Proceedings, The Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-16)

Implementation of an Automated Fire Support Planner

Byron R. Harder, Imre Balogh, and **Chris Darken**
Naval Postgraduate School, MOVES Institute,
1 University Cir., Monterey, CA 93943, USA

Abstract

Although the employment of fire support is a staple of modern military doctrine, today's constructive combat simulations depend on meticulous human input to generate any appropriate fire support plans. This status quo can be improved through AI techniques. We implement models of tactical risk, reduction of risk, and suppression effects in a representative combat simulation, as well as a greedy fire support planning algorithm that leverages these concepts. The algorithm is theoretically non-optimal, but testing shows that the resulting fire support plans are effective at improving simulated combat results and have some realistic emergent properties. The practical running time of the planner is less than 20 seconds for a company-sized unit, including navigation graph setup. The planner's best-first approach scales naturally in more time-constrained environments.

Introduction

Although the employment of fire support is a staple of modern military doctrine, today's constructive combat simulations depend on meticulous human input to generate any appropriate fire support plans. For the scenario designer (level designer), this involves the direct encoding of fire support unit locations, waypoints, target selection, and firing times. These static plans have several inherent problems: they are time-consuming to generate, brittle to changes in the supported maneuver plan or unexpected events in the simulation, and they depend entirely on the tactical skill and system proficiency of the designer. Reactive fire support, in contrast, often fails to make the most efficient use of resources.

Although combat involves a wide variety of situations and missions, we limit our scope here to fire support planning in support of *deliberate attack* operations, where the attacker has reasonable knowledge (military intelligence) of enemy defensive positions and capabilities. Good information about the enemy should allow the attacker to devise an effective plan, but actually generating such a plan involves some work. We assume that a maneuver plan, the set of movement instructions for units whose purpose is to engage and assault enemy positions at close range, is provided to us as input. Our goal is to automatically generate a fire support plan, a set of movement and firing orders for the remaining available units that improves the achievement of mission objectives. These units can include those designed for fire support, such as machine gun squads and artillery batteries, as well as maneuver-type units such as rifle squads that are not fully tasked by the maneuver plan.

Our fire support plans rely on the doctrinal concept of suppression. Suppression is the temporary reduction of a unit's combat power when it is subject to heavy fire. The idea is that the noise, dust, fragmentation, overpressure, and resulting sense of danger from nearby munition impacts will cause the targets to tuck into their protective positions, button up viewports, and so on, losing some of their focus on targeting. Suppression is distinguished from attrition by the fact that it does not need to cause any casualties. Maneuvering forces depend on suppression, provided by supporting units, to reduce the effects of enemy fire when they need to cross exposed terrain. Prepared fighting positions can be very effective at reducing the destructive effects of support weapons—a likely reason to conduct an assault in the first place—so suppression is often the best an attacker can get from fire support. In many games and simulations, suppression is not expressly modeled, but left to the hide-or-fight decision process of human players. Our interest is in more strategic "constructive" applications where small unit actions are computer-controlled, so we have modeled suppression effects directly. Our work may be of interest to developers of real time strategy (RTS) or other combat-themed games with suppression effects.

Related Work

This paper decribes an implementation of theoretical work on automated fire support planning (Harder and Darken 2016), which provides a mathematical description of the problem and offers an abstract greedy algorithm to address it. Although it provides a basis, it does not include enough detail to realize a functioning solution and does not have an evaluation component. Here, we address the missing details and test the theory.

The case for modeling suppression is originally made by Hughes (1995), who points out that real-world casualties are often much less than predicted by mathematical combat models. He then extends the Lanchester (1916) model to include a continuous suppression effect and shows that

firepower effectiveness, not force size, becomes the squared term. Our suppression model is more discrete and spatially-dependent than this, but the supporting arguments are fundamental for the direction of our research.

Tactical pathfinding is foundational for the techniques described below. Path planning while avoiding enemy visibility is described in the work of Rowe and Lewis (1989). Additional techniques, such as accounting for target acquisition time and observer mobility, are explained by van der Sterren (2002). Cooperative pathfinding (Silver 2005) is a closely related problem, useful for keeping units well-dispersed. Some air attack planning research has used probabilistic risk over a proposed flight plan (Secarea and Krikorian 1990; Gu et al. 2012). More recently, the MECH framework is used to reason about risk to a route based on parameters such as visibility (Wang et al. 2015). Our work differs from these approaches by allowing the risk calculation to be modified by suppression effects.

Recently, RTS AI research has pursued combat outcome prediction (Churchill and Buro 2013; Stanescu, Barriga, and Buro 2015; Uriarte and Ontaon 2015). Our efforts are related, but Blizzard Entertainment's StarCraft game, the chosen platform for much of that research, does not have a generic suppression model like the one we describe below.

