Domain decomposition based parallel Howard's algorithm

Domain decomposition based parallel Howard's algorithm

Adriano Festa
INSA Rouen LMI lab
adriano.festa@insa-rouen.fr
Avenue de l'Université,
76800 Saint-Étienne-du-Rouvray, France

February 14, 2020

Abstract

The Howard's algorithm, a technique of resolution for discrete Hamilton-Jacobi equations, is of large use in applications for its high efficiency and good performances. A useful characteristic of the method is the superlinear convergence which, in presence of a finite number of controls, is reached in finite time. Performances of the method can be significantly improved using parallel computing. Building a parallel version of the method is not trivial because of the hyperbolic nature of the problem. In this paper we propose a parallel version of the Howard's algorithm driven by an idea of domain decomposition. This permits to derive some important properties and to prove the convergence under standard assumptions. The good features of the algorithm are shown through some tests and examples.

Keywords: Howard's algorithm (policy iterations), parallel computing, domain decomposition
2000 MSC: 49M15, 65Y05, 65N55

1 Introduction

The *Howard's algorithm* (also known as *policy iteration algorithm*) is a classical method for computing the solution of a *discrete Hamilton-Jacobi (HJ) equation*. This technique, developed by Bellman and Howard [7, 20], is of large use in applications thanks to its good proprieties of efficiency and simplicity.

This algorithm is generally more efficient than other techniques of resolution – the convergence is superlinear and even quadratic in special cases (see [8]) – and always faster than *value iteration* and at least comparable to other iterative methods normally adopted (cf. Section 4, see also [21, 24]).

A high efficient alternative is represented by modern *fast techniques* such as Fast Marching [25] and Fast Sweeping [30, 31]. Nevertheless those approaches typically require some restrictive hypothesis on the dynamics of the system suffering in consequence of a limited applicability. The policy algorithm instead is extremely general and may be easily adapted to various special cases of interest (cf. Section 5) and to second order problems [12, §3.2]. As a drawback, the technique requires the resolution of large linear systems. This can be a difficult step when we approximate equations in spaces of a high dimension. In this paper, we deal with this problem using parallel computing.

The application of parallel computing to HJ equations is a subject of actual interest and recent development. It is in fact an effective tool to overcome difficulties caused by memory storage restrictions and CPU speed limitations in the resolution of real problems.

In literature, at our knowledge, the first parallel algorithm proposed in the context of HJ equations is [28]. In this paper the authors discuss the numerical solution of the Bellman equation related to an exit time problem for a diffusion process (i.e. for second order elliptic problems). A successive work is [10] in which an operator of semilagrangian kind is described and studied. More recently the issue was discussed in [32] where the authors pass to an equivalent quasi variational inequality and propose a domain decomposition technique. In [9, 18] there are presented two different multigrid approaches to obtain a decomposition of the domain in subsets that can be solved independently.

Our approach is slightly different. If we decompose the problem directly in its differential form we can give an easy and consistent interpretation of the condition to impose on the boundaries of the sub-domains. Thereafter we pass to a discrete version of such decomposed problem. Now it is easy to show the convergence of the algorithm to the discrete solution.

The paper is structured as follows: in Section 2 we recall the Howard's algorithm and the relation with the differential problem focusing on the case of its optimal control interpretation. In Section 3 we present the algorithm and we study the convergence. Section 4 is dedicated to test the performances and to show the advantages and speed-up factors with respect to the non parallel version. We end by presenting some possible extensions of the technique to some problems of interest: target problems, obstacle avoidance and max-min problems.

2 Howard's algorithm

The problem considered is the following. Let be Ω bounded open domain of $\mathbb{R}^d$ $(d \geq 1)$; the steady, first order HJ equation is:

$$\begin{cases} \lambda v(x) + H(x, Dv(x)) = 0 & x \in \Omega, \\ v(x) = g(x) & x \in \partial\Omega, \end{cases} \tag{1}$$

where, following the *optimal control interpretation*, $\lambda \in \mathbb{R}^+$ is the *discount factor*, $g : \Omega \to \mathbb{R}$ is the *exit cost*, and the *Hamiltonian* $H : \Omega \times \mathbb{R}^d \to \mathbb{R}$ is defined by: $H(x,p) := \inf_{\alpha \in \mathcal{A}}\{-f(x,\alpha) \cdot p - l(x,\alpha)\}$ with $f : \Omega \times \mathcal{A} \to \mathbb{R}$ (*dynamics*), $l : \Omega \times \mathcal{A} \to \mathbb{R}$ (*running cost*) and $\mathcal{A}$ a compact set. The choice of such Hamiltonian is not restrictive but useful to simplify the presentation. In Section 5 we extend the study to other Hamiltonians.

Under classical assumptions on the data (we can suppose $f(\cdot,\cdot)$ and $l(\cdot,\cdot)$ continuous, $f(\cdot,\alpha)$ and $l(\cdot,\alpha)$ Lipschitz continuous for all $\alpha \in \mathcal{A}$ and the *Soner's condition* [26] is verified), it is known (see i.e. [2, Th.3.1]) that the equation (1) admits a unique continuous solution $v : \overline{\Omega} \to \mathbb{R}$ in the *viscosity solutions* sense.

The solution v is the value function of the infinite horizon problem with exit cost, where τ_x is the *first time of exit* from Ω:

$$v(x) = \inf_{a(\cdot) \in L^\infty([0,+\infty[;\mathcal{A})} \int_0^{\tau_x(a)} l(y_x(s), a(s)) e^{-\lambda s}\, ds + e^{-\lambda \tau_x(a)} g(y_x(\tau_x(a))),$$

where $y_x(\cdot)$ is a.e. solution of $\begin{cases} \dot{y}(t) = f(y(t), a(t)) \\ y(0) = x\ . \end{cases}$

Many numerical schemes for the approximation of this problem have been proposed. Let us mention Finite Differences Schemes [14, 27], semilagrangian [15], Discontinuous Galerkin [13] and many others. In this paper we focus on a *monotone*, *consistent* and *stable* scheme (wide class including the first two mentioned above) for the approximation of (1).

Considered a discrete grid G with N points x_j, $j = 1, ..., N$ on the domain $\overline{\Omega}$, the finite N-dimensional approximation of v, V is the solution of the discrete equation ($V_j = V(x_j)$)

$$F_i^h(V_1, ..., V_N) = F_i^h(V) = 0, \qquad i \in \{1, ..., N\}, \tag{2}$$

where $h := \max diam S_j$ (biggest diameter of the family of simplices S_j on G) is the discretization parameter. The Dirichlet conditions of (1) are

$$F_j^h(V_1, ..., V_N) := V_j - g(x_j), \quad x_j \in \partial\Omega. \tag{3}$$

We assume on F some standard hypotheses:

(H1*) *Monotony.* For every choice of two vectors V, W such that, $V \geq W$ (component-wise) then $F_i^h(V_1, ..., V_N) \geq F_i^h(W_1, ..., W_N)$ for all $i \in \{1, ..., N\}$.

(H2*) *Stability.* If the data of the problem are finite, for every vector V, there exists a $C \geq 0$ such that V, solution of (2), is bounded by C i.e. $\|V\|_\infty = \max_{i=1,...,N} |V_i| \leq C$ independently from h.

(H3) *Consistency.* It is assumed that $F_i^h(\varphi(y_i)+\xi, ..., \varphi(y_i)+\xi) \to \lambda\varphi(x_i) + H(x_i, \varphi(x_i), D\varphi(x_i))$ for every $\varphi \in C^1(\Omega)$, $x_i \in \Omega$, with $h \to 0^+$, $y_i \to x_i$, and $\xi \to 0^+$.

Remark 1. *Under Hypotheses (H1*), (H2*), (H3) it has been discussed and proved [27] (other examples are [14, 15]) that V solution of* (2) *converges to v viscosity solution of* (1) *for $h \to 0^+$.*

The special form of the Hamiltonian H gives us a correspondent special structure of the scheme F, in particular, with a rearrangement of the terms, the discrete problem (2) can be written as the nonlinear system

$$V \in \mathbb{R}^N; \quad \min_{\alpha \in \mathcal{A}^N} (B(\alpha)V - c_g(\alpha)) = 0, \tag{4}$$

where B is a $N \times N$ matrix and c_g is a N vector. We underline that c_g contains all information about the Dirichlet conditions (3). The *policy iteration algorithm* (or Howard's algorithm) consists in a two-steps iteration with an alternating improvement of the policy and the value function, as shown in Table 1.

It is known [8] that under a monotonicity assumption on the matrices $B(\alpha)$ (we recall that a matrix is monotone if and only if it is invertible and every element of its inverse are non negative) the above algorithm is a non smooth Newton method [22] that converges superlinearly to the discrete solution of problem. The convergence of the algorithm is also discussed in the earlier works [24, 21] where the results are given in a more regular framework. Additionally, if $\mathcal{A}$ has a finite number of elements – this is the case of a discretized space of the controls – then the algorithm converges in a finite number of iterations.

