3-23-2017

Real-Time Heuristic Algorithms for the Static Weapon-Target Assignment Problem

Alexander G. Kline

Real-Time Heuristic Algorithms for the Static Weapon-Target Assignment Problem

THESIS

Alexander G. Kline, CPT, USA

AFIT-ENS-MS-17-M-139

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-17-M-139

REAL-TIME HEURISTIC ALGORITHMS FOR THE STATIC WEAPON-TARGET ASSIGNMENT PROBLEM

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Alexander G. Kline, BS

CPT, USA

March 2017

AFIT-ENS-MS-17-M-139

REAL-TIME HEURISTIC ALGORITHMS FOR THE STATIC WEAPON-TARGET ASSIGNMENT PROBLEM

THESIS

Alexander G. Kline, BS
CPT, USA

AFIT-ENS-MS-17-M-139

Abstract

The problem of targeting and engaging individual missiles (targets) with an arsenal of interceptors (weapons) is known as the weapon target assignment problem. As many solution techniques are based upon a transformation of the objective function, their final solutions rarely produce optimal solutions. We propose a nonlinear branch and bound algorithm to provide the first optimization approach to the untransformed problem found in the literature. Further, we propose a new heuristic based upon the branch and bound algorithm which dominates other heuristics explored in optimality gap. We also propose a heuristic based upon the optimal solution to the quiz problem which finds solutions within 6% of optimal for small problems and provides statistically similar results as one of the best heuristics found in the literature for larger problems while solving these problems in ten thousandths of the time.

Acknowledgements

First I would like to thank my family for their love and support during this time. You mean the world to me and I couldn't have done this without you.

I would like to thank my advisor, Dr. Darryl Ahner, for his patient guidance over the past 18 months. I was only able to achieve the success in this program because of your dedication to education, research, and above all, professional excellence which I hope to emulate in years to come.

I would also like to thank Dr. Brian Lunday for his constant support and professionalism in our time together. I would not be here nor would I have been awarded the opportunities that have been bestowed upon me without you.

Thank you all.

Alexander G. Kline

Table of Contents

List of Figures

List of Tables

REAL-TIME HEURISTIC ALGORITHMS FOR THE STATIC
WEAPON-TARGET ASSIGNMENT PROBLEM

I. Introduction

Since its introduction to the operations research community, the weapon target assignment problem has been a complicated problem of constant relevance. Mid-course interdiction of aerial threats was a subject of concern during the Cold War, and perpetual advances in ballistic missile capabilities make it an area of continued study. Despite numerous United Nations resolutions prohibiting the development and testing of ballistic missile technology, many nations continue to take efforts at improving these weapon systems. As the missile threat is growing, so too must our capacity to defend our borders and areas of operation of deployed forces. The problem of targeting and engaging individual missiles (targets) with an arsenal of interceptors (weapons) is known as the weapon target assignment problem. The literature on this problem provides many solution techniques which collectively provide efficient optimal and near optimal solutions. The first line of the 2010 Ballistic Missile Defense Review Report states

> The protection of the United States from the threat of ballistic missile attack is a critical national security priority (United States Department of Defense, 2010).

In this report, the US Department of Defense outlines the threat of ballistic missiles and the importance of our continuous efforts to improve the systems necessary to provide adequate defenses against these threats. Among others, North Korea and

Iran have maintained ballistic missile programs seeking to develop improved capabilities at aggressively targeting the US, its allies, and regions wherein deployed forces operate. On 7 February 2016, North Korea launched a three-stage missile which has the capacity to reach the US (MG Mahon, Francis (Ret.), 2016). This followed a May 2015 submarine launched missile test (Evans, Stephen, 2015). Despite the restrictions outlined in existing nuclear agreements, Iran has also been continuing its missile capabilities, with a ballistic missile test as recently as July 2016 (Tomlinson, Lucas, 2016). Our defense systems must improve to combat these aggressive developments in "system improvements that complicate an adversaries' situation through technological overmatch" (MG Mahon, Francis (Ret.), 2016). While the deployment and efficacy of air defense technology is of high importance in combating this threat, so to is the decision regarding which system or systems should be used to target ballistic missiles. The optimization of the assignment of air defense weapons to interdict ballistic missiles is the subject of this thesis.

The weapon target assignment (WTA) problem has been researched within the operations research community for nearly 60 years, with initial published attention from Manne (1958) based upon a talk given by Flood (1957). Given n incoming targets, solving the problem results in the assignment of m weapons to the targets so as to minimize the collective residual value of the targets. The value of the targets, V_j, corresponds to their destructive capacity and each weapon w_i has an associated probability p_{ij} of destroying target t_j. As the problem seeks to minimize the residual value of each target, known in literature as target leakage, the probability of survival is defined by $q_{ij} = 1 - p_{ij}$. The WTA problem formulation is nonlinear and is defined by

$$
\begin{aligned}
& min\sum_{j=1}^{n} V_j \prod_{i=1}^{m} q_{ij}^{x_{ij}} \qquad (1)\\
& st\sum_{j=1}^{m} x_{ij} \leq w_i \;\; for\, i = 1, ..., m,\\
& \qquad x_{ij} \in \mathbb{Z}^{+} \cup \{0\},\;\; for\, i = \{1, ..., m\},\;\; j = \{1, ..., n\}
\end{aligned}
$$

The WTA problem is typically approached from two sets of conditions: the static WTA (SWTA) problem, which has known quantities of weapons and targets, probabilities of kill, and target values, and the dynamic WTA (DWTA) problem, which may have multiple waves of targets, uncertain numbers of targets, or any probabilistic variation of the aforementioned. This paper focuses on the formulation and solutions surrounding the SWTA problem, specifically where there exists only one of each weapon type $w_i = 1\ \forall i = 1, \ldots, m$, and the number and properties of the targets are known.

The SWTA problem was first proposed by Flood (1957) as a military optimization problem with similar constraints to the personnel assignment problem but, due to its nonlinear objective, was inaccessible with existing computational capabilities (Manne, 1958). Manne defined the problem and demonstrated its likeness to the transportation problem by introducing a variable $-y_j = \sum_{i=1}^{m} x_{ij} ln(1 - p_{ij})$, which allows for the transformation of the objective to $\min\limits_{\boldsymbol{y_j}} \sum_{j=1}^{n} V_j e^{-y_j}$, a nonlinear program whose solution can be approximated by a convex set of lower bounding linear constraints. Both denBroeder *et al.* (1959) and Walkup & MacLaren (1964) propose solution methods to Mannes formulation and transformations, with denBroeder using an iterative approach which is similar to a maximum marginal return algorithm (denBroeder *et al.*, 1959) whereas Walkup uses a network graph approach (Walkup

& MacLaren, 1964).

Matlin (1970) described the characterization of the WTA model using five submodels: the weapon system, the target complex, the engagement, the damage model, and the algorithm. His submodels are used to develop complete models of the WTA problem and he describes the complexities with which each submodel can be defined.

Day (1966) designed a three-stage optimization procedure in which the author decomposed the problem into smaller problems to obtain an estimate for the number of weapons to assign to each target, from which he solved the overarching nonlinear problem. He used a nonlinear programming algorithm to solve for optimal assignments of the weapons to the targets and rounded the optimal decision variable values to integer values for his solution.

As interest in the WTA problem increased while computational resources were still limited, more research was devoted to the heuristic and transformation-based solutions to best approximate the optimal solutions. Castañon (1987) used a nonlinear network flow algorithm to estimate a near optimal solution from a lower bound in a manner similar to Day. Chang *et al.* (1987) developed an iterative linear network programming algorithm using denBroeder's maximum marginal return (MMR) approach and Castañon's nonlinear network flow methods to solve a linear transformation of the nonlinear objective. Wacholder (1989) then used a neural network heuristic and Lee *et al.* (2002) used immunity based ant colony optimization techniques to generate near optimal solutions to the problem.

Amidst the rapid development of heuristics to efficiently find near optimal solutions, exact algorithms continued to emerge. Johansson & Falkman (2009) developed an exhaustive search algorithm which they used for comparison to a genetic algorithm heuristic they developed. Rosenberger *et al.* (2005) developed a branch and bound algorithm to consider multiple weapon assignments per target. These algorithms, though exact, were only able to find solutions to small instances having 9 weapons and 9 targets.

The lack of exact algorithms and heuristics capable of solving large problems was addressed by Ahuja *et al.* (2007), who used the network flow concept as proposed by Castañon to develop lower bounding techniques for use in a branch and bound algorithm for a linear transformation to the problem. Though this generated approximated optimal solutions to the transformed problem, it was unable to run to completion for larger problems, and so a construction heuristic and a very large scale neighborhood search heuristic were developed using the network flow approach to obtain near optimal solutions for large scale problems. Their efforts provided accurate heuristics for some of the largest sized problems considered to date in the literature.

In this thesis, we propose the application of a branch and bound algorithm to optimize the untransformed objective function, something we noted as having been absent to date in the literature. We solve the SWTA problem via the optimization of the nonlinear objective function yields solutions that are not dependent on the quality of the approximation but instead exploits the special structure of the problem. Further, we put forth two search heuristics which achieve near optimal solutions to the problem. The first is based on our branch and bound algorithm and typically finds the superior solution when compared to other heuristics. The second heuristic is based upon the quiz problem solution and is able to achieve near optimal solutions to large scale problems within a few hundredths of a second.

The contributions of this thesis address the following questions:

Are there algorithms capable of finding exact solutions to the WTA problem? We implement a nonlinear branch and bound algorithm, not currently in the literature, as an exact algorithm for the optimization of the WTA problem taking advantage of improved contemporary computational power.

Are there faster approaches to finding near optimal solutions to large WTA problems than exist in the literature? We develop and present a Modified Quiz Problem Search Heuristic capable of finding near optimal solutions of some to the largest problem instances addressed in the literature within 0.03 seconds, real-time.

Are there methods of overcoming the shortcomings of greedy search techniques for heuristic solution techniques? We identify scenarios wherein greedy search techniques fail to make quality assignments. We use our knowledge of these scenarios to develop techniques capable of identifying when greedy searches are inferior and develop a Modified Quiz Problem Search Heuristic to address these shortfalls.

Are there unexplored heuristics which can find near optimal solutions for WTA problems better than existing heuristics? We develop a Greedy Branch and Bound Heuristic new to the literature. The solutions found using this heuristic typically dominated the other heuristics considered in this paper for small to medium sized instances of the SWTA problem. Solution time increased exponentially with respect to the size of the problem, precluding the identification of solutions to large problem instances.

