Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

A GLOBAL AND LOCAL SEARCH APPROACH TO QUAY CRANE SCHEDULING PROBLEM

Kyrylo Perelygin
Joseph J. Kim

Department of Civil Engineering and Construction Management
California State University, Long Beach
1250 Bellflower Boulevard
Long Beach, CA 90840, USA

ABSTRACT

The container flow in terminals at a port is often bottlenecked due to the slow operations of the quay cranes in a scarce terminal land space. The quay crane scheduling problem (QCSP) is a major problem because of the assignment of expensive quay cranes to multiple vessel-holds for container discharging and loading operation. This paper presents a hybrid QCSP Solver, which combines genetic algorithms for global search with steepest ascent hill climbing for local search. Numerical experiments are performed with small- and large-sized random QCSP instances. The experimental results revealed that the hybrid QCSP Solver provides a better solution than the stand-alone QCSP Solver. By scheduling the dynamic operation of quay cranes it is expected that the developed decision making tool will provide terminal planners with a guideline to enhancing the assignment of quay cranes to a vessel.

1 INTRODUCTION

The QCSP is a major problem because of the assignment of expensive quay cranes to multiple vessel-holds for container discharging and loading operations. Thus, the feasible optimal and/or near-optimal solutions affect the overall operational performance of the whole terminal containers. The main goal of studying the QCSP is to determine the sequence of discharging and loading operations for a quay crane (QC) to perform with the objective function of minimizing the completion time of a ship operation. The characteristics of the QCSP are that it is similar to the m-parallel machine problem and that it is different because precedence relationships exist among tasks and because certain tasks cannot be performed simultaneously. In other words, cranes could not cross with each other. Inputs necessary for the QCSP include a ship stowage plan with all constraints, time required to carry out each task, crane travel time between different tasks, and crane ready time. Attention has mainly focused on a variety of objective functions to find a solution for the QCSP. Examples of such an objective function include minimization of the aggregate vessel delay cost (Peterkofsky and Daganzo 1990; Daganzo 1989 and 1990), maximization of the total profit by finding a crane-to-job match (Lim et al. 2004), minimization of the maximum relative tardiness of vessel departures (Liu et al. 2005), and minimization of the vessel's overall completion time (Kim and Park 2004; Sammarra et al. 2007; Lee et al. 2008; Legato et al. 2008).

Since quay crane service times are not deterministic, the use of the IP formulation to search for the optimal solution is no longer truly representative of the discharge/loading operations in port container terminals. Also, most mathematical models do not account for uncertainty. Various meta-heuristic approaches, such as genetic algorithms (GAs), are increasingly applied to the complex non-deterministic hard problem due to its remarkable capabilities of overcoming the existing methods. Many researchers have attempted to find the optimality for the QCSP to reflect reality. One of the trends in the genetic algorithm research domain is to develop a new meta-heuristic method using artificial intelligence and biologi-

cally inspired techniques. The concept of a hybrid genetic algorithm is becoming increasingly popular and has been successfully applied to many engineering optimization problems as well as a variety of problems in different fields, such as aerodynamic design, signal analysis, and water resources planning and management, among others. Although hybrid approaches using meta-heuristic methods are becoming increasingly popular and have been successfully applied to many engineering problems, the use of a hybrid approach, which combines a global search with a local search to the QCSP, needs to be more explored. The development of an adaptive hybrid genetic algorithm for the optimization of the QCSP is driven and motivated by both the lack of success in finding an efficient optimal solution algorithm to the QCSP and the need for an adaptive hybrid global-local search approach to the QCSP.