We call, for a fixed vector $V \in \mathbb{R}^n$, the subspace of controls $\mathcal{A}(V) := \arg\min B(\alpha)V - c_g(\alpha)$

Proposition 1. *Assume the matrix $B(\alpha)$ is invertible. If (H1*) holds true, then $B(\alpha)$ is monotone and not null for every $\alpha \in \mathcal{A}(V)$ with $V \in \mathbb{R}^n$.*

Proof. For a positive vector V, consider a vector W such that $W - V \geq 0$ componentwise, then for H1*

$$\begin{aligned} B(\bar\alpha)W - c_g(\bar\alpha) \geq \min_{\alpha\in\mathcal{A}} B(\alpha)W - c_g(\alpha) \geq \min_{\alpha\in\mathcal{A}} B(\alpha)V - c_g(\alpha) \\ = B(\bar\alpha)V - c_g(\bar\alpha), \end{aligned}$$

Howard's Algorithm (HA)

Inputs: $B(\cdot)$, $c_g(\cdot)$.
Initialize $V^0 \in \mathbb{R}^N$ and $\alpha_0 \in \mathcal{A}^N$
Iterate $k \geq 0$:

i) Find $V^k \in \mathbb{R}^N$ solution of $B(\alpha^k)V^k = c_g(\alpha^k)$.
If $k \geq 1$ and $V^k = V^{k-1}$, then stop. Otherwise go to (ii).

ii) $\alpha^{k+1} := \operatorname*{arg\,min}_{\alpha \in \mathcal{A}^n} \big(B(\alpha)V^k - c_g(\alpha)\big)$.
Set $k := k+1$ and go to (i)

Outputs: V^k.

Table 1: Pseudo-code of HA

where $\bar{\alpha} \in \mathcal{A}(V)$, therefore

$$B(\bar{\alpha})(W - V) \geq 0.$$

Suppose now that the i^{th} column of $B^{-1}(\bar{\alpha})$ has a negative entry: choosing $W - V = e_i$ (e_i is i^{th} column of the identity matrix) multiplying the previous relation for $B^{-1}(\bar{\alpha})$ we have a contradiction. Then $B(\bar{\alpha})$ is monotone. $\square$

***Example* 1** (1D, upwind scheme)**.** *An example of matrix $B(\alpha)$ and vector $c_g(\alpha)$ is given by the upwind explicit Euler scheme (we limit the description to dimension one to avoid an over-complication of the notation)*

$$\begin{cases} V_0 = g(x_0) \\ \lambda V_i = \min\limits_{\alpha_i \in \mathcal{A}} \left(l(x_i, \alpha_i) + f_i^+(\alpha_i) \frac{V_{i+1} - V_i}{h} + f_i^-(\alpha_i) \frac{V_i - V_{i-1}}{h} \right), \quad i \in \{1, ..., N\} \\ V_{N+1} = g(x_{N+1}) \end{cases} \tag{5}$$

where $\{x_i\}$ are the points of a uniform discrete grid consisting in $N+2$ knots of distance h. Moreover, $f_i^+(\alpha_i) = \max\{0, f(x_i, \alpha_i)\}$ and $f_i^-(\alpha_i) = \min\{0, f(x_i, \alpha_i)\}$. In this case the system (4) *is*

$$B(\alpha) = \begin{pmatrix} 1 + \frac{[f_1^+ - f_1^-]}{h\lambda} & -\frac{f_1^+}{h\lambda} & 0 & \cdots & 0 \\ \frac{f_2^-}{h\lambda} & 1 + \frac{[f_2^+ - f_2^-]}{h\lambda} & -\frac{f_2^+}{h\lambda} & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & -\frac{f_{N-1}^+}{h\lambda} \\ 0 & \cdots & \cdots & \frac{f_N^-}{h\lambda} & 1 + \frac{[f_N^+ - f_N^-]}{h\lambda} \end{pmatrix},$$

and

$$c_g(\alpha) = \frac{1}{h\lambda}\begin{pmatrix} -f_1^-\ g(x_0) + hl(x_1,\alpha_1) \\ hl(x_2,\alpha_2) \\ \vdots \\ hl(x_{N-1},\alpha_{N-1}) \\ +f_N^+\ g(x_{N+1}) + hl(x_N,\alpha_N) \end{pmatrix}.$$

It is straightforward that the solution of Howard's algorithm, verifying $\min_\alpha B(\alpha)V - c_g = 0$, *is the solution of* (5).

***Example* 2** (1D, semilagrangian scheme)**.** *If we consider the standard 1D semilagrangian scheme, the matrix* $B(\alpha)$ *and the vector* $c_g(\alpha)$ *are*

$$B(\alpha) = \begin{pmatrix} 1-\beta b_1(\alpha_1) & -\beta b_2(\alpha_1) & \cdots & -\beta b_N(\alpha_1) \\ -\beta b_1(\alpha_2) & 1-\beta b_2(\alpha_2) & \cdots & -\beta b_N(\alpha_2) \\ \ddots & \ddots & \ddots & \ddots \\ -\beta b_1(\alpha_N) & \cdots & -\beta b_{N-1}(\alpha_N) & 1-\beta b_N(\alpha_N) \end{pmatrix},$$

and

$$c_g(\alpha) = \begin{pmatrix} hl(x_1,\alpha_1) + \beta b_0(\alpha_1) g(x_0) \\ hl(x_2,\alpha_2) \\ \vdots \\ hl(x_{N-1},\alpha_{N-1}) \\ hl(x_N,\alpha_N) + \beta b_{N+1}(\alpha_N) g(x_{N+1}) \end{pmatrix},$$

where $\beta := (1-\lambda h)$ *and the coefficients* b_i *are the weights of a chosen interpolation* $\mathbb{I}[V](x_i + hf(x_i,\alpha_j)) = \sum_{i=0}^{N+1} b_i(\alpha_j)V_i$.

Despite the good performances of the policy algorithm as a *speeding up* technique, in particular in presence of a convenient initialization (as shown for example in [1]) the technique requires to store data of very big size. For example a 3D problem on a squared grid of side n requires the resolution of linear systems with n^6 elements. This limits the efficacy of the method and give us a sound motivation to investigate the use of parallel computing to reduce the complexity of the sub problems and the memory storage.

3 Domain decomposition and parallel version

The dependence between various points of the domain in equation (1) makes the use of parallel computing not an easy task to accomplish. The main problem is to pass information between the threads, which is necessary without a prior knowledge of the characteristics of the problem.

3.1 Hamilton Jacobi equation on submanifolds

Our idea is to combine the policy iteration algorithm with a domain decomposition principle for HJ equations. We use the theoretical framework of the resolution of partial differential equations on submanifolds, presented for example in [23, 4]. We consider a decomposition of Ω on a collection of subdomains:

$$\Omega := \bigcup_{i=1}^{M_\Omega} \Omega_i \cup \bigcup_{j=1}^{M_\Gamma} \Gamma_j, \quad \text{with } \overset{\circ}{\Omega}_i \cap \overset{\circ}{\Omega}_j = \emptyset, \quad \text{for } i \neq j, \tag{6}$$

where the interfaces Γ_j, $j = 1, \cdots, M_\Gamma$ are some strata of dimension lower than d defined as the intersection of two subdomains $\overline{\Omega}_i \cap \overline{\Omega}_k$ for $i \neq k$.

The notion of viscosity solution on the manifold must be coherent with the definition elsewhere: we define

Definition 1. *An upper semicontinuos function u in Γ is a subsolution on Γ if for any $\varphi \in C^1(\mathbb{R}^d)$, any $\delta > 0$ sufficiently small and any maximum point $x_0 \in \Gamma_\delta := \{x \, s.t. \, |x - y| < \delta, y \in \Gamma\}$ of $x \to u(x) - \varphi(x)$, it is verified*

$$\lambda\varphi(x_0) + H^\delta(x_0, D\varphi(x_0)) \leq 0,$$

where with $H^\delta(\cdot, \cdot)$ we indicate the Hamiltonian H restricted on Γ_δ.
The definition of supersolution is made accordingly.

Remark 2. *It is worth to underline that, differently from multidomains problems (where the Hamiltonian is discontinuous [4, 23]) there is no need of introducing a special definition of solution on the interfaces. We use the standard definition of viscosity solution on an enlargement of Γ (called Γ_δ).*

Theorem 1. *Let us consider a domain decomposition as stated in* (6)*. The continuous function $\overline{v} : \Omega \to \mathbb{R}$ that verifies for a $\delta > 0$ in the viscosity sense the system*

$$\begin{cases} \lambda\overline{v}(x) + H(x, D\overline{v}(x)) = 0 & x \in \Omega_i, i = 1, ..., M_\Omega \\ \lambda\overline{v}(x) + H^\delta(x, D\overline{v}(x)) = 0 & x \in \Gamma_j, j = 1, ..., M_\Gamma, \\ \overline{v}(x) = g(x), & x \in \partial\Omega, \end{cases} \tag{7}$$

is coincident with the viscosity solution $v(x)$ of (1)*.*

Proof. It is necessary to prove the uniqueness of a continuous viscosity solution for (7). After that we invoke the existence and uniqueness results for the solution v (solution of the original problem) and we observe that it is also a continuous viscosity solution of the system to get the thesis.