Convexity of the WTA Problem.

As all efforts to the present have first performed a linear transformation of the WTA objective function, we seek to optimize the untransformed nonlinear objective function as solutions to the transformed problem only provide approximations of the optimal solution.

Theorem 1 *The objective function of the WTA problem is convex.*

Proof: Because $f(x) = \sum_{j=1}^{n} V_j \prod_{i=1}^{m} q_{ij}^{x_{ij}}$ is the sum of positively weighted ($i.e.$, $V_j > 0$, $\forall j \in J$) functions, it is sufficient to prove that $\prod_{i}^{m} q_{ij}^{x_{ij}}$ is convex for any $j \in J$. We show that this function is (strictly) convex for $|I| = 1$. We prove convexity of higher cardinality sets of I via the positive semidefinite property of the Hessian. By induction, we show that the determinant of the Hessian of $\prod_i^m q_{ij}^{x_{ij}}$ for any $j \in J$ equals 0 for $|I| = 2$ and, if it equals 0 for $|I| = n$, then it also equals 0 for $|I| = n + 1$.

When $|I| = 1$, the function is

$$f(x) = q^x, \ \ for \ x \geq 0,$$

and the second derivative for this function is

$$\frac{d^2 f}{dx^2} = q^x ln(q)^2.$$

We know that the resulting Hessian is positive definite because, for any value $q \in (0, 1)$, the expression is the product of two positive values, which is also positive. Because the 1×1 dimensional matrix containing the second derivative of this function is positive definite, the (trivial) principal minors are positive definite, and the corresponding function is (strictly) convex.

For $|I| = 2$, our function is

$$f(x) = q_1^{x_1} q_2^{x_2},$$

and the Hessian for this function, letting $a_1 = ln(q_1)$ and $a_2 = ln(q_2)$, is

$$H\left(f(x)\right) = q_1^{x_1} q_2^{x_2} \begin{bmatrix} a_1^2 & a_1 a_2 \\ a_1 a_2 & a_2^2 \end{bmatrix}$$

We know that $(q_1^{x_1} q_2^{x_2})$ is positive as the product of two positive numbers. Further, we know that the diagonals of the Hessian matrix are positive because they are the squares of a_1 and a_2, which are real numbers. We observe that for each increment of $|I|$, the resulting Hessian has an additional row which is a scalar multiple of the first row. Here the second row is the scalar multiple $\frac{a_2}{a_1}$ of the first row. We can therefore define a matrix B_2 which is the reduction of this Hessian by the addition of the negative scalar multiple of the first row to the second row. As we have defined B_2, the determinant of B_2 is equal to the determinant of the Hessian. Since B_2 is an upper triangular matrix, its determinant is the product of its diagonal elements, one of which is 0. Therefore, its determinant is 0. Since the Hessian is positive semidefinite and the principal minors are positive, the function is therefore convex.

Assuming that the function is convex for $|I| = n$, we can show that it is convex for $|I| = n+1$ by observing that the Hessian matrix has a factorable scalar of $\prod_i q_i^{x_i}$, which is the product of positive numbers. Further, the diagonal elements of the factored Hessian are a_i^2, $\forall i \in I$, which are positive since a_i are real numbers. Lastly, since the $(n+1)^{st}$ row of the Hessian is a scalar multiple $\frac{a_{n+1}}{a_1}$ of the first row, we can define a matrix B_{n+1} which is the reduction of the Hessian in the same manner as was demonstrated for $|I| = 2$. Here, B_{n+1} has the same first row as the Hessian and the following n rows are entirely zeros. The determinant of B_{n+1} is equal to the determinant of the Hessian. As B_{n+1} is an upper triangular matrix, its determinant is the product of its diagonal entries, only one of which is non-zero. Thus, the determinants of both B_{n+1} and the Hessian are 0, and the function is convex for $|I| = n+1$. $\square$

Computational Complexity.

The SWTA problem was known to be complex from its first considerations by Manne. Lloyd & Witsenhausen (1986) proved that the problem is NP-Complete, thus proving that obtaining optimal solutions are not possible in polynomial time and that alternate approaches transformations, heuristics, etc. are required in order to approximate the optimal solutions to larger sized problems.

II. Branch and Bound Algorithm

In this section, we propose a branch and bound algorithm capable of solving the WTA problem. A branch and bound algorithm can be characterized by four components: a bounding rule, a selection rule, a branching rule, and a method for finding an initial feasible solution to increase the efficiency of the algorithm.

For our bounding function, we used MATLABs *fmincon* NLP solver to evaluate an integral relaxation of our objective function using an interior point algorithm.

We use a depth first approach for our selection rule, selecting the node which minimizes the objective function for further searching. While this is less efficient for instances having smaller numbers of decision variables, our problem instances examined are predominantly composed of larger numbers of decision variables, and the breadth first or best first approach would require the storage of information of many nodes compared to the depth first approach, for which the node count is at most the number of decision variables.

For our branching rule, we use dichotomic branching, setting the selected decision variable at each node to 0 or 1 for subsequent searching. We identify the decision variable upon which to branch as the most uncertain decision variable in the relaxed solution. That is, the decision variable whose value is closest to 0.5 is chosen for branching at each node, or

$$(\hat{i}\hat{j}) \in \underset{i=1,\ldots,m,j=1,\ldots,n}{\arg\max} \big\{min\{x_{ij},(1-x_{ij})\}\big\}.$$

Our branch and bound algorithm is depicted in Figure 1. The algorithm is initialized by finding a feasible solution, described in detail later, and defining two variables from this solution. We define a solution x_{best} as the current best feasible solution with

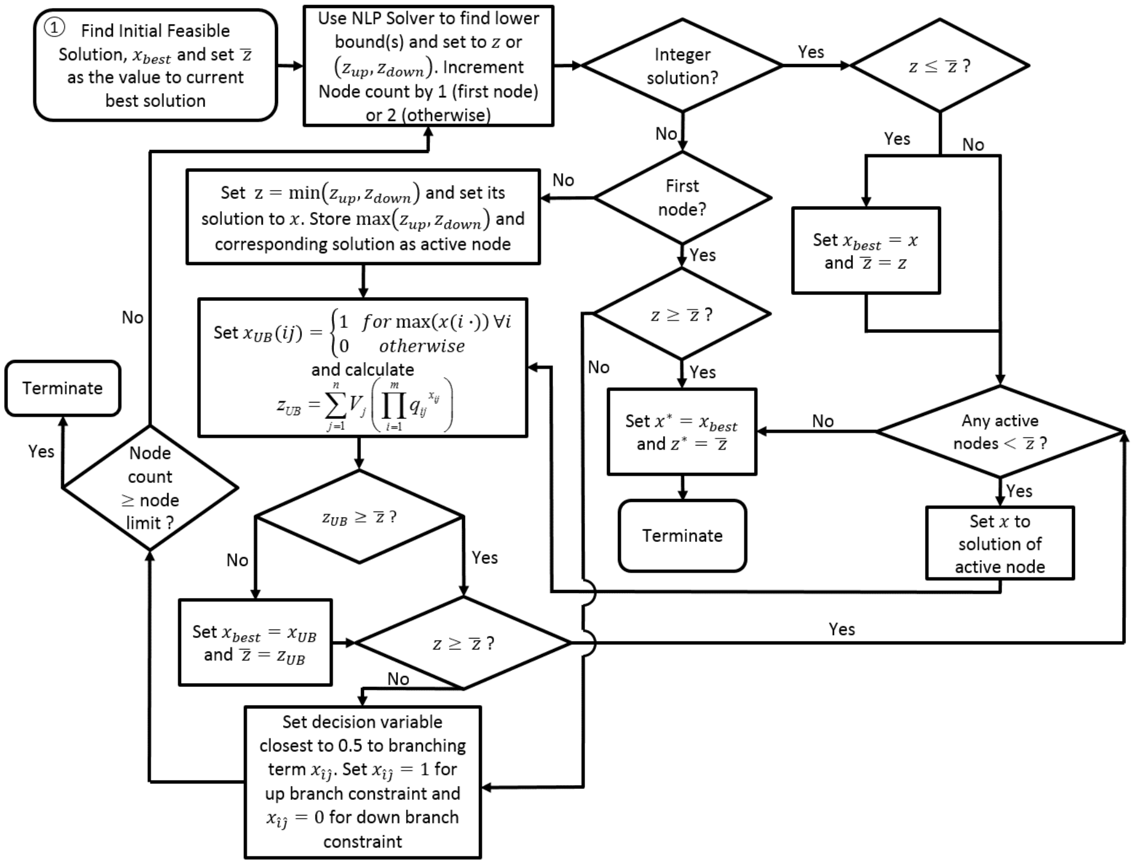

① Find Initial Feasible Solution, x_{best} and set $\bar{z}$ as the value to current best solution
Use NLP Solver to find lower bound(s) and set to z or (z_{up}, z_{down}). Increment Node count by 1 (first node) or 2 (otherwise)
Integer solution?
Yes
$z \leq \bar{z}$?
No
Yes
No
First node?
No
Set $z = \min(z_{up}, z_{down})$ and set its solution to x. Store $\max(z_{up}, z_{down})$ and corresponding solution as active node
Set $x_{best} = x$ and $\bar{z} = z$
Yes
$z \geq \bar{z}$?
No
Set $x_{UB}(ij) = \begin{cases} 1 & for \max(x(i\,\cdot))\,\forall i \\ 0 & otherwise \end{cases}$ and calculate $z_{UB} = \sum_{j=1}^{n} V_j \left(\prod_{i=1}^{m} q_{ij}^{x_{ij}} \right)$
No
Yes
Terminate
Set $x^* = x_{best}$ and $z^* = \bar{z}$
No
Any active nodes $< \bar{z}$?
Yes
Node count $\geq$ node limit ?
Yes
$z_{UB} \geq \bar{z}$?
Terminate
Set x to solution of active node
No
Yes
Set $x_{best} = x_{UB}$ and $\bar{z} = z_{UB}$
$z \geq \bar{z}$?
Yes
No
Set decision variable closest to 0.5 to branching term x_{ij}. Set $x_{ij} = 1$ for up branch constraint and $x_{ij} = 0$ for down branch constraint

Figure 1. Branch and Bound Algorithm

objective value $\bar{z}$ throughout the algorithm and we set the initial feasible solution to x_{best} and its objective value to $\bar{z}$. We then use MATLAB's nonlinear solver *fmincon* to find the integer relaxed lower bound to the problem, with current solution x and current objective value z. If the solution is integer, we compare the objective value of the lower bound to $\bar{z}$. If $z \leq \bar{z}$, we set $x_{best} = x$ and $\bar{z} = z$. Otherwise, we do not update the initial feasible solution. We then set our optimal solution $x^* = x_{best}$ with objective value $z^* = \bar{z}$ and terminate the algorithm.