We build upon a method for planning a single suppression task (Straatman, van der Sterren, and Beij 2005), extending to a collection of fire support providers and allowing movement to new positions. Our approach aims to be more holistic by reasoning about the reduction of risk to the friendly force's mission.

A Representative Combat Simulation

We have chosen to implement a new, relatively simple combat simulation system in which to prototype our planner. In general, production military simulations have large code bases, and most do not allow source code access—at least not to the wide research community. Our system is built on a modern game engine (Unity3D). We note that behavioral development in a game engine prior to implementation in a simulation system is itself the focus of a separate research effort (Miller 2016). In an effort to keep our work relevant to military modeling and simulation, our combat model is structurally similar to that of a production system in use by the U.S. Army and Marine Corps, called COMBAT XXI (CXXI) (Balogh and Harless 2003). We call our system WOMBAT XXI (WXXI).

Entity is our term for a modeled individual person (or vehicle). Like many modern combat simulations, WXXI maintains a separate object for each entity. Similar to CXXI, it includes a hierarchical unit organizational structure for both the friendly (attacker) and enemy (defender) side, and it has *formation* objects (collection classes) for moving entities from one or more units together as a single aggregate group. We have included a software class for *operational tasks*, which provide instructions to units at scheduled times, and embedded this in a hierarchical task network (HTN) similar to that described by van der Sterren (2013). Although the fire support planner does not use the full power of HTNs, working in that structure helps in integrating with a maneuver planner. We view this effort as complimentary to such a tool. For the experimentation phase of our research, we run WXXI scenarios in batch mode rather than realtime. Results are invariant to the graphical frame rate because we have implemented a discrete event simulation system through which all substantive combat events must pass.

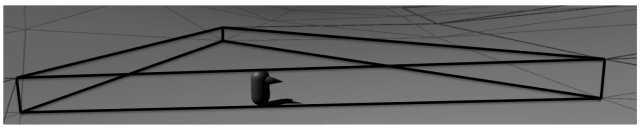

Figure 1: A graph node and entity

Our entities sense and engage targets individually. We include a mode that forces them to target members of a specific enemy unit, which is necessary for executing a fire support mission reliably. Entities have severely restricted weapon ranges while moving, as in the real world. They may be standing, prone, or dug in, which provides hit protection of 0.0, 0.9, and 0.999 respectively. This protection is ignored at extremely close range (configured to 10m). All enemy defenders are dug in, so the attackers have quite a challenge in surviving an assault, even with the 3:1 odds we have set up (according to common military wisdom for an attack). We have also implemented an aggressive movement mode for the assault phase, which helps attackers reach close enough range to dislodge the defenders. The intent of this model is to accentuate the tradeoffs between attacking and defending: tactical maneuverability versus efficiency of combat power.

Our terrain is based on real world elevation data with coarse polygons at a resolution similar to that of the military simulation systems we have in mind (triangles are 31m on the short edges). The planning algorithm is agnostic to the terrain resolution.

Modeling Risk and Suppression

A fundamental component of the planning system is the *annotated mobility graph*, our specialized navigation mesh. Our scenario maps tend to have undulating terrain, which affects line of sight determination. Each node of the graph is a volume bounded below by a triangle of the terrain skin and above by an equivalent triangle shifted up by the height of an entity (Fig. 1). The latter is called the *upper triangle*. Before any planning is done, we annotate each node with references to the defending entities that can see into its volume. Defenders are assumed to remain in place—a fundamental assumption for the deliberate attack, and a reasonable one (we claim) since the 0.999 protection bonus is lost by any entity that moves from its dug-in position.

Although we use entity-level visibility to annotate the mobility graph, we reason at the small unit level (groups of 2-40 entities organized by the scenario designer) for most of the planning work. Whenever a defending entity has visibility into a graph node, we mark that node with a reference to that entity's unit. We also mark the defending units that can be targeted from each node. A unit is considered potentially

targetable from a node if all of its members can be targeted from the corners and geometric centroid (average of the corners) of the node's upper triangle.

