Parallelizing differential evolution for 3D medical image registration

Michel Salomon[1]†, Guy-René Perrin† and Fabrice Heitz‡

†*LSIIT-ICPS*, ‡*LSIIT-AIDI*
Université Louis Pasteur, Strasbourg, UPRES-A CNRS 7005
Pôle API, Boulevard Sébastien Brant, F-67400 Illkirch

7th June 2002

Abstract

A common framework for 3D image registration consists in minimizing a cost (or energy) function that expresses the pixel or voxel similarity of the images to be aligned. Standard cost functions, based on voxel similarity measures, are highly non-linear, non-convex, exhibit many local minima and thus yield hard optimization problems. Local, deterministic optimization algorithms are known to be very sensitive to local minima. Global optimization methods (like simulated annealing or evolutionay algorithms), yield better, often close to the optimal solutions, but are time consuming.

In this paper we consider the parallelization of a general purpose global optimization algorithm based on random sampling and evolutionary principles: the differential evolution algorithm. The inherent parallelism of evolutionary algorithms is used to devise a data-parallel implementation of differential evolution. The performances of the parallel version is assessed on a 3D medical image registration problem. Although the proposed parallelization scheme induces an algorithm semantically different from the sequential one, the comparison of the results at convergence shows that both algorithms end in a solution of similar quality. Beside yielding accurate registrations, the parallel differential evolution exhibits fast convergence and a speedup almost growing linear with respect to the number of processors.

Keywords. Differential Evolution, Data-Parallel Programming Model, 3D Medical Image Registration

1 Introduction

Many tasks in computer vision and image analysis have been expressed as global optimization problems. The general issue is to find the global minimum of an objective (also called cost or energy) function which describes the interaction between the different variables modeling the image features in a given problem [8].

In particular, a common framework for image registration consists in minimizing a cost function that expresses the pixel or voxel similarity of the images to be aligned [14]. The purpose of image registration (also called image matching) is to geometrically align one image (the floating or source image) to another (the reference or target image) so that voxels (or pixels) representing the same physical structure may be superimposed. Image registration is an important preliminary step in many computer vision tasks involving for instance, the reconstruction of 3D information from 2D views [4], the measurement of motion or deformation fields in temporal image sequences [12, 13] or the fusion of multimodal images [22]. 3D medical imaging with its wide variety of sensors is probably one of the first application field, as are remote sensing, military imaging or multisensor robot vision. Applications of medical image registration range from computer-assisted surgery to the analysis of functional images used for example to follow the evolution of diseases or to assess the efficiency of a treatment.

Standard cost functions, based on voxel similarity measures [14, 23], are highly non-linear, non-convex and exhibit many local minima. Like NP-complete problems, they yield challenging optimization problems, with large and irregular search spaces, and thus require computationally demanding global

[1]Supported by the Pluri-Formations Program "Analyse et Synthèse Multi-Images", Ministry of National Education, Research and Technology, France (Grant 1997/2000).
email: salomon@icps.u-strasbg.fr

optimization algorithms to compute close to optimal solutions [14]. Less cpu intensive deterministic algorithms may be used instead, but they are sensitive to local minima and require a good initial guess. Global optimization algorithms include random sampling algorithms, such as simulated annealing [10], global heuristic search approaches such as tabu search [6, 7] and evolutionary approaches [1] such as genetic algorithms [9, 3] or differential evolution [19].

