ADVANCED ALGORITHM FOR OPTIMAL SENSOR-TARGET AND WEAPON-TARGET PAIRINGS IN DYNAMIC COLLABORATIVE ENGAGEMENT

Z. R. Bogdanowicz*, N. P. Coleman
Armament Research, Development and Engineering Center (ARDEC)
Picatinny Arsenal, New Jersey 07806

ABSTRACT

In this paper we introduce a new algorithm for assigning sensors and weapons to targets in the dynamic environment, where sensors, weapons, and targets are allowed to move freely (but predictably) over a certain region of a battlefield. In addition, we determine when such an assignment should be executed. This new algorithm for **S**ensors/**W**eapons **A**nd **T**argets pairings is named *SWAT.* In *SWAT* the optimization component *Swt_opt* is derived from the well-known auction algorithm and produces an optimal solution.

1. INTRODUCTION

Sensor-target and weapon-target (or briefly sensor/weapon-target) pairings in general are difficult optimization problems. These problems, however, are critical to the outcome of modern battles in net-centric warfare, which profoundly depends on the intelligent usage of all available sensors and weapons maximizing their effectiveness. The main difficulty of assigning sensors and weapons to a set of identified targets stems from the potential dependence of weapons on sensors (Bogdanowicz and Coleman, 2007). At the same time, smart weapons (i.e., weapons that depend on the information obtained from sensors) are expected to play an increasingly more dominant role in the net-centric warfare of the future.

In the dynamic environment, where sensors, weapons, and targets are mobile, this problem becomes even more difficult due to uncertainties involved in predicting where all entities will reside at a particular instance of time in the future. It also raises a problem of scheduling the engagement of the targets. However, the sensor/weapon-target pairing problem can be simplified to the well-known assignment optimization problem in mathematics (Bertsekas, 1990, 1992a, 1992b; Castanon, 1993; Galil, 1986; Hopcroft and Karp,1973; Micali and Vazirani, 1980). We have shown that for practical sensor/weapon-target pairings in a static environment (i.e., without mobile sensors, weapons or targets), an algorithm based on the well-known auction algorithm should be considered a preferred choice (Bogdanowicz and Coleman, 2007).

In this paper we introduce a new algorithm for assigning sensors/weapons to targets in the dynamic environment, where sensors, weapons, and targets are allowed to move freely (but predictably) over a certain region of a battlefield. This new algorithm for **S**ensors/**W**eapons **A**nd **T**argets pairings is named *SWAT* and its *exact* optimization component *Swt_opt* is derived from the auction algorithm (Bertsekas, 1990 1992a, 1992b; Bogdanowicz et al., 2004a, 2004b, 2005, 2007; Castanon, 1993).

SWAT consists of three main components, which are described in Sections 4 through 6. The main focus in this work is on optimization component *Swt_opt* that we describe in Section 6.

2. ASSUMPTIONS

We assume that all the sensors *S*, weapons *W*, and targets *R* are given along with their respective parameters and initial locations. The planning time horizon into the future *T* is also given. So, *T* represents the time interval in which the engagement of sensors/weapons with targets should take place. Based on the movement characteristics of the units (i.e., sensors, weapons, and targets), we assume that the preprocessor called *Predictor* accurately predicts the location of these units at time t, $T \geq t > 0$. Furthermore, for a given sensor-weapon-target triplet (i,j,k) we assume that the preprocessor called *Evaluator* establishes the precise benefit $b_{i,j,k}$ of assigning a given sensor-weapon pair (i,j) to target k., which represents a rational number. Based on this combined input, *SWAT* determines an optimal assignment of sensors/weapons to targets (i.e., an assignment that maximizes the total benefit $\sum_{i,j,k} b_{i,j,k}$) along with the proposed engagement time t, where $T \geq t > 0$. To do this, *SWAT* utilizes the optimizer called *Swt_opt* defined in Section 6.3.

To perform the optimization we assume that time progresses in discrete and equal steps, where δ denotes the time interval corresponding to a step under consideration. If a sensor/weapon can be simultaneously assigned into many targets, or a target can be simultaneously targeted by many sensors/weapons, then the transformation based on the following four rules applies.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **DEC 2008**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Advanced Algorithm For Optimal Sensor-Target And Weapon-Target Pairings In Dynamic Collaborative Engagement**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Armament Research, Development and Engineering Center (ARDEC) Picatinny Arsenal, New Jersey 07806**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**		
13. SUPPLEMENTARY NOTES **See also ADM002187. Proceedings of the Army Science Conference (26th) Held in Orlando, Florida on 1-4 December 2008, The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**8**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Rule 1: If sensor i can be assigned to m targets then such a sensor can be decomposed into m pseudo-sensors $s_{i,1}, s_{i,2}, \ldots, s_{i,m}$ such that each pseudo-sensor $s_{i,j}$, $m \geq j \geq 1$, can be assigned to at most one target. In addition, if the benefit of assigning sensor/weapon (i,j) into target k equals $b_{i,j,k}$ then the benefit of assigning a corresponding pseudo-sensor remains $b_{i,j,k}$.

