APPLYING TABU SEARCH FOR MINIMIZING RESHUFFLE OPERATIONS AT CONTAINER YARDS

Kun-Chih WU
PhD Student
Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road, Chung-Li,
Taiwan 32003, R.O.C.
Fax: +866-3-4638907
E-mail: s968907@mail.yzu.edu.tw

Ching-Jung TING
Associate Professor
Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road, Chung-Li,
Taiwan 32003, R.O.C.
Fax: +866-3-4638907
E-mail: ietingcj@saturn.yzu.edu.tw

Rafael HERNÁNDEZ
Graduate Student
Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road, Chung-Li,
Taiwan 32003, R.O.C.
Fax: +866-3-4638907
E-mail: s955449@mail.yzu.edu.tw

Abstract: The container handling operation is a critical issue that affects the throughput of the container yards. This paper focuses on determining the storage position of the reshuffled container during retrieving containers at container yards. The static problem is considered, i.e., the sequence of the retrieved containers is known, and arriving containers is not allowed during the period of retrievals. The objective of the problem is to retrieve all containers with the minimum number of reshuffles. A tabu search (TS) algorithm is proposed to solve the problem and a simple branch and bound (B&B) procedure is applied for comparison of the solution quality. 60 sets of instances are randomly generated for testing the efficiency of the proposed TS. The computational results show that the average gap between TS and B&B is 0.4% from the tested instances and the computational time is effective.

Key Words: *container yard, storage slot allocation, relocation, tabu search*

1. INTRODUCTION

Recently, the increasing demand of global transportation necessitates the concern of productivity of container yards. In addition, the allocation and reshuffles of containers is both time consuming and expensive, which is one of the most critical issues that decrease the productivity of container operations (Yang and Kim, 2006). Due to the limitation of the storage space in the container yard, block stacking is the most common stacking method for efficient usage of storage space (Kim and Hong, 2006). However, containers are stacked in vertical direction as well as the arrival and departure times of containers are uncontrollable. The conditions might cause the earlier retrieved container located underneath the later retrieved container, and that leads to the inevitability of the reshuffles in the container operations.

Avriel *et al*. (1998) considers a storage plan in a container ship. Some heuristics and rules are proposed to find out the minimum shifts to arrange the containers for increasing the port efficiency. Kim and Kim (2002) developed a cost model to determine the optimal amount of storage space and transfer cranes, and a perspective of terminal operators to treat with this subject. Simple solution procedures are suggested to obtain some numerical results. Kim *et al*. (2003) reduced the turnaround time of trucks by optimally sequencing the trucks for transfer operators. A modified dynamic programming and rules based on reinforcement learning were proposed. Zhang *et al*. (2003) studied the storage space allocation problem in a complex terminal yard in which inbound, outbound and transit containers are mixed together.

Some researches had been proposed to consider the issues of the expected number of reshuffles at the container yards. The most difficult issue for evaluating the productivity is to estimate the expected number of the reshuffles at the container yards, when the retrievals of containers are in a random manner. In order to estimate the number of the reshuffles, Watanabe (1991) suggested a simple method, called an accessibility index, for different carrier system. Then, Kim (1997) suggested a formula to obtain a better estimation than the Watanabe's (1991) method. Kim and Kim (1999) proposed a space allocation model to minimize the expected number of reshuffles, in which three cases of different arrival rates (constant, cyclic and dynamic) for import containers were analyzed.

In order to minimize the number of relocations, Kim *et al*. (2000) developed a mathematical model for locating export containers, and solved the problem by a dynamic programming technique. Due to the long computational time of the dynamic programming model, a tree search rules were proposed as well. Murty *et al*. (2005) developed a decision support system (DSS) for optimizing the operations of container terminals in Hong Kong with the rule of the reshuffle index (RI) to locate a container.

Kim and Hong (2006) pointed out that the process of determining location of containers can be divided into three stages, including the space allocation stage, the stage of locating individual containers, and the stage of locating relocated containers during pickup operations. In this paper, we concern about the last stage that is to reduce the number of reshuffles while retrieving containers. In practically, more than one container is needed to be retrieved from the container yards, which consist of the group. Therefore, Kim and Hong (2006) proposed a mathematical model and tackled the problem with group of containers by two methods: a branch and bound and a heuristic based on the estimation of expected number of the reshuffles.

Yang and Kim (2006) also addressed the problem with the group of containers to minimize relocations. They proposed a mathematical model and applied the dynamic programming technique as well as a genetic algorithm. In their paper, two categories for the problem of minimizing relocations have been mentioned, which are called static location problem and dynamic location problem. In the static storage location problem, arrival times and departure times of all storage demand unit (SDU) are known in advance. On the contrast, the arrival and the departure times of an SDU become known only after the arrival of the SDU for the dynamic storage location problem.

Lee and Hsu (2007) considered a yard served by rail mounted gantry (RMG) and applied an integer programming model to minimize the number of the pre-marshalling containers. The objective is to reduce mis-overlay of containers in advance, which is similar to the problem discussed in this paper. The model is embedded within a multi-commodity network flow and

involved some side constraints to deal with physical rules. Lee and Chao (2009) also addressed the same problem by a neighborhood search heuristic. The heuristic consists of two major subroutines and three minor subroutines, and each subroutine is designed for each specific target.

In this paper, we consider a static problem to minimize the number of reshuffle containers. More details about the problem are introduced in section 2. Then a tabu search algorithm is introduced in section 3, as well as a simple branch and bound approach for comparing the solution quality of tabu search. The numerical results are shown in section 4 and the conclusions are provided in the last section.

