Weapon Target Assignment with Combinatorial Optimization Techniques

Asım Tokgöz[1], Serol Bulkan[2]
Department of Industrial Engineering
Marmara University
Göztepe, Istanbul 34722, TURKEY

***Abstract*—Weapon Target Assignment (WTA) is the assignment of friendly weapons to the hostile targets in order to protect friendly assets or destroy the hostile targets and considered as a NP-complete problem. Thus, it is very hard to solve it for real time or near-real time operational needs. In this study, genetic algorithm (GA), tabu search (TS), simulated annealing (SA) and Variable Neighborhood Search (VNS) combinatorial optimization techniques are applied to the WTA problem and their results are compared with each other and also with the optimized GAMS solutions. Algorithms are tested on the large scale problem instances. It is found that all the algorithms effectively converge to the near global optimum point(s) (a good quality) and the efficiency of the solutions (speed of solution) might be improved according to the operational needs. VNS and SA solution qualities are better than both GA and TS.**

Keywords—Weapon Target Assignment; Combinatorial Optimization; Genetic Algorithm; Tabu Search; Simulated Annealing; Variable Neighborhood Search; WTA; WASA; TEWASA

I. INTRODUCTION

In military operations, problems in planning and scheduling often require feasible and close to optimal solutions with the limited computing recourses and within very short time periods [1]. Especially the weapons developed in contemporary technology give very less chance to defend friendly assets as enemy forces execute complex saturated attacks. Therefore, quick and efficient reactions to subsidise these attacks become very vital to survive in the combat arena. Thus, assigning the limited resources (own weapons) to the hostile targets to achieve certain tactical goals [2] becomes an important issue called as Weapon Target Assignment (WTA) problem.

The WTA problem is a classical constrained combinatorial optimization problem arising in the field of military operations research [2] and it is NP-complete [3]. The term "allocation" and "assignment" are used analogously in the literature. The complexity of this problem drastically increases if you add the temporal and spatial constraints of both the friendly forces and hostile targets. The allocation of available capabilities to the correct targets needs complex calculations in a very short time. There are various simple traditional algorithms in literature that solve this problem such as graph search, implicit enumeration algorithms, dynamic programming, branch and bound algorithms,[4, 5] and simulated annealing. Even though these algorithms are simple, it is difficult to implement them especially when the problem scale is large. With the evolution of computer technology, some naive algorithms are proposed like analytical hierarchy process [6], network flow based methods [7], neural networks [8], genetic algorithms, tabu search, ant colony algorithm (ACO) and particle swarm optimization (PCO), very large scale neighborhood (VLSN), and maximum marginal return (MMR).

There are many works in the literature regarding the algorithms mentioned above. All of them almost define the problem similar but there are some differences on the solution approach. These are generally based on many factors, such as capability of targets and weapons, defense strategies, current combat conditions, the dynamic or static solution, layered defense or not, protecting the friendly assets or killing hostile targets or both (asset-based or target based), assessing the threats according to the capability and intent. Therefore the problem formulation differs the way how you approach the solution. All of them state the WTA as hard assignment problem and a comprehensive mathematical problem formulation example can be found at [4, 2]. You can also find a detailed literature overview and algorithm comparison from different perspectives at [9, 2, 10]. Regarding these works, the problem formulation looks similar, but there are some key points that need to be emphasized;

- Lee et al. [11] uses partially mapped crossover (PMX), inversion mutation and simulated annealing as local search.
- Lee et al. [12] uses greedy reformation scheme so as to have locally optimal offspring (greedy eugenics) which is a kind of novel crossover operator (EX) that try to inherit the good genes from the parents. They give information about various eugenics mechanisms orderly simple eugenics, simulated annealing, immune operator, and greedy eugenics.
- Lu et al. [13] uses uniform generation of initial population, roulette wheel selection based on sigma truncation scaling fitness and self-adaptive parameters for genetic operations (dynamic probability of mutation and crossover). Crossover adopts the two-point-crossover operation while mutation employs swap mutation operator.
- Shang et al. [14] added an extra step to the classical GA, which comes after creating new chromosomes by crossover and mutation, which is to apply local search to create new chromosomes and then evaluate the new chromosomes. After that, select the chromosomes with

the best fitness from the population. For LS they propose the immune GA (IGA) which is composed from vaccination and immune selection, of which the former is used for raising fitness and later is for preventing the deterioration. They also used a new crossover operator called as elite preserving crossover operator to enrich a more effective search.

- Dou et al. [15] uses chromosomes in matrix representation and adopts the dynamically adjusting punishment gene and self-regulating punishment rates. Initial population is selected in a reasonable manner so as to satisfy the constraints automatically. Roulette wheel selection and dynamically adjusted crossover and mutation probabilities are used.
- Li et al. [16] uses matrix-type encoding for chromosomes and a new crossover/mutation operator called "circle swap".
- Zhihua et al. [17] uses local information to get a better initial feasible solution by deducing the survival value matrix (heuristic information).
- Johansson and Falkman [10] uses enhanced MMR, GA, and PSO.

II. Problem Definition

The military domain requires operators to evaluate the tactical situation in a very short time interval, almost real-time, and then make decisions to protect the friendly assets against the enemy threats by allocating the available own weapons to the hostile units. Furthermore, when the severe stressful battlefield conditions added to this decision making process, operators become more overburdened and they need computerized decision support system tools to handle this complex situation [18]. This brings forth the evaluation on tracks (targets) that might have the hostility intent and also capabilities that might harm the friendly assets briefly called as Threat Evaluation (TE). TE process mostly requires decision support tools as the number of targets that need to be evaluated is crowded and rapid decisions are required to eliminate the enemy assets in time.

After the TE, operators need to assign the appropriate and efficient weapon or weapon-sensor pairs to the targets in order to get the maximum benefit which is called as Weapon Assignment or Weapon Assignment and Sensor Allocation (WA/ WASA). This assignment/allocation level may be at different force structures such as single asset, task group, and force. In this paper we will investigate the efficiency of genetic algorithm, tabu search, simulated annealing and variable neighborhood search algorithms which make the assignments of weapons to targets by employing a shoot-look-shoot (SLS) engagement policy.

As stated in [4, 8], the objective may be to minimize the total threat of all targets or maximize the total value of assets surviving through the whole defense process.

In this calculation, finding the probability of kill given hit of targets (threats) is a controversial issue as this information mostly confidential and hard to get. You can use this information in the algorithms by considering the availability of it. The WTA process is usually an ongoing process as it requires observing the status of targets and friendly weapons continuously to replan the engagements. The annihilated targets and unavailable friendly weapons are discarded for the next iterative plan phase. This process can be considered as the famous OODA (Observe, Orient, Decide and Act) loop.

The most of the work in the literature (asset-based, target-based, static or dynamic or combination of them) assume that the weapons are fired simultaneously, which is not the case in reality, and this one step engagement period is called as stage. After the each stage a damage assessment is performed, and based on this assessment available weapons are reassigned to the surviving targets iteratively.

III. Problem Formulation

In this paper, the series of static WTA (SWTA) is assumed and the iterative defense process is used. The problem formulation is a subset of [4, 2], every computing result will mimic the results of the stages by updating the status of the assets, weapons and targets. The status after the execution of found solution will be input for next iterative computing. Therefore the stages are omitted from the problem formulization and it is defined as the following;

T is the number of offensive targets with their attack aims exposed, K is the number of assets of the defender, W is the weapons available to intercept the targets,

$X = \left[x_{ij} \right]_{WxT}$ is the decision variable, $x_{ij} = 1$ if weapon i is assigned to target j , $x_{ij} = 0$ otherwise,

T_k is the index set of the targets that threaten asset k,

W_j is the index set of the weapons that are assigned to intercept target j,

v_k is the value of asset k,

q_{jk} is the lethality probability that target j destroys asset k,

p_{ij} is the lethality probability that weapon i destroys target j.