Rule 2: If weapon i can be assigned to m targets then such a weapon can be decomposed into m pseudo-weapons $w_{i,1}, w_{i,2}, \ldots, w_{i,m}$ such that each pseudo-weapon $w_{i,j}$, $m \geq j \geq 1$, can be assigned to at most one target. In addition, if the benefit of assigning sensor/weapon (i,j) into target k equals $b_{i,j,k}$ then the benefit of assigning a corresponding pseudo-weapon remains $b_{i,j,k}$.

Rule 3: If target i can be assigned by up to to m sensors then such a target can be decomposed into m pseudo-targets $r_{i,1}, r_{i,2}, \ldots, r_{i,m}$ such that each pseudo-target $r_{i,j}$, $m \geq j \geq 1$, can be assigned by at most one sensor. In addition, if the benefit of assigning sensor/weapon (i,j) into target k equals $b_{i,j,k}$ then the benefit of assigning a corresponding pseudo-target remains $b_{i,j,k}$.

Rule 4: If target i can be targeted by up to to m weapons then such a target can be decomposed into m pseudo-targets $r_{i,1}, r_{i,2}, \ldots, r_{i,m}$ such that each pseudo-target $r_{i,j}$, $m \geq j \geq 1$, can be targeted by at most one weapon. In addition, if the benefit of assigning sensor/weapon (i,j) into target k equals $b_{i,j,k}$ then the benefit of assigning a corresponding pseudo-target remains $b_{i,j,k}$.

Furthermore, if the number of sensors is different from the number of weapons (or targets) then it can be translated to a symmetric input (for *SWAT* and for *Swt-opt*) in a straightforward way by augmenting it with pseudo-sensors or pseudo-weapons (or pseudo-targets) with benefit values set to zero. So, without loss of generality we can assume that *SWAT* and its optimizer *Swt_opt* are supplied with a symmetric input, where every sensor/weapon can be assigned exactly once to a target and vice versa. Let n be the number of sensors, which equals the number of weapons, and which equals the number of targets. Let b_{max} be the maximum benefit of assigning a sensor/weapon pair to a target. Because every benefit $b_{i,j,k}$ is a rational number then there exists an integer that converts $b_{i,j,k}$ to integer $b'_{i,j,k}$ through multiplication. Let C be the smallest such integer, i.e., $b'_{i,j,k} = b_{i,j,k}C$, where $b'_{i,j,k}$ is integer.

3. SOFTWARE ARCHITECTURE

An architecture that supports *SWAT* consists of the following key active components:

(a) Translator1,

(b) Predictor,

(c) Evaluator,

(d) Optimizer (*Swt_opt* Algorithm),

(e) Translator2,

where *Translator1* represents preprocessor, while *Translator2* represents postprocessors of the assignment optimization problem realized by *Predictor, Evaluator,* and *Optimizer*.

In particular, *Translator1* takes available weapons and sensors, and given targets, and based on its internal Knowledge Base (*KB*) it generates intermediate results that describe the status of weapons/sensors/targets at the initial instance of time. *Translator2* takes the outcome of optimization produced by *Optimizer* and reports the optimal weapon/sensor-target assignment along with the proposed time t when such assignment should be executed, where $t_{max} \geq t \geq t_{min}$. More detailed functionality of these components is described in Sections 4-6, and an architecture based on them is shown in Fig. 1.

