Command and Control Training Centers: Computer Generated Forces Meet Classical Planning *

Carmel Domshlak
Industrial Engineering & Management,
Technion, Israel

Ziv Even-Zur and **Yannai Golany**
Elbit Systems, Ltd.
Israel

Erez Karpas and **Yevgeni Nus**
Industrial Engineering & Management,
Technion, Israel

Abstract

We describe SHOGUN, a fully automated system for controlling tactical agents, developed for integration within simulation-based command and control training centers produced by Elbit Systems Ltd. In particular, we focus on describing the action planning module of SHOGUN: while controlling tactical agents in military-style domains involves dealing with uncertainty and partial information in adversarial settings, the planning module of SHOGUN is based on classical, deterministic planning only, and employs a general-purpose classical planner. We describe our embedding of classical planners within the commercial command and control training center, and report on a recent evaluation of SHOGUN in operational scenarios, confronting subject matter experts as trainees.

Introduction

Comprehensive training of forces responsible to react in complex adversarial situations is critical for military high and low intensity conflicts as well as for homeland security scenarios of border control and smart city environments. Such a training should put together teams of trainees at various levels of command, and train them in realistic setups to improve their command and control (C2) capabilities. A vastly dominating portion of C2 training is delegated these days to software simulation systems in which commands of both role-playing trainees and adversary-playing instructors are accomplished by the respective computer generated forces (CGF). Following the paradigm of "train as you fight", the trainees are connected to the virtual battlefield through their operational C2 systems and combat-net radio, coupled by the overall training system to the simulation.

These days, commercial simulations for C2 training already achieve a sufficiently high level of realism in terms of modeling the physical properties of both the environment and forces. The outcome of the training, of course, depends a lot on the effectiveness of the instructors playing the role of the adversary, and this turns out to be an issue. Putting together a team of skilled and coordinated role-players needed for a large-scale simulated exercise requires months of costly preparations, availability of instructors for a long period of time, and a suitable venue. These limitations of relying on human instructors in simulation-based training suggest at least partly replacing them with artificial adversary-players implementing this or another action planning technology. Here we describe SHOGUN, a fully automated system for controlling tactical agents within a commercial military training simulation. SHOGUN has been developed in a joint effort of Elbit Systems Ltd. and the Technion for subsequent integration within the line of large-scale simulation-based training centers produced by Elbit. This system has been recently deployed to Elbit, and successfully passed a detailed performance evaluation.

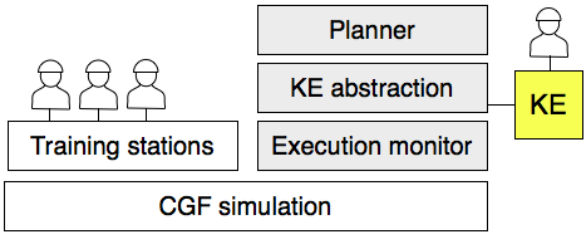
An interesting property of SHOGUN is that the planner it embeds is not just inspired by the artifacts of academic AI research, but actually is such a direct artifact. Moreover, while in general controlling tactical agents in relevant domains involves decision making under uncertainty and partial information in adversarial settings, our experience provides yet more evidence that successful reasoning about real-world systems of active entities does not necessarily have to take explicitly into account all that complexity when choosing between alternative courses of action. While classical planning, capturing single-agent problems with deterministic actions and effectively full knowledge, has been repeatedly criticized for being unrealistic and thus irrelevant to real-world problems, here we demonstrate that this criticism should be taken with lots of caution: The decision making module of SHOGUN is based on classical, PDDL-based planning only, and employs a general-purpose (and thus fully replaceable) classical planner.

In what follows we describe our embedding of classical planners within the commercial C2 training system, as well as the way in which we divide-and-conquer the details of the physical system between the planning and the simulation modules. We then describe the aforementioned evaluation of SHOGUN in operational scenarios, confronting professional military personnel as trainees.

