ICIC Express Letters
Volume **14**, Number **6**, June **2020**

 ISSN 1881-803X
pp. 577–583

REAL TIME ALGORITHMS FOR SCHEDULING PROBLEM IN UNPLANNED FIRING OPERATION

June-Young Bang[1], Ju-Yong Lee[2] and BongJoo Jeong[3,*]

[1]Department of Industrial and Management Engineering
Sungkyul University
53 Sungkyul University-ro, Manan-gu, Anyang 14097, Korea
jybang@sungkyul.ac.kr

[2]Division of Business Administration & Accounting
Kangwon National University
1 Kangwondaehak-gil, Chuncheon 24341, Korea
jy.lee@kangwon.ac.kr

[3]Department of Industrial and Management Engineering
Hannam University
70 Hannam-ro, Daedeok-gu, Daejeon 34430, Korea
*Corresponding author: jbj@hnu.kr

Received December 2019; accepted March 2020

Abstract. *We focus on the Real Time Fire Scheduling Problem (RFSP), which is the problem of determining the firing sequence to minimize the threatening probability to achieve tactical goals. In this paper, we assume that there are m weapons that can fire to n targets (> m) and that the weapons are already assigned to targets. A single weapon or multiple weapons may fire to a single target, and these operations must begin at the same time. We propose mathematical modeling for RFSP and several heuristic algorithms. Computational experiments are performed on randomly generated test problems and the results show that the proposed algorithm outperforms the commonly adopted launch method in field artillery.*
Keywords: Military, Firing scheduling, Real time algorithm

1. **Introduction.** In this research, we study Real Time Fire Scheduling Problem (RFSP) that determines the order of the firing targets to minimize the threat probability to achieve tactical goals. In this RFSP, we assume that information about location and property of enemy target is already known in advance. Since few researchers have studied FSP, we solve FSP through Multiple Travelling Salesman Problem (MTSP). Note that MTSP and FSP have similarities. Bozoki and Richard [1] introduced MTSP for the first time. However, in the early days, this research was not noticed by other researchers. In the 1990s, the computational power of computers was dramatically improved, and then MTSP began to be researched and actively researched by other researchers. Drozdowski [2] and Lee et al. [3] dealt with a lot of MTSP in their study. In addition, Bozoki and Richard [1], Bianco et al. [4] and Krämer [5] suggested a Branch and Bound algorithms (B&B) for MTSP.

In this study, we considered the Real Time Fire Scheduling Problem (RFSP) for the objective of minimizing threatening rate from enemy, and to destroy enemy targets and minimize threatening probability of enemy, we developed heuristic algorithms and examined the performance of the proposed algorithms with several test simulations.

This paper is organized as follows. In Section 2, detailed situation of the real-time unplanned firing scheduling problem is described. Although we formulated mathematical

DOI: 10.24507/icicel.14.06.577

model for this problem, we omit the equations for the complexity of the problem. In Section 3, we propose a heuristic algorithm that can present a real-time firing scheduling, and we examine the performance of the proposed heuristic algorithm based on actual unplanned fire situation in Section 4. Finally, Section 5 summarizes the contents of this study.

2. **Problem Descriptions.** In this section, we describe the RFSP with several own (friendly) artillery weapons (unit). Each weapon can fire targets that appear in an operational area dedicated to each weapon. Two or more own weapons can be assigned to the operational area where many enemy targets are expected. Also, operation areas could be allocated to overlap so that two or more our weapon can fire targets simultaneously.

The probability of destruction of enemy targets is assumed to decrease in proportion to the elapsed time since the first artillery fire. This assumption is based on the fact that if a friendly artillery fire is initiated, the enemy target will move after recognizing combat situations, or will additionally be equipped with protection. When the position of the moved target is reaffirmed, this probability of failure is reduced to the initial value, and it can be presented as a problem that is closer to reality. The following is a summary of the problem and mathematical model.

1) The operational area of the friendly weapon is given in advance, and the newly identified target in each operational area is handled by the assigned weapon.
2) A large target that cannot be destroyed by a single battalion in your area fires simultaneously with the battalion in charge of the nearby operational area.
3) To switch the weapon fire to a different target, a certain amount of time is required to switch on the fire.
4) The probability of destruction of the enemy target decreases proportionally to the elapsed time since the initial shooting.
5) We do not consider the power loss of allied forces due to enemy fire.

