Optimization of Weapon-Target Pairings Based on Kill Probabilities
Goal: for given weapons, targets, and kill probabilities, satisfy desired kill probabilities while minimizing overkill. The purpose of this paper is optimizing existing approaches rather than generating/demonstrating new ones.

Kill probabilities are not additive.
Uses a new combinatorial algorithm derived from the auction algorithm.
NP-hard problem with multiple, non-identical weapons, so uses a heuristic rather than an algorithm.
The probability of a set of weapons killing a target is higher than the probabilities of each individual weapon-target pair (WTP) in that set multiplied together resulting in a kill.
The optimization problem can be formulated as:

$$\text{minimize} \sum_{i=1}^{n} \left| Q_i\left(q(i)\right) - K_i \right|$$

subject to

$$Q_i\left(q(i)\right) \geq K_i \text{ for } S_i\left(q(i)\right) \neq \emptyset \tag{4}$$

$$S_i\left(q(i)\right) \bigcap S_{i'}\left(q(i')\right) = \emptyset \text{ for } i \neq i'. \tag{5}$$

Where K_i is the desired kill probability for a target, Q_i (q(i)) is the lower bound for kill target i with weapons $w_{i1} \ldots w_{iq(i)}$ and S_i (q(i)) is the subset of weapons $\{w_{i1} \ldots w_{iq(i)}\}$ being used to kill target i. Total probability of a kill must meet or exceed desired probability (constraint 4) and no weapon can be assigned to multiple targets (constraint 5).

Phase 1 of the algorithm is to generate an Attack Guidance Table (AGT),

Algorithm 1 Generate *AGT* (*AGT_Gen* algorithm)

Input: targets: $t_1, t_2, \ldots, t_n$,
weapons: $w_1, w_2, \ldots, w_m$,
upper bound for $q(1), q(2), \ldots, q(n)$: U,
effects: $P_1(w_1), P_1(w_2), \ldots, P_1(w_m)$,
$P_2(w_1), P_2(w_2), \ldots, P_2(w_m)$,
$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$
$P_n(w_1), P_n(w_2), \ldots, P_n(w_m)$,
desired effects: $K_1, K_2, \ldots, K_n$.

Output: *AGT*.

Step 1. Initialize *minimal set* matrix A by setting $a_{i,j} := \emptyset$ for each $a_{i,j} \in A$.

Step 2. Initialize benefit matrix B by setting $b_i^j := 0$ for each $b_i^j \in B$.

Step 3. Dispatch execution of depth-first search for the minimal sets to n cores.

Step 4. Wait for completion of depth-first searches from all n cores.

Step 5. Generate *AGT* by joining A and B, and STOP.

The purpose of this table is to construct minimal sets of weapons to assign to each target and record the benefits of assigning these minimal sets to the target.

However, the number of minimal sets grows exponentially with the number of weapons and targets, and is the main bottleneck for running this algorithm. As such, a global maximum number of weapons that can be assigned to a target, *U*, is specified by the user. *U* must be below approximately 30 for the algorithm to be practical.

This algorithm is designed for multicore processors where each target is given a dedicated core. Algorithm 2 generates all minimal sets for a target using depth first search.

Algorithm 2 Generate all minimal sets for target t_i by the ith core

Input: target: t_i,
weapons: $w_1, w_2, \ldots, w_m$,
upper bound for $q(i)$: U,
effects: $P_i(w_1), P_i(w_2), \ldots, P_i(w_m)$,
desired effect: K_i.

Output: matrices: A, B.

Step 1. Execute depth-first search for all the minimal sets with pruning based on $q(i) > U$ for target t_i.

Step 2. For every jth found, a minimal set that satisfies $|S_i^j(q(i))| \leq U$ do:

(i) save minimal set in A by setting

$$a_{i,j} := S_i^j(q(i));$$

(ii) save benefit in B by setting

$$b_i^j := 100 - Q_i^j(q(i)) + K_i;$$

Step 3. STOP.

Algorithm 3 assigns minimal sets to targets, attempting to minimize “adjusted benefit,” which is defined as follows:

The adjusted benefit a_{ji} of assigning minimal set S_{ji} to target t_i is equal to benefit b_{ji} reduced by the currently assigned scores for targets that have been assigned and share at least one weapon with S, where the scores are assigned in steps 1, 5, and 6.

