Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007)

Dynamic optimization of the strength ratio during a terrestrial conflict

Alexandre Sztykgold, GET/ENST-Bretagne, LUSSI department, CNRS TAMCIC UMR 2872

alexandre.sztykgold@enst.fr

Gilles Coppin, GET/ENST-Bretagne, LUSSI department, CNRS TAMCIC UMR 2872,

gilles.coppin@enst-bretagne.fr

Olivier Hudry, GET/ENST, Computer Science department, CNRS LTCI UMR 5141

olivier.hudry@enst.fr

***Abstract*—The aim of this study is to assist a military decision maker during his decision-making process when applying tactics on the battlefield. For that, we have decided to model the conflict by a game, on which we will seek to find strategies guaranteeing to achieve given goals simultaneously defined in terms of attrition and tracking. The model relies multi-valued graphs, and leads us to solve a stochastic shortest path problem. The employed techniques refer to Temporal Differences methods but also use a heuristic qualification of system states to face algorithmic complexity issues.**

Key words: decision aid, game theory, graph theory, viability theory, Temporal Differences methods, approximate dynamic programming.

I. Context: assistance to the military decision maker

In the military field, an important part of the decision-making process consists in re-expressing and translating one mission assignments into orders and tactics towards subordinate armies. While trying to determine tactics that ensure the reaching of operational goal, the utmost difficulty of this task is related to the great uncertainty that is attached to the enemy moves or strategies. Nowadays, a still classical but most often used approach consists in opposing only two or three hostile scenarios to the same amount of friend hypothetical tactics. With the increase use of new technologies in battlefield management emerged the idea and need to go further in tactical situations analysis and to propose new kinds of decision-making support tools to commanders. This paper presents a contribution to this research direction.

We propose to model the conflict by a "game" – in the sense of differential game theory introduced by Isaacs [7] – where two sides are conflicting. Using this game, we will try to find a strategy which ensures us the completing of mission, no matter what the adversary does.

In the differential game theory, games of tracking and attrition constitute some traditional ones and have been commonly studied but most often in a separate way. One of the originalities in our work is that the nature of our game which is at the same time a tracking game and an attrition game, while describing a case that has been stated as relevant by operational experts. The employed technique we implement in order to solve our decisional problem, is the temporal difference method defined by Sutton in [10], studied by Bertsekas and Tsitsiklis in [5] and used for the resolution of games like backgammon (Tesauro [13]). However, the use of this method for a game of our nature constitutes another originality of our work. In classical games - chess or backgammon for instance - one has to face the problem of opponent strategies but the *opponent goals are known* (or predictible) and often symmetrical for both sides. In our case, we are not able to know exactly the enemy's goals and we are generally not in a zero-sum game.

II. Modeling

A. Conventions and definition of the game

It is supposed that two **sides** (noted as A for friend and E for enemy) are involved in a conflict situation, and that they have at their disposal a number of armies (which we note respectively N_A and N_E). In the following, all notations relating to a side (e.g. A) are valid for the other (e.g. E).

We define the **mission** by an "effect" to produce on the ground and an effect to realize on the enemy, both of them to be effectuated in a limited amount of time. More precisely, the mission will be represented by a location to be reached Z_{obj} before the end of given time T, while preserving at any moment a minimum strength ratio R_{obj} between friendly and enemy armies. We consider Z_{obj}, T and R_{obj} like initially given data attached to the mission.

The **battlefield** is represented here by a multi-valued oriented graph (see [4] and [6]), where each vertex Z_i symbolizes a part of the ground, and where arcs express the actions which the armies are likely to carry out. The valuations express different costs or effectiveness from these actions. For example, we note as *Shoot*(Z_i, Z_j) the effectiveness (between 0 and 1) of fire that an army would deliver from the Z_i vertex towards the Z_j area.