The expected total value of assets surviving through the whole defense process to be maximized,

$$J(X) = \sum_{k=1}^{K} v_k \prod_{j \in T_k} \left[1 - q_{jk} \prod_{i \in W_j} (1 - p_{ij})^{x_{ij}} \right] \quad (1)$$

The constraints involved in WTA primarily include capability constraints, strategy constraints, resource constraints, and engagement feasibility constraints as follows,

$$\sum_{j=1}^{T} x_{ij} \le n_i \quad \forall_i \in \{1, 2, ..., W\} \quad (2)$$

$$\sum_{i=1}^{W} x_{ij} \le m_j \quad \forall_j \in \{1, 2, ..., T\} \quad (3)$$

$$\sum_{j=1}^{T} x_{ij} \le N_i \quad \forall_i \in \{1, 2,....,W\} \tag{4}$$

$$x_{ij} \le f_{ij} \ \forall_i \in \{1, 2,....,W\} \quad \forall_j \in \{1,2,....,T\} \tag{5}$$

The constraint set (2) reflects the capability of weapons which is to fire at multiple targets at the same time. Even though modern weapon systems can shoot multiple targets at a time, this kind of weapon systems can be evaluated as multiple separate weapons. As a result of these facts, we set $n_i = 1$ for $\forall i \in \{1, 2, . . .,W\}$ [2].

The constraint set (3) restricts the weapon cost for each target. The setting of m_j (j = 1, 2,...,T) generally based on the combat performance of available weapons and desired lethal probability [2]. In this research, it is assumed that $m_j = 1$ for $\forall j \in \{1, 2,...,T\}$. This is a rational setting for missile-based defense systems and the "SLS" engagement policy [19]. For artillery-based defense systems, the value of m_j (j = 1, 2,...,T) may be substantially increased under the same demand in defensive strength. Therefore, the constraints in (3) can be considered as a strategy constraint [2].

The constraint set (4) basically reflects the amount of ammunition provided for weapons. N_i (i = 1, 2, . . .,W) is the maximum number of times that weapon i can be used due to its ammunition capacity [2].

In the constraint set (5), f_{ij} indicates the actual engagement feasibility for weapon i assigned to target j. $f_{ij} = 0$ if weapon i cannot engage to target j with any potential reason (weapon range, blind zone etc.); $f_{ij} = 1$ otherwise [2]. The time window of targets and weapons is the main factor that influences the engagement feasibility [20, 21]. Some scholars also use the term "cue" or "deadline" when referring to temporal and spatial issues [22, 23]. This will influence the time frame on the engagement feasibility of weapons. In addition, it increases the complexity of Dynamic WTA (DWTA) problems as well and the difficulty of creating feasible solutions. In this situation, it is difficult to implement an appropriate operator that can generate new solutions and guarantee their feasibility as well.

IV. Algorithms Used to Solve WTA

This paper will not discuss the details of the algorithms implemented here as you can find them in many text books and in many scholars' work. The work here is focused on the application of combinatorial optimization algorithms (genetic algorithm, tabu search, simulated annealing, and variable neighborhood search) to the weapon target assignment problem. Combinatorial Optimization Library framework of Skalicka [24] is modified and used in this work. This framework allows easy implementation of own algorithms and provides effective tools for bench-marking different algorithms on a set of various problems.

A. Genetic Algorithm

The theory of natural selection claims that the species living nowadays are evolved by complying with the harsh environmental conditions through millions of years of adaptation. The changes in the environment had forced the species to alter their genetic characteristics in order to survive. In ecosystem, recourses have always been scarce; therefore, a certain number of organisms have always competed for the same resources in the habitat. The capability to acquire the resources will determine the future of that organism. If an organism can acquire the resources, then it will successfully procreate and its descendants will have a tendency to be numerous in the future. These organisms are considered as fitted to the ecosystem. Organisms with less capability have a liability to have fewer or no descendants in the next days. Eventually, the new generations in the population will be more fitted to the ecosystem than previous ones and evolution will continue likewise [25]. Genetic Algorithm (GA) is based on the same idea of natural evolution.

GA algorithm imitates the natural selection process during the search phase of the problem domain to reach better solutions. It is inspired by natural evolution, like inheritance, mutation, selection, and crossover and commonly used in optimization problems to generate efficient solutions. The pseudo code is given in Table I.

TABLE I. Pseudo Code For GA

```
Initialize Algorithm (population size,
mutation rate, and problem)
Create Initial Random Population ()
Find fitness of each and store the best
Check for end conditions
Do
   Optimize by selection, mutation and
   disaster
   Find fitness of new individuals and
   store the best
   (Kill the worst 2 individuals)
Until end conditions meet
```

1) Fitness

The fitness function for this problem is the objective function.

2) Individuals

An individual is a single solution and a group of the individuals forms the population. The structure of the individual is composed of various formatted data elements and it is called as chromosome. In this work, permutation encoding (real number coding) is used to represent the weapons and their order mimics the assignments of them to the targets sequentially.

Targets	1	2	3	4	5	6	7	8
Chromosome 1	3	2	1	5	4	8	6	7
Chromosome 2	5	7	8	2	1	3	4	6

Fig. 1. Sample chromosome structure

As you can see from the Figure 1, in chromosome 1, weapons 3, 2, 1, 5, 4, 8, 6, and 7 are assigned to targets 1, 2, 3, 4, 5, 6, 7, and 8 respectively. If weapon and target size is different then, dummy weapons/targets are used in order to make them equivalent. This number is also the population size of the genetic algorithm used in this work.

3) Selection

After creation of the initial population, first step is selecting the individuals for reproduction. This selection is random with a probability depending on the relative fitness of individuals, so that often the best ones have more chance to be selected then poor ones. [25]. Roulette Wheel Selection is used in this work.

4) Reproduction

In the second step, offspring are bred by the selected individuals. For generating new chromosomes, the algorithm can use recombination, mutation and disaster. In this paper a disaster is introduced with a probability that hoped to improve the solution by exchanging a small percentage of population with the new randomly generated individuals.

Crossover (Recombination) is the process of taking two parent solutions and producing two children from them. After selection (reproduction) the population is enriched with better people. Crossover operator executed on the appropriate couple with the hope that it generates a better offspring [25].

Crossover is a three step recombination operator [25]: (1) the reproduction operator selects at random a couple strings for mating. (2) A cross site is chosen at random along the string length. (3) Finally, the position values are exchanged between the two strings according to the cross site [25].

In this work Partially Mapped/Matched Crossover method (PMX) is used. For instance, 3 and 6 positioned genes are selected and the genes between these two points are swapped for the chromosomes of son and daughter. For son, the rest of the genes are copied from father to the same place if they do not exist at the son. Daughter is created analogously like son.

Father	3	2	1	5	4	8	6	7
Mother	5	7	8	2	1	3	4	6

Fig. 2. Chromosome structure of the parents

Son			8	2	1	3	6	7
Daughter		7	1	5	4	8		6

Fig. 3. Incomplete chromosome structure of the children

After that the holes filled as follows: (1) Take the gene from father check if it exists at the son. (2) if not, put the same gene to the same position at the son. (3) If that gene exists at the son, then find the same gene at the mother and return that gene position. Look at the gene at the father with position returned from the mother. If found new gene not exist at son, put that gene to the original hole position. Repeat this cycle until all the holes are filled.

Daughter is created analogously like son.

