Properties and numerical results of a parallel algorithm for global optimization problems

M. Gaviano, D. Lera

Dipartimento di Matematica e Informatica
University of Cagliari, Cagliari, Italy

Problem: Global Optimization

$$\text{find} \quad x^* \in S, \quad \text{such that} \quad f(x^*) \leq f(x), \quad \forall x \in S,$$

where $f : S \longrightarrow \mathbf{R}$ is a function defined on a set $S \subseteq \mathbf{R}^n$.

While there exist very efficient algorithms that find a local minimum of $f(x)$, the search of a global minimum can be a very hard problem.

Nemirovsky and Yudin [10], and Vavasis [12] have proved, under suitable assumptions, that the computational complexity of the global optimization problem is exponential.

Many procedures introduced in the literature to solve the global optimization employ local minimum algorithms; these mostly constructs, by starting from a point x_0, a sequence $\{x_j\}$, with $f(x_j) > f(x_{j+1})$, that converges to a local minimum x^*.

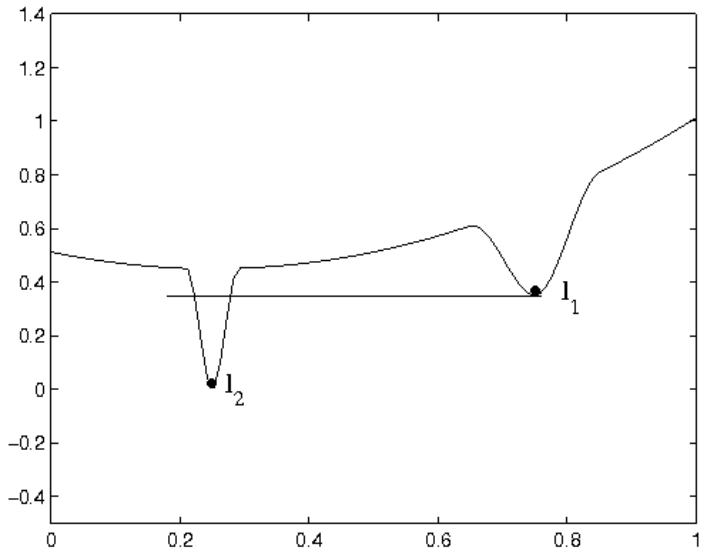
Clearly if we start from different points in S, we can expect to find all the local minima of $f(\cdot)$ and then its global minimum.

The researchers have proposed different strategies for selecting the starting points of the local searches; see the papers by Boender *et alias* [1], Cetin *et alias* [2], Desai *et alias* [3], Hedar *et alias*, [6] Levy *et alias*[8], Lucidi *et alias* [9], Oblow [11].

The main point to tackle is to avoid of finding the same local minima.

One can choose the starting point x_0 of a new local search such that $f(x_0)$ is less than the value of the last local minimum found. In such a way the local searches guarantees that a new local minimum point with function value less than the previous ones can be found.

On the other end, by proceeding in this way, we reduce the size of the region that could take us to the global minimum. That is depicted in the figure



For the global optimization problem we consider the following

Assumption 1.

i) $f(\cdot)$ has m local minimum points $l_i,\ i = 1, ..., m$ and $f(l_i) > f(l_{i+1})$;

ii) $meas(S) = 1$,

with meas(S) denoting the measure of S.

Algorithm scheme (Glob). Choose x_0 uniformly on S; $i \leftarrow 1$; $j \leftarrow 1$; $(x_1, fx_1) \leftarrow local\_search(x_0)$; $l_i = x_1$; $fl_i = fx_1$;

repeat
  choose x_0 uniformly on S; $j \leftarrow j + 1$;
  if $f(x_0) \leq fl_i$ or $(f(x_0) > fl_i$ and $rand(1) < d_i)$
    $(x_1, fx_1) \leftarrow local\_search(x_0)$;
    if $fx_1 < fl_i$
      $i \leftarrow i + 1$; $l_i \leftarrow x_1$; $fl_i \leftarrow fx_1$;
    end if
  end if
until a stop rule is met;
end

The parameters d_i are probability values, $d_i \in [0, 1]$. $rand(1)$ denotes a generator of random numbers in the interval $[0, 1]$. $local\_search(x_0)$ is any procedure that starting from a point x_0 returns a local minimum l_i of $f(x)$ and its function value.

We assume that the optimization problem satisfies all the conditions required to make the procedure $local\_search(x_0)$ convergent.

Proposition

Let assumption 1 hold and consider a run of algorithm Glob. Then the probability that l_i is a global minimum of the optimization problem tends to one as $j \longrightarrow \infty$.