We use the classical argument of “doubling of variables”. We recall the main steps of the technique (skipping some technical details that can be found in [2]). For two continuous viscosity solutions $\bar{u}, \bar{v}$ of (7) define the auxiliary function

$$\Phi_\epsilon(x, y) := \bar{u}(x) - \bar{v}(y) - \frac{|x - y|^2}{2\epsilon},$$

which has a maximum point in (x_ϵ, y_ϵ). We have that

$$\max_{x\in\overline{\Omega}}(\bar{u}-\bar{v})(x) = \max_{x\in\overline{\Omega}} \Phi_\epsilon(x,x) \le \max_{x,y\in\overline{\Omega}} \Phi_\epsilon(x,y) = \Phi_\epsilon(x_\epsilon, y_\epsilon).$$

The limit $\liminf_{\epsilon\to 0^+} \Phi_\epsilon(x_\epsilon, y_\epsilon)$ is proved to be non positive taking the derivative of Φ_ϵ and using the properties of (sub-) super-solution for $(\bar{u})$ $\bar{v}$, (for example, [2, Th.II.3.1]). To deal with the interface we can always extract a subsequence $(x_{\epsilon_n}, y_{\epsilon_n})$ definitely in Γ_δ and use the regularity of the Hamiltonian. Exchanging the role between $\bar{u}$ and $\bar{v}$ (both super and subsolutions) we have uniqueness. □

In the next section we propose a parallel algorithm based on the numerical resolution of the decomposed system above. This technique consists of a two steps iteration:

(i) Use (HA) to solve in parallel (n threads) the nonlinear systems obtained after discretization of (7) on the subdomains Ω_i (in this step the values of V are fixed on the boundaries);

(ii) Update the values of V on the interfaces of connection $\bigcup_j \Gamma_j$ by using (HA) on the nonlinear system obtained discretizing the second equation of (7) (in this case the *interior points* of Ω_j are constant).

As it is shown later, this two-step iteration permits the transfer of information through the interfaces during (ii). The procedure is not costless: the number of the steps necessary for its resolution is generally higher than (HA). The advantage is that we solve smaller problems possibly in parallel. Moreover in the case of a finite space of controls the coupling between phase (i) and (ii) produces a succession of results convergent in finite time.

Remark 3. *The point (ii) contains a detail that (although not affecting the convergence of the method) can be influential in the performances. In previous works it has been proven (cf. [10, 16]) that the communication between subdomains can be implemented either only on the interface (as in our case) or introducing an* overlapping region *belonging to both the subdomains in contact where the numerical solution is updated. Being this issue already discussed in literature we choose for simplicity the first case, underlining that better results, in term of performances and scalability, can be obtained with the use of the second technique.*

3.2 Parallel Howard's algorithm

Let us consider (as before) a uniform grid $G := \{x_j : j \in \mathcal{I}\}$, the indices set $\mathcal{I} := \{0, ..., N\}$, and a vector of all the controls on the knots $\alpha := (\alpha_1, ..., \alpha_N)^T \in \mathcal{A}^N$. The domain Ω is decomposed as $\Omega := \cup_{i=1}^n \Omega_i \cup \Gamma$, where, coherently with above $\Gamma := \cup_{j=1}^{M_\Gamma} \Gamma_j$. This decomposition induces a similar structure in the indices set $\mathcal{I} := \mathcal{I}_1 \cup \mathcal{I}_2 \cup ... \cup ... \mathcal{I}_n \cup \mathcal{J}$, where every point x_k of index in $\mathcal{I}_i$ is an "*interior point*" in the sense that for every $x_j \in B_h(x_k)$ (ball centred in x_k of radius h, defined as previously), $j \in \mathcal{I}_i$, for every $j \neq k$. The set $\mathcal{J}$ is the set of all the "*boundary points*", which means, for a $i \in \mathcal{J}$ we have that there exist at least two points x_j, $x_k \in B_h(x_i)$ such that $j \in \mathcal{I}_j$ and $k \in \mathcal{I}_k$ with $j \neq k$.

We build n discrete subproblems on the subdomains Ω_i using a monotone, stable and consistent scheme. In this case the discretization of the Hamiltonian gives for every subdomain Ω_i (therefore in relation with points x_j, $j \in \mathcal{I}_i$) a matrix $\hat{B}_i(\hat{\alpha}_i)$ and a vector $\hat{c}_i(\hat{\alpha}_i, \{V_j\}_{j\in\mathcal{J}})$. We highlight here the dependence of c_i from the boundary points which are, either, points where there are imposed the Dirichlet conditions (data of the problem) or points on the interface Γ which have to be estimated.

Assumed for simplicity that every $\mathcal{I}_i$ has the same number of k elements, called $\bar{k} := card(\mathcal{J})$, we have $k := \frac{N-\bar{k}}{n}$, and $\hat{B}_i(\cdot) \in \mathcal{M}_{k\times k}$, $\hat{c}_i(\cdot,\cdot) \in \mathbb{R}^k$. Solving over Γ we have a matrix $\hat{B}_{n+1}(\hat{\alpha}_{n+1})$ and a vector $\hat{c}_{n+1}(\hat{\alpha}_{n+1}, \{V_j\}_{j\in\mathcal{I}\setminus\mathcal{J}})$, in the spaces, respectively, $\mathcal{M}_{\bar{k}\times\bar{k}}$ and $\mathbb{R}^{\bar{k}}$. (For the 1D case e.g. we can easily verify that $\bar{k} = n-1$). In this framework, the numerical problem after the discretization of equations (7) is the following: *Find* $V := (V_1, ..., V_i, ..., V_n, V_{n+1}) \in \mathbb{R}^N$ *with* $V_i = \{V_j \in \mathbb{R}^k \mid j \in \mathcal{I}_i\}$ *for* $i = 1, ..., n$ *and* $V_{n+1} = \{V_j \in \mathbb{R}^{\bar{k}} \,|j \in \mathcal{J}\}$ *solution of*

$$\begin{cases} \min\limits_{\hat{\alpha}_i\in\mathcal{A}^k} \left(\hat{B}_i(\hat{\alpha}_i)V_i - \hat{c}_i(\hat{\alpha}_i, V_{n+1})\right) = 0, \qquad i = 1, ..., n; \\ \min\limits_{\hat{\alpha}_{n+1}\in\mathcal{A}^{\bar{k}}} \left(\hat{B}_{n+1}(\hat{\alpha}_{n+1})V_{n+1} - \hat{c}_{n+1}(\hat{\alpha}_{n+1}, \{V_j\}_{j\in\{1,...n\}})\right) = 0. \end{cases} \tag{8}$$

The resolution of the first and the second equation of (8) are called respectively *parallel part* and *iterative part* of the method. The parallel and the iterative part are performed alternatively, as a double step solver. The iteration of the algorithm generates a sequence $V^s \in \mathbb{R}^N$ solution of the two steps system

$$\begin{cases} \min\limits_{\alpha\in\mathcal{A}^N} \left(B_i(\alpha)V^{s+2} - c_i(\alpha, V^{s+1})\right) = 0, \qquad i = 1, ..., n, \\ \min\limits_{\alpha\in\mathcal{A}^N} \left(B_{n+1}(\alpha)V^{s+1} - c_{n+1}(\alpha, V^s)\right) = 0, \\ V^0 = V_0, \end{cases} \tag{9}$$

where $B_i(\cdot)$, $c_i(\cdot,\cdot)$ are the matrices and vectors in $\mathcal{M}^{N\times N}$ and $\mathbb{R}^N$ containing $\hat{B}_i(\cdot)$, $\hat{c}_i(\cdot,\cdot)$ and such to return as solution the argument of $c_i(\alpha,\cdot)$ elsewhere. $B_i(\cdot)$ $c_i(\cdot,\cdot)$ with $i \in \{1, ..., n\}$ are: equal to $\hat{B}_i$ in the $\{ik, .., (i+1)k-1\} \times \{ik, .., (i+1)k-1\}$ blocks and equal to the rows $\mathbb{I}_i$ of the identity matrix elsewhere $c_i = \hat{c}_i$ in the $\{ik, .., (i+1)k-1\}$ elements of the vector and $c_i(\cdot, V) = V$ elsewhere (we call these entries *identical arguments*). The same, in the $\{nk+1, .., N\} \times \{nk+1, .., N\}$ block, $\{nk, .., N\}$ elements of

Parallel Howard's Algorithm (PHA)

Inputs: $\hat{B}_i(\cdot)$, $\hat{c}_i\cdot, V_{n+1}^k)$ for $i = 1, ..., n+1$
Initialize $V^0 \in \mathbb{R}^N$ and α^0.
Iterate $k \geq 0$:

1) *(Parallel Part)* for each $i = 1, ..., n$
Call (HA) with inputs $B(\cdot) = \hat{B}_i(\cdot)$ and $c_g(\cdot) = \hat{c}_i(\cdot, V_{n+1}^k)$
Get $V_i^k = \{V^k(x_j) | j \in \mathcal{I}_i\}$.
2) *(Iterative Part)*
Call (HA) with inputs $B(\cdot) = \hat{B}_{n+1}(\cdot)$ and $c_g(\cdot) = \hat{c}_{n+1}(\cdot, \{V_i^k\}_{i=\{1,...,n\}})$
Get $V_{n+1}^k = \{V^k(x_j) | j \in \mathcal{J}\}$.
3) Compose the solution $V^{k+1} = (V_1^k, ..., V_n^k, V_{n+1}^k)$
If $\|V^{k+1} - V^k\|_\infty \leq \epsilon$ then *exit*, otherwise go to (1).

Outputs: V^{k+1}

Table 2: Pseudo-code of PHA

the vector for $i = n+1$. We underline that each equation of (9), negletting the trivial relations, is a nonlinear system on the same dimension than (8). A solution of (8) is a fixed point of (9).

Iteration (9) can be expressed as

$$\begin{cases} F_j^{h,i}(V^{s+2}, V^{s+1}) = 0 & j \in \mathcal{I}_i, \text{ with } i = 1, ..., n \\ F_j^{h,n+1}(V^{s+1}, V^s) = 0 & j \in \mathcal{J} \end{cases}$$

where $F_j^{h,i}(V, W) := \left[\min_{\alpha \in \mathcal{A}^N} (B_i(\alpha)V - c_i(\alpha, W)) \right]_j$ for $j \in \mathcal{I}_i$.