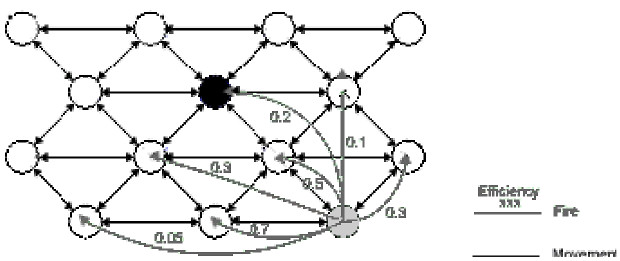

Figure 1 – An example of multi-valued graph

The armies are represented by two variables:

- Z_A^i (k) : location of i[th] army of *A* side at time *k*
- EFF_E^i (k): manpower of the i[th] army of E side at time *k*.

where *i* belongs to [1, N_A]and *k* to [0,*T*]

To facilitate the reading, we will note wit $A_i(k)$ the global state of i[th] army at time *k*, that is to say the couple ($Z_A^i(k)$;EFF_A^i (k)).

For each unit of time, we can order each army to carry out:

- a **movement** towards the successor vertices of the current position in the multivalued graph.
- a **fire** towards one of the the neighbor vertices that are occupied by an army of the opposite side.

We use the following functions:

- $Fire_X$ *(i, z, k)* the function with Boolean values which is equal to 1 if i[th] army of *X* side do fire on vertex *z* at the time *k* and equal to 0 if not.
- *Length* (z_1, z_2) the length between two vertices of the multi-valued graph.
- SP_A *(z, k)* the length of the shortest path between the vertices occupied at time *k* by the armies of *A* and one targeted position *z*:

$$SP_A(z,k) = \min_{i=1...N_A} Length(Z_A^i(k), z)$$

The goal of the game for A is of course to complete his mission whose **constraints of the mission** may be expressed as:

$$SP_A(Z_{obj}, T) = 0$$

$$\forall k = 0 \dots T, \sum_{i=1\dots N_A} EFF_A^i(k) \geq R_{obj} \times \sum_{i=1\dots N_E} EFF_E^i(k)$$

The first constraint expresses that at least 1 friendly army must occupy the target vertex at the final time *T*. It characterizes our game like a tracking one according to the differential game theory (see [7]).

The second constraint constantly forces to have a strength ratio greater than R_{obj}, which expresses the attritional nature of our game.

B. The game seen as system control

At any time *k*, the information characterizing a situation may be expressed along a vector **state**:

$$S(k) = (A^1(k); A^2(k); \dots ; A^{N_A}(k); E^1(k); \dots ; E^{N_E}(k))$$

This approach is inspired by the vision that Berge had of chess game in [3], where he did not treat the player actions by a simple position change on the chess-board, but like a state evolution in a dynamical system.

An **action** is defined for each army by its type and its related targeted position, *i.e.* the couple (action type, ground targets) that symbolizes the order that will be carried out by an army during a time unit. We notice $u_A^i(k)$ the action done by the i[th] army of *A* at time k. For example, action *(FIRE,* $Z_E^j(k)$*)* is a fire on the vertex occupied by the j^{th} enemy's army, and the control *(MOVE,* Z_j*)* represents the ordering of a movement towards the vertex Z_j. To make the reading easier, the type of the action $u_A^i(k)$ is noticed $u_A^i(k)$*.type* (resp. $u_A^i(k)$*.target* for its target).

A control (noticed $u_A(k)$ or $u_E(k)$) is the set of all actions affected to the armies of a side for one turn. We note $U_A(k)$ (resp. $U_E(k)$) the set of effectible controls that *A* (resp. *E*) can use at the time *k*.

At last, we call *F* the **system function** which associates at the current state *S(k)* and controls selected $u_A(k)$ or $u_E(k)$, the successor state *S(k+1)* :

$$S(k+1) = F(S(k); u_A(k), u_E(k))$$

with :

$$A^i(k+1) = (Z_A^i(k+1); EFF_A^i(k+1))$$

$$Z_A^i(k-1) = Z_A^i(k),\ if\ u_A^i(k).type = FIRE$$
$$Z_A^i(k-1) = u_A^i(k).target,\ if\ u_A^i(k).type = MOVE$$
$$EFF_A^i(k+1) = \max(0; EFF_A^i(k) - \sum_{j=1}^{j=N_E} EFF_E^j(k) \times Shoot(Z_j, Z_i) \times Fire_E(j, Z_A^i(k)))$$

and so on for the *E* side.