2. PROBLEM STATEMENT

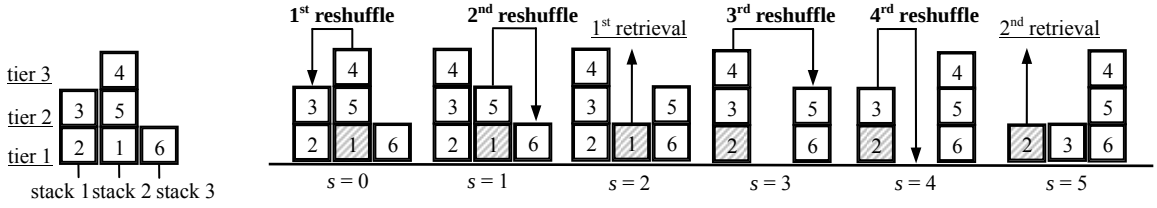
Since the block stacking is an efficient way for usage of storage space, it is the most common stacking method at the container yards. However, it is a common case that the retrieval containers are not on the top of stacks, which implies the reshuffles of containers cannot be avoided. Reshuffles of containers are not value-added activities and very time consuming, so that decreasing the number of reshuffles can reduce the container handling cost. Therefore, the purpose of this paper is to find an efficient algorithm to reduce the number of reshuffles.

A container yard bay, in which a given number of containers are stacked, is considered in this paper. The maximum number of stacks and number of tiers per stack that restrict the storage locations of containers in the yard is given as well. Instead of the dynamic storage location problem, we only consider a static situation which includes two assumptions. First, there are only export containers of same size are stacked within the yard bay and no extra arrival containers during the retrieval period. Second, the initial configuration of the yard bay and the departure times of containers are known, i.e., the pickup sequence of the containers is determined in advance, and containers have to be retrieved in order. In brief, the problem is to determine the storage locations of the reshuffled containers during a pickup sequence that the retrieval containers is given and to minimize the total number of reshuffles.

Figure 1 presents an example of the problem. Six containers are stored in the yard bay, in which the maximum number of stacks and tiers are both three. Figure 1(a) shows the initial configuration of the yard bay, the location of all containers would not be over the restricted number of the stacks and tiers in the initial configuration. The labeled number of the containers indicates the order to retrieve the containers. The problem is to retrieve all containers in order. Figure 1(b) demonstrates the optimal solution that can be examined in terms of a procedure consisting of reshuffles and retrievals. The symbol s represents the stage and the first five of them are shown in this example. Note that, after retrieving container 2 (s = 5), all the remainder containers can be retrieved without any reshuffles. Thus, the total number of reshuffles in this example is four.

The container retrieval problem is generally regarded as a combinatorial problem. Though the complexity of the problem is not be confirmed, Yang and Kim (2006) thought the problem is similar to one-dimensional cutting stock problem which is known as a NP-hard problem. Another reason leads the problem to be difficult is the size of solutions (i.e., the number of reshuffles), which is undetermined and increases as the problem size increases. As we know, this kind of problem cannot be solved in reasonable computational times by exact solution algorithms, such as branch and bound approaches, which only can tackle small size problems.

More appropriate methods for this problem are heuristic and meta-heuristics. Therefore we apply tabu search to solve the problem in this paper.



(a) initial configuration (b) containers retrieved procedure

Figure 1 Illustration of the containers retrieval problem

3. METHODOLOGIES

To our knowledge, no paper has applied tabu search (TS) to the container retrieval problem. Hence, this paper aims to propose a novel TS algorithm to solve this problem. For the comparison of the proposed method, a branch and bound algorithm would be introduced in the following section as well. Furthermore, two heuristics adopted from the literature are also introduced here to generate the initial solutions of the tabu search.

3.1 Tabu Search Algorithm

The basic concept of TS as described by Glover (1989, 1990) is "a meta-heuristic superimposed on another heuristic". The overall approach is to prevent entrainment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited (hence "tabu"). The TS is an efficient algorithm that has been successful applied to a variety of combinatorial optimization problems.

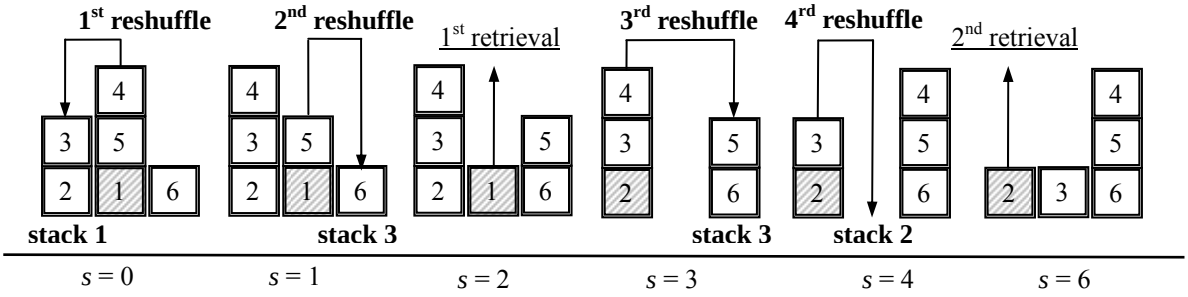
The TS is a local-search based approach, in which the move operator generates the following solution by changing the current solution partially. The moves from the recently visited solutions are memorized in the tabu list which are the solutions forbidden for a number of following iterations. An algorithmic device, called aspiration criteria, allows a move even that is tabu, if it results in a solution which is better than the current best-known solution. The whole procedure would be stopped when the termination criteria is satisfied. Some mechanisms of the tabu search are introduced in the following subsections, and the complete structure of the tabu search will be described at last parts of this section.