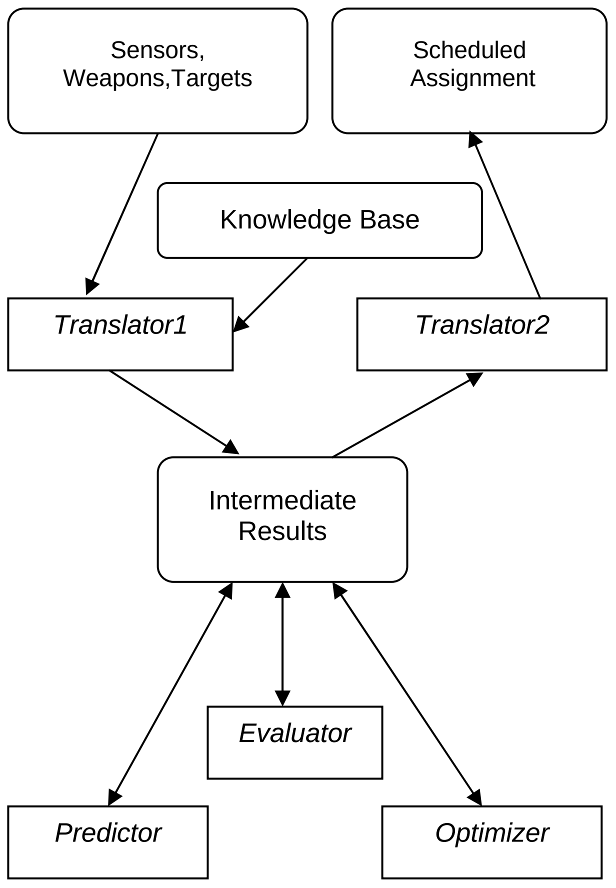

Fig. 1 – Architecture for the dynamic sensor/weapon-target pairing

4. LOCATIONS PREDICTION IN *SWAT*

One of the first steps in our *SWAT* algorithm relies on predicting the locations of weapons, sensors, and targets at a future time. Let t_0 be an initial time under consideration for weapons/sensors/targets, and let T be a planning time horizon for them into the future. Let t_0, t_1, …, t_r be the time instances such that the following relations are satisfied.

$$t_1 - t_0 = t_{min}\ , \tag{1}$$

$$t_r = T\ , \tag{2}$$

$$t_{i+1} - t_i = \delta\ , \tag{3}$$

$$t_{min} > \delta, \tag{4}$$

for $r > i \geq 1$.

For given i, $r > i \geq 1$, *Predictor* in *SWAT* determines the locations of sensors and weapons at time $t_i - \beta$, where $\beta \geq 0$ and $t_1 > \beta$, (Fig. 1). In addition, *Predictor* determines the locations of targets at time t_i. For given i let $L(S,W,R)$ be the predicted list of locations of sensors/weapons at time $t_i - \beta$, and targets at time t_i. So, we can say that *Predictor* generates $L(S,W,R)$, which we denote by $Predictor(S,W,R,KB,t_i) \rightarrow L(S,W,R)$. That is, *Predictor* generates $L(S,W,R)$ based on the given sensors S, weapons W, targets R, knowledge base KB, and a time instance t_i (example in Fig. 2).

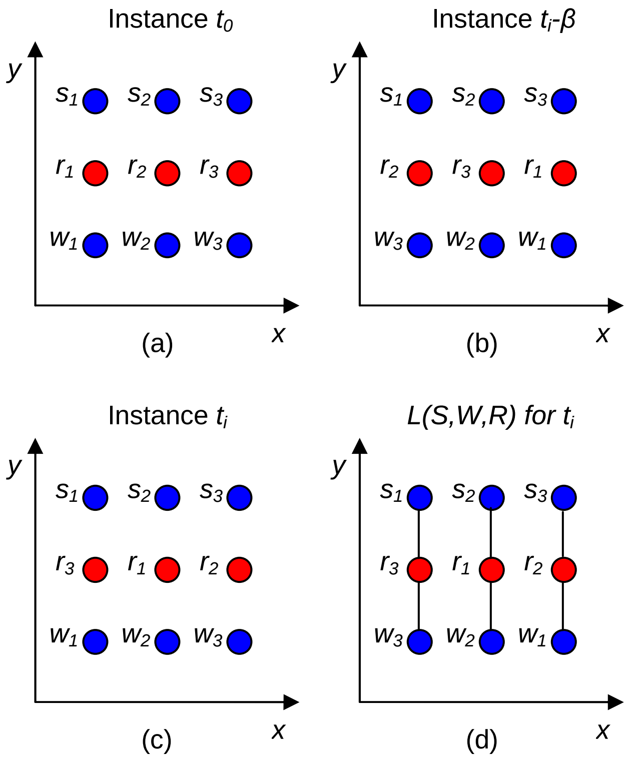

Fig. 2 - An example of establishing *L(S,W,R)* and corresponding preferred benefits

5. BENEFIT EVALUATION IN *SWAT*

Let $r \geq k \geq 1$. The benefits of assigning weapons and sensors to targets will be affected by the locations of weapons/sensors at time $t_k - \beta$, and targets at time t_k, and they will change for different values of t_k. For example, consider three sensors, three weapons, and three targets from Fig. 2. We consider here a scenario in two dimensions (x,y) only, where sensors s_1, s_2, s_3 are assumed stationary, and weapons/targets w_1, w_2, w_3, r_1, r_2, r_3 are moving only in x direction. Fig. 2a represents an initial (at time t_0) set of locations for S,W,R. Fig. 2c illustrates *Predictor*'s set of locations of S,W,R at time $t_i > t_0$, and Fig. 2b illustrates *Predictor*'s set of corresponding locations of S,W,R at time t_i-β. Based on the S,W locations in Fig. 2b and R locations in Fig. 2c *Predictor* generates list of locations $L(S,W,R)$ illustrated in Fig. 2d. Based on KB and $L(S,W,R)$ from Fig. 2d *Evaluator* determines the benefits for B, i.e., $Evaluator(L(S,W,R),KB) \rightarrow B$. In addition, if the best benefits for the assignments are determined exclusively by the shortest distances between sensors/weapons and targets, then *Evaluator* determines the best benefits that correspond to the edges in Fig.2d.