C. States qualifications

Initial data of our problem are *S(0)*, *T*, Z_{obj}, R_{obj} and the multi-valued graph. With these data, we can in principle enumerate all accessible states. In fact, in the initial state we know about the armies initial locations, and we can thus easily determinate the controls sets $U_A(0)$ and $U_E(0)$. For each combination of controls, using the system function F, we can determinate all successors of *S(0)*, and we have thus a simple iterative mechanism to enumerate all possible plays of our game.

But one can simply show that our game complexity increases exponentially with $T\times(N_A+N_E)$ (see [12] for more details). Looking for a winning strategy (i.e. a sequence of controls such that visited states respect the mission constraints), we cannot reasonably enumerate all solutions. Therefore we propose to use a different way of using and qualifying states.

Aiming at finding the optimal set of controls (i.e. that which minimize the probability of a mission failure) we suggest to qualify states according to the four categories *victorious*, *winning*, *loosing* and *viable*. These categories are defined as follows:

- **Victorious** : A state is said to be *victorious* if the state is final (current time is T) and all characteristic constraints of the mission are satisfied. We note V_{ict} the set of victorious states:

$$V_{ict} = \{E(T) | Z_{obj} \in \{Z_A^1; \ldots; Z_A^{N_A}\}, \sum_{i=1}^{i=N_A} EFF_A^i(T) \geq R_{obj} \times \sum_{i=1}^{i=N_E} EFF_E^i(T)\}$$

- **Winning** : A state is said to be *winning* if there is a set of friendly controls which guarantees to complete the mission (i.e. that it reaches a victorious state) *no matter what the enemy does*. We notice W_{in} the set of winning states, which is built recursively starting from victorious states:

$$W_{in} = V_{ict} \cup \{S(k) | \exists u_A(k)\ t.q.\ \forall u_E(k), F(S(k); u_A(k); u_E(k)) \in W_{in}\}$$

- **Loosing** : A state is *losing* if there is no strategy carrying out in a victorious state (what means that at least one of the characteristic constraints will not have been satisfied). We notice L_{oose} the set of loosing states :

$$L_{oose} = \begin{cases} \{S(k) | SP_A(Z_{obj}, k) \geq N - k\} \\ \cup \\ \{S(k) | \sum_{i=1\ldots N_A} EFF_A^i(k) \leq R_{obj} \times \sum_{i=1\ldots N_E} EFF_E^i(k)\} \end{cases}$$

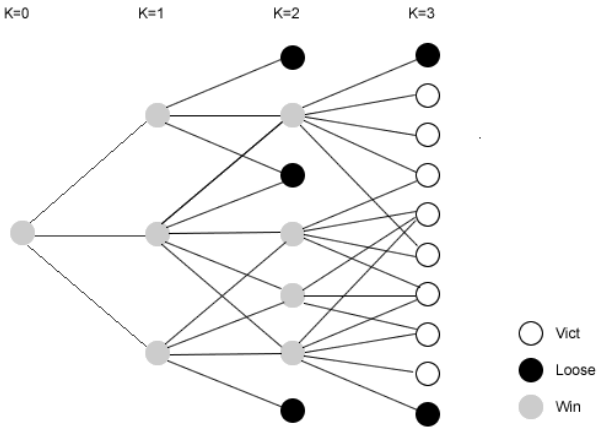

Figure 2 - States qualifications

Our goal is to know if $S(0) \in W_{in}$ and, in case it would be, to extract the strategy which leads to V_{ict}. But because of the problem complexity, it is seldom possible to know if *S(0)* is winning and even less to extract optimal strategy. We proposes therefore to introduce another state qualification, more flexible than W_{in} but which takes us along towards V_{ict}. This qualification is named **viable** (being inspired from viability theory - J.P. Aubin in [1]).

- **Viable**: A state is described as *viable* if there exists at least one control which makes possible to reach a viable state at next time occurrence *k* and a victorious one if next time is *T*. We build the set of viable states recursively starting from V_{ict} :

$$V_{iab} = V_{ict} \cup \{S(k) | \exists u_A(k), \exists u_E(k), F(S(k), u_A(k), u_E(k)) \in V_{iab}\}$$

The advantage of this set is that it is very easy to test if a state belongs to or not, and especially it evolves - by definition - towards V_{ict} with time, in respecting mission constraints. We must thus find a strategy which always keeps the current state in V_{iab}. Moreover, we can **evaluate** each control $u_A^i(k)$ with the viable states number which succeed to him.

