MIT Sloan School of Management

Working Paper 4464-03
July 2003

Exact and Heuristic Methods for the Weapon Target Assignment Problem

Ravindra K. Ahuja, Arvind Kumar, Krishna Jha, James B. Orlin

Exact and Heuristic Algorithms

for the

Weapon Target Assignment Problem

Ravindra K. Ahuja
Department of Industrial and Systems Engineering
University of Florida
Gainesville, FL 32611
ahuja@ufl.edu

Arvind Kumar
Department of Industrial and Systems Engineering
University of Florida
Gainesville, FL 32611
arvind22@ufl.edu

Krishna C. Jha
Department of Industrial and Systems Engineering
University of Florida
Gainesville, FL 32611
kcjha@ufl.edu

James B. Orlin
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139
jorlin@mit.edu

(July 4, 2003)

Exact and Heuristic Algorithms for the Weapon Target Assignment Problem

Ravindra K. Ahuja[*], Arvind Kumar[*], Krishna C. Jha[*], and James B. Orlin[**]

Abstract

The Weapon Target Assignment (WTA) problem is a fundamental problem arising in defense-related applications of operations research. This problem consists of optimally assigning *n* weapons to *m* targets so that the total expected survival value of the targets after all the engagements is minimum. The WTA problem can be formulated as a nonlinear integer programming problem and is known to be NP-complete. There do not exist any exact methods for the WTA problem which can solve even small size problems (for example, with 20 weapons and 20 targets). Though several heuristic methods have been proposed to solve the WTA problem, due to the absence of exact methods, no estimates are available on the quality of solutions produced by such heuristics. In this paper, we suggest linear programming, integer programming, and network flow based lower bounding methods using which we obtain several branch and bound algorithms for the WTA problem. We also propose a network flow based construction heuristic and a very large-scale neighborhood (VLSN) search algorithm. We present computational results of our algorithms which indicate that we can solve moderately large size instances (up to 80 weapons and 80 targets) of the WTA problem optimally and obtain almost optimal solutions of fairly large instances (up to 200 weapons and 200 targets) within a few seconds.

[*] Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611.

[**] James B. Orlin, Sloan School of Management, MIT, Cambridge, MA 02139.

1. Introduction

The *Weapon-Target Assignment* (WTA) problem is a fundamental problem arising in defense-related applications of operations research. The problem consists of optimally assigning weapons to the enemy-targets so that the total expected survival value of the targets after all the engagements is minimized. There are two versions of the WTA problem: *static* and *dynamic*. In the static version, all the inputs to the problem are fixed; that is, all targets are known, all weapons are known, and all weapons engage targets in a single stage. The dynamic version of the problem is a multi-stage problem where some weapons are engaged at the targets at a stage, the outcome of this engagement is assessed and strategy for the next stage is decided. In this paper, we study the static WTA problem; however, our algorithms can be used as important subroutines to solve the dynamic WTA problem.

We now give a mathematical formulation of the WTA problem. Let there be n targets, numbered 1, 2, …, n, and m weapon types, numbered 1, 2, …, m. Let V_j denote the value of the target j, and W_i denote the number of weapons of type i available to be assigned to targets. Let p_{ij} denote the probability of destroying target j by a single weapon of type i. Hence $q_{ij}=1 - p_{ij}$ denotes the probability of survival of target j if a single weapon of type i is assigned to it. Observe that if we assign x_{ij} number of weapons of type i to target j, then the survival probability of target j is given by $q_{ij}^{x_{ij}}$. A target may be assigned weapons of different types. The WTA problem is to determine the number of weapons x_{ij} of type i to be assigned to target j to minimize the total expected survival value of all targets. This problem can be formulated as the following nonlinear integer programming problem:

$$\text{Minimize } \sum_{j=1}^{n} V_j \left(\prod_{i=1}^{m} q_{ij}^{x_{ij}} \right) \tag{1a}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le W_i, \qquad \text{for all } i = 1, 2, \dots, m, \tag{1b}$$

$$x_{ij} \ge 0 \text{ and integer, for all } i = 1, 2, \dots, m, \text{ and for all } j = 1, 2, \dots, n. \tag{1c}$$

In the above formulation, we minimize the expected survival value of the targets while ensuring that the total number of weapons used is no more than those available. This formulation presents a simplified version of the WTA problem. In more practical versions, we may consider adding additional constraints, such as (i) lower and/or upper bounds on the number of weapons of type i assigned to a target j; (ii) lower and/or upper bounds on the total number of weapons assigned to target j; or (iii) a lower bound on the survival value of the target j. The algorithms proposed in this paper can be easily modified to handle these additional constraints.

Research on the WTA problem dates back to the 1950s and 1960s where the modeling issues for WTA problem were investigated (Manne [1958], Braford [1961], Day [1966]). Lloyd and Witsenhausen [1986] established the NP-completeness of the WTA problem. Exact algorithms have been proposed to solve the WTA problem for the following special cases: (i) when all the weapons are identical (DenBroder et al. [1958] and Katter [1986]) or (ii) when the targets can receive at most one weapon (Chang et al. [1987] and Orlin [1987]). Some of the heuristics proposed to solve the WTA problem are based on nonlinear network flow (Castanon et al. [1987]), neural networks (Wacholder [1989]), and genetic algorithms (Grant et al. [1993]). Green et al. [1997] applied a goal programming-based approach to the WTA problem. Metler and Preston [1990] have studied a suite of algorithms for solving the WTA problem efficiently, which is critical for real-time applications of the WTA problem. Maltin [1970], Eckler and Burr [1972] and Murphey [1999] provide comprehensive reviews of the literature on the WTA problem. Research to date on the WTA problem either solves the WTA problem for special cases or develops heuristics for the WTA problem. Moreover, since no exact algorithm is available to solve the weapon target assignment problems, it is not known how accurate are the solutions obtained by these heuristic algorithms.

In this paper, we propose several exact and heuristic algorithms to solve the WTA problem. Our branch and bound algorithms are the first implicit enumeration algorithms that can solve moderately size instances of the WTA problem optimally. We also propose heuristic algorithms which generate almost optimal solutions within a few seconds. Our paper makes the following contributions:

- We formulate the WTA problem as an integer linear programming problem, that is, as a generalized integer network flow problem on an appropriately defined network. The linear programming relaxation of this formulation gives a lower bound on the optimal solution of the WTA problem. We describe this formulation in Section 2.1.