2 PROBLEM DEFINITION

The QCSP is a search problem involving the loading and unloading of containers with the use of quay cranes. The proposed algorithm assumed that a berth schedule has already been provided, although in practice the berth scheduling problem is affected by the vessel handling time which is dependent on the distance between the berthing position of the vessel and the storage area, the number of quay cranes assigned to the vessel, and the number of internal transport vehicles assigned to vessels' quay cranes (Boile et al. 2009). The objective is to find a way to minimize the time a ship must wait in port (load/unload the containers as rapidly as possible) and maximize the effective use of quay cranes available (they should not be idle). To simplify the complexity of the problem, a few assumptions are made: (1) QCs are identical in terms of productivity that loads and unloads containers, and (2) A berth schedule has already been provided. Consider just the ship and the QCs assigned to the ship. The QCSP has the following constraints: (1) QCs are on one track, and therefore cannot cross each other (if quay crane k handles ship bay b and quay crane k' handles ship bay b' then $k + 1 \leq k'$), (2) Each QC is functional and can be used at any time (if it's not already in use), (3) Some tasks precede others (A > B implies that A needs to be finished before B can start), (4) Some tasks cannot be performed simultaneously due to QCs interfering with each other, and (5) Every ship bay is handled by only one QC, and a QC can handle only one ship bay at any time.

3 HYBRID GENETIC ALGORITHM

This section presents a hybrid strategy to develop a hybrid QCSP Solver using genetic algorithm for global search and steepest ascent hill climbing for local search, followed by the step-by-step procedure of the algorithm development. The step-by-step procedure for development of a hybrid QCSP Solver includes chromosome encoding, fitness evaluation and objective function, GA input parameters, selection operation, reproduction operation using one-point crossover, invert mutation operation, local search using steepest ascent hill climbing algorithm, and termination conditions.

The elitist genetic algorithm (EGA), which is used as a base platform in developing an adaptive hybrid genetic algorithm for the QCSP, employs four basic operators, such as elite roulette wheel selection, one-point crossover, invert mutation, and steepest ascent hill climbing for local search. The initial population of possible solutions to the QCSP is created to apply the algorithm in the very first step of the global search. A fitness value of an individual in an initial population is calculated by constructing the sequence of quay cranes. The selection of the parent individuals is made through the elitist roulette wheel selection operator for the next generation. Using the parent individuals obtained from the selection operator, one-point crossover operator is performed by exchanging parent individual segments and then recombining them to produce two resulting offspring individuals. The invert mutation operator is performed to play the role of random local search, which searches a much smaller portion than hill climbing algorithm. The invert mutation is kept as a simple mutation to avoid conflicts with hill climbing algorithm. The local search using the steepest ascent hill climbing is then achieved before the move to the new population embedded in the EGA.

An integer string is used because it has the added advantage of readability compared to a binary representation. The fitness value calculated is the reciprocal of the objective function. The objective function is formulated using Weighed (α, alpha) time at which all tasks are completed + Weighed (β, beta) total quay crane completion time. The two default values for weighing are alpha = 0.75, and beta = 0.25. Note that the time at which all tasks are completed is the makespan. The hybrid algorithm continuously keeps track of the chromosome that represents the current most-fit solution. Upon the algorithm reaching the termination condition, that chromosome is generated as an output. The makespan is calculated by constructing a schedule out of the DNA sequence. It is assumed that the quay cranes can be correctly positioned since the start. So, if the start of the sequence is {1, 6, 7}, the quay cranes #1, #2, and #3 will be in position 1, 6, and 7 respectively. The hybrid algorithm is developed in five steps as follows:

Step 1: Choose two quay cranes that can work on bay k. Let's call them qcl and qcr. These quay cranes are chosen by looking to the left and right of bay k, and selecting the first two quay cranes encountered (one from both sides). This ensures that the constraint that the quay cranes cannot cross each other is enforced. If there is only one quay crane, then that's the winner and therefore the quay crane that is used for loading and unloading the bay – go to step 4. However, if there are two quay cranes, then go to step 2.

Step 2: Choose the quay crane with the smaller completion time. This will be the winner – move on to step 4. If the completion times of qcl and qcr are equal, go to step 3.

Step 3: Out of the two quay cranes, choose the one that is closer to bay k. (Absolute value of [current quay crane position – bay position]). Move on to step 4.

Step 4: Check all the other quay cranes, and if their completion times are smaller than the winner that we have chosen, set their completion time to the completion time of the winner. This is done because since other quay cranes weren't chosen for this bay, it means they were blocked by the winner, and so for their next move they would have to wait for the winner to move on to the next bay.

