in which a complete solution is obtained. The minimum relocation is four according to the beam nodes at last level.

Let N_0 indicates the root, *B* is the set of beam nodes, *C* is the nodes pool of all descendants that generated in the same level. The procedure of beam search is presented as follows:

```
Beam Search Procedure
Step 1: [initialization]
    Set $B = \{N_0\}$
Step 2: [retrieval procedure]
    Step 2.1: For each node $N \in B$
        Do
            If target container $T$ is on the top of a stack in the bay $Y$ of the node $N$
                Retrieve the $T$ from $Y$
        Until there is a container blocked the target container.
    Step 2.2: If a configuration $Y$ in a node $N \in B$ is empty, then go to Step 5.
Step 3: [branch procedure]
    Step 3.1: Set the descendants pool $C = \{\emptyset\}$
    For each node $N \in B$
```

```
        Let $S_t$ is the stack with target container $T$ in node $N$, and $D = N$
        Step 3.2: **For** each stack $S$ in node $D$
            **If** stack $S$ is not full and $S \neq S_t$
                Perform the relocation $R(S_t, S)$
        Step 3.3: $C = C \cup D$.
Step 4: **[relocation procedure]**
    Step 4.1: For each node $D \in C$
        Perform a **Heuristic** to evaluate node $D$.
    Step 4.2: Select best $Min\{\beta, |C|\}$ nodes from $C$ to replace beam nodes in $B$. Go to Step 2.
Step 5: **[termination]**
    Return $Min\{$number of relocations for node $N$, $N \in B\}$. Stop the procedure.
```

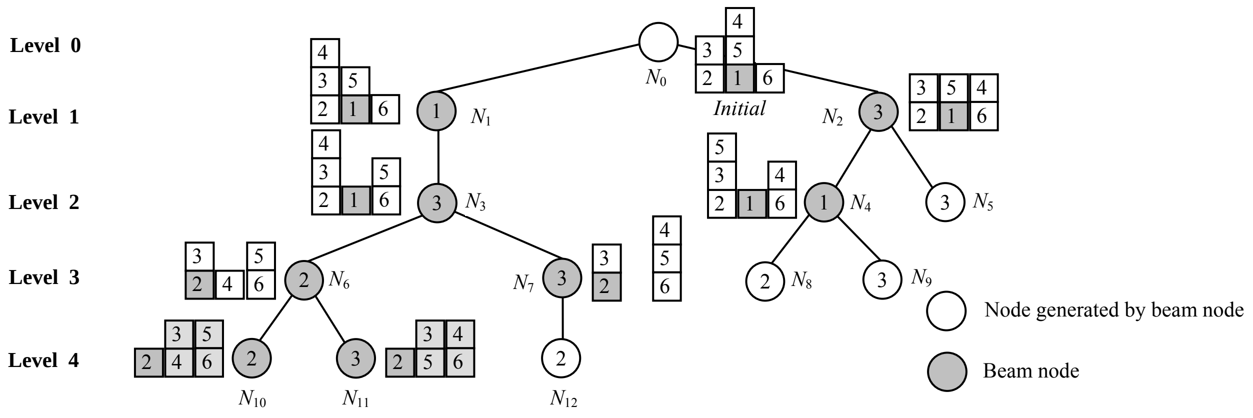

Figure 4. Representation of beam search for container reshuffle problem

3.3 Branch and Bound

We developed a brand-and-bound procedure in this paper as well. The depth-first procedure is applied to reduce memory usage. The node that is sprouted out first will be visited first. Once a feasible solution that has retrieved every container at the yard-bay is found, the procedure will then backtrack to its parent to search another branch. Because the level indicates the number of relocations, if a feasible solution is found at level k, then the further search over level k-1 is not necessary. All generated nodes will be visited to confirm the optimal solution is achieved.

4. EXPERIMENTAL RESULTS

Both the beam search and the branch-and-bound were tested to compare the performance of each other. The three heuristics introduced in section 2.2 were tested as well. All algorithms were coded in Borland C++ Builder 6.0, and run on a PC with AMD Athlon Dual Core +3800 2.0 GHz and 2 GB memory. The testing data are generated randomly for experiments.

