OPERATOR SCHEDULING USING MIN-CONFLICTS AND TABU SEARCH BASED HEURISTIC IN CONTAINER TERMINALS

Vlado Popović [a]

[a] University of Belgrade, Faculty of Transport and Traffic Engineering, Serbia

Abstract: *In this paper a new approach to the operator scheduling problem in container terminals is presented. The problem is related to assigning operators to handling equipment while satisfying requirements for operators over shift and work regulations. It is considered in a case of disruption, when it is impossible to make a schedule so that all requirements for operators and work regulations are satisfied. In that situation the least bad solution is needed. For that purpose the problem is considered like a maximum constraint satisfaction problem (Max CSP) and solved by a heuristic based on Min-conflicts heuristic and Tabu search.*

Keywords*: operator scheduling, container terminals, Min-conflicts heuristic.*

1. INTRODUCTION

Resource scheduling in container terminals is a very important activity due to a great impact on the efficiency and the quality of service, and thus on the terminal business. It is especially related to scheduling of handling equipment, but also to workers (operators) who operate with them. In fact, operators are highly qualified workforce and cause a big expense for terminals. On the other hand, for the sake of their safety and safety of goods, their work is regulated by numerous strict work regulations.

Kim at al. (2004) pointed out that operators are usually scheduled after determining their number in each shift and scheduling of the operation time of the equipment. Accordingly, they defined operator scheduling as the problem of assigning the certain number of operators to time slots of equipment usage while satisfying the requirement for the number of operators in a specific time slot, various regulations for workforce utilization, and the requirements of operators for assignments. Considering that definition, the problem corresponds to a CSP and Kim at al. (2004) solved it using ILOG Solver. However, in that case the possibility of obtaining a schedule that does not meet all regulations and requirements is eliminated. Consequently, if some disruptions occur such as an unplanned operator absence or an increased workload, there would be a chance that non schedule could be made i.e. non solution could be found.

For that reason in this paper operator scheduling is considered as the problem of assigning operators to the equipment with the aim of satisfying as many requirements for operators and work regulations as possible. That means that the sum of intervals in which the required number of operators is satisfied and satisfied work regulations is maximized, where satisfying each work regulation are observed for each operator separately. According to that, the problem corresponds to a Max CSP, because its aim is actually meeting as many constraints as possible. In

order to solve that Max CSP, a heuristic based on Min-conflicts heuristic (MCH) and Tabu search (TS) is developed.

The rest of the paper is organized as follows. In the next chapter the problem is described and related literature is given. The third chapter is dedicated to MCH, whereas the fourth to the developed heuristic. The fifth chapter presents the result of the application of developed heuristic to an operator scheduling problem in the port container terminal, and the last gives concluding observations.

2. PROBLEM STATEMENT AND LITERATURE REVIEW

Environment in which the problem is considered involves the following: 1.) The handling equipment is related to machines such as container cranes, yard cranes and yard trucks; 2.) The shift is composed of a known number of one hour time slots. 3.) For each piece of equipment in each time slot is determined whether it is used or not, based on the schedule of the operation time of equipment. 4) Work regulations are the same as those in the container terminal in Pusan (Republic of Korea) and they are presented in Kim at al. (2004). They specify the minimum and maximum operating time allowed per shift (S, L), the minimum and maximum consecutive operating time (session) allowed for every operator (S_1, L_1), types of equipment that can be assigned to each operator, the minimum length of the rest period (F), necessity for the rest after a session. They also involve disallowance of assigning one operator to multiple pieces of equipment in one time slot. 5.) Given the great complexity of the problem, in this paper an assumption with the aim of the problem relaxation is introduced. It is assumed that each operator can handle any type of equipment. Accordingly, types of the pieces of equipment, which work in a specific time slot are no longer important, but the total number of them. That number represents the requirement for operators in a specific time slot.

Considering these characteristics of the problem, the input data for the problem are the total number of time slots in the shift *N*, the total number of pieces of equipment *M,* the total number of pieces of equipment which work by time slot e_i ($i = 1,2, \dots, N$), the total number of operators in the shift *R* and parameters in the work regulations. The output of the problems is the schedule that defines for each operator when and on which piece of equipment work.

It is adopted that it is equally important to satisfy every requirement for operators and work regulation, except of the last regulation that takes precedence over all others. In mathematical terms the problem consists of constraints that represent the work regulations and the requirements for operators. Here, all constraints are expressed by variable formulated as x_{ij} ($i = 1,2, \dots, N; j = 1,2, \dots, M$) whose value r ($r = 0,1,2, \dots, R$) represents the operator number assigned to time slot i on equipment j (the decision variable); and $x_{ij} = 0$ when no operator is assigned to time slot *i* on equipment *j*.