SHOGUN Architecture and Design Decisions

In this section we describe the overall architecture of SHOGUN, focusing on the adopted planning and execution formalism and its support within the system. At high level, SHOGUN comprises a standard architecture of iterative planning, depicted in Figure 1a. It consists of three major mod-

*The work was partly funded by a Magneton Grant. The authors would like to thank Yoav Manor and Gilad Mandel from Elbit Systems for their devoted work on the project.



(a)

input: planning task stub $\Pi = \langle V, A, G \rangle$
local: partial-order plan ρ
$\rho = \langle \cdot \rangle$
forever:
  receive from EM the current state of knowledge σ and plan status $\{\rho_{\text{done}}, \rho_{\text{exe}}, \rho_{\text{next}}\}$
  s = TRANSLATE(σ, Π)
  s' = PROGRESSION(s, ρ_{exe})
  if VERIFY-PLAN(s', ρ_{next}) fails **then**
    ρ = MAKE-PLAN$(\langle V, A, s', G \rangle)$
    send ρ to EM

(b)

Figure 1: High-level (a) structure of the system, and (b) flow of the planner interacting with the KE abstraction mapping.

ules: (i) a *planning module*, (ii) a plan *execution monitor*, and (iii) a real-time high fidelity 3D tactical computer generated forces (CGF) *simulation* in which the actions selected by both trainees and instructors are actually simulated.

Planning, Execution simulation, and Monitoring

The CGF simulation maintains the entire battlefield arena, and supports an arbitrary number of force types (such as tanks, artillery, reconnaissance, etc.), as long as the simulation is provided with their respective physical models. The simulation runs a full 3D virtual environment of the terrain and physical models of sensors (such as line of sight and detection) and actuators (such as ballistics, path planning, and movement). The battlefield comprises two adversarial forces, blue force and red force, each comprising a, possibly heterogeneous, set of acting units. The trainees fully control the blue force troops and interact with the virtual arena via a training station using a high-level language of command. The planning module replacing the instructors fully controls the red force, and communicates with the simulation via effectively the same language of command. The control of the red force is achieved via a planning and execution loop that takes place during the entire training session. The overall loop is described below and the perspective of the planning module on that loop is pseudo-coded in Figure 1b.

- The execution monitor pulls from the CGF simulation all the data σ required to provide the planner with the current state of the red units (their locations, heading, ammunition, etc.), as well as with those parts of the state of the blue units that are considered by the simulation to be observable by the red units. Status of the blue units not detected by any red unit is not provided to the planner. Likewise, the execution monitor pulls from the CGF simulation a status of the currently executed plan ρ of the red force. Since the execution is continuous, at the moment of the query some of the actions of ρ have been already accomplished, some have started and are still executing, and some are yet to be started. Note that "accomplished" can stand here for both "successfully accomplished" and "failed". In any case, both the collected state of knowledge σ and the plan status $\{\rho_{\text{done}}, \rho_{\text{exe}}, \rho_{\text{next}}\}$ are passed to the planning module.
- The CGF simulation is the core of the virtual arena of Elbit's strategic and tactical training centers, designed to communicate with the training stations of human operators. Hence, the information σ about the current state of the (observable) world takes the form of a raw data. This raw data is then translated to a state of the world description s, corresponding to the abstraction of σ in terms of the planning problem operated by the planner. This translation is based on a knowledge engineering layer that is devoted to bridge between the physical view of the simulation and the symbolic view of the planner.
- Given state s and plan status $\{\rho_{\text{done}}, \rho_{\text{exe}}, \rho_{\text{next}}\}$, the planning module estimates whether the current plan ρ of the red force is still valid. In case the goal of reds turns out to be unachievable from s along the still unaccomplished part of ρ, a new plan is generated from the new initial state s, and passed to the execution monitor.

Classical planner: Why and How.

The heart of SHOGUN is its planning system. The first decision we had to make is whether to develop a special-purpose planner, or to adopt a generic, model-oriented planning system. The second, and in a sense, tangential decision we had to make was what details of the problem the planner should take into account and what details it could ignore without sacrificing the quality of the training.