Remark 4. *The hypotheses (H1*-H2*) are adapted to the new framework as below. Such hypotheses are verified by Examples 1,2 and their n-dimentional extensions.*

(H1) Monotony. *For every choice of two vectors V, W such that, $V \geq W$ (component-wise) then $F_j^{h,i}(V, \cdot) \geq F_j^{h,i}(W, \cdot)$ for all $j \in \{1, ..., N\}$ and $i = 1, ..., n+1$.*

(H2) Stability. *If the data of the problem are finite for every vector V and every W s.t. $\|W\|_\infty \leq +\infty$ (where $\|W\|_\infty = \max_j |W_j|$) there exists a $C \geq 0$ such that V solution of $F_j^{h,i}(V, W) = 0$ with $j \in \{1, ..., N\}$ and $i \in \{1, ..., n+1\}$ is bounded by C independently from h.*

From the assumptions on the discretization scheme some specific properties of $B_i(\cdot)$ and $c_i(\cdot,\cdot)$ can be derived.

Proposition 2. *Let us assume $H1-H2$ and*

(H4) *if $W_1 \geq W_2$ then $c_i(\alpha, W_1) \geq c_i(\alpha, W_2)$, for all $i = 1, ..., n+1$, for all $\alpha \in \mathcal{A}$.*

Then it holds true the following.

1. *If invertible, the matrices $B_i(\alpha)$ are monotone, not null for every $i \in \{1, ..., n+1\}$ and for every $\alpha \in \mathcal{A} \cap \arg\min B_i(\alpha)V - c_i(\alpha, V)$ with $V \in \mathbb{R}^N_+$.*

2. *If $\|W\|_\infty < +\infty$ we have that for all $i \in \{1, ..., n+1\}$ and for every $\alpha \in \mathcal{A}$ there exists a $C > 0$ such that*
$$\|c_i(\alpha, W)\|_\infty \leq C\|B_i(\alpha)\|_\infty. \tag{10}$$

3. *V^* is the fixed point of (9). If we have $V \leq V^*$ (resp. $V \geq V^*$) then there exists a $\alpha \in \mathcal{A}$ such that, for all $i = 1, ..., n+1$,*
$$B_i(\alpha)V - c_i(\alpha, V) \leq 0 \quad \textit{(resp. } B_i(\alpha)V - c_i(\alpha, V) \geq 0\textit{).} \tag{11}$$

Proof. To prove 1 we observe that the monotony of $\hat{B}_i(\cdot)$ is sufficient and necessary for the monotony of $B_i(\cdot)$ (elsewhere $B_i(\cdot)$ is a diagonal block matrix with all the other blocks invertible). Therefore the argument is as in Proposition 1, starting from two vectors $W - V := \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} - \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} \in \mathbb{R}^N_+$ with the only difference that we need assumption H4 to get
$$\hat{B}_i(\bar{\alpha})(W_1 - V_1) \geq \hat{c}_i(\bar{\alpha}, W_2) - \hat{c}_i(\bar{\alpha}, V_2) \geq 0, \quad \forall i = 1, ..., n+1;$$
or equivalently
$$B_i(\bar{\alpha})(W - V) \geq c_i(\bar{\alpha}, W) - c_i(\bar{\alpha}, V) \geq 0, \quad \forall i = 1, ..., n+1;$$
then the thesis.

To prove 2 we observe $c_i(\alpha, W) = B_i(\alpha)V$. Thanks to H2 we obtain the thesis. The proof of 3 is a direct consequence of monotony assumption H1 with the definition of V^* as
$$B_i(\alpha)V^* - c_i(\alpha, V^*) = 0, \quad \forall i = 1, ..., n+1.$$

□

Here we introduce a convergence result for the (PHA) algorithm.

Theorem 2. *Assume that the function* $\alpha \in \mathcal{A}^N \to B_i(\alpha) \in \mathcal{M}^{N\times N}$*, with* $B_i(\alpha)$ *invertible,* $(\alpha, x) \in \mathcal{A}^N \times \mathbb{R}^n \to c_i(\alpha, x) \in \mathbb{R}^N$ *are continuous on the variable* α, x *for* $i = 1, ..., n+1$*,* $\mathcal{A}$ *is a compact set of* $\mathbb{R}^d$ *and* $(H1, H2, H4)$ *hold.*

Then there exists a unique V^* *in* $\mathbb{R}^N$ *solution of* (8)*. Moreover the sequence* V^k *generated by the (PHA)* (9) *has the following properties:*

(i) Every element of the sequence V^s *is bounded by a constant* C*, i.e.* $\|V^s\|_\infty \leq C < +\infty$.

(ii) If $V^0 \leq V^*$ *then* $V^s \leq V^{s+1}$ *for all* $k \geq 0$*, vice versa, if* $V^0 \geq V^*$ *then* $V^s \geq V^{s+1}$.

(iii) $V^s \to V^*$ *when* s *tends to* $+\infty$.

Proof. The existence of a solution comes directly from the monotonicity of the matrices $B(\alpha)$, the existence of an inverse and then the existence of a solution of every system of (8). Let us show that such solution is limited as limit of a sequence of vectors of bounded norm. Observing that,

$$\|V^s\|_\infty = \max\{\|V_i^s\|_\infty\}_{i=1,...,n+1}$$

without loss of generality we assume that $\|V^s\|_\infty \equiv \|V_{i^*}^s\|_\infty$. Considering the problem

$$\min_{\alpha\in\mathcal{A}} B_{i^*}(\alpha)V^s - c(\alpha, V^{s-1}) = 0,$$

we have for H2 that if V^{s-1} is bounded then $\|V^s\|_\infty \leq C$. Adding that V^0 is chosen bounded, the thesis follows for induction.

Let us prove the uniqueness: taken $V, W \in \mathbb{R}^N$ two solutions of (9), we define the vector W^* equal to V in the identical arguments of $c_i(\alpha, \cdot)$ and equal to W elsewhere, for a $i \in \{1, ..., n+1\}$. We have that, for a control β (for Proposition 2.3),

$$B_i(\beta)V - c_i(\beta, V) \geq 0 \geq B_i(\beta)W^* - c_i(\beta, W^*) = B_i(\beta)W - c_i(\beta, V)$$

then $B_i(\beta)(V - W) \geq 0$ and for monotonicity $V \geq W$. Exchanging the role of V and W, and for the arbitrary choice of i we get the thesis.

(i) To prove that $V^k \in \mathbb{R}^N$ is an increasing sequence, it is sufficient to show that taken $V_1, V_2 \in \mathbb{R}^N$ solution of

$$\min_{\alpha\in\mathcal{A}} B_i(\alpha)V_2 - c_i(\alpha, V_1) = 0$$

with (the opposite case is similar) $V_1 \leq V^*$, for a choice of $i \in \{1, ..., n+1\}$ is such that $V_2 \geq V_1$. Let us observe, for a choice of $\beta \in \mathcal{A}$ and using (11) of Prop. 2

$$\begin{aligned} 0 = \min_{\alpha\in\mathcal{A}} B_i(\alpha)V_2 - c_i(\alpha, V_1) &\leq B_i(\beta)V_2 - c_i(\beta, V_1) \\ &\leq B_i(\beta)V_2 - (B_i(\beta)V_1 - c_i(\beta, V_1)) - c(\beta, V_1) \end{aligned}$$

then $B_i(\beta)(V_2 - V_1) \geq 0$ and $V_2 \geq V_1$.
We need also to prove that $V_2 \leq V^*$: if it should not be true, then

$$0 \geq B_i(\beta)V_2 - c_i(\beta, V_1) \geq B_i(\beta)V_2 - (B_i(\beta)V_2 - c_i(\beta, V_2)) - c_i(\beta, V_1)$$

and for H4, $V_1 \geq V_2$. This contradicts what stated previously. □

We show that in presence of a finite number of controls the method reaches the fixed point in a finite time.