Now, we give notation to express the problem mathematically.

Indices and parameters

i	index for (friendly) weapons ($i = 1, \ldots, m$)
j	index for (enemy) targets ($j = 1, \ldots, n$)
V_j	initial threat of target j
E_j	set of weapons assigned to target j
d_{ij}	required firing duration of weapon i to destroy target j
p_{ij}	initial probability of destruction if weapon i attacks target j at the time of its detection
s_i	setup time of weapon i to change the target aiming
α_j	reduction rate of destruction probability due to delay of firing
$W(j,k)$	set of weapon required to fire target j and k simultaneously
M	big number ($>$ 1,000,000)

Decision variables

x_{jk}	binary decision variable which represents the sequence of target. The value becomes 1 if target j is scheduled before target k, otherwise 0
t_j	start time of fire on target j

The following is the mathematical model for unplanned firing operation problem.

$$\text{Min} \sum_{j=1} V_j \prod_{i \in E_j} \{1 - \max(p_{ij} - t_j \alpha_j, 0)\}$$

Subject to

$$t_k - t_j + M(1 - x_{jk}) \leq \max_{i \in W(j,k)} (d_{ij}) + s_i \quad \forall i \in W(j,k), j, k \tag{1}$$

$$t_j - t_k + M x_{jk} \leq \max_{i \in W(j,k)} (d_{ik}) + s_i \quad \forall i \in W(j,k), j, k \tag{2}$$

$$t_j \geq 0 \quad \forall j \tag{3}$$

$$x_{jk} \in \{0, 1\} \quad \forall j, k \, (j < k) \tag{4}$$

The objective formula of the real-time scheduling problem considered in this study is defined as the sum of the target threat levels and to minimize this value. Note that the objective function is multiplication of probability, and if the number of weapons is more than 1, the problem will have non-linear feature. As the probability of destruction of the target decreases proportionally with time, we want to determine the order of shooting that minimizes the threat of the target to the friendly. Constraints (1) and (2) are disjunctive constraints that express the sequence of the two target pair j and k.

3. **Heuristic Algorithms.** In this section, we suggest heuristic algorithms that determine the order of the firing targets in RFSP. The mathematical model of RFSP can be solved by nonlinear integer programming, which consumes considerable time in small size problems. Due to the nature of the real fire scheduling problem that must be obtained in real time, the firing sequence of the sequence must be calculated within a reasonable time. Therefore, a real time heuristic methodology is needed to obtain the solution easily and quickly, to give order to the weapons and to recalculate the existing firing sequence if a new enemy target appears during the firing operation. If a new enemy target emerges, or if an existing target moves and is re-detected during the firing, the probability of destruction is reset to the initial value p_{ij}. In this study, we proposed a method to obtain the efficient firing sequence based on the dispatching rule and a method to improve the given solution.

The methodology presented in this study can also be used as an upper bound in the planned firing scheduling problem if no new enemy target appears.

3.1. **Dispatching rule based constructive algorithm.** Dispatching rule-based scheduling algorithms have the advantage of being able to quickly derive solutions with simple rules, but the results differ greatly according to the priority rules that determine the firing sequence of the targets. In this study, we propose the following six priority rules. However, if two or more weapons are to be fired at the same enemy target, and if the firing sequence is determined by priority rule, the corresponding weapons shall be involved in the firing at the same time.

1) Target with shortest firing duration first rule (SPT: Shortest Processing Time)
   This priority rule is to fire from a target with a shortest firing duration required to destroy the target. Targets to be fired by one or more weapons are defined as the sum of the firing duration time needed by each weapon. SPT is a rule designed to rapidly reduce the number of targets.
2) Target with the largest threatening first priority rule (LTR: Largest Threatening Rate First)
   Firing the target with the highest threatening rate V_j is one of the most accepted firing priority rules in practice.
3) Target with the largest emergency rate ($V_j \cdot \alpha_j$) first rule (LER: Largest Emergency Rate First)
   We consider the threatening rate V_j and probability of destruction α_j at the same time. In this rule, target with high threatening rate and high mobility has higher priority to be fired.
4) Target with the largest threatening rate over firing duration first rule (LTR/D $\times$ Largest Threatening Rate over Duration first)

LTR/D is a rule that prioritizes targets with a high threat and short shooting time. It is a combination of emergency and SPT rules.