While in principle special-purpose solutions can be more efficient and effective than generic ones, their development requires the enterprise to establish a development team in the respective area of expertise. Along with the fact that the development basically starts "from scratch", that adds numerous risks to the project. Generic planners obviously do not exhibit these risks by the virtue of being generic, having potential to be reused between various verticals. Of course, model-oriented generic planners come with their own risks such as capability of the respective model to capture the desired domain, the computational efficiency of the planner on the domain of interest, etc. However, in contrast to the risks associated with developing a brand new special-purpose system, these risks can be verified in very short time at the beginning of the project using an off-the-shelf planner.

Considering now the choice of the planning formalism, decision making in C2 environments of our interest always involves action non-determinism, partial information, and adversarial settings (Wilkins and Desimone 1992; Tate et al. 2000; Kott et al. 2005). A priori, this sug-

gests that our planning tasks should be specified in terms of much more complicated action models than that of classical planning because the latter assumes deterministic actions, effectively full knowledge, and single-agent setting. Adopting complex planning formalisms, however, comes with a price: the performance of planning for such formalisms currently does not meet the requirements of large-scale C2 training. On the other hand, the performance of classical planners has been dramatically improved over the last two decades, and today these are capable of generating in seconds plans of hundreds of steps in state models of more than 2^{1000} states. In addition, it is of growing understanding that successful reasoning about real-world systems of active entities does not necessarily have to explicitly take into account all the complexity of the reality while choosing between alternative courses of action. This property of many real-world domains has been exploited in the past both in experiments (Yoon, Fern, and Givan 2007; Yoon et al. 2008), as well as in ambitious applications of AI reasoning (Muscettola et al. 1998).

Departing from this matter of business, we have decided to start with a *fully off-the-shelf* satisficing classical planner, adapting it only when really needed and only via external wrappers. Specifically, in the experiments described later on, SHOGUN was using the very popular these days Fast Downward planner (Helmert 2006), using its greedy best first and WA* search engines, and the seminal FF heuristic (Hoffmann and Nebel 2001). The actual planning tasks have been encoded using the PDDL language of the International Planning Competitions (IPC) [1]. PDDL allows representing planning tasks concisely using first-order literals and logical connectives. Fast Downward compiles PDDL input into a ground representation with variables of arbitrary finite range (Helmert 2006). The representation used in Fast Downward is based on the SAS$^+$ action language (Bäckström and Nebel 1995), and extends it with conditional effects and derived predicates. A SAS$^+$ planning task[2] is given by a quadruple $\Pi = \langle V, A, s_0, G\rangle$, where:

- $V = \{v_1, \ldots, v_n\}$ is a set of *state variables*, each associated with a finite domain $dom(v_i)$.
- the *initial state* s_0 is a complete assignment, and the *goal* G is a partial assignment to V.
- $A = \{a_1, \ldots, a_N\}$ is a finite set of *actions*, where each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *preconditions* and *effects*, respectively. Each action $a \in A$ is associated with a non-negative real-valued cost $\mathcal{C}(a)$.

An action a is applicable in a state $s \in dom(V)$ iff $s[v] = \mathsf{pre}(a)[v]$ whenever $\mathsf{pre}(a)[v]$ is specified. Applying a changes the value of v to $\mathsf{eff}(a)[v]$ if $\mathsf{eff}(a)[v]$ is specified. A sequence ρ of actions applicable in the respective states starting from s_0 is a plan for Π if the resulting state s satisfies G.

[1] http://ipc.icaps-conference.org/

[2] For ease of presentation, we present here only the core SAS$^+$ language; for details of the extension we refer the reader to (Helmert 2006).