Step 5: Add the completion time of bay k, to the completion time of the winner QC. Set the position of the winner to the position of bay k, and repeat from step 1. Once the schedule has been constructed, and all the different time values have been summed up, the fitness value is calculated.

The selection operation is implemented with a method called stochastic sampling or more commonly known as the "Roulette Wheel" selection method. The selection operation generally plays a role of choosing parent chromosomes for crossover operation. This method allows for 'elitism,' because the more fit chromosomes will get chosen more frequently, however, lower fitness solutions also get a chance. The reproduction of the population is handled with a one-point crossover technique. A crossover operator combines pieces of information coming from different individuals in a population. The one-point crossover operator can preserve schemata in a more effective manner because it keeps the first half of both parents intact and is less random than the UX3. The probability of disrupting short defining length is rather low, even though the crossover operation in the beginning of an individual is likely to disrupt schema (Goldberg 1989). This involves randomly determining a crossover point in the integer chromosome representation and swapping all the data beyond that point in both parents. Therefore, the offspring has data before the pivot point from parent A and data after the pivot point from parent B.

In the hybrid genetic algorithm, a technique called "invert" mutation is employed. The invert mutation consists of choosing two random points in the chromosome and inverting the order of everything between those two points. The invert mutation is proved to be the most effective mutation technique out of the four well-known four mutations of swap, insert, invert, and scramble mutations. This is mainly due to the no-crossing of quay cranes constraint during the construction of the schedule. For example, let us assume that two quay cranes are available at positions 3 and 8, respectively and the sequence in the DNA is {5, 4, 6}. Quay crane 1 starts work on bay number 5 because it is closer than quay crane 2. Now bay 4 is available in the sequence, but quay crane 2 cannot access it since it would have to cross quay crane 1 (so it is effectively 'blocked'). Quay crane 2 is now forced to wait until quay crane 1 is completed with its

work. However, if the order of those two last numbers is switched and the sequence becomes {5, 6, 4}, quay crane 2 would be allowed to start working on bay 6 while quay crane 1 is on bay 5. This occurs constantly when solving the QCSP and thus, a well-timed mutation can lead to finding a better solution. It is important to note that while in the example above, a simple 'swap' mutation might do the trick it is much more efficient to do an 'invert' mutation since the swap would only work if the swap occurs between two adjacent numbers in the DNA sequence. The range of numbers to invert mutation is not chosen at random, but instead it is the number of quay cranes. This allows the user to overcome problems like in the example described above, but with more than two quay cranes sometimes swap mutation wouldn't be suitable.

This procedure is repeated five times for each chromosome in the population (Kim and Ellis 2009). It can be argued that the current mutation operation that is implemented here is redundant as it performs a function like the local search. Instead it can be used for a more extreme exploring of the search space by making very large mutations in the DNA sequence. This, however, has not shown a noticeable improvement in the results because for accurate results the hybrid QCSP Solver is run a couple of times for each problem. While the more complex mutation might play a role in one of the results by allowing for a different search space to be explored during execution, it would be very difficult to determine if the better result was obtained because of the more extreme mutation operation, or due to the way the random population was generated in the beginning of each generation. The mutation was kept as a simple invert mutation.

4 COMPUTATIONAL EXPERIMENTS

The hybrid QCSP Solver is written in C++, which allows for good performance as well as portability among different systems (the program runs on Windows, Linux, and *BSD). Figure 1 shows the GUI of the hybrid QCSP Solver. Genetic algorithms lend themselves nicely to abstraction via C++ classes, making the program code clean and easy to understand. The hybrid QCSP Solver has three main classes which represent the QCSP problem; CSpecies() represents the whole species and therefore the whole algorithm. Inside, it contains a CGeneration() class, which represents one generation and one population of a developing species, and finally COrganism() represents one organism in a population, containing the fitness value, schedule, and genetic representation of the schedule. The size of the population as well as rates for mutation and crossover can be defined via the command line.

