Probabilistic Dynamic Maps to Visualize the Dynamics of Monte Carlo Simulations

Paul R. Cohen[1], John Warwick[1], Gary W. King[1], Clayton Morrison[1],
Marco Ramoni[2] and Paola Sebastiani[3], and

1. Department of Computer Science
University of Massachusetts, Amherst MA 01003

2. Knowledge Media Institute
Open University, Milton Keynes, UK

3. Department of Statistics
Imperial College, London, UK

Abstract

Probabilistic dynamic maps visualize the dynamics of state variables in Monte Carlo simulations. Each trial in a batch of simulations produces a time series of state variables, and it can be difficult to make sense of so much data. Maps summarize dynamics, and probabilistic maps relate dynamics to the probabilities of outcomes of simulations. The technique is illustrated with case studies from a simulator of war games.

Introduction

In the final moments of a game, the coach calls a time-out and scribbles some circles and arrows, diagramming a play.   The circles represent players, the arrows, movements. Similar representations are used by military tacticians, as shown in Figure 1. The icons represent units and the tactical engagements between them. Other icons represent terrain, such as mountains. Lines represent natural boundaries, roads, and areas of responsibility. Phase lines (denoted PL) are places that units meet or otherwise adjust the timing of their operations. Although this kind of diagram represents the commander's intent – what he or she expects to happen – it clearly does not represent what actually happens when units move and engage.

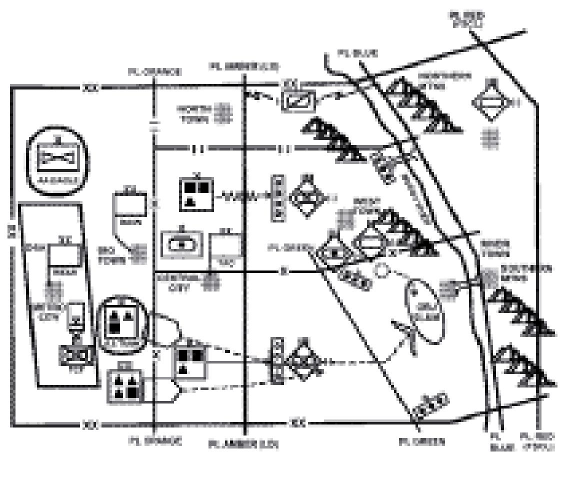

Figure 1. A diagram of a course of action, taken from Army Field Manual FM 101-5.

To grasp the dynamics of courses of action such as the one in Figure 1, the commander's staff engages in war games. These laborious, manual exercises typically explore only a tiny fraction of the space of possible tactical situations. This is why we developed the Capture the Flag system. Like a "chess master in a box" that you can buy at any games store, Capture the Flag includes a powerful algorithm for playing war games, as well as a simulation of the dynamics of military engagements. Humans can play against Capture the Flag, or the system can play itself, or it can follow – more or less strictly – plans supplied by humans. In this way, military courses of action can be explored in Monte Carlo simulation against an intelligent adversary [1,2,3].

The problem is to visualize the results of hundreds of Monte Carlo trials, each of which is a lengthy time series of a vector of state variables, so as to inform the commander of the strengths and weaknesses of a course of action. Following an introduction to the Capture the Flag system, we will describe a visualization of the dynamics of simulated courses of action. This visualization, called a *probabilistic dynamic map,* relates dynamical properties of a war game to the probability of winning the game. Sometimes, a game produces clusters of outcomes that are so qualitatively different that they should be visualized separately.   We describe an algorithm for finding these clusters automatically.

Capture the Flag

The Capture the Flag system is a simulator of war games and an automated planner that commands units on one or both sides of a conflict. Figure 2 is a screen from one scenario, modeled after a training exercise at the Command and General Staff College at Fort Leavenworth. Red units are heading toward a Blue flag in the southwest corner of the map; meanwhile, Blue forces have broken through a Red line in the northern part of the map and are about to give chase.