For 3 at the father at position 1; 4 is placed at position 1 at son after total of 12 checks as follows:

Father	3(1)	2	1(7)	5	4(10)	8(4)	6	7
Mother	5	7	8(6)	2	1(9)	3(3)	4	6

Fig. 4. Iterations at the chromosome of the parents

Son	4(11)		8(5)	2	1(8)	3(2)	6	7
Daughter		7	1	5	4	8		6

Fig. 5. Iteartions at the incomplete chromosome of the children

One check after the 11th is to control if 4 does exist at the son, seeing not there, and then positioned it to 1 at son. The final new two offspring are as follows:

Son	4	5	8	2	1	3	6	7
Daughter	2	7	1	5	4	8	3	6

Fig. 6. Completed chromosome structure of the children

Following the crossover, the chromosomes are exposed to the mutation which will recover the algorithm from local minimum trap. Mutation has the possibility to introduce the lost genetic materials as it randomly alters the genetic structure. It ensures against losing the irreversible genetic material and also has been considered as a simple search operator.

While crossover is improving the current solution with better ones, mutation is expected to explore the whole search space and maintains the genetic diversity in the population. It incorporates new genetic structures in the population by randomly changing some of their components [25].

In this work, swap mutation is used with the probability 0.1. A uniform random number is generated and compared with the mutation probability for each offspring. If this number is smaller than this probability, then individual is subject to mutation. If we assume that son is subject to mutation with the positions 2 and 6 then, we simply exchange the genes at those positions.

Son	4	5	8	2	1	3	6	7
Daughter	2	7	1	5	4	8	3	6

Fig. 7. Chromosome structure of the children after the crossover

After mutation, the new individuals are as follows;

Son	4	3	8	2	1	5	6	7
Daughter	2	7	1	5	4	8	3	6

Fig. 8. Chromosome structure of the children after the mutation

With a very small probability a natural disaster is introduced in that a small portion of the population is replaced with randomly created new individuals. It is useful for diversification.

5) Evaluation

After the reproduction, the fitness of the new chromosomes is evaluated.

6) Replacement

During the last step, the worst fitted two individuals are killed from the population.

B. Tabu Search

Tabu Search (TS) is unquestionably distinguishable from the local search technique, as it incorporates intelligence. It keeps track of the history of iterative search or, equivalently, to enable the search with memory [26]. Thus, it will avoid both applying the same operations repeatedly and revisiting local optima. A tabu list is incorporated into the algorithm to serve as the memory function and it is an important mechanism to recover from local optima [2].

Even though random components are introduced to overcome the technical difficulties, in fact, tabu search is not

based on the chance. The basic idea of tabu search is to use the memories (tabu list) to explore the search space of the problem and move from one solution to a neighboring solution continually [26]. Yet it may be seen as local search to some people, some methods are used to carry out the jumps in the solution space, in this manner, TS is not a pure local search.

In this research, the candidate list is constructed from the possible *n(n-1)/2* moves except the tabu ones where n is the problem size. Sequentially all moves are tried and the first candidate operation that makes the fitness better is executed. In order to get rid of the local optima, an aspiration criteria, which is to accept the current iteration best move among the all possible moves with a probability of 0.8 even though it is not better than the fitness found so far, is applied. For tabu list a short term memory is used with random duration. The structure of it as follows [26]:

		Site (Targets)				
		1	***2***	***3***	***4***	***5***
Objects (Weapons)	***1***	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0
	5	0	0	0	0	0

Fig. 9. Tabu list

		Site (Targets)				
		1	***2***	***3***	***4***	***5***
Objects (Weapons)	***1***	0	0	10	0	0
	2	10	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0
	5	0	0	0	0	0

Fig. 10. Filled sample tabu list

Let's assume our current solution is (2, 4, 1, 5, 3) that is the assignment of weapons 2, 4, 1, 5, 3 to targets 1, 2, 3, 4, 5 respectively and we found a better move as the exchange of weapons (2 and 1) at sites 1 and 3 then we can fill the place of them with randomly generated duration as in Figure 10.

As it can be seen at Figure 10, 10 is randomly generated and assigned to the cells (2,1) and (1,3) of the tabu list. This assignment will prevent the allocation of weapon 1 to target 3 and weapon 2 to target 1 for the next 10 iteration. The random duration is a uniform iteration count between 1 and *lifespan + 1*. Lifespan is calculated as follows:

TABLE II. LIFE SPAN CALCULATION FOR TABU LIST

```
if ( problem.Dimension() < 10 ){
  lifespan=5*log10(problem.Dimension());
}else{
  lifespan= 8*log10(problem.Dimension());
}
lifespan++;
```

If the problem size is too large, then using the problem dimension as uniform upper bound will tend to prevent the efficient moves that might help to get better solutions. Thus, taking the logarithm of problem dimension will smooth the durations in the tabu list.

At initialization, a random configuration is created for the initial solution. After that TS move operations and aspiration criteria are used together to get better solutions.

C. Simulated Annealing

The annealing technique is to heat a material earlier to impart high energy to it and then cool it down slowly by keeping at each stage a temperature of sufficient duration. The controlled decrease strategy of the temperature ends up with a crystallized solid state. This is the stable state that corresponds to an absolute minimum of energy. The temperature must be decreased gradually and systematically to avoid from the defects. If so, this can lead to an amorphous structure, a metastable condition that equivalent to a local minimum of energy [26].

The behavior of the temperature in annealing seems as the same with the control parameter in optimization. The temperature has a role to guide the algorithm towards the better solutions. This can be done by lowering the temperature gradually in a controlled manner. If the temperature is lowered suddenly, then the algorithm stops with a local minimum [26].

How the thermodynamic balance of a physical system at a given temperature successfully achieved can be simulated by the Metropolis algorithm [27] on the basis of a given configuration. The system is subjected to an elementary proposed modification and this modification is accepted if it improves the objective function of the system more than the current value, on the other hand, if it improves the objective function ΔE, but below the current value, it is also accepted with a probability of $\exp(-\Delta E/T)$. In practice, this condition is realized by drawing a random real number ranging between 0 and 1, and if this random number is lower than or equal to $\exp(-\Delta E/T)$, then, even though the configuration causes a ΔE degradation in the objective function, it is still accepted. When this Metropolis rule of acceptance is observed subsequently, a chain of events is generated whose configuration depends on the one that immediately precedes it. Under these assumptions, when the chain is limitless, it can be shown that the system can reach its thermodynamic balance at the measured temperature which leads us to a Boltzmann distribution of the energy states at this temperature [26].

From this point of view, the function given to the temperature by the Metropolis rule is well understood. When the temperature is high, $\exp(-\Delta E/T)$ approximates to 1. Thus, most of the moves are accepted and the algorithm turns into a simple random walk in the configuration space. On the contrary, when the temperature is too low, $\exp(-\Delta E/T)$ approximates to 0. In this case, the moves that increase the energy are wrongly rejected.

Thus, the algorithm seems like the classical iterative improvement. At an intermediate temperature, the algorithm occasionally approves the transformations that degrade the objective function; hence a chance is given to the system to escape from a local minimum [26]. Upon reaching the thermodynamic equilibrium at a certain temperature, the temperature is reduced "slightly" and a new Markov chain is executed for this new temperature level [26].

In our case following simulated annealing approach [26] is used:

1) Initial Configuration

A random configuration is used as a starting solution.