In this work we assume that $Evaluator(L(S,W,R),KB)$ generates the benefit values, which are the rational numbers in B (recall Section 2). This assumption should not diminish the practicality of *SWAT*.

6. OPTIMIZATION IN *SWAT*

The purpose of the optimizer is to take the given weapons/sensors/targets along with a corresponding benefit matrix describing the benefits of assigning weapon/sensor pairs to targets, and to find an optimal assignment (i.e., an assignment that maximizes the total benefit of assigning weapon/sensor pairs to the targets).

6.1 Mathematical Formulation

Consider $n^2 \times n$ benefit matrix A, where each row corresponds to a unique combination of sensor-weapon pair, and each column corresponds to a unique target (Fig. 3).

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{n^2,1} & a_{n^2,2} & \dots & a_{n^2,n} \end{pmatrix}$$

Fig.3 – Benefit matrix *A*

In addition, each element $a_{i,j}$ in A is a nonnegative rational number, which represents a benefit of assigning row i to column j, and consequently a benefit of assigning i'th sensor-weapon pair to j'th target. We can now arrange our benefit matrix as follows. Let $a_{i',j'} = b^k_{i,j'}$ where $i' = (i-1)n+j$ and $j' = k$. Then we have

$$B = \begin{pmatrix} b^1_{1,1} & b^2_{1,1} & \dots & \dots & b^n_{1,1} \\ \dots & \dots & \dots. & \dots & \dots \\ b^1_{1,n} & b^2_{1,n} & \dots & \dots & b^n_{1,n} \\ b^1_{2,1} & b^2_{2,1} & \dots & \dots & b^n_{2,1} \\ \dots & \dots. & \dots & \dots & \dots \\ b^1_{2,n} & b^2_{2,n} & \dots & \dots & b^n_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ b^1_{n,1} & b^2_{n,2} & \dots & \dots & b^n_{n,1} \\ \dots & \dots & \dots & \dots & \dots \\ b^1_{n,n} & b^2_{n,n} & \dots & \dots & b^n_{n,n} \end{pmatrix}$$

Fig.4 – Benefit matrix *B* with rational benefits

Let $C = B \times X$, where $X = [x_{i,j}]_{n \times n^2}$. If sensor *i* and weapon *j* are assigned to target *k* then $x_{i',j'} = 1$, where $i' = k$ and $j' = (i-1)n + j$. Otherwise $x_{i',j'} = 0$. The problem is to find matrix *X* that maximizes *Tr(C)* and which satisfies the following rules:

(1) Each sensor is assigned exactly once.
(2) Each weapon is assigned exactly once.
(3) Exactly one sensor is assigned to a target.
(4) Exactly one weapon is assigned to a target.

So, we can state the optimization problem as follows:

$$\max \sum_{i=1}^{n^2} c_{i,i} \qquad (5)$$

subject to:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} b^k_{i,j} = 1 \qquad (6)$$

$$\sum_{i=1}^{n} \sum_{k=1}^{n} b^k_{i,j} = 1 \qquad (7)$$

$$\sum_{j=1}^{n} \sum_{k=1}^{n} b^k_{i,j} = 1 \qquad (8)$$

6.2 Why Exact Algorithm

To solve an assignment optimization problem in *SWAT* focused on sensors, weapons and targets, we might first consider if it makes sense to use an exact optimization algorithm vs. an approximate heuristic. Since the number of sensors, weapons and targets in realistic battlefield scenarios should run up to the hundreds, we estimate that an exact optimization algorithm should perform time-efficiently (i.e., in order of tens of seconds). Hence, our attention should be focused on finding an exact optimization algorithm for the assignment problem for this range.

One of the better-documented exact optimization algorithms for an assignment problem is the auction algorithm (Bertsekas, 1990, 1992b; Castanon, 1993). For the above input size an algorithm derived from the auction algorithms should run in order of seconds, as it has been shown in (Bogdanowicz and Coleman, 2007). Furthermore, the bidding and assignment phases of such derived algorithm are highly parallelizable (Bertsekas, 1992b), which makes it scalable. That is, the bidding and the assignment can be carried out for all sensors, weapons and targets simultaneously, which could extend the range of input to thousands of sensors, weapons and targets and beyond.

