A DYNAMIC DOMAIN DECOMPOSITION
FOR THE EIKONAL-DIFFUSION EQUATION

SIMONE CACACE AND MAURIZIO FALCONE
Dipartimento di Matematica, Sapienza Università di Roma
Piazzale Aldo Moro 5, 00185 Roma, Italy

ABSTRACT. We propose a parallel algorithm for the numerical solution of the eikonal-diffusion equation, by means of a dynamic domain decomposition technique. The new method is an extension of the patchy domain decomposition method presented in [5] for first order Hamilton-Jacobi-Bellman equations. Using the connection with stochastic optimal control theory, the semi-Lagrangian scheme underlying the original method is modified in order to deal with (possibly degenerate) diffusion. We show that under suitable relations between the discretization parameters and the diffusion coefficient, the parallel computation on the proposed dynamic decomposition can be faster than that on a static decomposition. Some numerical tests in dimension two are presented, in order to show the features of the proposed method.

1. Introduction. Domain decomposition techniques have become popular in the numerical solution of partial differential equations, whenever the dimension of the problem is high and/or the discretization requires a huge number of degrees of freedom. In this context, the expression curse of dimensionality properly denotes the difficulties related to the numerical approximation of such problems, both in terms of memory requirements and computational efforts. These drawbacks arise very often in several applications to, e.g., fluid dynamics, electromagnetism, optimal control problems and differential games.

Given a decomposition of a domain in subsets of manageable size and prescribed suitable transmission conditions at the interfaces or overlapping regions between the sub-domains, massive parallel computations can be performed exploiting the computing power of modern CPU and/or GPU clusters. New efficient and accurate parallel algorithms are increasingly demanded, aiming at computing the solution to real-life problems in a reasonable time.

In [5] the authors proposed the patchy domain decomposition method, a parallel algorithm for the solution of deterministic optimal control problems, based on a semi-Lagrangian discretization of the corresponding first order Hamilton-Jacobi-Bellman (HJB) equations. The main idea there is to build a domain decomposition driven by an approximation of the optimal vector field for the underlying control problem. This produces a rather complicated geometrical subdivision of the computational box, but gives the fundamental property that each sub-domain is, up to

2010 Mathematics Subject Classification. Primary: 65N99, 65N55; Secondary: 49L20.

Key words and phrases. Domain decomposition, semi-Lagrangian schemes, Hamilton-Jacobi equations, eikonal-diffusion equation, stochastic optimal control.

This research was supported by the following grants: AFOSR Grant no. FA9550-10-1-0029, ITN-Marie Curie Grant no. 264735-SADCO.
an error, independent from the others. This feature allows for an efficient parallelization, since transmission conditions at the interfaces of the sub-domains can be completely avoided. Despite the non-trivial implementation, the overall performance of the patchy decomposition overcomes that of a static domain decomposition.

Another technique widely used to reduce the computational efforts when computing the solution to first order Hamilton-Jacobi equations, is to exploit the so called causality property. This term refers to the fact that hyperbolic equations propagate information along characteristics at finite speed. Reproducing this feature at the discrete level, one can accelerate significantly the convergence of the schemes that discretize the equations. In this spirit, fast marching methods \[17, 14, 15\] process the nodes of a given mesh in a suitable order that reduces the number of iterations to reach convergence. It turns out that each node converges in a predetermined and finite number of iterations, only one iteration in the most favorable cases. For this reason these methods are also termed single-pass methods. Moreover, the computation is localized, at each iteration, on few nodes that bring the relevant information, allowing a very low memory requirement.

Keeping these ideas in mind, in this paper we revisit the patchy domain decomposition in the context of second order semi-linear equations. We try to extend the main features of the method in order to solve the following boundary value problem for the eikonal-diffusion equation:

\[
\begin{align*}
-\varepsilon \Delta u(x) + c(x)|\nabla u(x)| &= 1 & x \in \Omega \\
u(x) &= 0 & x \in \partial \Omega
\end{align*}
\]

where \(\Omega\) is an open subset of \(\mathbb{R}^n\), \(\varepsilon \geq 0\) is the diffusion coefficient (possibly a function depending on \(x\)) and \(c\) is a real valued positive Lipschitz continuous function. The semi-Lagrangian local solver of the original patchy method is modified in order to deal with diffusion. Indeed, employing a weak notion of characteristics in the context of stochastic differential equations, we can approximate the diffusion by means of discrete time Markov chains. The resulting scheme is able to compute, by construction, the solution of equations with very degenerate diffusions.

In this more general setting the application of the patchy domain decomposition is not trivial. The main obstacle is that the transmission conditions at the interfaces cannot be avoided, due again to the presence of diffusion. A natural question arises: is the patchy domain decomposition still better than a static domain decomposition? We show that the answer to this question depends on a ratio between \(c\) and \(\varepsilon\), a characteristic quantity similar to the Reynolds number for Navier-Stokes equations. Moreover, we show that a remarkable speedup in the computation can be obtained employing, despite the diffusion, the causality property of hyperbolic equations discussed above.