Proposition 3. *If $Card(A) < +\infty$ and convergence requests of Theorem 2 are verified then (PHA) converges to the solution in less than $Card(A)^N$ iterative steps.*

Proof. Let us consider the abstract formulation $P : x \to y$, where $P(x)$ is determined by N_P parameter in A and $Q : y \to x$, where $Q(y)$ is determined by N_Q parameter in A. If we consider the iteration

$$\begin{array}{l} P(x^k) = y^k \\ Q(y^k) = x^{k+1} \end{array} \tag{12}$$

and we suppose (Theorem 2) $x^k \leq x^{k+1}$, $y^k \leq y^{k+1}$, then called α^k the $N_P + N_Q$ variables in A associated to (x^k, y^k) we know that there exist a k and a l where $k < l \leq Card(A)^{N_P+N_Q}$ such that $\alpha^k = \alpha^l$ and again $(x^k, y^k) = (x^l, y^l)$. Hence (x^k, y^k) is a fixed point of (12).
It is sufficient identifying the process P with the (parallel) resolution on the sub-domains and Q with the iteration on the interfaces between the sub domains to obtain the thesis □

Remark 5. *It is worth to notice that the above estimation is worse than (HA). In fact, the classical algorithm finds the solution in $Card(A)^N$ and (PHA) has the same number of iterative steps. This number has to be multiplied for $Card(A)^{(M_1+M_2)}$ (M_1 is the maximum number of nodes in a sub-domain and M_2 is the number of nodes belonging to the interface). Hence the total number of steps is $Card(A)^{(N+M_1+M_2)}$: more than the classical case. In this analysis it is not possible to see the advantages of the decomposition technique: any computational step involves smaller and simpler problem with a reduction of time and memory storage needed.*

4 Performances, tuning parameters

The performances of the algorithm and its speeding up traits are tested in this section. We use a standard academic example where we can observe all the main features of our technique. The efficiency of (HA) in relation to iterative methods as value iteration (VI) (and its successive modifications as the monotone value iteration in the set of subsolutions presented in [2]

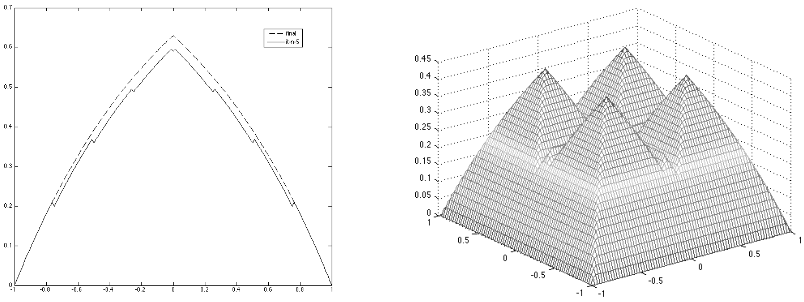


Figure 1: Approximated solution of the (PHA) (left) in the 1D case, final time (dotted) and fifth iteration (solid), in the 2D case (right, 3rd iteration).

and the Gauss-Seidel variation in [19]) has been studied and discussed in [1]. In this section we focus on comparing the non parallel version (HA) with the parallel (PHA). Where useful, we use also the standard value iteration method (VI) as a reference for the reader.
The tests are performed on a normal laptop using the GPU as parallel machine. In particular we run the main structure of the code on the main processor 2.8 GHz Intel Core i7 (the code is implemented in Mathworks' MATLAB R2015a). This main part (which computes also the *sequential step* of (PHA)) opens in parallel $n-$threads on the GPU (1 thread-1 core in local memory) using the Parallel Computing Toolbox (version 6.6 [29]). The GPU is a Nvidia Geforce with 384 CUDA cores of specs: Clock up to 900 MHz, Local memory DDR3/GDDR5 (Interface width 128 bit, Bandwidth 80 GB/sec).

1D problem Consider the unidimensional problem

$$\begin{cases} u(x) + |Du(x)| = 1 & x \in (-1,1), \\ u(-1) = u(1) = 0. \end{cases} \tag{13}$$

It is well known that this equation (*Eikonal equation*) models the distance from the boundary of the domain, scaled by an exponential factor. Through a standard upwind scheme we obtain the problem in the form (4). In Table 4 we show a comparison in term of speed and efficacy of our algorithm and the (HA) in the case of a two thread resolution. It is possible to see as the parallel technique is not convenient in all the situations. This is related to the low number of parallel threads which are not sufficient to justify the construction. In the successive test, keeping a fixed number of nodes processed and tuning the number of threads, it is possible to notice the influence of such variable in the performances.

Table 3: Testing performances, 1D. Our method compared with the (HA) and (VI) with two sub-problems. Performances are described in terms of time in seconds (t.), iterations (it.) relative to the parallel part of the algorithm (par.), the iterative part (itp.) and speed-up factor (SU=time(HA)/time(PHA))

	VI		HA		PHA (2-threads)				
dx	t.	it.	t.	it.	t. (par.)	it.	t. (itp.)	t. (tot.)	SU
2E1	8.9E-3	115	1E-3	10	1E-4	4	1E-5	1E-3	1
4E1	51E-3	232	6E-3	20	8E-4	5	1E-5	3E-3	2
8E1	0.88	521	0.09	40	7E-3	6	2E-5	0.04	2.3
1.6E2	2.12	882	0.32	80	0.048	8	1E-4	0.36	0.9
3.2E2	16.8	1420	2.22	160	0.34	14	8E-4	3.26	0.7

Table 4: Testing performances, 1D. Our method compared with the classic Howard's with various number of threads

nodes: $8E1$	HA		PHA				
threads	t.	it.	t. (par.)	it.	t. (itp.)	t. (total)	SU
2			0.48	4	1E-4	3.6E-1	0.9
4			8E-3	6	1E-4	8.6E-2	3.7
8	0.32	80	18E-4	7	6E-4	1.4E-2	22.9
16			7E-4	10	4E-4	$\boxed{9.5E\text{-}3}$	$\boxed{33.7}$
32			2E-4	8	6E-3	1.1E-2	29.1

In Table 4 we compare the iterations and the time necessary to reach the approximated solution. We study in the various phases of the algorithm, the maximal time necessary to solve every sub-problem (first column), number of iterations and time elapsed for the iterative part (which passes the information through the threads, next column), the total time and finally the speed-up factor of our technique compared to (HA). It is highlighted the optimal choice of number of threads (16 thread); it is clear that choice changes varying the number of the nodes processed. Therefore an additional work is required to tune the number of threads according to the characteristics of the problem: otherwise it is possible to loose completely the gain obtained through parallel computing and to get worse performances even compared with the (HA) (cf. 1D problem Table 3).

2D problem Let us consider the approximation of the scaled distance function from the boundary of the square $\Omega := (-1, 1) \times (-1, 1)$ solution of

Table 5: Testing performances, 2D. Comparison between (HA) and (PHA) with 4 threads

	VI		HA		PHA (4-threads)				
nodes	t.	it.	t.	it.	t. (par.)	it.	t. (itp.)	t. (total)	SU
4E2	0.24	168	5E-2	11	9E-3	2	2E-2	4E-2	1.3
1.6E3	10.8	21	2.41	21	5E-2	2	3E-2	0.14	17.2
6.4E3	351.5	521	73.3	40	2.5	2	0.15	7.83	9.4
2.56E4	>E5	-	>E5	-	5	76	1.293	383.3	-

the eikonal equation

$$\begin{cases} u(x) + \inf\limits_{a \in B(0,1)} \{-a \cdot Du(x)\} = 1 & x \in \Omega, \\ u(x) = 0 & x \in \partial\Omega. \end{cases} \tag{14}$$

where $B(0,1) \in \mathbb{R}^2$ is the usual unit ball. For the discretization of the problem we use the standard upwind discretization.

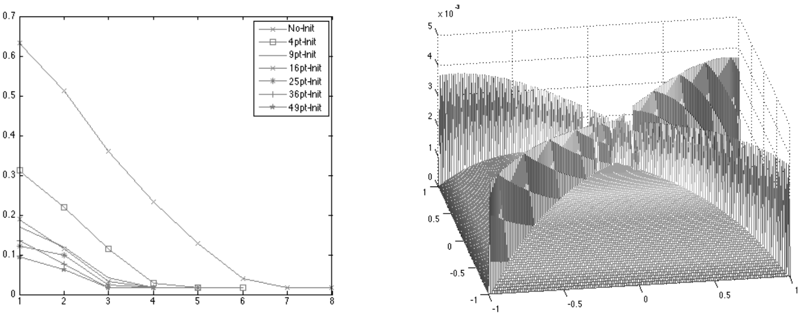


Figure 2: Comparison of the speed of convergence of our method in the case of various initial guess to in the L^∞-norm (left). Distribution of the error $dx = 0.125$, 16 threads (right).

In Table 5 performances of the (HA) are compared with (PHA). In this case the number of threads are fixed to 4; the parallel technique is evaluated in terms of: maximum elapsed time in one thread (first column), time and number of iterations of the iterative part (third and fourth columns), total time and speed-up factor. In both of the cases the control set $A := B(0,1)$ is substituted by a $32-$points discrete version. It is evident, in the comparison, an improvement of the speed of the algorithm more consistent than the 1D case.

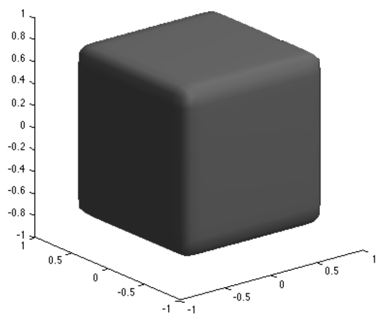
In Table 6, we compare the performances for various choices of the number of threads, for a fixed number of nodes to compute. As in the 1D case it is possible to see how an optimal choice of the number of threads can

Table 6: Testing performances, 2D. Comparing different choices of the number of threads

nodes: 6.4E3	HA		PHA				
threads	t.	it.	t. (par.)	it.	t. (itp.)	t. (total)	SU
4			2.5	2	1.5E-1	7.83	9.4
9			9E-1	5	0.5	5.08	14.4
16	73.3	9	5E-2	13	1.6	$\boxed{1.87}$	$\boxed{40.1}$
25			3E-2	12	2.4	2.52	29.1
36			1.6E-2	18	6.04	6.11	12

drastically strike down the time of convergence. In Figure 2 it is possible to see the distribution of the error. As predictable, the highest concentration corresponds to the non-smooth points of the solution.