2) Initial Temperature

The algorithm can be initialized with user configurable initial temperature and geometric low coefficient (α) or it can be calculated as a preliminary step using the following algorithm if these parameters are unknown:

(1) 100 disturbances created randomly and the average ΔE variations evaluated

(2) Initial rate of acceptance τ_0 of degrading perturbations of quality τ_0 = 50% is selected (starting at high temperature) and T_0 is calculated as $T_0 = -\Delta E/ln(\tau_0)$ deduced from $\tau_0 = exp(-\Delta E/T_0)$

3) Acceptance Rule of Metropolis

An elementary proposed modification is accepted if this transformation causes an improvement in the objective function of the system; on the contrary (ΔE > 0), a number r in [0, 1] is drawn randomly, and accept the disturbance if r < exp (–ΔE/T), where T indicates the current temperature.

4) Change in Temperature Stage

It can take place as soon as a better solution is found or if no improvement seen up to 100 trials.

5) Decrease of the Temperature

It can be carried out according to the geometrical law: $T_{k+1} = \alpha * T_k$, where α is usually between 0.8 and 0.9999.

6) Program Termination

It can be done by the stetted start parameters such as time and iteration count or if the current temperature approaches zero. The operations used in the study are random swaps up to three.

D. Variable Neighborhood Search

Variable Neighborhood Search (VNS) is a simple and efficient metaheuristic for combinatorial and global optimization that systematically change the neighborhood within a possibly randomized local search algorithm. Local search methods starts usually by performing a sequence of local changes to the initial solution which gradually improve the value of the objective function each time until no further improvements are found, that is the local optimum. The VNS does not follow a trajectory; rather it explores the distant neighborhoods of the current solution and jumps from there to a new solution if and only if an improvement was made. After reaching the local optimum of the first algorithm used in the neighborhood structure set, the next algorithm or subroutine tries to improve the current solution. This iterative cycle continues until certain stopping criteria are satisfied. In this way, most of the time, favorable characteristics of the current solution will be kept and used to get the next promising solutions [28, 29].

VNS uses the pre-selected neighborhoods structures of size k_{max} and this structure set is donated as N_k where k is from 1 to k_{max}. Hansen and Mladenovic [28] have studied the structure of the VNS and proposed a couple of VNS algorithms one of which presented at Table III. If a deterministic rule is applied to improve the search in VNS neighborhoods, there is possibility to be trapped in cycle. To avoid this drawback, shaking (in step 2a) is introduced and point x' is selected randomly in the algorithm [28].

TABLE III. STEPS OF BASIC VNS [28]

```
Initialization:
     Select the set of neighborhood structures N_k, k=1,…,k_max,
     Find an initial solution x,
     Choose the stopping conditions.
Repeat the following until stopping conditions are met:
      (1)   Set k=1
      (2)   Repeat the following steps until k = k_max
            (a)  Shaking: Generate a point x' at random from the kth
                          neighborhoodof x ( x'∈ N_k(x))
            (b)  Move or not:
                          If this point is better than the incumbent, move there
                          (x = x'), and continue the search with N_1(k=1)
                          otherwise, k = k + 1;
```

The selection of neighborhood structures, how many of them will be used, their order and changing search strategy from one to the other depends on the problem characteristics and it's not very easy to identify. Jarbouia et al. [30] give 5 good examples of neighborhood structures for the location routing problem with multiple capacitated depots and one incapacitated vehicle per depot. These neighborhood structures are namely *insertion*, *swap(interchange)*, *extended Or-opt*, *inverse Or-opt* and *add/drop depot* and they are explained in detail at the article.

At WTA problem, we have used three neighborhood structures sequentially (1) a number of swap or interchange moves, (2) insertion neighborhood and (3) triple swap or interchange moves. While running the experiments, it was seen that the most efficient neighborhood structure that improved the solution was the first one. As mentioned earlier, selection of

the neighborhood structures and their order essentially problem dependent.

V. Experiments and Results

In the literature, there are benchmark problems for well-defined problems such as travelling salesman problem (TSP). The aim of the TSP is the same for everyone: given a list of cities and the distances between each pair of cities, find the shortest possible route by visiting each city exactly once and return to the origin city. For benchmarking purpose of TSP problem, researchers have proposed certain well-defined scenarios *publicly* and every researcher can use this data set to test his/her algorithm. The goal is not changed from scenario to scenario and from researcher to researcher, always find the shortest possible tour. When it comes to WTA domain, there are no benchmark problems in WTA domain as in TSP. This may be considered as normal because the studies in literature are not formulated exactly the same and various formulation assumptions are proposed that come up with different objective function formulations. That is why, every study uses different unpublished scenarios (mostly random) and different objective function formulations. Therefore, almost every study has different solutions. For these reasons, we couldn't compare the results of this study with the aforementioned studies.

As we couldn't find the benchmark scenarios in the literature, we also created random scenarios like our colleagues. In order to compare the effectiveness of the algorithms herein applied to WTA, a benchmark reference solution is needed and for that purpose, the problem is also formulated in mathematics and solved with the commercial General Algebraic Modeling System (GAMS) version 23.5.1. In this work, to the best of our knowledge, for the first time GAMS solutions are used as reference to compare the effectiveness and quality of heuristic algorithms applied to WTA. We claim that this approach can be used by the community for benchmarking. The test is conducted with a Notebook PC that has an Intel Core i3 CPU at 1.13GHz and a 4 GB RAM.

The Combinatorial Optimization Library [24] used here is developed for education of basic algorithms and problems. Some performance issues like speed of solution are sacrificed for the sake of education. But the algorithms and problems can be implemented and compared with each other.

TABLE IV. Sample Wta Solution Summary

Scenario 6	GA	TS	SA	VNS
CPU Time (msec)	1560	530	1529	1388
Optimization Counter	10809	20	37991	33
Depth	N/A	19	3591	29
Fitness	80.7	80.7	80.7	80.7