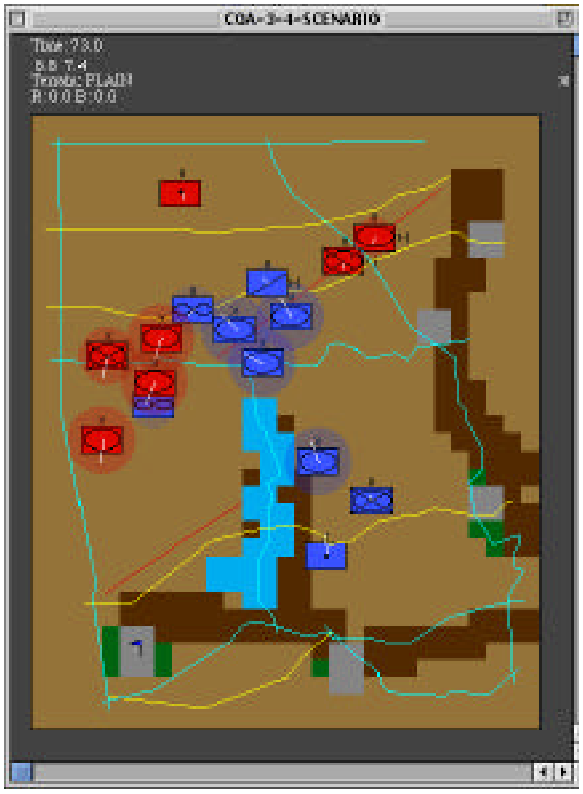

Figure 2. A screen from Capture the Flag, 73 time steps into a simulated course of action. Icons represent mechanized, infantry, artillery, and air cavalry units. Colored areas represent terrain: Light brown is open terrain; dark brown is hilly terrain; blue is a lake; green, forest; and gray, towns. Lines represent roads and boundaries of areas of responsibility.

In this scenario, both Red and Blue forces are controlled by the planner, that is, the machine is playing against itself. This is the easiest way to generate Monte Carlo evaluations of courses of action: One creates a version of the planner for each side, Red and Blue; specifying a course of action for each; and the planners attempt to achieve the specified goals by implementing the specified tactics. Because the game is a dynamical system, small differences in the timing of actions and initial placements of units produce highly variable outcomes.

Dynamic Maps

We developed dynamic maps to visualize the variable dynamics of Monte Carlo trials in Capture the Flag, as well as the behavior of mobile robots and the semantics of verbs [5]. *State variables* are measures of situations such as locations and velocities of units, force ratios, attrition rates, and so on. A *trajectory* is a time series of the vector of one or more state variables. Each simulation of a course of action generates a trajectory. A *map* is an overlay of dozens or hundreds of trajectories. A map for a course of action is an overlay of trajectories for that course of action. Said differently, a map is a compilation of many experiences executing a course of action. For example, Figure 3a shows ten engagements between a Blue unit that is trying to capture a Red flag and a Red unit that is trying to defend its flag. Ten trajectories of a single state variable – distance between the attacking Blue unit and the Flag – are shown. The colors denote who won the engagement; Red successfully defended its flag six times. Games tend to cluster into a handful of qualitatively different engagements. Figure 3a shows two distinct ways that Red can win (corresponding to blocking Blue – the squiggly lines – and directly engaging and attritting Blue, the straight red lines), and one way in which Blue can win (corresponding to a direct engagement that attrits Red).

Probabilistic Maps

If one colors trajectories by outcomes and lays a grid over the map, as in Figure 3a, then the proportion of trajectories of a given color in a given grid cell is an estimate of the probability of the corresponding outcome given that the trajectory enters that cell. More formally, let *x* and *t* be a state variable and time, respectively, and let the coordinate x,t define a cell of the map, of some fixed area, centered on the point x,t. Then the probability of an outcome, such as a win, is estimated as

$$\Pr(\text{win} \mid x,t) = \frac{\text{number of winning trajectories through } x,t}{\text{total number of trajectories through } x,t} \;.$$

We can generate a three-dimensional map from coordinates *x*, *t*, Pr(win|*x*,*t*), as shown in Figure 3b. Every cell on the surface is a probability of a Red win for a trajectory that goes through the cell defined by *x*,*t*.

