Institute of Transportation Studies
University of California at Berkeley

Performance Comparison of Crane Double Cycling Strategies

Anne V. Goodchild and Carlos Daganzo

WORKING PAPER
UCB-ITS-WP-2005-2

December 2005
ISSN 0192 4141

PERFORMANCE COMPARISON OF CRANE DOUBLE CYCLING STRATEGIES

ANNE V. GOODCHILD

DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING

UNIVERSITY OF WASHINGTON, SEATTLE, WA 98195, USA

ANNEGOOD@U.WASHINGTON.EDU

CARLOS F. DAGANZO

DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING

UNIVERSITY OF CALIFORNIA, BERKELEY, CA 94720, USA

ABSTRACT. This report compares the performance of three double-cycling algorithms used to determine the sequence with which to load and unload containers from a vessel with a quay crane. Double cycling is a technique which can improve the efficiency of a quay crane and container port by unloading and loading containers in the same crane cycle. The three algorithms, the greedy strategy, the proximal strategy, and Johnson's rule, are introduced and results from applying the three strategies to a set of simulated vessels are compared. While Johnson's rule provides the minimum number of cycles required to unload and load a vessel using double cycling, it is operationally less convenient. The proximal strategy is operationally convenient, but provides a smaller benefit when compared to single cycling. All strategies provide significant benefit (about a 40% reduction over single cycling for hatchless ships), and the results are not particularly sensitive to the algorithm used.

INTRODUCTION

Double-cycling is a technique that can be used to improve the efficiency of quay cranes by eliminating some empty crane moves. Instead of using the current method, where often all relevant containers are unloaded from the vessel before any are loaded (single-cycling),

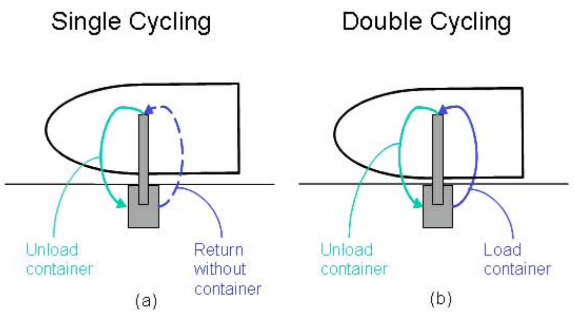

FIGURE 1. (a) Unloading using single-cycling (b) Unloading and loading with double-cycling.

containers are loaded and unloaded simultaneously (see Fig. 1). This allows the crane to carry a container while moving from the apron to the ship (one move) as well as from the ship to the apron; doubling the number of containers transported in a cycle (or two moves). This crane efficiency improvement can be used to reduce ship turn-around time and therefore improve port throughput. For more background on the motivation behind double cycling, and other strategies ports have undertaken to improve port throughput, see [1].

For the purposes of this report, we will assume the ship's loading plans are given, and that they are the same with and without double-cycling. We assume the metric for comparing strategies is the number of cycles required to complete loading and unloading operations for one row of the vessel. When considering the benefits of double-cycling, we assume that existing planning tools had been used to create a loading plan, as is current practice, and that this loading plan has made no accommodations for double-cycling. We therefore consider changes only to the crane's sequence of operations. The ideas presented here are not meant to substitute for detailed terminal and vessel planning programs, but to provide portable insights into double-cycling at a more general level. For a more thorough analysis of double cycling see [1].

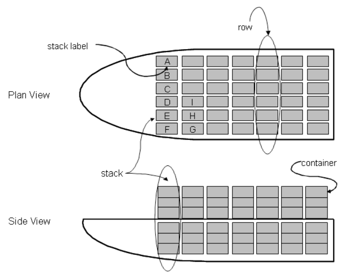

FIGURE 2. Plan and side views of a simplified ship (number of containers shown not representative of typical ship size).

In the next section a framework is set for analysis of the problem. The section also includes an example that is used to illustrate the problem's basic properties. In Section 2 we discuss a scheduling formulation for the problem. Section 3 presents the greedy algorithm. Section 4 introduces the proximal stack strategy. In Section 5 we provide some detail regarding the computer program that was used to generate vessel data. In Section 6 results are presented from the computer program which allows for comparison of the three strategies. The report concludes with Section 7.