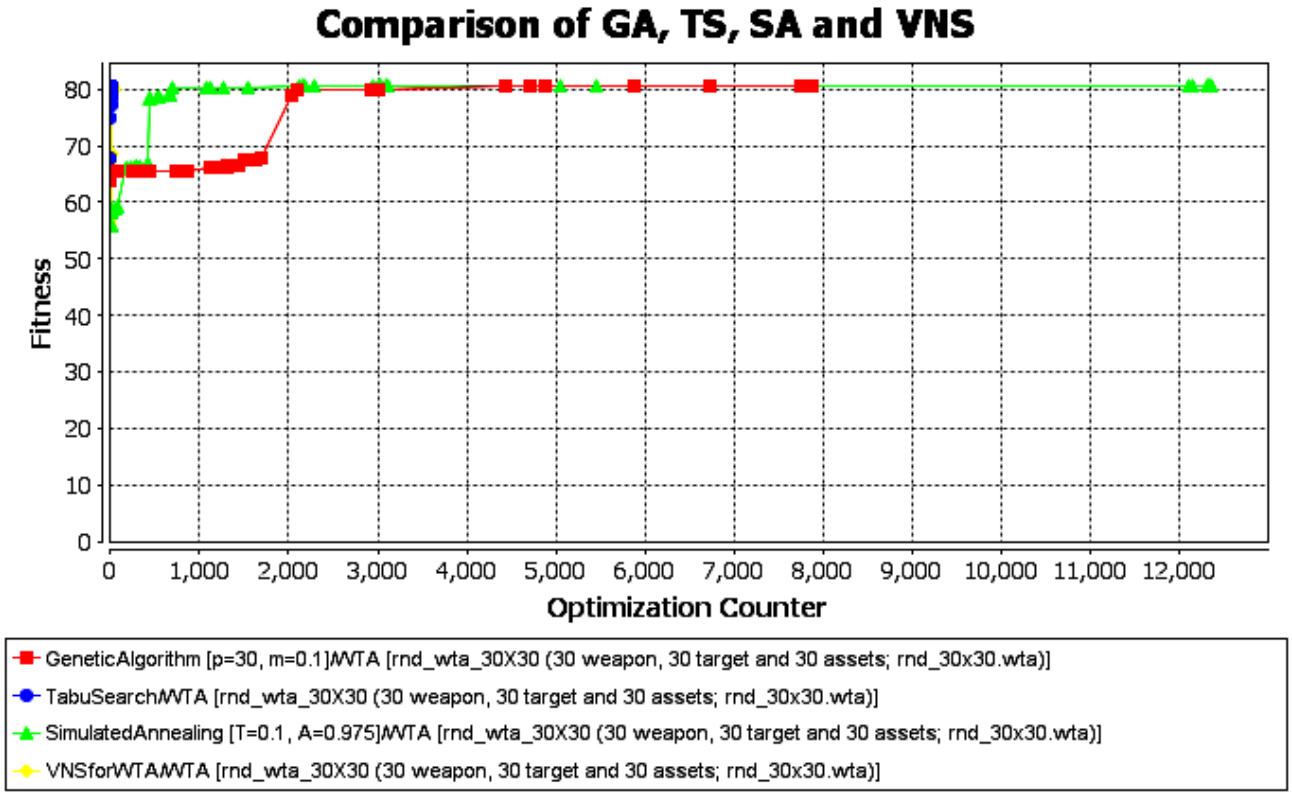


Fig. 11. A Sample (Scenario 6) WTA Solution

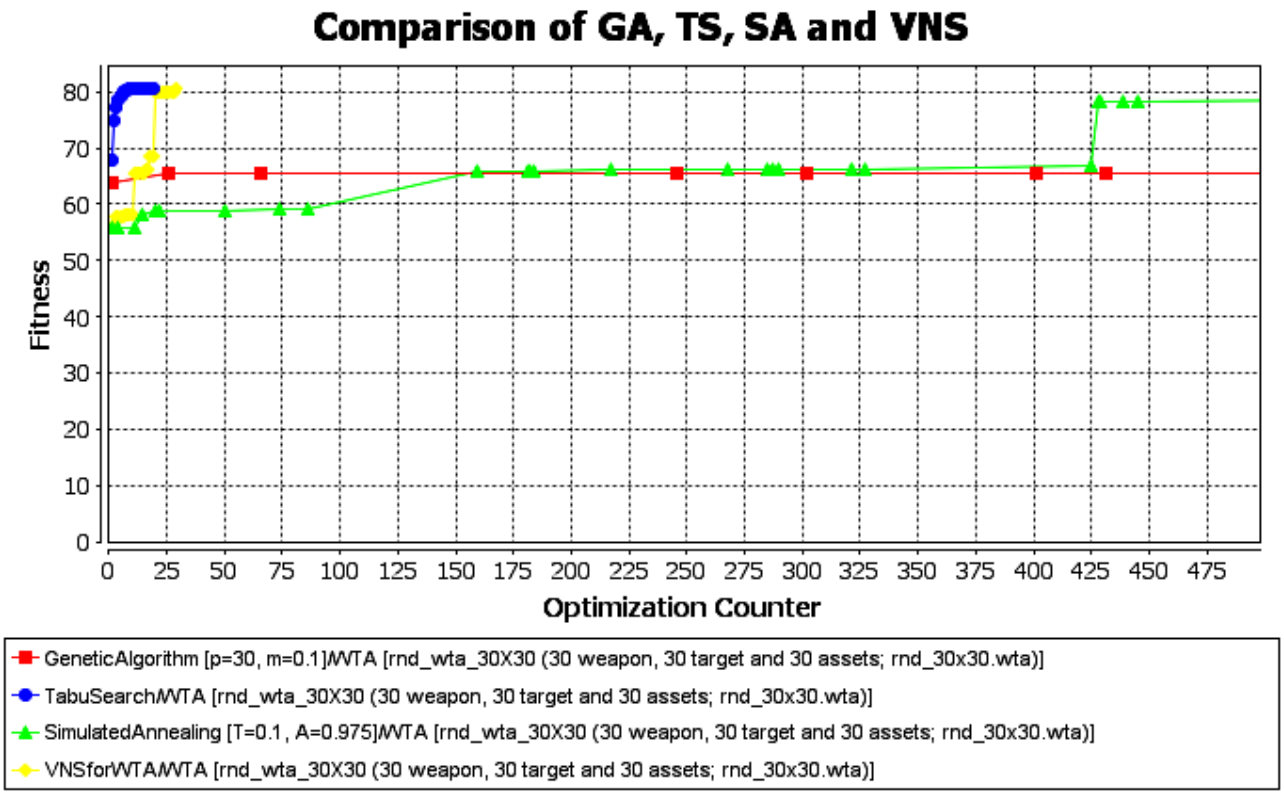


Fig. 12. A Sample (Scenario 6) WTA Solution (zoomed in up to 475 optimization counter)

The algorithms have variable iteration complexity, some of them are very simple and takes less time while others may need more arithmetic operations and exhaust much more time. So the *efficiency* is measured to compare the exhausted time for the same feasible solution, and the *quality* is to compare the feasible solution reached in same time [13].

As it can be seen at the above sample solution (Table IV, Figure 11 and Figure 12), even though the GA, VNS and SA exhausted almost the same CPU time they have quite different optimization/iteration counters. TS has both less optimization/iteration counter and CPU time than all the other three algorithms namely GA, VNS and SA. Even though Figure 11 and Figure 12 show the solution results of the same scenario, Figure 12 is added to discriminate the behavior of the TS and VNS for better perception at lower optimization counter level.

For experiment, 19 random scenarios are created starting from dimension of 5 to 95 with 5 step increments as in the Table IV. Thus these scenarios cover both simple and large-scale instances of the WTA. A moderate WTA combat situation is usually not greater than 30.

TABLE V. WEAPON-TARGET SIZE TO SCENARIO MAPPING

Scenario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Weapon	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95
Target	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95

These scenarios are solved with GAMES Couenne and Bonmin MINLP solvers and these solutions are used as reference for the solutions of implemented algorithms. GAMS solvers executed the scenarios once while the GA, TS, SA and VNS algorithms tried ten times.

As it can be seen from the minimum and maximum values, the gap is not so large between them. That is why we assumed that ten run is enough for significance. CPU time is recorded as solution time for the algorithms. The Table VI, Figure 13 and Figure 14 summarize the observed time values.