The landscape plot in Figure 3b shows a probabilistic map constructed in this way for 1000 trials in which a Blue unit attacks a Red flag. The blue spot on the surface represents a state of a game; its vertical altitude represents the probability that Red will win in this state. The arrow leaving the blue spot is the most common trajectory leaving that state. You can see that Red is most likely to lose.

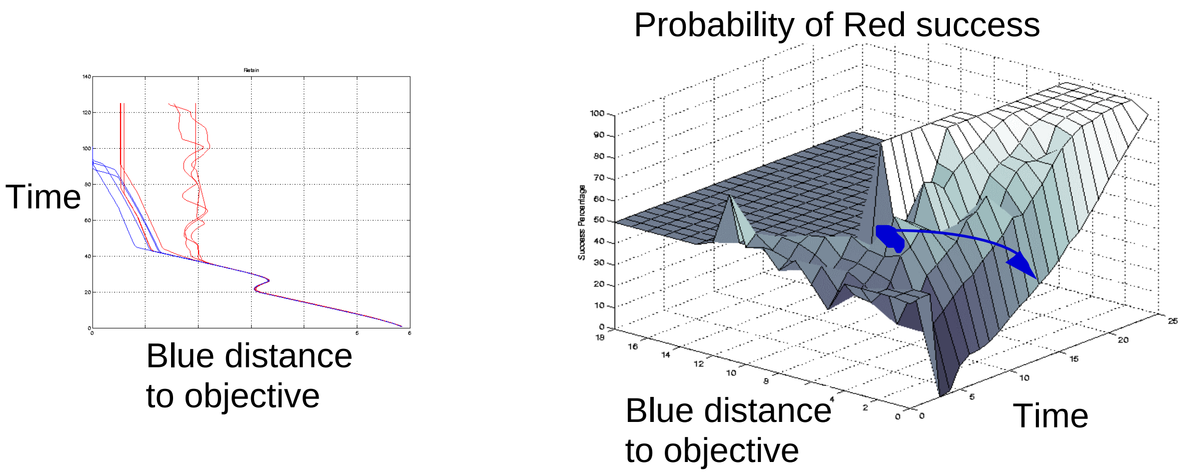

Figure 3 On the left (3a) is a map of ten trajectories, time series of distance to the objective. On the right (3b) is a map of 1000 trajectories, showing the same state variable plotted against the probability of a successful outcome.

The probabilistic map in Figure 3 is "craggy," not flat or smoothly rising or falling. Sharp slopes on dynamic maps correspond to what commanders call *decisive points*. These are points or regions where the probability of success changes relatively quickly. The idea of nonlinearity, particularly nonlinear dynamics, is slowly permeating tactical planning centers in the U.S. military, and can be seen as a natural successor to Clausewitz, who revolutionized tactics by viewing warfare as a branch of physics [4]. The surface in Figure 3b appeals to commanders because it provides them an abstract view of the dynamics of the battlefield, one in which dangerous areas for both sides are explicit. Commanders recognize that if they have maps like Figure 3b and their opponents do not, then they have an information advantage that they might exploit by "steering" their opponents into advantageous areas of the map.

As noted, the maps in Figure 3 represent the dynamics of engagements between a single Red and a single Blue unit. Will the same approach work – i.e., produce legible, informative surfaces relating state variables to probabilities of success – for more complex scenarios, such as the one in Figure 2? We have found no analytical answer to this question. In fact, the same approach works well for complex scenarios, as we describe later, yet fails for some simpler ones. For instance, when we changed the scenario for Figure 3 a little, to have two Blue units attack a Red flag defended by two

Red units, the resulting map was unreadable. Figure 4 shows a “thresholded” version of the map: As before, the axes represent time and the state variable (although in this case the state variable is the relative strength of the two units) and the cell colors represent probabilities. Red cells represent probabilities of Red wins greater than a threshold value, and blue cells correspond to high probabilities of Blue wins. There is no obvious continuity or pattern relating these probabilities to time and the state variable, as there was in Figure 3.