A comparison study is performed to demonstrate the effectiveness of the algorithm based on numerical experiments. Two different experiments were performed to examine the performance of both stand-alone QCSP Solver and hybrid QCSP Solver. The experiments consist of small-sized QCSP and large-sized QCSP instances. The QCSP instances are generated using the QCSP Problem Generator that is embedded in hybrid QCSP Solvers. The QCSP Problem Generator allows the user to create random instances by specifying the number of bays, the number of quay cranes, and minimum and maximum bay completion times in minutes. For the numerical results for hybrid QCSP Solver, the results obtained from the hybrid QCSP Solver by solving small-sized random QCSP instances are compared against both fitness lower bound and best possible solution (BPS) algorithm in addition to the comparison between their runtimes to examine the difference between their speeds to reach solutions. The 16 large-sized random QCSP instances are solved to compare the results between fitness bound and fitness value obtained from the hybrid QCSP Solver by three different combinations of objective function weighting values, α and β. Input parameter values for both stand-alone QCSP and hybrid QCSP Solver include the population size, crossover and mutation rates are set to 50, 0.5, and 0.1 for supporting the solution diversity, respectively. Large populations generally result in better solution, but they also increase computational costs and memory requirements. The algorithm terminated with the number of generation of 100. The objective function weighting values are set to 0.75 and 0.25 for weighed (α, alpha) time at which all tasks are completed and weighed (β, beta) total quay crane completion time, respectively.

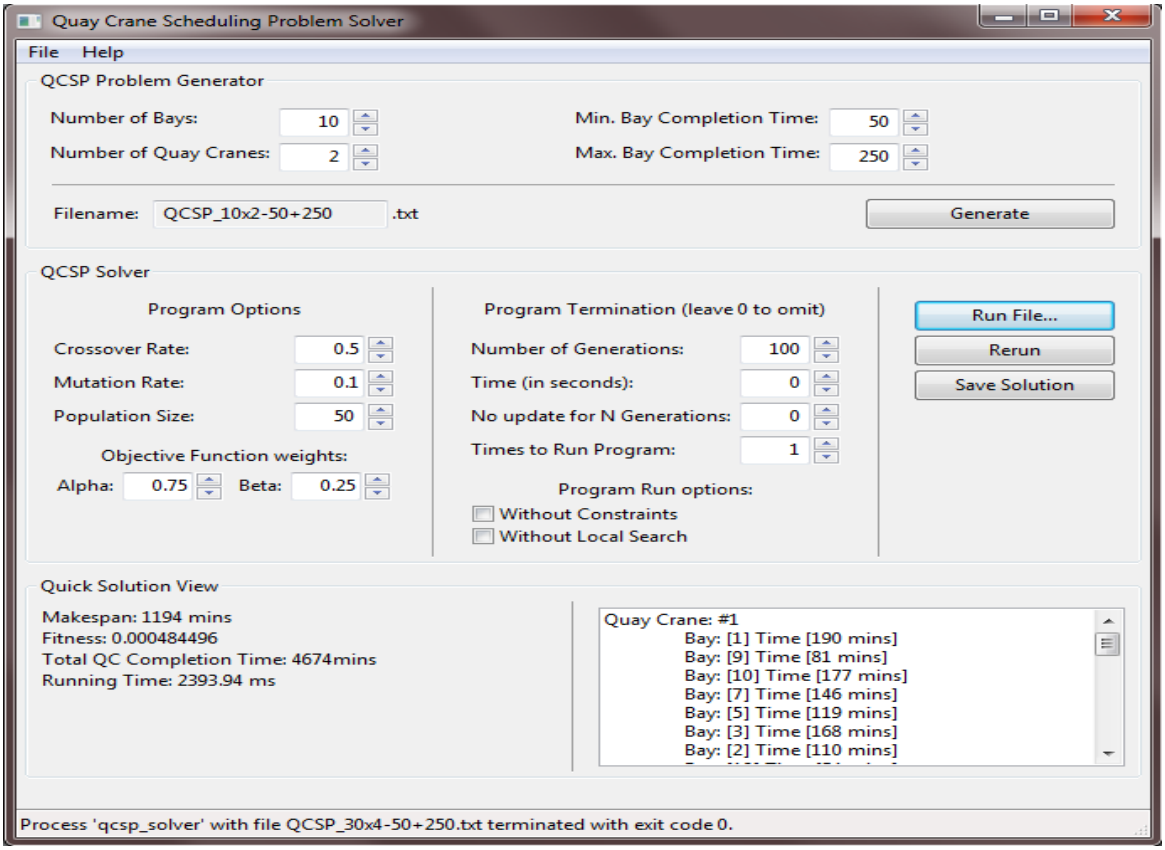

Figure 1: GUI for hybrid QCSP Solver.

