Genetic Algorithms for Wargame Operational Planning

John A. Pav, Eric T. Jamieson
Booz Allen Hamilton
Arlington, Virginia
Pav_John@bah.com, Jamieson_Eric@bah.com

Adam B. Haywood
HAF A5/7
Washington, D.C.
Adam.Haywood.2@us.af.mil

Abstract

Assigning weapons to targets is a key aspect of operational planning in wargames. In past wargames, these assignments were built by subject matter experts with a deep understanding of each weapon system, platform, and target. However, the time-consuming process of manually building an Integrated Tasking Order (ITO) slows the wargame's pace. To adapt to increasing demand for speed and effectiveness, A.I. methods to automate the assignments must be explored. It is a classic example of the Weapon-Target Assignment (WTA) problem in which a set of various weapons is assigned to a set of targets to maximize damage to the enemy. The WTA problem can be modeled and defined using tactical and logistical constraints, including compatibility, availability, lethality, and range of platforms and munitions. This paper describes a customized genetic algorithm (GA) and demonstrates its ability to solve a feasible weapon-to-target operational plan. Due to the vast solution space of all possible assignments, finding an optimal solution is computationally expensive, but rapid results are imperative in a fast-paced wargaming environment. Empirical testing of this GA on the problem and Monte Carlo simulation results demonstrate the GA's capability to obtain an effective ITO within the time frame.

Key Words: genetic algorithms, optimization, Weapon-Target Assignment Problem, wargames, simulation

About the Authors

John Pav is a Lead Engineer at Booz Allen Hamilton. He has a Bachelor's in Applied Mathematics from Davidson College and a Master's in Computer Science from George Washington University. He currently leads the development of wargaming software tools and war simulations for Air Force Futures.

Eric Jamieson graduated from Virginia Polytechnic Institute and State University in 2017 with a Bachelor's in Aerospace Engineering. He previously worked for Lockheed Martin on the F-16 and F-22 programs and is currently a Lead Engineer at Booz Allen Hamilton, contracted to Air Force Futures' Modeling and Simulation Team.

Adam Haywood graduated from the Air Force Institute of Technology in 2021 with a Ph.D. in Operations Research. He is currently employed by the Department of Defense as an Operations Research Analyst in Air Force Futures' Modeling and Simulation Team.

Genetic Algorithms for Wargame Operational Planning

John A. Pav, Eric T. Jamieson
Booz Allen Hamilton
Arlington, Virginia
Pav_John@bah.com, Jamieson_Eric@bah.com

Adam B. Haywood
HAF A5/7
Washington, D.C.
Adam.Haywood.2@us.af.mil

The United States Air Force Futures designs and executes Title 10 Wargames. These wargames take place over the course of two weeks and consist of 250 to 400 participants. The goal is to inform senior leadership on how best to budget, design, and present their force for future conflicts based on operational experimentation (Georgetown University Wargaming Society, 2020). Subject matter experts (SMEs) from both Blue and Red teams design and plan operations based on the commander's intent. Each team has two days to coordinate each move, a battle plan involving hundreds of platform movements. The teams are divided into groups, one of which is responsible for coordinating all defensive counter-air, offensive counter-air, fighter strike, bomber strike, and escort aircraft missions. The large quantity of mission decisions in two days makes it impossible to ensure the group employs the optimal mix of aircraft, weapons packages, and targets in the two-day period, as there are billions of combinations.

Computers can handle vast amounts of data and perform algorithms to manipulate data at much higher rates than human planners. The Air Force Future's Modeling & Simulation Team (comprised of government civilians and Booz Allen Hamilton contractors) focuses on developing software tools to analyze the data and optimize the building of an operational plan, with the goal of producing more efficient solutions more quickly than human SMEs. Utilizing a genetic algorithm ultimately speeds up the move process, allowing for more moves in the two-week wargame and the potential for greater learning opportunities through computer assistance.

OPERATIONAL PLANNING

Operational planning translates a strategy into executable activities, operations, and campaigns, within resource and policy limitations to achieve objectives (United States Air Force, 2020). In Title 10 Wargaming, the combination of both strategy and operational planning is produced in the form of a flying schedule. Once the flying schedule is tasked to targets from the joint integrated prioritized target list (JIPTL), it becomes an Air Tasking Order. Furthermore, if the flying schedule consists of multiple U.S. military branches, it becomes an Integrated Tasking Order (ITO). The JIPTL is a prioritized list of targets approved and maintained by the joint force commanders (United States Air Force, 2018). This list shows a target's importance in relation to other targets within an operational system (United States Air Force, 2021).