The paper is organized as follows: in Section 2 we briefly recall the patchy domain decomposition method for first order HJB equations. Section 3 is devoted to the extension of the semi-Lagrangian local solver to the case of the eikonal-diffusion equation. In Section 4, we show how to adapt the patchy decomposition method to this more general setting, establishing the conditions on the parameters that guarantee the effectiveness of the dynamic domain decomposition. Finally, in Section 5, we present some numerical tests in dimension two, in order to show the features of the proposed method.

2. Patchy decomposition for first order HJB equations. In this section we review the patchy domain decomposition method, proposed in \[5\] to solve boundary
value problems for first order Hamilton-Jacobi equations of the form
\[
\begin{align*}
\max_{a \in A} \{-f(x,a) \cdot \nabla u(x)\} &= 1 & x \in \Omega \\
u(x) &= 0 & x \in \partial \Omega 
\end{align*}
\]
where \(\Omega \subset \mathbb{R}^n\) is an open set, \(A \subset \mathbb{R}^m\) is a compact set, \(f : \Omega \times A \to \mathbb{R}^n\) is a vector field and \(u : \Omega \to \mathbb{R}\) is a scalar function.

It is well known that equation (1) has a clear interpretation in the context of deterministic optimal control theory. Indeed, it is the Hamilton-Jacobi-Bellman equation associated to a minimum time problem with target \(\partial \Omega\). In this setting, the vector field \(f\) represents the dynamics driving the system, controlled by choosing the parameter \(a\) in the set of admissible controls \(A\). The optimal control problem consists in finding, for each \(x \in \Omega\), a trajectory that minimizes the time of arrival to the target \(\partial \Omega\). Via the celebrated dynamic programming principle, it can be proved that the value function \(u\), i.e. the minimum time of arrival to the target, is the unique viscosity solution to (1). A typical assumption on \(f\) is the Lipschitz continuity in \(x\) uniformly in the control variable. We refer the reader to [1, 2, 3] for further details.

The main advantage of this approach is that, once the value function \(u\) is computed, we can quite easily synthesize an optimal control for the minimum time problem, by taking
\[
a^*(x) = \arg \min_{a \in A} \{f(x,a) \cdot \nabla u(x)\} \quad x \in \Omega.
\]
Note that \(a^*\) is in feedback form, namely it depends only on the the state an not on the time. It follows that, for each \(x \in \Omega\), we can compute the optimal trajectory \(y^*(\cdot)\) starting at \(x\) (i.e. a characteristic curve of the hyperbolic equation (1)) by simply solving the following system of ordinary differential equations:
\[
\begin{align*}
\dot{y}^*(t) &= f(y^*(t), a^*(y^*(t))) , & t > 0 \\
y^*(0) &= x
\end{align*}
\]
By choosing the control set \(A = B_1\) (the unit ball in \(\mathbb{R}^n\)) and the dynamics \(f(x,a) = c(x)a\) (for a given real valued and positive Lipschitz continuous function \(c\)), we recover the classical first order eikonal equation. Indeed, in this case, the maximum in (1) is attained, at each point \(x \in \Omega\), for \(a(x) = -\nabla u(x)/|\nabla u(x)|\).

Here, we prefer to keep the discussion free from technical details, and mainly focus on the ideas behind the construction of the patchy parallel algorithm.

We consider two different discretizations \(G\) and \(G_c\) of \(\Omega\). The grid \(G\) denotes the actual grid on which we want to solve the problem (1) (the fine grid), whereas \(G_c\) is very coarse compared to (and possibly contained in) \(G\).

The first step of the method consists in computing a coarse solution \(u_c\) on \(G_c\), also via parallel computations using a classical (static) domain decomposition technique. Due to the low resolution of the grid, this is a very cheap operation, but gives a first rough approximation \(\hat{u}\) on \(G\) of the actual solution \(u\). Note that \(\hat{u}\) is reconstructed by interpolation of \(u_c\) on \(G\).

Now we employ \(\hat{u}\) to compute a feedback optimal control \(\hat{a}^*\), via an appropriate discrete version of the synthesis procedure (2). We stress that \(\hat{a}^*\) is just an approximation of the actual optimal control \(a^*\), but it is defined on the fine grid \(G\). This is enough to start the construction of the patchy domain decomposition.
We divide the boundary $\partial \Omega$ in a fixed number $N$ of disjoint sub-sets, denoted by $\Gamma_p$, with $p = 1, ..., N$. Then, for each $p$, we compute the sub-domain $\Omega_p \subset \Omega$ as the numerical domain of dependence of $\Gamma_p$ through the optimal dynamics $f^*(\cdot) = f(\cdot, \tilde{a}^*(\cdot))$. Let us clarify this point in a continuous setting. We denote by $\chi_{\Gamma_p}$ the characteristic function of the sub-boundary $\Gamma_p$, and consider the following Cauchy-Dirichlet problem for the advection equation:

$$\begin{cases}
\partial_t \phi_p(x, t) - f(x, \tilde{a}^*(x)) \cdot \nabla \phi_p(x, t) = 0 & (x, t) \in \Omega \times (0, +\infty) \\
\phi_p(x, 0) = 0 & x \in \Omega \\
\phi_p(x, t) = \chi_{\Gamma_p}(x) & (x, t) \in \partial \Omega \times (0, +\infty)
\end{cases}$$