The navigation penalty of each node n is proportional to the expected losses to a group of friendly entities located inside it for the maximum time to traverse it. We make this calculation using ground truth: the same probability of hit and kill functions that the simulation will use to adjudicate combat results. Let E and F (for enemy and friendly) be the sets of defending and attacking units, and let $\tau_{f,z}$ represent the maximum time for a unit $f \in F$ to traverse the maximum length edge (line segment) z of n. We define $\psi : \{E \times F \times N\} \rightarrow \mathbb{R}$ as the *primitive risk value* function (N is the node set of the mobility graph), modeled as

$$\psi(e, f, n) = L(e, n)|e|\rho_e p_{HK}(e, f, d(e, \vec{c}(n)))\tau_{f,z}. \quad (1)$$

$L(e, n) = 1$ if e is in the visibility annotation of n (and 0 otherwise), $|e|$ is the member count of e, ρ_e is e's rate of fire per member, and p_{HK} gives the probability of hit and kill for a member of e firing at a member of f at the distance to $\vec{c}(n)$, the centroid of n's upper triangle. In short, ψ gives the expected number of kills e will inflict on f in $\tau_{f,z}$ seconds under the assumption that e is at full strength and never runs out of targets. The traversal (g) cost for a partial path $\pi = (n_1, ..., n_k)$ in the tactical pathfinding search (see van der Sterren 2002) is a linear combination of distance and $\Sigma_{i=1}^{k}\psi(e, f, n_i)$; the balance between the edge and node cost is set to heavily favor concealment over speed. Once a node path is found, we apply postprocessing[1] to get more direct routes without changing the node sequence. We then compute the location and time at which the unit's formation leader will transition between nodes or cross a discontinuity in p_{HK}, such as maximum weapon range, for any relevant defending unit. The route can now be described as a sequence of line segments mapped to time intervals.

We use a four-tiered data structure to represent the tactical risk of a planned operational task. Starting from the highest tier, a *risk set* is simply a set of *risk intervals*. A risk interval $r = (e, f, t_a, t_b, s)$ represents the risk presented by defending unit e to attacking unit f during time interval $[t_a, t_b]$, and contains a nonempty indexed set of *risk subregions* $s = \{s_1, s_2, ...\} = \{(t_1, t_2, z_1), (t_2, t_3, z_2), ...\}$ such that $\{t_1, t_2, t_3, ...\}$ exactly partitions time interval $[t_a, t_b]$. Each component $z_i = \{z_{i,1}, z_{i,2}, ...\} = \{[t_{i,1}, t_{i,2}], [t_{i,2}, t_{i,3}], ...\}$ is an indexed set of *risk segments* further partitioning its subregion's time interval $[t_i, t_{i+1}]$. When a risk interval is first created, its time interval $[t_a, t_b]$ corresponds to an unbroken period of time that e can target f based on the annotations of nodes in a route sequence. A new risk interval has just one subregion, whose segment endpoints (and times) are determined by the node and weapon range crossing points described above. Each operational task in the maneuver plan is assigned a risk interval set containing all risk intervals for its routes and static positions. This set can contain risk intervals from several different defending units, which may overlap in time. The risk set for the entire plan is the union of all risk intervals of its component tasks.

We produce a numerical score for each risk interval in a manner similar to the primitive risk function for pathfinding edges (pathfinding computations are discarded because path modifiers make changes to the route). The *base risk value* $\Psi(z_{i,j})$ of a segment is the integral of the instantaneous risk over its time interval $[t_{i,j}, t_{i,j+1}]$ of duration τ. Assuming the probability of kill per shot for a pair (e, f) is a function only of target range at any time t, denoted $d(e, f, t)$, we need to compute the formula

$$\Psi(z_{i,j}) = |e|\rho_e \int_0^\tau p_{HK}(e, f, d(e, f, t))dt. \quad (2)$$

Since we subdivided segments at every discontinuity, we do not need a selector function such as $L(e, n)$ from Eq. 1. We instead derive a function $\Psi_{e,f,i}(z), 1 < i < k$, for each of the k continuous intervals of p_{HK}. Alternatively, we could use an estimator function like ψ, but Ψ prevents error accrual when we have to further subdivide a segment (see below) and partition its risk value.

We overload the symbol $\Psi(\bullet)$ to describe the base risk values of risk subregions, risk intervals, and risk sets. Each is simply the sum of the base risk values of its components.

Lanchester equations are known to give the expected value of a stochastic death process with certain properties (Billard 1979). One interpretation of our approach is that we are using spatial Lanchester-like equations to generate expected values for planning. Our equations are one-sided because they do not include risk from the enemy's point of view. It is common to use the two-sided approach to predict when enemy units will be destroyed (for example, see Stanescu, Barriga, and Buro (2015)), but our defenders are all but invulnerable until assaulted.

Often, a single enemy unit will have multiple potential targets during a time interval. In the current version, we deal with this by using the defending unit's full combat power for *all* possible targets' Ψ calculations. Although this is not correct in the sense that a unit cannot apply its full combat power to more than one simultaneous target during the simulation, it is the computationally simplest approach, and does well enough as a first approximation. The result of this choice is that enemy units with many potential targets are heavily weighted for targeting, a reasonable tactic.