Integrated Tasking Order

The purpose of the ITO is to convey the operations of the commander's intent and capture it in data form. ITOs organize missions by move, pulse, and package: a move is a unified battle plan for a 12- to 24-hour time frame, consisting of all platform movements, and all moves are played in chronological order; a pulse is a smaller time frame within a move, between 2 and 6 hours; and a package is a grouping of missions within a pulse. Each mission is detailed with location, timing, platform, weapon, and target information. A portion of a sample ITO is shown in Table 1.

Table 1: Sample ITO of Mock Data

Move	Pulse	Package ID	Mission ID	Mission Type	Base	AO	Platform Type	Tail	Take Off	Time on Target	Land Time	SCL	Target
1	1	A	1	SA	Base1	AO1	AC1	1	20200101 00:00	20200101 04:00	20200101 08:00	2x Weapon A; 6x Weapon B	Target 1
1	1	A	1	SA	Base1	AO1	AC1	2	20200101 00:00	20200101 04:00	20200101 08:00	2x Weapon A; 6x Weapon B	Target 1; Target 4
1	1	B	1	SA	Base2	AO2	AC2	1	20200101 02:00	20200101 06:00	20200101 10:00	10x Weapon B	Target 3
1	1	B	1	SA	Base2	AO2	AC2	2	20200101 02:00	20200101 06:00	20200101 10:00	10x Weapon B	Target 2; Target 3
1	1	C	1	XINT	Base1		AC3	1	20200101 01:00	20200101 02:30	20200101 04:00	16x Weapon A	Target 6; Target 7
1	1	D	1	XINT	Base3		AC3	1	20200101 03:00	20200101 05:30	20200101 06:00	4x Weapon A; 4x Weapon B	Target 5; Target 8

For previous Title 10 Wargames, the process of creating the ITO was manual and tedious, captured by hand using Microsoft Excel. Multiple individuals contributed to building the ITO, providing scheduled mission data with varying degrees of completeness. Logging the ITO in a concise and easy-to-comprehend format assists operational planners in developing a logical plan and adjudicators in examining more engagements between Blue and Red teams to heighten the fidelity within the wargame.

The need for efficiency is paramount as more data is being captured with each wargame. Proper data management and data analysis are key to optimizing an operational plan from these large data sets. The development of a tool to assist and provide operationally feasible weapon-to-target solutions is required for future wargaming events.

WEAPON TARGET ASSIGNMENT PROBLEM

This paper focuses on a class of optimization concepts known as the Weapon-Target Assignment (WTA) problem. It seeks to determine the optimal assignment of a set of various weapons to a set of targets to cause maximum damage to the opponent. Any weapon can be assigned to any target, but each weapon has its own probability of destroying each target, and each target has a minimum number of weapon strikes required to be destroyed. Therefore, it is possible to calculate the total damage by summing the expected damages to the valued targets (Huaiping et al., 2006).

In the Air Force, the delivery methods of weapons are crucial. Different numbers of weapons and platforms are stationed at each base, and each platform has its own standard conventional loads (SCL), which define the mix of weapons the platform can carry. Maximizing damage to the enemy targets largely depends on efficiently moving the weapons within range. This requires a logically balanced flight schedule that maximizes the use of aircraft and allocates weapons to targets appropriately.

GENETIC ALGORITHM

A Genetic Algorithm (GA) is an optimization technique that is based on natural selection (Holland, 1992). A GA begins by creating a set of random candidate solutions to the problem. It generates each solution, called an individual, by randomly assigning values to the variables of the problem, and these variable assignments are the genetic makeup of that individual. Upon creating this first generation, the GA measures the individuals for certain criteria. It eliminates the worst performers and retains the best, which will be used as "parents" by passing on their genes to new offspring in the next generation. Offspring can be created in several ways: selection (reusing the same genetic makeup of parents), crossover (splicing genes from multiple parents), and mutation (changing parent genes). Upon creating a new generation, the GA measures the offspring against the same criteria, and once again selects the best performing individuals and discards the rest. The cycle of creating new generations will terminate after a set number of generations is reached, and the GA will return its best solution (Katoch et al., 2021).

Genetic algorithms have proven to be a powerful optimization technique for many different types of assignment problems that overwhelm conventional (global) solvers (Chu et al. (1997), Lee et al. (2003), Kline et al. (2019)). Due to the time constraints in a wargame, a metaheuristic approach, such as a GA, will efficiently explore the search space to discover near-optimal solutions. The structure of the GA employed for this problem (herein referred to as

KILLCHAIN) does not differ from most GAs in its format and is described below. KILLCHAIN was developed by the Air Force Futures modeling and simulation team in order to assign weapons to targets given a flying schedule and was deployed for use in an Air Force Title 10 wargame in February 2023.