1. MODELING FRAMEWORK

The layout of containers on a ship can be modelled as a 3-dimensional matrix. Containers are stacked on top of one another, and arranged in rows (see Fig. 2). One row stretches across the width of the ship. Large container vessels today typically hold 20 stacks of containers across the width of the ship, and up to 20 stacks along the length of the ship (40-foot equivalent units). Of course, we expect these figures to increase with the market penetration of Malacca-max carriers. Figure 2 gives a top and side view of a typical vessel (although the number of container stacks is not representative).

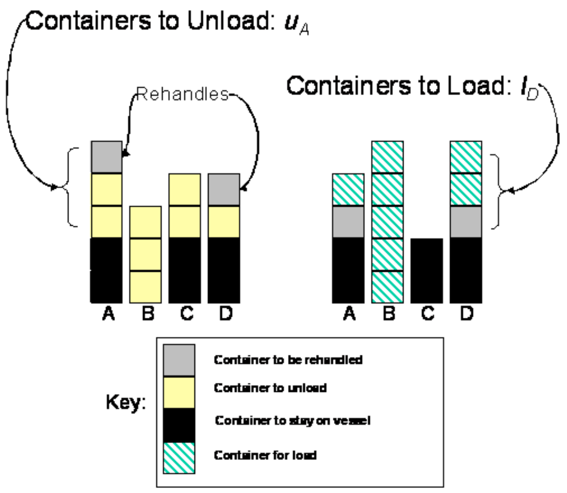
Containers to Unload: u_A
Rehandles
Containers to Load: l_D
A B C D
A B C D
Key:
Container to be rehandled
Container to unload
Container to stay on vessel
Container for load

FIGURE 3. Detailed plan for containers to be unloaded and loaded.

We assume that ships lack hatch-coverings, or doors on the deck that separate above-deck and below-deck storage. For an analysis of vessels with deck hatches see [1].

Consider the case where a ship arrives in port with a set of containers on board to be unloaded and a loading plan for containers to be loaded. The loading plan indicates the placement of containers on the ship. Given are u_c and l_c, the number of containers to be unloaded and loaded, respectively, in each stack labelled c. Figure 3 is an example problem that will be used for illustrative purposes. Notice that in Fig. 3, $u_A = 3$ and $l_A = 2$. A rehandle is a container that must be moved to access containers below it, but will then be stowed again before the ship departs. Note that if any rehandles are necessary, we include these in the total number of loads and unloads.

The number of cycles necessary to complete loading and unloading will be represented by the variable w. We will consider double-cycling within one row of the ship. Due to the difficulty with which the crane moves laterally along the ship, it is not practical to consider double-cycling across two rows. We will complete unloading and loading of one row before moving the crane lengthwise along the ship to the next row. A key feature of double-cycling is the order in which the stacks within each row are handles, and this is explained below.

Let S denote the set of stack labels in a row, $|S| = N$ the number of stacks in the set, and Π a permutation of S indicating an ordering of the stacks. A permutation is a one-to-one correspondence between the set of $n \in \{1, \ldots, N\}$ and $c \in S$ such that $\Pi(n) = c$, or $n = \Pi^{-1}(c)$. For example, in Fig. 3, the set of stack labels is $S = \{A, B, C, D\}$. A permutation of these is $\{B, A, C, D\}$ given by the function Π_e where $\Pi_e(1) = B$, $\Pi_e(2) = A$, $\Pi_e(3) = C$, and $\Pi_e(4) = D$. We will restrict our attention to special cases of the generic double-cycling method described below:

- Choose an unloading permutation, Π'. Unload all containers in the first stack of the permutation, then all containers in the second stack of the permutation, proceed in this fashion until all stacks have been unloaded.
- Choose a loading permutation, Π, and load the stacks in that order. Load all containers in the first stack, then in the second, etc. Loading can start in any stack as soon as it is empty or it contains just containers that should not be unloaded at this port. Once loading has begun in a stack, continue loading until that stack is complete.

In all cases single- or double-cycling, we assume that the crane starts and finishes on the dock.