(3)

It is well known that the boundary datum acts as a source of information, that flows (backward) in $\Omega$ along characteristics, according to the drift $f^*$. The limit in time

$$\phi_p^\infty(x) = \lim_{t \to +\infty} \phi_p(x, t)$$

is still a characteristic function, since the hyperbolic equation preserves the properties of $\chi_{\Gamma_p}$ (e.g. the maximum and the singularities). Then, we define the $p$-th patch of our dynamic decomposition as

$$\Omega_p = \{ x \in \Omega : \phi_p^\infty(x) = 1 \} .$$

We remark that each patch is a bundle of characteristics enjoying, by construction, the fundamental property of being invariant with respect to the optimal dynamics, i.e. $f^*(\Omega_p) \subseteq \Omega_p$. This is not completely true, since $f^*$ is built using only the approximate control $\tilde{a}^*$. Moreover, at the discrete level, the projection of the patches on the grid $G$ introduces an additional error, in particular if the dynamics $f^*$ defines very bended characteristic curves.

Figure 1 shows the patchy decomposition for two test dynamics, in dimension two and three respectively. Note that, by construction, the patches do not overlap and the interfaces are sharp.

(a) (b)

**Figure 1.** Patchy domain decompositions for two test dynamics ($B$ denotes a small ball at the origin): (a) $\Omega = \mathbb{R}^2 \setminus B$, $N_p = 4$; (b) $\Omega = \mathbb{R}^3 \setminus B$, $N_p = 8$.

The next step is the parallel computation on the patches. This can be done avoiding completely the transmission conditions, exploiting the invariance property just discussed above. Since this feature is weakened at the discrete level, it can be enforced in the computation by imposing state constraint conditions at the interfaces.
Finally, all the solutions are merged together, producing a solution on the whole domain $\Omega$. As expected, this *patchy solution* is slightly different from that computed using a classic domain decomposition method. Nevertheless, the error is localized at the interfaces between the patches and does not propagate in the interior of the sub-domains, provided that the grid $G_c$ is not too coarse with respect to the fine grid $G$. This is shown in [5] by numerical evidence, but a precise error estimate is still missing.

Despite the small errors, the absence of transmission conditions can give to the parallelization a remarkable speedup. In this respect a key role is played by the relative sizes of the patches. Indeed, we remark that the construction of the patchy decomposition is completely driven by the dynamics of the optimal control problem. Hence, even a subdivision of the boundary $\partial \Omega$ in sub-sets of the same size can produce a highly unbalanced domain decomposition. In these cases the performance of the parallelization is very poor, since the processors associated to the smaller patches complete their job (and remain idle) much earlier than those corresponding to the larger patches. This drawback was pointed out in [5] and can be overcome via a multi-level technique, which is currently under development. The idea is to alternate the construction of the patchy decomposition with the computation of the patchy solution. More precisely, one can start and continue the construction of the patches as long as they have about the same size, obtaining a first level of balanced sub-domains. The solution is then computed on the first level sub-decomposition. The new boundaries (possibly divided again in sub-sets of the same size) are employed to start and build the second level sub-decomposition. Moreover, the values of the first level solution at the corresponding points (correct values due to the *causality property* of hyperbolic equations) are assigned as boundary data for the computation of the second level solution. This procedure is then iterated and terminates when the sub-decompositions cover the whole domain $\Omega$.

3. A semi-Lagrangian scheme for the eikonal-diffusion equation. In this section we present a semi-Lagrangian discretization for the eikonal-diffusion equation. The resulting scheme will be employed as local solver for the extension of the patchy method to this more general setting.

As discussed in the previous section, we can write the eikonal-diffusion equation in a control theory perspective:

\[
\begin{cases}
-\varepsilon \Delta u + \max_{a \in B_1} \{ -c(x)a \cdot \nabla u(x) \} = 1 & x \in \Omega \\
u(x) = 0 & x \in \partial \Omega
\end{cases}
\]

where $\Omega \subset \mathbb{R}^n$ is an open set, $B_1$ is the unit ball in $\mathbb{R}^n$ and $\varepsilon \geq 0$ is the diffusion coefficient, possibly a function depending on $x$. Note that the dynamics $c(x)a$ naturally spans all the directions of the space and this implies that the boundary $\partial \Omega$ can be always reached, even for $\varepsilon = 0$.

In this case, a weak notion of characteristics is still available, interpreting these curves as *generalized trajectories*, namely solutions to the following system of controlled *stochastic* differential equations:

\[
\begin{cases}
dX(t) = c(X(t))\alpha(t)dt + \sqrt{2\varepsilon}dW(t), & t > 0 \\
X(0) = x
\end{cases}
\]

Here $X(t)$ is a progressively measurable process, representing the state of a system evolving in $\Omega$ starting from $x$, the process $\alpha(t)$ is the control applied to the system
at the time $t$ with values in the control set $B_1$ and $W(t)$ is an $n$-dimensional Wiener process.