Chromosome Structure

Each line of the ITO represents a single tail (aircraft) and the following information: pulse, takeoff base, platform, SCL, target, weapon type and quantity. Given the vast number of interdependent ITO line items, splitting this matrix into a more traditional chromosomal structure is a challenge. The population members of KILLCHAIN remain in matrix form for the duration of the algorithm to allow for simpler feasibility checks throughout. The genetic information contained in KILLCHAIN consists of the platform-SCL-weapon-target combinations. The pulse, base, range, etc. are static characteristics of platforms and are important in considering these combinations but are not variables themselves.

Generating a Feasible Population

The process of generating a feasible solution starts with move and pulse timing. To adhere to the operational plan, weapon-to-target pairs are generated by pulse, and within each pulse, all platform tails are randomly ordered. Then, in this order, each tail is examined to determine if it can attack the next available, highest-priority target that is not already sufficiently targeted by weapons. The determining factor for the platform is range, and its randomly chosen SCL is accepted if its weapons' range, effectiveness, and availability are adequate. If a tail is unable to attack the next available priority target, it will continue iterating through the JIPTL until a feasible target can be struck. Tails may not be assigned to a target for the following reasons: no weapons are available at the takeoff base, no targets are in weapon range, or no loadouts contain a feasible weapon with which to attack a target. Once all tails in every pulse have been assigned a SCL and paired to a target (if possible), a feasible ITO solution is generated.

Fitness Values

The score, or fitness value, for a typical WTA solution is based on an aggregation of target damage multiplied by the target value (Huaiping et al., 2006), but the JIPTL does not provide target values, only a prioritized list of targets. It is always better to destroy Priority 1 Target over Priority 2, but it cannot be deduced at what point it may be better to destroy a numerous subset of lower priority targets over Priority 1. As such, the assumption was made that it is always better to destroy the higher priority target. One way to ensure the score is greater for the destruction of a higher priority target, is by weighting each higher priority target as twice that of its successive lower priority target. The weights are in form 2^{n+1-i}, where n is the total number of targets and i is the target priority in the JIPTL. The final fitness value is calculated by summing the product of the damages to the targets and these weights, as shown in Equation (1).

$$Fitness = \sum_{i=1}^{n} 2^{n+1-i} \cdot damage_i \qquad (1)$$

For example, in a three-target scenario in which the ITO solution leaves Priority 1 Target fully destroyed, Priority 2 Target 50% destroyed, and Priority 3 Target 70% destroyed, the fitness value of the solution is shown in Equation (2).

$$\sum_{i=1}^{3} 2^{3+1-i} \cdot damage_i = 2^3 \cdot 1 + 2^2 \cdot 0.5 + 2^1 \cdot 0.7 = 11.4 \qquad (2)$$

This method of calculation allows for prioritization of targets to be incorporated in the fitness value, and the maximum fitness value, where all n targets are destroyed, is shown in Equation (3).

$$Fitness_{Max} = \sum_{i=1}^{n} 2^{n+1-i} \cdot 1 \qquad (3)$$

Death and Immigration

The user defines percentages of the population that will be removed (death) from each generation of KILLCHAIN, as well as a percentage of the population that will be newly randomly generated solutions (immigration). For example, in a population of size 100, if the user defines death and immigration percentages as 10% and 5%, respectively, then the 10 least-fit members will be removed from the population and five randomly generated solutions will be introduced to the population. As with most GA implementations, KILLCHAIN removes the population members with lower fitness as opposed to random removal because the benefits of keeping the high-fitness population members outweighs the risk of losing good traits from lower-fitness population members. It is important that the immigration percentage is smaller than the death percentage, otherwise the population will have no room for child solutions via the crossover method, since we assume a fixed population size.

Crossover Method

The first step in creating child solutions via crossover method is to determine the ITO parents. For simplicity's sake, all population members remaining after death and immigration are eligible parents, and two parents are required for the generation of a single child using a customized crossover method.

Once two parent ITOs have been selected, the child solution is defined to be the merged result of the two ITOs in which every unique row is passed to the child ITO; this child solution may be adjusted in a following step if deemed infeasible. For example, if two parent ITOs each contain nine completely unique lines to one another, and their 10th lines are identical, then the child ITO will have a resulting 19 total lines. The child ITO line items are sorted by the target priority in descending order and then the following steps occur to ensure a feasible ITO:

1) Examine the Tail-to-SCL Pairings: If a particular tail is loaded with SCL 1 in the Parent 1 ITO but SCL 2 in the Parent 2 ITO, this creates an infeasibility in the child ITO since each tail can only carry one SCL. If this is determined to happen in the child ITO, the row containing the higher-numbered index (lower priority target) is removed.
2) Examine the Distances Traveled by Each Tail: Since the ITOs of two parents have been merged, it is possible that a tail is now traveling to more targets in the child solution than it did in either parent alone. Tails have maximum travel distances based on fuel capacity, so the flights to targets that would exceed the fuel capacity are removed from the ITO.
3) SCL Weapon Totals: If two parents used the same SCL for a tail (and did not violate (1)), but attacked different targets with that SCL, it is possible that the child solution uses more weapons than the SCL allows. For example, if a tail uses SCL 1 to attack Target 1 with 5 Weapon Xs in the Parent 1 ITO, but uses SCL 1 to attack Target 2 with 5 Weapon Xs in the Parent 2 ITO, and SCL 1 only contains 8 of Weapon X, then an infeasibility is occurring in the child ITO. The ITO row with a lower target priority is decremented of Weapon X until only 8 total are used from the violating tail. (2 fewer Weapon Xs would be assigned to the lower priority target.)
4) Base Weapon Allocations: The two parents are feasible in the number of weapons they are using to supply aircraft SCLs from each base. However, when the parent solutions are merged into a child solution, the child may be overusing weapons from some base inventories. Step (3) helps to curb this issue, but the possibility of an infeasibility due to this condition may still occur, so any tails that contribute to a violated base-weapon constraint will be randomly decremented until the base supply is not negative.

Termination Criteria

KILLCHAIN will terminate for one of three reasons: 1) maximum run time is met, 2) maximum iterations are reached or 3) the population converged. Due to the time constraints in a wargaming environment, it is imperative for KILLCHAIN to return its best solution within a given time frame. If given a larger time frame, the user instead can define a set number of generations. Finally, if the population converges and more than a certain, user-defined percentage of the rows of the population ITOs are identical, then KILLCHAIN will terminate.

Pseudocode

Algorithm 1 KILLCHAIN

```
Generate random population
Calculate fitness values
while Generation count & time limit have not been exceeded do
    Remove appropriate amount of population members due to death
    Add appropriate amount of randomly generated immigrants
    for Each new child that needs to be created do
        Choose two random population members to be parents
        Use Crossover Method to generate child solution
        Ensure feasibility of child solution using various techniques
        Add child solution to population
    end for
    Calculate fitness values of population
    Sort population by fitness values (in descending order)
end while
```

Figure 1. KILLCHAIN Algorithm

Testing

KILLCHAIN is tested on three sizes of the underlying problem (small, medium, and large). The parameters for each problem size are shown in Table 2. The ranges listed in Table 2 represent the randomly sampled range for each weapon type.

Table 2: Testing Parameters

Parameter	Small	Medium	Large
Randomly Generated Instances	10	10	1
Areas of Operation	10	30	50
Targets	100	200	500
Bases	5	15	25
Platform Types	10	20	30
Platform Quantity per Mission	4	4	4
Integrated Air Defense Systems	10	20	30
Weapon Types	15	20	25
SCLs per Platform Type	5	7	9
Minimum Weapons Required to Kill Each Target	10-20	15-25	30-40
Weapons Available at Each Base	50-100	100-500	500-800
Length of Flying Schedule (Missions)	100	150	200

At the onset of a new test run, the minimum number of each weapon type required to destroy each target is randomly generated and not uniform. This mimics the reality of JIPTL targets requiring different numbers and types of weapons due to their different strengths, vulnerabilities, size, resilience, and defensive measures. Moreover, when a target requires a certain number of weapons to be "destroyed", this means the target requires a sufficient number of weapons to achieve a pre-defined confidence that it would be completely destroyed. But the confidence is never 100%, so even when the sufficient number of weapons is assigned to that target, there is still a chance that the target would not be destroyed.

For each test, the results of interest are the number of generations KILLCHAIN was able to achieve in one hour (max 100), how many targets the ITO can fully destroy if executed properly, and the fitness values of the members in each generation. Each instance was solved on a 3.6 GHz PC with 32.0 GB of RAM and an AMD Ryzen 7 3700X 8-Core

Processor using RStudio (Version 2022.07.1 Build 554) with R (Version 4.2.1) implementing parallel processing for added efficiency.

Results

The complexity of this problem, given the quantity of parameters and the vast solution space, leads to the following results. Initial 40-member populations took approximately 20, 60, and 240 minutes to generate for small, medium, and large instances, respectively. (Relative to other GAs, KILLCHAIN's initial population takes a long time to generate due to the feasibility checks and corrections. Without these, KILLCHAIN would actually take much longer, wasting time comparing infeasible solutions.) After generating the initial population, the subsequent generations were allowed a runtime of one hour and then returned the most fit population member (ITO). While arbitrarily set, the one hour of runtime is used because it closely mimics "game speed", or the speed at which results need to be produced during a Title 10 Wargame. In an ideal situation the KILLCHAIN output would be compared to the known global solution. Unfortunately, this problem has yet to be modeled as a nonlinear, mixed-integer program, and a global optimal comparison is not possible. However, KILLCHAIN output can be compared to a known ideal point, one in which all targets from the JIPTL are destroyed.