4.1 Testing Data

For comparing the performance of algorithms, different sizes of problem were generated. Kim and Hong (2006) mentioned that a yard bay usually contained 4-7 tiers with 6-10 stacks. Hence, without loss of generality, we created problems between 3-8 tiers and 3-10 stacks. We generated 48 problem classes, and each of them has 40 instances, so there are total 1920 instances generated. In each instances, we randomly located containers with different priorities into a yard-bay. In addition, the number of containers (N) is determined by W and H. Given W and H, the total number of slots at the yard-bay is $W \times H$, and the most number of relocations to retrieve one container is $H - 1$. For confirming the feasibility, Eq. (1) ensures that enough space is reserved for relocations of containers.

$$N = W \times H - (H - 1) \tag{1}$$

4.2 Computational Results

We first tested the heuristics with three various rules on each class, and the results are shown as Table 1. The first column and sixth column indicate the number of stack (*W*) with range from 3 to 10. The second column and seventh column indicate the number of tiers (*H*) with range from 3 to 8. The column 3-5 and column 8-10 represent the results of three heuristics. Each cell shows the average number of reshuffles over 40 instances. The results demonstrated that the RIL outperforms both TLP and RI at every class. Because the computational times of the heuristics are quite fast (less than 0.01 seconds), they are not included in the table.

The beam searches with different beam width as well as branch-and-bound algorithm were tested. Ideally, a more accurate estimation leads to a better result of beam search. Therefore, the RIL is embedded in our beam search for global evaluation, due to the better estimation of relocations.The detail results of beam search in each class were shown in Table 3. As the beam width increases, the elapsed time increases linearly, but the improving rate on the average number of reshuffles decreases. The average number of relocations will finally converge to a certain value. An exhaustive search may take too much time but gain little improvement. Hence, the beam search provides a compromise between time and solution quality.

5. CONCLUSION

In this paper, we deal with a container reshuffle problem, in which each container has a particular priority to retrieve. We first proposed a modified heuristic RIL and compared the heuristic with the other two existing heuristics (RI and TLP). Then, we proposed a beam search heuristic to solve the problem, in which the RIL is embedded to provide a global evaluation. We randomly generate 48 problem classes and 40 instances in each class, and there were total 1920 instances generated. The experiments were performed by running those algorithms on generated instances.

The computational results show that the proposed modified heuristic outperforms the other two existing heuristics at every problem class. A branch-and-bound also provided to compare with the beam search. The result shows that the beam search can reach most optimal solution when beam width (β) equal to 15, and all optimal solution can be obtained as beam width equal to 70.

6. REFERENCES

Avriel, M., Penn, M., Shipirer, N. and Witteboon, S. (1998) Stowage Planning for Container Ships to Reduce the Number of Shifts. *Annals of Operations Research, 76:* 55-71.

Caserta, M. and Voß, S. (2009) A Corridor Method-based Algorithm for the Pre-marshalling Problem. *Lecture Notes in Computer Science, 5484:* 788-797.

Caserta, M., Schwarze, S. and Voß, S. (2009) A New Binary Description of the Blocks Relocation Problem and Benefits in a Look Ahead Heuristic. *Lecture Notes in Computer Science, 5482:* 37-48.

Fox, M. S. (1983) *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*, PhD Dissertation, Computer Science Department, Carnegie-Mellon University.

Kim, K. H. and Hong, G. P. (2006) A Heuristic Rule for Relocating Blocks. *Computers & Operations Research, 33:* 940-954.

Kim, K. H. and Kim, H. B. (1999) Segregating Space Allocation Models for Container Inventories in Port Container Terminals. *International Journal of Production Economics, 59:* 415-423.

Kim, K.H. and Kim, H.B. (2002) The Optimal Sizing of the Storage Space and Handling Facilities for Import Containers. *Transportation Research Part B, 36:* 821-835

Kim, K. H., Park, Y. M. and Ryu, K. R. (2000) Deriving Decision Rules Locate Export Containers in Container Yards. *European Journal of Operational Research, 124:* 89-101

Lowerre, B.T. (1976) The HARPY Speech Recognition System. Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA.

