Communications to SIMAI Congress,
ISSN 1827-9015, Vol. 1 (2006)

DOI: 10.1685/CSC06059

A Fast Marching Method for Pursuit-Evasion Games

Emiliano Cristiani
Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate
Università di Roma "La Sapienza"
`cristiani@dmmm.uniroma1.it`

Maurizio Falcone
Dipartimento di Matematica
Università di Roma "La Sapienza"
`falcone@mat.uniroma1.it`

Fast Marching (FM) methods have been introduced by Sethian in [6] to solve efficiently the Hamilton-Jacobi equation $H(x, \nabla u) = 0$ related to a front propagation problem, i.e. $H(x, \nabla u) = c(x)|\nabla u| - 1$. Since then some improvements and extensions have been proposed in order to extend the FM technique to more general equations [2,5,7]. The main assumptions for these methods are the continuity and the convexity of the Hamiltonian H.
We present here an extension of the Fast Marching semi-Lagrangian FM (FM-SL) method proposed in [3] to non convex Hamiltonians, in particular to minmax Hamiltonians which appear in the analysis of differential games. We will discuss the main ideas which are behind this new algorithm also showing some numerical results on classical problems.

Let us consider the Isaacs equation for the lower value of a pursuit-evasion game as in [1]

$$\begin{cases} v(x) + \min_{b\in B}\max_{a\in A}\big\{ -\nabla v \cdot f(x,a,b)\big\} = 1 & x \in \mathbb{R}^2 \backslash \mathcal{T} \\ v(x) = 0 & x \in \partial\mathcal{T} \end{cases} \tag{0.1}$$

where $f : \mathbb{R}^n \times A \times B \to \mathbb{R}^n$ is the dynamics for the game, A and B are two compact sets in $\mathbb{R}^m$ representing respectively the control set for the first player (pursuer) and the second player (evader) and $\mathcal{T}$ is a closed set with non empty interior representing the target for the game (see [1] for more details). The fully discrete scheme based on the discrete dynamic programming principle (see [1,4]) is

$$\begin{cases} w(x_i) = \max_{b\in B}\min_{a\in A}\big\{e^{-h} w(x_i + hf(x,a,b))\big\} + 1 - e^{-h} & x \in (Q\backslash\mathcal{T}) \cap G \\ w(x_i) = 0 & x \in \mathcal{T} \cap G \end{cases} \tag{0.2}$$

where Q is a set containing the target $\mathcal{T}$ and G is the set of the grid nodes x_i, $i = 1, \dots, N$. It is well known that at every node $x_i \in G$ we need to compute the value $w(x_i + hf(x_i, a, b))$ for all $a \in A$ and $b \in B$ by interpolation using the values of the neighboring nodes. For a reconstruction based on linear interpolation in $\mathbb{R}^2$, we will write

$$w(x_i + hf(x_i, a, b)) = I(w(x_{i,1}), w(x_{i,2}), w(x_{i,3}))$$

where I is the interpolation operator and the values $w_{i,k}$, $k = 1, 2, 3$ correspond to the vertices of the triangle containing $x_i + hf(x_i, a, b)$. Note that the scheme can work on structured as well as on unstructured grids. Obviously the choice of the nodes $x_{i,1}, x_{i,2}, x_{i,3}$ to be used in the interpolation depends also on a and b via the dynamics f. We will denote by $x^*_{i,1}, x^*_{i,2}, x^*_{i,3}$ the triple corresponding to the optimal controls a^* and b^*.

Definition 1. *We define the* reachable set *at iteration* n *as the set*

$$\mathcal{R}^n \equiv \{x_i \in G : x^*_{i,1}, x^*_{i,2}, x^*_{i,3} \textit{ are } \text{narrow band} \textit{ or } \text{accepted} \textit{ nodes}\}.$$

The above definition means that if the dynamics is in $\mathcal{R}^n$ then player P(ursuer) can drive the dynamics in the computed zone (that is he wins) whatever player E(vader) does. This allows, in some sense, to get rid of the second player E so that the problem is reduced to a 1-player game. Now let us give a brief sketch of the algorithm.