3.1.1 Representation and Evaluation of solutions

A solution structure in this problem is critical to the tabu search, because it includes the whole procedure of retrievals and reshuffles and makes itself complicated to encode the solution. The solution structure relating to the evaluations as well as move operators should be designed in an appropriate fashion. A complete solution of this problem might include an entire configuration in each stage, however, too many information would be within it. A more suitable way is only to consider the operations, including retrievals and reshuffles, which can be used to construct the complete procedure. Moreover, the stack from which the container is picked up is not necessary to consider, because the stack that the container is located is determined in each stage.

We propose a one-dimensional vector to express the solution as shown in Figure 2(b), and the

corresponding retrieval procedure is shown as Figure 2(a). Two parts are included in the vector: the positive values and the negative values indicate the reshuffle operations and retrieval operations, respectively. The positive value denotes the stack where the reshuffled containers should be moved to. The negative value is composed of a minus sign and an integer number, in which the minus sign is used to distinguish the retrieve operation from the reshuffle operation, and the integer number stands for the order of the retrieved container. Therefore, the order of the solution represents the sequences of operations. The first two elements "1" and "3" in the vector in Figure 2(b), for example, correspond to the first two reshuffle operations in Figure 2(a). That means container 4 and container 5 are reshuffled to the stack 1 and the stack 3 sequentially. The next element "-1" then implies that container 1 has to be retrieved from the yard.



(a) The entire procedure of solution decoding

1	3	-1	3	2	-2	-3	-4	-5	-6

(b) The one-dimensional vector for representing solution

Figure 2 Representation of the solution

The complete retrieval procedure can be generated by the solution decoding mechanism. In the decoding procedure, once a container which is needed to be reshuffled would be relocated to a given stack according to the solution. For example, container 4 has to be reshuffled in the first stage in figure 2(a), then the first cell of the solution shows the stack that is relocated to is stack 1.

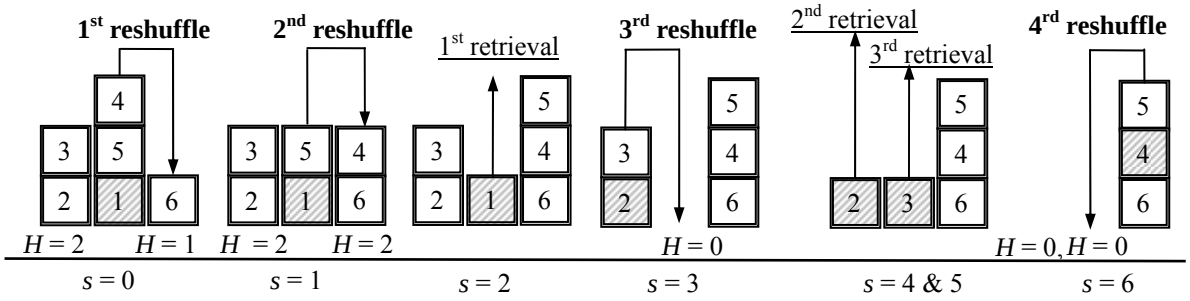
The size of the vector (Z) can be separated into two segments: first one is the fixed number of retrievals (N) that is equal to the number of containers involved; second one is the number of reshuffles (R). And the equation $R = Z - N$ is hold. Hence, the evaluation of the reshuffles can be measured easily by the difference between the size of the vector and the number of containers

3.1.2 Generation of the initial solution

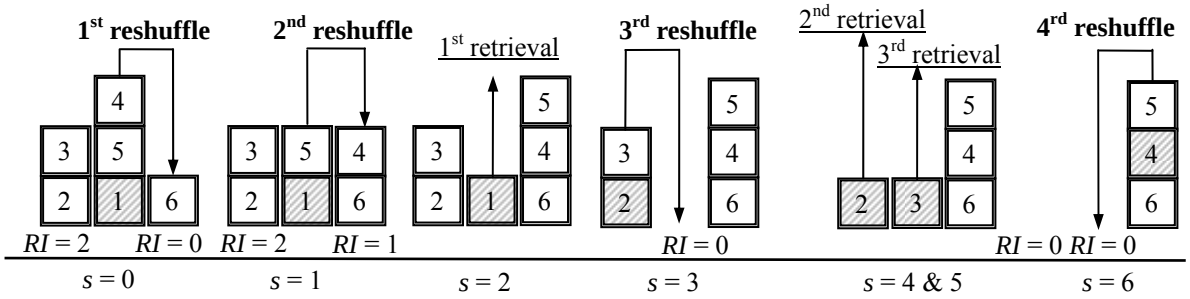
Two simple heuristics had been proposed to allocate reshuffled containers in the literature. Zhang (2000) proposed a heuristic rule, called lowest position (TLP), to deal with the problem. The idea of this method is to decrease the average height of container stacks, since containers with lower average height is more likely to have few reshuffles. According to this rule, when a container is needed to be reshuffled, the stack has the fewest containers would be considered first. In other words, the reshuffled container would be relocated to the stack with the lowest height. A graphical example is demonstrated in Figure 3(a). The symbol H indicates the height of stacks in this example. In the second reshuffle, there is a tie for the

same number of containers in stacks 1 and 2, and then breaking ties arbitrarily.