- We propose a minimum cost flow formulation that yields a different lower bound on the optimal solution of the WTA problem. This lower bound is, in general, not as tight as the bound obtained by the linear programming formulation described above but it can be obtained in much less computational time. We describe this formulation in Section 2.2.

- We propose a third lower bounding scheme in Section 2.3 which is based on simple combinatorial arguments and uses a greedy approach to obtain a lower bound.

- We develop branch and bound algorithms to solve the WTA problem employing each of the three bounds described above. These algorithms are described in Section 3.

- We propose a very large-scale neighborhood (VLSN) search algorithm to solve the WTA problem. The VLSN search algorithm is based on formulating the WTA problem as a partition problem. The VLSN search starts with a feasible solution of the WTA problem and performs a sequence of *"cyclic and path exchanges"* to improve the solution. We describe in Section 4 a heuristic method that obtains an excellent feasible solution of the WTA problem by solving a sequence of minimum cost flow problems, and then uses a VLSN search algorithm to iteratively improve this solution.

- We perform extensive computational investigations of our algorithms and report these results in Section 5. Our algorithms solve moderately large size instances (up to 80 weapons and 80 targets) of the WTA problem optimally and obtain almost optimal solutions of fairly large instances (up to 200 weapons and 200 targets) within a few seconds.

2. Lower-bounding Schemes

In this section, we describe four lower bounding schemes for the WTA problem, using linear programming, integer programming, minimum cost flow problem, and a combinatorial method. These four approaches produce lower bounds with different values and have different running times.

2.1 A Lower Bounding Scheme using an Integer Generalized Network Flow Formulation

In this section, we formulate the WTA problem as an integer-programming problem with a convex objective function value. This formulation is based on a result reported by Manne [1958] who attributed it to Dantzig (personal communications).

In formulation (1), let $s_j = \prod_{i=1}^{m} q_{ij}^{x_{ij}}$. Taking logarithms on both sides, we obtain, $log(s_j) = \sum_{i=1}^{m} x_{ij} log(q_{ij})$ or $-log(s_j) = \sum_{i=1}^{m} x_{ij}\left(-log(q_{ij})\right)$. Let $y_j = -log(s_j)$ and $d_{ij} = -log(q_{ij})$. Observe that since $0 \le q_{ij} \le 1$, we have $d_{ij} \ge 0$. Then $y_j = \sum_{i=1}^{m} d_{ij} x_{ij}$. Also observe that $\prod_{i=1}^{m} q_{ij}^{x_{ij}} = 2^{-y_j}$. By introducing the terms d_{ij} and y_j in formulation (1), we get the following formulation:

$$\text{Minimize } \sum_{j=1}^{n} V_j 2^{-y_j} \tag{2a}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le W_i \qquad \text{for all } i = 1, 2, ..., m, \tag{2b}$$

$$\sum_{i=1}^{m} d_{ij} x_{ij} = y_j \qquad \text{for all } j = 1, 2, ..., n, \tag{2c}$$

$$x_{ij} \ge 0 \text{ and integer} \qquad \text{for all } i = 1,..., m \text{ and for all } j = 1,..., n, \tag{2d}$$

$$y_j \ge 0 \qquad \text{for all } j = 1, 2, ..., n. \tag{2e}$$

Observe that (2) is an integer programming problem with separable convex objective functions. This integer program can also be viewed as an integer generalized network flow problem with convex flow costs. Generalized network flow problems are flow problems where flow entering an arc may be different than the flow leaving the arc (see, for example, Ahuja, Magnanti, and Orlin [1993]). In a generalized network flow problem, each arc (i,j) has an associated multiplier γ_{ij} and the flow x_{ij} becomes $\gamma_{ij} x_{ij}$ as it travels from node i to node j. The formulation (2) is a generalized network flow problem on the network shown in the Figure 1. We give next some explanations of this formulation.

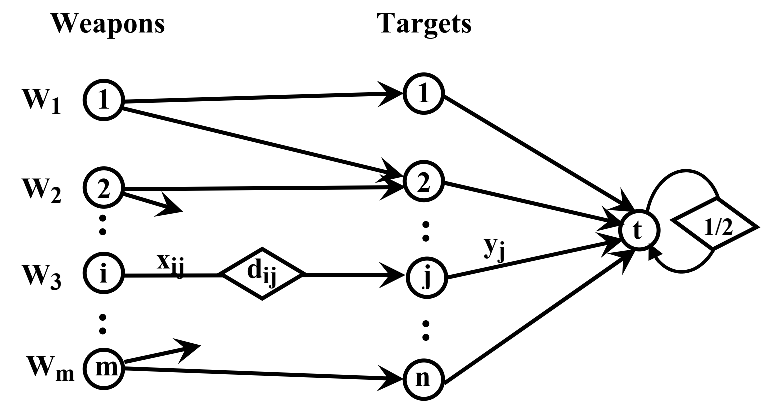

Figure 1. Formulating the WTA problem as an integer generalized network flow problem.

The network contains *m* weapon nodes, one node corresponding to each weapon type. The supply at node *i* is equal to the number of weapons available, W_i, for the weapon type *i*. The network contains *n* target nodes, one node corresponding to each target. Further, there is one sink node *t* whose demand equals the sum of all supplies. The supplies/demands of target nodes are zero. We now describe the arcs in the network. The network contains an arc connecting each weapon node to each target node. The flows on these arcs are given by x_{ij}, representing the number of weapons of type *i* assigned to the target *j*. The multipliers for these arcs are d_{ij}'s. Since there is no cost coefficient for x_{ij}'s in the objective function, the cost of flow on these arcs is zero. The network contains an arc from each of the target nodes to the sink node *t*. The flow on arc (*j, t*) is given by y_j and the cost of flow on this arc is $V_j 2^{-y_j}$. Finally, there is a loop arc (*t*, *t*) incident on node *t* with multiplier ½. An appropriate flow on this arc is sent so as to satisfy the mass balance constraints at node *t*.

In formulation (2), the cost of the flow in the network equals the objective function (2a); the mass balance constraints of weapon nodes are equivalent to the constraint (2b); and mass balance constraints of target nodes are equivalent to the constraint (2c). It follows that an optimal solution of the above generalized network flow problem will be an optimal solution of the WTA problem.