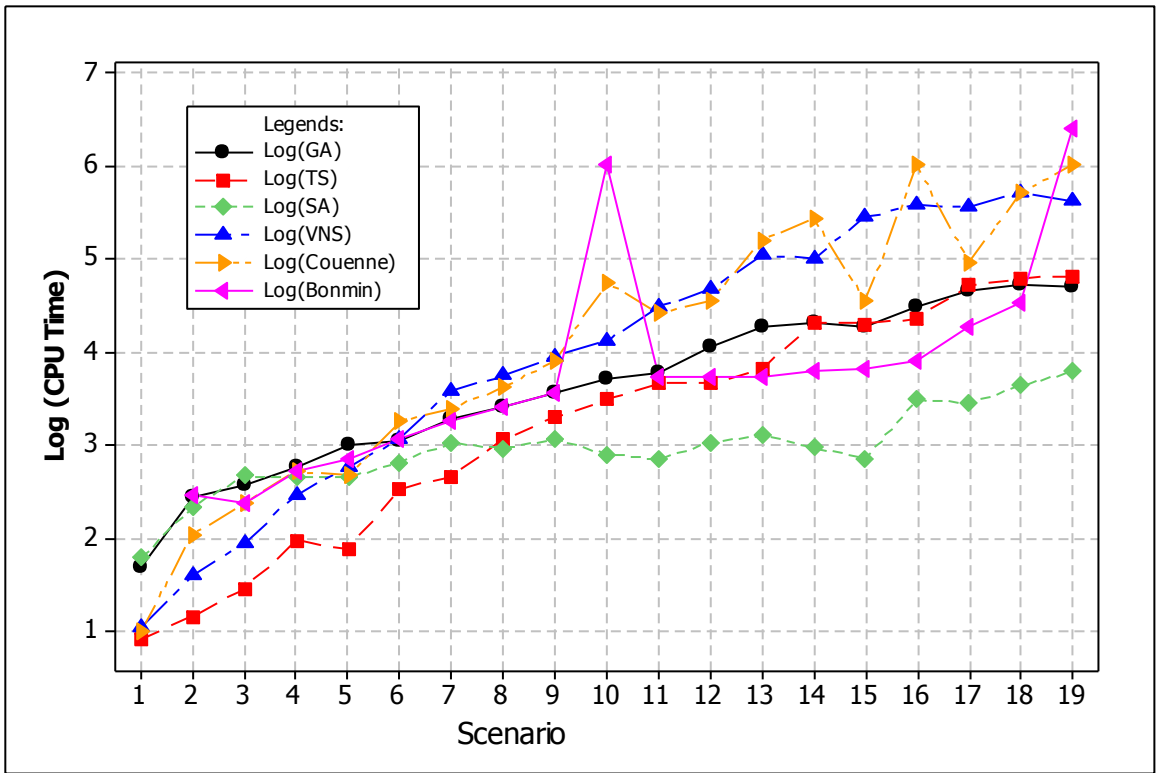


Fig. 13. Time Comparison of Algorithms and GAMS Solvers

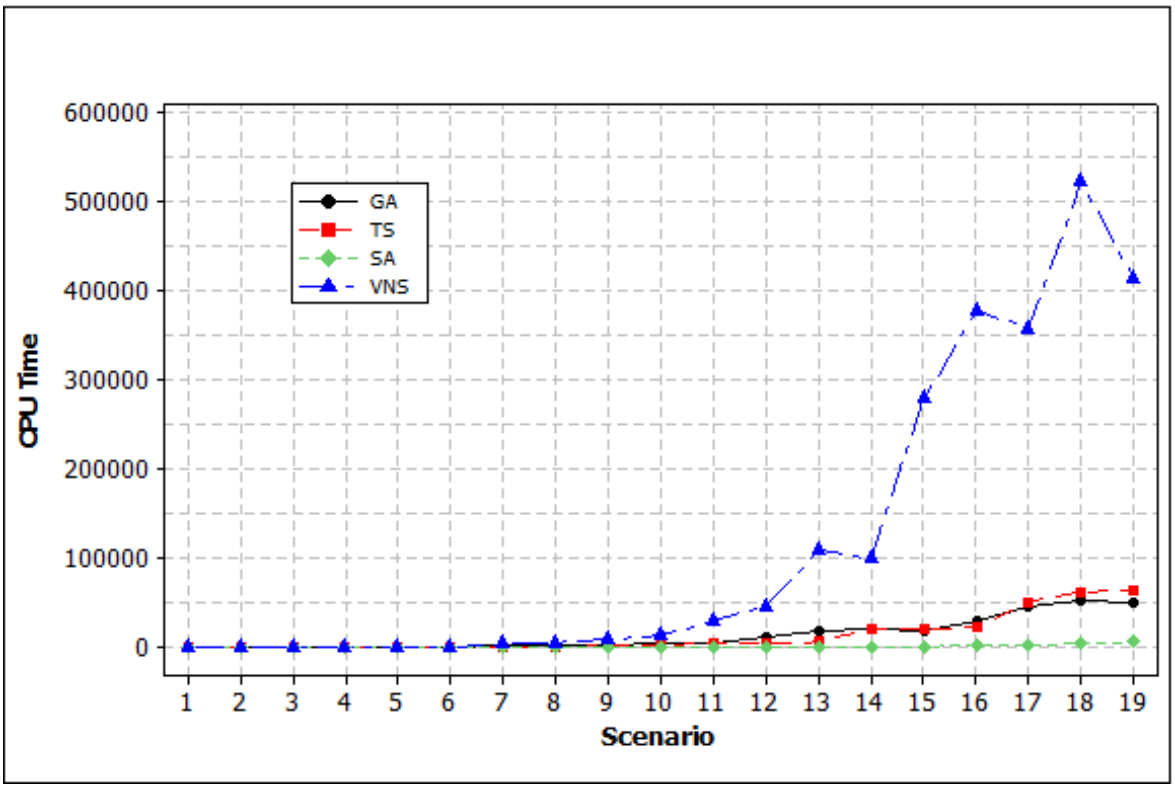


Fig. 14. CPU Time Comparison of Algorithms

TABLE VI. TIME COMPARISON OF SOLUTIONS

S.N.	Average				GAMS	
	GA	*TS*	*SA*	*VNS*	*Couenne*	*Bonmin*
1	48.3	48.3	61.1	11.1000	10	0
2	279.3	279.3	216.8	39	109	281
3	373	373	463.4	88.7000	234	234
4	563.3	563.3	444.7	290	515	515
5	971.9	971.9	441.6	575.7	464.86	717
6	1106	1106	647.1	1176.2	1794	1154
7	1923.3	1923.3	1057.8	3870.2	2481	1779
8	2572.6	2572.6	915.6	5534.6	4087	2605
9	3524.3	3524.3	1148.2	8839.1	8159	3604
10	4998.3	4998.3	792.6	13026	55271	1012670
11	5948	5948	695.8	30170.8	26333	5366
12	11261.6	11261.6	1067	46541.2	34866	5367
12	18561.1	18561.1	1266.7	109367.3	157171	5382
14	20127.4	20127.4	932.7	99990.6	269320	6271
15	18292.5	18292.5	713	279313.4	34825	6474
16	29904	29904	3071.9	379383.6	1004070	7800
17	44904.7	44904.7	2812.7	358432.5	92087	18517
18	53274.1	53274.1	4311.5	524971.6	506473	34133
19	50031.2	50031.2	6084.1	414502.5	1022700	2446890

TABLE VII. FITNESS OF GAMS SOLUTIONS

S.N.	GAMS Found Fitness		GAMS Best Possible Fitness	
	Couenne	*Bonmin*	*Couenne*	*Bonmin*
1	204.0261	204.0261	204.0261	204.0261
2	289.7098	289.7098	289.7098	289.7098
3	128.5998	128.5998	128.5998	128.5998
4	119.6860	120.1146	120.4492	120.1146
5	58.5471	58.9760	59.0120	58.9760
6	80.6784	80.1635	80.8405	80.1635
7	33.7837	33.6791	33.8938	33.6791
8	13.0186	13.0687	13.0856	13.0687
9	21.3787	21.3802	21.6859	21.3802
10	5.6773	5.4292	5.9119	6.3052
11	1.7663	1.7831	1.9030	1.7831
12	6.9788	6.9673	7.2542	6.9673
12	2.6835	2.5334	2.7313	2.5334
14	0.6800	0.6809	0.6915	0.6809
15	0.2573	0.2608	0.2783	0.2608
16	0.2798	0.2805	0.3171	0.2805
17	0.3315	0.3281	0.3424	0.3281
18	0.3523	0.3539	0.3870	0.3539
19	0.0422	0.0424	0.0538	0.0424

When Table VI is examined, GAMS Bonmin has 2 spikes at the scenarios 10 and 19 while GAMS Couenne has 3 spikes at the scenarios 16, 18 and 19. Because of these spikes, when the CPU times are plotted, the distinction among the solutions can't be discriminated easily. Therefore, in order to get a better visualization, we plotted the logarithm of the CPU times at Figure 13. The CPU times of the solutions of the algorithms, but GAMS solutions, are also plotted as in Figure 14.