5) Target with following value first (MinC)

$$V_j \prod_{i \in E_j} \{1 - \max(p_{ij} - t_j \alpha_j, 0)\}$$

As a rule to consider the reduction of the threatening probability and the destruction probability by time, the firing duration time t_j is set to the fastest shooting point considering the partial sequence of the targets that have already determined the order of shooting.

6) Target with following value first rule (MinC/D)

$$V_j \prod_{i \in E_j} \{1 - \max(p_{ij} - t_j \alpha_j, 0)\} \Big/ d_j$$

MinC/D is the rule that the target which divides the value contributing to the objective formula by the shooting time is fired first. It is a priority rule proposed with the intention to increase the objective value effectively in the early stage before the shooting.

3.2. **Modified NEH algorithm.** NEH (Nawaz Enscore Ham) algorithm was developed by Nawaz et al. [6] for m-machine flow shop. In the typical NEH, jobs are sorted in order that the sum of the processing time on the machine does not increase in order to get obtaining sequencing priority. Then, a job is selected in the order of sequencing priority, and a feasible schedule is constructed by inserting the selected job at the best position of the current partial sequence until the entire sequence is obtained. In order to apply NEH to the problem of this study, when sorting the jobs (targets), the jobs are sorted in descending order of emergency rate ($V_j \cdot \alpha_j$) instead of the processing time of jobs since probability of destruction of target j (α_j) and initial threatening rate (V_j) are important factors of the urgency. After that, follow the NEH method as it is, but insert target into the partial sequence at the position where the following value is maximized.

$$\sum_{j=1} V_j \prod_{i \in E_j} \{1 - \max(p_{ij} - t_j \alpha_j, 0)\}$$

3.3. **Improvement algorithm.** First, an initial solution is obtained by algorithm in Section 3.1 or Section 3.2 and set the initial solution as a current solution. Next, we generate neighborhood solutions from the current solution using the local search method. Then, the solution with the best objective value among the generated solutions is selected as the new current solution and this process is repeated until the entire sequence is obtained. Four methods, Insertion (IS), Interchange (IC), Insertion-Interchange (ISIC), and Interchange-Insertion (ICIS) were developed and applied to generating the neighborhood solutions. In IS algorithm, we select a job from current solution and insert the job into each position (after the selected job) to generate neighborhood solutions. On the other hand, IC algorithm generated neighborhood solutions by interchanging two selected jobs from the current solution. The detailed steps of the IS and IC algorithms are as follows.

IS algorithm

Step 1. Let the given initial solution Λ_0 be the current solution Λ. Set index $p = 1$ and $q = 2$ respectively.

Step 2. Select p-th job (target) of Λ and insert the job between q-th job and $(q+1)$-th job. Let the new generated solution be the Λ'.

Step 3. Compute solution value of $\Lambda' = \sum_j V_j \prod_{i \in E_j} \{1 - \max(p_{ij} - t_j \alpha_j, 0)\}$.

Step 4. If solution value of Λ' is smaller than those of Λ, set $\Lambda \leftarrow \Lambda'$ and go to step 2; otherwise go to step 5.
Step 5. If $q = |\Lambda|$, go to step 6; otherwise, set $q = q + 1$ and go to step 2.
Step 6. If $p + 1 = |\Lambda|$, stop; otherwise, set $p = p + 1$, $q = p + 1$ and go to step 2.

IC algorithm

Step 1. Let the given initial solution Λ_0 be the current solution Λ. Set index $p = 1$ and $q = 2$ respectively.
Step 2. Select p-th job (target), q-th job and interchange two jobs with each other. Let the new generated solution be the Λ'.
Step 3. Compute solution value of $\Lambda' = \sum_j V_j \prod_{i \in E_j} \{1 - \max(p_{ij} - t_j \alpha_j, 0)\}$.
Step 4. If solution value of Λ' is smaller than those of Λ, set $\Lambda \leftarrow \Lambda'$ and go to step 2; otherwise go to step 5.
Step 5. If $q = |\Lambda|$, go to step 6; otherwise, set $q = q + 1$ and go to step 2.
Step 6. If $p + 1 = |\Lambda|$, stop; otherwise, set $p = p + 1$, $q = p + 1$ and go to step 2.