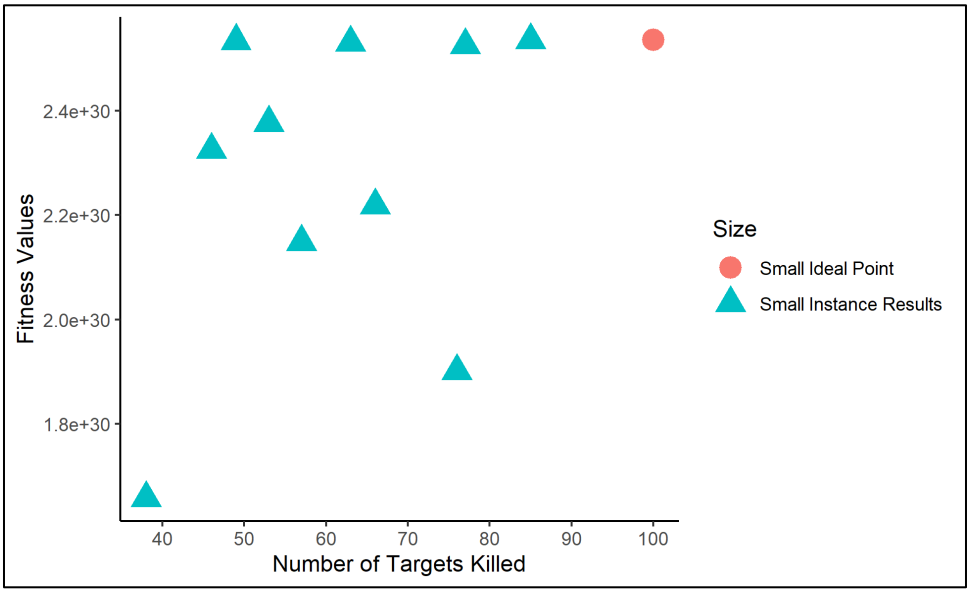

Figure 2. KILLCHAIN Solutions for Small Instance

Figure 2 depicts the KILLCHAIN solutions returned for 10 randomly generated small instances compared to the ideal solution point. This plot indicates that randomly generated instances can vary widely in their complexity for this particular problem set, as evidenced by the wide range of fitness values and number of targets destroyed. Figures 3 and 4 reveal insights of another generated instance: KILLCHAIN found a great solution in its first generation and could not improve upon it over the course of the 19 generations. However, the average fitness values of the 40 population members (ITOs) increased, demonstrating convergence of the entire population and continual improvement of the algorithm's progress. The 19th generation contains more than one Pareto optimal solution and provides multiple good choices in ITO selection. Figure 3 displays the number of targets killed in addition to the fitness values observed in Figure 4. The 17th generation is clustered in a desirable region of this solution space, and a Pareto Front can be identified.