Similarly to the deterministic case, we obtain that (4) is the Hamilton-Jacobi-Bellman equation associated to the stochastic minimum time problem with target $\partial\Omega$ for the system (5). We refer the interested reader to [7] and the references therein for further explanations and details.

Following [7], the semi-Lagrangian discretization of equation (4) can be performed introducing a time step $h$, interpreting the first order term as a directional derivative of $u$ along the controlled dynamics $c(x)a$ and approximating the Wiener process in the stochastic differential equation (5) by means of discrete time Markov chains. We introduce in the state space a structured grid $G$ with uniform step $\Delta x$ in each coordinate direction and nodes $x_i$ for $i = 1, \ldots, N$. This leads to the following scheme, which is a nonlinear system in the form of a fixed point operator $S: \mathbb{R}^N \to \mathbb{R}^N$:

$$
\begin{cases}
    x_{i,k}^{a,s} = x_i + hc(x_i)a + s\sqrt{2\varepsilon h}e_k & s = \pm 1, k = 1, \ldots, n \\
    u(x_i) = \min_{a \in B_1} \left\{ \frac{1}{2n} \sum_{k=1}^{n} \sum_{s=\pm 1} u(x_{i,k}^{a,s}) + h \right\} =: S_i[u] & x_i \in G \cap \Omega \\
    u(x_i) = 0 & x_i \in G \cap \partial\Omega
\end{cases}
$$

(6)

where $e_k$ denotes the $k$-th element of the canonical base of $\mathbb{R}^n$. The evaluation of the min operator is done by direct comparison, discretizing the unit ball $B_1$ with a fixed number of points (but other more accurate methods can be applied, see [9]). As usual in semi-Lagrangian approximations, the $2n$ points $x_{i,k}^{a,s}$ do not lie in general on the grid and the values of $u$ at these points need to be reconstructed by interpolation. Moreover, it may happen that a point $x_{i,k}^{a,s}$ falls out of the domain $\Omega$.

In this situation the time step $h$ is locally reduced so that the condition $x_{i,k}^{a,s} \in \partial\Omega$ holds.

We conclude this section observing that the term $c(x)a$ in (4) can be replaced by more general dynamics of the form $f(x,a)$, without any modification. Moreover, the proposed semi-Lagrangian scheme can handle by construction very degenerate diffusions (see the numerical experiments in Section 5). Indeed, at points where the diffusion coefficient $\varepsilon = 0$, we naturally recover the semi-Lagrangian scheme presented in [5] for first order Hamilton-Jacobi-Bellman equations. This is not the case for other types of discretization, e.g. finite elements, where the degeneracy in the diffusion produces instabilities that need to be treated by stabilization techniques. This is due to the fact that the variational forms associated to the equations under consideration suffer a lack of coercivity. We refer the interested reader to [11] for details and insights on this topic.

4. Patchy decomposition for the eikonal-diffusion equation. In this section we aim at extending the patchy decomposition method to the case of the eikonal-diffusion equation presented in the previous section.

The main idea is really simple: we first compute the patchy domain decomposition in the hyperbolic case with $\varepsilon = 0$. Then we solve in parallel the full equation with $\varepsilon \neq 0$ using that patchy decomposition.

At a first look this attempt could seem meaningless. Indeed, due to the diffusion, information spreads instantaneously from the boundary to the whole domain and then it cannot be confined in independent sub-domains, as for the first order case.
This implies that, in order to compute the correct solution in this more general setting, we cannot replace the transmissions between the patches with state constraint boundary conditions, as discussed in Section 2. So we lose the main advantage of the patchy decomposition compared to an arbitrary and static decomposition.

Nevertheless, we show through the next sections that the patchy method with transmission conditions can still be competitive if we combine two different features, described in the following sub-sections.

4.1. **The upwind diffusion ball condition.** Here we present the first ingredient for the extension of the patchy method. In particular we show that, under suitable relations between the parameters, the discretization of equation (4) behaves in a sense more like an hyperbolic than an elliptic equation.

To simplify the presentation we consider the two dimensional case \( n = 2 \). We choose a time step \( h \) such that

\[
h c_{\text{max}} + \sqrt{2 \varepsilon h} < \Delta x,
\]

where \( c_{\text{max}} \) denotes the maximum of the speed \( c \) on \( \Omega \). In this way, for each node \( x_i \), the 4 points \( x_{i,k}^{a,s} \) (\( k = 1,2 \), \( s = \pm 1 \)) in the scheme (6) fall in the first neighboring cells of \( x_i \). The reason of this choice is twofold. First, we do not want to get a too low accuracy in the approximation of the solutions of the controlled stochastic differential equation (5). Second and more important, we want to keep the stencil of the scheme strictly localized, in order to reduce the computational efforts.