In this paper we consider differential evolution as an appealing candidate for general purpose global optimization [19, 18]. We show that this algorithm is particularly suited for the registration of medical images using similarity measures. To register images in a reasonable amount of time we propose a parallelization of the algorithm. Classical parallelizations of evolutionary algorithms consists in distributing the population of candidate solutions maintained by the algorithms, making it evolve in parallel. In our approach we have one potential solution per processor, thereby we define a fully synchronous data-parallel algorithm. The parallel algorithm has been applied successfully to register 3D MR images of the brain, exhibiting a nearly linear speedup. Moreover the semantical difference between the parallel and the sequential algorithms has no effect on the quality of the registration

The paper is organized as follows. In the next section, we present the similarity-based registration algorithm considered in our application. In section 3 we give a brief review on evolutionary algorithms, followed by a detailed description of differential evolution. A general presentation of how parallelizing an evolutionary algorithm and the proposed parallelization, in a data-parallel programming model [15], are described in the section 4. The fifth section is devoted to the experimental assessment of the performances of the proposed parallel algorithm. A validation on a significant number of 3D image registration problems, with known ground truth, shows the efficiency and robustness of the data-parallel approach.

2 Medical images registration using similarity measures

A complete review of standard registration techniques has been proposed by Brown [2]. A classification, in the framework of medical image analysis may be found in [21]. Similarity measure-based approaches rely on the minimization of cost functions that express the pixel or voxel similarity of the images to be aligned. They have been proposed for both single and multimodal image registration. Standard similarity metrics are related to least squares estimation or to the maximization of the correlation function. Other similarity measures, based on first or second order image statistics such as mean, variance, entropy measures or texture moments have also been described.

Pixel (or voxel) similarity metric-based registration consists in estimating the parameters Θ of the rigid transformation T_Θ minimizing a cost function
$C\left(I_{ref}(.),\, I_{reg}(T_\Theta(.))\right)$, that expresses the similarity between the single or multimodal image pair:

$$\Theta_{\min} = \arg\min_{\Theta} \left[C\left(I(.),\, J(T_\Theta(.))\right)\right] , \tag{1}$$

where (for 3D images):

$$\Theta = (t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)^T \tag{2}$$

is a vector containing the 3D translation parameters (t_x, t_y, t_z) with respect to the X, Y and Z axis and the Euler rotation angles $(\theta_x, \theta_y, \theta_z)$. $I(.)$ represents the reference image and $J(.)$ the floating image (to be registered).

A classical similarity measure, widely used for the registration of single modal images is the quadratic similarity measure [2]. This similarity metric assumes that the two registered images differ only by an additive Gaussian noise, leading to the following cost function:

$$C\left(I(.),\, J(T_\Theta(.))\right) \;=\; \sum_s [I(s) - J(T_\Theta(s))]^2 \; . \tag{3}$$

where s designates the pixel (or voxel) coordinates.

The quadratic cost function (3) is highly non-linear and has multiple local minima [2]. In most image registration methods based on the minimization of a cost function, deterministic optimization algorithms are applied. They are known to be very sensitive to local minima, unless they are initialized close to the optimal solution. In [14], Nikou *et al.* introduce a hierarchical simulated annealing algorithm in order to

obtain close to optimal solutions. To reduce the time complexity, the stochastic optimization is applied on a multigrid sequence of images of increasing resolution.

To obtain an additional gain in time complexity, compatible with a use of 3D registration in clinical routine, we consider in this work the parallelization of the global optimization step. Simulated annealing algorithms [14] are ill-suited for efficient parallel implementations and we thus consider a general purpose evolutionary algorithm (differential evolution) which is intrinsically parallel [19]. Contrary to simulated annealing, the algorithm shows low sensitivity to the choice of tuning parameters and the speedup of the parallel version grows linearly with respect to the number of processors.

3 Optimization by differential evolution

3.1 A short overview on evolutionary algorithms

Early in the sixties, several researchers simultaneously suggested to simulate the process of natural evolution. This early work led to the development of a new class of robust and efficient search algorithms, known as evolutionary algorithms. The most popular ones are certainly the genetic algorithms, which were developed by Holland [9] and De Jong [3]. One generally distinguishes two other classes among evolutionary algorithms: evolutionary programming [5] and evolution strategies [16, 17]. A survey of these three mainstreams of evolutionary algorithms can be found in Bäck's book [1]. More recently, Storn and Price [19] used concepts from genetic algorithms and evolution strategies to introduce differential evolution.