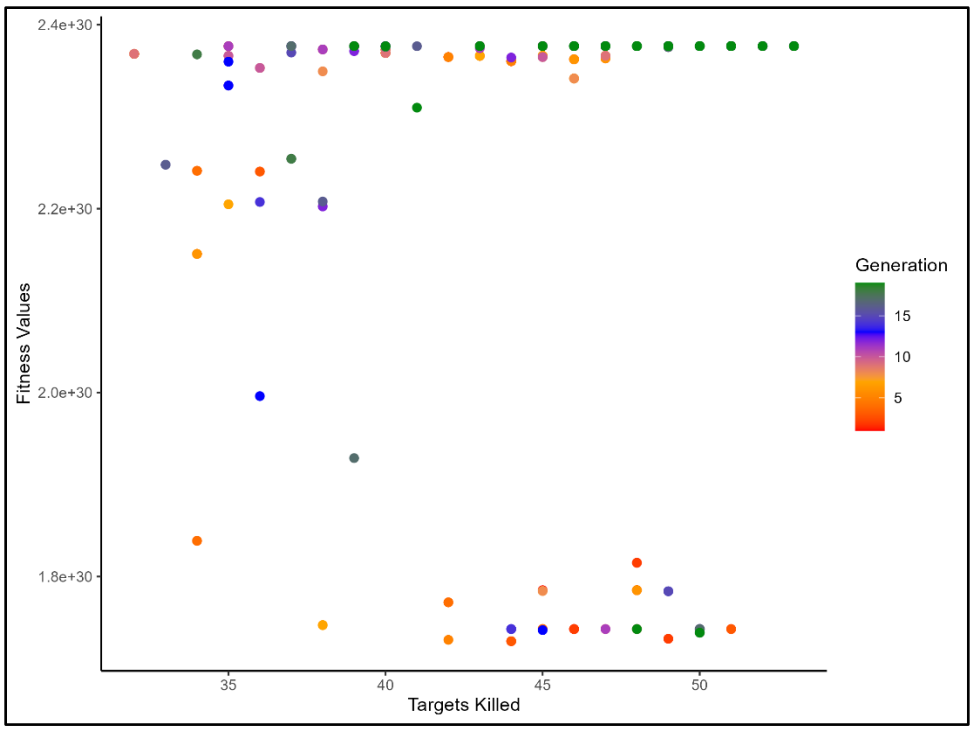

Figure 3. Small Instance - Fitness Values vs. Targets Destroyed

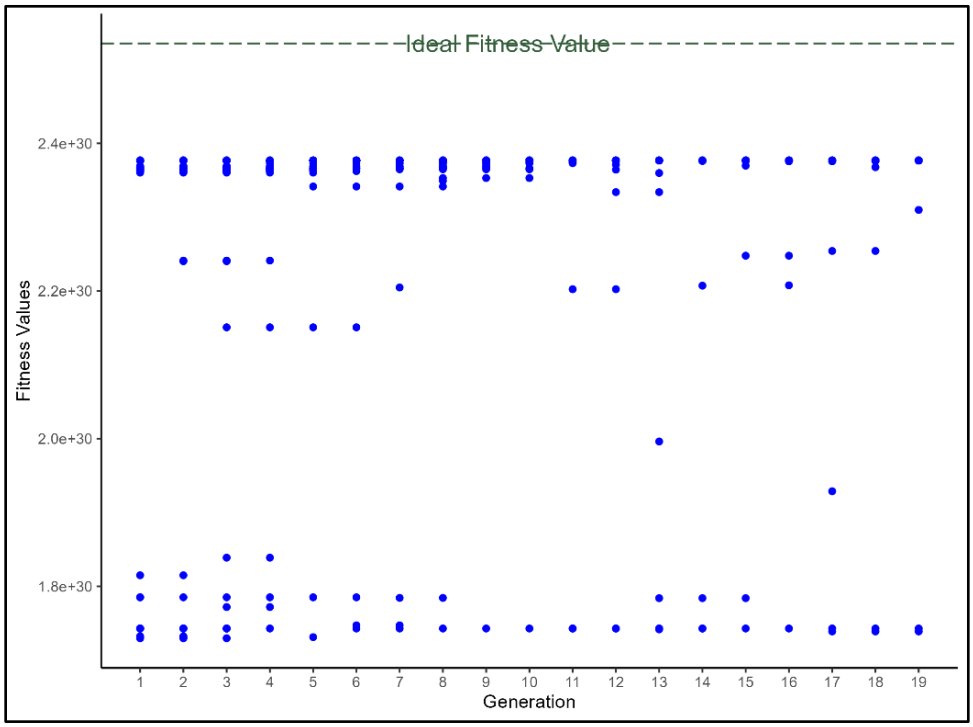

Figure 4. Small Instance – Fitness Values vs. Generation

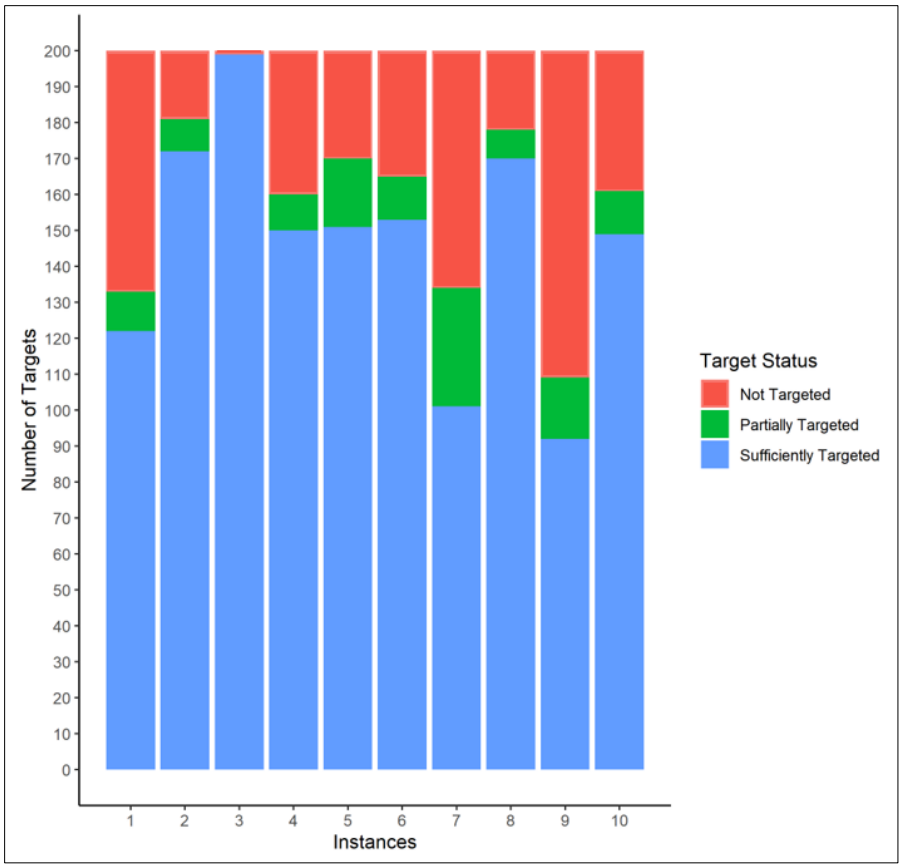

Figure 5. KILLCHAIN Solutions for Medium Instances

Figure 5 depicts KILLCHAIN's performance on 10 randomly generated medium instances and specifically displays the total count for each target health status. While the solutions returned by KILLCHAIN for each of the 10 instances differ slightly, they all had the following in common: targets 1-16 were targeted enough to be rendered fully destroyed, and they also sufficiently targeted 40 or more of the top 50 priority targets. This is evidence that KILLCHAIN is performing well with respect to examining the solution space, as the differing solutions still result in similarly large fitness values that are greater than $3x10^{60}$.

Figures displaying the performance of KILLCHAIN on the large instance are omitted because after a four-hour run time to create a population, KILLCHAIN is only able to iterate through two generations. These generations do not differ except for a few population members. We thus conclude that the medium-sized problem is the limit of KILLCHAIN's useful ability for wargaming purposes.

MONTE CARLO SIMULATION & ANALYSIS

To confirm KILLCHAIN's solutions are continually improving not only in fitness but in practice, a test is designed to simulate the individual interactions of the weapon-target parings. Since there is a damage probability associated with each pairing (originally used in calculating the number of weapons needed for each target), it can be modeled with a random number