Using this states categorization, we have finally re-expressed our problem as a classical dynamic programming one [2], but still have to fix and define properly the way we face the classical "curse of dimensionality" thanks to the use of viable subsets of states. The use we do of these special class of states and our related heuristic is presented in the following paragraphs.

III. RESOLUTION WITH TEMPORAL DIFFERENCES METHOD

A. Modeling the enemy

We look for a control policy which guarantees to fulfill the mission against unknown enemy's intentions. We propose to use a stochastic model for the enemy and consequently to attach probabilities to his behavior. This supposes that we must be able to estimate in any state the occurrence probability of each enemy control that we note $P[S(k+1) \mid (S(k), u_A^i(k))]$. For sake of simplicity, it is supposed here that these probabilities depend only of current state, so our problem is a Markovian process (more information on this subject in [12]).

To determine these probabilities, we modeled the enemy by a set of predicates that label future states and could make sense for him when considering *local and short-term reaction to the current situation*. The predicates express situational or ground local features ("being in a high position", which has a direct impact upon fire efficiency, for instance). The use of predicates as some kind of simplified utility functions allows us to compute an average value for each potential future state (i.e. each potential enemy control) and therefore to order enemy's preferences on his controls.

We note $P_{ref}(S(k))$ the function which returns the sum of validated predicates weights $S(k)$, and for a given friendly control $u_A^i(k)$, we determine the probabilities of choice of the enemy control by :

$$P[S(k+1)|(S(k),u_A(k))] = \frac{Pref(F(S(k),u_A(k),u_E(k)))}{\sum_{u\in U_E(k)} Pref(F(S(k),u_A(k),u))}$$

B. The reward function to be optimized

From expertise extraction from military commanders, we have chosen to reward the greatest margin of freedom, which means we always try to be as close as possible to Z_{obj} and to keep a strength ratio as high as possible. These two concepts are expressed by both following evaluation functions:

- Geographical margin of freedom:

$$f_{move}(S(k)) = T - k - SP_A(Z_{obj}, k)$$

- Effectiveness margin of freedom:

$$f_{fire}(S(k)) = \sum_{i=1}^{i=N_A} EFF_A^i(k) - R_{obj} \cdot \sum_{j=1}^{j=N_E} EFF_E^j(k)$$

It can be seen that if one of these functions is negative then the current state is in L_{oose} because one of the mission constraints is not satisfied. *This also means that these functions define the viable states mentioned previously* especially while denote the **depth** into the viability set of the current state (there is one function for each constraint dimension of this space).

Since we do not want to visit loosing states, we put a big penalty on them. Reciprocally, we allow a great reward to victorious states. Thus, we use a constant G that is very big in accordance to the evaluation function values, and the reward function is used as following:

$$Eval(S(k)) = \begin{cases} G, & if\ S(k) \in V_{ict} \\ -G, & if\ S(k) \in L_{oose} \\ f_{move}(S(k)) + f_{fire}(S(k)) & if\ S(k) \in V_{iab} \end{cases}$$

C. Optimality equations

Extending the evaluation function from states to controls, we note:

$$Eval(u_A(k)) = \sum_{u\in U_E(k)} P[F(S(k),u_A(k),u)|S(k)] \times Eval(F(S(k),u_A(k),u))$$

The Bellman equations (defined in [2]) corresponding to our problem can therefore be defined as follows, in which V_k^* denotes the optimal Bellman evaluation of the decisional states (see Figure 2):

$$\forall S(k), \forall k = 0;\ldots;T \quad 1,$$

$$V_k^*(S(k)) - \max_{u_A(k)\in U_A(k)} \{Eval(u_A(k)) + \sum_{u\in U_E(k)} P[F(S(k),u_A(k),u)|(S(k),u_A(k))] V_{k+1}^*(S(k+1))\}$$

$$\text{with } V_T^*(S(T)) = Eval(S(T))$$

To compute the solutions of these equations, dynamic programming (DP) or Monte-Carlo methods (MC) are commonly applied (see [9]). These two algorithms share the same basic principles: exploring the tree of the possible executions and using backward evaluation of final leaves. The difference consists in the depth with which explorations are performed, while dynamic programming explores with a depth of 1, whereas Monte-Carlo methods, in each state, explore the full depth of the tree. We have chosen to use the intermediary solution offered by the temporal-difference method *TD(λ)* ([2]) (that allows to cover the previous two ones in setting respectively λ to 0 and to 1).