Algorithm

1. The nodes belonging to the target $\mathcal{T}$ are located and labelled as *accepted* setting their values to $w = 0$. All other nodes are set to $w = 1$ and labelled as *far*.
2. The initial *narrow band* is defined as all the neighbors of the *accepted* nodes such that they are in the *reachable set*.
3. The node in the *narrow band* with the minimal value is labelled as *accepted* and it is removed from the *narrow band*.
4. Neighbors not *accepted* of the last *accepted* node are computed and inserted in the *narrow band* only if they are in the *reachable set*.
5. IF the *narrow band* is not empty go to Step 3, ELSE stop.

Theorem 1. *Let $f(x, a, b^*)$ be a dynamics for which the 1-player FM-SL converges. Then, the FM-SL algorithm for differential games described above computes an approximate solution of (0.1).*

Sketch of the proof.

Let us consider the last *accepted* value w_{min} at the n-th iteration. We have to prove that this value is optimal for both players P and E. Considering the computational procedure of the algorithm, if the value w_{min} was assigned to the node x_i, it means that there exists a trajectory which drives the dynamics from x_i to the target in time $\widehat{T} = -\ln(1 - w_{min})$ (that is P can win in a time $T \leq \widehat{T}$). Therefore any value $w > w_{min}$ can not be optimal for P. On the other hand, it is impossible to get a value w lower than w_{min} at the same node x_i. This result follows by the proof of the classical FM-SL method in [4].

Remark. The choice of discretization step h is fundamental to achieve the convergence of the numerical scheme. In order to compute the optimal controls a^* and b^* we choose a time step h such that the point $x_i + hf(x_i, a, b)$ is in the four neighboring cells of the point x_i. More precisely, if $\Delta x = \Delta y$, we have $h = h_i = \Delta x / \max_{a,b} \|f(x_i, a, b)\|$. Once (a^*, b^*) is computed, we choose $h = h_i = \Delta x / \|f(x_i, a^*, b^*)\|$ in order to avoid to use the value at x_i in the linear interpolation (this is important to establish convergence in a finite number of steps).

Numerical experiments

All numerical experiments were performed on a PC with a Pentium IV 3.06 GHz processor and 512 MB RAM. The CPU times refer to a Matlab (version 7) implementation.

Test 1 (Tag-Chase game)
Two boys P and E are running one after the other in the plane $\mathbb{R}^2$. P wants to catch E in minimal time whereas E wants to avoid the capture. Both of them are running with constant velocity (respectively denoted by v_P and v_E) and they can change their direction instantaneously. In reduced coordinates (see [1] for details) the game corresponds to the choices

$$f(x,y,a,b) = v_P a - v_E b\,, \qquad A = B = B_2(0,1)$$
$$Q = [-2,2]^2\,, \qquad \mathcal{T} = B_2(0,0.1)$$

In the computations G has 51×51 nodes (corresponding to $\Delta x = \Delta y = 0.08$) and the unit ball $B_2(0,1)$ is discretized in 36 controls for both players P and E. We have chosen $v_P = 2$, $v_E = 1$ in order to guarantee the capture of E.
The exact solution is $T(x,y) = \sqrt{x^2+y^2} - 0.1$ (i.e. it is the distance from the target).

method	Δx	L^∞ error	L^1 error	CPU time (sec)
FM-SL	0.16	0.0433	0.5416	51
SL iterative	0.16	0.0449	0.5407	276
FM-SL	0.08	0.0257	0.2918	180
SL iterative	0.08	0.0286	0.2927	1845

In Fig.0.1 we show the level sets of the value function $T = -\ln(1-w)$ computed by FM-SL and the optimal trajectories in the real plane when P starts from the point $(3, 3.5)$ and E starts from the point $(2, 2)$.