Formulating planning tasks in SAS$^+$ comes to provide us with a high-level abstraction of the underlying system dynamics: capturing world states at the level of physical simulation would require huge sets of state variables and actions, some state variables are not necessarily observable at any given moment, simulated actions are very much not deterministic, the adversarial blue forces controlled by trainees affect the environment, etc. However, planning at the level of SAS$^+$ has the advantage of fast problem solving, having the potential for compensating for the abstraction coarseness via rapid monitor-and-replan iterations. In what comes next we describe our abstraction mapping of planning tasks from the level of simulation to SAS$^+$.

- *Symbolic abstraction of the physical world.* The function TRANSLATE used by the planning module in Figure 1b to map a physical state σ to a SAS$^+$ state s is implemented via a knowledge engineering sub-module (KE). The latter comes to bridge between the general-purpose planner and the specifics of the simulated domain; as such, it is used twofold. First, KE allows a user to define various layers of information over the map of the training area. These layers describe strategic points, passable areas, ballistically dominating areas, etc., and for most, they can be derived automatically from the digital map used by the simulation. This processing can be performed once per map, and thus completely offline not only to a specific training session, but to the training in general. In addition, the subject matter expert (SME) in charge of the training session can use KE to further enrich this information by specifying, e.g., regions that he prefers not to be used for movements/positions of specific units. Based on the now defined information layers of the map, KE maps status messages received from the execution monitor to proper values of the respective SAS$^+$ variables. The abstraction of the geographic data such as unit locations and headings is archived via a, possibly non-uniform, grid overlaid on the map.
- *Non-determinism of actions.* While basically all actions of the units are simulated to have stochastic effects, the entropy of the underlying probability distributions is usually low, and typically they have single peaks that take most of the probability mass. A natural abstraction of such actions to fully deterministic SAS$^+$ actions simply ignores all but the most likely outcome of each action. SHOGUN uses precisely that simple abstraction, corresponding to a degenerate form of hindsight optimization, an "online anticipatory strategy" for control problems that has previously been successfully applied to problems of online scheduling (Wu, Chong, and Givan 2002) and probabilistic planning (Yoon, Fern, and Givan 2007; Yoon et al. 2008).
- *Partial observability.* Partial observability in the domain of battlefield training stems from the true modeling of reality in which the information that is available to the planner is only what the red force "sees": blue units which are not detected by any red units are not reported to the planner. We use "optimistic sensing" to get rid of this partial observability as follows: when a red unit performs

a sensing action (that is, looks in some direction, trying to find blue units) the expected effects of that action are that no blue forces will be detected. If there are indeed no blue forces - the plan can proceed normally. If there are blue forces there, then the current plan is most likely no longer valid, and therefore re-planning is performed, this time accounting for the "new" blue forces.

- *Optimization objectives.* In most battlefield scenarios, the mission is to achieve some objective, while trying to minimize friendly losses. Since we use single-agent planning, we do not directly account for enemy actions, and specifically, we do not plan for friendly units to be destroyed. Therefore, we do not directly try to minimize friendly losses, but rather try to minimize *risk*. We associate a risk level with each action, by assigning higher costs to riskier actions. For example, maneuvering in a flat area at the base of an enemy-occupied hill is riskier than maneuvering on top of a hill, and is therefore more expensive. Although the planner we use is not an optimal planner, it does try to find a low-cost plan, which directly translates to a low-risk plan.

Domain Formulation

In formulating the domain schema, we made several choices that affect the entire system. First, as stated before, we divide the map into locations, which are arranged on a grid, where each location can hold multiple friendly units, and multiple enemy units. Each grid location is represented by an object in the planning problem, and thus locations are used as parameters for operators and predicates. The translation of world knowledge to a planning state involves mapping units at specific coordinates to the corresponding grid locations.