Figure 4(a) is a diagram for a single-cycling operation where the stacks of Fig. 3 are handled in the order $\{A, B, C, D\}$ both for loading and unloading. Time is expressed in cycles. Note that loading operations must wait until cycle $w = 10$, when unloading is

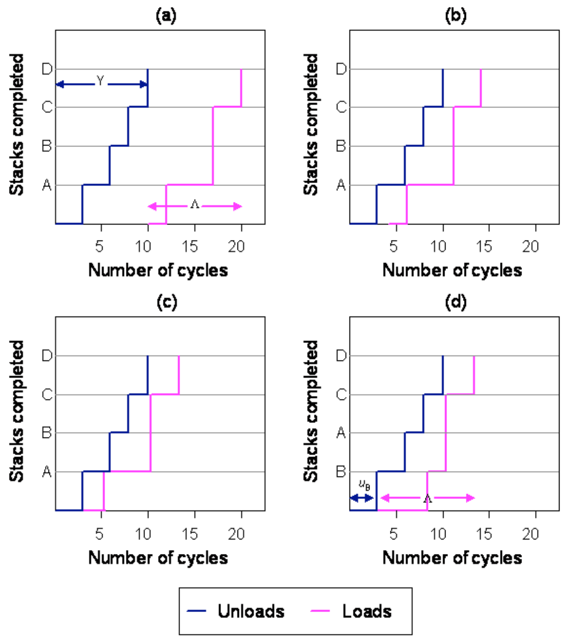

FIGURE 4. Turn-around time with different methods: (a) single-cycling with ordering A,B,C,D unloading starts at w = 10, 20 cycles (b) double-cycling with ordering A,B,C,D unloading starts at w = 4, 14 cycles. (c) double-cycling with ordering A,B,C,D unloading starts at w = 3, 14 cycles. (d) double-cycling ordering B,A,C,D unloading starts at w = 3, 13 cycles.

finished. The process requires $w = 20$ cycles. With single cycling, we assume the crane unloads each row of the vessel before loading any containers.

If we double-cycle, we can still plot the unloading curve on the same diagram. Now, using the same sequence for unloading and loading, $\Pi' = \Pi = A, B, C, D$, we can shift the loading curve to the left as far as possible without overlapping the unloading curve. Figure 4(b) shows the maximum shift. Loading can start as early as w = 4 and the process would require only 14 cycles. The same number of cycles is obviously obtained if we start loading each stack as early as possible, as in Fig. 4(c). This introduces some delay as the loading operations must wait one cycle for the unloading operations to be completed in stack B, but does not change the completion time.

With single-cycling one cycle is required for every container. With double-cycling, however, the number of cycles will depend on the sequence. Fig. 4(d) shows that if the loading and unloading sequence is B, A, C, D, then the completion time is $w = 13$.

This framework considers the work of one crane, working on individual rows of a vessel. A scheduling formulation for this problem is discussed in the next section.

2. Scheduling Approach

Although double-cycling involves only one physical machine (the quay crane), the problem can be formulated as a two-machine flow shop scheduling problem where one job corresponds to one stack and each job has two operations: an unloading operation that must be completed first, and a subsequent loading operation. The crane performs as the loading machine when its trolley is moving from the apron to the vessel and as the unloading machine when its trolley is moving from the vessel to the apron. We can assume that there is a separate machine for loading and another for unloading because in every cycle the crane can perform exactly one task of each type, and as a result the number of crane cycles available to do a task of either type is unaffected by what the crane does with tasks of the other type. Thus, the crane can operate on both tasks in the same exact way as two independent machines that could handle one container per cycle.

Our problem is to determine the best unloading and loading sequences to minimize the maximum completion time (makespan). The formulation (shown in the Appendix) includes a technological constraint: stacks must be unloaded before they are loaded, but no precedence constraints. We assume all rehandles (containers that must be moved to access another container, but are to stay on the vessel) are loaded back into the stack from which they are unloaded, and that there are no constraints on the order in which a set of stacks is operated on by an individual machine (loading or unloading).