Definition

- $A_{0,j} \equiv \{x \in S \mid$ starting from x, $local\_search(\cdot)$ returns local minimum $l_j\}$;
- $A_{i,j} \equiv \{x \in S \mid f(x) \leq f(l_i)$; starting from x, $local\_search(\cdot)$ returns local minimum $l_j\}$;
- $p_{0,j} = meas(A_{0,j})$;
- $p_{i,j} = meas(A_{i,j})$.

We have

$$\sum_{i=1}^{m} p_{0,i} = meas(S) = 1.$$

Assumption 2.

- algorithm *Glob* runs an infinite number of iterations;
- the number of function evaluations required by *local_search* is $k = constant$.

Theorem 1.

The average number of function evaluations so that algorithm *Glob*, having found a local minimum l_i, finds any new one is given by

$$evals_1(d_i) = f_i \frac{1}{Prob_{i,*}}, \quad i = 1, ..., m-1, \tag{2}$$

with

$$\begin{aligned} Prob_{i,*} &= \sum_{j=i+1}^{m} p_{i,j} + d_i \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right), \\ f_i &= k \sum_{j=i+1}^{m} p_{i,j} + k d_i (1 - \sum_{j=i+1}^{m} p_{i,j}) + (1 - d_i)(1 - \sum_{j=i+1}^{m} p_{i,j}). \end{aligned}$$

In [4], the following was stated and investigated

Problem 1.

Consider the optimization and let the values k, $p_{0,j}$ and $p_{i,j}$ be given. Find value d_i^* such that

$$evals_1(d_i^*) = \min_{d_i} \ evals_1(d_i).$$

It comes out that the derivative sign of $evals_1(d_i)$ is constant in $[0, 1]$ and is greater than or equal to zero if

$$k \geq \frac{(\sum_{j=i+1}^{m} p_{0,j})(1 - \sum_{j=i+1}^{m} p_{i,j})}{(\sum_{j=i+1}^{m} p_{i,j})(1 - \sum_{j=i+1}^{m} p_{0,j})}. \tag{3}$$

The latter links the probability $p_{i,j}$ with the number k of function evaluations performed at each local search in order to choose the most convenient value of d_i: if the condition is met, $d_i = 0$ is suitable to be chosen otherwise $d_i = 1$. That is,

$$d_i = \begin{cases} 0 & \text{if } k > (p_2 \cdot (1 - p_3))/(p_3 \cdot (1 - p_2)), \\ 1 & \text{otherwise,} \end{cases} \tag{4}$$

with

$$p_2 = \sum_{j=i+1}^{m} p_{0,j}, \quad p_3 = \sum_{j=i+1}^{m} p_{i,j},$$

In real problems usually the values $p_{0,j}$ and $p_{i,j}$ are not known; hence the choice of probabilities $d_1, d_2, ..., d_{m-1}$ in the optimization of the function in problem 1 cannot be calculated exactly.
Algorithm Glob has been completed with d_i given by 4 and p_2, p_3 and k approximated as follows

$$\begin{aligned} p_2 &= 1/(\text{number of searches carried out }); \\ p_3 &= 1/(\text{number of iterations already carried out}); \\ k &= \text{mean number of function evaluations carried out} \\ &\quad \text{in each local search.} \end{aligned} \tag{6}$$

.

New results

We consider the overall minimization process from the starting point until the global minimum has being found.

We calculate the computational cost of this process.

A comparison with the local analysis is done.

.

Notation and Definition

- $t_i \equiv$ the probability that having found the local minimum l_i, in a subsequent iteration no new local minimum is detected;
- $tr(i_1, ..., i_p) \equiv$ the set (trajectory) of p local minimum points $l_{i_1}, ..., l_{i_p}$ found in a run of algorithm *Glob*;

Notation and Definition

- $Prob_{i,j}(d_i) \equiv$ the probability that algorithm *Glob* having found the local minimum l_i can find the local minimum l_j in a subsequent iteration;
- $Prob_{i,j}^{(\infty)}(d_i) \equiv$ the probability that algorithm *Glob* having found the local minimum l_i can find l_j assuming that an infinity number of iterations are carried out;
- $Prob_{tr}^{(n)}(d_{i_1}, ..., d_{i_{p-1}}) \equiv$ the probability that algorithm *Glob* constructs the trajectory $tr = (i_1, ..., i_p)$ in n iterations.

We have two lemmas.