The 1996 ICEC contest on the minimization of multivariable real functions [19] has shown the excellent performances (in terms of robustness and efficiency) of differential evolution. The differential evolution algorithm ranked 3rd in the ICEC contest and was distinguished as the best evolutionary algorithms.

Evolutionary algorithms are based on the paradigm of a population of individuals evolving by means of stochastic operators, in order to favor emerging individuals of better fitness. Each individual in the population represents a potential solution to the optimization problem. The fitness value of an individual gives a qualitative information reflecting his behavior in the environment (related to the function to be optimized) and is obtained by transformation of the objective function value. The first population is usually initialized at random or may use some specific knowledge on the problem to be solved. Natural evolution is simulated by three stochastic operators, applied to the population: crossover, mutation and selection. Crossover (or recombination) performs exchange of information across several individuals and mutation introduces new information by disrupting them. These two operators ensure the exploration of the search space, whereas selection guides the search by favoring the reproduction of the best adapted individuals.

Let $P(t)$ represent the parent population in generation t, containing μ individuals a_j, $j \in \{1, ..., \mu\}$, and let Φ denote the fitness function. Following the description given above, an evolutionary algorithm may be outlined as following:

$t = 0$ /* **Generation counter** */
Initialization$(P(t))$ /* $P(t) = \{a_1(t), ..., a_\mu(t)\}$ */
Evaluate$(P(t))$ /* $\{\Phi(a_1(t)), ..., \Phi(a_\mu(t))\}$ */
While *termination criterion not met* **do**
 $t = t + 1$
 $P'(t) =$ **Crossover**$(P(t-1))$
 Mutate$(P'(t))$
 Evaluate$(P'(t))$
 $P(t) =$ **Select**$(P'(t) \cup Q(t))$, $Q(t) \in \{\emptyset, P(t-1)\}$
End while

Depending on the class of evolutionary algorithms, selection operates on the offspring population $P'(t)$ or takes into account the parent population too. A maximum number of generated populations is typically used as a termination criterion.

3.2 Differential evolution

The key idea in differential evolution is to use the difference between two randomly chosen individuals in the population for disrupting a third one. Differential evolution uses a real-valued representation, working directly on the parameter vector to be optimized. Therefore for the medical image registration problem described in section 2, we take $a = \Theta$, and the fitness function is simply defined as the cost function C, resulting in $\Phi(a) = C(I(.), J(T_\Theta(.)))$. Storn and al. [19] distinguish two operators in the reproduction step: first an operator generating a new population by disrupting $P(t)$, followed by a two point crossover of each parent with its disrupted form. In the following we merge these two operators in a single one.

Let us assume that the population contains μ individuals, each individual containing n components $a_j = (a_{j,1}, a_{j,2}, ..., a_{j,n}), j \in \{1, ..., \mu\}$. Denoting a'_j the offspring resulting from reproduction of a_j, we can give the formal description of the reproduction operator ($\chi_1, \chi_2 \in \{1, ..., n\}$ are the two crossover positions):

$$\chi_1 = (\chi \bmod n) + 1 \ , \ \chi_2 = ((\chi + L - 1) \bmod n) + 1 \tag{4}$$

where χ and L are two integers uniformly drawn in set $\{1, ..., n\}$, for each individual. χ defines the first position of the crossover, whereas L is the number of parameters to be exchanged. Integer L is chosen so that $P(L \geq \nu) = (p_c)^{\nu-1}$, where $\nu > 0$ and $p_c \in [0, 1]$ is the crossover probability given by the user. The new individual $a'_j = \left(a'_{j,i}\right)_{i \in \{1,...,n\}}$ is then formed according to the rule presented below:

$$a'_{j,i} = \begin{cases} a_{j,i} + \lambda \cdot (a_{min,i} - a_{j,i}) + F \cdot (a_{S,i} - a_{T,i}) \\ \qquad \text{if } (\chi_1 \leq i \leq \chi_2) \vee (\chi_2 < \chi_1 \wedge (1 \leq i \leq \chi_2 \vee \chi_1 \leq i \leq n)) \\ \\ a_{j,i} \ \text{ otherwise} \end{cases} \tag{5}$$

with a_{min} denoting the current minimum; $S, T \in \{1, ..., \mu\}$ satisfying $(S \neq T) \neq j$ and $F, \lambda \in [0.1, 1]$ are factors controlling the amplification of the differential vectors. Usually to reduce the number of parameters λ is set equal to F.

The new population is then obtained using a deterministic selection corresponding to a binary tournament between each parent and its respective offspring. More formally the selection operator can be describe for a minimization task by:

$$\text{Select}(P' \cup P) = \{s(a_j, a'_j), j \in \{1, ..., \mu\}\} \tag{6}$$

where $s(a_j, a'_j) = (1 - p_s(a'_j)) \cdot a_j + p_s(a'_j) \cdot a'_j$ and $p_s(a'_j) = 1_{\mathbb{R}^+}(\Phi(a_j) - \Phi(a'_j))$.

This kind of selection is called μ-elitist. Resulting from several experiments with test functions and real problems, rules have been proposed [18] to guide the choice of the different parameters (μ, F, λ and p_c) or for initializing the first population.

4 Parallelization

Parallelizing an algorithm can be motivated by several reasons, the main ones are: reducing the computation time by dividing equally the computation cost over the whole processors; be able to solve larger problems than possible in sequential; last but not least, designing new optimization strategies based on sequential methods.

4.1 How parallelizing an evolutionary algorithm

Evolutionary algorithms are intrinsically parallel, since as in nature, individuals evolve simultaneously. An evolutionary algorithm can be parallelized on three levels [20, 11]: the fitness evaluation level, the population level, and the individual level. The first level will not be discussed as it is a problem dependent parallelization, whereas the two other levels are general parallelization.

Population-based parallelization consists in dividing the population in as much subpopulations as processors, each processor running the evolutionary algorithm on his own subpopulation. This kind of

parallelization is said coarse grained. The subpopulations are usually called "island" (or demes). They can be connected following a defined neighborhood so that migration of some individuals from a subpopulation to another is possible, replacing the worst individuals in the target subpopulation: usually it is the individual having the best fitness which migrates. The interacting scheme is usually called "connected island model". The background idea for allowing migration of individuals is to insure diversity over the subpopulations. In the scheme without interactions, the only communication takes place at the end in order to select the best individual among all subpopulations. Unfortunately, the lack of communications can result in a subpopulation exploring a bad region of the search space, ignoring more promising regions explored by other processors.

Introducing the parallelization on the individual level consists in distributing the population so that each individual is assigned to one virtual processor, resulting in a data-parallel implementation with a finer granularity. In this scheme, the topology of the processors network is very important, since it induces the degree of isolation across the population and in this way the diversity of the whole population. The diffusion of individuals is done by the selection and reproduction operators as they are done locally on each processor, considering a neighborhood structure. Consequently the neighborhood is the major component in this parallel algorithm: the four or eight nearest neighbors of an individual form the most typical neighborhoods used. Local interactions between neighbor processors prevent premature convergence, since the diffusion of the individuals across the population is very slow, reducing the probability of seeing ermerge a superindividual.

Parallelization has been mainly studied on genetic algorithms, as they are the most popular evolutionary algorithms and the ones having the widest range of application. Our contribution is to propose a parallelization for differential evolution, a new evolutionary algorithm.

4.2 Parallelizing differential evolution: a data-parallel implementation

In this work, we propose an individual based parallelization for the differential evolution algorithm. In fact a study of the sequential algorithm showed that a population with a small size was enough to give good results. Therefore a coarse grained parallelization is irrelevant: the subpopulations would have been too small.