Now, for each node \( x_i \) and each control \( a \in B_1 \), consider the half space defined by \( \pi = \{ x \in \mathbb{R}^2 : a \cdot (x - x_i) \geq 0 \} \) and also the ball \( B \) of radius \( \sqrt{2 \varepsilon h} \) centered at \( x_i + hc(x_i)a \), enclosing the four points \( x_{i,k}^{a,s} \) for \( k = 1,2 \) and \( s = \pm 1 \) (see Figure 2).

\[\text{(a) (b)}\]

**Figure 2.** The upwind diffusion ball condition: (a) \( B \subseteq \pi \), the interpolating (green) grid nodes are upwind w.r.t. the vector field \( c(x_i)a \); (b) \( B \not\subseteq \pi \), some downwind (red) grid nodes enter in the computation.

The key argument of our construction is that, if the diffusion ball \( B \) is contained in the half space \( \pi \), then the value \( u(x_i) \) is computed using only grid nodes that are upwind with respect to the vector field \( c(x_i)a \), as shown in Figure 2a. If we combine this property with a suitable order for processing the grid nodes (see next
subsection) we can accelerate the convergence of the scheme in the same spirit of fast-marching methods, reducing significantly the number of iterations. On the contrary, if part of $B$ crosses $\pi$, then also downwind nodes are employed in the local reconstruction of $u(x_i)$, as shown in Figure 2b. It means that something is flowing in directions opposite to the vector field $c(x_i)a$, so that $u(x_i)$ cannot be computed in a single pass fashion and the number of iterations to reach convergence increases. This is clearly expected, since the diffusion uniformly spreads information in all the directions. But, we can try to force the above geometric condition on the diffusion ball by imposing that the time step $h$ satisfies also

$$hc_{\text{min}} - \sqrt{2\varepsilon h} > 0, \quad (8)$$

where $c_{\text{min}} > 0$ denotes the minimum of the speed $c$ on $\Omega$. We define $\omega = c_{\text{min}}/2\varepsilon$, $\gamma_c = c_{\text{max}}/c_{\text{min}}$ and we choose $h = \frac{1}{1+\gamma_c} \frac{\Delta x}{c_{\text{min}}}$. It is easy to see that conditions (7)-(8) are not always compatible. They can be simultaneously satisfied if and only if the following \textit{upwind diffusion ball condition} holds:

$$\frac{1}{\omega} < \frac{\Delta x}{1 + \gamma_c}. \quad (9)$$

Consequently, if (9) fails, only one of the two conditions (7)-(8) can be satisfied, and we will always assume (7) for accuracy reasons.

For a fixed and small mesh size $\Delta x$, condition (9) can be clearly fulfilled if the controlled dynamics \textit{dominates} diffusion, i.e. for $\omega >> 1$. In this regime the semi-Lagrangian scheme exhibits the same peculiarities of the hyperbolic case. Note that $\omega$ is a characteristic quantity of the problem. It resembles the Reynolds number for Navier-Stokes equations and the case $\omega >> 1$ is of great interest in the applications [11]. Equivalently, we can fix $\omega$ and choose the mesh size $\Delta x$. According to (9), if the mesh is quite coarse, the approximation “looks like” that of a first order eikonal-equation. Note that this effect is also related to the degree of inhomogeneity $\gamma_c \geq 1$, in the sense that the larger is $\gamma_c$ the coarser should be the mesh. On the other hand, for a sufficiently fine mesh, diffusion starts to get noticed and this hyperbolic behavior disappears as $\Delta x \to 0$. This is expected, due to the consistency of the semi-Lagrangian scheme with the considered eikonal-diffusion equation.

4.2. The role of causality. Under the regime given by condition (9), we have a chance to get a good performance of the patchy method. To this end, the second crucial point to extend the patchy decomposition to this more general setting is the \textit{causality property} of hyperbolic equations outlined in the Introduction.

Starting from the work by Tsitsiklis [17] and then Sethian [13], a lot of research has been devoted to find an implicit order of the nodes of a grid that allows one to compute the solution in just one or very few iterations, the so called \textit{single-pass property}. With this idea in mind, the celebrated fast marching method has been proposed to solve the eikonal equation. It can be proved that, in this special case, the right order corresponds to process the nodes by ascending values, progressively accepting as correct (and then removing from the computation) the node with the minimal value. This translates in computing the solution following its level sets, namely propagating information along its gradient curves. Since for the eikonal equation gradient curves coincide with characteristics, we get the correct solution.

We mention here also the fast sweeping method [16, 18] and some of its generalizations [4], based on another technique that, in a weaker sense, still exploits the causality property. The grid nodes are alternatively swept in a pre-determined
number of directions according to the dimension of the problem, until convergence is reached. This makes the method iterative by construction, but it allows to compute the solution to very general equations of Hamilton-Jacobi type. The number of iterations to reach convergence is strongly dependent on the problem and the mesh structure. Nevertheless, it can be proved that only $2^n$ sweeps are needed to compute the solution to the eikonal equation in dimension $n$.

In Section 2, we denoted by $\tilde{u}$ the interpolation on the fine grid $G$ of the coarse solution $u_c$ on the coarse grid $G_c$. Our strategy is then to sort the grid nodes of $G$ in a fast marching fashion, according to the increasing values of $\tilde{u}$. This can be accomplished with some additional but cheap time consumption, using some state-of-the-art sorting algorithm, embedded in the pre-computation step.