Finally, the nature of sensor-target and weapon-target pairings should allow a benefit scaling, which could produce matrix *B* with all integral benefits b_{ij} (Section 6.3). This in turn could further improve the performance of auction-based algorithms. In fact, this is the key for an efficient implementation of any auction algorithm.

For much larger input sizes one could also consider variants of the interior point algorithm (Adler et al., 1989; Todd, 1992). However, such inputs would be rather rare for sensor/weapon-target pairing in the real world. In addition, one could still address this class of problems with the parallel implementation of an auction-based algorithm.

6.3 *Swt_opt* Algorithm

Assume that the weapons/sensors/targets are given along with matrix *X* and the corresponding arranged benefit matrix B from Fig. 4. The benefits $b_{i,j}^{k}$ in matrix B are all nonnegative rational numbers, and elements $x_{i,j}$ in matrix X are all equal zero.

In B each row corresponds to a unique sensor/weapon combination and each column corresponds to a target. That is, b_{ij} represents a benefit of assigning i'th distinct sensor/weapon combination to target j. Such a translation requires $O(n^3)$ operations. So B has n^2 rows and n columns, and represents the only input to our *Swt_opt* algorithm, which executes predominantly as a standard auction algorithm. It assigns n out of n^2 rows to n columns in B with the following exceptions.

a) If sensor s_i and weapon w_j are currently assigned to target r_k, based on the best bid (i.e., $s_i w_j \rightarrow r_k$), then $s_{i'} w_{j'} \rightarrow r_{k'}$ is not considered for assignment to target k', $k' \neq k$ if either $i=i'$ or $j=j'$. This assures that a sensor/weapon or target is not assigned more than once in an optimal solution.

b) Based on the best $s_i w_j \rightarrow r_k$ assignment a second best assignment for target k', $k' \neq k$ is determined by $s_{i'} w_{j'} \rightarrow r_{k'}$, where $i'=i$. This allows calculation of a penalty cost for auction bids.

Swt_opt algorithm for sensor/weapon-target pairings can be presented in 12 steps as follows.

```
Step 1:  Initialize $\varepsilon < \frac{1}{n}$, $m=1$, $v_1 = \ldots = v_n = 0$, and
         $w_1 = \ldots = w_n = 0$.

Step 2:  Transform $B$; $B \rightarrow B' = [b'_{i,j}]_{n \times n}$, where $b'_{i,j}$ is
         integer for $n \geq i \geq 1$, $n \geq j \geq 1$. It follows by
         $B := B'$.

Step 3:  If there exists unassigned sensor $i_m$, so
         $\sum_{i'=1}^{n} \sum_{j'=n(i_m-1)+1}^{n(i_m-1)+n} x_{i',j'} = 0$ then $i = i_m$.
         Otherwise, STOP.

Step 4:  Select $j_1$, $k_1$ such that
         $\min(b_{i,j_1}^{k_1} - v_{k_1}, b_{i,j_1}^{k_1} - w_{j_1}) =$
         $\max_{j,k} (\min(b_{i,j}^{k} - v_k, b_{i,j}^{k} - w_j))$.

Step 5:  Select $j_2$, $k_2$ such that
         $\min(b_{i,j_2}^{k_2} - v_{k_2}, b_{i,j_2}^{k_2} - w_{j_2}) =$
         $\max_{j \neq j_1, k \neq k_1} (\min(b_{i,j}^{k} - v_k, b_{i,j}^{k} - w_j))$.

Step 6:  Set $\varphi := \min(b_{i,j_1}^{k_1} - v_{k_1}, b_{i,j_1}^{k_1} - w_{j_1})$ -
         $\min(b_{i,j_2}^{k_2} - v_{k_2}, b_{i,j_2}^{k_2} - w_{j_2})$.

Step 7:  Update $v_{k_1} := \max(v_{k_1}, w_{j_1}) + \varphi + \varepsilon$.

Step 8:  Update $w_{j_1} := v_{k_1}$.

Step 9:  If $x_{i',j'} = 1$ for $i' = k_1$, $j' \neq j_1$ (mod $n$) then
         $x_{i',j'} := 0$.

Step 10: If $x_{i',j'} = 1$ for $i' \neq k_1$, $j' \equiv j_1$ (mod $n$) then
         $x_{i',j'} := 0$.

Step 11: If $x_{i',j'} = 1$ for $i' = k_1$, $j' \equiv j_1$ (mod $n$) then
         $x_{i',j'} := 0$.

Step 12: Set $x_{k_1,(i-1)n+j_1} := 1$, $m := m+1$ and return to
         Step 3.
```

Steps 1-2 in *Swt_opt* represent initialization steps. A minimum bidding increment parameter ε is set to $\varepsilon<1/n$ (for example ε can be set to *0.0099* if $n = 100$). This set-up assures that any complete assignment generated by our algorithm is optimal (Bogdanowicz et al., 2005). In Step 2 arranged benefit matrix B is scaled up by multiplying it by scalar C, so every element $b'_{i,j}$ in the transformed matrix B' becomes an integer. This scaling is possible because elements $b_{i,j}$ in B are the rational numbers. It is an essential step for making *Swt_opt* time-efficient.