ISIC and ICIS are improvement algorithms that combine IS and IC. ISIC obtains the first solution by implementing IS and improves the first solution by using IC. On the other hand, ICIS gets the final solution by implementing IC first and IS next.

4. **Computational Test.** First, for the small size problem, the optimal solutions were obtained by solving the mathematical model given in Section 2. To solve the mathematical model, we used ILOG CPLEX 11.0. The mathematical model (although the detailed description is omitted in this article) is non-linear programming with exponential objective function. CPLEX can solve mathematical model with quadratic objective function and linear constraints. Therefore, for the small size problem, optimal solutions were calculated by limiting the number of weapons to 2.

For this test, we generated 33 instances randomly, all combinations of the 1 level (two weapons) of the number of weapons, 11 levels ($10 \sim 20$) of the number of targets, and three levels (20%, 50% and 80%) for ratio of the number of weapons to the number of targets. The results are shown in Table 1. The table gives average CPU time and the number of instances that have not been solved within time limits. We set time limit as 3,600 seconds. As can be seen from the table, it can be seen that it takes more than 3,600 seconds to derive the optimal solution even for the small size problem.

To compare performances of heuristic algorithms, we generated 135 instances randomly, all combinations of the 3 levels (3, 6, 9) of the number of weapons, 3 levels (22, 24, 26) of the number of targets, and three levels (20%, 50% and 80%), and three levels for the percentage of single-weapon targets among all targets (20%, 50%, and 80%).

Table 2 summarizes the relative performances (RDI; Relative Deviation Index) of dispatching rules and constructive algorithm proposed in this study. From the results, we can know MinC/D and MNEH outperformed all the other algorithms. Although MinC/D is the simple dispatching rule and MNEH is the improvement algorithm, MinC/D has similar performance to that of MNEH. MNEH showed slight better performance than MinC/D.

To compare performances of four improvement algorithms, MNEH's result solution is used as an initial solution to improvement algorithms. The results are summarized in Table 3. ICIS outperformed all the other algorithms.

Table 4 gives RDI and the number of instances which the algorithm found the best solutions for overall cases (NBS). From the results, we can see that ICIS shows the best performance. It is very hard to find the exact reason why the individual heuristic algorithm shows such a result. Therefore, we can explain why ICIS outperforms ISIC intuitively as follows. IC algorithm improves the given job sequence by pair-wise interchange of two

TABLE 1. Results of the CPLEX

		CPLEX	
W[a]	T[b]	CPUT[c]	NPNS[d]
2	10	0.958	0
	11	2.068	0
	12	90.323	0
	13	225.078	0
	14	140.672	0
	15	650.036	0
	16	1,305.934	1
	17	2,492.024	2
	18	2,545.585	2
	19	3,600.033	3
	20	3,600.417	3

[a,b]number of weapons/targets

[c]average CPU time (in seconds) required to solve a problem

[d]number of problems (among 15 problems) that were not solved to the optimality in 3600 seconds

TABLE 2. Performance of the heuristic algorithms

W[a]	T[b]	LTR	LER	LTR/D	MinC	MinC/D	MNEH
	22	0.876[c](0.137)[d]	0.699(0.177)	0.823(0.156)	0.556(0.249)	0.596(0.230)	0.450(0.144)
3	23	0.775(0.143)	0.834(0.193)	0.772(0.146)	0.514(0.246)	0.649(0.221)	0.425(0.169)
	24	0.834(0.136)	0.743(0.144)	0.891(0.109)	0.543(0.249)	0.647(0.223)	0.537(0.215)
	22	0.806(0.177)	0.712(0.197)	0.658(0.150)	0.520(0.279)	0.300(0.122)	0.298(0.201)
6	23	0.778(0.196)	0.759(0.174)	0.619(0.154)	0.452(0.194)	0.298(0.133)	0.222(0.126)
	24	0.857(0.181)	0.786(0.167)	0.711(0.145)	0.458(0.269)	0.345(0.166)	0.364(0.267)
	22	0.658(0.268)	0.720(0.203)	0.725(0.258)	0.687(0.234)	0.435(0.239)	0.474(0.239)
9	23	0.642(0.275)	0.684(0.275)	0.733(0.208)	0.673(0.265)	0.503(0.174)	0.436(0.260)
	24	0.596(0.264)	0.760(0.215)	0.655(0.283)	0.587(0.240)	0.459(0.212)	0.464(0.310)