Remark 6. *The method is sensible to a good initialization of the "internal boundary" points. As is shown in Figure 2 a right initialization, even obtained on a very coarse grid, affects consistently the overall performances (column it. (PHA) in the tables). In this section, all the tests are made with an initialization of the solution on a 4^d-points grid, with d dimension of the domain space. The time necessary to compute the initial solution is always negligeable with respect to the global procedure.*



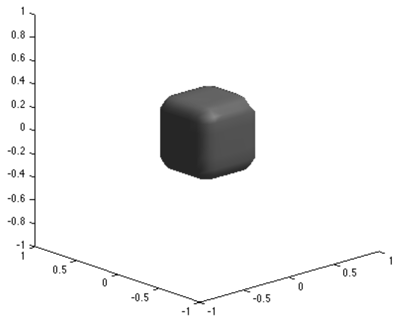


Figure 3: Two level sets (corresponding to levels $u(x) = 0.192$ (left) $u(x) = 0.384$ (right)) of the approximated solution obtained with a $dx = 0.1$ and an $8-$threads PHA.

3D problem We obtain analogue results in the approximation of a 3D problem. Let us consider the domain $\Omega := [-1, 1]^3$ and the equation (14) where $A := B(0, 1)$ unitary ball in $\mathbb{R}^3$. In Figure 3 there are shown two level

Table 7: Testing performances, 3D. Comparison with classical method and (PHA) with 8 threads

	VI		HA		PHA (8-threads)				
nodes	t.	it.	t.	it.	t. (par.)	it.	t. (itp.)	t. (tot.)	SU
1.2E2	18E-3	52	4E-3	4	3E-3	4	2E-3	5E-2	8E-2
1E3	1.12	64	0.22	6	2.6E-2	6	1.6E-2	5.2E-2	4.2
8E3	1.4E3	88	1.64E2	9	0.7	8	1.1	6.78	24.2
6.4E4	>1E5	-	>1E5	-	164	5	4.98	4.94E2	-

Table 8: Testing performances, 3D. Comparing different choices of the number of threads

nodes: 8E3	HA		PHA				
threads	t.	it.	t. (par.)	it.	t. (itp.)	t. (total)	SU
4			4.5	12	1.3	55.3	3
8	1.64E2	9	0.7	8	1.1	$\boxed{6.78}$	$\boxed{24.2}$
18			0.5	10	4.6	9.6	17.1
24			0.6	21	9.4	17.8	9.2

sets of the solution obtained. A comparison with the performances of the (HA) are shown in Table 7 and 8.

Remark 7 (Speed-up and Scalability)**.** *Summing up the results obtained in this section we observe some features of the technique proposed. We can see (results reported in Figure 4) as the parallelization of the Howard algorithm permits to obtain very good performances in term of SU and efficiency (SU divided by the number of threads) where the techniques available in literature (cf. in particular the results in [17] – only in 2D) provided maximal efficiency rates around 1 (typically SU* ≈ 10 *for* $n \approx 10$ *threads). In our case, the coupling between a 'fast technique' like the (HA) and the use of parallel computing is highly effective. We note that the technique suffers of a non satisfactory scalability: we observe again in Figure 4 as the optimal number of processor is finite and determined by the size of the problem. This issue is due to two aspects of the* iterative part *of (PHA): the number of the points to compute in this step and the memory management. The first point is discussed in the next remark while the latter can be overcome with some expedients discussed in [17] as the creation of some regions of overlapping stored in a shared memory between the threads.*

Remark 8. *A particular attention should be dedicated to the resolution of the iterative part to avoid loosing the advantages of parallel computing. To*

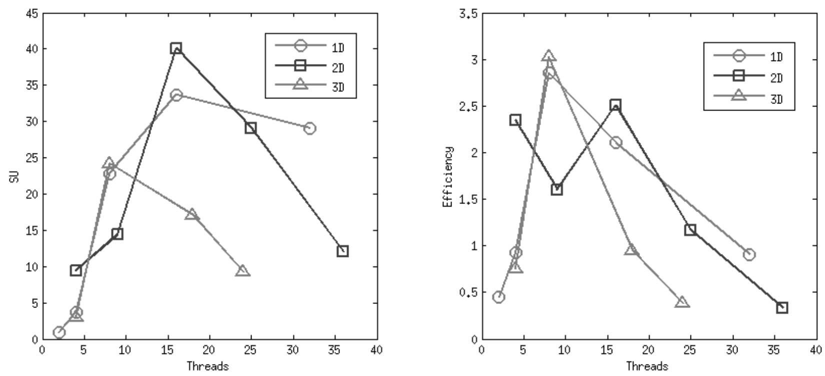


Figure 4: Summary of the results obtained in the 1D-2D-3D tests. Speed-up factor (SU) and Efficiency ($SU/threads$) on number of threads.

explain this, suppose to simplify the procedure considering a square domain (in dimension $d = 1, 2, 3, ..$ *an interval, a square, a cube..) and a successive splitting in equal regular subdomains. Calling* N *the number of total variables and* N_s *the number of the splitting (which generates a division in* N_s^d *subdomains) the number of the elements in every thread of the parallel part is* $\frac{N}{(N_s)^d}$*, and the number of the variables in the iterative part* $\frac{N}{\sqrt[d]{N}}(N_s - 1)d$*. Clearly the optimal choice of the number of threads is such that the elements of the iterative part are balanced with the nodes in each subdomain, so it is straight forward to find the following optimal relation between number of splitting and total elements*

$$N = \Big(N_s^d (N_s - 1)d\Big)^d .$$

Therefore, for a very high number of elements (Figure 5) it is worthless to use a large and non optimal number of threads. This contradiction comes from a bottleneck effect of the resolution on the interfaces of communication between the subdomains. Indeed the complexity of this subproblem grows with the number of threads instead to decrease, complicating the resolution. The problem can be overcome with an additional parallel decomposition of the iterative pass, permitting to decompose each subproblem to an acceptable level of complexity. Imagine to be able to solve (for computational reasons, memory storage, etc.) only problem of dimension "white square" (we refer to Figure 5, right) and to want to solve a bigger problem ("square 1") with an arbitrary number of processors available. Through our technique we decompose the problem in a finite number of subproblems "white square" and a (possibly bigger than the others) problem "square 2". We replicate our parallel procedure for the "square 2" obtaining a collection of manageable

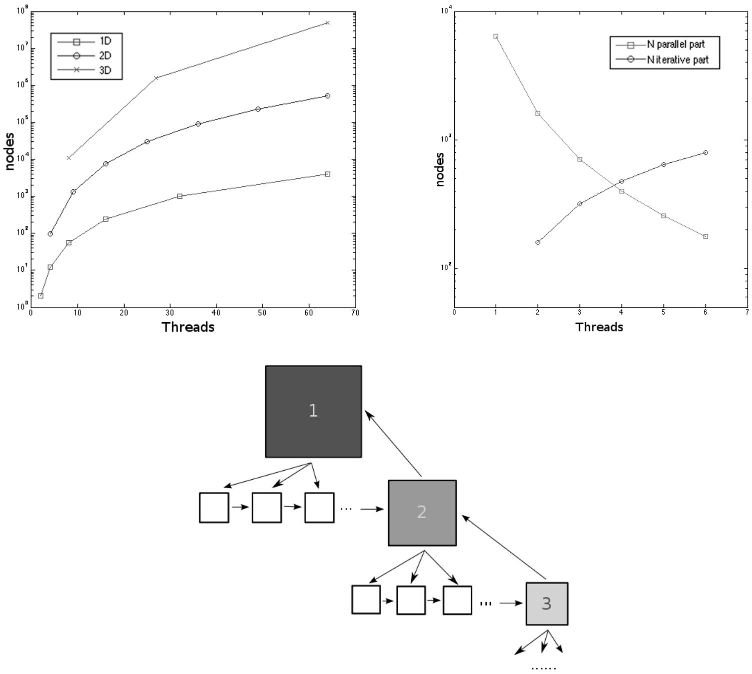


Figure 5: Optimal number of splitting for number of variables in the discretization (left/above) and and number of element (here the 2D case) in the parallel part and in the iterative part (right/above), the optimum is the point of crossing between the two, (as obtained experimentally in Table 6). Recursive structure of the algorithm (right/below) to reduce the original problem (dark gray) to a fixed number of variables sub-problems (white).

problems and a "square 3". Through a reiteration of this idea we arrive to a decomposition in subproblems of dimension desired.

5 Extensions and special cases

In this section we show some non trivial extensions of the technique. We discuss in particular how to adapt the parallelization procedure to the case of a target problem, an obstacle problem and max-min problems, where the special structure of the Hamiltonian requires some cautions and remarks.

5.1 Target problems

An important class of problems to which we want to extend our approach are target problems, where a trajectory is driven to arrive to a *Target set* $\mathcal{T} \subset \Omega$ optimizing a cost functional.