Another rule which was proposed by Murty *et al*. (2005) is called the rule of reshuffle index (RI). The reshuffle index measures the number of earlier departing containers than the reshuffled container for each of the stacks, then the reshuffled container will be placed on top of the stack with the smallest reshuffle index. If the stack has no earlier departing containers before the reshuffled container, the reshuffle index of the stack is defined as zero. A graphical example is demonstrated in Figure 3(b). The rule for breaking tie is similar to TLP by choosing a stack arbitrarily.



(a) The lowest position (TLP) heuristic



(b) The reshuffle index (RI) heuristic

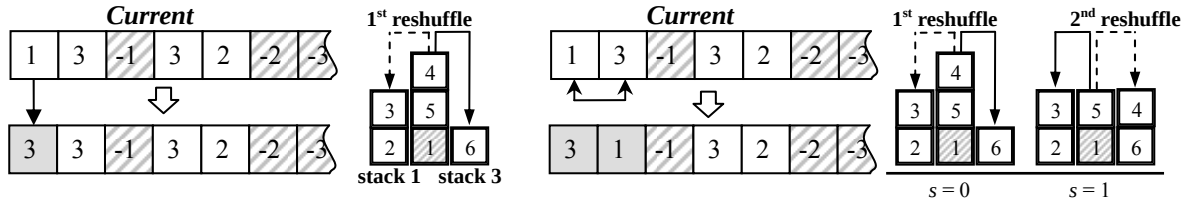
Figure 3 Procedure of TLP heuristic and RI heuristic

3.1.3 Neighborhood structure

A neighborhood solution is generated by applying move operators that transform the current solution into another solution. The move operators generally adjust a small part of the current solution by a defined mechanism. Since the retrieval operations in the solution do not affect the number of reshuffles, only adjustment of reshuffle operations is considered in the move operators. In our TS, two move operators are defined, the *single point move* and the *swap move*. The single point move varies one element from the current stack, where the container is relocated, to another one. For example, in Figure 4(a), the first element in the current vector stands for that the reshuffled container 4 is moved to stack 1 (the dash line). After single point move, the first element is changed to 3, which means container 4 is relocated to stack 3 (the solid line). The swap move only swaps two reshuffle elements between two retrievals, where a swap relating to different retrievals is not allowed, as shown in Figure 4(b). In Figure 4(b), the first two reshuffles move container 4 to stack 1 and container 5 to stack 3, respectively (the dash lines). After the swap move, container 4 is now moved to stack 3 while container 5 is moved to stack 1 (the solid line).

However, two situations are not considered in a move operator: either the stack where the container is picked up or the stack is full. The stack which is full or the container is picked up

from cannot be recognized from the solution, in order to obtain them, the decoding procedure has to be applied. And an efficient manner to generate the neighborhood is to carry out all of them just in one decoding procedure.



(a) The single move operator (b) The swap move operator

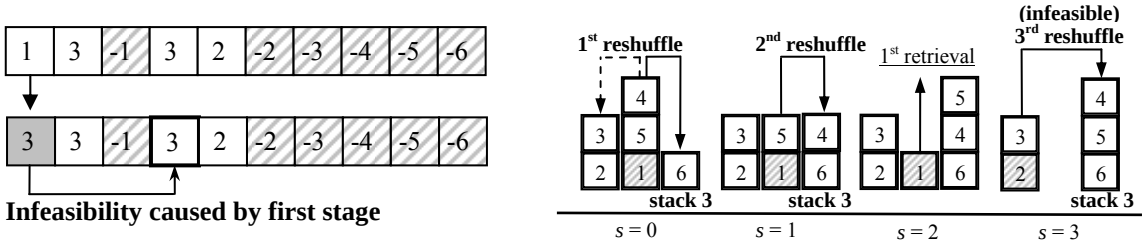
Figure 4 Illustration of move operators

Since varying the earlier stages would affect the following stages, after generating the entire neighborhoods, the following stages have to be verified by the decoding procedure again. Three possible cases might occur in either single point move or swap move:

Infeasibility: If an adjustment of the earlier stage(s) causes a container to be relocated to a full stack in the following stages, then the solution is infeasible and it needs to be removed from the neighborhood structures. An example is shown in Figure 5, the adjustment of the initial stage causes the third stage to be infeasible.

Improving: If an adjustment of the earlier stage(s) causes some following stages be redundant, then the redundant reshuffle stages have to be removed from the solution. In this case, the number of reshuffles is decreased so that the solution is improved.

Deterioration: If an adjustment of the earlier stage(s) causes not enough reshuffle stages to retrieve all the containers, then the extra stages have to be inserted into the solution. In such a case, the number of reshuffles is increased so that the solution is deteriorated. Thus, this is a non-improving move.