If the solution we get from MATLAB's *fmincon* is not integer, we compare the objective value of the lower bound to the initial feasible solution. We know that applying constraints to the problem will not decrease its objective function value, so if $z \geq \bar{z}$, we set our optimal solution $x^* = x_{best}$ with objective value $z^* = \bar{z}$ and terminate the algorithm. If $z < \bar{z}$, we define a branching term as previously defined.

We use *fmincon* to solve two subproblems with the branching term constrained to 1 and 0, and define solutions x_{up} and x_{down} with values z_{up} and z_{down}. If neither solution is integer, we define $z = min(z_{up}, z_{down})$ and x as the solution for z. We store the other solution and its objective value to an active node. If either x_{up} or x_{down} is integer, we compare the objective value to $\bar{z}$, and we store the non-integer solution and its objective value to an active node. If $z \leq \bar{z}$, we respectively redefine our x_{best} and $\bar{z}$ terms with the integer solution and its objective value. We then compare the objective value of any active nodes to $\bar{z}$. If the objective value to any active node is $\geq \bar{z}$, we set our optimal solution $x^* = x_{best}$ with objective value $z^* = \bar{z}$ and terminate the algorithm. Otherwise, we use the solution of the active node to define

$$x_{UB}(i,j) = \begin{cases} 1 & max\,(x(i,\cdot)) \;\; \forall i = 1, \ldots, m \\ 0 & \text{otherwise} \end{cases}$$

We set our upper bound objective value z_{UB} to the solution of the objective function using x_{UB}. If the objective value of the upper bound is less than that of the current best objective value, $z_{UB} < \bar{z}$, we update $x_{best} = x_{UB}$ and $\bar{z} = z_{UB}$. Otherwise, we do not update these terms. We then check to see if $z < \bar{z}$. If not, we check to see if the value of any active nodes is less than $\bar{z}$ and follow as we described previously. Otherwise, the algorithm continues as previously defined by setting the branching term.

We set an iteration count termination criteria for the algorithm, as the number of nodes the algorithm explores typically exceeds one million for the problem with 10 weapons and 20 targets, and the number of nodes increases exponentially with the increase of the problem size.

Branch and bound algorithms benefit greatly from an initial feasible solution from which an objective function value can be used to fathom nodes immediately. Without this initial feasible solution, the algorithm is unable to fathom any nodes until a feasible solution is reached. We obtained our initial feasible solution for block 1 in Figure 1 by implementing a strategy first explored by Castañon (1987) and presented by Ahuja *et al.* (2007) in modeling the problem as a minimum cost network flow problem. We define a network $G = (N, A)$ as depicted in Figure 2 by having nodes that correspond to our weapons (W_i) and a node for each weapon-target pairing (T_j) as well as a single terminus (t) which has demand equaling the total number of weapons, or $\sum_{i=1}^{n} w_i$. As can be seen in Figure 2, we divide the set of all arcs into two subsets: A_1 is the set of all arcs from each weapon to each target and A_2 is the set of all arcs from each target to the terminus. Each target has n nodes, $T_j^1, T_j^2, \ldots, T_j^n$, which define the number of weapons assigned to it and the order in which each is assigned. The arcs connecting each weapon to nodes T_j^1 correspond to the first weapon assigned to target j and have an associated cost $V_j p_{ij}$. The remaining arcs cannot be calculated exactly as their cost is dependent on all preceding assignments. That is, the cost of an arc connecting a weapon to node T_j^2 is $V_j p_{ij} q_{\hat{i}\hat{j}}$. Since this heuristic is used to provide a feasible upper bound to the problem, we assume each previous assignment was made with the weapon having the lowest p_{ij} value for target j. This means that we estimate the cost of the arc connecting a weapon to node T_j^2 to be $V_j p_{ij} q_{ij}^{max}$. In defining the arcs connecting each weapon to node T_j^2 in this way, we assure that the true cost of this arc is no worse than our approximation. We can therefore define each arc $(i,j)\in A_1$ as having an associated cost $c_{ij} = V_j p_{ij} \left(q_j^{max}\right)^{k-1}$, where k is the index defining the order of each weapon to target assignment $T_m^k, k = 1, \ldots, n$. Each arc $(j,t)\in A_2$ has cost $c_{jt} = 0, \forall j\in J$. Each arc in A has capacity 1, which is important as it limits the number of weapons assigned to T_j^1 to 1, preventing multiple assignments along maximum valued arcs. Since no hard side-constraints exist, this heuristic generates a feasible solution by using a greedy approach by maximizing the expected value obtained by assigning a weapon to a target while penalizing multiple assignments to the same target.

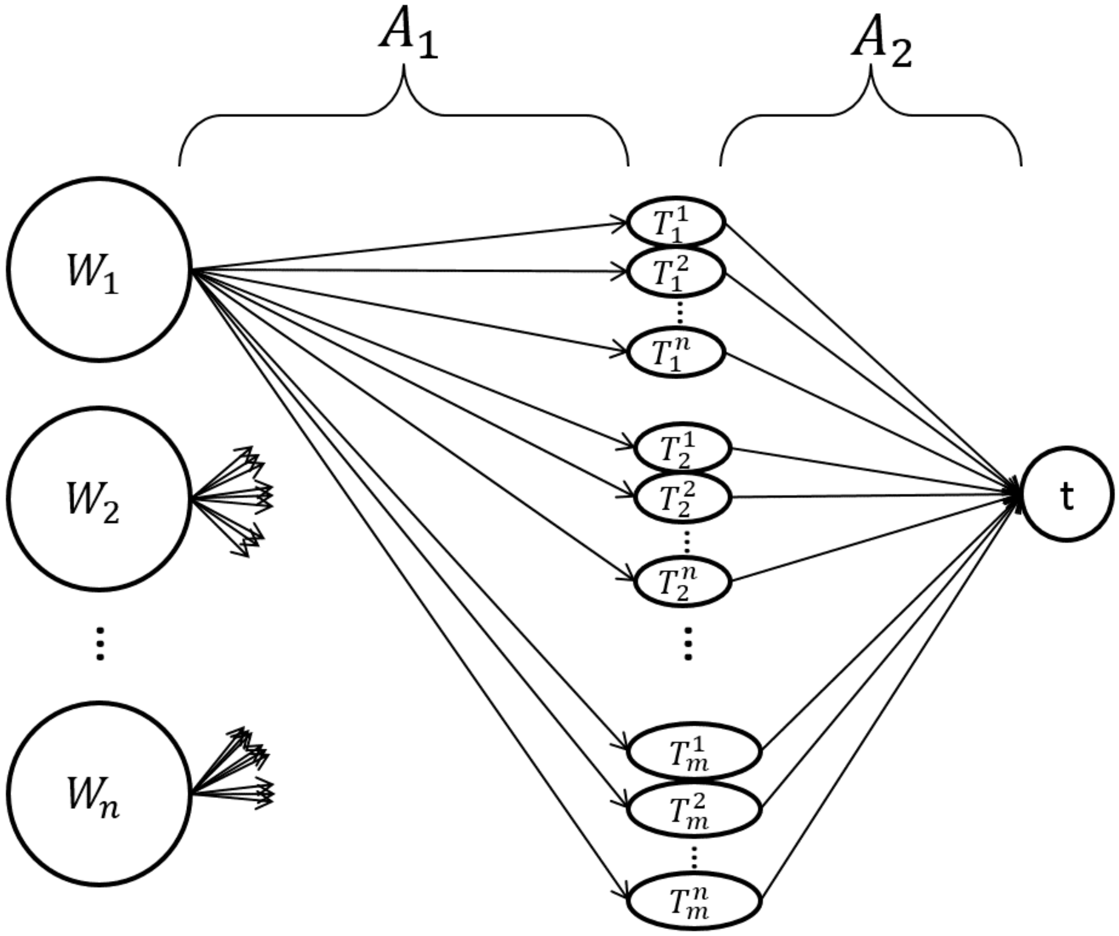


Figure 2. Minimum Cost Network Flow Lower Bounding Scheme

We were only able to run to completion the branch and bound algorithm for smaller problem instances due to the large number of decision variables and correspondingly large trees produced during the execution of the algorithm. The value of the algorithms efficiency as compared to a brute force algorithm is shown in §4.