Second, entities in operational domains often act in line with some standard operating procedures (SOP), and thus, in particular, act in *formations*. In maneuvering, for instance, a formation could be either a single entity moving by itself, 2 entities moving side-by-side, 3 entities moving in a single column, or any other arrangement. Types of formations are defined by the overall set of SOPs, and can be provided by a subject matter expert. We chose to formulate our domain so that all actions are performed by some formations of entities. For example, moving from one location on the grid to a neighboring location is done by using the *Move* action on a formation, which describes the entities to be moved, and their internal arrangement (side-by-side, column, etc.). Two special types of action, *Set-Formation* and *Break-Formation*, allow entities to rearrange themselves in different (possibly larger or smaller) formations. Note that this is similar in spirit to the well-known Logistics planning benchmark from IPC-1998 and IPC-2000, where a formation can be thought of as a truck, and an entity can be thought of as a package loaded into the truck (aka joining formation). This engineering methodology appears to be quite useful in general; for instance a very similar technique has been used by Balla and Fern (2009) in their recent work on action selection for tactical assault, evaluated by the authors on Wargus computer games.

Third, we had to deal with enemy units on the battlefield, and their partial observability. We model enemy presence by creating a state variable for the number of enemy units at each grid location. The possible values for this range are either a number (between 0 and some bound) or *unknown* - a special value indicating that we have no knowledge about enemy presence in that grid location. Thus, the (expected) effect of performing a sensing action on a given grid location is that if the number of enemy units in that location was *unknown*, it becomes 0, and otherwise, there is no effect. This formulation also allows us to ignore the identities of enemy units, which are not part of the knowledge provided to the planner anyway.

Load Balancing and Parallelization

One addition to the standard classical planning setting that we found essential was load balancing between the red units. At high level, the load balancer in SHOGUN pre-assigns each sub-goal to a subset of units, decomposes the overall planning task into several smaller tasks that are planned for independently, and then combines their solutions into a plan for the overall task. This procedure is important for balancing the workload between different role-players, causing forces to act in a more "coordinated matter", and reduces the size of the individual planning tasks solved by the planner.

Similarly to all other system components, the load balancer in SHOGUN is completely domain independent. It starts by assigning to each goal a subset of units that can achieve it as cheaply as possible in the (easy to solve) delete-relaxed version of the planning task. Then, the rest of the units are assigned in proportion to the cost of the relaxed plan for each of the goals, so that goals that are riskier to achieve are assigned more units. Finally, the goals are grouped based on the transitive closure of the forces assigned to them, and each such group of goals is planned for using only the forces assigned to it. In the domain considered here, plan combination is trivial, since no two plans can interfere with each other.

One thing to note is that, although the load balancer might assign several units to a single goal, the planner might still not utilize all of these units (the plan found could involve just a single unit). In order to force SHOGUN to act more realistically, we artificially increase the cost of action repetitions. This puts a heavy bias toward using more than a single unit in each plan, resulting in plans allocating forces to goals in ad hoc proportion to the size of the goal.

Finally, though the quality metric for our plans is risk reduction, and thus we employ a cost-oriented, sequential planning, the actions of different units often can (and if so, should) be applied concurrently. While we do not plan for this second objective directly, we do convert our sequential plans into partial order plans allowing concurrent action execution. If our domain had been formulated in plain STRIPS, then simple Partial Order Causal Link (POCL) backward analysis of the plan would have given us a desired partial order. However, our domain formulation uses conditional effects, and this requires slight extension of the standard POCL analysis. To establish hypothetical causal relations between the actions, we first simulate sequential

execution of the initial plan, determine which conditional effects of which action instances along the plan have been fired, compile the fired conditional effects into now unconditional effects of the respective actions, and then perform the standard POCL backward analysis of the plan. The resulting parallelization is sound and complete, and results in realistic schedule of plans for our multi-unit forces. Overall, the simultaneous acting effect, achieved in SHOGUN via the mixture of load balancing and plan parallelization, achieves the effect of a standard military C2 methodology called "mission-oriented C2", allowing for solving large-scale problems by wisely delegating its sub-parts to different planners.