Consider a grid of virtual processors, having a size 4×4. We distribute the population $P(t)$ on this grid in order to have one individual per virtual processor. Thus parallelizing differential evolution consists in parallelizing the different steps of the algorithm, namely: initialization, evaluation, reproduction and selection.

Clearly initialization and evaluation can be done in parallel without any problems. Furthermore, each individual $a'_j, j \in \{1, ..., \mu\}$ ($\mu = 4 \times 4$), resulting from reproduction will be located on the same virtual processor as a_j. Consequently, as the selection operator consists in a comparison of this two individuals for each value of j, no communications are needed and the parallelization is direct. Finally, we have to parallelize the reproduction operator. Equation (5) shows that to form a new individual a'_j, apart from a_j three other individuals from the population are requested:

- the individual of minimum cost in the current population $P(t)$, denoted a_{min};
- two individuals a_S and a_T, randomly chosen in the population, mutually different and also different from a_j ($(S \neq T) \neq j$).

As shown by figure 1(a), broadcasting a_{min} across the grid seems to result in irregular communications. But broadcast should be optimized by the compiler, since it's a basic operation in data-parallelism. To draw in parallel the two individuals a_S and a_T, we define around each virtual processor a neighborhood corresponding to the eight-nearest neighbors of the considered processor. a_S and a_T are then randomly chosen in this neighborhood and communicated to the processor where the corresponding a_j is located, as presented in figure 1(b). The choice of these two individuals is the same for all the processors, yielding a regular communication pattern. Obviously introducing a neighborhood to choose a_S and a_T gives a parallel algorithm semantically different from the sequential one.

The main question can be stated in the following word: "Has the semantical modification introduced in the parallel algorithm any effect on its results?". The next section will answer to it, showing clearly the relevance of our approach.

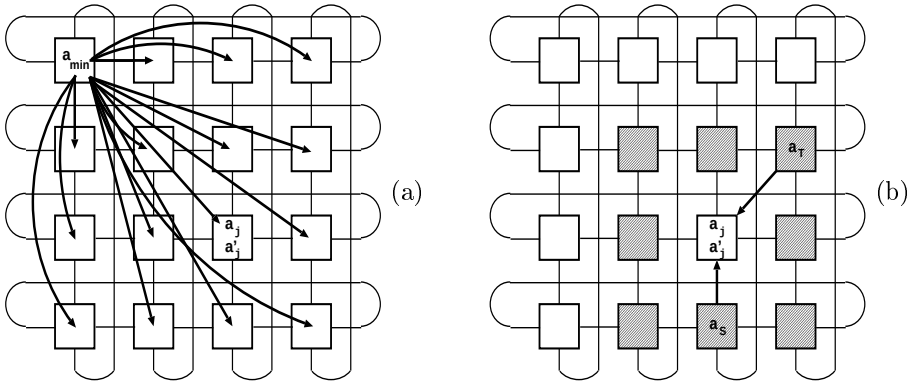

Figure 1: Communications induced by the reproduction operator on a grid of 4×4 virtual processors. (a) Broadcast of a_{min} and (b) transmission of a_S and a_T (the neighborhood is in grey).

5 Computational results

5.1 Experimentation framework

We combine our data-parallel approach with a multigrid optimization of the similarity measure. The parallel algorithm is applied on a sequence of multiresolution 3D grids, from coarse to fine resolution, until the full 3D image resolution is reached. In practice, the cost function (3) is calculated by successively considering a voxel out of 81, 27, 9, 3 and finally every voxel in the MRI images. The same number of populations is generated on the four first resolutions and only one population on the final resolution. Multigrid matching is usually motivated by the significant computational gain obtained in the registration. As noticed by several authors [8], multigrid algorithms are also far less sensitive to local minima in the cost function than single resolution optimization schemes. It has indeed been conjectured that multigrid analysis may, to a certain extent, smooth the "landscape" of the objective function to minimize. This yields fast convergence towards good solutions [8].