The generalized network flow formulation (2) is substantially more difficult than the standard generalized network flow problem (see, Ahuja et al. [1993]) since the flow values x_{ij}'s are required to be integer numbers (instead of real numbers) and the costs of flows on some arcs is a convex function (instead of a linear function). We will approximate each convex function by a piecewise linear convex function and relax the integer flows by real-valued flows so that the optimal solution of the modified formulation gives a lower bound on the optimal solution of the generalized formulation (2).

We consider the cost function $V_j 2^{-y_j}$ at values y_j that are integer multiples of a parameter p > 0, and draw tangents of $V_j 2^{-y_j}$ at these values. Let $F_j(p, y_j)$ denote the upper envelope of these tangents. It is easy to see that the function $F_j(p, y_j)$ approximates $V_j 2^{-y_j}$ from below and for every value of y_j provides a lower bound on $V_j 2^{-y_j}$. Figure 2 shows an illustration of this approximation.

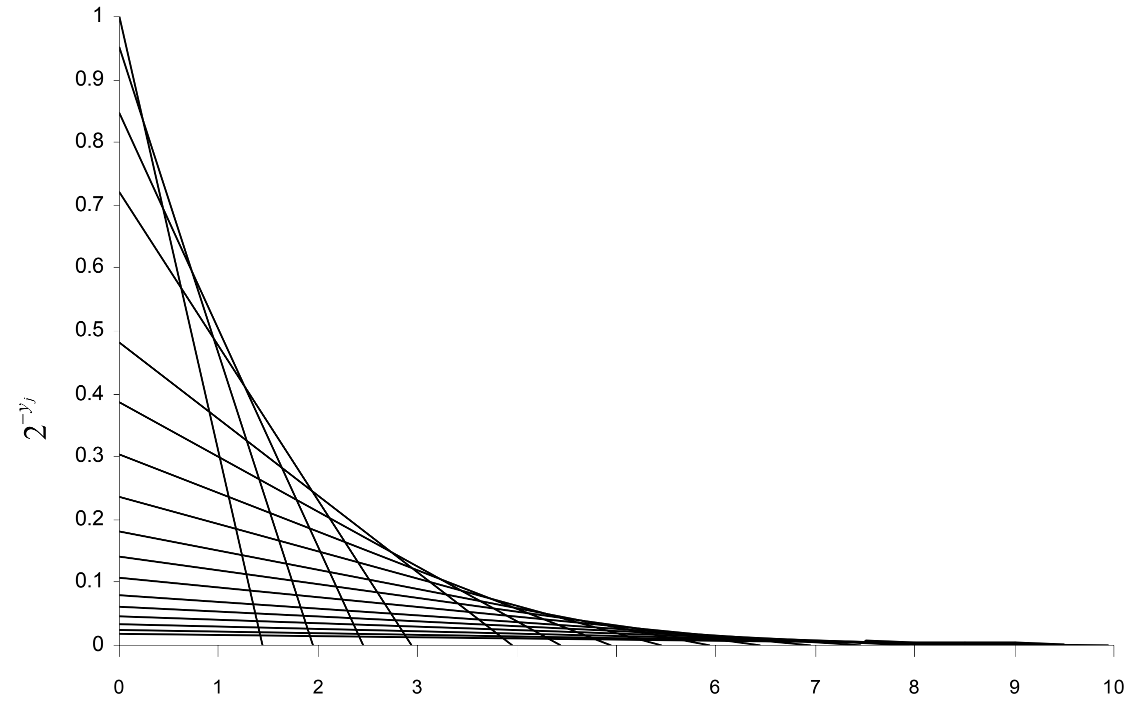

Figure 2: Approximating a convex function by a lower envelope of linear segments.

Thus, in formulation (2) if we replace the objective function (2a) by the following objective function:

$$\sum_{j=1}^{n} F_j(p, y_j), \tag{2a$'$}$$

we obtain a lower bound on the optimal objective function of (2a). Using this modified formulation, we can derive lower bounds in two ways:

LP Based Lower Bounding Scheme: Observe that the preceding formulation is still an integer programming problem because are flows x_{ij}'s are required to be integer valued. By relaxing the integrality of the x_{ij}'s, we obtain a mathematical programming problem with linear constraints and piecewise linear convex objective functions. It is well-known (see, Murty [1976]) that linear programs with piecewise linear convex functions can be transformed to linear programs by introducing a variable for every linear segment. We can solve this linear programming problem to obtain a lower bound for the WTA problem. Our computational results indicate the lower bounds generated by this scheme are not very tight.

MIP Based Lower Bounding Scheme: In this scheme, we do not relax the integrality of the x_{ij}'s, which keeps the formulation to be an integer programming formulation. We, however, transform the piecewise linear convex functions to linear cost functions by introducing a variable for every linear segment. We then use cutting plane methods to obtain a lower bound on the optimal objective function value. We have used the built-in routines in the software CPLEX 8.0 to generate Gomory and mixed integer rounding cuts to generate fairly tight lower bounds for the WTA problem.

We summarize the discussion in this section as follows:

Theorem 1. *Both the LP and MIP based lower bounding schemes give a lower bound on the optimal objective function value for the WTA problem.*

2.2 A Minimum Cost Flow Based Lower Bounding Scheme

The objective function of the WTA problem can also be interpreted as maximizing the expected damage to the targets. In this section, we develop an upper bound on the expected damage to the targets. Subtracting this upper bound on the expected damage from the total value of the targets (that is, $\sum_{i=1}^{m} W_i$) will give us a lower bound on the minimum survival value. We will formulate the problem of maximizing the damage to targets as a maximum cost flow problem. We show the underlying network G for the maximum cost flow formulation in Figure 3.