Usage Practice and Performance Evaluation

One of the critical requirements to "AI driven" systems facing users is that the amount of exposition of the users to the technical details of the system will be as little as possible. In that sense, SHOGUN seems to achieve an extremely high level of transparency. The overall usage of SHOGUN comprises three steps: (i) domain, or action schema, modeling, (ii) training scenario specification, and (iii) the actual training session. The action schema is modeled offline, once for each type of units such as *tank/modelX*, *border-patrol/modelY*, etc. This step does require a subject matter expert to be familiar with the SAS$^+$ action description language, either directly or via a dedicated GUI. However, since it is performed once for all subsequent trainings, it is fully realistic to provide this skill to a small group of SMEs.

Next, possibly at a distant point in time, the properties of a specific training session are specified by an instructor in charge of the training content, e.g., a battalion intelligence officer. This step consists of annotating the map with information relevant to the training (such as markup of areas passable by different types of vehicles, dominating areas, etc.), defining the initial positions of the forces, defining the goal of the training scenario, etc. All these specifications are made through our knowledge engineering tool (already discussed in the previous sections), and requires no knowledge of planning technology whatsoever. Finally, the trainees also need to know neither what SHOGUN is about, nor even the fact that they are trained not by human instructors, but by a fully automated system.

In what follows, we describe the evaluation of SHOGUN that has recently been accomplished by Elbit Systems Ltd. In each training session, an SME controlled the entire blue force, SHOGUN controlled the entire red force, and both operated in a realistic modeling of the situation awareness fog of war. The context of all training sessions was the same 4×4 kilometers area featuring jagged, hilly, terrain, and having several spots dominating large areas and thus having high tactical value. Several regions of the map were marked at the stage of training scenario specification as non-passable, disallowing movement through these regions. Figure 2 depicts a starting position of the forces in CGF simulation. The blue force, controlled by the trainee, comprised an extended armored company of 13 tanks, located initially at a defensive position in the northern part of the map, which

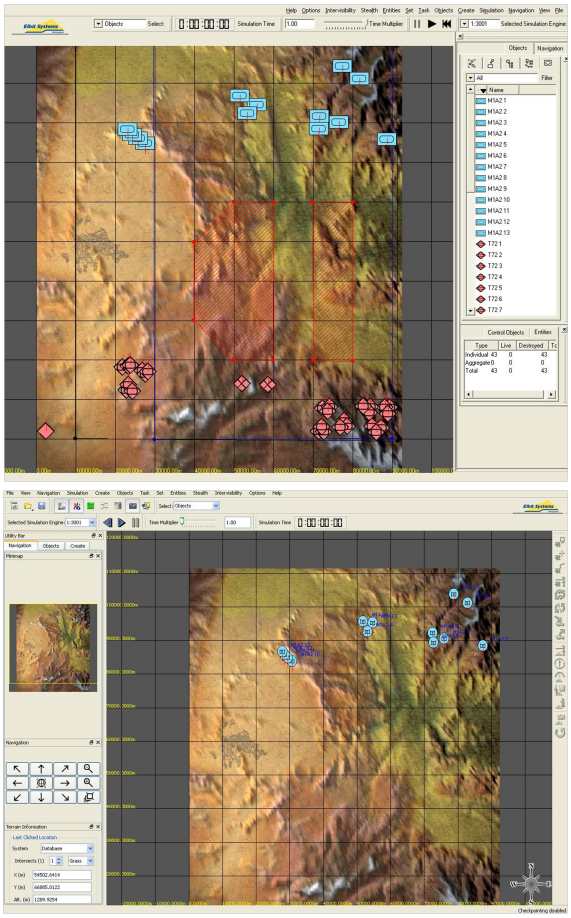


Figure 2: Full (top) and role-player's (bottom) views of a starting position in the Elbit's CGF simulation. Blue/red forces are located in the north/south, respectively. Non-passable areas are highlighted in red.

is the high ground in the training area. The red force, controlled by SHOGUN, comprised an armored battalion of 27 tanks in 9 platoons, supported by 2 reconnaissance units and limited artillery. The mission of the trainee was to defend the blue force's starting region, while SHOGUN's mission was to neutralize all blue units in a predefined region of the map, either by destroying them or causing them to retreat. This magnitude of forces is typical for a mission to neutralize an area of this size, and according to the military best practices, the defending blue force a priori has an advantage in this scenario.