(a) Vary the first element in the single point move (b) Infeasibility verification by decoding

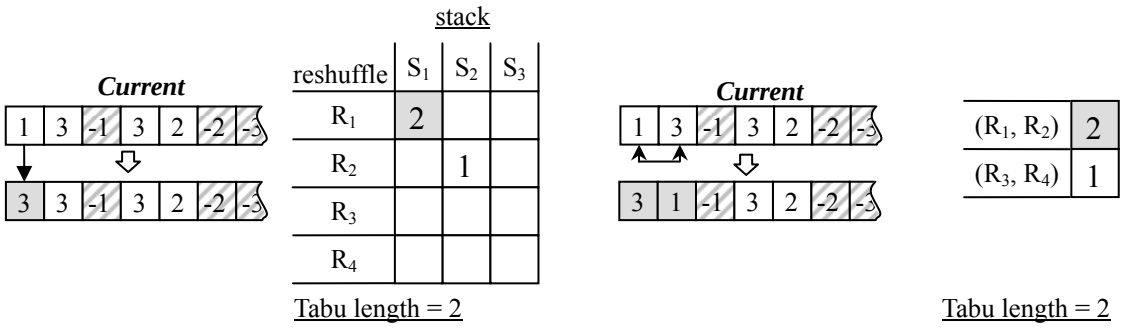
Figure 5 Infeasibility caused by move

3.1.4 Tabu list

Tabu list is to record the move path that has been visited in recent iterations and to avoid obtaining the same solution. This is the main idea of TS to prevent getting trapped in the local optima. In the tabu list, the number of iterations that the move is forbidden is called tabu length. The tabu list structure is related to the neighborhood structures, and is typically constructed as a matrix form. We define two different tabu lists for the single point move and the swap move, respectively.

Figure 6 shows the two different structures of tabu lists for single point move and swap move, respectively. The tabu list of single point move takes the form of a matrix of $R \times C$ dimension

as shown in figure 6(a). The first dimension denotes the stage of the solution, and the second dimension denotes the stack that the container is reshuffled to. The tabu list for the swap move operator takes the form of a matrix of $R \times R$ dimension, and is equal to its neighborhood space as shown in figure 6(b). The two dimensions of the matrix represent the two stages of solutions that were swapped correspondingly. However, only upper half matrix (upper triangle) would be used in the tabu list of swap move, since we only consider short term memory in this paper.



(a) The tabu list for the single point move (b) The tabu list for the swap move

Figure 6 Examples of the tabu list

Because only adjustment of reshuffle operations in the move operator, only R possible moves are considered in the tabu list, and reshuffle number R is not determined so that the sizes of both matrices are dynamic and varied from iteration to iteration.

3.1.5 Aspiration and termination criteria

An aspiration criterion is taken to free the forbidden move. If a neighborhood solution is better than the current best-known solution, even if it is tabu, then the move would be accepted. The termination criterion in our TS is a pre-defined number of iterations, so that the TS would stop when the given number is reached.

3.1.6 Tabu search procedure

The proposed tabu search follows the pseudo-code described as below:

Tabu Search Pseudo-Code

Input *Initial_Configuration*

Output *best* // best solution
best_reshuffle# // best objective value

```
current = Heuristic (Initial_Configuration)
best ← current
best_reshuffle# ← current_reshuffle#
Loop
    For i ∈ the set of neighborhood
        candidates(i) ← the ith Neighborhood(current)
        candidates_reshuffle# (i) ← Evaluation(candidates(i))
    End
    Sort(candidates) By candidates_reshuffle#
    For i ∈ the set of neighborhood
        If candidates(i) is not in tabulist Or Aspiration Rule is satisfied
```

```
                    current ←candidates(i)
                    current_reshuffle# ←candidates_reshuffle# (i)
                    Update(tabulist)
                    Break the loop
                End
            End
        If current_reshuffle# < best_reshuffle#
            best ← current
            best_reshuffle# ←current_reshuffle#
        End
    Until termination condition is satisfied
```

In the beginning of the procedure, the TLP or RI is adopted as the heuristic to generate the initial solution. Then the search procedure repeats for a given number of iterations. In each iteration, an entire neighborhood structure would be generated in which the solution has the minimal reshuffles without violating the tabu restriction would be accepted as the solution for the next iteration. The tabu list is updated when the accepted solution is determined.

3.2 Branch and Bound Approach

A branch and bound (B&B) approach is proposed for confirming the solution quality of TS. According to the branch and bound proposed by Kim and Hong (2006), the implementation of our branch and bound applies depth-first and backtracking strategy as well, due to the limitation of computer memory. The branches are generated along with different options of the stacks. Due to the depth-first strategy the earlier creating node would be visited first. For reducing the searching space, an upper bound is adopted. The depth indicates the number of reshuffles in our branch and bound, therefore if a feasible solution is obtained in the depth d, then no more searching over the depth d is necessary, i.e., a solution with fewer reshuffles only occurs when a feasible solution with the depth $d' \leq d$ is obtained. An example is given in Figure 7. We find a feasible solution with 4 reshuffles at branch T5. This solution will be the upper bound for searching the rest of branches.

4. EXPERIMENTAL RESULTS

Our proposed tabu search and branch and bound approaches are coded in Borland C++ 6.0 and run on a PC with Intel Core 2 Due E8400 3.0 GHz with 2.00 GB of RAM in Windows XP.