4.3. The patchy algorithm. Here we summarize all the implementation steps of the new patchy method for the eikonal-diffusion equation. We refer the reader to [5] for the discrete version of the synthesis procedure (2) and the actual construction of the patchy decomposition discussed in Section 2.

**Initialization:**
- Build two grids $G_c$ and $G$ such that $G_c << G$.
- Fix tolerances $\tau, \tau_c$ and the number of patches $N_P$.
- Build the initial guess $u^0_c$ on $G_c$ equal to $0$ on $G \cap \partial \Omega$ and $+\infty$ (i.e. a very big value) otherwise.

**Pre-computation:**
- Starting from $u^0_c$, compute $u_c$ on $G_c$ iterating the scheme (6) with $\epsilon \equiv 0$, until convergence (up to $\tau_c$).
- Build $\tilde{u}$ on $G$ by interpolation of $u_c$.
- Compute $\tilde{a}^*$ on $G$ using $\tilde{u}$.
- Divide the boundary nodes of $G$ in $N_P$ sub-sets.
- For $p = 1, ..., N_P$, use $\tilde{a}^*$ to compute on $G$ the patch $\Omega_p$.
- For $p = 1, ..., N_P$, sort the nodes of $\Omega_p$ according to the increasing values of $\tilde{u}$ (to exploit causality).

**Computation:**
- For $p = 1, ..., N_P$, starting from $u_p^0 = \tilde{u}$, compute $u_p$ on $\Omega_p$ iterating the scheme (6), i.e. $u_{p}^{k+1/2} = S[u_{p}^{k}]$, until convergence (up to $\tau$).
  At the end of each iteration, update $u_{p}^{k+1/2}$ at the nodes where $\Omega_p$ intersects other patches, using the transmission condition
  $$u_{p}^{k+1}(x_i) = \min\{u_{p}^{k+1/2}(x_i), u_{q}^{k+1/2}(x_i)\} \text{ for all } q \neq p \text{ such that } \emptyset \neq \Omega_p \cap \Omega_q \ni x_i$$
- Build the solution $u$ on $G$, merging all the $N_P$ solutions $u_p$ on $\Omega_p$.

Note that all the steps of the method can be parallelized. In particular, the solution $u_c$ on the coarse grid $G_c$ can be computed by means of a standard domain decomposition technique. In the following section we will compare this dynamic domain decomposition to the classical static domain decomposition. The interested reader can find in [12] a good introduction to this topic and in [10, 8] some static domain decomposition methods for first order Hamilton-Jacobi equations.

We conclude this section by remarking that, in the case of the eikonal-diffusion equation, the issue of balancing the size of the patches can no longer be addressed via the multi-level approach discussed in Section 2 for first order equations. Indeed, due to the presence of the diffusion, we are not guaranteed that the values of the solution
at the boundary of the current level decomposition are correct. Some information could flow back in the future from patches that have not yet been built. In principle, we can continue the construction of the decomposition, postponing the computation of the solution. To this end we need more processors, exactly \( N_P N_L \), where \( N_L \) is the number of levels, which is clearly bounded but a priori unknown. In the case of only \( N_P \) available processors, an alternative could be to find an iterative method to solve the following optimization problem: build an initial subdivision of the boundary \( \partial \Omega \) such that the corresponding patches have about the same sizes. An interesting question is to understand if the additional computational cost of this optimization procedure is compensated or not by the balancing in size of the patches.

5. Numerical experiments. In this section, we present some numerical experiments in dimension two. In particular, we show that the semi-Lagrangian scheme (6) is able to compute the solution to the eikonal-diffusion equation for very degenerate diffusion coefficients \( \varepsilon \). Moreover, under the regime given by condition (9), we give a first confirmation of the effectiveness of the extension of the patchy method. In all the following tests, we set \( \Omega = [-1,1]^2 \) and we use 16 points to discretize the control set \( B_1 \). Finally, we choose a diffusion coefficient \( \varepsilon \) possibly depending on \( x \).

More precisely, we consider a function of the form \( \varepsilon(x) = \bar{\varepsilon} \chi_S(x) \), where \( \bar{\varepsilon} \geq 0 \) is a real number and \( \chi_S \) denotes the characteristic function of a generic subset \( S \) of \( \Omega \). It follows that equation (4) is elliptic in \( S \) and hyperbolic outside.

Figure 3 shows the level sets of the solution for the choice \( \bar{\varepsilon} = 0.1 \), \( S = [0,0.75] \times [-0.6,0.6] \) and \( c(x) \equiv 1 \). We can immediately distinguish the two different behaviors, by looking at the smoothness of the solution in \( S \) and \( \Omega \setminus S \). In particular, the sharp corners on the diagonals, corresponding to the shocks in the hyperbolic case, are completely smoothed out in \( S \).

![Figure 3](image)

**Figure 3.** Level sets of the solution to a degenerate problem: the diffusion is present only inside the (red) rectangle, affecting the smoothness of the solution.