Lee, Y. and Chao, S-L. (2009) A Neighborhood Search Heuristic for Pre-marshalling Export Containers. *European Journal of Operational Research, 196:* 468-475.

Murty, K.G., Liu J., Tseng, M. M., Leung, E., Lai, K-K. and Chiu, W.C. (2005) Hong Kong International Terminals Gains Elastic Capacity Using a Data-intensive Decision Support System. *Interfaces, 35:* 61-75.

Ow, P. S. and Morton, T. E. (1988) Filtered Beam Search in Scheduling. *International Journal of Production Research 26:* 35-62

Ow, P. S. and Morton, T. E. (1989) The Single Machine Early/Tardy Problem. *Management Science, 35:* 177-191.

Ow, P. S. and Smith, S. F. (1988) Viewing Scheduling as an Opportunistic Problem-solving Process. *Annals of Operations Research, 12:* 85-108.

Sabuncuoglu, I. and Bayiz, M. (1999) Job Shop Scheduling with Beam Search. *European Journal of Operational Research, 118:* 390-412.

Sabuncuoglu, I., Gocgun, Y. and Erel, E. (2008) Backtracking and exchange of information - Methods to enhance a beam search algorithm for assembly line scheduling. *European Journal of Operational Research, 186:* 915-930.

Valente, J. M. S. and Alves, R. A. F. S. (2005) Filtered and Recovering Beam Search Algorithms for the Early/Tardy Scheduling Problem with No Idle Time. *Computers & Industrial Engineering, 48:* 363–375.

Yang, J.H. and Kim, K. H. (2006) A Grouped Storage Method for Minimizing Relocations in Block Stacking Systems. *Journal of Intelligent Manufacturing, 17:* 453-463.

Zhang, C. (2000) *Resource Planning in Container Storage Yard*, PhD Dissertation, Department of Industrial Engineering, The Hong Kong University of Science and Technology.

Table 1. Average Numbers of Reshuffles of Evaluation Heuristics in Each Class

class		TLP	RI	RIL	class		TLP	RI	RIL
W	*H*				*W*	*H*			
3	3	3.58	3.42	3.4	7	3	11.50	10.10	9.88
3	4	6.67	6.10	5.95	7	4	19.40	20.10	18.80
3	5	10.60	9.80	9.45	7	5	33.00	30.90	28.30
3	6	15.40	13.60	13.20	7	6	45.80	45.00	41.30
3	7	20.20	18.10	17.30	7	7	66.10	59.80	54.50
3	8	27.20	24.10	22.90	7	8	82.60	76.60	68.70
4	3	5.67	5.03	4.92	8	3	11.60	11.90	11.70
4	4	10.50	9.05	8.80	8	4	22.90	20.90	20.00
4	5	16.30	14.50	13.70	8	5	37.60	34.70	32.10
4	6	23.20	19.10	17.90	8	6	55.00	50.60	46.50
4	7	33.40	28.90	27.60	8	7	79.20	68.40	61.60
4	8	44.80	37.90	34.60	8	8	93.90	88.90	79.50
5	3	6.95	5.90	5.75	9	3	14.10	12.40	12.10
5	4	14.40	12.20	11.80	9	4	24.80	24.70	23.60
5	5	21.00	18.10	17.40	9	5	44.50	37.50	35.10
5	6	31.80	25.60	24.10	9	6	61.50	55.70	50.20
5	7	46.00	36.30	33.70	9	7	88.50	76.70	68.80
5	8	61.60	49.80	44.50	9	8	109.00	97.90	87.40
6	3	8.95	7.92	7.80	10	3	14.90	14.60	14.40
6	4	16.00	13.20	12.80	10	4	28.90	26.20	24.70
6	5	26.90	22.60	21.40	10	5	47.90	41.20	38.50
6	6	41.20	32.60	30.10	10	6	67.00	60.20	54.40
6	7	56.90	45.50	41.70	10	7	101.00	85.10	75.80
6	8	75.00	57.20	53.90	10	8	121.00	111.00	96.70
					avg		39.708	34.950	32.069