Our suppression model is a three-state system. The simulation maintains for each entity a suppression weight value that increases each time it is fired on but not killed. Suppression is also applied to entities near the intended target by an amount that decreases with distance. The weight per shot is stochastic and decreases with target range. The suppression weight of each shot only lasts for a fixed amount of "cool-off" time. When an entity's suppression weight total reaches certain threshold values, it transitions from the unsuppressed state to a partially, and then fully suppressed state, where its rate of fire and probability of hit are penalized (similar to Hughes (1995)). We do not claim this to be a validated psychological model, but it approximates the doctrinal definition of suppression (Department of Defense 2015). The three-state model is analogous to that of CXXI.

[1] We use radius and funnel modifiers; see http://arongranberg.com/astar/docs/modifiers.php

Consider a risk interval $r = (e, f_1, t_a, t_b, s)$. We define a fire support task as $w = (f_2, \vec{c}, e, t_1, t_2, t_3)$, where attacking unit f_2 moves to position $\vec{c}$ during $[t_1, t_2]$ and fires on e during $[t_2, t_3]$. If $[t_a, t_b]$ and $[t_2, t_3]$ overlap, then the effectiveness of e is reduced by suppression during the overlapping interval. In Lanchester terminology, the efficiency coefficient of e should be lowered during this time. As a first step, we subdivide the subregions (and segments) of r at times t_2 and t_3 where they fall within the open interval (t_a, t_b), recomputing base risk values for broken segments. Let s_i be the subregion of r bounded by $[t_2, t_3]$. We use a *residual risk function* $\Phi(s_i, W)$ to compute the remaining risk of s_i after the tasks of W have been applied. For the current system, we use the simplest possible Φ: a constant coefficient β for each affecting task. In other words, $\Phi(s_i, \{w\}) = \beta\Psi(s_i)$. If k tasks in W are relevant to s_i, we have $\Phi(s_i, W) = \beta^k\Psi(s_i)$. As with Ψ, we overload $\Phi(\bullet, W)$ for risk intervals and risk sets as the sum of residual risk of their components.

The score of a fire support task w is the total reduction of risk over all affected risk intervals in the partial plan W, termed $\Delta_w(W)$. However, it is entirely possible for a fire support task to result in a net increase in the total risk of W (a negative Δ score) because it can introduce new risk intervals that might not be outweighed by its benefits.

Our risk values and suppression effects are numerical representations of two different underlying phenomena. A risk value, which represents expected losses during some operational task, is allowed to exceed the number of entities in the unit. So-called *overkill* means that we expect to lose the whole unit before it completes the task. When the risk value is less than the number of entities, we expect some portion of the unit to survive the task. Contrarily, we do not allow suppression to bring the risk value down to 0 or less ($1 > \beta > 0$). If we have an enemy unit threatening a route, then there is always some risk that we will take casualties.

The Fire Support Planner

The fire support planner takes as input a partial plan W_0 consisting of a fixed set of movement orders for some of the friendly units—in other words, the maneuver plan. Included with W_0 is the set A of *availability tasks*, or time intervals during which fire support units may be tasked. The planner's job is to minimize the tactical risk of the partial plan by replacing availability tasks with fire support tasks. Algorithm 1 outlines the procedure; for greater detail see Harder and Darken (2016). P is the set of potential fire support tasks that have not been chosen yet, and R is the risk interval set of developing plan W. Initially, $W = W_0$.

We create a potential task for each fire support resource against each risk interval that it can possibly affect, including (if necessary) movement to a new firing position. We only do the minimum pathfinding for each resource to engage each enemy unit, so the positions found are dependent on the starting position of the availability task. The planner works not forward or backward in time, but best-first, always picking the potential task w with the best score. The second-to-last line of the algorithm lets units provide mutual support by considering risk intervals of other fire supporters.

Algorithm 1 PlanFireSupport

for all $a \in A$ **and** $r \in R$ **do**
  Find a position $\vec{c}$ for a's unit f to engage r's unit e
  Add a potential task $w = (f, \vec{c}, e, t_1, t_2, t_3)$ to P
while $\Phi(R, W) >$ goalScore **and** $A \neq \emptyset$ **do**
  Choose $w \in P$ with the best score $\Delta_w(W \cup \{w\})$
  Add w to W
  Remove from A the task a used to generate w
  Remove from P all tasks generated from a
  $\forall w' \in P$, recompute $\Delta_{w'}(W)$
  Add unused portions a_1, a_2 of a to A
  Generate new task sets P_1, P_2 from a_1, a_2
  Generate new task set P_3 for w's new risk intervals
  $P \leftarrow P \cup P_1 \cup P_2 \cup P_3$

We store mappings from fire support tasks to affected risk intervals and from defending units to risk intervals. This improves efficiency by only considering the risk intervals that each task can affect. We also use the subregion data elements to store and reuse the results of computations.