In *Swt_opt* a single iteration is defined by Steps 3 through 12. In particular, i'th bid corresponds to a bid in the auction algorithms and is executed in *Swt_opt* by sensor i on one of n targets, and having n weapons to its disposal. In Step 3 it is determined if there exists an unassigned sensor in m'th iteration. If such a sensor i_m exists then it is considered for assignment in iteration m. Selection of such a sensor would depend on specific implementation of *Swt_opt*. For example, a lowest indexed unassigned sensor can be chosen. In Step 4 the best triplet (i,j_1,k_1) (i.e., triplet identifying sensor/weapon/target) for assignment is determined for given sensor $i = i_m$. Similarly, in Step 5 the second best triplet (i,j_2,k_2) for assignment is determined for given sensor i. Then in Step 6 a penalty φ is calculated, which expresses the difference between the best assignment and the second best assignment of the current sensor i to a target. This penalty is added in Steps 7, 8 to variables v_{k_1}, w_{j_1} associated with a target and weapon being assigned in the current iteration. It follows by resetting the assignment flags $x_{i',j'}$ in Steps 9 through 11 for the weapons/targets being unassigned in the current iteration. Finally, in Step 12 an assignment flag $x_{k_1,(i-1)n+j_1}$ is set that corresponds to a sensor/weapon/target pairing being assigned in the current iteration. The iterations in *Swt_opt* continue as long as not all sensors are assigned.

Consider now the worst-case running time complexity of accomplishing an optimal assignment. The assignment problem can be modeled with a complete bipartite graph $G=(V,E)$, where the number of vertices $|V(G)|=n^2+n$ and the number of edges $|E(G)|=n^3$ in G. Let $a_{i,j}$ be a benefit of assigning vertex v_i to vertex v_j if a corresponding edge (i,j) exists in G. Let $b_{max}=\max_{(i,j)\in E(G)} a_{ij}$. Because B consists of the rational numbers only then there exists an integer that converts B to B', where B' consists of only integers. Let C be the smallest such integer. Without loss of generality consider B' as an input to *Swt_opt*, which means that integer Cb_{max} represents the largest benefit in B'. The total number of iterations in which a target receives a bid is no more than $(Cb_{max}+\varepsilon)/\varepsilon$. In addition, each iteration of *Swt_opt* involves a bid by a pair of a single sensor along with the best available (unassigned) weapon. So, the total number of iterations is no more than n times $(Cb_{max}+\varepsilon)/\varepsilon$, and since $\varepsilon < 1/n$ and every bid requires $O(n^2)$ operations, the worst running time of the algorithm is

$$O(n^4 Cb_{max}). \tag{9}$$

To illustrate the optimization scenario for *Swt_opt* consider an arranged benefit matrix B from Fig. 4 for $n=3$ and with the following values:

$$B^T = \begin{pmatrix} 25 & 29 & 29 & 56 & 73 & 77 & 11 & 7 & 3 \\ 27 & 33 & 31 & 69 & 72 & 73 & 11 & 9 & 57 \\ 31 & 16 & 23 & 61 & 67 & 71 & 13 & 6 & 21 \end{pmatrix}.$$

Sensor-weapon pair s_iw_j corresponds to column $3(i-1)+j$, and target r_k corresponds to row k in B^T above, where $n \ge i \ge 1$, $n \ge j \ge 1$, $n \ge k \ge 1$. Note, scaling (i.e., Step 2 in *Swt_opt*) is not needed here. The sequence of the assignments executed by *Swt_opt* based on this instance of B is as follows:

Iteration 1:	$s_1\, w_2 \rightarrow r_2$,
Iteration 2:	$s_2\, w_3 \rightarrow r_1$,
Iteration 3:	$s_3\, w_3 \rightarrow r_2$,
Iteration 4:	$s_1\, w_1 \rightarrow r_3$,
Iteration 5:	$s_2\, w_2 \rightarrow r_1$.

In *Swt_opt* i'th iteration corresponds to columns $3(i-1)(mod\ 9)+1$, $3(i-1)(mod\ 9)+2$, and $3(i-1)(mod\ 9)+3$ in B^T. The final optimal assignment generated by *Swt_opt* in this case can be represented by incomplete bipartite graph in Fig. 5, where left bipartition (i.e., vertices s_iw_j) corresponds to rows and right bipartition (i.e., vertices r_k) corresponds to columns in B.