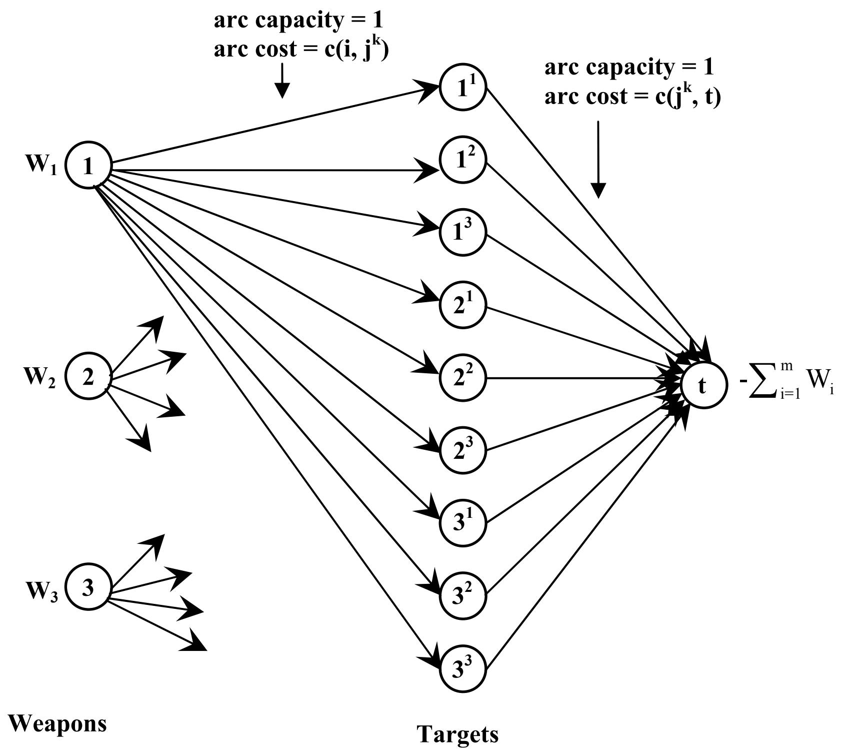

Figure 3. A network flow formulation of the WTA problem.

This network has three layers of nodes. The first layer contains a supply node i for every weapon type i with supply equal to W_i. We denote these supply nodes by the set N_1. The second layer of nodes, denoted by the set N_2, contains nodes corresponding to targets, but each target j is represented by several nodes $j^1, j^2,\ldots, j^k$, where k is the maximum number of weapons that can be assigned to target j. A node j^p represents the p^{th} weapon striking the target j. For example, the node labeled 3^1 represents the event of the first weapon being assigned to target 3, the node labeled 3^2 represents the event of the second weapon being assigned to target 3, and so on. All nodes in the second layer have zero supplies/demands. Finally, the third layer contains a singleton node t with demand equal to $\sum_{i=1}^{m} W_i$.

We now describe the arcs in this network. The network contains an arc (i, j^k) for each node $i \in N^1$ and each node $j^k \in N_2$; this arc represents the assignment of a weapon of type i to target j as the k^{th} weapon. This arc has a unit capacity. This network also contains an arc (j^k, t) with unit capacity for each node $j^k \in N^2$.

We call a flow x in this network a *"contiguous flow"* if it satisfies the property that if $x(i,j^k) = 1$, then $x(i,j^l) = 1$ for all l=1,2,…,k-1. In other words, the contiguous flow implies that a weapon i is assigned to target j as the k^{th} weapon provided that (k-1) weapons have already been assigned to it. The following result directly follows from the manner we have constructed the network G:

Observation 1. *There is one-to-one correspondence between feasible solutions of the WTA problem and contiguous flows in G.*

While there is a one-to-one correspondence between feasible solutions, it is not a cost preserving correspondence if we require costs to be linear. We instead provide linear costs that will overestimate the true non-linear costs. We define our approximate costs next.

The arc (i, j^k) represents the assignment of a weapon of type i to target j as the k^{th} weapon. If k = 1, then the cost of this arc is the damage caused to the target:

$$c(i, j^1) = V_j(1-q_{ij}) \tag{3}$$

which is the difference between the survival value of the target before strike (V_j) and the survival value of the target after strike ($V_j q_{ij}$). Next consider the cost $c(i, j^2)$ of the arc (i, j^2) which denotes the change in the survival value of target j when weapon i is assigned to it as the second weapon. To determine this, we need to know the survival value of target j before weapon i is assigned to it. But this cost depends upon which weapon was assigned to it as the first weapon. The first weapon striking target j can be of any weapon type 1,2,…,m and we do not know its type a priori. Therefore, we cannot determine the cost of the arc (i, j^2). However, we can determine an upper bound on the cost of the arc (i, j^2). We will next derive the expression for the cost of the arc (i, j^k) which as a special case includes (i, j^2).

Suppose that the first (k-1) weapons assigned to target j are of weapon types i_1, i_2, ..., i_{k-1}, and suppose that the type of the k^{th} assigned weapon is of type i. Then, the survival value of target j after the first (k-1) weapons is $V_j q_{i_1 j} q_{i_2 j} \ldots q_{i_{k-1} j}$ and the survival value of the target j after k weapons is $V_j q_{i_1 j} q_{i_2 j} \ldots q_{i_{k-1} j} q_{ij}$. Hence, the cost of the arc (i, j^k) is the difference between the two terms, which is

$$c(i, j^k) = V_j q_{i_1 j} q_{i_2 j} \ldots q_{i_{k-1} j}(1 - q_{ij}) . \tag{4}$$

Let $q_j^{max} = \max\{q_{ij}: 1, 2, \ldots, m\}$.Then, we can obtain an upper bound on $c(i, j^k)$ by replacing each q_{ij} by q_j^{max}. Hence, if we set

$$c(i, j^k) = V_j (q_j^{\max})^{k-1}(1 - q_{ij}) , \tag{5}$$

we get an upper bound on the total destruction on assigning weapons to targets. It directly follows from (5) that

$$c(i, j^1) > c(i, j^2) > \ldots > c(i, j^{k-1}) > c(i, j^k), \tag{6}$$

which implies that the optimal maximum cost flow in the network G will be a contiguous flow. It should be noted here that since this is a maximization problem, we solve it by first multiplying all arc costs by -1 and then using any minimum cost flow algorithm. Let z* represent the upper bound on destruction caused to targets after all the assignments obtained by solving this maximum cost flow problem. Then, the lower bound on the objective function of formulation (1) is $\sum_{i=1}^{m} W_i$ - z*.

We can summarize the preceding discussion as follows:

Theorem 2. *If z* is the optimal objective function value of the maximum cost flow problem in the network G, then* $\sum_{i=1}^{m} W_i$ *- z* is a lower bound for the weapon target assignment problem.*

2.3 Maximum Marginal Return Based Lower Bounding Method

In this section, we describe a different relaxation that provides a valid lower bound for the WTA problem. This approach is based on underestimation of the survival of a target when hit by a weapon as we assume that every target is hit by the best weapons.