Let us denote d_k the **temporal difference** for a fixed friend control:

$$d_k = Eval(u_A(k)) + \sum_{u\in U_E(k)} P[F(S(k),u_A(k),u)|(S(k),u_A(k))] \times V(F(S(k),u_A(k),u))$$

The update rule of estimate value in state $S(k)$ with *TD(λ)* is:

$$V(S(k)) \leftarrow V(S(k)) + \sum_{m=k}^{m=T-1} \lambda^{m-k} d_m$$

The principle of the temporal differences method is to estimate Bellman values in each state with a *specific control policy*. Then while reiterating, we are able to determine the best policy, that gives in any state the best control. It is shown that the convergence of this kind of method is very fast ([5]), which ensures that in few iterations we quickly approach the expected values.

We have chosen to use *TD(λ)* because it simulates a fixed number of trajectories, updating dynamically the current value, in an adjustable time window (leading to an slightly abusive call "forward checking method" terminology, as proposed by Bertsekas and Tsitsiklis [5]) . We consider that simulating trajectories in forward checking could be close to the simulation of course of actions, that are proposed in many cognitive models of decision-making experts (see [8] or [11] for instance).

IV. Application to an Example

A. About results

To seek solutions of a stochastic problem raises generic difficulties in understanding and presenting results. Indeed, it is extremely rare that a static strategy is victorious regardless enemy controls. That is why we have to solve the optimality equations in any state that has a non-zero probability to be visited.

Thus, our technique allows us to know in any state which control gives us the greatest probability to lead us to the victorious states set, but cannot elicit a fixed strategy to apply from the initial state onward the complete duration of the mission.

For these reasons, we will only give, in this section, the found control at the initial state (which is the only one that we are sure to apply) of the example which we study.

B. Example presentation

The battlefield that we use (see Figure 3) is represented by a multi-valued graph containing 16 vertices, and 2 armies in each side ($N_A = N_E = 2$). The mission consists in being at vertex 7 at time 4 and in preserving the strength-ratio above a 2/1 limit ($N_A \geq 2\ N_E$).

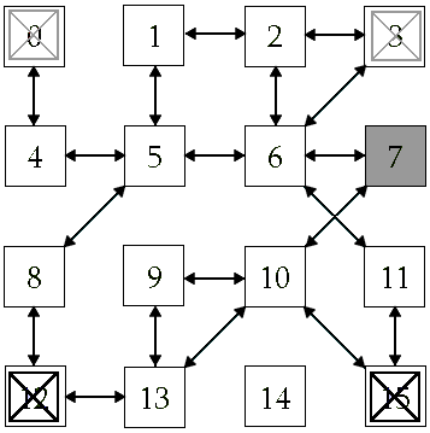

Figure 3 – Battlefield

The efficiency of fire between vertices can be calculated with the following rules:

$Shoot(Z_i, Z_j) = 0$	if they are not neighbor by movement.
$Shoot(Z_i, Z_j) = 0,65$	si $i < j$.
$Shoot(Z_i, Z_j) = 0,35$	si $i > j$.
$Shoot(Z_i, Z_j) = 0,5$	si $i = j$.

That means it is better to be located in smallest indexed vertices.

At initial state *S(0)*, friend armies are located on vertices 12 and 15 and correspond respectively to 20 and 40 troops. Enemy armies are located in 0 and 3, and have respectively 10 and 20 troops. So have we the initial state:

Army	Vertex	Manpower
$A^1(0):$	$Z_A^1(0) = 16$	$EFF_A^1(0) = 40$
$A^2(0):$	$Z_A^2(0) = 12$	$EFF_A^2(0) = 20$
$E^1(0):$	$Z_E^1(0) = 4$	$EFF_E^1(0) = 20$
$E^2(0):$	$Z_E^2(0) = 1$	$EFF_E^2(0) = 10$

This situation is particularly interesting because there is no obvious victorious tactics: this comes from the fact that enemy armies are better located (from the "firing" point of view) whereas the friend armies are superior in manpower.