We now present a more suggestive and complicated example, namely we consider a slightly more general controlled dynamics \( f \) of the form

\[
f(x, a) = \frac{1}{1 + |x|^2} \left( R_\theta \frac{x}{|x|} + \frac{\eta}{2} a \right),
\]

where \( \eta \in \{0,1\} \) is a fixed switch parameter to activate/deactivate the control and \( R_\theta \) is a fixed counter-clock-wise rotation with \( \theta < \frac{\pi}{2} \). This term represents a whirling drift from the origin to the boundary, introducing in the equation both
inhomogeneity and strong anisotropy. Note that, in this case, the particular form of the drift guarantees the reachability of the whole boundary, but we can play with the control \( a \) to solve the problem in minimum time. Here we apply the diffusion on the whole domain, namely we choose \( S = \Omega \). In Figure 4 we compare the level sets of the solutions and the corresponding optimal dynamics for three different cases.

![Figure 4](image-url)

**Figure 4.** An inhomogeneous and anisotropic problem, level sets of the solution and optimal dynamics: (a) diffusion without control \((\bar{\varepsilon} = 0.1, \eta = 0)\), (b) diffusion with control \((\bar{\varepsilon} = 0.1, \eta = 1)\), (c) control without diffusion \((\bar{\varepsilon} = 0, \eta = 1)\).

The first case (see Figure 4a) represents a pure advection-diffusion equation, with \( \bar{\varepsilon} = 0.1 \) and no control, i.e. \( \eta = 0 \). In the second case (see Figure 4b) the control is active, i.e. \( \eta = 1 \), and we can clearly see how it produces a resistance to the drift that allows to reach the boundary in a smaller time. This is much more evident in the third case (see Figure 4c) where the control is still active and the diffusion is switched off, i.e. \( \bar{\varepsilon} = 0 \).

Finally, we also consider the case where the diffusion is present only in the upper half of the square, i.e. \( S = \{x = (x_1, x_2) \in \Omega : x_2 > 0\} \). Figure 5 shows the level sets of the solution and the corresponding optimal dynamics.

We proceed with a comparison between the computation on a standard domain decomposition and that on the patchy decomposition presented in Section 4. We choose a constant diffusion coefficient \( \varepsilon \geq 0 \) and the constant speed \( c(x) \equiv 1 \). We do not report results related to the full implementation of the parallel algorithm, which is currently under development, but we mainly focus on a comparison in terms of iterations to reach convergence. An extensive analysis of the new patchy method, applied to more general second order semi-linear equations, is the subject of an upcoming work [6]. Here the aim is to convince the reader that, in order to obtain a remarkable speedup in the computation, the fundamental ingredients are
the causality property and the upwind diffusion ball condition discussed in Section 4. To this end, we build the patchy domain decomposition starting from a natural subdivision of the boundary $\partial \Omega$ in four parts, namely the sides of the square $\Omega$. Figure 6 shows the resulting dynamic decomposition compared to an arbitrary and static one.

![Figure 6](image)

**Figure 6.** Domain decompositions in 4 sub-domains: (a) the patchy dynamic decomposition, (b) a static decomposition.

Note that, in this case, the sub-domains of the two decompositions (the four triangles and the four squares respectively) have exactly the same size. We sort the grid nodes of the patchy decomposition according to the increasing values of the pre-computed coarse solution interpolated on the fine grid. As already remarked, for the eikonal equation this corresponds to the optimal order to exploit causality in a fast marching fashion. We choose the space step $\Delta x = 0.02$, i.e. a $100 \times 100$ grid, so that the upwind diffusion ball condition (9) reads (note that in this case $\omega = c_{\text{min}}/2\varepsilon = 1/2\varepsilon$ and $\gamma_c = 1$)

$$
\varepsilon < \frac{1}{2} \times 10^{-2} =: \tau_{\varepsilon}
$$

In Table 1 we report, for different values of the diffusion coefficient $\varepsilon$, the number of iterations to reach convergence for a standard domain decomposition method (DD) and the patchy domain decomposition (PDD) method. Moreover, we measure the speedup in the computation by means of the ratio of the corresponding iterations.
Table 1. DD vs PDD in terms of iterations to reach convergence and speedups. Values in bold show the advantage of PDD in the regime given by the upwind diffusion ball condition.

| $\varepsilon$ | 0     | $10^{-5}$ | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | $2 \times 10^{-2}$ | $5 \times 10^{-2}$ | $10^{-1}$ |
|---------------|-------|-----------|-----------|-----------|-----------|-------------------|-------------------|----------|
| DD            | 50    | 54        | 57        | 63        | 165       | 297               | 888               | 1922     |
| PDD           | 1     | 10        | 15        | 22        | 112       | 245               | 838               | 1865     |
| Speed-up      | 50    | 5.4       | 3.8       | 2.9       | 1.4       | 1.21              | 1.05              | 1.03     |

These results confirm that, despite the pre-computation step and the transmission conditions, we can still have a remarkable advantage in using the patchy decomposition for second order equations. Indeed, the upwind diffusion ball condition sounds and we clearly see that, for $\varepsilon$ below the threshold $\varepsilon$, the speedup increases. On the contrary, above $\varepsilon$, the speedup starts converging to 1 as $\varepsilon$ increases. This means that the hyperbolic behavior of the approximation is lost, replaced by diffusion.