4.1 Testing Data

We tested the described algorithms on the heavy density instances that generated randomly by a given number of stacks and a given number of tiers with a given number of containers. The number of containers (N) in each set is determined by two factors: the maximum number of stacks (W), the maximum number of tiers (H). The number of containers involved in the set follows the rule:

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

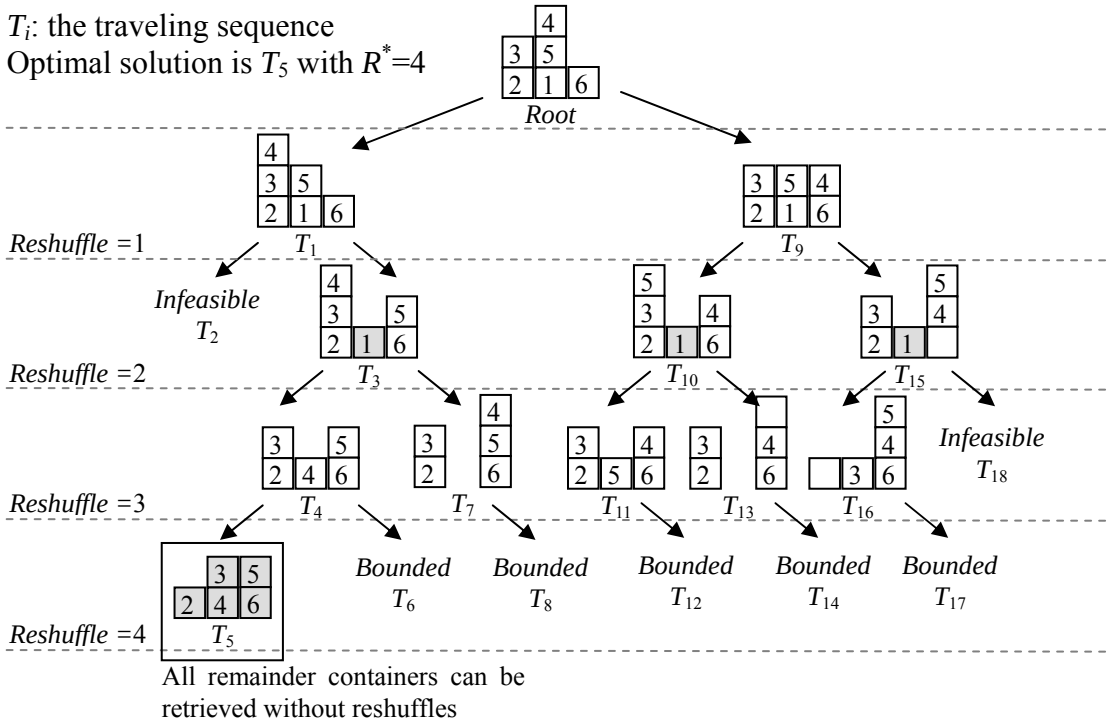


Figure 7 Branch and bound procedure

This rule includes two parts: the first term ($W \times H$) stands for the total number of available slots for the bay and the second term ($H - 1$) indicates reserved slots for reshuffled containers. Because a retrieved container would be underneath other ($H - 1$) containers in the worst case, there must be at least ($H - 1$) slots reserved for reshuffling these containers. This implies the maximum loading containers in the bay except the reserved slots for reshuffles.

Because W and H are rarely smaller than 3 in practice and it is possible that a bay in a container yard has maximum 12 stacks with 8 tiers, we generate the 60 problem sets by setting the number of stacks (W) from 3 to 12 and the number of tiers (H) from 3 to 8. The number of containers in each set follows the rule (1), and then constructing the initial configuration by locating containers into storage position randomly. We generate 10 instances in each problem set, and totally 600 instances are generated and tested in this study.

4.2 Parameter Analysis

Some parameters and mechanisms are crucial for the results of TS, including the heuristics for initial solutions, the move operators for neighborhood solutions, the tabu length and the termination criterion. According to our preliminary experiments, we give an initial setting of parameters, for which RI is adopted as the initial solution heuristic, tabu length is equal to 11 and the number of iterations is equal to 600.

It is reasonable that the combination of the single move and swap move have at least the same performance to individual one regardless of other parameters. Because it is known that the RI heuristic has better performance than the TLP heuristic, it is also expected that the TS using RI as the initial solution heuristic has better performance than the one using TLP. Therefore, we examined the mechanism of move operators and the initial heuristics first, and the tabu length and the number of iterations are tested later.

We applied two different move operators in this paper: the single point move and the swap move. The quality of TS with these move operators are presented in Table 1. The first two columns indicates single point move and swap move work alone, and the last column stands for the combination of these two move operators. The numbers in the brackets is the number of problem sets in which the specific move operator outperforms the others. For example, in Table 1, the results in 13 problem sets solved by the single point move outperforms the results obtained by the swap move and the combination of the two move operators As expected, the results show that the combination of single point move and swap move has better performance than using the other two move operators individually. Besides, the swap move did not perform well by itself. However, the swap move only needs small computational time, the combination of these two move operators is the best choice due to acceptable elapsed time.

Table 1 Comparisons of move operators

	move operator		
	single point	swap	single point & swap
Average # of reshuffles	31.778 (13[a])	35.958 (0[a])	**31.447 (31[a])**
Average time (sec)	0.888	0.127	1.022

[a]: the number of sets has smaller number of reshuffles than the other move operators.

Two heuristics, lowest position rule (TLP) and reshuffle index (RI), for generating the initial solution are tested. The experimental results are demonstrated in Table 2. The value shown in the table is the average number of reshuffles over 600 instances. Again, the numbers within the bracket in TLP and RI indicates the number of problem sets that has smaller number of reshuffles than the other heuristics. The last two columns are two TS that applied TLP and RI as their initial solution generation, respectively. Similarly, the numbers in both TS are also obtained by the comparison between each other. The results show that the solution quality of RI is always equal to or better than that obtained by TLP. In addition, either the results of the heuristic or the results obtained by TS, RI outperforms TLP in both number of sets and average number of reshuffles. It is rational that RI is better than TLP, because that RI considers about the information of the sequence but TLP does not.