III. Heuristics

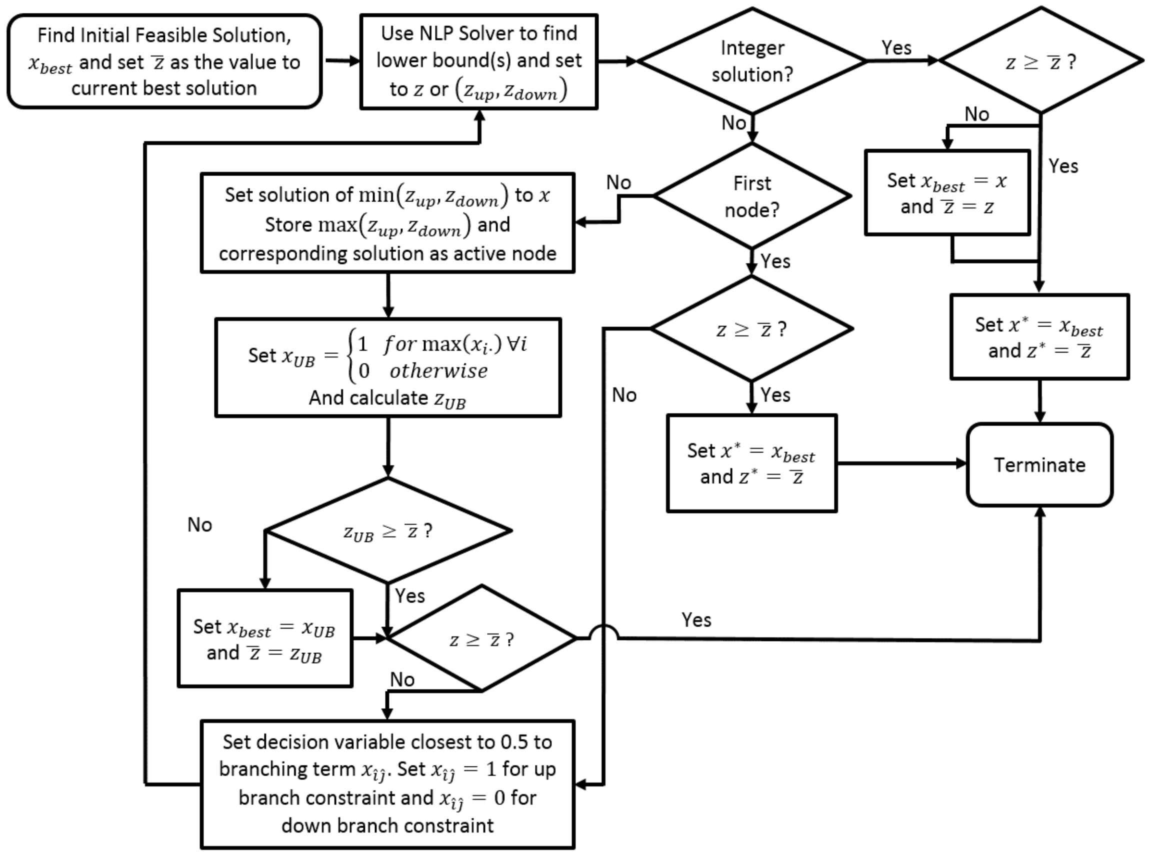
As mentioned in §1, the WTA problem is NP Complete and obtaining an optimal solution to larger sized problems is not practical. We sought to develop and compare heuristic search techniques to obtain near optimal solutions to larger sized instances of the WTA problem in minimal computational time. We developed two heuristics, a Greedy Branch and Bound Heuristic and a Modified Quiz Problem Search Heuristic, to which we compare the effective Construction Heuristic as developed by Ahuja *et al.* (2007).

Minimum Cost Flow Formulation Based Construction Heuristic.

We considered a heuristic developed by (Ahuja *et al.*, 2007) which utilized the minimum cost network flow method of determining an upper bound. This borrows from the network programming heuristics proposed by Chang *et al.* (1987), who developed an iterative linear network programming heuristic and Castañon (1987), who developed a nonlinear network flow heuristic. The Construction Heuristic first solves the WTA problem as described in §2, identifying the assignments which give an upper bound to the solution. The arcs of this network fall into two categories: exact-cost arcs, which are computed with known values, and approximate-cost arcs, which are computed with the maximum survival assignment as upper bounds. The heuristic affixes the assignments to define a positive flow on the exact-cost arcs and redefines the network by removing the exact-cost arcs with positive flow and updates the remaining arc costs by using the updated target values, $V_j^1 = V_j q_{\hat{i}\hat{j}}$. This is similar to the maximum marginal return approach first proposed by denBroeder *et al.* (1959). In redefining the network, some of the approximate-cost arcs become exact-cost arcs. The process is repeated until all weapons have been assigned.

Greedy Branch and Bound Heuristic.

We develop a heuristic based upon our branch and bound algorithm which solves the static WTA with small optimality gaps. We utilize the depth first approach to our branch and bound algorithm, selecting the node with the lowest integral relaxed solution as it has the lower bound. This heuristic is different from traditional branch and bound algorithms in its termination criteria. We design the heuristic to terminate at the arrival of the first feasible solution and do not consider any active nodes beyond this point.



Find Initial Feasible Solution, x_{best} and set $\overline{z}$ as the value to current best solution
Use NLP Solver to find lower bound(s) and set to z or (z_{up}, z_{down})
Integer solution?
Yes
No
$z \geq \overline{z}$?
No
Yes
First node?
No
Yes
Set $x_{best} = x$ and $\overline{z} = z$
Set solution of $\min(z_{up}, z_{down})$ to x Store $\max(z_{up}, z_{down})$ and corresponding solution as active node
Set $x_{UB} = \begin{cases} 1 & for \max(x_{i\cdot})\ \forall i \\ 0 & otherwise \end{cases}$ And calculate z_{UB}
$z \geq \overline{z}$?
No
Yes
Set $x^* = x_{best}$ and $z^* = \overline{z}$
Set $x^* = x_{best}$ and $z^* = \overline{z}$
Terminate
No
$z_{UB} \geq \overline{z}$?
Yes
Set $x_{best} = x_{UB}$ and $\overline{z} = z_{UB}$
$z \geq \overline{z}$?
Yes
No
Set decision variable closest to 0.5 to branching term $x_{\hat{i}\hat{j}}$. Set $x_{\hat{i}\hat{j}} = 1$ for up branch constraint and $x_{\hat{i}\hat{j}} = 0$ for down branch constraint

Figure 3. Greedy Branch and Bound Heuristic

While similar to the Branch and Bound Algorithm, the Greedy Branch and Bound Heuristic, shown in Figure 3, differs in its termination criteria. The first criterion upon which this heuristic will terminate is when an integer solution is found using the NLP solver. If the value of the solution is greater than or equal to the current best value, the heuristic is terminated, reporting the current best solution and value. Otherwise, the current best solution and objective value are first set to the integer solution and its objective value before the heuristic is terminated. The second criterion which will terminate the heuristic is if the value of the lower bound(s) calculated by the NLP solver is(are) greater than the current best objective value. In this case, the current best solution and objective value are reported.

Modified Quiz Problem Search Heuristic.

We developed a heuristic based upon the application of the optimal quiz problem solution as a heuristic which was first proposed by Ahner (2005). The quiz problem states that an individual presented with a series of questions $u = 1, \ldots, n$ with value v_u has probability p_u of correctly answering a question. Further, the individual will be able to answer questions until he answers one incorrectly. The objective of the quiz problem is to identify the order in which to select the questions in order to maximize the value of those correctly answered. Bertsekas & Castañon (1999) showed that the strategy for maximum return is to select the questions in descending values of y_u, where $y_u = \frac{v_u p_u}{1-p_u} = \frac{v_u p_u}{q_u}$.

We use Ahner's strategy to define the value of each weapon-target assignment as y^0_{ij}, which allows us to select the maximum return for a weapon-target assignment $x_{\hat{i}\hat{j}}$. We then redefine our target value $V_{\hat{j}} = V_{\hat{j}} q_{\hat{i}\hat{j}}$, which is the residual value of the selected target given the weapon assigned. We also redefine our probabilities of kill for the selected weapon $w_{\hat{i}}$ to be $p_{\hat{i}\cdot} = 0$, since we have only one weapon of each type. Using these updated values, we update our value array as y^1_{ij}. We repeat this process until each of the weapons is assigned to a target.

Figure 4 shows the Modified Quiz Problem Search Heuristic. The heuristic initializes by setting the assignment solution, x, to an n x m zeros matrix for a problem instance having n weapons and m targets. As referenced above, we build a value array y, which is defined by

$$y(i,j) = \frac{V_j p_{ij}}{q_{ij}}.$$

We then identify the two largest values in y, which we call $y(i_1, j_1)$ and $y(i_2, j_2)$,and check if they are in the same row of y. If not, we check to see if the greedy assignment is preferred according to a process defined in the next paragraph. If the largest values are in the same row of y, we set $x(i_1, j_1) = 1$ and increment a counter k by 1. If our counter is equal to the number of weapons after this increment, we terminate the heuristic. Otherwise, we update our value array y by redefining $V_{j_1} = V_{j_1} q_{i_1 j_1}$ and setting the probabilities of weapon i_1 to zero, or $p_{i_1 j} = 0 \ \forall j$.

We note a shortcoming with the Construction Heuristic and other greedy selection based heuristics that needs to be addressed. These heuristics seek to define pairings based upon the greatest expected value of the weapon-target assignment, that is $V_j p_{ij}$, which is in line with a greedy algorithm. The shortcoming in this approach can be illustrated by taking a trivial case with two weapons and two targets, with target values $\{V_1, V_2\}$ and probabilities of kill $\{p_{11}, p_{12}, p_{21}, p_{22}\}$. If we define the greatest expected value to be

$$max \left(\begin{bmatrix} V_1 p_{11} & V_2 p_{12} \\ V_1 p_{21} & V_2 p_{22} \end{bmatrix} \right) = V_1 p_{11}$$

And we further assume that, for this case,

$$V_1 p_{11} - V_2 p_{12} < V_1 p_{21} - V_2 p_{22}$$

We can see that, with a simple rearrangement of the above inequality

$$V_1 p_{11} + V_2 p_{22} < V_1 p_{21} + V_2 p_{12}$$

This means that selecting the pairing with the greatest expected value will result in a lower solution value than selecting the alternative. We incorporate into the Quiz Problem Search Heuristic a step which checks whether

$$y_{ab} - \max_{j \neq b}\{y_{aj}\} < y_{cd} - \max_{j \neq d}\{y_{cj}\}$$

where $y_{ab} > y_{cd} > \ldots > \min_{i,j}\{y_{ij}\}$. In the case where the above inequality holds true, we chose y_{cd} rather than y_{ab} as our assignment for that iteration.

We only incorporated this modification into the Quiz Problem Search Heuristic for several reasons. The Greedy Branch and Bound Heuristic does not select assignments based upon a defined value but rather solves the problem using a branch and bound technique and terminates at the first feasible solution. This greedy shortcoming modification cannot be implemented in such a heuristic. The Construction Heuristic, which is the benchmark to which we compare our heuristics, should not be modified since its performance as published by Ahuja *et al.* (2007) do not use this modification. Therefore, only the Quiz Problem Search Heuristic can be modified by this method to avoid the shortcoming of a greedy selection criterion.