We conclude this section with some comments on the apparently strange speedup factor 50 in the first column of Table 1. For $\varepsilon = 0$, we are considering the case of the hyperbolic first order eikonal equation, whose optimal vector field is shown in Figure 7. We see that characteristics are straight lines, moving from the boundary until they intercept the diagonals of the square. This is the most explicit example of causality we can imagine, which gives immediately the correct order for processing the grid nodes, i.e. exactly the one we employ in sorting the patches. Since we always use also Gauss-Seidel-like updates in the scheme (6), it easily follows that the PDD method converges in just one iteration. Conversely, in the case of an arbitrary decomposition, we have no such information and we choose a fixed processing order, e.g. from left to right and from up to down. Keeping in mind this order, we refer again to Figure 6b. It is easy to deduce that, at the end of the first iteration, only the blue patch and half of the red and green patches (i.e. everything above the diagonal $x_1 = x_2$) will contain correct values of the solution. For the remaining part of the domain, only the first neighboring nodes of the boundary are correctly updated. The same holds in the following iterations, as long as information goes.
back until the middle of the 100 × 100 grid. This gives precisely 50 iterations for the DD method.

In this respect, we remark that the fast sweeping method discussed in Section 4 can be employed without any (even coarse) knowledge of the solution, namely without any pre-computation. It is enough to alternately sweep the grid nodes in all the possible directions (two for each dimension) until convergence. In particular, we can compute the solution to the eikonal equation in just four iterations. This add-on can make the DD method comparable with the PDD method, even faster if we also consider the PDD’s pre-computation step.

Nevertheless, for more general equations, e.g. including inhomogeneity and anisotropy, the number of sweeps becomes unpredictable, strongly dependent on the considered problem. Indeed, it is known that the fast sweeping method works well for equations with quite straight characteristics, whereas fast marching methods are suitable for general inhomogeneous problems with mild anisotropy. In principle, one could drop completely the construction of dynamic decomposition and decide to use a static decomposition combined with a hybrid method, possibly employing some coarse information on the characteristics via a pre-computation step.

However, the patchy method can be still competitive for problems with degenerate diffusion. Comparing the level sets of the solutions and, in particular, the corresponding optimal dynamics shown in Figure 4c and Figure 5, we see that some degree of invariance can be still present in the degenerate case (see the region contained between the two shocks in the lower left part of the domain). This invariance could be exploited by removing the transmission conditions at some part of the interfaces between the patches, partially getting again the main advantage of the patchy method for first order equations.

Acknowledgments. The authors would like to thank Emiliano Cristiani for the fruitful discussions they had with him during the preparation of this work, and also the anonymous referee whose suggestions and comments contributed to the improvement of the paper.

REFERENCES

[1] M. Bardi and I. Capuzzo Dolcetta, Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations, Birkhäuser, 1997.
[2] M. Bardi and M. Falcone, An approximation scheme for the minimum time function, SIAM Journal of Control and Optimization, 28 (1990), 950–965.
[3] M. Bardi and M. Falcone, Discrete approximation of the minimal time function for systems with regular optimal trajectories, in Analysis and Optimization of Systems (eds. A. Bensoussan, J.-L. Lions), Lecture Notes in Control and Information Sciences, 144, Springer-Verlag, 1990, 103–112.
[4] S. Cacace, E. Cristiani and M. Falcone, Two semi-lagrangian fast methods for Hamilton-Jacobi-Bellman equations, in System Modeling and Optimization (eds. C. Pötzsche, et al.), 443, Springer, 2014, 74–84.
[5] 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 (2012), 2625–2649.
[6] S. Cacace and M. Falcone, A dynamic domain decomposition for a class of second order semi-linear equations, in preparation, arXiv:1502.01629.
[7] F. Camilli and M. Falcone, An approximation scheme for the optimal control of diffusion processes, Mathematical Modelling and Numerical Analysis, 29 (1995), 97–122.
[8] F. Camilli, M. Falcone, P. Lanucara and A. Seghini, A domain decomposition method for Bellman equations, in Domain Decomposition methods in Scientific and Engineering Computing (eds. D. E. Keyes and J. Xu), Contemporary Mathematics, 180, AMS, 1994, 477–483.
[9] M. Falcone and R. Ferretti, *Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi Equations*, SIAM, 2014.

[10] M. Falcone, P. Lanucara and A. Seghini, A splitting algorithm for Hamilton-Jacobi-Bellman equations, *Applied Numerical Mathematics*, 15 (1994), 207–218.

[11] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Oxford University Press, 1999.

[12] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Springer, 2008.

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

[14] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 1999.

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

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

[17] J. N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, *IEEE Trans. Autom. Control*, 40 (1995), 1528–1538.

[18] H. Zhao, A fast sweeping method for Eikonal equations, *Math. Comp.*, 74 (2005), 603–627.

Received September 2014; revised February 2015.

E-mail address: cacace@mat.uniroma1.it
E-mail address: falcone@mat.uniroma1.it