Table 2 Comparison between TLP and RI

	TLP	RI	(TLP[a])TS	(RI)TS
Average # of reshuffles	44.840 (0[b])	**38.603 (59[b])**	31.848 (17[c])	**31.447 (26[c])**
Average time (sec)	0.001	0.001	1.057	1.022

[a]: the heuristic for generating initial solution.
[b]: the number of sets has smaller number of reshuffles than the other heuristic.
[c]: the number of sets has smaller number of reshuffles than the other TS.

We adopt RI as the initial solution heuristic and the combination of the two move operators to examine the influence of the tabu length and total number of iterations at the same time. Five different tabu lengths are tested: 7, 9, 11, 13 and 15, as well as three different numbers of iterations: 300, 600 and 900. As we can see, Table 3 illustrates that a better performance can be obtained when tabu length is equal to 9 and the number of iterations is equal to 900.

Table 3 Comparisons of different setting of tabu length and the number of iterations

Iter.	Tabu length									
	7		**9**		**11**		**13**		**15**	
	Res.	Time (sec)	Res.	Time (sec)	Res.	Time (sec)	Res.	Time (sec)	Res.	Time (sec)
300	31.440	1.019	31.400	1.015	31.447	1.020	31.410	1.021	31.415	1.023
600	31.247	2.000	31.152	1.987	31.223	1.998	31.190	1.999	31.192	2.004
900	31.137	2.968	**31.038**	2.950	31.123	2.964	31.075	2.960	31.132	2.970

4.3 Results

In order to examine the quality of the solution, the branch and bound approach is applied to compare with the TS. However, due to long computational time, only 18 problem sets that satisfy the limitation of one hour are tested. According to pervious testing, we choose the best set of parameters and move operators for comparison, which include: tabu length is equal to 9, total iterations is 900, RI heuristic is applied to generate the initial solution and the combination of two move operators is adopted.

The final results for those problem sets show that B&B can find the optimal solutions within the computational time limit are shown in Table 4. The gap is measured by the difference of the number of reshuffles (R) between TS and B&B divided by the optima. The gap increases when the problem size increases. The gaps by our TS of all compared problem sets are all less than 2.68% with average gap of 0.4%. However, the branch and bound approach only can solve a few instances in reasonable computational times. The problem sets of over 10 stacks can not be solved in the average elapsed time of one hour. Thus, the computational time and solution quality of TS is more acceptable than the branch and bound approach. The more detail computational results of all 60 problem sets are shown in Table 5 for comparing the TS with the TLP heuristic and RI heuristic. Due to that the computational times of both simple heuristics are quite small, the computation times are not reported in Table 5.

5. CONCLUSION

This research focuses on determining the storage position of the reshuffled container during static retrieving outbound containers, where the initial configuration of the yard and the retrieval order of each individual container are given. A tabu search approach is proposed to deal with this problem and a depth-first branch and bound algorithm is applied for comparison. We test different tabu lengths and some mechanisms, such as the move operators and the generation heuristics of initial solutions, of tabu search algorithm.

According to the experimental results, we find that the heuristic based on the reshuffle index outperforms the lowest position heuristic and the combination of single point move and swap move is adequate for the tabu search. The comparison of the tabu search and the branch and bound shows that the average gap is 0.40% in the tested instances. However, the computational time of branch and bound is over the time limit of one hour when the number of stacks of the sets is over 10. Therefore the results obtained by our TS are acceptable for larger problem sets.

Table 4 Comparisons of TS and branch and bound

set		B&B		TS		Gap (%)
W	*H*	*R*	Time (s)	*R*	Time (s)	
3	3	3.5	0.000	3.5	0.014	0.00%
3	4	5	0.000	5	0.036	0.00%
3	5	8.5	0.002	8.5	0.063	0.00%
3	6	11	0.008	11	0.100	0.00%
3	7	14.9	0.309	15.3	0.183	2.68%
3	8	20.6	1.300	20.9	0.286	1.46%
4	3	3.5	0.000	3.5	0.033	0.00%
4	4	8.2	0.003	8.2	0.089	0.00%
4	5	11.7	0.141	11.7	0.164	0.00%
4	6	17	30.900	17.1	0.311	0.59%
5	3	5.6	0.002	5.6	0.078	0.00%
5	4	10.3	0.075	10.3	0.175	0.00%
5	5	17	275.000	17.2	0.400	1.18%
6	3	7.7	0.019	7.8	0.148	1.30%
6	4	12	7.100	12	0.295	0.00%
7	3	10.1	1.070	10.1	0.258	0.00%
8	3	9.3	10.200	9.3	0.303	0.00%
9	3	12	311.000	12	0.520	0.00%
average		10.439	35.396	10.500	0.192	0.40%

Our experiments are tested on an ideal situation without any external variable such a crane operator's skills. Future research about this topic should be compared with real data from container yards, as well as the dynamic case can be studied based on knowledge obtained from the static cases.

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, Vol. 76, No. 1**, 55-71.

Glover, F. (1989) Tabu search, Part I, **ORSA Journal on Computing, Vol. 1**, **No. 3**, 190-206.

Glover, F. (1990) Tabu search, Part II, **ORSA Journal on Computing, Vol. 2**, **No. 1**, 4-32.