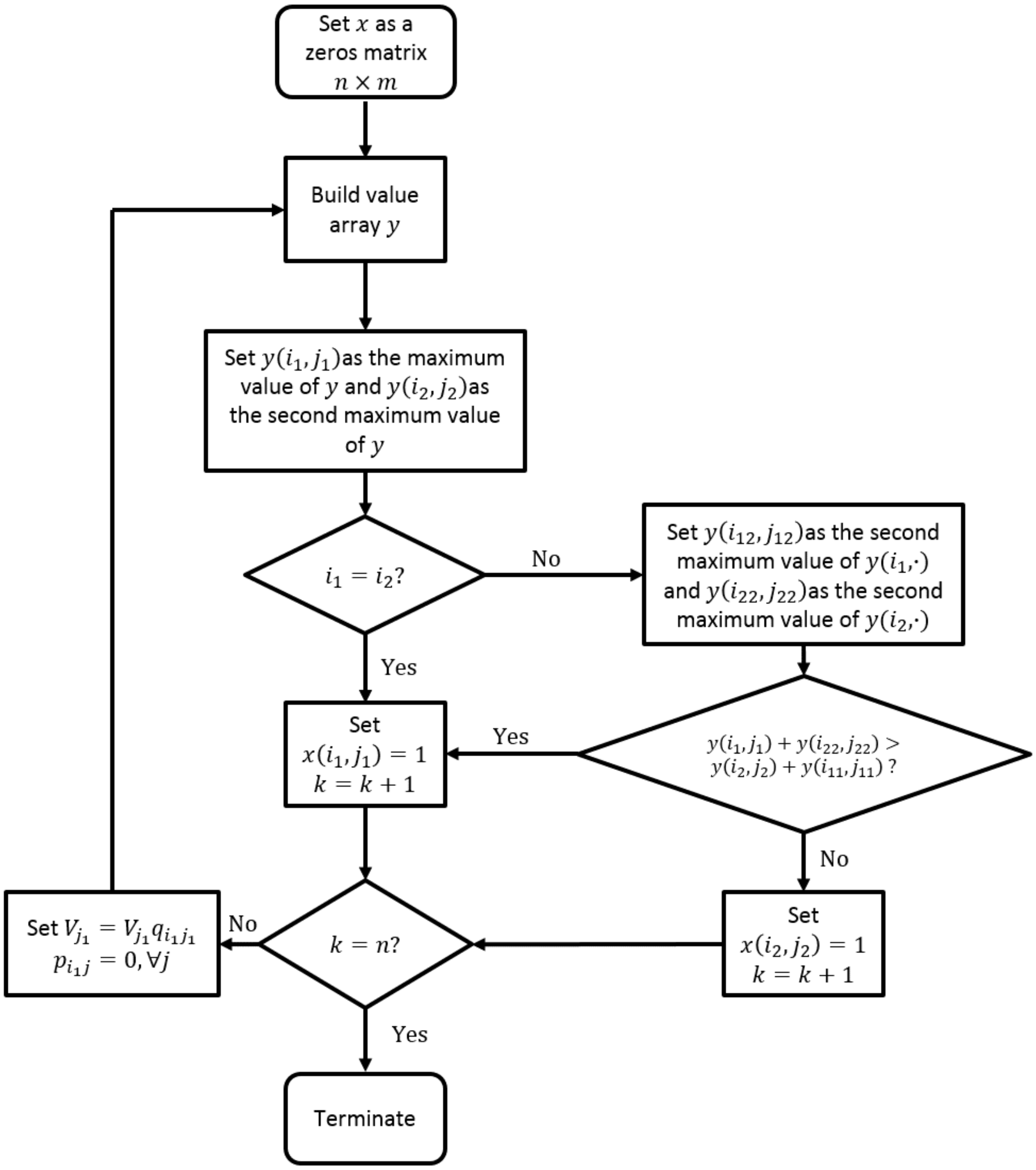

Set x as a zeros matrix $n \times m$
Build value array y
Set $y(i_1, j_1)$ as the maximum value of y and $y(i_2, j_2)$ as the second maximum value of y
$i_1 = i_2$?
No
Set $y(i_{12}, j_{12})$ as the second maximum value of $y(i_1, \cdot)$ and $y(i_{22}, j_{22})$ as the second maximum value of $y(i_2, \cdot)$
Yes
Set $x(i_1, j_1) = 1$ $k = k + 1$
Yes
$y(i_1, j_1) + y(i_{22}, j_{22}) > y(i_2, j_2) + y(i_{11}, j_{11})$?
No
Set $V_{j_1} = V_{j_1} q_{i_1 j_1}$ $p_{i_1 j} = 0, \forall j$
No
$k = n$?
Set $x(i_2, j_2) = 1$ $k = k + 1$
Yes
Terminate

Figure 4. Modified Quiz Problem Search Heuristic

IV. Computational Results

We test each algorithm and heuristic defined heretofore to solve a set of instances by designing random parameters within various instance sizes ranging from 5 weapons and 5 targets to 80 weapons and 160 targets, and by testing each heuristic and algorithms on the same parameters. We consider 15 problem sizes, first defined in Table 1, which set our number of weapons and targets. We assign a randomly generated target value as a uniformly distributed continuous variable $[25, 100]$, and we also assigned randomly generated probabilities of kill as uniformly distributed continuous variables $[0.6, 0.9]$ so that each weapon has a different probability of kill for each target. We generate 20 problem instances of random numbers for each of our 15 problem sizes and performed all tests on an Intel Xeon E5-2650 v2 processor computer with 128 GB RAM PC. Each solution method is applied to the same set of 20 problems for the purpose of enabling direct comparisons of solution values and computational times. Our results, insights, and analysis are presented in the following sections.

Computational Complexity.

In this thesis, we consider problem instances of various sizes, as shown in Table 1 below. The first set of instances, each with five weapons and five targets, results in $5^5 = 7776$ permutations from which one is the optimal solution. This is a relatively simple problem to solve via full enumeration and each of the 20 randomly generated instances took an average of approximately 0.6 seconds to solve. However, the next problem considered has $6^{10} = 60466176$ permutations, which takes an average of 2,302 seconds (more than 38 minutes) to solve on an Intel Xeon E5-2650 v2 processor computer with 128 GB RAM PC. Clearly, the number of permutations and subsequent computational time to find a solution by fully enumerating the permutations quickly becomes excessive and exceeds computational capacity. We ran all permutations of the enumeration algorithm from 5 weapons and 5 targets to 9 weapons and 9 targets and developed a model to project the computational time required to solve the larger problem instances, as we could not run them to completion due to required time. Our model has an adjusted R^2 value of 0.99 and takes the form

$$f(w,t) = -1319.157 + 203.97w - 0.0000407t^w + 0.0000792wt^w$$

The computational times for the first two problems in Table 1 are the averages of 20 experiments and the remaining 13 computational times are projections from our model to estimate time required for larger instances. For a 20 weapon, 20 target instance, full enumeration is estimated to take 3.703×10^{15} years.

Table 1. Projected Computational Time Using Full Enumeration

Weapons	Targets	Time (sec)
5	5	0.605
10	5	2302.045
10	10	7.510E+06*
10	20	7.690E+09*
15	10	1.147E+12*
20	10	1.543E+17*
20	20	1.618E+23*
20	40	1.696E+29*
40	10	3.126E+37*
40	20	3.437E+49*
40	40	3.779E+61*
40	80	4.155E+73*
80	40	9.197E+125*
80	80	1.112E+150*
80	160	1.344E+174*

* Model projections

Branch and Bound results.

We present the results of our branch and bound algorithm in this section in two metrics. The first is the computational time required for our branch and bound algorithm as compared to the full enumeration, or brute force, algorithm. Due to the large number of permutations that accompany even moderate sized problems, the brute force algorithm is cannot solve even these moderate sized problems within a practical amount of time (e.g., less than a week). We are unable to solve any problem size greater than 10 weapons and 5 targets using the brute force algorithm; even the problem of this small size took an average of more than 38 minutes to solve. We therefore test the performance of the brute force algorithm for various sized problems up to 9 weapons and 9 targets and generate a model to predict the time required to solve via brute force algorithm.

Table 2. Computational Time Requirements (sec)

		Branch & Bound		Brute	
Weapons	Targets	Average	St Dev	Average	St Dev
5	5	1.739	1.738	0.605	0.0519
10	5	32.025	55.669	2302.045	44.460
10	10	507.942	831.781	7.510E+06	-
10	20	1429.216	137.236	7.690E+09	-
15	10	580.712	65.902	1.147E+12	-

We observe that the computational time required for the Branch and Bound algorithm is an improvement on that of the brute force algorithm, which is computationally intractable when considering problems having more than 10 weapons and 10 targets. Table 2 shows the computational improvements of the branch and bound algorithm as compared to the brute force algorithm. The first two problem instances were run to completion with the brute force algorithm, and the solutions generated match those of the branch and bound algorithm. The remaining times for the brute force algorithm are projected from our brute force algorithm computational time model and therefore have no standard deviation. With exception of the modeled projections, these figures were generated by averaging the computational requirements of the same 20 randomly generated instances of each problem size. Figure 5 shows the time required or estimated for the three problems with 10 weapons and varying numbers of targets for the brute force algorithm as compared to the performance of our Branch and Bound algorithm. While the Branch and Bound Algorithm grows with problem size, it appears to grow at a much lesser rate than full enumeration.