Parallel differential evolution has been implemented on the Origin2000 from Louis Pasteur University (32 processors, 8 giga-byte memory), using the data-parallel language High Performance Fortran (HPF). The Origin2000 is a virtually shared memory computer, but physically, it's a distributed memory computer. Moreover the nodes are connected with an hypercube topology network.

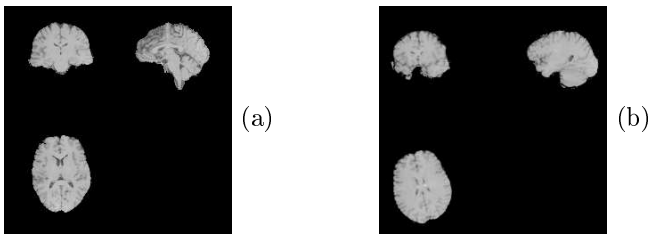

Figure 2: MRI/MRI registration. (a) Reference image and (b) one of the 20 target images (multiplanar - sagittal, coronal, transversal - view).

To evaluate the parallel algorithm on a representative set of 3D MRI/MRI registration problems, we have applied twenty randomly generated rigid transformations on a reference MRI provided by the Institut de Physique Biologique (Strasbourg University Hospital). Statistics on the registration errors were computed on this set of 20 different registrations problems, involving translation parameters between -20 and $+20$ voxels and rotations between -20 and $+20$ degrees. Let us notice that large rotations are generally difficult to handle, leading to objective function landscapes with many local minima. Figure 2 presents a typical example of reference image and the corresponding transformed volume MRI.

5.2 Performances of the approach

Suitable tuning parameters for differential evolution were found in some preliminary experiments ($F = \lambda = 0.525, p_c = 0.8$), and as termination criterion we use a maximum number of generated populations that we set to $t_{max} = 57$.

5.2.1 Quality of the registrations

Figure 3 presents the average evolution of the minimum cost, for the twenty registration problems, during the optimization process. The solid curve corresponds to the parallel algorithm executed on eight physical processors, the dashed line depicts the sequential version. Clearly, the sequential and the parallel algorithms have the same average behavior, resulting in a final parameter vector with similar cost, and the semantical modification introduced by our approach is without any effect.

To assess the accuracy of the registration obtained by the parallel version, we compare the estimated registration parameters to the true parameters. It can be seen from statistics on the registration errors, gained for the set of twenty images (Table (1)), that our algorithm achieves subvoxel accuracy. This accuracy is comparable to the results obtained by the fast simulated annealing algorithm, which has been validated on real-world cases in [14].

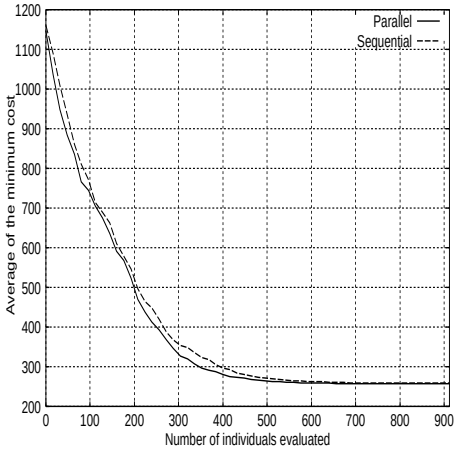

Figure 3: Curves giving the average evolution of the minimum cost.

Table 1: The average and the standard deviation of the registration errors.
Single Modal Registration (3D) MRI/MRI

Parameters	Δt_x	voxels	0.20 ± 0.15	$\Delta \theta_x$	degrees	0.03 ± 0.02
	Δt_y		0.15 ± 0.10	$\Delta \theta_y$		0.02 ± 0.02
	Δt_z		0.16 ± 0.09	$\Delta \theta_z$		0.03 ± 0.03

The good results of our data-parallel implementation can be explained by the choice made for the neighborhood during the reproduction, since it is this operator that differentiates the parallel version from the sequential one. For the sequential version a population of nine individuals is sufficient to permit the convergence to a final parameter vector giving an accurate registration. As we choose the eight nearest neighbors to form the neighborhood, we see precisely that during the reproduction on a virtual processor a subpopulation of nine individuals is considered.