The evaluation consisted of ten training sessions with varying starting positions of the red force, and were done (as detailed in Table 1) by three SMEs: two company commanders in reserve duty (trainees 1 and 2), and one battalion XO in reserve duty (trainee 3). As Table 1 shows, SHOGUN won in 6 out of 10 training sessions, winning all the first training sessions by the individual SMEs, and sometimes losing later on to the same SME, after action review and elaboration of lessons learnt for employing defense battle techniques. From the loss/win pattern followed by the evaluation of the SMEs, SHOGUN provides a realistic adversary

1/R
1. Blue organizes for defense.
2. Red sends first attack with a single company and suffers heavy losses (2 platoons are destroyed, 1 falls back).
3. Red sends full scale attack in 3 different areas, and breaches the blue defense in the east.
4. Red destroys the rest of the blue entities in the area.

1/R
1. Red sends 2 platoons to flank from the east, and one platoon to the attack zone in the west.
2. The red platoon destroys the 4 blue tanks on the hill controling the west zone.
3. The red force takes advantage of the situation and sends a massive attack through the opened west zone (in parallel to units which move through the east zone)
4. Blue tries to organize his defense in order to cover the the west zone, but the red attack progresses too fast and all reinforcement blue forces are destroyed.
5. The remaining blue forces try to avoid further contact but eventually are caught and destroyed by the red force.

1/B
1. Red force sends a preemptive attack. Blue force puts two lines of defense.
2. Red force does not concentrate the attack, and blue force takes advantage of that by destroys the red platoons one by one.
3. The balance of power shifts towards blue force, and the attack fails.

1/B
1. Blue organize in defensive positions which have absolute control on anticipated red attack: moves west passage force to higher ground, and 2 tanks to high control road.
2. Red send 3 platoons to the center and east passages, and 2 platoons to the west passage.
3. Long range sight of the red platoons enables blue forces a safe defense. Red suffer heavy losses in the initial attack, and the remaining force is outmatched in position and fire power.

1/R
1. Red organizes for attack, sends 2 companies with a platoon
2. Red artillery destroys a blue tank controlling the center, yet no reinforcement is sent there by the blue force.
3. Red takes advantage of this: sends a company to the center zone and 2 platoons to the east.
4. Blue is able to hold center attack with eastern defense, shifting its attention on the center.
5. 2 red platoons in the east reach the blue tank and destroy it. This enables the center force to breach in the center area into the blue zone.
6. The blue force defense is breached, eventually loses all troops.

1/B
1. Blue organizes in defense.
2. Red force sends 4 platoons: 2 in the east, 1 in the center, 1 in the west.
3. Blue force moves to higher positions and destroys 2 out of the 4 platoons.
4. 1 of the 2 red platoons left is able to destroy the blue defense in the east zone.
5. Blue tries to defend in the east, but suffers more losses during the defense reinforcement.
6. 1 red tank breaches the blue area in the west, stops and waits for reinforcement.
7. Red organizes for an additional massive attack in the east? but this gives the blue force time to establish there a solid defense.
8. The new positions of the blue force create 2 zones of its full dominance in the west and center, which the red force falls into.

2/R
1. Red sends 4 platoons to the east and center passage.
2. Blue relocates the 4 tanks controlling the west zone.
3. The 4 blue tanks cause damage to the red progress in the east, but are spotted and destroyed.
4. Red artillery weakens the blue defense in the center zone, while the red force destroys the blue defense in the east passage.
5. Blue force tries to reorganize the defense.
6. Red sends a full scale attack.
7. Blue force sends 3 tanks through the eastern passage to counter attack the red forces, but they are neutralized with only minor losses to reds.
8. The single remaining blue tank is surrounded.