When we compare the GA, TS, SA and VNS solution times, the CPU time values are close to each other up to scenario 10 and after that, there are steep slopes for the GA, TS and especially for VNS. SA solution time goes on a very low slope and it can be easily seen that SA outperforms these algorithms when the problem dimension gets bigger.

On the other hand, SA cooling down needs special care especially when the problem dimension gets bigger in order to converge to the global optimum. For cooling down in the trials up to scenario 15, $\alpha = 0.95$ is used while at scenarios 16, 17 and 18, $\alpha = 0.975$ and at scenario 19, $\alpha = 0.985$ are used.

TABLE VIII. GENETIC ALGORITHM FITNESS COMPARISON

S.N.	Fitness			Best Found GAMS Fitness
	Minimum	*Average*	*Maximum*	*Approximation %*
1	204.0261	204.0261	204.0261	100.000
2	289.7098	289.7098	289.7098	100.000
3	128.5998	128.5998	128.5998	100.000
4	119.8600	119.6903	120.1146	99.647
5	58.9760	58.9760	58.9760	100.000
6	80.6788	80.6788	80.6788	100.001
7	33.7987	33.8114	33.8146	100.082
8	13.0041	13.0235	13.0687	99.654
9	21.3830	21.3830	21.3830	100.013
10	5.7034	5.7074	5.7078	100.530
11	1.7754	1.7808	1.7832	99.871
12	6.9693	6.9785	6.9800	99.996
12	2.4192	2.6690	2.6968	99.460
14	0.6560	0.6791	0.6819	99.744
15	0.2627	0.2627	0.2630	100.708
16	0.2798	0.2804	0.2805	99.970
17	0.3334	0.3334	0.3334	100.568
18	0.3553	0.3557	0.3559	100.521
19	0.0427	0.0427	0.0427	100.798

TABLE IX. TABU SEARCH FITNESS COMPARISON

S.N.	Fitness			Best Found GAMS Fitness
	Minimum	*Average*	*Maximum*	*Approximation %*
1	204.0261	204.0261	204.0261	100.000
2	289.7098	289.7098	289.7098	100.000
3	128.5998	128.5998	128.5998	100.000
4	120.1146	120.1146	120.1146	100.000
5	58.9760	58.9760	58.9760	100.000
6	80.6788	80.6788	80.6788	100.001
7	33.7987	33.7987	33.7987	100.044
8	13.0041	13.0041	13.0041	99.505
9	21.3830	21.3830	21.3830	100.013
10	5.7078	5.7078	5.7078	100.537
11	1.7832	1.7832	1.7832	100.008
12	6.9800	6.9800	6.9800	100.017
12	2.6968	2.6968	2.6968	100.497
14	0.6818	0.6818	0.6818	100.134
15	0.2630	0.2630	0.2630	100.825
16	0.2805	0.2805	0.2805	100.008
17	0.3334	0.3334	0.3334	100.568
18	0.3558	0.3558	0.3558	100.531
19	0.0427	0.0427	0.0427	100.812

When the GA, TS and SA algorithms compared with GAMS solutions, they had reasonably better temporal solutions as it can be seen at Figure 13. On the other hand, VNS has almost the same time values with the GAMS solutions as it tries three local search techniques sequentially which takes more time. Even with a single neighborhood structure for VNS, we had good fitness results with less time. Thus for this kind of WTA problem, less neighborhood structure can be applied that will end with good results.

The optimization/fitness values of the GAMS models are given at Table VII. These values are used as reference values for the solutions of the algorithms. GA, TS, SA, and VNS solution fitness values are compared with the GAMS models and found that they converge to the global optimum at most trials and the approximation results are promising as can be seen at Table VIII, IX, X and XI. The VNS algorithm produced better results than the other three algorithms and GAMS models. So for the non-real time constrained problems, VNS may give promising solution results. But as mentioned before, construction of the VNS topology requires special care.

TABLE X. SIMULATED ANNEALING FITNESS COMPARISON

S.N.	Fitness			Best Found GAMS Fitness
	Minimum	*Average*	*Maximum*	*Approximation %*
1	204.0261	204.0261	204.0261	100.000
2	289.7098	289.7098	289.7098	100.000
3	128.5998	128.5998	128.5998	100.000
4	120.1146	120.1146	120.1146	100.000
5	58.9760	58.9760	58.9760	100.000
6	80.6788	80.6788	80.6788	100.001
7	33.8146	33.8146	33.8146	100.092
8	13.0041	13.0429	13.0687	99.802
9	21.3830	21.3830	21.3830	100.013
10	5.7074	5.7077	5.7078	100.536
11	1.7794	1.7822	1.7831	99.954
12	6.9492	6.9742	6.9800	99.934
12	2.4163	2.6674	2.6967	99.400
14	0.6783	0.6805	0.6817	99.945
15	0.2532	0.2575	0.2615	98.716
16	0.2791	0.2801	0.2804	99.874
17	0.3291	0.3323	0.3333	100.248
18	0.3536	0.3554	0.3557	100.429
19	0.0426	0.0426	0.0427	100.636

TABLE XI. VNS FITNESS COMPARISON

S.N.	Fitness			Best Found GAMS Fitness
	Minimum	*Average*	*Maximum*	*Approximation %*
1	204.0261	204.0261	204.0261	100.000
2	289.7098	289.7098	289.7098	100.000
3	128.5998	128.5998	128.5998	100.000
4	120.1146	120.1146	120.1146	100.000
5	58.9760	58.9760	58.9760	100.000
6	80.6788	80.6788	80.6788	100.001
7	33.8146	33.8146	33.8146	100.092
8	13.0687	13.0687	13.0687	100.000
9	21.3830	21.3830	21.3830	100.013
10	5.7078	5.7078	5.7078	100.537
11	1.7832	1.7832	1.7832	100.009
12	6.9800	6.9800	6.9800	100.018
12	2.6968	2.6968	2.6968	100.497
14	0.6819	0.6819	0.6819	100.150
15	0.2630	0.2630	0.2630	100.834
16	0.2805	0.2805	0.2805	100.008
17	0.3334	0.3334	0.3334	100.568
18	0.3559	0.3559	0.3559	100.554
19	0.0427	0.0427	0.0427	100.812

VI. CONCLUSIONS

For the reasons explained in the experiments and results section, we couldn't compare our results with the studies in the literature.

That is why we created random scenarios from simple to hard ones and solved them with the proposed algorithms. In order to verify and validate the correctness of the solutions, we also solved the same scenarios with GAMS software. Our main goal is to investigate the most efficient algorithm that solves WTA problem in a reasonable time interval. The exact solvers can also solve the problems optimally but usually it takes too much time to use them in the military domain.

The NP-Complete WTA problem [3] is hard to solve and different heuristics were used to solve them in the past. After the advances in the computing power at computer technology, the algorithms that need huge computer power now can be applied to solve WTA problems. This study is made to measure the efficiency and quality of GA, TS, SA, and VNS algorithms that applied to WTA problem.

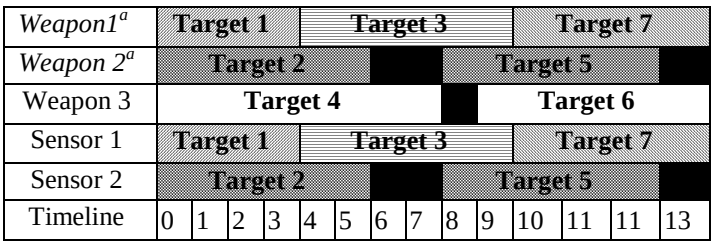
The quality of algorithms applied to WTA problem seems very significant while the efficiency may need to be improved considering the problem dimension. For small size WTA problems, all the four algorithms are promising, but for the large size WTA problems, only SA is good. The VNS algorithm applied here with three neighborhood structure took significant time to solve the large sized WTA problem. A hybrid algorithm that start with SA and continue with the TS and VNS will probably produce good results that meet the temporal and objective constraints, which are worth to try.