It would be wasteful to target defending units that have been successfully assaulted, so we assume at the outset that all assaults will succeed and delete risk subregions resulting from defenders after their planned assault time. This assumption is not always valid, but it avoids useless targeting.

Since fire support units may move to new positions and planning is best-first (not forward or backward), it is often the case that a potential task needs to (potentially) replace the route of the unit's subsequent fire support task. For example, if f is planning to move from an availability task a at position $\vec{c}_0$ to another position $\vec{c}_2$, and the planner considers replacing a with potential task w' at $\vec{c}_1$, the original route to $\vec{c}_2$ is now conditionally invalid. To calculate $\Delta_{w'}(W \cup \{w'\})$, the planner must determine the new route, compute its associated risk intervals (with values reduced by $W \cup \{w'\}$), and ignore the risk intervals of the original route $\vec{c}_0 \rightarrow \vec{c}_2$. We store this information with potential task w'.

A further complication would arise from assigning risk intervals to availability tasks. The best approach is to ignore the "risk" of these ephemeral tasks, since they will mostly be replaced with fire support tasks. We would not want to plan a fire support task against an availability task's risk interval, only to have it soon disappear. A similar issue for replaced routes is less impactful because part of the new route is often still supported by the existing tasks.

Experimental Analysis

We evaluate the efficiency and performance of the fire support planner using three terrain maps:

A: Cayucos Creek, CA: 2 km^2, 8192 tris
B: Palo Alto, VA: 4 km^2, 32,768 tris (Fig. 2)
C: Buckeye Peak, CO: 8 km^2, 131,072 tris

Our combat unit types are:

- Observation post: 2 entities with rifles (range 550m)
- Fire team: 4 entities with rifles
- Rifle squad: 3 fire teams and one leader

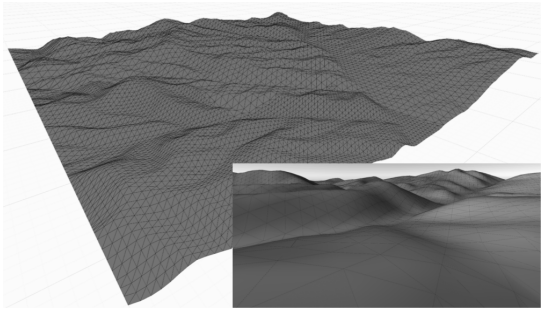

Figure 2: Terrain B. Palo Alto, VA: 4 km^2, 32,768 tris

- Machine gun (MG) squad: 7 entities with 2 machine guns (range 1800m) and 5 rifles
- Infantry platoon (Plt): 3 rifle squads and 2 MG squads
- Infantry company (Co): 3 Plt (rifle squads only) and 3 MG squads
- Infantry battalion (Bn): 3 Co plus 3 more MG squads

Each experimental setup consists of an "Attacker vs Defender" combat unit type pairing and a map (see the Scenario column of Table 1), based on a 3:1 attacker force size advantage. Defenders include one observation post (three in scenario #6). For each setup, we place the defending entities by hand around an arbitrary terrain feature that the attackers will attempt to seize. We assign maneuver tasks to rifle squads by manually specifying attack positions and objectives; our tools do the intermediate path-planning. We choose positions that, together, resemble a higher-echelon form of maneuver such as a frontal or flanking attack. Each unit includes a switch to make it available to the fire support planner; we enable this for all attacking squads. MG squads are available for the duration of the plan, and rifle squads become available after their last maneuver tasks. The fire support planner receives no direct human input; it adds fire support tasks to the plan using its risk intervals, availability tasks, and the annotated mobility graph (inclusive of defender positions). It runs to completion before the simulation begins; we do not perform any replanning for this experiment. We set the minimum suppression time to 60 (simulated) seconds; shorter tasks are ignored. We run each setup 10 times with the fire support planner enabled and 10 times disabled. In the latter mode, MG squads are provided one manually-determined firing position from which they simply engage the closest available target. For the larger scenarios, we run additional replications with limits on the number of iterations through the planning loop (see below). In Table 1, "best" means the number of iterations that had the highest $\Phi(R, W)$ score.