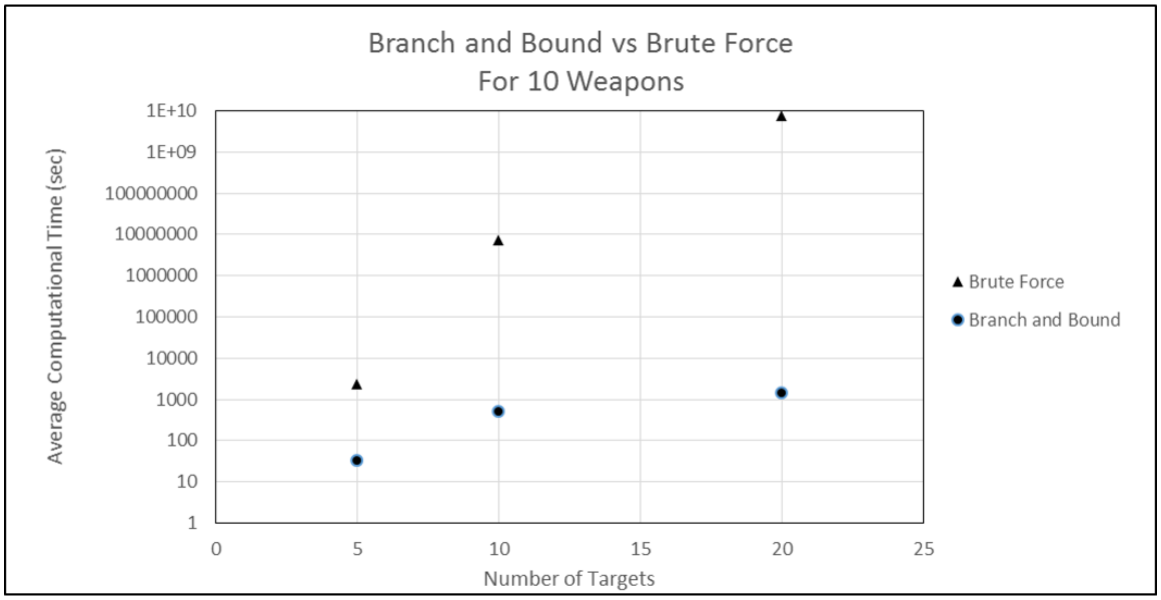


Figure 5. Computational Time Requirements for Optimization With 10 Weapons

We were unable to run to completion the Branch and Bound Algorithm for larger sized problems due to two prevailing factors. The first factor was the vast number of nodes generated when running the algorithm. For the problem of 10 weapons and 20 targets, our algorithm was unable to find the optimal solution after considering 1,000,000 nodes, to include nodes which were fathomed without exploration. The lag due to the large number of nodes was exacerbated for larger problems by the computational time required for the NLP solver to find the integer relaxed solution at each node. With smaller sized problems, an optimal solution could be found using an interior point algorithm in less than a tenth of a second, but as the problem sizes increased, the time required for each optimization increased. For the same problem of 10 weapons and 20 targets, the time for each integer relaxed optimization took several seconds, which therefore causes the algorithm to run for long durations when considering all nodes. As such, we decided to run the larger sized problems with a ceiling on the maximum number of nodes generated and to observe the gap between the best feasible solution, or upper bound, with the lowest relaxed solution, or lower bound, to an active node as depicted in the far right columns of Table 3. From this, we were able to identify that our algorithm was, at worst, within a specific percentage of the optimal solution, though the exact optimal solution is not known. Table 3 shows the performance of the algorithm for the small and moderate sized problems with a maximum number of nodes set to 1000.

Table 3. Branch and Bound Algorithm Performance

		Branch & Bound					
Weapons	Targets	Nodes Visited		Time (sec)		Within % optimality	
		Average	St Dev	Average	St Dev	Average	St Dev
5	5	14.7	18.184	1.739	1.738	-	-
10	5	129.4	232.118	32.025	55.669	-	-
10	10	905.5	1466.203	507.942	831.781	-	-
10	20	1000	0	1429.216	137.236	0.311	0.0476
15	10	1000	0	580.712	65.902	0.246	0.0536
20	10	923.3	229.487	787.678	438.806	0.0630	0.0448
20	20	1000	0	5079.692	586.856	0.0547	0.0188

We can extract several insights from Table 3. The first is that the computational time increases quickly with regard to the problem size, both in numbers of weapons and targets, although it increases faster relative to target number increase. However, we observe that the average computational time for a problem with 10 weapons and 20 targets is nearly double the average computational time with 20 weapons and 10 targets. This occurs because the number of dimensions across which the weapons are to be assigned is equal to the number of targets, and so each relaxed problem takes longer to solve. Next, we see that the average maximum gap with the optimal solution appears to vary greatly. This result occurs because an integer feasible solution does not allow for a weapon to have partial allocation to multiple targets as the lower bounding solution does. That is, the lower bounding solution will likely assign fractions of each weapon to the targets, thus providing an optimal coverage of all targets and minimizing leakage. In contrast, feasible solutions may leave targets unassigned in general but will necessarily leave targets unassigned for problems with more targets than weapons or those with a remainder after dividing the number of weapons by the number of targets. As the objective function evaluates the sum of residual expected target values, or expected leakage, the feasible solution to these problems, which leaves targets unassigned, will have much larger values relative to the lower relaxed solutions than the feasible solutions to problems wherein the number of targets is greater than or equal to the number of weapons or there is no remainder after dividing the number of weapons by the number of targets. Lastly, the high standard deviations of solution time and node count where the algorithm ran to completion illustrate that the performance of the Branch and Bound algorithm will vary greatly due to the uncertain number of nodes required for the algorithm to find the optimal solution.

Comparison of Heuristics.

Next, performance of the Construction Heuristic developed by Ahuja *et al.* (2007) is compared to our Greedy Branch and Bound Heuristic and our Modified Quiz Problem Search Heuristic.

A note on Ahuja *et al.* (2007).

We note a statement by the authors regarding the performance of their branch and bound algorithm (i.e., the Construction Heuristic):

> For [problems with more than 80 weapons and 80 targets], the branch-and-bound algorithm could not be executed until optimality. However, observe that the minimum of the lower bounds of the active (node not pruned yet) nodes gives an overall lower bound on the objective function.

In contrast to this statement, the authors present results in Table 3 of their work the results of their testing, which include 0% relative optimality gaps attained for several of the instances, including four of the eight largest instances tested. This result contradicts the aforementioned statement; assuming the first statement to hold, we conjecture that the authors employed an additional measure to identify the solution found via their Construction Heuristic to be optimal for the larger instances.

Upon further investigation, the Ahuja *et al.* (2007) do additionally state

> We also observed that when [the number of weapons] < [the number of targets], then the number of weapons assigned to most targets is at most one in the optimal solution.

Such a statement does not require, when $W < n$, that *no* target may have more than one weapon assigned to it in an optimal solution. However, this is the only statement the authors provide that intimates at method that could be employed to conclude that a solution attained via the Construction Heuristic for a larger instance is optimal. We suspect that such a solution characteristic was observed for smaller problem instances and assumed to hold for larger problem instances. We have contacted the authors for clarification and look forward to their reply.

The following example demonstrates that an optimal solution may have more than one weapon assigned to a given target when $W < n$, setting aside the possibility of using such a "typically observed" property to draw conclusions regarding the optimality gap attained via the Construction Heuristic by examining the characteristics of the solution. Take a scenario wherein there are three targets and two weapons. Suppose that the values of these targets are

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 100 \\ 25 \\ 25 \end{bmatrix},$$

and the probability of kill for each weapon to each target is 0.6. With identical weapons, we can now follow the maximum marginal return approach, assigning weapons one at a time. Since we are minimizing the residual target value, the best assignment for weapon 1 is target 1, resulting in the target values

$$\begin{bmatrix} V_1^1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 40 \\ 25 \\ 25 \end{bmatrix}$$

Weapon 2 should also be assigned to target 1 since it results in a lesser cumulative residual value than assigning either the first, second, or both weapons to the other targets. For example, if we consider the objective solution in situations where both weapons are assigned to target 1, one is assigned to target 1 and one to target 2, one to target 1 and one to target 3, and one is assigned to target 2 and one to target 3, we see that the best objective value occurs where both weapons are assigned to target 1.

$$100(0.6)^1(0.6)^1 + 25(0.6)^0(0.6)^0 + 25(0.6)^0(0.6)^0 = 66$$

$$100(0.6)^1(0.6)^0 + 25(0.6)^0(0.6)^1 + 25(0.6)^0(0.6)^0 = 75$$

$$100(0.6)^0(0.6)^1 + 25(0.6)^1(0.6)^0 + 25(0.6)^0(0.6)^0 = 75$$

$$100(0.6)^0(0.6)^0 + 25(0.6)^1(0.6)^0 + 25(0.6)^0(0.6)^1 = 120$$

This procedure seems to influence the perception of the construction heuristic's performance, which was noted by the authors as finding the optimal solution whenever the number of weapons was less than the number of targets. In Table 4, we show that both our Greedy Branch and Bound and Modified Quiz Problem Search heuristics improve upon the construction heuristic in most instances wherein this condition holds. We believe that the degradation of performance for the Construction Heuristic can be attributed to either the omission of solutions having more than one weapon assigned to a target or due to the error inherent to the piecewise approximation which was necessary when applying the linear transformation to the objective. As our solution was not based upon a tranformation, our optimality gap is exact, whereas the optimality gap using the approximated optimal solution for the transformed problem may inaccurately indicate an optimal solution.

We show in Table 4 the number of the 20 same randomly generated instances for each problem size where each heuristic dominates in terms of objective function value. In some cases, more than one heuristic dominates. We represent performance in terms of relative response rather than optimality gap. Additionally, our most accurate heuristic, the Greedy Branch and Bound Heuristic, is a slower heuristic which is dependent upon the speed with which our nonlinear solver is able to solve each node's relaxed problem. As the problem size increases, both the number of nodes and the size of the NLPs grow, increasing the computational time and preventing the heuristic's implementation for large problems ($W > 40 \,\&\, T > 40$). This is reflected in Table 4 by the absence of Greedy Branch and Bound Heuristic performance results for larger problem sizes.

Table 4. Best Solution Comparison of Heuristics

Weapons	Targets	Modified Quiz Problem Search	Construction	Greedy Branch & Bound
5	5	8	16	20
10	5	5	14	18
10	10	6	9	17
10	20	5	5	20
15	10	5	8	12
20	10	1	10	14
20	20	0	5	18
20	40	11	3	20
40	10	0	8	14
40	20	1	2	18
40	40	1	19	-
40	80	16	4	-
80	40	1	19	-
80	80	2	18	-
80	160	20	0	-

Table 5 shows the percent optimality gap of the heuristics where we are able to execute the branch and bound algorithm to completion, thereby obtaining valid lower bounds and calculating optimality gaps. We observe that our Greedy Branch and Bound Heuristic is the most accurate and reliable of the heuristics, with a maximum average relative optimality gap of 0.53%. We further note that, if we examine Table 6, the computational time required for each heuristic to achieve the optimality gaps shown in Table 5 are comparable. The Greedy Branch and Bound Heuristic takes roughly seven seconds longer than the Construction Heuristic in the problem having 10 weapons and 10 targets but yields a superior optimality gap. Additionally, we believe that the difference in the observed performance of the construction heuristic compared to the results given by Ahuja *et al.* (2007) and those that we observed arise from the comparison to an optimal solution that was incorrectly identified via the approximation.