Let q_j^{min} be the survival probability for target j when hit by the weapon with the smallest survival probability, i.e., $q_j^{min} = \min\{q_{ij}: i = 1, 2, \ldots, m\}$. Replacing the term q_{ij} in formulation (1) by q_j^{min}, we can formulate the WTA problem as follows:

$$\text{Minimize } \sum_{j=1}^{n} V_j \prod_{i=1}^{m} (q_j^{\min})^{x_{ij}} \tag{7a}$$

subject to

$$\sum_{j=1}^{n} x_{ij} \le W_i, \quad \text{for all } i = 1, 2, \ldots, m, \tag{7b}$$

$x_{ij} \geq 0$ and integer for all $i = 1, 2, \ldots, m$ and for all $j = 1, 2, \ldots, n$. (7c)

Let $x_j = \sum_{i=1}^{m} x_{ij}$, and if we let $g_j(x_j) = V_j(q_j^{\min})^{x_j}$, then we can rewrite (7) as:

Minimize $\sum_{j=1}^{n} g_j(x_j)$ (8a)

subject to

$\sum_{j=1}^{n} x_{ij} \leq W_i$, for all $i = 1, 2, \ldots, m$, (8b)

$\sum_{i=1}^{n} x_{ij} = x_j$ for all $i = 1, 2, \ldots, m$, (8b)

$x_{ij} \geq 0$ and integer for all $i = 1, 2, \ldots, m$ and for all $j = 1, 2, \ldots, n$. (8d)

It is also possible to eliminate the variables x_{ij} entirely. If we let $W = \sum_{i=1}^{m} W_i$ then we can rewrite (8) as an equivalent integer program (9):

Minimize $\sum_{j=1}^{n} g_j(x_j)$ (9a)

subject to

$\sum_{j=1}^{n} x_j \leq W$, (9b)

$x_j \geq 0$ and integer for all $j = 1, 2, \ldots, n$. (9c)

It is straightforward to transform a solution for (9) into one for (8) since all weapon types are identical in formulation (8).

Observe that the formulations (1) and (7) have the same constraints; hence, they have the same set of solutions. However, in the formulation (7), we have replaced each q_{ij} by q_j^{min}= min$\{q_{ij}$: i =1, 2, …, $m\}$, where $q_j^{min} < q_{ij}$. Noting that the optimal solution value for problems (7), (8) and (9) are all identical, we get the following result:

Theorem 3. *An optimal solution value for (9) is a lower bound for the WTA problem.*

Integer program (9) is a special case of the knapsack problem in which the separable costs are monotone decreasing and concave. As such it can be solved using the greedy algorithm. In the following, assigned(j) is the number of weapons assigned to target j, and value(i, j) is the incremental cost of assigning the next weapon to target j.

```
algorithm combinatorial-lower-bounding;
begin
    for j := 1 to n do
    begin
        assigned(j) := 0;
        value(j) := $g_j$(assigned(j)+1) – $g_j$(assigned(j));
    end
    for i = 1 to m do
    begin
        find j corresponding to the minimum value(j);
        assigned(j) := assigned(j) + 1;
        value(j) := $g_j$(assigned(j)+1) – $g_j$(assigned(j));
    end;
end.
```

This lower bounding scheme is in fact a variant of a popular algorithm to solve the WTA problem which is known as the *maximum marginal return algorithm*. In this algorithm, we always assign a weapon with maximum improvement in the objective function value. This algorithm is a heuristic algorithm to solve the WTA problem but is known to give an optimal solution if all weapons are identical.

We now analyze the running time of our lower bounding algorithm. If we store value(j) in a Fibonacci heap. We point that that after the initialization of the heap with the initial values, we perform W "find-min" operations and W decrease-key steps, for a total running time of O(W) time. In our implementation, we used binary heaps, which run in O(W log W) time but are comparably fast for the problem sizes that we considered.

3. A Branch and Bound Algorithm

We developed and implemented four branch and bound algorithms based on the four lower bounding schemes described in the previous section. A branch and bound algorithm is characterized by the branching, lower bounding, and search strategies. We now describe these strategies for our approaches.

Branching strategy: To keep the memory requirement low, the only information we store at any node is which variable we branch on at that node; and the lower and upper bounds at the node. To recover the partial solution associated with a node of the branch and bound tree, we trace back to the root of the tree. The branching strategy we have used in our implementation is based on the maximum-marginal return. For each node of the branch and bound tree, we find the weapon-target combination which gives best improvement and set the corresponding variable as the one to be branched on next. Ties are broken arbitrarily.

Lower bounding strategy: We used the three lower bounding strategies described in Section 2. We provide a comparative analysis of these bounding schemes in Section 5.

Search strategy: We implemented both the breadth-first and depth-first search strategies. We found that for smaller size problems (i.e., up to 10 weapons and 10 targets), breadth-first strategy gave overall better results; but for larger problems, depth-first search had a superior performance. We report the results for the depth-first search in section 5.

4. A Very Large-Scale Neighborhood Search Algorithm

In the previous two sections, we described branch and bound algorithms for the WTA problem. These algorithms are the first exact algorithms that can solve moderate size instances of the WTA problem in reasonable time. Nevertheless, there is still a need for heuristic algorithms which can solve large-scale instances of the WTA problems. In this section, we describe a neighborhood search algorithm for the WTA problem which has exhibited excellent computational results. This algorithm is an application of very large-scale neighborhood (VLSN) search to the WTA problem. A VLSN search algorithm is a neighborhood search algorithm where the size of the neighborhood is very large and we use some implicit enumeration algorithm to identify an improved neighbor. We refer the reader to the paper by Ahuja et al. [2002] for an overview of VLSN search algorithms.

A neighborhood search algorithm starts with a feasible solution of the optimization problem and successively improves it by replacing it by an improved neighbor until it obtains a locally optimal solution. The quality of the locally optimal solution depends both upon the quality of the starting feasible solution and the structure of the neighborhood, that is, how we define the neighborhood of a given solution. We next describe the method we used to construct the starting feasible solution followed by our neighborhood structure.