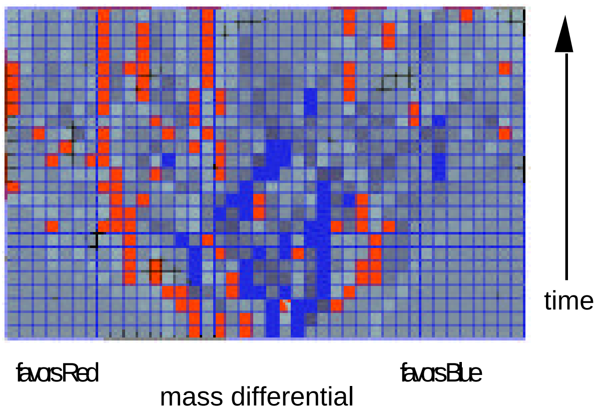

Figure 4..A probabilistic dynamic map for a two-on-two scenario in which two Blue units try to capture a Red flag defended by two Red units. The state variable (horizontal axis) is the relative strength (or mass) of the units. The vertical axis represents time. The map has been thresholded so all cells with a high probability of win for Blue are colored blue, and similarly for Red. Tactically, the attacking units might concentrate on the flag, or the defending units, or split their efforts between these goals. This tactical variability is one reason that the map is not very informative.

The reason is that the two-on-two situation can unfold in several qualitatively different ways, and when the trajectories for these different dynamics are superimposed in a single map, the result is pretty meaningless. It is as if we tried to visualize the dynamics of several different pieces of music, or normal and irregular heartbeats, or placid and volatile markets, in one map. In any of these cases, we would first group trajectories that have similar dynamics, then build a separate map for each group.

Clustering by Dynamics

We have developed a method for unsupervised clustering of univariate, categorical time series called Bayesian Clustering by Dynamics (BCD) [9,10]. (See [6,7,8] for methods for clustering continuous and multivariate time series.) Suppose we represent a military operation at time ***t*** with a state variable that takes one of k values. Between time *t* and time *t+1*, the state of the operation changes or remains the same, but in either case can be represented by a *transition* from a state $s_t \quad \{1 \ldots k\}$ to the next state $s_{t+1} \quad \{1 \ldots k\}$. Clearly, $k \times k$ state transitions are possible, and the *frequencies* with which transitions i,j occur in a military operation of duration N time steps can be stored in a *transition table* like this one:

	a	b	c	d	e
a	17	10	0	16	1
b	0	14	0	3	2
c	0	0	3	8	1
d	0	0	5	12	0
e	6	0	0	0	23

By convention, the vertical axis represents the state of an operation at time *t*, the horizontal axis, the state at *t+1*. Thus, the operation exhibited 10 transitions from state **a** to state **b**, and 16 transitions from **a** to **d**. The sum of the frequencies in a transition table is N–1 for an operation of duration N.

Suppose we run a Monte Carlo analysis of 100 variations on a course of action. We will have 100 tables, like the one above, each representing a single "playing out" of the plan in Capture the Flag. To cluster these variations, to find the ten or twenty qualitatively different ways the plan plays out, we need a way to judge the similarity of transition tables. The Kulback-Leibler (KL) distance is such a method. We assess the KL distance for every pair of plan instances and cluster together those that are similar. Clustering is agglomerative, which means we begin with each plan instance in its own cluster and repeatedly merge the most similar clusters; or failing that the second most similar, and so on; stopping only when no further merging would improve a quality metric. The metric in this case is the *marginal likelihood* of the data given the clustering so far. The BCD algorithm essentially does hill-climbing in a space of possible clusterings of military operations, where the height of the hill associated with any given clustering is the marginal likelihood.