The defenders' locations, protection, rate of fire, and accuracy is tuned such than an unsupported attack usually results in catastrophe. The fire support planner's key to success is suppressing the defenders' fire enough to allow the maneuver units to get in amongst them; at a range of 10m or less, they fall quickly. The planner achieves this quite well for a majority of the objective areas.

Scenario			Mobility Graph Prep	Fire Support Planning			Mission Accomplishment		FER
#	Forces	Map	Avg Time (sec)	Avg Time (sec)	Path-finding	Iterations	Manual	Auto-mated	Δ
1	Plt vs Squad	A	0.85	0.22	44%	7	40.0%	66.7%	+3.90
2	Plt vs Squad	B	1.28	0.53	46%	6	36.7%	76.7%	+0.72
3	Plt vs Squad	C	3.57	0.56	85%	4	30.0%	87.5%	+6.71
4	Co vs Plt	B	2.80	3.89	39%	Best: 21	30.0%	68.9%	+1.29
				5.10	41%	Max: 31		74.4%	+2.01
5	Co vs Plt	C	7.45	10.43	90%	Best: 13	46.0%	77.0%	+2.21
				11.16	87%	Max: 18		75.0%	+2.17
6	Bn vs Co	C	20.12	68.82	54%	Fast: 12	17.8%	36.3%	+0.18
				200.17	65%	Best: 100		79.3%	+2.04
				362.51	53%	Max: 236		83.3%	+2.10
All scenarios ("Best" number of iterations where applicable)							33.4%	76.0%	+5.35

Table 1: Quantitative results

Quantitative Results

We profile the fire support planner for each experimental setup in Unity3D's Editor mode. Each replication is run on a single 2.7 GHz Intel Xeon CPU core with 1866 MHz DDR3 memory. Table 1 shows the graph preparation (scanning plus annotation) and fire support planning time separately since the results of those steps can be reused in different situations. Pathfinding comprises 40-90% of the planning effort; the rest involves generation, application, and updating of potential fire support tasks. The planner completes in less than 0.6 seconds (not counting graph preparation) for the Plt. scenarios and 12 seconds for the largest Co. scenario. The most intense scenario (#6) requires over 6 minutes for the most exhaustive processing, but can achieve positive results with about a minute of planning.

We measure the algorithm's performance against the simple manually-generated plans by mission accomplishment: the percentage of 50-100m radius objectives seized. Each objective is considered seized only if there are no defenders remaining within it at the end of the run. We also show attrition results by the difference in fractional exchange ratio (FER), a traditional operations analysis measure (Helmbold and Kahn 1986). FER$_Y$ is given by X_E/X_F, where X_E (X_F) is the fraction of E's (F's) combat power that was destroyed during configuration Y. Δ FER = FER$_A$ - FER$_M$, where A is a configuration with the planner enabled and M the manual configuration for the same scenario. The planner outperforms the simple, manual plan for both metrics and all planner configurations. All results are statistically significant at $\alpha = 0.05$. A multi-factor ANOVA and parameter estimate for the effects of force pairing, map, and manual vs. automated planning confirms the same for both metrics ($P < 0.0001$).

Clearly, the planner's running time does not scale linearly with input size. Our analysis (not shown here) indicates a polynomial upper bound $O((|F|+|E|)^5|N|^3)$. However, the best-first design of the planner is a nice feature for real-time applications. If we cut off the planner before it exhausts all potential tasks, we still get an executable plan with the most important tasks—as well as Δ scores can determine. We also note that the planner can generate poorly-performing tasks,