This problem can be solved optimally with Johnson's rule [2]. It has three key features. First, the assumption of uninterrupted loading and unloading of stacks is not restrictive; preemption cannot improve the solution. Second, it is sufficient to consider schedules in which the processing orders on the two machines are identical. Third, if the processing times are interchanged then an equivalent inverse problem results.

Although Johnson's rule can be used to determine the optimal sequence for a specific vessel it does not yield a simple formula for the number of cycles that could be used for port planning. In the next section we introduce an algorithm that is more amenable to developing port planning formulae.

3. A Greedy Strategy

We propose to unload and load each stack as soon as possible, assuming that the loading and unloading sequences are given by the same "greedy" permutation, $\Pi' = \Pi = G$. The greedy permutation is obtained by ordering the stacks in descending order of the variable d_c where:

$$d_c = l_c - u_c \quad when \quad \Lambda \geq Y \tag{1}$$

$$d_c = u_c - l_c \quad when \quad Y > \Lambda \tag{2}$$

The rationale for equations 1 and 2 is that we want the unloading operations to run ahead of the loading operations as much as possible.

This strategy should provide a smaller benefit than the optimal strategy, but it is useful because the sequence can be easily calculated and the results can be tightly bounded from above and below by a simple formula. See [1] for more detail.

4. The Proximal Stack Strategy

The proximal stack strategy is based on the strategy typically used in current operations. Let R be the number of rows on a vessel, and let C_i be the number of stacks in row i. Rows are numbered $i = 1..R$ starting from the bow. Let $\{1...C_i\}$ be the set of stack numbers, c, in one row of a vessel. The first number, $c = 1$ is the stack closest to the shore, and C_i the stack nearest the water.

Definition 1 (Proximal Stack Strategy)**.** The crane processes rows one at a time in order of increasing i. For each row it:

(1) Unloads all containers in the stack closest to the shore, $c = 1$, then all containers in stacks $c = 2, 3$, etc. until all stacks in the row have been unloaded.
(2) Loads the stacks using the same ordering. Loading can start in a stack as soon as it is empty or contains just containers that should not be unloaded at this port. Once loading has begun in a stack it is continuously loaded until complete.

We expect the benefit with this strategy to be smaller than that with the optimal or greedy strategy. We choose to focus on this strategy because it is operationally and mathematically convenient, and the strategy used in practice. In the next section we evaluate the reduction in the number of cycles (over single cycling) achieved by using double-cycling. We compare the three strategies with results from a set of simulated vessels. We also present details of the simulation program used to generate the vessel data.

5. Computer Program

A computer program was developed that generates ship data, and calculates the number of moves to turn around the ship using the three strategies. The purpose of the program was to provide large sets of data upon which to evaluate algorithms. The program provides turn around times (in number of moves) using single cycling, the greedy algorithm, Johnson's rule, and the proximal stack strategy. The program can generate data for many different ship designs and market conditions (imports and exports, origins and destinations, and number of stacks). Table 1 describes how different parameter values can be used to simulate different market environments.

Necessary inputs into the simulator are:

- number of stacks
- h_i, maximum stack height (in containers) for imports
- h_e, maximum stack height (in containers) for exports
- p_i and q_i, parameters for the beta distribution, applied to imports
- p_e and q_e parameters for the beta distribution, applied to exports

The number of containers to unload and load in each stack, u_c and l_c, are assumed to be independent and generated by sampling from a beta distribution. According to the input parameters p_i, q_i, p_e, and q_e, the beta distribution is sampled twice for each stack, obtaining a value between 0 and 1, with a different distribution for imports and exports. Each sampled value is then multiplied by the maximum stack height, then rounded down to the nearest integer. Notice some stacks will have 0 containers.

Description	Parameter Values
Shallow (River Boat)	Maximum stack height ≤ 10
Deep (Ocean Vessel)	Maximum stack height > 10
Small Vessel	Number of stacks in row ≤ 10
Medium Vessel	Number of stacks in row $> 10, \leq 15$
Large Vessel	Number of stacks in row > 15
Balanced Load (similar demand for imports and exports)	maximum stack height, p and q same for imports and exports
Unbalanced Load (different demand for imports and exports)	maximum stack height, p and q different for imports and exports
Varying Demand and Loading	Vary parameters p and q

TABLE 1. How computer program parameters can be varied to reflect different market conditions.