BCD has been tested with data from Capture the Flag and with time series of robot data, financial data, and Bach fugues [9,10]. In particular, we used BCD to cluster data from 81 trials of the two-on-two scenario that produced the unsatisfactory map in Figure 4. BCD produced eight clusters. One of them contained 94% of the plan instances in which Blue won. Three contained plan instances in which Red always won, but interestingly, the *manner* of Red's victory was very different in the three clusters; for example, in one cluster, Red won by attrition, in another it made a mad dash to grab the flag. Four clusters contained cases in which Red won (in different ways) or time expired.

Once the Monte Carlo trials are clustered, one simply builds a map, or shows the trajectories, for each cluster. Figure 5 shows the trajectories that constitute each of two clusters, one in which Blue always won, one in which both sides won some engagements. The dynamics of the clusters are quite different; one can see why superimposing trajectories from these (and the other six clusters) produces an illegible map (Fig. 4).

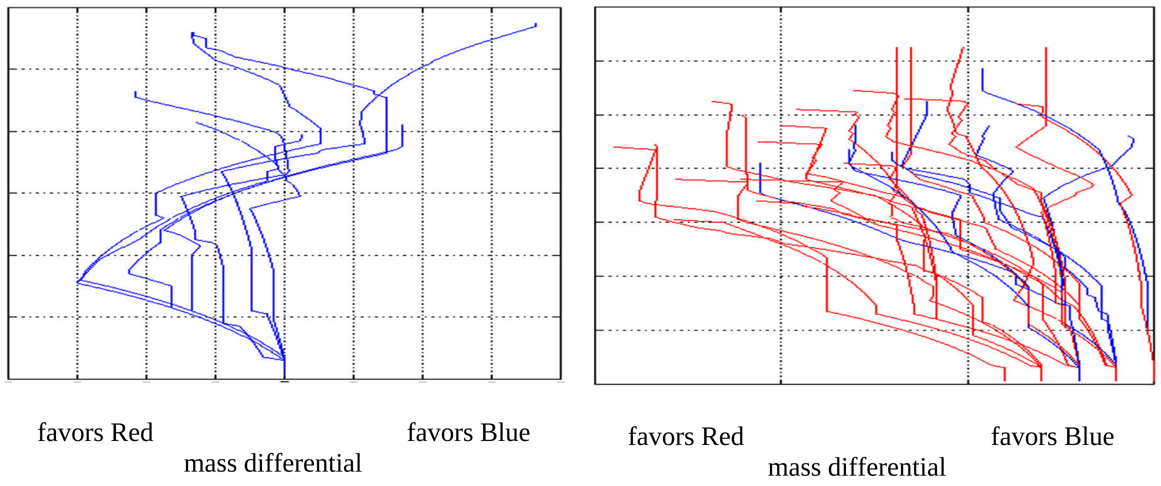

Figure 5. Maps for two clusters produced by the BCD algorithm. The state variable, mass differential (horizontal axis), behaves differently over time (vertical axis) in the clusters. On the left, the trial begins with Blue losing mass through attrition relative to Red but in most cases these losses "peak" early and Blue begins to regain ground relative to Red. In three cases, the differential later swings back toward Red, but Blue wins the games anyway. On the right, in nearly all cases, Blue loses mass monotonically throughout the game, and in most cases loses the game.

Maps for Complex Scenarios

The complex scenario in Figure 2 involves nine Red and nine Blue units with three different kinds of dynamics: mobile units, air cavalry, and artillery; and courses of action that involve at least two tactical phases. To build an informative map for this scenario, we need informative state variables. Because probabilistic maps devote only one of three dimensions to a state variable, it is necessary to design state variables that effectively summarize the entire battle. One, Mass-Near-the-Action (MNTA), is a measure of how much effective mass is concentrated near the center-of-mass (COM) of the opponent. The effective mass of a side is the sum of the effective mass of each of its units, and the effective mass of a unit is its average strength against each of the remaining enemy units. Strength is a unit's mass – roughly, how big it is – multiplied by unit-type effects (for example, artillery has a multiplier of 1.333 against infantry). Thus effective mass combines the masses of the units and the types of the units and the types of the enemy units. MNTA, in turn, is implemented as follows: For one side, say Red, look at each Red unit that is artillery or is close to Blue's center of mass, and measure the effective mass of each unit against all the Blue blobs that it is close to. (Artillery is assumed to be always “close” to the action because it can operate over long ranges.) Roughly, MNTA measures the degree to which units are engaged with opposing units.