4.1 A Minimum Cost Flow formulation based Construction Heuristic

We developed a construction heuristic which solves a sequence of minimum cost flow problems to obtain an excellent solution of the WTA problem. This heuristic uses the minimum cost flow formulation shown in Figure 3, which we used to determine a lower bound on the optimal solution of the WTA problem. Recall that in this formulation, we define the arc costs (i, j^1), (i, j^2), …, (i, j^k), which, respectively, denote the cost of assigning the first, second and k^{th} weapon of type i to target j. Also recall that only the cost of the arc (i, j^1) was computed correctly, and for the other arcs, we used a lower bound on the cost. We call the arcs whose costs are computed correctly as *exact-cost arcs*, and the rest of the arcs as *approximate-cost arcs*.

This heuristic works as follows. We first solve the minimum cost flow problem with respect to the arc costs as defined earlier. In the optimal solution of this problem, exact-cost arcs as well as approximate-cost arcs may carry positive flow. We next fix the part of the weapon target assignment corresponding to the flow on the exact-cost arcs and remove those arcs from the network. In other words, we construct a partial solution for weapon-target assignment by assigning weapons only for exact-cost arcs. After fixing this partial assignment, we again compute the cost of each arc. Some previous approximate-cost arcs will now become exact-cost arcs. For example, if we set the flow on arc (i, j^1) equal to 1, we know that that weapon i is the first weapon striking target j, and hence we need to update the costs of the arcs (l, j^k) for all $l = 1, 2, …, m$ and for all $k \geq 2$. Also observe that the arcs (l, j^2) for all $l = 1$, 2, …, m now become exact cost arcs. We next solve another minimum cost flow problem and again fix the flow on the exact-cost arcs. We recompute arc costs, make some additional arcs exact-cost, and solve another minimum flow problem. We repeat this process until all weapons are assigned to the targets.

We tried another modification in the minimum cost flow formulation which gave better computational results. The formulation we described determines the costs of approximate-cost arcs assuming that the worst weapons (with the largest survival probabilities) are assigned to targets. However, we observed that in any near-optimal solution, the best weapons are assigned to the targets. Keeping this observation in mind, we determine the costs of valid arcs assuming that the best weapons (with the smallest survival probabilities) are assigned to targets. Hence, the cost of the arc (i, j^k), which is $c(i, j^k) = V_j q_{i_1 j} q_{i_2 j} \ldots q_{i_{k-1} j}(1-q_{ij})$ is approximated by $c(i, j^k) = V_j [q_{min}(j)]^{k-1}(1-q_{ij})$. Our experimental investigation shows that this formulation generates better solutions compared to the previous formulation. We present computational results of this formulation in Section 5.

4.2. The VLSN Neighborhood Structure

The WTA problem can be conceived of as a partition problem defined as follows. Let $\boldsymbol{S} = \{a_1, a_2, a_3, \ldots, a_n\}$ be a set of n elements. The partition problem is to partition the set $\boldsymbol{S}$ into the subsets S_1, S_2, S_3, …, S_K such that the cost of the partition is minimum, where the cost of the partition is the sum of the cost of each part. The WTA problem is a special case of the partition problem where the set of all weapons is partitioned into n subsets $S_1, S_2, \ldots, S_n$, and subset j is assigned to target j, $1 \leq j \leq n$. Thompson and Orlin [1989] and Thompson and Psaraftis [1993] proposed a VLSN search approach for partitioning problems which proceeds by performing *cyclic exchanges*. Ahuja et al. [2001, 2003] proposed further refinements of this approach and applied it to the capacitated minimum spanning tree problem. We will present a brief overview of this approach when applied to the WTA problem.

Let $\boldsymbol{S} = (S_1, S_2, \ldots, S_n)$ denote a feasible solution of the WTA problem where the subset S_j, $1 \leq j \leq n$, denotes the set of weapons assigned to target j. Our neighborhood search algorithm defines neighbors of the solution $\boldsymbol{S}$ as those solutions that can be obtained from $\boldsymbol{S}$ by performing *multi-exchanges*. A *cyclic multi-exchange* is defined by a sequence of weapons i_1- i_2- i_3-…- i_r- i_1 where the weapons i_1, i_2, i_3, …, i_r belong to different subsets S_j's. Let $t(i_1)$, $t(i_2)$, $t(i_3)$, …, $t(i_r)$, respectively, denote the targets to which weapons i_1, i_2, i_3,…, i_r, are assigned. The *cyclic multi-exchange* i_1- i_2- i_3-…- i_r- i_1 represents that weapon i_1 is reassigned from target $t(i_1)$ to target $t(i_2)$, weapon i_2 is reassigned from target $t(i_2)$ to target $t(i_3)$, and so on, and finally weapon t_r is reassigned from target $t(i_r)$ to target $t(i_1)$. We can similarly define a *path multi-exchange* by a sequence of weapons i_1- i_2- i_3-…- i_r which differs from the *cyclic multi-exchange* in the sense that the last weapon i_r is not reassigned and remains assigned to target $t(i_r)$.

The number of neighbors in the multi-exchange neighborhood is too large to be enumerated explicitly. However, using the concept of *improvement graph*, a profitable multi-exchange can be identified using network algorithms. The improvement graph $G(\boldsymbol{S})$ for a given feasible solution $\boldsymbol{S}$ of the WTA problem contains a node r corresponding to each weapon r and contains an arc (r, l) between every pair of nodes r and l with $t(r) \neq t(l)$. The arc (r, l) signifies the fact that weapon r is reassigned to target (say j) to which weapon l is currently assigned and weapon l is unassigned from its current target; the cost of this arc, c_{rl}, is set equal to the change in the survival value of the target. Let V'_j denote the survival value of the target j in the current solution. Then, the cost of the arc (r, l) is $c_{rl} = V'_j\left((q_{rj} / q_{lj}) - 1\right)$. We say that a directed cycle $W = i_1\text{-}\ i_2\text{-}\ i_3\text{-}\ldots\text{-}\ i_k\text{-}\ i_1$ in $G(\boldsymbol{S})$ is *subset-disjoint* if each of the weapons i_1, i_2, i_3, …, i_k is assigned to a different target. Thompson and Orlin [1989] showed the following result:

Lemma 1. *There is a one-to-one correspondence between multi-exchanges with respect to* ***S*** *and directed subset-disjoint cycles in G(***S***) and both have the same cost.*

This lemma allows us to solve the WTA problem using the following neighborhood search algorithm:

```
algorithm WTA-VLSN search;
begin
        obtain a feasible solution S of the WTA problem;
        construct the improvement graph G(S);
        while G(S) contains a negative cost subset-disjoint cycle do
        begin
                obtain a negative cost subset-disjoint cycle W in G(S);
                perform the multi-exchange corresponding to W;
                update S and G(S);
        end;
end;
```

Figure 4. The VLSN search algorithm for the WTA problem.