Test 2 (Tag-Chase game with constraints on the directions)
This game has the same dynamics of the previous one. The only difference is that now the pursuer P has a constraint on his displacement directions. He can choose his control $a = (\cos\theta, \sin\theta)$ only for $\theta \in [\pi/4, 7\pi/4]$. We chose a grid of 51×51 nodes, (corresponding to $\Delta x = \Delta y = 0.08$) and 16 controls for both players P and E. The CPU time for FM-SL was 36 seconds and that for the classical iterative SL scheme was 662 seconds.
In Fig. 0.2 we show the value function $T = -\ln(1-w)$ computed by FM-SL and its level sets. Fig. 0.3 shows the level sets of the solution computed by the classical iterative SL scheme. Finally, Fig. 0.4 shows the optimal trajectories in the real plane when P starts from the point $(-3.5, 0)$ and E starts from the point $(-2, 1.5)$.
We can see some oscillations in the value function computed by FM-SL. This is due to the fact that in this case the propagation of the level sets is anisotropic (more details will be given in a forthcoming paper).

REFERENCES

1. M. Bardi, M. Falcone, P. Soravia, *Numerical methods for pursuit-evasion games via viscosity solutions*, Stochastic and differential games, Ann. internat. Soc. Dynam. Games, Vol. 4, pp. 105-175, Birkhäuser Boston, Boston, MA, 1999.

2. E. Carlini, E. Cristiani, N. Forcadel, *A non-monotone Fast Marching scheme for a Hamilton-Jacobi equation modelling dislocation dynamics*, accepted for publication in Proceedings of ENUMATH 2005, Santiago de Compostela (Spain), July, 18-22, 2005.

3. E. Cristiani, M. Falcone, *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, preprint, Dipartimento di Matematica, 2005. Submitted. Preprint server: `http://cpde.iac.rm.cnr.it/`

4. M. Falcone, *Numerical solution of dynamic programming equations*, Appendix A in M. Bardi, I. Capuzzo Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, Birkhäuser, 1997.

5. E. Prados, S. Soatto, *Fast Marching Method for Generic Shape From Shading*, Proceedings of VLSM'05 (3rd International Workshop on Variational, Geometric and Level Set Methods in Computer Vision), Volume 3752, 320-331, October 2005.

6. J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, **93** (1996), 1591-1595.

7. Y.R. Tsai, L.T. Cheng, S. Osher, H. Zhao, *Fast sweeping algorithms for a class of Hamilton-Jacobi equations*, SIAM J. Numer. Anal., **41** (2003), 673-694.

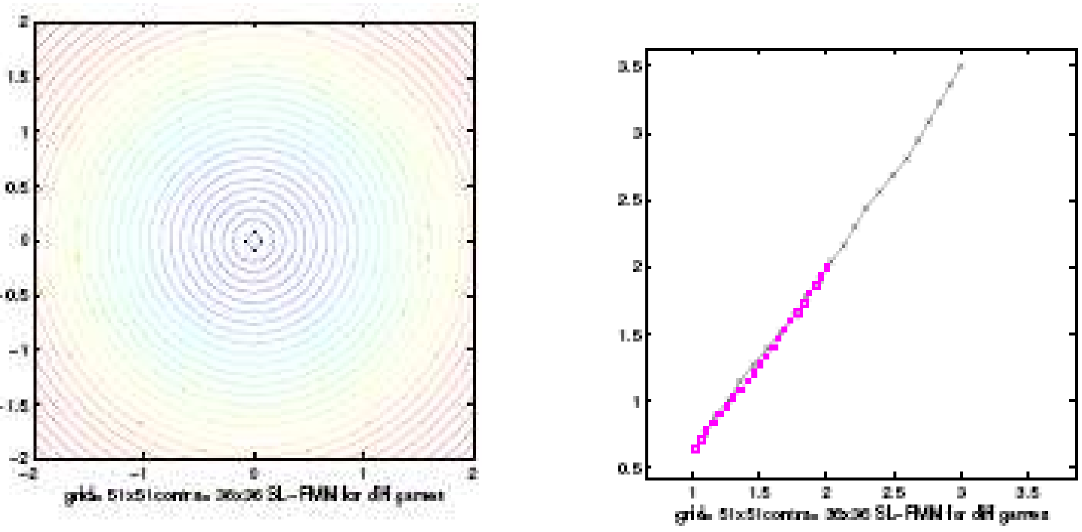

Figure 0.1: Tag-Chase game, FM-SL. Level sets of $T = -\ln(1-w)$ (left) and optimal trajectories (right).