4.1 Small-sized Random QCSP Instances

Five small-sized random QCSP instances are generated using the QCSP Problem Generator. The processing time of a bay is generated from a uniform distribution from 50 to 250, allowing for an accurate simulation of a real-life problem. To show the performance of the hybrid QCSP Solver, the results obtained from the hybrid QCSP Solver are compared with the results obtained by calculating a fitness bound and the BPS. Table 1 tabulates the results obtained from the hybrid QCSP Solver, fitness bound, and BPS, by solving five small-sized random QCSP instances. The fitness bound is a lower bound on the objective function and it helps in the evaluation of the performance of the hybrid QCSP Solver. It is calculated by removing the no-crossing constraint, therefore allowing the quay cranes to move to the bay of their choice, regardless if they cross other quay cranes or not. Although practically it is not possible to achieve a value identical to the lower bound it is still a good indication of how efficient a heuristic is. The best possible solution is achieved by populating the current species with an enumeration of all the permutations of the genetic sequence. This means that every possible combination of creating the QCSP schedule is checked, and the 'best' solution is found. BPS is implemented only for small-sized random QCSP instances because it would not be feasible to implement for large instances. The BPS Difference column calculates a percentage of how far the solutions obtained from the hybrid QCSP Solver is from the actual solution to validate the results. It was shown that the results obtained from the hybrid QCSP Solver exactly match the results obtained from the BPS algorithm. This result indicates that the hybrid QCSP Solver can accurately find a solution and in these cases, the best one to the QCSP. It is also important to note that the runtime of the hybrid QCSP Solver does not grow exponentially like the problem search space but instead takes a constant 30-50 ms extra for each new bay.

Table 1: Comparison results for small-sized instances.

Experiment No.	Size (bays x cranes)	Fitness Bound	BPS		Hybrid QCSP Solver		BPS Difference
			Value	Runtime (ms.)	Value	Runtime (ms.)	
1	7 x 2	2.06186	2.05867	9	2.05867	136	0%
2	8 x 2	1.58228	1.5804	89	1.5804	167	0%
3	9 x 2	1.15207	1.15207	882	1.15207	214	0%
4	10 x 2	1.03093	1.02617	10,051	1.02617	258	0%
5	10 x 3	1.18483	1.17751	12,104	1.17751	303	0%

4.2 Large-sized Random QCSP Instances

Table 2 tabulates the results obtained from the hybrid QCSP Solver and fitness bound by solving 16 large-sized random QCSP instances for three different combinations of objective function weighting values. This result shows that when the importance of the total quay crane completion time represented with Beta increases from 0 to 0.5, the better fitness is also obtained. A value of 0.5 means that a solution places equal importance on having the lowest makespan and the lowest total quay crane completion time. The results indicate that the solutions produced by both stand-alone QCSP Solver and hybrid QCSP Solver have a pattern in that as the total quay crane completion time increases, the better fitness values are obtained.

Table 2: Comparison results for large-sized instances.