An easy way to modify our algorithm to this case consists in changing the construction procedure for B and C:

$$\left[B'(\alpha)\right]_i := \begin{cases} [B(\alpha)]_i\,, & \text{if } x_i \notin \mathcal{T}, \\ [\mathbb{I}]_i\,, & \text{otherwise;} \end{cases} \quad c'(\alpha)_i := \begin{cases} c(\alpha)_i, & \text{if } x_i \notin \mathcal{T}, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

***Example* 3** (Zermelo's navigation problem)**.** *A well known benchmark in the field is the Zermelo's navigation problem where a dynamic is driven by a force of comparable power with respect to the control. The target is a ball of radius equal to* 0.005 *centred at the origin, the control is in* $A = B(0,1)$*. The other data are:*

$$f(x,a) = a + \begin{pmatrix} 1 - x_2^2 \\ 0 \end{pmatrix}, \quad \Omega = [-1,1]^2, \quad \lambda = 1, \quad l(x,y,a) = 1. \tag{16}$$

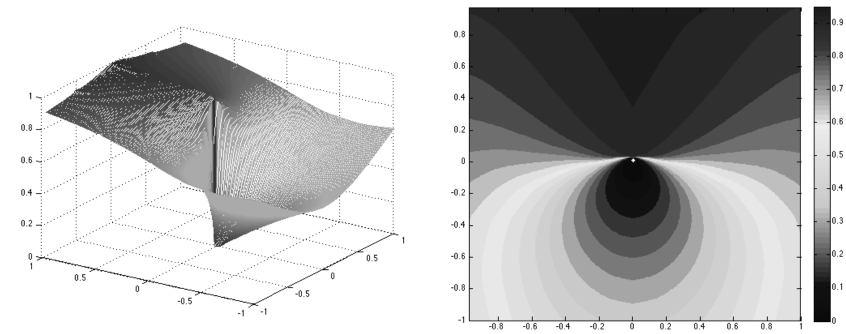


Figure 6: Approximated solution for the Zermelo's navigation problem on a grid of $4e4$ nodes.

In Table 9 we compare the number of threads and performances. We are in presence of characteristics not aligned with the grid, but the efficacy of the method are poorly effected. Convergence is archived with performances comparable to the already described case of the Eikonal Equation.

5.2 Obstacle problem

Various techniques have been proposed to deal with an optimal problem with constraints using the Bellman's approach. In this section we consider an implicit representation of the constraints through a level-set function. Let us to consider the general single obstacle problem

$$\begin{cases} \lambda v(x) + \min\left(H(x, Dv(x)), v(x) - w(x)\right) = 0 & x \in \Omega, \\ v(x) = g(x) & x \in \partial\Omega, \end{cases} \tag{17}$$

Table 9: Zermelo's navigation problem. Comparison of various choices of the number of threads

nodes: 6.4E3	HA		PHA				
threads	t.	it.	t. (par.)	it.	t. (itp.)	t. (total)	SU
4			1.31	4	0.13	5.4	7.02
9			0.5	7	0.7	4.2	9.02
16	37.9	20	3.1E-2	7	1.38	$\boxed{1.53}$	$\boxed{24.8}$
25			2E-2	7	2.7	3.9	9.7
36			1E-2	8	5.19	5.28	7.18

where the Hamiltonian H is of the form discussed in Section 2 and the standard hypothesis about regularity of the terms involved are verified. The distinctive trait of this formulation is about the term $w(x) : \Omega \to \mathbb{R}$, assumed regular, typically stated as the opposite of the signed distance from the boundary of a subset $K \subset \Omega$. The solution of this problem is coincident, where defined, with the solution of the same problem in the space $\Omega \setminus K$. This explains the name of "obstacle problem" (cf. [11]). After a finite dimensional approximation of the problem we arrive to the following modified version of (17)

$$\text{Find } V \in \mathbb{R}^N; \quad \min_{\alpha \in \mathcal{A}^N} \min(B(\alpha)V - c_g(\alpha), V - W) = 0, \tag{18}$$

where the term W is a sampling of the function w on the knot of the discretization grid.

We can see how changing the definition of the matrix B and c, it is possible to reduce the problem to (4). Adding an auxiliary control to the set $\mathcal{A}' := \mathcal{A} \times \{0, 1\}$ and re-defying the matrices B and c as

$$\begin{aligned} \left[B'(\alpha)\right]_i &:= \begin{cases} [B(\alpha)]_i\,, & \text{if } B(\alpha)V - c_g(\alpha) \leq V - W \\ [\mathbb{I}]_i\,, & \text{otherwise;} \end{cases} \\ c'_g(\alpha)_i &:= \begin{cases} c_g(\alpha)_i, & \text{if } B(\alpha)V - c_g(\alpha) \leq V - W \\ W_i, & \text{otherwise;} \end{cases} \end{aligned} \tag{19}$$

$$\text{for } i = 1, ..., N,$$

(where the X_i is the $i-$row if X is a matrix, and the $i-$ element if X is a vector, and $\mathbb{I}$ is the identity matrix), the problem becomes

$$\text{Find } V \in \mathbb{R}^N; \quad \min_{\alpha \in \mathcal{A}'} (B'(\alpha)V - c'_g(\alpha)) = 0. \tag{20}$$

Remark 9. *The verification of Hypotheses (H1-H4) by the numerical scheme associated to the transformation* (19) *is easy. It is in some cases also possible the direct verification of conditions of convergence in the obstacle*

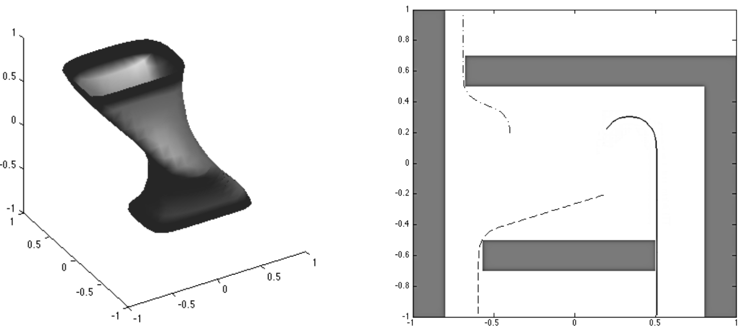


Figure 7: Value function of Dubin car problem (left, free of constraints) and some optimal trajectories in the case with constraints (right).

problem deriving them from the free of constraints case. For example if we have that the matrix $B(\alpha)$ is strictly dominant (i.e. $A_{ij} \leq 0$ for every $j \neq i$, and there exists a $\delta > 0$ such that for every i, $A_{ii} \geq \delta + \sum_{i \neq j} |A_{ij}|$), then the properties of the terms are automatically verified, (i.e. since all $B_i(\alpha)$ are strictly dominant and thus monotone).

***Example* 4** (Dubin car with obstacles)**.** *A classical problem of interest is the optimization of trajectories modelled by*

$$f(x,y,z,a) := \begin{pmatrix} c\cos(\pi z) \\ c\sin(\pi z) \\ a \end{pmatrix}, \quad \lambda := 10^{-6}, \quad l(x,y,z,a) := 1, \quad c \in \mathbb{R};$$

which produces a collection of curves in the plane (x,y) with a constraint in the curvature of the path. Typically this is a simplified model of a car of constant velocity c with a control in the steering wheel.

The value function of the exit problem from the domain $\Omega := (-1,1)^2$, $\mathcal{A} = [-1,1]$ discretized uniformly in 8 points is presented in Figure 7. We consider the same problem with the presence of constraints. This problem can be handled with the technique described above producing the results shown in Figure 7, where there are some optimal trajectories (in the space (x,y)) to exit from $\Omega := (-1,1)^2$ in presence of some constraints.

5.3 Max-min problems

The last, more complex extension of the Howard's problem (4) is about max-min problems of the form

$$\text{Find } V \in \mathbb{R}^N; \quad \max_{\beta \in \mathcal{B}^N} \left(\min_{\alpha \in \mathcal{A}^N} \left(B(\alpha,\beta)V - c(\alpha,\beta) \right) \right) = 0. \tag{21}$$

PHA (Max-min case)

Initialize $V^0 \in \mathbb{R}^N$ α^0 for all $i \in \{1, ..., n+1\}$.
k:=1;

1) Iterate *(Parallel Part)* for every $i = 1, ..., n$ do:
   $s := 0$

   1.i) Find $V_i^s \in \mathbb{R}^n$ solution of $F_i^\beta(V_i^s) = 0$.
   If $s \geq 1$ and $V_i^s = V_i^{s-1}$, then $V_i := V_i^s$, and exit (from inner loop).
   Otherwise go to (1.ii).

   1.ii) $\beta_i^{s+1} := \underset{\alpha \in \mathcal{A}^n}{\arg\min} F_i^\beta(V_i^s) = 0$.
   Set $s := s + 1$ and go to (1.i)

2) Iterate *(Iterative Part)* for $t \geq 0$

   2i) Find $V_{n+1}^t \in \mathbb{R}^h$ solution of $F_{n+1}^\beta(V_{n+1}^t) = 0$.
   If $t \geq 1$ and $V_{n+1}^t = V_{n+1}^{t-1}$, then $V_{n+1} = V_{n+1}^t$, and go to (3).
   Otherwise go to (2ii).

   2ii) $\beta_{n+1}^{t+1} := \underset{\beta_{n+1} \in \mathcal{B}^h}{\arg\min} F_{n+1}^\beta(V_{n+1}) = 0$.
   Set $t := t + 1$ and go to (2i)

3) Compose the solution $V^{k+1} = (V_1, V_2, ..., V_n, V_{n+1})$
   k:=k+1;
   If $V^{k+1} = V^k$ then *exit*, otherwise go to (1).

Table 10: Pseudo-code of PHA for max-min problems.

Those non linear equations arises in various contexts, for example in differential games and in robust control. The convergence of a parallel algorithm for the resolution of such problem is also discussed in [17].

Also in this case, a modified version of the policy iteration algorithm can be shown to be convergent (cf. [8]). Our aim in this subsection is to give some hints for the use of (PHA). The convergence of the algorithm is not guaranteed but only observed experimentally.