We now give some details of the VLSN search algorithm. We obtain the starting feasible solution ***S*** by using the minimum cost flow based heuristic described in Section 4.1. The improvement graph $G(\boldsymbol{S})$ contains W nodes and $O(W^2)$ arcs and the cost of all arcs can be computed in $O(W^2)$ time. We use a dynamic programming based algorithm (as described by Ahuja et al. [2003]) to obtain subset-disjoint cycles. This algorithm first looks for profitable two-exchanges involving two targets only; if no profitable two-exchange is found, it looks for profitable three-exchanges involving three targets; and so on. The algorithm either finds a profitable multi-exchange or terminates when it is unable to find a multi-exchange involving k targets (we set $k = 8$). In the former case, we improve the current solution, and in the latter case we declare the current solution to be locally optimal and stop. The running time of the dynamic programming algorithm is $O(W^2\, 2^k)$ per iteration, and is typically much faster since most cyclic exchanges found by the algorithm are swaps.

5. Computational Results

We implemented each of the algorithm described in the previous section and extensively tested them. We tested our algorithms on randomly generated instances as data for the real-life instances is classified. We generated the data in the following manner. We generated the target survival values V_j's as uniformly distributed random numbers in the range 25-100. We generated the kill probabilities for weapons engaging with the targets as uniformly distributed random numbers in the range 0.60-0.90. We performed all our tests on a 2.8 GHz Pentium 4 processor computer with 1 GB RAM PC. In this section, we present the results of these investigations.

5.1 Comparison of the lower bounding schemes

In our first investigation, we compared the tightness of the lower bounds generated by the lower bounding algorithms developed by us. We tested the three lower bounding schemes described in Section 2: (i) LP based lower bounding scheme; (ii) MIP based lower bounding scheme; (iii) the minimum cost based lower bounding scheme; and (iv) the maximum marginal return based lower bounding scheme. We tested an additional lower bounding scheme which is a variant of the LP based scheme.

The table shown in Figure 5 gives the computational results of these four lower bounding schemes. For each of these schemes, the first column gives the % gap from the optimal objective function value and the second column gives the time taken to obtain bound. The following observations can be derived from this table: (i) the MIP lower bounding scheme gives the tightest lower bounds but also takes the maximum computational time; (ii) the minimum cost flow based bounding scheme gives fairly tight lower bounds when the number of weapons is less than or equal to the number of targets; and (iii) the maximum marginal return algorithm takes the least amount of time to obtain lower bounds.

# of Weapons	# of Targets	LP Scheme		MIP Scheme		Min Cost Flow Scheme		Maximum Marginal Return Scheme	
		% Gap	Time (in secs)	% Gap	Time (in secs)	% Gap	Time (in secs)	% Gap	Time (in secs)
5	5	8.03	0.015	0.21	0.016	1.66	<0.001	10.61	<0.001
10	10	3.63	0.015	0.12	0.031	0.00	<0.001	11.01	<0.001
10	20	19.70	0.015	0.04	0.062	0.00	<0.001	1.45	<0.001
20	10	11.88	0.015	0.53	0.156	21.32	<0.001	19.00	<0.001
20	20	7.28	0.031	0.25	0.109	1.32	<0.001	6.40	<0.001
20	40	23.35	0.046	0.04	0.296	0.00	<0.001	1.57	<0.001
40	10	14.79	0.015	2.12	0.609	42.41	<0.001	46.89	<0.001
40	20	7.06	0.031	0.45	0.359	25.52	0.015	13.53	<0.001
40	40	6.83	0.078	0.11	0.703	1.63	0.015	3.05	<0.001
40	80	21.69	0.14	0.03	1.812	0.00	0.046	0.88	<0.001

Figure 5. Comparison of four lower bounding schemes.

5.2 Comparison of Branch and Bound Algorithms

We developed branch and bound algorithms using the preceding lower bounding schemes. Figure 6 gives the results of these algorithms. The branch and bound algorithm using the LP based lower bounding scheme did not perform well at all and we do not present its results. We replaced this algorithm by another algorithm which we call the *hybrid algorithm*. The hybrid algorithm computes lower bounds using both the minimum cost flow based and the maximum marginal return based lower bounding schemes and uses the better of these two bounds. We find that the branch and bound algorithm using the MIP based lower bounding gives the most consistent results and is able to solve the highest size problems (containing 80 weapons and 80 targets). We also find that the hybrid algorithm also gives excellent results for those instances where the number of weapons is less than or equal to the number of targets.

# of Weapons	# of Targets	MIP Based B&B Algorithm		Min Cost Flow Based B&B Algorithm		Maximum Marginal. Return Based B&B Algorithm		Hybrid Algorithm	
		Nodes Visited	Time (in secs)	Nodes Visited	Time (in secs)	Nodes Visited	Time (in secs)	Nodes Visited	Time (in secs)
5	5	15	0.14	11	<0.001	23	<0.001	11	<0.001
10	10	29	0.56	1	<0.001	181	<0.001	1	<0.001
10	20	23	0.83	1	<0.001	83	<0.001	1	0.015
20	10	101	7.27	-	-	2,8611	1.34	20,251	2.52
20	20	109	6.56	2,383	4.39	15936	0.94	1,705	2.50
20	40	105	16.58	1	<0.001	111,603	10.14	1	0.015
40	10	1,285	327.27	-	-	-	-	-	-
40	20	205	35.19	-	-	$\sim 10^8$	13,651.9	$\sim 10^7$	25,868.9
40	40	211	50.96	$\sim 10^6$	10,583.62	$\sim 10^6$	943.03	38,3275	1,891.83
40	80	385	235.41	1	0.031	-	-	1	0.031
80	40	117,227	43,079.55	-	-	-	-	-	-
80	80	44905	58,477.31	-	-	-	-	-	-
80	160	1055	3,670.49	1	0.062	-	-	1	0.062

Figure 6. Comparison of branch and bound algorithms.