In addition to the paper (Kim et al., 2004), Legato and Monaco (2004) dealt with this problem too. They solved it for a longer period in which they determine days off for the operators, and for a shorter period in which they assign operators to the shifts, tasks and resources. However, the operating rules apply only in determining the days off, free time between shifts, etc. Murty et al. (2005) dealt with determining the number of workers by shifts in the horizontal transport processes, while respecting the rule of break during the shift. Zampelli at al. (2013) carried out the assignment of group of workers to the cranes during a specific time period. Operators group (gang) was viewed as a workforce "consumed" by the crane, except during the break period.

3. MAX CSP AND MIN-CONFLICTS HEURISTIC

Generally, CSP is a problem that involves a set of constraints and a set of variables, where each variable has each own domain of values. It is solved by using various types of search, and the

solution is found if each variable has a value and all constraints are satisfied. A CSP becomes a Max CSP if a solution which does not satisfy all constraints is acceptable. In that case the objective of the problem solving, and hence of the search, is finding a solution which satisfies as many constraints as possible. According to that, methods of local search are naturally suitable for Max CSP, because Max CSP can be considered like optimization problem with optimization objective of minimizing the number of violated constraints. Therefore, most local search algorithms for CSP can be directly applied to Max CSP, since their evaluation function directly corresponds to the optimization objective of Max CSP (Rossi at al., 2006).

Probably the best known and most widely used method of local search for solving CSP (hence Max CSP) is MCH, due to good results which are obtained by its application to the complex CSPs. Particularly good results were obtained for the well-known problems: "Telescope Problem" and "N-Queen Problem" (Russel and Norvig, 2003). It represents an iterative improvement algorithm which changes the value of a randomly selected variable located in the conflict set in each iteration. The conflict set consists of variables present in the constraints violated by the solution of the previous iteration. Each variable in the conflict set has a certain number of conflicts, which is equal to the number of violated constraints which contain a specific variable. A total number of conflicts is obtained by adding up conflicts of every variable from the conflict set and being updated along with the conflicts set after each iteration. The new value that is assigned to the chosen variable should provide a minimum total number of conflicts in comparison to other values from the variable domain. If more than one value from the domain has this feature, the choice between them is random again. The searching process is stopped when the values for all variables are found so that the total number of conflicts is equal to zero, i.e. all constraints are satisfied, or after reaching a predefined number of iterations.

Like most iterative improvement methods, MCH can get stuck in a local optimum of the underlying evaluation function; and therefore it is essentially incomplete (Rossi at al., 2006). In order to overcome that problem the rules such as “restart” or “random walk” or some metaheuristics are usually used. A metaheuristic that is often used in a combination with the MCH is TS. It help the search process escape from a local optimum by forbidding moves which take the solution, in the next iteration, to points in the solution space previously visited (hence "tabu").That is implemented by putting tabu on each new pair of variable-value, for a fixed number of iterations.

4. HEURISTIC

For problem solving, a special heuristic based on MCH and supported by TS is developed. MCH is chosen because it is generally one of the best methods of local search for CSPs, but also due to the feature that tends to provide a solution that is the least different from the initial one. This can be useful for the problem discussed in this paper if the schedule is made a day before. Min-conflicts heuristic is therefore described as a "repair" or "online" technique (Russel and Norvig, 2003).

Variable formulated as y_{ir} (i=1,2,…,N; r=1,2,…,K) is introduced for the implementation of the heuristic. Its value represents the number of constraints in time slot i which are unsatisfied because of an operator r, i.e. the number of conflicts that the operator r has in time slot i. Variable y_{ir} increases its value by one if the operator r is assigned to a piece of equipment in the time slot i, i.e. value r is assigned to one of the variables x_{ij} (j=1,2,…,M), and it causes some constraints to become unsatisfied. Also, variable y_{ir} increases its value by one if the operator r is not assigned to any piece of equipment in the time slot i, and it causes some constraints to become unsatisfied. If one of these constraints relates to the last regulation it is defined that the corresponding variable y_{ir} enormously increases its value. Hence, it is impossible for it to be violated in a solution. Therefore, the number of conflicts of each value r in each time slot is considered in this case, and not the number of conflicts of each x_{ij} variable. This is done due to

the fact that all constraints regulate operators work in time, and it is easier to make algorithm of the heuristic with variable defined as y_{ir}.Though, in this way a slightly deviation of MCH is made.

Because of using TS, variable formulated as a_{ir} (i=1,2,...,N; r=1,2,...,K) is also involved in the heuristic. It demonstrates the value of tabu for the corresponding y_{ir} variable. The value of tabu is assigned to a_{ir} variable when one of the variables x_{ij} (j=1,2,...,M) obtains a new r value. Due to that, the number of conflicts that the operator r has in the time slot i is not considered during the next iterations whose number is equal to the value of tabu.