Exp. No.	Size (bays x cranes)	Max. Runtime (sec.)	Alpha / Beta: 1.0 / 0.0		Alpha / Beta: 0.75 / 0.25		Alpha / Beta: 0.5 / 0.5	
			Fitness Bound	Fitness Value	Fitness Bound	Fitness Value	Fitness Bound	Fitness Value
1	16 x 3	0.69	1.26103	1.25945	0.840689	0.84016	0.630517	0.630318
2	18 x 3	0.85	1.01317	1.01317	0.675676	0.675676	0.506842	0.506842
3	20 x 3	1.02	0.913242	0.911577	0.608828	0.608273	0.456621	0.456517
4	22 x 3	1.23	0.967118	0.966184	0.644745	0.644434	0.483559	0.483442
5	24 x 3	1.38	1.02459	1.02354	0.683177	0.683177	0.512426	0.512295
6	26 x 3	1.57	0.727273	0.726744	0.484966	0.484614	0.363769	0.363702
7	28 x 3	1.79	0.715308	0.714286	0.476872	0.476531	0.357654	0.35727
8	30 x 3	2.02	0.681199	0.678887	0.454133	0.453361	0.340599	0.340078
9	16 x 4	0.79	1.83824	1.81488	1.05125	1.04548	0.736106	0.734214
10	18 x 4	0.98	1.34409	1.32626	0.768344	0.763942	0.537924	0.535332
11	20 x 4	1.20	1.33156	1.31062	0.76089	0.759157	0.532623	0.531632
12	22 x 4	1.38	1.50602	1.49254	0.861141	0.857817	0.602954	0.601866
13	24 x 4	1.64	1.10865	1.09649	0.633513	0.628141	0.443459	0.442478
14	26 x 4	1.96	0.978474	0.969932	0.559284	0.556715	0.391543	0.390244
15	28 x 4	2.15	1.04384	1.03093	0.596481	0.59312	0.417537	0.41632
16	30 x 4	2.44	0.855432	0.844595	0.488938	0.483033	0.34229	0.34118

The fitness value results obtained from the hybrid QCSP Solver are very promising and always within 1% of the lower bound. As mentioned in the results of Table 1, the runtime of the algorithm does not grow exponentially. Instead, a constant increase of 150-200 ms per each extra bay in the problem is observed for problems with three quay cranes, while a constant increase of 200-300 ms is also observed for problems with four quay cranes. This is mostly due to the linear complexity of the local search algorithm, and by reducing the number of maximum iterations performed by the hill climbing technique, there is a decrease in the execution time. It is important to strike a balance between performance and execution time.

Table 3 tabulates the makespan and QC completion time for each experiment used in Table 2 by objective function weighting value. The intention of the comparison is to examine how the changing of the two weighting values, alpha and beta, affects the outcome of the hybrid algorithm. In exp. 1 and exp. 12, the hybrid QCSP Solver finds the best solution and therefore changing the weighting values will not lead to a better or worse solution for smaller problem sizes. However, already in exp. 9, which is a more complex problem, it is shown that the hybrid QCSP Solver can find the lowest makespan but there are different QC completion times to choose from. In the first case (Alpha/Beta: 1.0/0.0) it did not take that into consideration. However, when the beta weight is given a value of more than 0.0, the hybrid QCSP Solver minimizes it as well, leading to a better result. In addition, it is important to note that experiments 15 and 16 are highly complex problems. Here the hybrid QCSP Solver must make trade-offs, leading to a higher makespan when adding the Beta weighting value but a lower QC completion time. While the goal of the hybrid QCSP Solver is to minimize the makespan, it is important that the total QC completion time is taken into consideration so that out of the best solutions with the lowest makespan, the hybrid QCSP Solver will choose the one that has a lower total QC completion time.

Table 3: Comparison of makespan and QC completion time for large-sized instances.

From Experiment	Size (bays x cranes)	Alpha / Beta: 1.0 / 0.0		Alpha / Beta: .75 / .25		Alpha / Beta: .5 / .5	
		Makespan (min.)	QC Completion Time (min.)	Makespan (min.)	QC Completion Time (min.)	Makespan (min.)	QC Completion Time (min.)
1	16 x 3	794	2379	794	2379	794	2379
2	18 x 3	987	2959	987	2959	987	2959
3	20 x 3	1097	3285	1097	3285	1096	3285
4	22 x 3	1035	3102	1035	3102	1035	3102
5	24 x 3	977	2928	976	2927	977	2927
6	26 x 3	1376	4123	1377	4123	1376	4123
7	28 x 3	1400	4194	1400	4194	1404	4194
8	30 x 3	1473	4409	1473	4404	1477	4404
9	16 x 4	551	2181	551	2173	551	2173
10	18 x 4	754	2981	754	2974	754	2982
11	20 x 4	763	3015	755	3004	755	3007
12	22 x 4	670	2653	670	2653	670	2653
13	24 x 4	912	3617	920	3608	908	3612
14	26 x 4	1031	4105	1033	4086	1034	4091
15	28 x 4	970	3857	968	3840	972	3832
16	30 x 4	1184	4723	1193	4702	1185	4677