Lemma 1.

$$
\begin{aligned}
t_i &= \sum_{j=1}^{i} p_{0,j} + \Big(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j}\Big)(1-d_i), \\
Prob_{i,j}(d_i) &= p_{i,j} + d_i(p_{0,j} - p_{i,j}), \\
Prob^{(n)}_{tr}(d_1,\ldots,d_{p-1}) &= p_{0,i_1}\cdot(p_{i_1,i_2} + (p_{0,i_2} - p_{i_1,i_2})d_{i_1})\cdot\ldots\cdot \\
&\cdot \quad (p_{i_{p-i},i_p} + (p_{0,i_p} - p_{i_{p-1},i_p})d_{i_{p-1}})\cdot \\
&\cdot \sum_{\substack{j_1,\ldots,j_{p-1}=0 \\ j_1+\ldots+j_{p-1}\le n-p}}^{n-p} t_{i_1}^{j_1}\cdot t_{i_2}^{j_2}\cdot\ldots\cdot t_{i_{p-1}}^{j_{p-1}}.
\end{aligned}
$$

with $n > p$ and $tr = (i_1, ..., i_p)$, $i_p = m$.

Lemma 2.

$$Prob_{i,j}^{(\infty)}(d_i) = \frac{p_{i,j} + d_i(p_{0,j} - p_{i,j})}{\sum_{l=i+1}^{m}(p_{i,l} + d_i(p_{0,l} - p_{i,l}))},$$

$$Prob_{(i_1,\ldots,i_p)}^{(\infty)}(d_1,\ldots,d_{p-1}), = p_{0,i_1} \cdot Prob_{i_1,i_2}^{(\infty)} \cdot \ldots Prob_{i_{p-1},i_p}^{(\infty)},$$

with $i_p = i_m$.

Theorem 2.

The average number of function evaluations so that algorithm *Glob* finds the global minimum point is given by

$$evals_2(d_1, ..., d_{m-1}) = \sum_{tr(\cdot) \in T} Prob_{tr}^{(\infty)}(\cdot)(k + f_{i_1} \frac{1}{Prob_{i_1,i_2}}, ... + f_{i_{p-1}} \frac{1}{Prob_{i_{p-1},i_p}})$$

where

$$f_i = k \sum_{l=i+1}^{m} p_{i,l} + k d_i (1 - \sum_{l=i+1}^{m} p_{i,l}) + (1 - d_i)(1 - \sum_{l=i+1}^{m} p_{i,l}).$$

and T denotes the set of all feasible trajectories $tr(i_1, ..., i_p)$ whose last local minimum point is a global one.

Problem 2.

Let consider problem (1) and values k, $p_{0,j}$ and $p_{i,j}$ be given. Find values d_i^*, $i = 1, ..., m-1$, so that

$$evals_2(d_1^*, ..., d_{m-1}^*) = \min_{d_1,...,d_{m-1}} evals_2(d_1, ..., d_{m-1}). \tag{7}$$

with $(d_1, ..., d_{m-1}) \in I \equiv \{x \in R^n \mid 0 \leq x_j \leq 1, j = 1..., n\}$.

We prove

Lemma 3.

$evals_2(d_1, ..., d_{m-1})$ attains its minimum at a vertex of the simplex I.

Remarks

- $evals_1(d_i)$ is part of the expression of $evals_2(d_1, ..., d_{m-1})$.
- A counterexample shows that the optimal values d_i for $evals_1(d_i)$ are not necessarily optimal for $evals_2(d_1, ..., d_{m-1})$.

. Counterexample

Let $m = 3$. We have four trajectories and

$$T = \{(3), (1,2,3), (1,3), (2,3)\}.$$

$evals_2(d_1, d_2) =$

$$\begin{aligned}
&+ \quad k + p_{0,1}\frac{(k-1)(p_{1,2}+p_{1,3})+1+d_1(k-1)(1-p_{1,2}-p_{1,3})}{p_{1,2}+p_{1,3}+d_1(p_{0,2}-p_{1,2}+p_{0,3}-p_{1,3})} \\
&+ \quad p_{0,1}\frac{p_{1,2}+d_1(p_{0,2}-p_{1,2})}{p_{1,2}+p_{1,3}+d_1(p_{0,2}-p_{1,2}+p_{0,3}-p_{1,3})} \cdot \frac{(k-1)p_{2,3}+1+d_2(k-1)(1-p_{2,3})}{p_{2,3}+d_2(p_{0,3}-p_{2,3})} \\
&+ \quad p_{0,2}\left(\frac{(k-1)p_{2,3}+1+d_2(k-1)(1-p_{2,3})}{p_{2,3}+d_2(p_{0,3}-p_{2,3})}\right).
\end{aligned}$$

$$
\begin{aligned}
evals_1\_d_1(d_1) &= \frac{(k-1)(p_{1,2}+p_{1,3})+1+d_1(k-1)(1-p_{1,2}-p_{1,3})}{p_{1,2}+p_{1,3}+d_1(p_{0,2}-p_{1,2}+p_{0,3}-p_{1,3})} \\
evals_1\_d_2(d_2) &= \frac{(k-1)p_{2,3}+1+d_2(k-1)(1-p_{2,3})}{p_{2,3}+d_2(p_{0,3}-p_{2,3})}.
\end{aligned}
$$