Figure 6 shows the trajectories of MNTA for 100 Monte Carlo trials corresponding to the scenario in Figure 2. Trajectories are colored red if Red forces captured the flag in the south-west corner, blue if the Red forces were disabled by Blue, and green if time expired. MNTA is shown on the vertical axis, time on the horizontal axis. Clearly, Blue won most of the trials. In general, trials begin with Blue MNTA increasing to a maximum value at around time 100, then dropping off as Red and Blue units are destroyed or disabled.

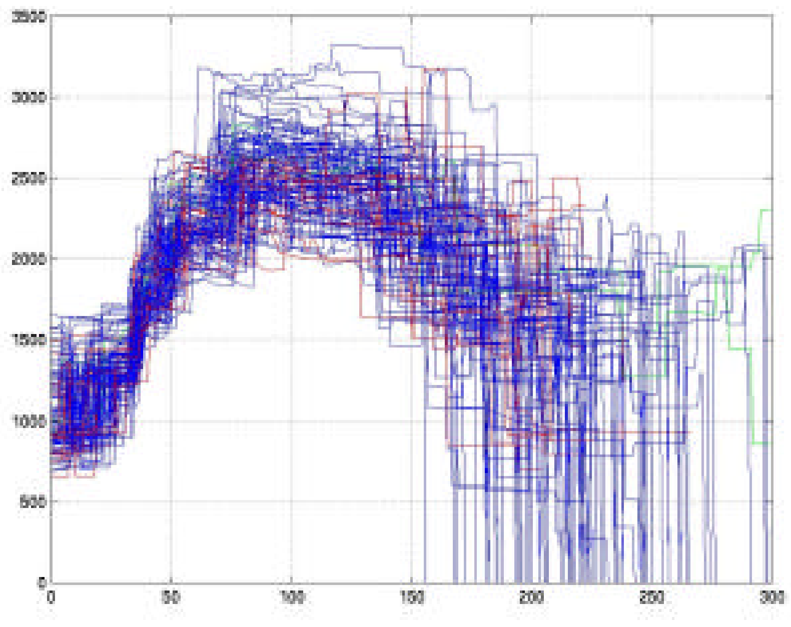

Figure 6. The trajectories of 100 trials of the scenario in Figure 2. Blue lines denote Blue victory, red lines, Red victory, and green lines indicate that time expired before either side won. The horizontal axis is time, the vertical axis is MNTA.

Two probabilistic maps of these trials are very informative. Recall that a probabilistic map for an outcome, such as Blue winning, has three coordinates: a state variable, time, and a probability of the outcome given that a trajectory passes through the cell defined by coordinate *x*,*t*. An alternative is to calculate and plot the probability of winning if a trajectory *does not* go through a cell:

$$\Pr(\text{win} \mid \text{not } x,t) = \frac{\text{number of winning trajectories that don't go through } x,t}{\text{number of winning trajectories}}$$

Both kinds of map are shown in Figure 7, in which higher probabilities are represented by brighter cells (black cells in Fig. 7a and gray cells in 7b are those which contained no trajectories). The axes of these figures are normalized so that time runs from 0 to 100 (horizontal axis) and so does MNTA (vertical axis). Figure 7a is for the probabilities Pr(win | *x*,*t*) and Figure 7b is for Pr(win | not *x*,*t*). The former tells much the same story as Figure 7, but the latter shows something surprising: A black cell at time = 15, MNTA = 47.5, corresponding to a zero probability of a Blue win if Blue's trajectory does not transit that cell. To a military commander, Figure 7b gives a remarkable imperative: Go through this cell, or lose! By avoiding the lighter cells in Figure 7b, the commander can trace out the safer trajectories, because lighter cells correspond to higher probabilities of winning if one's trajectory *doesn't* transit the cell.