generator to determine an outcome of the interaction. Sampling ITO solutions across generations as well as across different instances is important to ensure results are not isolated trends. For every sampled solution, each weapon-to-target interaction is simulated and the outcomes are recorded. Targets that are damaged are counted toward an aggregated fitness score as outlined in Equation (1). In Monte Carlo fashion, each simulation runs 1,000 times to produce different outcomes and calculates the average fitness of the 1,000 scores. The graph below is sorted by KILLCHAIN fitness from least to greatest to highlight the correlation.

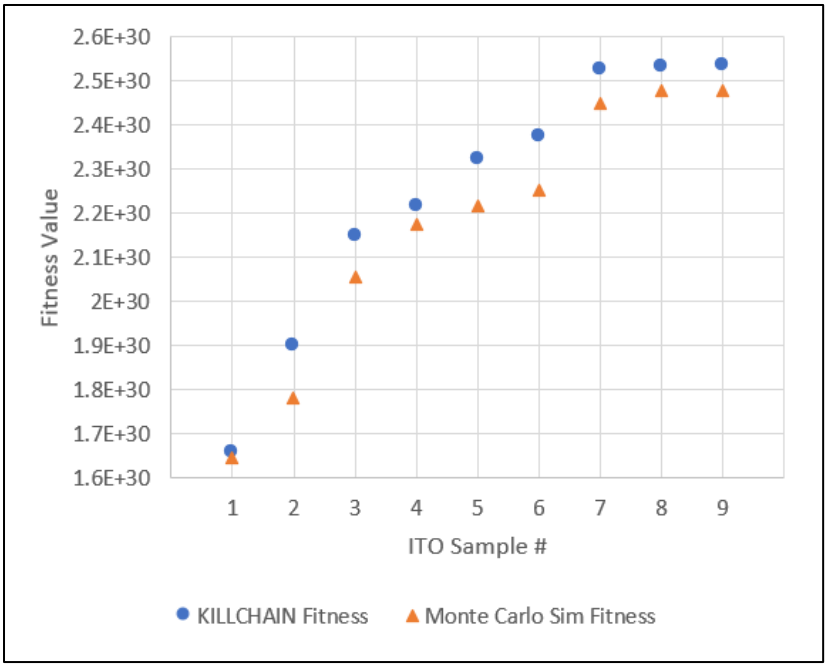

Figure 6. Fitness Values for Sample ITOs

Comparing the fitness values generated by KILLCHAIN and the average fitness values generated by the Monte Carlo simulation, a direct correlation is observed: as an ITO's KILLCHAIN fitness increases, so too does its simulation fitness. This demonstrates that the solutions with higher KILLCHAIN fitness values are correlated with better performance in a wargame. Also, the simulated fitness values are noticeably smaller than their KILLCHAIN fitness values. This is likely due to the KILLCHAIN fitness function not accounting for the small chances that targets deemed sufficiently targeted are not destroyed. When each weapon strike is simulated over many runs, these small chances become more apparent. Sometimes targets that KILLCHAIN plans to fully destroy are only partially damaged in the Monte Carlo simulation, hurting the fitness score. However, it is unwise to use too many munitions on each target (overkill), as it will waste munitions that could otherwise be utilized more effectively.