Table 5. Percent Optimality Gaps of Heuristic Solutions to Smaller Problems

		MQP Search		Construction		Greedy B&B	
Weapons	Targets	Average	St Dev	Average	St Dev	Average	St Dev
5	5	4.789%	7.466%	0.460%	2.058%	0.000%	0.000%
10	5	5.244%	4.560%	1.808%	3.488%	0.531%	1.242%
10	10	5.266%	5.237%	3.138%	3.429%	0.383%	1.267%

While Table 5 shows a relatively poor performance of the Modified Quiz Problem Search Heuristic for small problems, we point to its performance as being exceptionally strong when considering the computational time required for even the largest problems as shown in Table 6. Regardless of the size of the problem, the computation time of the Modified Quiz Problem Search Heuristic was dominant relative to the Construction and Greedy Branch and Bound heuristics. We observed that, though it was the accurate heuristic when optimality gap was considered, the Greedy Branch and Bound heuristic was the most computationally demanding of the heuristics. We were able to run to completion the Greedy Branch and Bound Heuristic for the 40 weapons and 20 targets problem, but for larger problems the heuristic took far too long (i.e., over 24 hours) and wasn't run to completion. This increase in computational time was the result of the time required to solve the integer relaxation for each node of the Greedy Branch and Bound Heuristic. The Construction heuristic was an efficient and reliable technique for certain problem sizes, but it was neither the most accurate nor the most efficient when considering problems with more targets than weapons, especially with larger problem instances.

For all sized problems, we consider the performance of the Modified Quiz Problem Search and Construction heuristics by observing the relative performance of each averaged over the 20 instances of each problem. Table 7 shows that, when the Modified Quiz Problem Search Heuristic was inferior to the Construction Heuristic, its solution was at most 7% greater than that of the Construction Heuristic.

Table 6. Heuristic Computational Time Comparisons (sec)

		MQP Search		Construction		Greedy B&B	
Weapons	Targets	Average	St Dev	Average	St Dev	Average	St Dev
5	5	0.00333	0.01064	0.224	0.0976	0.568	0.257
10	5	0.00113	0.000176	0.445	0.0917	2.019	0.674
10	10	0.00111	0.000121	0.441	0.0975	7.657	2.726
10	20	0.00110	0.000105	0.487	0.107	242.267	55.907
15	10	0.00155	0.000104	0.767	0.0531	38.020	11.086
20	10	0.00202	0.000109	1.608	0.255	41.986	13.726
20	20	0.00215	0.000120	2.208	0.346	232.426	88.660
20	40	0.00280	0.000153	4.455	0.609	28443.757	6393.071
40	10	0.00407	0.000121	17.997	1.863	711.324	223.755
40	20	0.00536	0.000220	20.569	1.181	2691.799	632.036
40	40	0.00670	0.000171	14.311	0.432	-	-
40	80	0.00912	0.000151	47.077	20.162	-	-
80	40	0.0156	0.00121	309.641	10.313	-	-
80	80	0.0194	0.000245	177.106	6.210	-	-
80	160	0.0284	0.000274	1348.815	391.135	-	-

Table 7. Modified Quiz Problem Search and Construction Heuristics Relative Performance

		Heuristic	
Weapons	Targets	MQP Search	Construction
5	5	0.0437	-
10	5	0.0342	-
10	10	0.0211	-
10	20	-	0.0119
15	10	0.0303	-
20	10	0.0689	-
20	20	0.0225	-
20	40	-	0.0195
40	10	0.0720	-
40	20	0.0364	-
40	40	0.0277	-
40	80	-	0.0143
80	40	0.0471	-
80	80	0.0188	-
80	160	-	0.0377

We consider the performance of the Modified Quiz Problem Search Heuristic and the Construction Heuristic from the perspective of statistical significance. Table 8 shows whether the Modified Quiz Problem Search Heuristic or the Construction Heuristic is statistically superior to the other, as computed using the two-tailed t distribution based confidence interval method for the 20 randomly generated problem instances. We built confidence intervals for each heuristic as $\bar{y} \pm t_{\frac{\alpha}{2}}^{(n-1)} \left(\frac{s}{\sqrt{n}} \right)$ to identify whether the confidence interval of the dominant heuristic contained the mean value of the other heuristic. We see that, with $100(1-\alpha)\%$ certainty, neither heuristic consistently dominates the other in performance.

Table 8. Statistically Significant Heuristic Dominance

Weapons	Targets	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.01$
5	5	-	-	-
10	5	-	-	-
10	10	-	-	-
10	20	-	-	-
15	10	-	-	-
20	10	Construction	Construction	Construction
20	20	-	-	-
20	40	-	-	-
40	10	-	-	-
40	20	Construction	-	-
40	40	Construction	Construction	-
40	80	MQP	-	-
80	40	Construction	Construction	Construction
80	80	Construction	Construction	-
80	160	MQP	MQP	MQP

V. Conclusion and Future Research

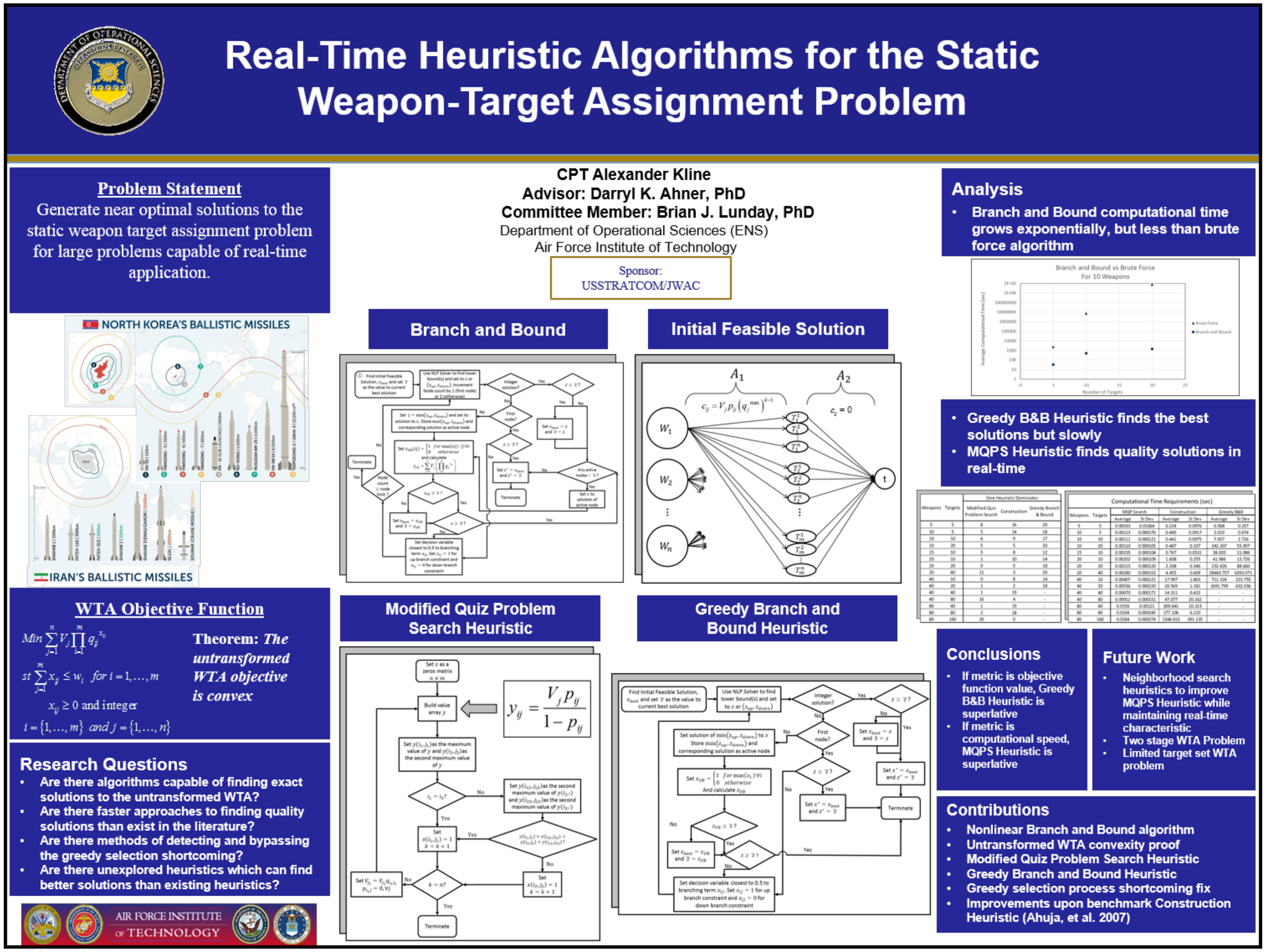
We propose a branch and bound algorithm which was able to find the optimal solution of smaller sizes of the untransformed SWTA problem. We then use this algorithm to generate a search heuristic, the Greedy Branch and Bound Heuristic, which is able to find near optimal solutions that rival those found in the literature. We also generate a Quiz Problem Search Heuristic, which is able to generate close to optimal solutions of even the largest sized problems in less than a few hundredths of a second. While the optimal solution to larger sizes of the SWTA problem remains unsolved, these heuristics allow us to obtain solutions which improve on existing solutions using contemporary methods found in the literature.

We observe that, when able to run to completion, the Greedy Branch and Bound Heuristic usually finds superior solutions than the Construction and Quiz Problem Search Heuristics, although at a much higher computational cost. The reported objective function value of the Greedy Branch and Bound solution is typically but not always superior, and the difference in computational time is at most less than eight seconds for smaller problems ($W \leq 10$ & $T \leq 10$). Because the solutions to small sized WTA problems are found relatively quickly when compared to the Brute Force or full Branch and Bound Algorithms, we consider the solution values found as a comparative metric. As the Greedy Branch and Bound Heuristic has the best average solution value and performed the best for the majority of each of the 20 problem instances for a given number of weapons and targets, we identify it as the superlative heuristic among those examined for small WTA problems.

In considering the performance of the three heuristics examined in this thesis, we observe that no heuristic always dominates in its reported solution. Often, the best solution we are able to determine is shared amongst two of the heuristics. When one heuristic is dominant, the other two are typically very close in value. We use this fact to further compare the performance of the heuristics with regard to their computational speed. Using this criteria, there is a very clear ranking in heuristic performance. While all three found solutions in less than a second for the smallest problem considered, and the Construction Heuristic continued to find solutions in less than a second for problem sizes up to the problem having 15 weapons and 10 targets, both the Greedy Branch and Bound Heuristic and the Construction Heuristic are far slower than the Quiz Problem Search Heuristic. The slowest average solution time for the Quiz Problem Search Heuristic was nearly eight times faster than the fastest average solution time from either of the other two heuristics. Therefore, we assert that the Quiz Problem Search Heuristic is the dominant heuristic considered in this paper when computational time is considered and, therefore, also dominant for real-time allocation applications.