The end of heuristic consists of several steps which attempt to improve the best solution obtained by the part of heuristic based on MCH (if there are conflicts). It is performed through exchanging positions (pieces of equipment) between an operator r that causes conflicts in a time slot with other operators who also work in that time slot. This part is introduced because the previous implies only changing a chosen value r. However, it may only be needed to replace that value with the corresponding value from the same time slot, in order to avoid conflicts.

The algorithm of heuristic is presented in the following subsection. It involves 16 steps, where the part for improving begins from the step 11. In order to facilitate its explanation, three matrices **X**, **Y** and **A** are introduced. The matrix **X** (*NxM*) contains values r of the variables x_{ij} and represents a solution of the problem in an iteration. Matrix **Y** (*NxK*) contains values of y_{ir} variables. The sum of the values of all y_{ir} variables represents the number of conflicts that a solution **X** has, and at the same time the value of the objective function in the heuristic. Matrix **A** (*NxK*) contains tabu values that are related to y_{ir} variables, i.e. values of a_{ir} variables. In addition to the matrices, the sets I = {1,2 ..., N}, J = {1,2 ..., M} and R = {0,1, ..., K} are also introduced.

4.1 Algorithm of heuristic

1.) Set the *global minimum* on a sufficiently big value (bigger than the maximum number of conflicts), equate the *iteration number* to zero and define the tabu value and maximum number of iterations. Assign value from the initial solution to each x_{ij} variable, or zero if the initial solution does not exist, remember that set of values (matrix **X**) as the best solution. Go to the next step.

2.) Increase the *iteration number* by one. If the *iteration number* is smaller than or equal to the defined maximum number of iterations then go to the next step, otherwise go to the step 10.

3.) Calculate values of all y_{ir}variables. Set the *number of conflicts in iteration (nci)* = maximum (maximum > the maximum possible number of conflicts) and go to the next step.

4.) Among y_{ir} variables choose the one which has the biggest value (let it be variable $y_{\boldsymbol{ir}}$), for which the corresponding variable $a_{\boldsymbol{ir}}$=0. If there are more than one such variable, choose the first and go to the next step.

5.) Find variables $x_{\boldsymbol{ij}}$ ($j = 1,2, \dots, M$) which have the $\boldsymbol{r}$ value. If none has that value go to the 6a step, otherwise go to the 6b step.

6a.) For each variable $x_{\boldsymbol{ij}}$ ($j = 1,2, \dots, M$) check how many conflicts would have a solution if the $\boldsymbol{r}$ value would be assigned to that variable. Assign the $\boldsymbol{r}$ value to the variable for which the solution has the smallest number of conflicts (let it be $x_{\boldsymbol{ij}}$ variable). If that number of conflicts is smaller than the current value of *nci*: remember $x_{\boldsymbol{ij}}$ variable and the $\boldsymbol{r}$ value as its *new value*, forget the previously remembered and its *new value*. Go to the step 7.

6b.) Choose the first $x_{\boldsymbol{ij}}$ variable which has the $\boldsymbol{r}$ value (let it be $x_{\boldsymbol{ij}}$ variable). For each value $r_1 \in R \setminus \{\boldsymbol{r}\}$, check how many conflict have a solution if that one would be assigned to $x_{\boldsymbol{ij}}$ variable. Assign the r_1 value for which the solution has the smallest number of conflicts to the $x_{\boldsymbol{ij}}$ variable (let it be the $\boldsymbol{r_1}$ value). If the solution with a smaller number of conflicts than the

current value of *nci* is reached: remember that number of conflicts as a new value of *nci*, remember x_{ij} variable and the $\boldsymbol{r_1}$value as its *new value,* forget the previously remembered and its *new value*. Go to the step 7.

7.) If there are still some y_{ir} variables which should be checked (which have the same value as the previously checked y_{ir}, and for which the corresponding tabu values are equal to zero), choose the next and go to the step 5, otherwise go to the next step.

8.) Assign the *new value* to the remembered x_{ij} variable. Update matrix **X**. Mark the y_{in} variable as tabu, i.e. assign the tabu value to the a_{in} variable, where *n=new value.* Update matrix **A** and go to the next step.

9.) In the matrix **A** decrease each value bigger than zero, by one. If the value of *nci* is smaller than value of the *global minimum* then: set the new value of *global minimum* to be equal to the value of *nci*: forget the previously remembered best solutions and remember the matrix **X** as the best solution. If the value of *nci* is equal to the value of the *global minimum*, remember the matrix **X** as the second found best solution. Return to the step 2.

10.) If *global minimum*=0, mark the first best solution as final and go to the step 16. Otherwise choose the first best solution and go to the next step.