Table 2. Average Numbers of Reshuffles of Beam Search with Varied Beam Width and the Branch-and-Bound

class		Beam Search												B&B	
		BW=5		BW =10		BW =15		BW =20		BW =50		BW =70		Opt.	
W	*H*	**Res**	**Time**	**Res**	**Time**	**Res**	**Time**	**Res**	**Time**	**Res**	**Time**	**Res**	**Time**	**Res**	**Time**
3	3	**3.38**	0.000	**3.38**	0.000	**3.38**	0.000	**3.38**	0.000	**3.38**	0.000	**3.38**	0.000	**3.38**	0.000
3	4	**5.67**	0.000	**5.67**	0.000	**5.67**	0.000	**5.67**	0.000	**5.67**	0.000	**5.67**	0.000	**5.67**	0.000
3	5	**8.40**	0.000	**8.40**	0.000	**8.40**	0.000	**8.40**	0.001	**8.40**	0.001	**8.40**	0.001	**8.40**	0.001
3	6	**11.50**	0.001	**11.50**	0.001	**11.50**	0.001	**11.50**	0.002	**11.50**	0.003	**11.50**	0.004	**11.50**	0.009
3	7	15.10	0.001	15.10	0.001	**15.00**	0.002	15.10	0.003	**15.00**	0.006	**15.00**	0.008	**15.00**	0.060
3	8	19.00	0.002	18.90	0.003	18.80	0.004	18.80	0.005	18.70	0.010	**18.60**	0.014	**18.60**	1.510
4	3	**4.85**	0.000	**4.85**	0.000	**4.85**	0.000	**4.85**	0.000	**4.85**	0.000	**4.85**	0.000	**4.85**	0.000
4	4	**8.43**	0.000	**8.43**	0.001	**8.43**	0.001	**8.43**	0.001	**8.43**	0.002	**8.43**	0.003	**8.43**	0.007
4	5	12.30	0.001	**12.20**	0.002	**12.20**	0.002	**12.20**	0.003	**12.20**	0.006	**12.20**	0.007	**12.20**	0.241
4	6	15.80	0.002	15.70	0.003	15.70	0.004	15.70	0.005	**15.60**	0.012	**15.60**	0.016	**15.60**	11.000
4	7	23.00	0.003	22.90	0.006	22.80	0.008	22.70	0.010	22.60	0.023	22.60	0.031	–	–
4	8	28.90	0.005	28.40	0.009	28.20	0.013	28.10	0.017	28.00	0.038	27.90	0.052	–	–
5	3	**5.75**	0.000	**5.75**	0.000	**5.75**	0.001	**5.75**	0.000	**5.75**	0.002	**5.75**	0.002	**5.75**	0.002
5	4	**11.00**	0.001	**11.00**	0.002	**11.00**	0.002	**11.00**	0.002	**11.00**	0.006	**11.00**	0.008	**11.00**	0.308
5	5	15.70	0.002	15.70	0.004	**15.60**	0.005	**15.60**	0.007	**15.60**	0.014	**15.60**	0.019	**15.60**	40.900
5	6	21.30	0.004	21.30	0.007	21.20	0.009	21.20	0.013	21.10	0.030	21.10	0.040	–	–
5	7	28.30	0.006	28.10	0.012	28.10	0.017	27.90	0.022	27.80	0.052	27.80	0.070	–	–
5	8	37.50	0.011	37.00	0.020	36.90	0.029	36.90	0.038	36.50	0.089	36.40	0.121	–	–
6	3	**7.65**	0.000	**7.65**	0.002	**7.65**	0.002	**7.65**	0.002	**7.65**	0.004	**7.65**	0.005	**7.65**	0.034
6	4	12.10	0.002	**12.00**	0.003	**12.00**	0.004	**12.00**	0.005	**12.00**	0.012	**12.00**	0.016	**12.00**	11.200