5.3 Performance of the VLSN Search Algorithm

We now present computational results of the minimum cost flow based construction heuristic and the VLSN search algorithm. The table shown in Figure 7 gives the objective function values of the solutions obtained by the construction heuristic and the improved values when VLSN search algorithm is applied to these solutions. We observe that the construction heuristic obtained optimal solutions for over 50% of the instances and for the remaining instances the VLSN search algorithm converted them into optimal or almost optimal solutions. The computational times taken by these algorithms are also very small and even fairly large instances are solved within 3 seconds.

# of Weapons	# of Targets	Construction Heuristic		VLSN Algorithm	
		Optimality Gap	Time (in seconds)	Optimality Gap	Time (in seconds)
10	5	0%	<0.001	0%	<0.001
10	10	0%	<0.001	0%	<0.001
10	20	0%	<0.001	0%	<0.001
20	10	0%	<0.001	0%	<0.001
20	20	0%	<0.001	0%	<0.001
20	40	0%	0.015	0%	0.015
20	80	0%	0.015	0%	0.031
40	10	1.79%	0.015	0%	0.031
40	20	0.33%	0.015	0%	0.015
40	40	0%	0.015	0%	0.015
40	80	0%	0.031	0%	0.078
40	120	0%	0.062	0%	0.109
80	20	2.33%	0.109	0%	0.156
80	40	0.10%	0.062	0%	0.109
80	80	0.0003%	0.093	0.0003%	0.156
80	160	0%	0.172	0%	0.219
80	320	0%	0.390	0%	0.625
100	50	0.79%	0.120	0.0015%	0.437
100	100	0.001%	0.187	0.0009%	0.250
100	200	0%	0.375	0%	0.609
200	100	0.01%	0.656	0.0059%	0.828
200	200	0.001%	0.921	0.0008%	1.109
200	400	0%	1.953	0%	2.516

Figure 7. Results of the construction heuristic and the VLSN search algorithm.

6. Conclusions

In this paper, we consider the weapon target assignment problem which is considered to be one of the classical operations research problems that has been extensively studied in the literature but still has remained unsolved. Indeed, this problem is considered to be the holy grail of defense-related operations research. Though weapon target assignment problem is a nonlinear integer programming problem, we use its special structure to develop LP, MIP, network flow, and combinatorial lower bounding schemes. Using these lower bounding schemes in branch and bound algorithms gives us effective exact algorithms to solve the WTA problem. Our VLSN search algorithm also gives highly impressive results and gives either optimal or almost optimal solutions for all instances it is applied to. To summarize, we can now state that the WTA problem is a well-solved problem and its large-scale instances can also be solved in real-time.

Acknowledgements

We wish to thank Mr. Robert Murphey at the Air Force Research Laboratory, Eglin Air Force Base to bring this problem to our notice and for helpful discussions. The research of the first three authors was partially supported by the NSF grant DMI-0217359. The fourth author gratefully acknowledges partial support from NSF grant DMI-0217123.

References

Ahuja, R.K., O. Ergun, J.B. Orlin, and A.P. Punnen. 2002. A survey of very large scale neighborhood search techniques. *Discrete Applied Mathematics* **123**, 75-102.

Ahuja, R.K., J.B. Orlin, and D. Sharma. 2001. Multi-exchange neighborhood search algorithms for the capacitated minimum spanning tree problem. *Mathematical Programming* **91**, 71-97.

Ahuja, R.K., J.B. Orlin, and D. Sharma. 2003. A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem. *Operations Research Letters* **31**, 185-194.

Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey.

Braford, J.C. 1961. Determination of optimal assignment of a weapon system to several targets. AER-EITM-9, Vought Aeronautics, Dallas, Texas.

Chang S.C., R.M. James, and J.J. Shaw. 1987. Assignment algorithm for kinetic energy weapons in boost defense. *Proceedings of the IEEE 26th Conference on Decision and Control*, Los Angeles, CA.

Castanon D.A. 1987. Advanced weapon-target assignment algorithm quarterly report. TR-337, ALPHA TECH Inc., Burlington, MA.

Day, R.H. 1966. Allocating weapons to target complexes by means of nonlinear programming. *Operations Research* **14,** 992-1013.

DenBroader G. G., Jr., R. E. Ellison, and L. Emerling. 1958. On optimum target assignments. *Operations Research* **7,** 322-326.

Eckler, A.R., and S.A. Burr. 1972. Mathematical models of target coverage and missile allocation. Military Operations Research Society, Alexandria, VA.

Grant, K.E. 1993. Optimal resource allocation using genetic algorithms. 1993. *Naval Review*, Naval Research Laboratory, Washington, DC, pp. 174-175.

Green D.J., J.T. Moore, and J.J. Borsi. 1997 An integer solution heuristic for the arsenal exchange model (AEM). *Military Operations Research Society,* Volume 3 Number 2.

Katter, J.D. 1986. A solution of the multi-weapon, multi-target assignment problem. Working paper 26957, MITRE.

Lloyd, S.P., and H.S. Witsenhausen. 1986. Weapons allocation is NP-complete. In *Proceedings of the 1986 Summer Conference on Simulation*, Reno, NV.

Maltin, S.M. 1970. A review of the literature on the missile-allocation problem. *Operations Research* **18**, 334-373.

Manne, A.S. 1958. A target-assignment problem. *Operations Research* **6**, 346-351.

Metler, W.A., and F.L. Preston. 1990. A suite of weapon assignment algorithms for a SDI mid-course battle manager. NRL Memorandum Report 671, Naval Research Laboratory, Washington, DC.

Murphey, R.A. 1999. Target-based weapon target assignment problems. *Nonlinear Assignment Problems: Algorithms and Applications*. P. M. Pardalos and L. S. Pitsoulis (Eds.), Kluwer Academic Publishers, pp. 39-53.

Murty, K.G. 1976. *Linear and Combinatorial Optimization*. John Wiley.

Orlin, D. 1987. Optimal weapons allocation against layered defenses. *Naval Research Logistics* **34**, 605-616.

Thompson, P.M., and J.B. Orlin. 1989. The theory of cyclic transfers. MIT Operations Research Center Report 200-89, Cambridge, MA.

Thompson, P.M., and H.N. Psaraftis. 1993. Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. *Operations Research* **41,** 935-946.

Wacholder, E. 1989. A neural network-based optimization algorithm for the static weapon-target assignment problem. *ORSA Journal on Computing* **4**, 232-246.