11.) Choose the first y_{ir}variable which has the value bigger than zero (let it be y_{ir} variable) and go to the next step.

12.) Among x_{ij} variables ($j = 1,2, \dots, M$) find the ones which have the $\boldsymbol{r}$ value, choose the first (let it be x_{ij} variable) and go to the next step. If there are no such variables, go to the step 14.

13.) Check the total number of conflicts which is obtained by replacing values between variable x_{ij}=$\boldsymbol{r}$ and every second variable x_{ij_1}, ($j_1 \in J \setminus \{j\}$). If one of the replacements provides solution which has the total number of conflicts smaller than the value of the *global minimum*: set that number of conflicts as value of the *global minimum;* carry that replacement out; Update the matrix **X**; forget the previous improved solution and remember matrix **X** as the improved solution. If the value of the *global minimum* is equal to zero, mark the improved solution as the final and go to the step 16, otherwise go to the next step.

14.) If there are still some y_{ir} variables with values bigger than zero, choose the next and go to the step 12. Otherwise go to the next step.

15.) If there are still best solutions, choose the next and go to the step 11. Otherwise, mark the improved solution as the final one, if exists, and go to the step 16. If does not exist, mark each best solution as the final and go to the next step.

16.) Finish the algorithm.

5. EXAMPLE

Problem selected for the example is similar to the solved one in Kim at al. (2004). It differs only in the number of pieces of equipment which work in the last time slot. It is set that three of them work instead of two. Before its solving with the developed heuristic, it was checked whether it can be solved as a CSP by using Microsoft Solver Foundation (MSF). A Constraint Programming Solver within MSF did not manage to solve it, which means that there isn't any solution that meets all constraints. Solving the problem as a Max CSP, using the developed heuristics in the Visual Basic environment, the solution with one conflict, i.e. one violated constraint is obtained. It refers to the work regulation that regulates the minimum number of consequent time slots which an operator has to operate when starts. The solution, together with the problem parameters, is given in Table 1. In the solution the numbers are referred to the operators, the columns to the pieces of equipment, whereas the rows to the time slots. The underlined number

indicates the operator who violates the regulation, i.e. the *r* value which generates the conflict. Therefore, the operator three violates the regulation by working on the second piece of equipment in the sixth time slot.

Table 1. Example of the problem with solution

Problem parameters	Search characteristics	The solution
N=6, M=4, K=6, tabu=25,e_1=3, e_2=4, e_3=4, e_4=3, e_5=4, e_6=3, L=4, S=2, L_1=3, S_1=2, G=1, F=2;	The smallest number of conflicts reached during the search: 2; Total number of solutions with the same number of conflicts: 8; They occur in iterations: 120, 165, 175, 477, 584, 974, 1040, 1128; The final solution is obtained through improving the third best solution and has one conflict.	1 4 3 0 1 4 3 6 5 2 3 6 5 2 0 6 5 2 1 4 0 <u>3</u> 1 4

6. CONCLUSION

Using the new approach, the operator scheduling problem in container terminals is considered as Max CSP. In order to be able to solve it, the heuristic based on MCH and TS is developed, which is therefore the major contribution of this work. In this way it is ensured that the problem has a solution in a situation when it is not possible to meet all work regulations and requirements for operators. That solution has the smallest number of unsatisfied requirements and regulations, which can be found by the heuristic. Also, in this way the approach used in Kim et al. (2004) is extended. The problem could be also solved as a Weighted Constraint Satisfaction Problem (WCSP) by using the developed heuristic. It would only require defining a larger number of conflicts for the violation of constraints related to the more important regulations and requirements. Although it is not applied here, this option would very likely be useful in the real life conditions.

REFERENCES

[1] Kim, K.H., Kim, K.W., Hwang, H., Ko, C.S., (2004). Operator-scheduling using a constraint satisfaction technique in port container terminals. *Computers and Industrial Engineering* 46(2), pp. 373–381.

[2] Legato, L., Monaco, M.F., (2004). Human resources management at a marine container terminal. *European Journal of Operational Research* 156(3), pp. 769–781.

[3] Murty, K.G., Liu, J.,Wan, Y.W., Linn, R., (2005). A decision support system for operations in a container terminal. *Decision Support Systems* 39(3), pp. 309–332.

[4] Rossi, F., Van Beek, P., Walsh, T., (2006). Handbook of Constraint Programming, Elsevier.

[5] Russell, S., Norvig, P., (2004). Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall.

[6] Zampelli, S., Vergados, Y., Van Schaeren, R., Dullaert, W., Raa, B., (2004). The berth allocation and quay crane assignment problem using a CP approach. Proceedings of 19th International Conference: ‘‘CP 2013’’, Uppsala, Sweden, pp 880–896.