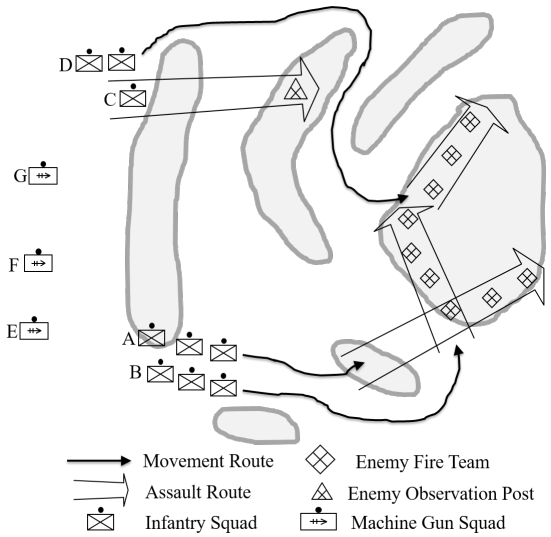

Figure 3: Maneuver plan input to the fire support planner

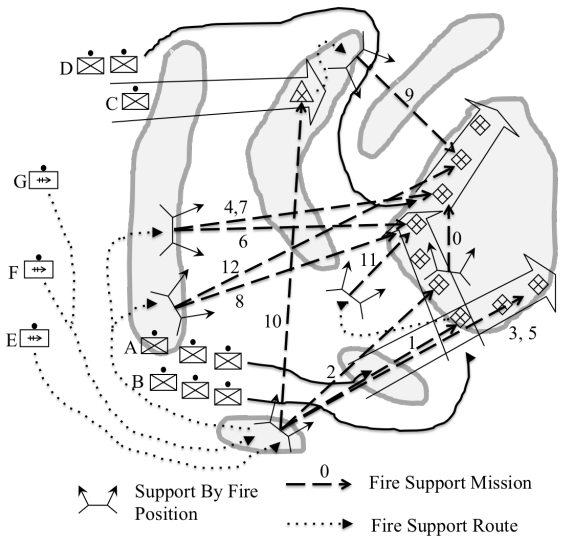

Figure 4: Fire support plan generated by the planner

and if left unchecked will add them when all useful tasks have been exhausted. This may explain the dip in performance for scenario #5 ("Max" iterations). Sometimes tasks that look bad initially can do well in combination, but that tends not to happen in later iterations. The planner could be augmented with a single backtracking point to return the best-scoring plan it produced during processing.

Qualitative Results

We illustrate some of the interesting properties of generated plans with the help of an example. The maneuver plan shown in Fig. 3 was input to the planner, resulting in the fire support plan in Fig. 4 and 5 (we culled some of the actual output's 21 tasks for readability). The numbers shown in Fig. 4 represent the order in which they were added to the plan, not the order of execution.

Suppressing enemy units while friendly units are exposed is the critical fire support objective. By comparing the timing and targets of assaults and fire support tasks of the example, the reader can gain a sense of how this works—as well as how difficult a task it is for a human in all but the simplest scenarios.(The simulation environment includes detailed 3D terrain, which is only crudely represented here.) If we do not have enough assets to suppress everything, which targets are most important and when? The computer can reason about this much faster than a human could, although given enough time we believe a human could find a better plan.

An advantage of a planning-based approach over a reactive one is that we can improve coordination and mutual support. This is particularly important when a fire support unit needs to change positions: it must depart early enough to begin suppression before the threat manifests. Our approach relieves a player or designer of this calculation. It does not, however, make any changes to the maneuver plan.

In a typical human-planned assault, the suppressing unit begins by firing on the defender that most threatens the advance. When friendly forces get close to the initial targets, the suppressing unit shifts to targets ahead of the advance and finally lifts its fire—that is, stops shooting—when the assault reaches the secondary targets. Since the risk intervals for route segments close to the objective have greater risk values (due to the defenders' shorter targeting range to the assaulting unit), the targets just ahead of the assaulting unit tend to be chosen for suppression. Since we did not directly encode shifting and lifting logic, we view this result as a correct emergent property (Ilachinsky (2004) makes an argument for emergence as partial validation). If some large, nearby unit that is not on the objective presents a numerically greater risk, the planner will target that threat first—a correct decision, but one that a human designer might miss. In the example, we can see Unit F shifting from task 8 to 12 and Unit G shifting from task 6 to 7.

Another interesting property emerges when the maneuver plan calls for objectives to be attacked in sequence. In real world plans, it is common for the first assaulting unit to support the second assault once it has secured the first objective. Our planner usually does exactly that. Since we only find the minimum set of firing positions, the first unit will use its own objective as a firing position if possible, such as Unit B's task 0. Unit A's task 11 is similar, but the assigned squad has to move to a new position to gain line of sight.

We are also pleased to note that the observation post does not distract the planner from more important targets at critical times. With only 2 entities, it is only worth suppressing just as C closes to its objective.

Conclusions and Future Work

We described our combat simulation environment and our approach to modeling tactical risk and risk reduction by suppression effects. We then presented an implementation of a fire support planner module, along with some unique considerations and insights that were revealed during the effort.

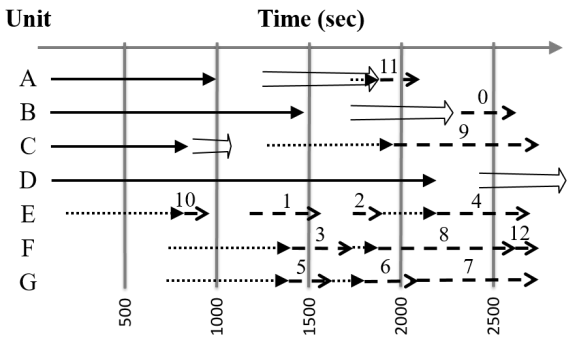

Figure 5: Schedule view of the fire support plan

These include a minimal position-finding approach, computational efficiency techniques, route replacement for best-first planning with its impact on potential task scores, and how to deal with the risk of an availability task. We described our experimental setup, which scaled up in terrain size and unit echelons, and we pointed out some interesting and realistic behavior that emerged in the output plans. The planner prototype was shown to run fast enough to support use as a scenario development tool, and even as a run-time dynamic function for some scenarios. Larger scenarios may be supportable by limiting processing time, which is naturally supported by the best-first planner design.