Let us introduce the function $F_i^\beta : \mathbb{R}^n \to \mathbb{R}$, for $\beta \in \mathcal{B}^n$ and $i \in \mathcal{I}$ defined by

$$F_i^\beta(V) := \min_{\alpha \in \mathcal{A}^n} (B_i(\alpha, \beta)V - c_i(\alpha, \beta, V) \tag{22}$$

The problem (21), in analogy with the previous case, is equivalent to solve the following system of nonlinear equations

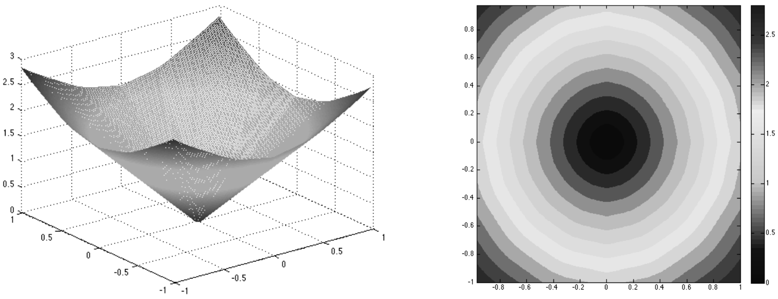


Figure 8: Approximated solution of the pursuit evasion game on a grid of $4e4$ nodes.

$$\begin{cases} \min\limits_{\beta\in\mathcal{B}^k} F_i^{\beta}(V_i) = 0 & i = 1, ..., n \\ \min\limits_{\beta\in\mathcal{B}^h} F_{n+1}^{\beta}(V_{n+1}) = 0 & \end{cases} \tag{23}$$

The (PHA) in the case of a max-min problem is summarized in Table 10.

Remark 10. *It is worth to notice that at every call of the function F^β it is necessary to solve a minimization problem over the set $\mathcal{A}$. This can be performed in an approximated way, using, for instance, the classical Howard's algorithm. This gives to the dimension of this set a big relevance on the performances of our technique. For this reason, if the cardinality of $\mathcal{A}$ (in the case of finite sets) is bigger than $\mathcal{B}$, it is worth to pass to the alternative problem $-max_{\alpha\in\mathcal{A}} \min_{\beta\in\mathcal{B}}(B(\alpha,\beta)V - c(\alpha,\beta))$ (here there are used the Isaacs' conditions [2]) before the resolution, inverting in this way, the role of $\mathcal{A}$ and $\mathcal{B}$ in the resolution.*

***Example* 5** (A pursuit-evasion game)**.** *One of the most known example of max-min problem is the pursuit-evasion game; where two agents have the opposite goal to reduce/postpone the time of capture. The simplest situation is related to a dynamic*

$$f(x, y, z, a, b) := \begin{pmatrix} a_1/2 - b_1 \\ a_2/2 - b_2 \end{pmatrix}$$

where controls are taken in the unit ball $\mathcal{A} = \mathcal{B} = B(0, 1)$ and capture happens when the trajectory is driven to touch the small ball $B(0, \rho)$, ($\rho =$ 0.15, in this case). The passage to a target problem is managed as described previously. In Figure 8 we show the approximated value function of that problem.

6 Conclusions

The main difficulty in the use of the Howard's Algorithm, i.e. the resolution of big linear systems can be prevented using parallel computing. This is important despite the fact that we must accept an important drawback: the double loop procedure illustrated (or multi-loop procedure as sketched in Remark 8) does not permit to archive a superlinear convergence, as in the classical case. We suspect (looking at Figure 2) that such rate is preserved on the iterative part. At every step, the algorithm solves (possibly in parallel) $n-$reduced problems. This brings a significant reduction of the time of computation even without parallel computing.

A substantial point is the choice of the solver for every linear problem. In this paper we opted for the easier (but expensive) choice the exact inversion of the matrix. With the use of an iterative solver, with the due caution about the error introduced, we expect better performances (cf. [1]).

Acknowledgements

This work was supported by the European Union under the 7th Framework Programme FP7-PEOPLE-2010-ITN SADCO, Sensitivity Analysis for Deterministic COntroller design.
The author thanks Hasnaa Zidani of the UMA laboratory of ENSTA for the support in developing the subject and the anonymous referees that help to improve the clearness of this paper.

References

[1] A. Alla, M. Falcone, D. Kalise, An efficient policy iteration algorithm for the solution of dynamic programming equations, SIAM J. Sci. Comput. 37 (1) (2015), 181–200.

[2] M. Bardi, I. Capuzzo-Dolcetta, Optimal control and viscosity solution of Hamilton-Jacobi-Bellman equations, Birkhäuser, Boston Heidelberg, 1997.

[3] M. Bardi, T.E.S. Raghavan, T. Parthasarathy, Stochastic and differential games: Theory and Numerical Methods, Birkhäuser, Boston, 1999.

[4] G. Barles, A. Briani, E. Chasseigne, A Bellman approach for two-domains optimal control problems in $\mathbb{R}^N$, ESAIM Contr. Opim. Ca. 19(3) (2013) 710–739.

[5] G. Barles, A. Briani, E. Chasseigne, A Bellman approach for regional optimal control problems in R^N, SIAM J. Control Optim. 52 (3) (2014) 1712–1744.

[6] R.C. Barnard, P.C. Wolenski, Flow invariance on stratified domains, Set Valued Var. Anal. 21 (2013) 377–403.

[7] R. Bellman, Dynamic programming, Princeton University Press, Princeton NJ, 1957.

[8] O. Bokanowski, S. Maroso, H. Zidani, Some convergence results for Howard's algorithm, SIAM J. Numer. Anal. 47 (4) (2009) 3001–3026.

[9] S. Cacace, E. Cristiani, M. Falcone and A. Picarelli A patchy dynamic programming scheme for a class of Hamilton–Jacobi–Bellman equations. *SIAM Journal on Scientific Computing*, 34(5):2625–2649, 2012.

[10] F. Camilli, M. Falcone, P. Lanucara, A. Seghini, A domain decomposition method for Bellman equations, Contemp. Math. 180 (1994) 477–483.

[11] F. Camilli, P. Loreti, N. Yamada, Systems of convex Hamilton-Jacobi equations with implicit obstacles and the obstacle problem, Comm. Pure Appl. Math. 8 (2009) 1291–1302.

[12] E. Carlini, A. Festa, F. J. Silva, and M.-T. Wolfram, "A Semi-Lagrangian scheme for a modified version of the Hughes' model for pedestrian flow," *Dyn. Games Appl.*, 2016.

[13] Y. Cheng, C-W. Shu, A discontinuous Galerkin finite element method for directly solving the Hamilton-Jacobi equations, J. Comput. Phys. 223 (1) (2007) 398–415.

[14] M.G. Crandall, P.L. Lions, Two approximations of solutions of Hamilton-Jacobi equations, Math. Comp. 43 (167) (1984) 1–19.

[15] M. Falcone, R. Ferretti, Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations, Applied Mathematics series, SIAM, 2013.

[16] M. Falcone, P. Lanucara and A. Seghini, A splitting algorithm for Hamilton-Jacobi-Bellman equations, Applied Numerical Mathematics, 15 (1994) 207–21

[17] M. Falcone, P. Stefani in: A.S. Nowak, K. Szajowski (Eds), Advances in Dynamic Games, Birkhäuser, Boston, 2005, pp. 515–544.

[18] A. Festa. Reconstuction of Independent Sub-Domains for a class of Hamilton Jacobi Equations and Application to Parallel Computing, *ESAIM:M2AN*, 50(4):1223–1240, 2016.

[19] L. Grüne, Numerical stabilization of bilinear control systems, SIAM J. Control Optim. 34 (1996) 2024–2050.

[20] R.A. Howard, Dynamic programming and Markov processes, The MIT Press, Cambridge MA, 1960.

[21] M. Puterman, S.L. Brumelle, On the convergence of policy iteration in stationary dynamic programming, Math. Oper. Res. 4 (1) (1979) 60–69.

[22] L. Qi, J. Sun, A nonsmooth version of Newton's method, Math. Program. 58 (1993) 353–367.

[23] Z. Rao, H. Zidani, in: K. Bredies, C. Clason, K. Kunisch, G. von Winckel (Eds), Control and Optimization with PDE Constraints, Springer, Basel, 2013, pp. 93–116.

[24] M. Santos, J. Rust, Convergence properties of policy iteration, SIAM J. Control Optim. 42 (6) (2004) 2094–2115.

[25] J.A. Sethian, A. Vladimirsky, Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms, SIAM J. Numer. Anal. 41 (1) (2003) 325–363.

[26] H.M. Soner, Optimal control problems with state-space constraints, SIAM J. Control Optim. 24 (1986) 552–562.

[27] P. Souganidis, Approximation schemes for viscosity solutions of Hamilton-Jacobi equations, J. differ. equations 59 (1) (1985) 1–43.

[28] M. Sun, Domain decomposition algorithms for solving Hamilton Jacobi-Bellman equations, Num. Func. Anal. Opt. 14 (1993) 145–166.

[29] Various authors, Parallel Computing Toolbox, User's Guide, The MathWorks, Inc., 2017 `http://www.mathworks.com/help/pdf_doc/distcomp/distcomp.pdf`

[30] H. Zhao, A fast sweeping method for eikonal equations, Math. Comp. 74 (2004) 603–627.

[31] H. Zhao, Parallel implementations of the fast sweeping method, J. Comput. Math. 25 (4) (2007) 421–429.

[32] S.Z. Zhou, W.P. Zhan, A new domain decomposition method for an HJB equation, J. Comput. Appl. Math. 159 (1) (2003) 195–204.