6	5	19.40	0.004	19.40	0.007	19.40	0.011	19.40	0.013	19.40	0.030	19.30	0.041	–	–
6	6	26.50	0.007	26.30	0.013	26.30	0.019	26.30	0.025	26.10	0.057	26.10	0.079	–	–
6	7	35.80	0.012	35.60	0.023	35.30	0.034	35.20	0.045	35.00	0.107	34.90	0.148	–	–
6	8	44.30	0.018	44.20	0.036	43.80	0.052	43.50	0.068	43.20	0.162	43.20	0.226	–	–
7	3	**8.95**	0.001	**8.95**	0.002	**8.95**	0.003	**8.95**	0.004	**8.95**	0.007	**8.95**	0.011	**8.95**	0.834
7	4	15.50	0.003	15.50	0.005	15.50	0.008	15.50	0.010	15.50	0.023	15.50	0.031	–	–
7	5	21.60	0.005	21.50	0.011	21.40	0.016	21.40	0.020	21.40	0.049	21.40	0.066	–	–
7	6	31.60	0.011	31.30	0.022	31.30	0.032	31.20	0.042	31.00	0.099	31.00	0.136	–	–
7	7	40.50	0.019	40.20	0.036	40.00	0.053	39.80	0.069	39.50	0.165	39.40	0.229	–	–
7	8	51.70	0.031	51.00	0.060	50.80	0.089	50.50	0.116	50.00	0.277	49.90	0.383	–	–
8	3	**9.72**	0.002	**9.72**	0.003	**9.72**	0.004	9.72	0.005	**9.72**	0.011	**9.72**	0.015	**9.72**	6.510
8	4	17.90	0.004	17.90	0.008	17.90	0.012	17.90	0.016	18.00	0.035	18.00	0.048	–	–
8	5	25.60	0.009	25.50	0.017	25.50	0.025	25.40	0.033	25.40	0.079	25.40	0.107	–	–
8	6	36.80	0.017	36.40	0.033	36.20	0.048	36.20	0.064	36.00	0.153	36.00	0.209	–	–
8	7	46.20	0.028	45.60	0.054	45.60	0.080	45.60	0.105	45.20	0.251	45.10	0.345	–	–
8	8	59.30	0.048	58.50	0.091	58.50	0.136	58.00	0.179	57.50	0.427	57.30	0.593	–	–
9	3	**11.40**	0.002	**11.40**	0.004	**11.40**	0.007	**11.40**	0.009	**11.40**	0.018	**11.40**	0.025	**11.40**	147.000
9	4	19.20	0.006	19.20	0.013	19.10	0.018	19.10	0.024	19.10	0.055	19.10	0.075	–	–
9	5	28.90	0.013	28.90	0.025	28.80	0.038	28.80	0.050	28.70	0.118	28.70	0.161	–	–
9	6	40.80	0.025	40.60	0.048	40.50	0.072	40.30	0.095	40.00	0.227	40.00	0.316	–	–
9	7	52.80	0.043	52.40	0.084	52.20	0.124	52.00	0.162	51.70	0.396	51.50	0.547	–	–
9	8	68.10	0.073	67.20	0.139	67.20	0.209	66.90	0.277	66.40	0.665	66.00	0.910	–	–
10	3	11.90	0.003	11.90	0.006	11.90	0.009	11.90	0.011	11.90	0.024	11.90	0.033	–	–
10	4	22.50	0.009	22.40	0.017	22.40	0.025	22.40	0.034	22.40	0.079	22.40	0.108	–	–
10	5	32.10	0.019	31.90	0.036	31.90	0.054	31.90	0.072	31.80	0.170	31.80	0.235	–	–
10	6	44.70	0.035	44.40	0.069	44.20	0.101	44.10	0.134	43.90	0.321	44.00	0.451	–	–
10	7	59.20	0.065	58.60	0.127	58.40	0.184	58.00	0.243	57.70	0.588	57.50	0.817	–	–
10	8	73.00	0.097	72.20	0.189	72.10	0.279	71.70	0.367	71.40	0.896	71.20	1.240	–	–
avg		30.349	0.023	30.124	0.045	30.049	0.066	29.959	0.087	29.805	0.210	29.752	0.292		