6. RESULTS COMPARISON

To understand the benefits on a larger scale a computer program was used to generate problem instances and calculate the number of moves for each algorithm. The number of containers to unload and load in each stack, u_c and l_c, were determined with independent draws of beta random variables with parameters: (p_i, q_i) for imports and (p_e, q_e) for exports. Since beta random variables have a range between 0 and 1, each sampled value was multiplied by the maximum stack height and then rounded down to the nearest integer. These values were in general different for imports and exports. Some stacks could have 0 containers. Parameter settings used to generate the figures are shown in Table 2. The results are shown in Figure 5. Each data point represents the average value of 40 generated vessels.

Parameter	Setting
Number of stacks	Varies (as indicated in the figure)
Beta Distribution Parameters	5 runs with $p_i = 1, q_i = 1, p_e = 1, q_e = 2$ 5 runs with $p_i = 1, q_i = 1, p_e = 2, q_e = 1$ 10 runs with $p_i = 1, q_i = 1, p_e = 2, q_e = 2$ 5 runs with $p_i = 2, q_i = 2, p_e = 1, q_e = 1$ 10 runs with $p_i = 2, q_i = 2, p_e = 2, q_e = 2$ 5 runs with $p_i = 2, q_i = 2, p_e = 2, q_e = 1$
Maximum number of imports in one stack	20
Maximum number of exports in one stack	20

TABLE 2. Parameter settings used to generate vessel data.

As expected, the benefits using the greedy strategy are greater than the benefits using the proximal stack strategy, but smaller than the benefits using the optimal strategy, but the differences are small. For a row of 20 stacks, there is a 45% reduction in the number of moves over single cycling for the optimal strategy, a 44% reduction using the greedy strategy, and about a a 40% reduction using the proximal stack strategy. Notice that the savings range in the figures has been reduced to allow closer comparison of the values, and that benefits above 35% are commonplace.

7. CONCLUSIONS

The proximal stack strategy is presented more completely in [1], where a formula is developed for longer term port planning. The greedy strategy allows us to develop some simple formula for the number of moves required to unload and load a specific vessel. That work is presented in [1], along with the optimal strategy, which provides the smallest