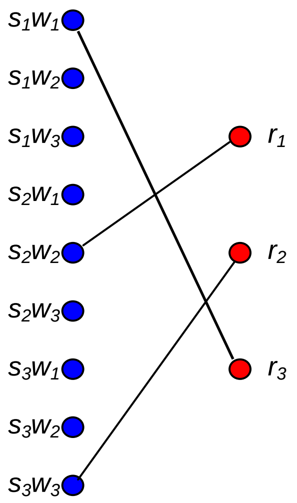

Fig. 5- Bipartite incomplete graph representing sensor/weapon-target final assignment

7. *SWAT* ALGORITM

Based on *Predictor, Evaluator* and *Swt_opt* components described in the previous sections, we can now present the *SWAT* algorithm for the optimal assignment of sensors/weapons to targets. The input to *SWAT* consists of sensors S, weapons W, targets R, and a Knowledge Base KB. We also assume that parameters t_{min}, T, and δ defined in Section 4 are given, they satisfy (1-4), and they are included in KB. In addition, as we discussed in previous sections, $|S| = |W| = |R| = n$. Let $L(S,W,R)$ be the list of locations of sensors, weapons and targets at a time corresponding to an intended hypothetical engagement. *SWAT* can be executed in 7 steps as follows.

```
Step 1: Initialize $C_{opt} := 0$, $t := t_{min}$ and
        $X := X_{opt} := 0$.

Step 2: If $t \le T$ then execute Steps 3 through 7.
        Otherwise, $Report(t_{opt},X_{opt},KB)$ and STOP.

Step 3: $L(S,W,R) := Predictor(S,W,R,KB,t)$.

Step 4: $B := Evaluator(L(S,W,R),KB)$.

Step 5: $X := Swt\_opt(B,X)$.

Step 6: If $\max \sum_{i=1}^{n^2} c_{i,i} > C_{opt}$ then
        $t_{opt} := t$, $X_{opt} := X$; $C_{opt} := \max \sum_{i=1}^{n^2} c_{i,i}$ .

Step 7: Set $t := t + \delta$, $X :=$ 0 and return to Step 2.
```

SWAT starts with initialization in Step 1. The main iterations are defined by Steps 2 through 7, where Step 2 assures that there are at most T/δ such iterations. In Step 3 *Predictor* generates a list of locations for *S, W, T*; i.e., $Predictor(S,W,R,KB,t) \rightarrow L(S,W,R)$. In Step 4 *Evaluator* takes $L(S,W,R)$, and based on sensors/weapons/targets parameters from *KB* it establishes matrix $B = [b_{i,j}]_{n^2 \times n}$ ($Evaluator(L(S,W,R),KB) \rightarrow B$) with all rational elements $b_{i,j}$. In Step 5 *Swt_opt* algorithm first scales up $B \rightarrow B'$, so every element $b'_{i,j}$ in the transformed matrix B' becomes an integer. Then *Swt_opt* algorithm reassigns $B := B'$ and optimizes sensor/weapon-target pairings, i.e., $Swt\_opt(B,X) \rightarrow X$. Step 6 tests if the sensor/weapon-target pairings found in the current iteration are optimal. If they are optimal then the engagement time t and optimal assignment X are saved ; i.e., $t \rightarrow t_{opt}$, $X \rightarrow X_{opt}$. In Step 7 an engagement time t is updated and sensor/weapon-target pairings X is reset. *SWAT* terminates in Step 2 if $t > T$. In this case, a postprocessor $Report(t_{opt},X_{opt},KB)$ takes saved engagement time t_{opt} along with the optimal sensor/weapon-target pairs found X_{opt} and generates an appropriate report based on *KB*.

In *SWAT* we can assume that Step 5 dominates Step 3 (i.e., $Swt\_opt(B,X)$ dominates $Predictor(S,W,R,KB,t)$) if the time horizon T is within a certain limit; when it's not too large. Otherwise, the prediction process becomes more computationally expensive and *Predictor* might dominate *Swt_opt*. Consider the worst-case running time complexity of *SWAT* when *Swt_opt* dominates *Predictor*. *Evaluator* in Step 4 evaluates n^2 x n elements of B with a fixed amount of time per element. So, *Evaluator* in *SWAT* requires $O(n^3)$ operations. On the other hand, based on (9) derived in Section 6, *Swt_opt* requires $O(n^4Cb_{max})$ in the worst case, where integer Cb_{max} represents the largest benefit in B obtained after scaling. So, each iteration in *SWAT* requires at most $O(n^4Cb_{max})$. Based on relations (1-4) there are at most T/δ iterations defined by Step 2 in *SWAT*. Hence, if *Swt_opt* dominates *Predictor* the worst execution time of *SWAT* is