[a,b]see footnote of Table 1

[c]average RDI

[d]standard deviation of RDI

TABLE 3. Performance of the improvement algorithms

W[a]	T[b]	IS	IC	ISIC	ICIS
	22	0.093[c](0.065)[d]	0.103(0.057)	0.075(0.063)	0.020(0.026)
3	23	0.069(0.045)	0.077(0.053)	0.080(0.060)	0.021(0.032)
	24	0.104(0.051)	0.090(0.065)	0.061(0.052)	0.031(0.042)
	22	0.098(0.048)	0.057(0.046)	0.061(0.049)	0.017(0.035)
6	23	0.106(0.038)	0.064(0.043)	0.051(0.055)	0.016(0.023)
	24	0.098(0.081)	0.081(0.060)	0.076(0.052)	0.031(0.038)
	22	0.096(0.088)	0.071(0.066)	0.078(0.074)	0.022(0.037)
9	23	0.119(0.105)	0.093(0.061)	0.091(0.101)	0.051(0.061)
	24	0.123(0.114)	0.128(0.121)	0.088(0.069)	0.037(0.051)

[a,b,c,d]see footnote of Table 2

Table 4. Performance of algorithms for overall cases

Methods	Relative Deviation Index		NBS
	Mean	Standard deviation	
LER	0.744	0.196	0
LER/D	0.732	0.199	0
MinC	0.555	0.253	4
MinC/D	0.470	0.231	0
MNEH	0.408	0.234	6
IS	0.101	0.074	18
IC	0.085	0.068	21
ISIC	0.073	0.065	21
ICIS	0.027	0.040	64

jobs, while IS algorithm improves the given job sequence by inserting a job to other position. IC seems to generate the sequence change more frequent than IS, and IS seems to find local optimum easily. Therefore, applying IS after IC (= algorithm ICIS) can have more chance to find better solution than ISIC.

5. **Conclusions.** In this study, we propose the Real Time Fire Scheduling Problem (RFSP) for the objective of minimizing threatening rate from enemy. To efficiently operate our weapons and respond to enemy movements quickly, we develop mathematical model and heuristic algorithms for solving RSFP. For small size problems, optimal solutions were obtained by solving mathematical model we proposed with CPLEX. Also, we compare the relative performance of heuristic algorithms by solving randomly generated instances.

The dispatching rule that the target with short firing duration time and large threatening rate is fired first gave the best performance among dispatching rules. In addition, the improvement algorithms we developed gave near optimal solutions. All algorithms we proposed in this study could solve our test instances within 0.001 seconds. Therefore, these heuristic algorithms can be useful to make firing schedules in real time and minimize threats from enemies.

Acknowledgement. This work was supported by 2020 Hannam University Research Fund.

REFERENCES

[1] G. Bozoki and J. P. Richard, A branch and bound algorithm for the continuous-process job-shop scheduling problems, *AIIE Transactions*, vol.2, pp.246-252, 1970.

[2] M. Drozdowski, Scheduling multiprocessor tasks – An overview, *European Journal of Operational Research*, vol.94, pp.215-230, 1996.

[3] C.-Y. Lee, L. Lei and M. Pinedo, Current trends in deterministic scheduling, *Annals of Operations Research*, vol.70, pp.1-41, 1997.

[4] L. Bianco, P. Dell'Olmo and M. G. Speranza, Nonpreemptive scheduling of independent tasks with prespecified processor allocations, *Naval Research Logistics*, vol.41, pp.959-971, 1994.

[5] A. Krämer, Branch and bound methods for scheduling problems with multiprocessor tasks on dedicated processors, *OR Spektrum*, vol.19, pp.219-227, 1997.

[6] M. Nawaz, E. E. Enscore and I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, vol.11, pp.91-95, 1983.