Kim, K. H. (1997) Evaluation of the number of rehandles in container yards, **Computers & Industrial Engineering, Vol. 32, No. 4**, 701-711.

Kim, K. H. and Hong, G. P. (2006) A heuristic rule for relocating blocks, **Computers & Operations Research, Vol. 33, No. 4**, 940-954.

Kim, K. H. and Kim, H. B. (1998) The optimal determination of the space requirement and the number of transfer cranes for import containers, **Computers & Industrial Engineering, Vol. 35, No. 3(3-4)**, 427-430.

Table 5 Complete computational results of TLP, RI and TS

class		TLP	RI	TS		class		TLP	RI	TS	
W	*H*			*R*	time(s)	*W*	*H*			*R*	time(s)
3	3	3.6	3.6	3.5	0.014	8	3	10.5	9.8	9.3	0.303
3	4	6.0	5.2	5.0	0.036	8	4	21.8	19.3	17.3	0.747
3	5	10.6	9.9	8.5	0.063	8	5	35.0	29.9	26.1	1.400
3	6	14.3	12.9	11.0	0.100	8	6	46.4	43.1	34.3	2.300
3	7	20.5	18.2	15.3	0.183	8	7	71.9	60.9	48.4	4.070
3	8	26.5	25.7	20.9	0.286	8	8	101.0	86.8	67.1	6.340
4	3	3.8	3.6	3.5	0.033	9	3	13.3	12.6	12.0	0.520
4	4	9.5	8.9	8.2	0.089	9	4	24.0	22.0	19.3	1.050
4	5	15.4	13.8	11.7	0.164	9	5	40.5	34.1	28.0	1.950
4	6	22.7	21.8	17.1	0.311	9	6	58.9	49.7	40.2	3.440
4	7	31.3	28.6	24.0	0.492	9	7	80.3	67.6	52.3	5.460
4	8	44.1	40.4	31.2	0.819	9	8	103.0	86.0	65.5	7.850
5	3	6.3	6.2	5.6	0.078	10	3	14.0	13.8	13.1	0.702
5	4	12.5	11.5	10.3	0.175	10	4	27.0	25.5	22.0	1.440
5	5	22.3	19.6	17.2	0.400	10	5	45.9	38.4	32.5	2.670
5	6	29.5	26.7	22.7	0.608	10	6	64.8	55.0	43.9	4.450
5	7	40.4	36.7	28.5	0.944	10	7	96.9	78.6	62.2	7.800
5	8	59.3	52.6	40.0	1.580	10	8	122.0	98.1	75.5	11.300
6	3	8.5	8.0	7.8	0.148	11	3	15.6	14.4	13.9	0.884
6	4	13.5	12.8	12	0.295	11	4	32.7	28.8	24.9	2.050
6	5	26.6	23.6	19.6	0.631	11	5	51.7	43.1	35.7	3.730
6	6	36.8	35.4	28.1	1.050	11	6	72.8	58.7	48.8	6.120
6	7	48.5	42.8	34.0	1.580	11	7	107.0	90.0	68.7	10.200
6	8	64.9	58.1	44.8	2.440	11	8	134.0	112.0	85.1	15.800
7	3	11.2	10.6	10.1	0.258	12	3	16.7	15.0	14.5	1.090
7	4	18.8	16.9	14.8	0.526	12	4	35.0	30.9	27.1	2.710
7	5	30.7	25.6	21.6	0.917	12	5	55.6	47.6	38.4	4.700
7	6	44.3	38.1	30.0	1.600	12	6	81.9	67.2	54.2	8.060
7	7	68.3	57.8	45.5	2.720	12	7	117.0	93.9	73.9	13.900
7	8	84.5	76.8	57.7	4.300	12	8	158.0	131.0	97.6	21.100

Kim, K. H. and Kim, H. B. (1999) Segregating space allocation models for container inventories in port container terminals, **International Journal of Production Economics, Vol. 59, No. 1-3**, 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, Vol. 36, No. 9**, 821-835

Kim, K. H., Lee, K. M. and Hwang, H. (2003) Sequencing delivery and receiving operations for yard cranes in port container terminals, **International Journal of Production Economics, Vol. 84, No. 3**, 283-292.

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, Vol. 124, No. 1**, 89-101

Lee, Y. and Hsu, N-Y. (2007) An optimization model for the container pre-marshalling problem, **Computers & Operations Research, Vol. 34, No. 11**, 3295-3313.

Lee, Y. and Chao, S-L. (2009) A neighborhood search heuristic for pre-marshalling export containers, **European Journal of Operational Research, Vol. 196, No. 2**, 468-475.

Murty, K. G., Wan, Y., Liu, J., Tseng, M. M., Leung, E., Lai, K-K. and Chiu, W.C. (2005) Hongkong international terminals gains elastic capacity using a data-intensive decision-support system, **Interfaces, Vol. 35**, **No. 1**, 61-75.

Watanabe, I. (1991) Characteristics and analysis method of efficiencies of container terminal – An approach to the optimal loading/unloading method, **Container Age**, **Vol. 27**, 36-47.

Yang, J. H. and Kim, K. H. (2006) A grouped storage method for minimizing relocations in block stacking systems, **Journal of Intelligent Manufacturing**, **Vol. 17**, **No. 4**, 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.

Zhang, C., Liu, J., Wan, Y., Murty, K. G. and Linn, R. J. (2003) Storage space allocation in container terminals, **Transportation Research Part B, Vol. 37, No. 10**, 883-903.