$$O(n^4Cb_{max}\, T/\delta). \qquad (10)$$

CONCLUSIONS

In this paper we introduced the novel algorithm named *SWAT* for optimized pairing of sensor-weapon pairs with targets at proposed time $t \leq T$, where T represents the planning time horizon. *SWAT* can significantly enhance the effectiveness and lethality of collaborative engagement of multiple targets in modern battlefields. Our algorithm consists of three main components - *Predictor*, *Evaluator*, and *Optimizer* - that were described in Sections 4-6. In this work we focused most attention on *Optimizer* (i.e., exact *Swt_opt* algorithm) that we described in Section 6.3. More research is anticipated (and needed) on the design and implementation of *Predictor* and *Evaluator*, which by their own right are the challenging problems. In particular, *Evaluator* requires establishing the benefits for every sensor/weapon-target combination. The difficulty here lies in establishing correct/realistic relative values of such benefits. One possible source for establishing such benefits might be Joint Munitions Effectiveness Manual (JMEM), but this covers only weapon-target pairings. Another possibility would be to employ a human domain expert who would manually enter benefits for sensor/weapon-target combinations to the specialized *KB* system.

For limited time horizon *T*, which has to be empirically determined for given scenario(s), the time complexity of *Swt_opt* dominates the time complexities of *Predictor* and *Evaluator*. So, the expected running time for *SWAT* in such a case is $O(n^4Cb_{max}\, T/\delta)$.

Computational results for independent sensor/weapon-target pairings (Bogdanowicz and Coleman, 2007) based on the modified algorithm implemented in $O(n^3Cb_{max})$ indicate that for $n \leq 120$ such an algorithm requires, on the average, a few seconds to execute. This in turn suggests that *Swt_opt* introduced in this work should perform well (i.e., order of seconds) for inputs defined by $n \leq 100$. So, for the small unit (e.g., squad or platoon) network lethality and collaborative engagement, *Swt_opt* (and hence *SWAT*) should perform efficiently. This performance of *Swt_opt*, however, needs to be studied and verified. For n much greater than *100* (e.g., support for collaborative engagement on brigade level) either a good heuristic approach or the distributed implementation of *Swt_opt* algorithm could be a critical factor to yield high performance. In particular, implementation of *Swt_opt* for parallel computing seems to be well suited here.

REFERENCES

Adler I., Karmarkar N., Resende M., Veiga G., 1989: An Implementation of Karmarkar Algorithm for Linear Programming, *Math. Programming* **44**, 297-335.

Bertsekas D., 1992a: Linear Network Optimization: Algorithms and Codes, The MIT Press.

Bertsekas D., 1992b: Auction Algorithms for Network Flow Problems: A Tutorial Introduction, *Comp. Optimiz. and Appl.* **1**, 7-66.

Bertsekas D., 1990: The Auction Algorithm for Assignment and Other Network Flow Problems, *Interfaces* **20**, 133-149.

Bogdanowicz Z. and P. Coleman N., 2007: Optimization of Sensor/Weapon-Target Pairings Based on Auction Algorithm, *WSEAS Transactions on Mathematics*, **6**, 730-735.

Bogdanowicz Z., Coleman N., and Kaniyantethu S., 2005: Combat Decision Support Subsystem for Optimal Weapon-Target Assignment, *DCDIS Proceedings*, Watam Press, 11-15.

Bogdanowicz Z. and Coleman N., 2004a: Auction Algorithm for Weapons/Targets Pairing Application, *24th ASC Proceedings*, CS-03, Orlando, Florida.

Bogdanowicz Z. and Coleman N., 2004b: Efficient methodology and robust infrastructure for assigning weapons to targets, *AMCS Proceedings*, CSREA Press, 289-295.

Castanon D., 1993: Reverse Auction Algorithms for Assignment Problems, *Algorithms for Network Flows and Matching – American Math. Society* 407-429.

Galil Z., 1986: Efficient Algorithms for Finding Maximum Matchings in Graphs. *ACM Computing Surveys*, **18**, 23-38.

Karmarkar N., 1984: A New Polynomial-time Algorithm for Linear Programming, *Combinat.* **4**, 373-395.

Lovász L. and Plummer M., 1986: Matching Theory. North-Holland, Amsterdam

Micali S. and Vazirani V., 1980: An O($\sqrt{|V||e|}$) algorithm for finding maximum matching in general graphs, *Proc. 21st Symp. Foundations of Computing*, 17-27.

Todd M., 1992: A Low Complexity Interior Point Algorithm for Linear Programming, *SIAM Journal of Optimization* **2**, 198-209.