OPERATIONAL FEASIBILITY

While KILLCHAIN seeks to optimize an ITO based on killing the greatest number of higher-priority targets, the solution is not necessarily best in practice. If a higher-priority target is deeper within the enemy threat, it is often advantageous to first to eliminate targets on the perimeter of the enemy's position for increased probability of kill and survivability purposes. While the JIPTL provides that ordered target list, the order of targets to attack may be different to achieve operational objectives. This can provide insight on JIPTL order versus attack order and how the JIPTL can be organized. Operational feasibility is also affected by: pulse timing and the aircraft designated to perform missions during those pulses, revised numbers of weapons to destroy a target based on sensing grid and enemy defenses, and aircraft and weapon range limitations. Knowing these operational constraints, KILLCHAIN can provide a solution space that demonstrates the effects of different locations of areas of operation (AOs), quantities of weapons, and target priorities. This solution space can determine where superior AO placement for increased survivability as well as base weapon inventory amounts to allow for more target engagements. This can further lead into planning missions that package with bomber and other long-range strike missions (i.e., planning feasible offensive counter-air missions to escort bombers).

CONCLUSION

The KILLCHAIN algorithm was developed to find feasible and efficient ITOs faster than human wargame planners have been able to achieve in the past. The algorithm follows a standard GA formula with a few customized elements and is tested on three different sizes of a variation of the WTA problem. Small, medium, and large instances are randomly generated, and KILLCHAIN is assessed on both the fitness values achieved as well as the number of targets

destroyed after one hour of runtime. In the 10 randomly generated small instances, KILLCHAIN performed well, with about half of the returned solutions within an acceptable range of the ideal solution. In the 10 randomly generated medium instances, KILLCHAIN's performance is still acceptable as all solutions kill at least 50 out of 200 targets, including the top 16 in every instance. The large instance proved to be too difficult for KILLCHAIN, as it failed to generate any meaningful growth across the two generations that it completed in one hour of runtime. However, Title 10 Wargames have not presented a problem as complex as the large test instance, so they are currently within KILLCHAIN's capability. In testing KILLCHAIN's solutions in a Monte Carlo simulation, the higher-ranked solutions performed better in the simulation than the lower-ranked solutions, demonstrating KILLCHAIN's value added in a wargame, where every planned strike is adjudicated as realistically as possible under the time constraints.

Future work on the KILLCHAIN algorithm will focus on improving its efficiency, specifically in finding random solutions to seed the initial population. Allowing some level of infeasibility in population members incurs the risk of producing an infeasible final solution, but may drastically speed up the GA. This was not attempted for the current version of KILLCHAIN, as the feasibility of solutions was necessary for its use in the 2023 Title 10 Wargame. KILLCHAIN is slated to be used for the next Title 10 Wargame, to be conducted in mid-2024.

DISCLAIMER

The discussion of non-federal entities, methods, products, or services does not imply any endorsement by the Department of the Air Force, the Department of Defense, the U.S. Government, or the organizations that employ the authors.

REFERENCES

Chu, P. C., & Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, *24*(1), 17-23.

Georgetown University Wargaming Society. (2020, October 12). *USAF Title 10 Wargaming with Mitch Reed* [Video]. YouTube. USAF Title 10 Wargaming with Mitch Reed

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, *267*(1), 66-73.

C. Huaiping, L. Jingxu, C. Yingwu and W. Hao (2006). Survey of the research on dynamic weapon-target assignment problem. *Journal of Systems Engineering and Electronics*, vol. 17, no. 3, pp. 559-565, doi: 10.1016/S1004-4132(06)60097-2.

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, *80*, 8091-8126. https://doi.org/10.1007/s11042-020-10139-6

Kline, A., Ahner, D., & Hill, R. (2019). The weapon-target assignment problem. *Computers & Operations Research*, *105*, 226-236.

Lee, Z. J., Su, S. F., & Lee, C. Y. (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.

United States Air Force. (2018, September 28). Air Force Doctrine Publication (AFDP) 3-60 Joint Targeting. https://jdeis.js.mil/jdeis/new_pubs/jp3_60.pdf

United States Air Force. (2020, December 1). JP 5-0, Joint Planning, 01 December 2020 - Federation of American ... https://irp.fas.org/doddir/dod/jp5_0.pdf

United States Air Force. (2021, November 12). Air Force Doctrine Publication (AFDP) 3-60 Targeting. https://www.doctrine.af.mil/Portals/61/documents/AFDP_3-60/3-60-AFDP-TARGETING.pdf