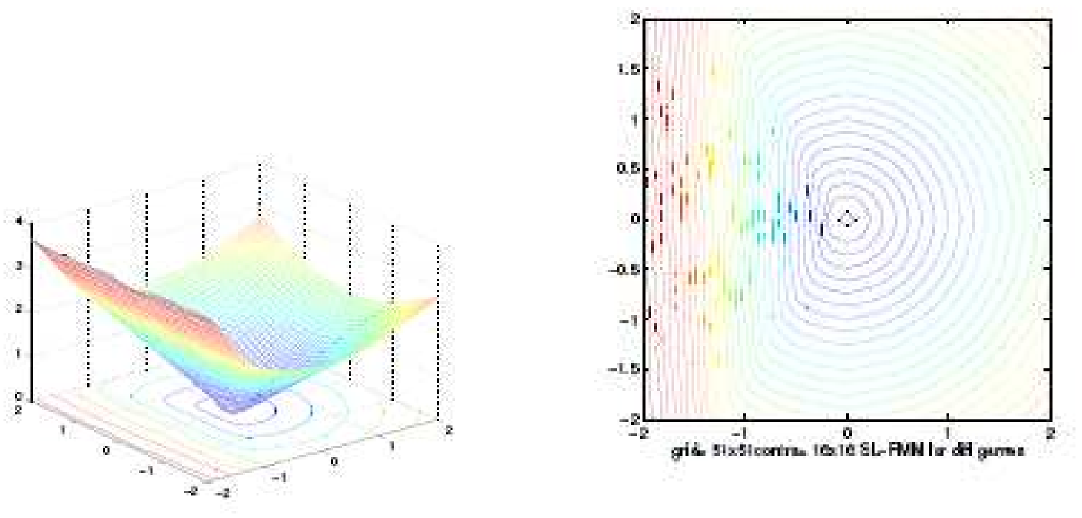

Figure 0.2: Constrained Tag-Chase game, FM-SL. Value function $T = -\ln(1-w)$ (left) and its level sets (right).

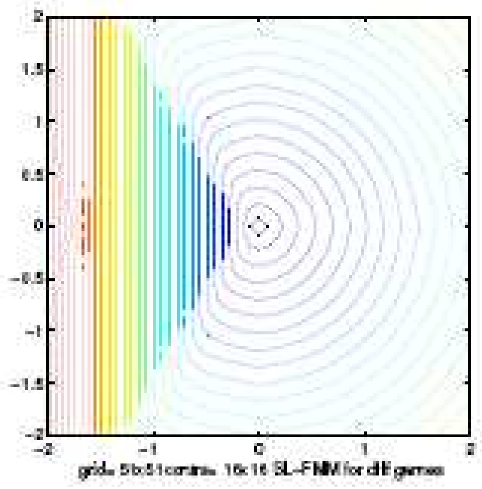

Figure 0.3: Constrained Tag-Chase game, iterative SL scheme. Level sets of the value function.

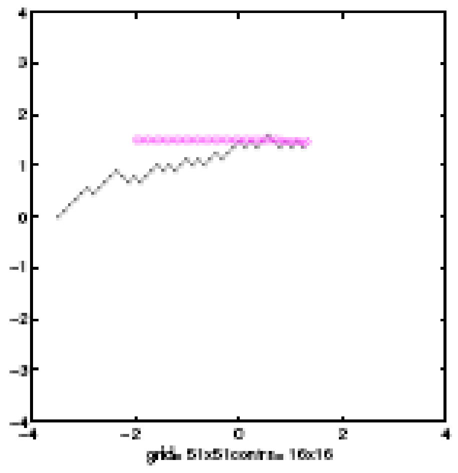

Figure 0.4: Constrained Tag-Chase game, FM-SL. Optimal trajectories.