5.2.2 Performances

The parallel algorithm has been executed on 2, 4, 8 and 16 physical processors. As reference on one processor we take the sequential execution time. The left curve from figure 4 shows the decrease of the execution time when passing from the sequential version to the parallel versions. We see that differential evolution takes approximately 19 minutes in sequential, and the parallel execution times decrease regularly, taking on 16 processors less than 1 minutes 30 seconds. The corresponding speedup curve 4(b) reflects a slowly decrease when increasing the processor number: for 2 processors the speedup is linear, then the curve moves away more and more from ideal linear speedup. Nevertheless, the lost in performances is reasonable, since the speedup on 16 processors is about 13.5.

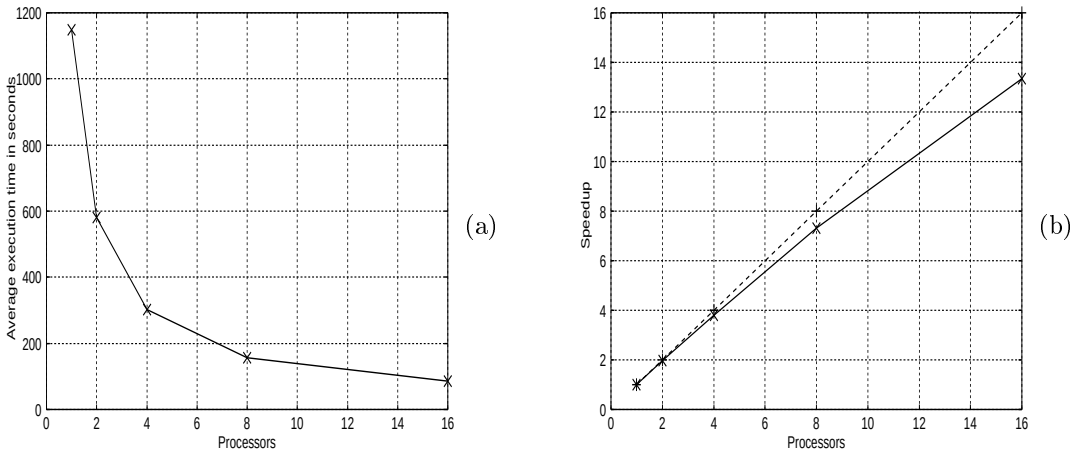

Figure 4: (a) Average run times and (b) corresponding speedup values.

The little loss in performances observed when increasing the processor number finds its origin in the increasing number of irregular communications between physical processors, needed to broadcast the current minimum a_{min}, even if this operation is optimized by the compiler. This problem could become cumbersome for applications requiring a larger population than the one considered here and that could be executed on more processors. We think that there is certainly an upper bound on the population size, beyond which an island based parallelization should be prefered, and conversely.

Another potential problem is the fact that the crossover length L is randomly chosen by each virtual processor. Therefore the crossover has not the same cost across the processor grid, so when the individual's contain a great deal of parameters, a load balancing problem may appear. However, for the single modal registration application considered in this paper, we were not faced with this problem.

6 Conclusions and further developments

We have outlined a data-parallel implementation for differential evolution that produces subvoxel registrations with a quasi-linear speedup following the processor number. The experiments show clearly that the semantic modification induced by our parallelization scheme has no effect on the algorithm behavior when compared with the sequential version. Furthermore, differential evolution is a very flexible optimization algorithm, since it is only based on the cost associated to a parameter vector, so that we can easily adapt it to another cost function. This point is very interesting in image registration, namely for the multimodal case which involves a more complex cost function, that does not have a calculable derivative.