2/R
1. Red sends 4 platoons: 3 from the east and 1 through the center passage.
2. Blue relocates the 4 tanks controlling the west zone in a safe passage, trying to occupy safer positions ("learning from experience").
3. The red platoon at the center passage is destroyed by the blue defense.
4. Red starts moving platoons in the west zone.
5. Red company at the east passage destroys the blue defense there, and falls back.
6. One of the red platoons at the west zone encounters the 4 blue tanks, and destroys them with only minor losses to itself. The west area is now totally opened.
7. Red platoon advances in the eastern passage and destroys blue second line of defense.
8. Red starts a full scale attack with artillery support on all 3 passages.
9. Blue destroys a platoon in the east, but the blue area is totally overrun by the red force on all 3 fronts, and eventually the blue force is all destroyed.

2/B
1. Blue organizes differently for defense ("experience from last two loses"): west area tanks move around north hill in a safe route and occupy a high position which controls a huge portion of the terrain. A platoon moves towards a control route in the north part of the terrain.
2. Red sends 2 platoons to the east passage and 1 platoon to the center passage.
3. Blue high position destroys red forces moving in east and center passage in mid way.
4. Red moves to full scale attack including a company in the west.
5. Blue dominant positions disallow the red forces to progress into the blue territory.
6. More than half of the red battalion is destroyed with no losses to the blue force.
7. The entire red company in the west area is destroyed.
8. Red starts artillery fire, and dstroyes the first blue tank.
9. The remaining 5 red tanks fall back and wander around the southeast part of the terrain.
10. Blue moves into offense and starts to progress towards the red forces.
11. Red force artillery destroys one more blue tank.
12. The blue force intercepts, destroys the remaining red tanks and wins.

3/R
1. Blue sends 2 tanks for patrol over high area road which controls most of the terrain.
2. Red sends 3 platoons: 1 in the center, 2 in the east.
3. Red center platoon is destroyed by the 2 tanks patrolling the higher route, but east platoons destroy blue force defense.
4. 2 blue tanks in the high route destroy another red platoon.
5. Red sends a full scale attack accompanied with artillery (2 platoons in the center and 3 platoons in the east).
6. Blue centers his forces in the north west corner of the terrain.
7. The blue force succeeds to hold off 3 red platoons from the east zone while suffering minor losses.
8. Red breaches the blue area in the center zone and destroys all blue troops not located in the northwest corner.
9. Red organizes his troops for final assault on the remaining blue forces and eventually destroys the last blue tank in the area.

Table 1: Summary of system evaluation training sessions. Each table entry marked with X/Y corresponds to a training session by trainee X, ended with the win of Y force. Per trainee, the sessions are listed chronologically. Double line separates indicates a change of either the initial conditions or the trainee.

in a military high intensity conflict. The gradual improvement of the trainees is inline with the objectives of training, and this improvement was indicated not only by the binary outcome of the training sessions, but also by the quality of actions the trainees learned to select from session to session.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Comp. Intell.* 11(4):625–655.

Balla, R., and Fern, A. 2009. UCT for tactical assault planning in real-time strategy games. In *IJCAI*.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Kott, A.; Budd, R.; Ground, L.; Rebbapragada, L.; and Langston, J. 2005. Building a tool for battle planning: Challenges, tradeoffs, and experimental findings. *Applied Intelligence* 23(3):165–189.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. C. 1998. Remote Agent: To boldly go where no AI system has gone before. *AIJ* 103(1-2):5–47.

Tate, A.; Levine, J.; Jarvis, P.; and Dalton, J. 2000. Using AI planning technology for army small unit operations. In *AIPS*.

Wilkins, D., and Desimone, R. V. 1992. Applying an AI planner to military operations planning. In *Intelligent Scheduling*, 685–709. Morgan Kaufmann.

Wu, G.; Chong, E. K. P.; and Givan, R. 2002. Burst-level congestion control using hindsight optimization. *IEEE Transactions on Automatic Control* 47(6):979–991.

Yoon, S. W.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic planning via determinization in hindsight. In *AAAI*, 1010–1016.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *ICAPS*, 352–359.