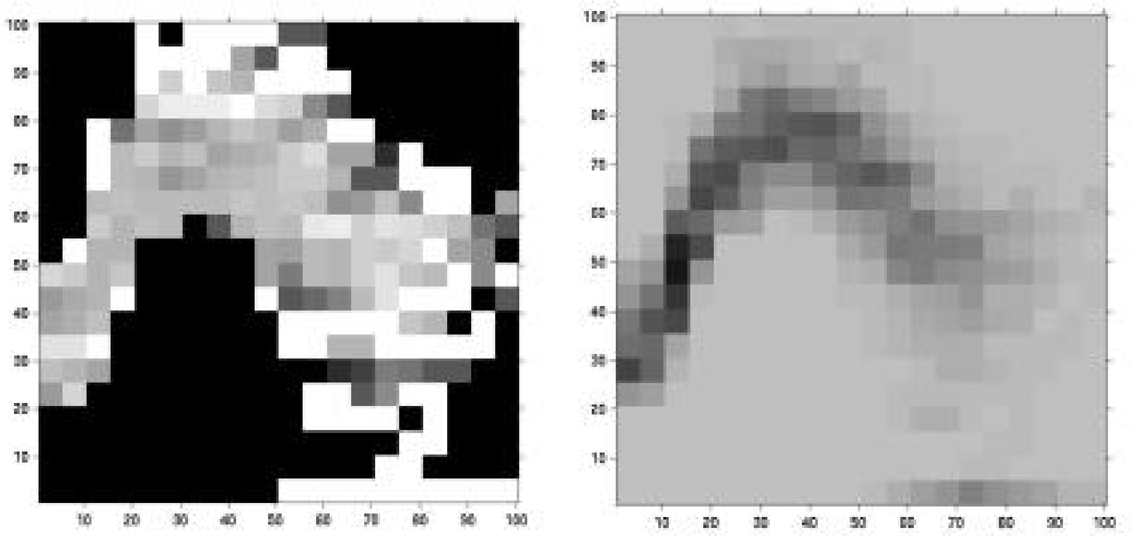

Figure 7. Probabilistic dynamic maps for 100 trials of the scenario in Figure 2. On the left, brightness of a cell is proportional to the probability of a Blue win given that a trajectory transits that cell. On the right, brightness is proportional to the probability of a Blue win if a trajectory does *not* transit a cell.

Discussion

Probabilistic dynamic maps represent probabilities of outcomes given the dynamics of processes. We illustrated the idea by visualizing the probabilities of wins in war games given the dynamics of state variables such as distance to a goal and MNTA, but we suspect the technique is more general. Monte Carlo simulation is a valuable way to explore a state space, but not if one must pore over the trajectories of state variables in every trial. By superimposing these trajectories in a map, and coloring trajectories by outcomes, a great deal of simulation data can be reduced to a surface that represents conditional probabilities of outcomes. As we have seen, one can condition the probabilities on the event that a trajectory passes through a cell, or, as in Figure 7b, on the event that a trajectory passes through some *other* cell.

Some human art goes into probabilistic dynamic maps, but we are working to automate the process. One trick is to find an informative state variable. By informative we mean the state variable over time is a good predictor of outcomes. Finding informative state variables is a statistical model selection problem. One could easily use regression or Bayes' networks, or related techniques, to select predictive state variables were it not for the requirement that the state variable *over time* be predictive. What we need, then, is a characterization of the *dynamics* of a state variable which can be used as a predictor of outcomes. One way to characterize the dynamics of a variable is to cluster the trajectories of the variable by their dynamics, as described previously, then use the cluster membership of a trajectory as a predictor of the outcome of the trajectory. In a recent experiment, this approach selected "distance from the flag" and "relative mass of Red and Blue units" as the state variables whose dynamics best predict the outcome of the two-on-two scenario described earlier. These results are very preliminary but suggest that one aspect of building probabilistic maps might be automated.