In future work, we plan to address known issues such as "unsafe" directions of fire, non-ideal firing position choices, and routes that stray beyond the forward line of troops. We hope to deal with some of these issues with tradeoffs between pathfinding depth and size of the potential firing position set. Above and beyond fire support planning, we intend to automate more of the maneuver planning component. We believe that tools such as these can fundamentally change the way that combat simulations are used in support of analysis and training. When scenario designers can rely on automation to generate reasonable tactical plans, they can raise the focus of their efforts by at least one level of abstraction. Additionally, we hope to see an increase in collaboration between the game AI and military simulation community, whose developments are often complementary.

Acknowledgments

Travel for this effort was funded by the Office of Naval Research, Code 30.

References

Balogh, I., and Harless, G. 2003. An overview of the COMBAT XXI simulation model: A model for the analysis of land and amphibious warfare. In *Proceedings of the 71st Military Operations Research Society Symposium*.

Billard, L. 1979. Stochastic lanchester-type combat models i. Technical Report NPS55-79-022, Naval Postgraduate School.

Churchill, D., and Buro, M. 2013. Portfolio greedy search and simulation for large-scale combat in StarCraft. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*.

Department of Defense. 2015. *Joint Publication 1-02: Department of Defense Dictionary of Military and Associated Terms*.

Gu, X.; Chen, J.; Li, J.; and Liu, H. 2012. Genetic vector ordinal optimization algorithm based on RSM for UCAV attack planning. In *2nd International Conference on Computer Science and Network Technology*, 1973–1977.

Harder, B. R., and Darken, C. 2016. Automated fire support planning for combat simulations. In *Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction and Behavior Representation in Modeling and Simulation*. http://sbp-brims.org/2016/proceedings/IN_7.pdf.

Helmbold, R. L., and Kahn, A. A. 1986. *Combat History Analysis Study Effort (CHASE): Progress Report for the Period August 1984-June 1985*. National Technical Information Service.

Hughes, W. P. 1995. Two effects of firepower: Attrition and suppression. *Military Operations Research* 1(3):27–35.

Ilachinsky, A. 2004. *Artificial War: Multiagent-Based Simulation of Combat*. New Jersey: World Scientific Publishing Co. Pte. Ltd.

Lanchester, F. W. 1916. *Aircraft in warfare: The dawn of the fourth arm*. London: Constable and Company Limited.

Miller, D. 2016. Hierarchical task network prototyping in Unity3D. Master's thesis, Naval Postgraduate School.

Rowe, N. C., and Lewis, D. H. 1989. Vehicle path-planning in three dimensions using optics analogs for optimizing visibility and energy cost. In *Proceedings of the NASA Conference on Space Telerobotics*.

Secarea, V. V., and Krikorian, H. F. 1990. Adaptive multiple target attack planning in dynamically changing hostile environments. In *Proceedings of the IEEE 1990 National Aerospace and Electronics Conference*.

Silver, D. 2005. Cooperative pathfinding. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*, 117–122.

Stanescu, M.; Barriga, N.; and Buro, M. 2015. Using Lanchester attrition laws for combat prediction in StarCraft. In *Proceedings of the Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 86–92.

Straatman, R.; van der Sterren, W.; and Beij, A. 2005. Killzone's AI: dynamic procedural combat tactics. In *Proceedings of the 2005 Game Developers Conference*.

Uriarte, A., and Ontaon, S. 2015. Automatic learning of combat models for RTS games. In *Proceedings of the Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 212–218.

van der Sterren, W. 2002. *Tactical Path-Finding with A**. Boston, MA: Course Technology. 294–306.

van der Sterren, W. 2013. *Hierarchical Plan-Space Planning for Multi-unit Combat Maneuvers*. Game AI Pro: Collected Wisdom of Game AI Professionals. CRC Press. chapter 13, 169–183.

Wang, X.; George, S.; Lin, J.; and Liu, J.-C. 2015. Quantifying tactical risk: A framework for statistical classification using MECH. In *Proceedings of the 8th International Conference of Social Computing, Behavioral-Cultural Modeling, and Prediction*, 446–451.