We intend on extending this research by applying time-bounded neighborhood search heuristics to our Quiz Problem Search Heuristic to try to find real-time solutions closer to optimality without significantly increasing the required computational effort. We want to use the heuristics defined in this paper to address two-stage WTA problems and other dynamic instances of the WTA problem, to include stochastic instances. Additionally, we want to address the problem of limited sets, wherein weapons are capable of shooting targets within a defined range, resulting in targets which can only be shot by some of the weapons. Lastly, we want to bring together these two problems and solve a two-stage limited set problem. This research provides a foundation for this future research.

Appendix A. Story Board



Real-Time Heuristic Algorithms for the Static Weapon-Target Assignment Problem

CPT Alexander Kline
Advisor: Darryl K. Ahner, PhD
Committee Member: Brian J. Lunday, PhD
Department of Operational Sciences (ENS)
Air Force Institute of Technology

Sponsor:
USSTRATCOM/JWAC

Problem Statement

Generate near optimal solutions to the static weapon target assignment problem for large problems capable of real-time application.

WTA Objective Function

$$Min \sum_{j=1}^{n} V_j \prod_{i=1}^{m} q_{ij}^{x_{ij}}$$

$$st \sum_{j=1}^{n} x_{ij} \le w_i \quad for\ i = 1,\ldots,m$$

$$x_{ij} \ge 0 \text{ and integer}$$

$$i = \{1,\ldots,m\} \quad and\ j = \{1,\ldots,n\}$$

Theorem: ***The untransformed WTA objective is convex***

Research Questions

- Are there algorithms capable of finding exact solutions to the untransformed WTA?
- Are there faster approaches to finding quality solutions than exist in the literature?
- Are there methods of detecting and bypassing the greedy selection shortcoming?
- Are there unexplored heuristics which can find better solutions than existing heuristics?

AIR FORCE INSTITUTE of TECHNOLOGY

Branch and Bound

Initial Feasible Solution

$$c_{ij} = V_j p_{ij} \left(q_j^{\max}\right)^{k-1} \qquad c_{ij} = 0$$

Modified Quiz Problem Search Heuristic

$$y_{ij} = \frac{V_j P_{ij}}{1 - P_{ij}}$$

Greedy Branch and Bound Heuristic

Analysis

- Branch and Bound computational time grows exponentially, but less than brute force algorithm

Branch and Bound vs Brute Force
For 10 Weapons
Average Computational Time (sec)
Number of Targets
Brute Force
Branch and Bound

- Greedy B&B Heuristic finds the best solutions but slowly
- MQPS Heuristic finds quality solutions in real-time

Weapons	Targets	One Heuristic Dominates		
		Modified Quiz Problem Search	Construction	Greedy Branch & Bound
5	5	8	16	20
10	5	5	14	18
10	10	6	9	17
10	20	5	5	20
15	10	5	8	12
20	10	1	10	14
20	20	0	5	18
20	40	11	3	20
40	10	0	8	14
40	20	1	2	18
40	40	1	19	-
40	80	16	4	-
80	40	1	19	-
80	80	2	18	-
80	160	20	0	-

Computational Time Requirements (sec)							
Weapons	Targets	MQP Search		Construction		Greedy B&B	
		Average	St Dev	Average	St Dev	Average	St Dev
5	5	0.00333	0.01064	0.224	0.0976	0.568	0.257
10	5	0.00113	0.000176	0.445	0.0917	2.019	0.674
10	10	0.00111	0.000121	0.441	0.0975	7.657	2.726
10	20	0.00110	0.000105	0.487	0.107	242.267	55.907
15	10	0.00155	0.000104	0.767	0.0531	38.020	11.086
20	10	0.00202	0.000109	1.608	0.255	41.986	13.726
20	20	0.00215	0.000120	2.208	0.346	232.426	88.660
20	40	0.00280	0.000153	4.455	0.609	28443.757	6393.071
40	10	0.00407	0.000121	17.997	1.863	711.324	223.755
40	20	0.00536	0.000220	20.569	1.181	2691.799	632.036
40	40	0.00670	0.000171	14.311	0.432	-	-
40	80	0.00912	0.000151	47.077	20.162	-	-
80	40	0.0156	0.00121	309.641	10.313	-	-
80	80	0.0194	0.000245	177.106	6.210	-	-
80	160	0.0284	0.000274	1348.815	391.135	-	-

Conclusions

- If metric is objective function value, Greedy B&B Heuristic is superlative
- If metric is computational speed, MQPS Heuristic is superlative

Future Work

- Neighborhood search heuristics to improve MQPS Heuristic while maintaining real-time characteristic
- Two stage WTA Problem
- Limited target set WTA problem

Contributions

- Nonlinear Branch and Bound algorithm
- Untransformed WTA convexity proof
- Modified Quiz Problem Search Heuristic
- Greedy Branch and Bound Heuristic
- Greedy selection process shortcoming fix
- Improvements upon benchmark Construction Heuristic (Ahuja, et al. 2007)

Bibliography

Ahner, Darryl K. 2005. Planning and control of unmanned aerial vehicles in a stochastic environment. *Doctor of Philosophy dissertation, Boston University.*

Ahuja, Ravindra K, Kumar, Arvind, Jha, Krishna C, & Orlin, James B. 2007. Exact and heuristic algorithms for the weapon-target assignment problem. *Operations Research*, **55**(6), 1136–1146.

Bertsekas, Dimitri P, & Castañon, David A. 1999. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, **5**(1), 89–108.

Castañon, DA. 1987. Advanced weapon-target assignment algorithm quarterly report. *Tr-337, ALPHA TECH Inc., Burlington, Massachusetts.*

Chang, Shi-chung, James, Ronald M, & Shaw, Jonh J. 1987. Assignment algorithm for kinetic energy weapons in boost phase defence. *Pages 1678–1683 of: Decision and Control, 1987. 26th IEEE Conference on*, vol. 26. IEEE.

Day, Richard H. 1966. Allocating weapons to target complexes by means of nonlinear programming. *Operations Research*, **14**(6), 992–1013.

denBroeder, GG, Ellison, RE, & Emerling, L. 1959. On optimum target assignments. *Operations Research*, **7**(3), 322–326.

Evans, Stephen. 2015 (May). *North Korea 'test-fires submarine-launched missile'*. `http://www.bbc.com/news/world-asia-32671101`. Retrieved on 18 Jan 2017.

Flood, M. 1957. Target-assignment model. *In: Proceedings of the Princeton University Conference on Linear Programming, Princeton (NJ).*

Johansson, Fredrik, & Falkman, Göran. 2009. An empirical investigation of the static weapon-target allocation problem. *In: Proceedings of the 3rd Skövde Workshop on Information Fusion Topics (SWIFT2009). Sweden.*

Lee, Zne-Jung, Lee, Chou-Yuan, & Su, Shun-Feng. 2002. An immunity-based ant colony optimization algorithm for solving weapon–target assignment problem. *Applied Soft Computing*, **2**(1), 39–47.

Lloyd, Stuart P, & Witsenhausen, Hans S. 1986. Weapons allocation is NP-complete. *Pages 1054–1058 of: 1986 Summer Computer Simulation Conference.*

Manne, Alan S. 1958. A target-assignment problem. *Operations Research*, **6**(3), 346–351.

Matlin, Samuel. 1970. A review of the literature on the missile-allocation problem. *Operations Research*, **18**(2), 334–373.

MG Mahon, Francis (Ret.). 2016 (February). *US Air and Missile Defenses - A Critical Gap.* `http://www.realcleardefense.com/articles/2016/02/11/us_air_and_missile_defenses_-_a_critical_gap_109012.html`. Retrieved on 10 March 2016.

Rosenberger, Jay M, Hwang, Hee S, Pallerla, Ratna P, Yucel, Adnan, Wilson, Ron L, & Brungardt, Ed G. 2005. *The generalized weapon target assignment problem.* Tech. rept. DTIC Document.

Tomlinson, Lucas. 2016 (July). *Iran Conducts 4th missile test since signing nuke deal.* `http://www.foxnews.com/world/2016/07/15/exclusive-iran-conducts-4th-missile-test-since-signing-nuke-deal.html`. Retrieved on 18 Jan 2017.

United States Department of Defense. 2010 (January). *Ballistic Missile Defense Review Report* . `https://www.defense.gov/Portals/1/features/defenseReviews/BMDR/BMDR_as_of_26JAN10_0630_for_web.pdf`. Retrieved on 13 Jan 2017.

Wacholder, Eitan. 1989. A neural network-based optimization algorithm for the static weapon-target assignment problem. *ORSA Journal on computing*, **1**(4), 232–246.

Walkup, David W, & MacLaren, M Donald. 1964. *A multiple-assignment problem.* Tech. rept. DTIC Document.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 23–03–2017	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2015 — Mar 2017

4. TITLE AND SUBTITLE REAL-TIME HEURISTIC ALGORITHMS FOR THE STATIC WEAPON-TARGET ASSIGNMENT PROBLEM	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Kline, Alexander G, CPT	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENS-MS-17-M-139
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) STRATCOM/JWAC, Attn: Jill Morrissett, J902, 4048 Higley Rd, Dahlgren, VA 22448	**10. SPONSOR/MONITOR'S ACRONYM(S)** JWAC
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The problem of targeting and engaging individual missiles (targets) with an arsenal of interceptors (weapons) is known as the weapon target assignment problem. As many solution techniques are based upon a transformation of the objective function, their final solutions rarely produce optimal solutions. We propose a nonlinear branch and bound algorithm to provide the first optimization approach to the untransformed problem found in the literature. Further, we propose a new heuristic based upon the branch and bound algorithm which dominates other heuristics explored in optimality gap. We also propose a heuristic based upon the optimal solution to the quiz problem which finds solutions within 6% of optimal for small problems and provides statistically similar results as one of the best heuristics found in the literature for larger problems while solving these problems in ten thousandths of the time.

15. SUBJECT TERMS

Weapon-Target Assignment Problem,Branch and Bound Algorithm, Real-Time Heuristics

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Darryl K. Ahner, AFIT/ENS
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	47	(937) 255-6565, x4708; darryl.ahner@afit.edu