Algorithm 3 Serial optimizer (*Min_Kill* algorithm)

Input: *AGT*.
Output: Assignment of minimal sets to targets.

Step 1. Initialize target scores $s_1 := s_2 := \ldots := s_n := 0$.

Step 2. If there exists unassigned target t_i with the associated minimal set of positive adjusted benefit, then execute Steps 3–9. Otherwise, STOP.

Step 3. Find j that maximizes adjusted benefit a_i^j for t_i.

Step 4. Find $j' \neq j$ that maximizes adjusted benefit $a_i^{j'}$ for t_i.

Step 5. For every target t_k (where $t_k \neq t_i$)) with currently assigned weapon $w_{k'} \in S_i^j(q(i))$ do:

(i) $s_i := s_i + s_k$;

(ii) reset score $s_k := 0$;

(iii) unassign assigned minimal set S_k;

(iv) for every weapon $w_{k''}$ currently assigned to t_k unassign $w_{k''}$.

Step 6. If j' exists, then calculate score for target t_i as follows:

$$s_i := s_i + b_i^j - max(0, a_i^{j'}) + \epsilon.$$

Else, calculate score for target t_i as follows:

$$s_i := s_i + b_i^j + \epsilon.$$

Step 7. Assign minimal set $S_i^j(q(i))$ to target t_i;

$$S_i^j(q(i)) \leftrightarrow t_i.$$

Step 8. For every weapon $w_{k'} \in S_i^j(q(i))$ assign $w_{k'}$ to target t_i; $w_{k'} \leftrightarrow t_i$.

Step 9. Go to Step 2.

This algorithm converges in a finite number of steps.

Algorithm 4 does the same thing as algorithm 3, but in parallel rather than in serial.

Algorithm 4 Parallel optimizer (*Min_Kill_Par* algorithm)

Input: *AGT*.

Output: Assignment of minimal sets to targets.

Step 1. Initialize target scores $s_1 := s_2 := \ldots := s_n := 0$.

Step 2. If there exists unassigned target t_i with associated minimal set of positive adjusted benefit then execute Steps 3–11. Otherwise, STOP.

Step 3. For every unassigned t_r dispatch to core r, search for $a_r(1), a_r(2)$.

Step 4. Wait until all dispatched tasks to the cores are completed.

Step 5. Pick unassigned target t_i based on *a priori* selection criterion:

(i) $\max_r(a_r(1))$; or

(ii) $\max_r(G_r)$.

Step 6. Identify j that maximizes adjusted benefit a_i^j for t_i.

Step 7. For every target t_k with currently assigned weapon $w_{k'} \in S_i^j(q(i))$, do:

(i) $s_i := s_i + s_k$;

(ii) reset score $s_k := 0$;

(iii) unassign assigned minimal set S_k;

(iv) for every weapon $w_{k''}$ currently assigned to t_k unassign $w_{k''}$.

Step 8. If j' exists then calculate score for target t_i as follows:

$$s_i := s_i + b_i^j - max(0, a_i^{j'}) + \epsilon.$$

Else, calculate score for target t_i as follows:

$$s_i := s_i + b_i^j + \epsilon.$$

Step 9. Assign minimal set $S_i^j(q(i))$ to target t_i.

Step 10. For every weapon $w_{k'} \in S_i^j(q(i))$ assign $w_{k'}$ to target t_i.

Step 11. Go to Step 2.

Algorithm 5 is designed for deconfliction (ie, not destroying things like friendly units or schools). Probably not relevant to our case.

Algorithm 5 Set deconfliction flag d_i^j for minimal set S_i^j

Input: minimal set: S_i^j,
lethality: $f(w_1, t_i), f(w_2, t_i), \ldots, f(w_m, t_i)$,
friendly units: $u_1, u_2, \ldots, u_x$.
Output: Deconfliction flag d_i^j in matrix D.

Step 1. Set $R_i :=$ LARGE.
Step 2. For every friendly unit u_k do:
calculate $R_i := min(R_i, g(u_k, t_i))$.
Step 3. Set $d_i^j := 0$.
Step 4. For every weapon w_k in minimal set S_i^j do:
if $f(w_k, t_i) > R_i$ then

$$d_i^j := 1.$$

Step 5. STOP.

This algorithm is fast enough (for up to 120 weapons/targets) to be usable in military engagements and is noticeably better than a simple greedy algorithm.