5 CONCLUDING REMARKS

This paper presented a hybrid genetic algorithm to allocate and schedule a given number of quay cranes to vessels planned to arrive in the planning horizon. To develop the hybrid QCSP Solver, stand-alone QCSP Solver not having local search was first developed using genetic algorithm alone. Then the hybrid QCSP Solver having local search was developed on top of the stand-alone QCSP Solver based on a hybrid strategy that combines a global search using genetic algorithm with a local search using steepest ascent hill climbing algorithm. A comparison study demonstrates the effectiveness of the algorithm based on numerical experiments. It is notable that the hybrid QCSP Solver is expandable to incorporate the real quay crane operation conditions as it can serve as a platform from which one can build. This paper can be

utilized as a stepping stone for further research by identifying a way to grow terminal automation of quay cranes to increase capacity, safety, productivity, and reliability while reducing operation costs.

ACKNOWLEDGMENT

This material is based upon work supported by the Office of Naval Research, under Prime Agreement No. N00014-09-C-0923 with the California State University, Long Beach Foundation, Center for the Commercial Deployment of Transportation Technologies (CCDoTT).

REFERENCES

Boile, M., Golias, M., and Theofanis, S. 2009. "Scheduling of Berthing Resources at a Marine Container Terminal via the Use of Genetic Algorithms: Current and Future Research." *Evolutionary Computation*, Wellington Pinheiro dos Santos (Ed.), October 2009, InTech, Vienna, Austria, 572.

Daganzo, C. F. 1989. "The Crane Scheduling Problem." *Transportation Research B*, 23(3), 159-175.

Daganzo, C. F. 1990. "Crane Productivity and Ship Delay in Ports." *Transportation Research Record*, 1251, 1-9.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Kim, J.-L., and Ellis, R. D. 2009. "Robust Global and Local Search Approach to Resource-Constrained Project Scheduling." *Canadian Journal of Civil Engineering*, 36(3), 375-388.

Kim, H., and Park, Y. M. 2004. "A Crane Scheduling Method for Port Container Terminals." *European Journal of Operations Research*, 156(3), 752-768.

Lee, D., Wang, H. Q., and Miao, L. 2008. "Quay Crane Scheduling with Non-Interference Constraints in Port Container Terminals." *Transportation Research Part E*, 44, 124-135.

Legato, P., Mazza, R. M., and Trunfio, R. 2008. "Simulation-based Optimization for the Quay Crane Scheduling Problem". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 2717-2725. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Peterkofsky, R. I., and Daganzo, C. F. 1990. "A Branch and Bound Solution Method for the Crane Scheduling Problem." *Transportation Research B*, 24, 159-172.

Sammarra, M., Cordeau, J.-F., Laporte, H., and Monaco, M. F. 2007. "A Tabu Search Heuristic for the Quay Crane Scheduling Problem." *Journal of Scheduling*, 10, 327-336.

AUTHOR BIOGRAPHIES

KYRYLO PERELYGIN, M.S., is a Senior Systems Software Engineer at NVIDIA and was a Graduate Assistant of Dept. of Civil Engineering and Construction Management at California State University, Long Beach, where he worked for a research project regarding various scheduling optimization techniques. His email address is kperelygin@gmail.com.

JOSEPH J. KIM, Ph.D., P.E., LEED AP BD+C is an Associate Professor of Dept. of Civil Engineering and Construction Management at California State University, Long Beach. He is a director of Green Building Information Modeling (Green BIM) laboratory at CSULB. He has earned a doctorate degree in Civil Engineering from the University of Florida, majoring Construction Engineering and Management with a minor in Statistics. He spent several years as a field engineer and safety engineer. He is a registered professional engineer in Florida. His research interests include sustainable design and construction, simulation-based resource scheduling, optimization techniques, building information modeling, information technology in construction, and engineering educational research methods. He is a member of ASCE and ASEE. His email address is jin5176@gmail.com.