The efficiency, speed of solution, is ignored during the implementation of the algorithms. A better coding of the algorithms might produce much better results that meet the temporal constraints also.

VII. FUTURE WORK

WTA is usually considered as part of weapon assignment and sensor allocation scheduling that assessed together in military fields and it is a challenging problem. In this schedule, the factors such as weapon and sensor ranges, weapon and sensor blind zones, kill probabilities, weapon and sensor availability times, ammunition supply have to be considered and after that an engagement must be scheduled to annihilate the enemy targets and defend friendly assets. A simple example plan can be seen at Figure 15.

The static or dynamic multi-staged defense strategies usually don't produce realistic engagement schedules as they assume that all the engagements are simultaneous or weapons are fired all together which is not common in reality. In future, realistic WASA schedules will be studied.



[a] Semi-active weapon system which is guided by a sensor.

Fig. 15. A Sample WASA engagement schedule

REFERENCES

[1] Toet Alexander and Waard Huub de (1995). The Weapon Target Assignment Problem. CALMA Report CALMA.TNO.WP31.AT.95c.

[2] Xin Bin, Chen Jie, Zhang Juan, Dou Lihua and Peng Zhihong (2010). "Efficient Decision Makings for Dynamic Weapon-Target Assignment by Virtual Permutation and Tabu Search Heuristics, IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications and Reviews, 40(6):649-662.

[3] Lloyd S.P. and Witsenhausen H.S (1986). Weapon allocation is NP-complete. IEEE Summer Simulation Conference, Reno; NV (USA); 28-30 July 1986. pp. 1054-1058.

[4] Hosein, Patrick A and Athans, Michael (1989). Dynamic Weapon-Target Assignment Problem. Symposium on C2 Research, Washington, DC. [online]. Available at: http://www.dtic.mil/dtic/tr/fulltext/u2/a210442.pdf.

[5] Hosein, Patrick A and Athans, Michael (1990). Some Analytical Results for the Dynamic Weapon Target Allocation Problem. February 1990. [online]. Available at: https://dspace.mit.edu/bitstream/1721.1/3179/1/P-1944-21258906.pdf.

[6] Cheng CH and Mon DL, (1994). Evaluating Weapon System by Analytical Hierarchy Process Based on Fuzzy Scales. Journal of Fuzzy Sets and Systems, 63:1-10.

[7] Ahuja, R. K., A. Kumar, K. Jha and J. B. Orlin (2003). Exact and Heuristic Methods for the Weapon Target Assignment Problem. MIT Sloan School of Management, Working Paper No 4464-03.

[8] Owechko, Y and Shams, S,(1994). Comparison of Neural Network and Genetic Algorithms for a resource allocation problem. Neural Networks. IEEE World Congress on Computational Intelligence, 7:4655-4660.

[9] Naidoo, Shahen (2008). Assignment to Resource Allocation for Emergency Response – A Literature Survey. [online]. Available at: http://www.orssa.org.za/wiki/uploads/Johannesburg/Naidoo2008.pdf.

[10] Johansson, Fredrik and Falkman, Göran (2011). Real-time Allocation of Firing Units To Hostile Targets. Journal of Advances in Information Fusion, 6(2):187-199.

[11] [11] Lee Zne-Jung, Su Shun-Feng, and Lee Chou-Yuan (2002). A Genetic Algorithm with Domain Knowledge for Weapon-Target Assignment Problems. Journal of the Chinese Institute of Engineers, 25(3):287-295.

[12] Lee Zne-Jung, Su Shun-Feng, and Lee Chou-Yuan (2003). Efficiently Solving General Weapon-Target Assignment Problem by Genetic Algorithms with Greedy Eugenics. IEEE Transactions On Systems, Man, and Cybernetics—Part B: Cybernetics, 33(1):113-121.

[13] Lu Houqing, Zhang Hongjun, Zhang Xiaojuan and Han Ruixin (2006). An Improved Genetic Algorithm for Target Assignment, Optimization of Naval Fleet Air Defense. 6th World Congress on Intelligent Control and Automation, Dalian, China, pp. 3401-3405.

[14] Shang Gao, Zaiyue Zhang, Xiaoru Zhang and Cungen Cao (2007). Immune Genetic Algorithm for Weapon-Target Assignment Problem. Workshop on Intelligent Information Technology Application, pp.145-148.

[15] [15] Dou Jihua, Yang Xingbao and Lu Yonghong (2009). Improved Genetic Algorithm For Multichannel Ship-To-Air Missile Weapon System WTA Problem. 2nd IEEE International Conference on Computer Science and Information Technology, pp.210-214.

[16] Li Peng, Wu Ling and Lu Faxing (2009). A Mutation-Based GA for Weapon-Target Allocation Problem Subject to Spatial Constraints. International Workshop on Intelligent Systems and Applications, pp.1-4.

[17] Zhihua Song, Fashun Zhu and Duolin Zhang (2009). A heuristic genetic algorithm for solving constrained Weapon-Target Assignment problem. Intelligent Computing and Intelligent Systems, 1:336-341.

[18] Roux JN and Vuuren JH van (2007). Threat evaluation and weapon assignment decision support: A review of the state of the art. ORION, 23(2):151–187.

[19] Karasakal, Orhan (2008). Air defense missile-target allocation models for a naval task group. Computers & Operations Research, 35(6):1759–1770.

[20] Cai H., Liu J., Chen Y., and Wang H. (2006). Survey of the research on dynamic weapon-target assignment problem. Journal of Systems Engineering and Electronics, 17(3):559–565.

[21] Li J., Cong R., and Xiong J. (2006). Dynamic WTA optimization model of air defense operation of warships' formation. Journal of Systems Engineering and Electronics, 7(1):126–131.

[22] Blodgett D., Gendreau M., Guertin F., and Potvin J. Y. (2003). A tabu search heuristic for resource management in naval warfare. Journal of Heuristics, 9:145–169.

[23] Wu L., Xing C., Lu F., and Jia P. (2008). An anytime algorithm applied to dynamic weapon-target allocation problem with decreasing weapons and targets. IEEE Congress on Evolutionary Computation, Hong Kong, China, pp. 3755–3759.

[24] Skalicka Ondrej (2010). Combinatorial Optimization Library, Master's Thesis, Czech Technical University in Prague.

[25] Sivanandam S.N. and Deepa S.N.(2008). Introduction to Genetic Algorithms. Springer. ISBN9783540731900.

[26] Dréo Johann, Siarry Patrick, Pétrowski Alain and Taillard Eric (2006). Metaheuristics for Hard Optimization. Springer. ISBN-139783540230229.

[27] Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., and Teller E.(1953). Equation of state calculation by fast computing machines. Journal of Chemical Physics, 21(6):1087–1092.

[28] Hansen, Pierce and Mladenovic, Nenad (2001). Variable Neighborhood Search: Principles and applications. European Journal of Operational Research, 130:449-467.

[29] Mladenovic, Nenad and Hansen, Pierce (1997). Variable Neighborhood Search. Computers & Operations Research, 24(11):1097-1100.

[30] Jarbouia B, Derbela H, Hanafic S and Mladenović N (2013). Variable neighborhood search for location routing. Computers & Operations Research, 40(1):47–57.