The limited size of our example has allowed us to apply an enumeration method and to identify two controls as maximizing the probability to finish in V_{ict}. That comes from the fact the best way to win cross the states where friend armies are both on vertex 10 at time 2.

$V^*(S(0))$	$u_A^1(0).type$	$u_A^1(0).target$	$u_A^2(0).type$	$u_A^2(0).target$
1.3×10^6	MOVE	Z_{12}	MOVE	Z_{15}
1.3×10^6	MOVE	Z_{10}	MOVE	Z_{15}

The result provided by such an enumeration – that could not usually be elicited for real-sized instances because of the complexity of our problem – will allow us here to benchmark our heuristic. First results of this benchmark are

presented below.

C. Initial control policy

We present here our first results with the control policy $\pi^r(S(k)) = u_A(k)$, which associates a randomly chosen control among the set of available controls $U_A(k)$ with any state $S(k)$.

For each iteration of the method, one trajectory is simulated, updating values in visited states. We note N the number of iterations. Obviously the greater N is, the more the state values $V_k^{\pi^r}$ approximate Bellman values V_k^*.

D. Results

We implemented temporal differences with N = 10000, which led to very short CPU time of computation (the algorithm spends less than 10 seconds to run) and checked that this performance did not depend on the value of λ.

λ	total trials done	# of trials providing optimal control
$\lambda = 1$	25	21
$\lambda = 0,5$	25	24
$\lambda = 0,1$	25	23

We can observe that more than 84% of trials return the optimal control, and the value of λ does not change this ratio whereas the little number of simulations we did.

V. Conclusion and Perspectives

On a simplified military example, only a few number of trials is enough for Temporal Differences methods to give a policy that is close to optimal. The extension of the presented model and resolution technique to a concrete real problem is therefore an interesting study. Thus, we could obtain a helping tool that could advice military decider in all possible situations. But this kind of application will certainly need to provide a way of evaluating the approximation in order to be properly integrated in the decision system. The next challenge we have is to give a good measurement of solutions according to number of trials.

References

[1] J.-P. Aubin. Viability, control and games: Regulation of complex evolutionary systems under uncertainty and viability constraints. Springer-Verlag, 2005.
[2] R. Bellman. Dynamic programming. Princeton University Press, 1957.
[3] C. Berge. Théorie des graphes et ses applications. Dunod, 1963.
[4] C. Berge. Graphes. Gauthier Villard, 1983.
[5] D. Bertsekas, J. Tsitsiklis. Neuro Dynamic Programming. Athena Scientific,1996.
[6] M. Gondran, M. Minoux. Graphes et algorithmes. Eyrolles, 1995.
[7] R. Isaacs. Jeux différentiels. Théorie des jeux appliqués aux dommaines de la guerre, des poursuites, du contrôle et de l'optimisation. Dunod, 1968.
[8] Calderwood, Klein, Orasanu and Zsambock. Decision making in action : models and methods. Ablex publishing corporation, 1993.
[9] J. Pearl. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley, 1984.
[10] Sutton. Learning to predicted by the method of temporal differences. Machine learning. Technical report, 1988.
[11] A. Sztykgold. Synthèse de l'exercice Destrier du 22/11/2004. Technical report, Centre de Recherche Operationnelle et de Simulation de l'Armée de Terre, 2004.
[12] A. Sztykgold. Compte rendu de recherche 2005-2006 : Optimisation dynamique du rapport de force lors d'un engagement terrestre de haute intensité. Rapport, Division de Simulation et de Recherche Operationnelle, Avril 2006.
[13] G. Tesauro. Practical issues in temporal difference learning. In John E.Moody, Steve J. Hanson, and Richard P. Lippmann, editors, Advances in Neural Information Processing Systems, volume 4, pages 259–266. Morgan Kaufmann Publishers, Inc.