After this first evaluation on single modal image registration, we plan to apply our method on multimodal image registration. A further challenging work is the study of the parallel differential evolution ability to perform 3D deformable matching [13], characterized by a dramatic increase of the number of parameters to be optimized. These works are currently under development. As far as the parallelization is concerned, a comparison between the languages HPF and OpenMP is also in progress.

References

[1] T. Bäck. *Evolutionary algorithms in theory and practice*. Oxford University Press, 1996.

[2] L.G. Brown. A survey of image registration techniques. *ACM Computing Survey*, 24(4):325–376, 1992.

[3] K. De Jong. *An analysis of the behavior of a class of genetic adaptative systems*. PhD thesis, University of Michigan, 1975.

[4] O. Faugeras. *Three-dimensional computer vision : a geometric viewpoint*. MIT Press, 1993.

[5] L.J. Fogel. Toward inductive inference automata. In *Proc. of the Int. Federation for Information Processing Congress Conf.*, pages 395–399, Munich, 1962.

[6] F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–206, Summer 1989.

[7] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2(1):4–32, Winter 1990.

[8] F. Heitz, P. Perez, and P. Bouthemy. Multiscale minimization of global energy functions in some visual recovery problems. *Comput. Vis. Graphics Image Proc. Image Understanding*, 59(1):125–134, 1994.

[9] J.H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[10] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[11] K. Miettinen, M.M. Mäkelä, P. Neittaanmaki, and J. Périaux, editors. *Evolutionary algorithms in engineering and computer science*. Wiley and Sons, 1999.

[12] A. Mitiche and P. Bouthemy. Computation and analysis of image motion : a synopsis of current problems and methods. *Int. J. Computer Vision*, 19(1):29–55, 1996.

[13] O. Musse, F. Heitz, and J.-P. Armspach. 3d deformable image matching using multiscale minimization of global energy functions. *IEEE Conf. on Computer Vision and Pattern Recognition*, 1999.

[14] C. Nikou, F. Heitz, J.-P. Armspach, I.-J. Namer, and D. Grucker. Registration of mr/mr and mr/spect brain images by fast stochastic optimization of robust voxel similarity measures. *Neuroimage*, 8:30–43, 1998.

[15] G.-R. Perrin and A. Darte, editors. *The data parallel programming model*, volume 1132 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[16] I. Rechenberg. *Evolutionsstrategie : optimierung technischer systeme nach prinzipien des biologischen evolution*. Frommann-Holzboog Verlag, Stuttgart, 1973.

[17] H.-P. Schwefel. *Numerische optimierung von computer-modellen mittels der evolutionsstrategies*, volume 26 of *Interdisciplinary Systems Research*. Birkhäuser, Basel, 1977.

[18] R. Storn. On the usage of differential evolution for function optimization. In *NAFIPS*, pages 519–523, Berkeley, 1996.

[19] R. Storn and K. Price. Minimizing the real functions of the icec'96 contest by differential evolution. In *Int. Conf. on Evolutionary Computation*, Nagoya, Japan, 1996.

[20] M. Tomassini. *Parallel and distributed evolutionary algorithms*, chapter 7, pages 113–133. In Miettinen et al. [11], 1999.

[21] P.A. van den Elsen, E.J.D. Paul, and M.A. Viergever. Medical image matching - a review with classification. *IEEE Engineering in Medicine and Biology*, 12(4):26–39, 1993.

[22] S.T.C. Wong, R.C. Knowlton, R.A. Hawkins, and K.D. Laxer. Multimodal image fusion for noninvasive epilepsy surgery planning. *IEEE Computer Graphics and Applications*, 16(1):30–38, 1996.

[23] R.P. Woods, J.C. Maziotta, and S.R. Cherry. Mri-pet registration with automated algorithm. *Journal of Comput. Assist. Tomography*, 17(4):536–546, 1993.