Another kind of automation is provided by the BCD algorithm. Maps can be difficult to interpret when several qualitatively different dynamics are superimposed (Fig. 4), but finding qualitatively different trajectories by hand is tedious, error-prone work. The BCD algorithm saves us the trouble. It clusters trajectories with similar dynamics, so when we plot maps for each cluster, the dynamics are much clearer (Fig. 5).

It is equivocal whether maps would be more legible, informative, and acceptable to military experts if they did away with the time dimension. Phase portraits, which physicists use to visualize dynamics, usually plot a state variable x at time t against x at $t+i$, so time becomes implicit and relative, but transitions from state to state are clearer. For some aspects of war games absolute time matters, for others, relative time suffices. Phase lines (Fig. 1) describe both kinds of time: Some dictate where a unit should be at, say, 1300 hours, others dictate that a unit should not proceed beyond the line until it meets another unit. The explicit time dimension of our current maps makes it easy to represent absolute time constraints, but may create the illusion that an event must happen at an absolute time instead of a relative time. On the other hand, absolute time constraints are difficult to represent in phase portraits.

From the military perspective, the greatest weakness of probabilistic dynamic maps is that they do not tell the commander what he or she should *do* in a given state. Clearly, if a state is disadvantageous, the commander should move to a better one, but our maps do not specify how the commander should do this. Perhaps the right course is to retreat, perhaps it is to get reinforcements, or seek high ground, or engage the opponent. As time passes and the commander acts, the state of the game changes, so we can see a *forward* relationship between action and states in every trajectory. What we lack is a backward relationship that tells the commander what he or she should do to effect a transition to a desired state from the current state. Once this problem is solved, probabilistic dynamic maps will be an effective planning tool.

References

1) Atkin, M., D. L. Westbrook and P. R. Cohen. 1999. Capture the Flag: Military Simulation Meets Computer Games. Presented at the *AAAI Spring Symposium on AI and Computer Games.*

2) Atkin, Marc and Paul R. Cohen. 1998. Physical Planning and Dynamics. In *Working Notes of the AAAI Fall Symposium on Distributed Continual Planning. Pp. 4-9.*

3) Atkin, Marc, David L. Westbrook, Paul R. Cohen and Gregory D. Jorstad. 1998. AFS and HAC: Domain-General Agent Simulation and Control. In *Workshop on Software Tools for Developing Agents, AAAI-98*. Pp. 89-95.

4) von Clausewitz, Karl. On War. Viking. 1983.

5) Cohen, Paul. Dynamic maps as representations of verbs. *Proceedings of the Thirteenth European Conference on Artificial Intelligence.*

6) Oates, Tim. 1999. Identifying Distinctive Subsequences in Multivariate Time Series by Clustering. *Proceedings of KDD-99, International Conference on Knowledge Discovery and Data Mining*.

7) Oates, T., Schmill, M. and Cohen, P. 1999. Identifying qualitatively different experiences: Experiments with a mobile robot. *The Sixteenth International Joint Conference on Artificial Intelligence,* IJCAI-99.

8) Oates, Tim, Laura Firoiu and Paul R. Cohen. 1999. Clustering Time Series with Hidden Markov Models and Dynamic Time Warping. To be presented at *IJCAI-99 Workshop on Sequence Learning*.

9) Sebastiani, P., M. Ramoni and P. R. Cohen. 1999. Classification of Sensory Inputs. Presented at the *IJCAI-99 Workshop on Sequence Learning.*

10) Sebastiani, P., M. Ramoni, P. R. Cohen, J. Warwick and J. Davis. 1999. Discovering Dynamics using Bayesian Clustering. In *Proceedings of The Third Symposium on Intelligent Data Analysis*s