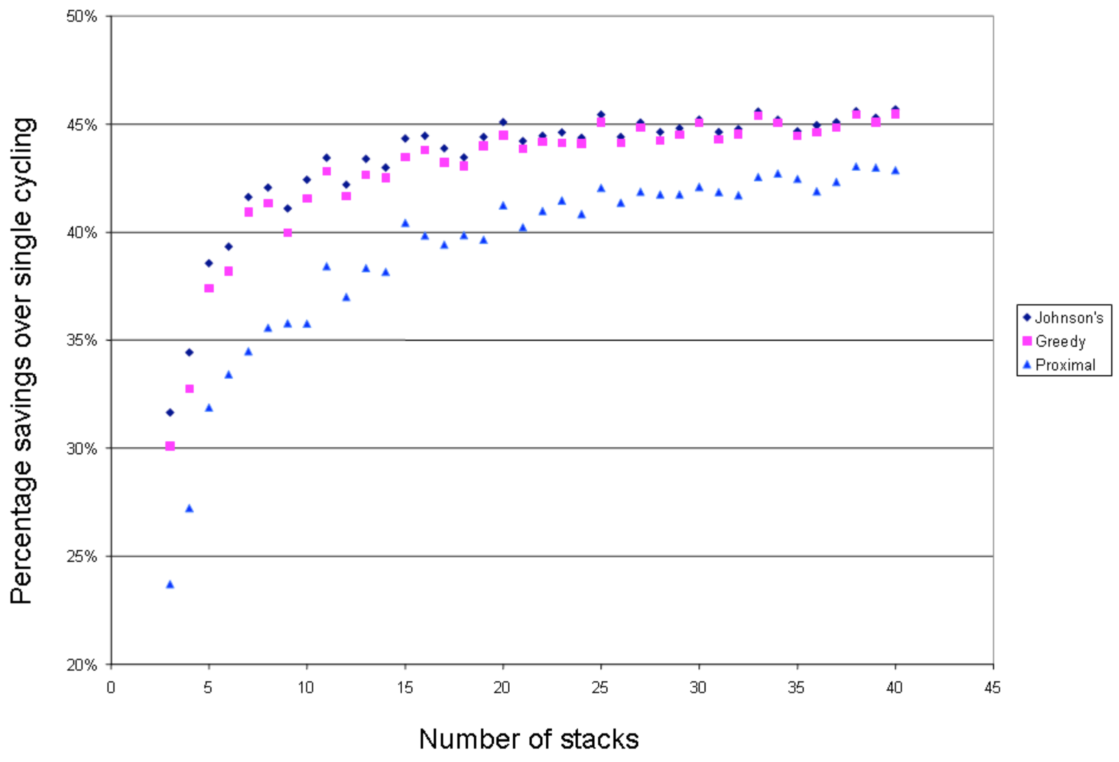

FIGURE 5. Performance comparison of greedy strategy and Johnson's rule to single cycling for vessels without deck hatches. Each data point shows the percentage savings over single cycling and is the average result for 40 generated vessels.

number of moves required to turn around a vessel. This report defines and compares the three algorithms. Outside of these mathematical manipulations, all strategies provide an opportunity for significant improvement over the status quo.

REFERENCES

[1] A. V. Goodchild. *Crane Double Cycling in Container Ports: Algorithms, Evaluation, and Planning*. PhD thesis, University of California at Berkeley, Department of Civil and Environmental Engineering, 2005.

[2] S.M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1:61–68, 1954.

8. Appendix

The double-cycling problem can be formulated as the 2-machine flow shop problem below. We use the following notation:

- u_c - number of containers to unload in stack $c \in S$
- l_c - number of containers to load in stack $c \in S$
- FU_c - completion time of unloading $c \in S$
- FL_c - completion time of loading $c \in S$
- w - maximum completion time
- X_{kj} - binary variable to for ordering of unloading jobs (1 if $j \in S$ is unloaded after $k \in S$ and 0 otherwise)
- Y_{kj} - binary variable to for ordering of loading jobs (1 if $j \in S$ is loaded after $k \in S$ and 0 otherwise)
- M - a large number

The scheduling problem (SP) is to minimize the maximum completion time of all jobs subject to constraints. The result is to uniquely identify the permutations Π and Π', and a feasible set of job start and end times. It is assumed that the process starts at time zero. The formulation is:

(3a) $$(SP) \qquad minimize \quad w$$

(3b) $$subject \quad to \quad w \geq FL_c \quad \forall c \in S,$$

(3c) $$FL_c - FU_c \geq l_c \quad \forall c \in S,$$

(3d) $$FU_k - FU_j + MX_{kj} \geq u_k \quad \forall j, k \in S,$$

(3e) $$FU_j - FU_k + M(1 - X_{kj}) \geq u_j \quad \forall j, k \in S,$$

(3f) $$FL_k - FL_j + MY_{kj} \geq l_k \quad \forall j, k \in S,$$

(3g) $$FL_j - FL_k + M(1 - Y_{kj}) \geq l_j \quad \forall j, k \in S,$$

(3h) $$FU_c \geq u_c \quad \forall c \in S,$$

(3i) $$X_{kj}, Y_{kj} = 1, 0 \quad \forall j, k \in S.$$

These constraints completely define the double-cycling problem. Constraints (3b) ensure that the makespan is greater than or equal to the completion of loading of all stacks. Constraints (3c) ensure that stacks are only loaded after all necessary stacks have been unloaded. Constraints (3d), (3e) and (3i) ensure that every stack is unloaded after the previous one in Π' has been unloaded. This is achieved by specifying for every pair of stacks (j, k) that either stack k is unloaded before stack j (if $X_{kj} = 1$) or the reverse (if $X_{kj} = 0$), and that the time difference between the two events is large enough to unload the second of the two stacks. Constraints

(3f), (3g) and (3i) are equivalent to (3d), (3e) and (3i) but for loading jobs. Constraints (3h) ensure that each unloading completion time allows for enough time to at least unload that stack.