Values $p_{i,j}$ are given by

$p_{0,1} = 0.5$	$p_{0,2} = 0.4$	$p_{0,3} = 0.1$
$p_{1,2} = 0.05$	$p_{1,3} = 0.04$	$p_{2,3} = 0.01$

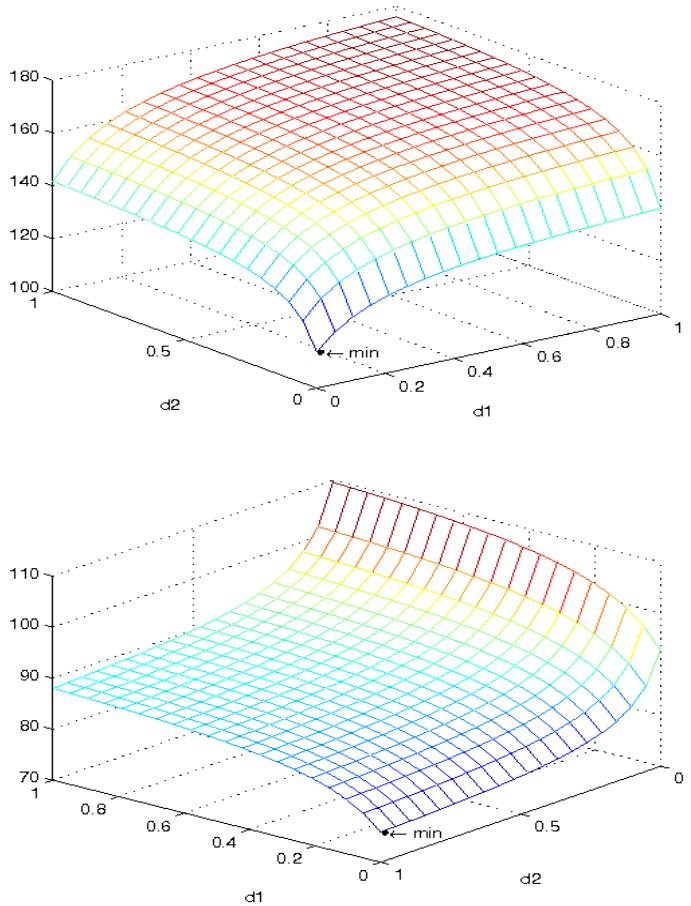
$evals_2(d_1, d_2)$ is evaluated at each vertex of the simplex I.
$evals_1\_d_1(d_1)$ and $evals_1\_d_2(d_2)$ are evaluated at the endpoints of [0.1]

For $k = 16$ and $k = 4$, $evals_2(d_1, d_2))$, $evals_1\_d_1(d_1)$ and $evals_1\_d_2(d_2)$ attain the minimum at the same values of d_1 and d_2. That does not hold for $k = 8$.

k	$evals_2(0,0)$	$evals_2(0,1)$	$evals_2(1,0)$	$evals_2(1,1)$
16	120.06	150.56	140.00	176.00
8	98.63	80.33	109.60	88.00
4	87.92	45.22	87.92,	44.00

k	$evals_1\_d_1(0)$	$evals_1\_d_1(1)$	$evals_1\_d_2(0)$	$evals_1\_d_2(1)$
16	26.11	32.00	115.00	160.00
8	18.11	16.00	107.00	80.00
4	14.11	8.00	103.00	40.00

The number of fun evaluations for $k = 16$ [min (0,0)] and $k = 8$ [min(0,1)].



. Numerical Results

In [5] a parallel version of Glob, was presented.
This follows the MIMD model.
N processors: one server and N-1 clients.

Server task

- reads all the initial data and sends them to each client;
- receives the intermediate data from a sender client;
- combines them with all the data already received;
- sends back the updated data to the client sender;
- gathers the final data from each client.

Client task

- receives initial data from server;
- runs algorithm *Glob*;
- sends intermediate data to server;
- receives updated values from server;
- stops running *Glob* whenever its stop rule is met in any client execution;
- sends final data to server.

The communication between the server and each client concerns the parameters p_2, p_3 and k.

Each client, as soon as either finds a new local minimizer or after fixed number of iterations, sends the following data to the server.

- last minimum found;
- the number of function evaluations since last message sending;
- the number of iterations since last message sending;
- the number of local searches carried out since last message sending;
- status variable of value 0 or 1 denoting that the stop rule has been met.

The server processes them and sends back the new values.

. Test Problems

$$(1) min\ f(x) = \frac{\pi}{n}\{10sin^2(\pi y_1) + \sum_{i=1}^{n-1}[(y_i - 1)^2(1 + 10sin^2(\pi y_{i+1}))] + (y_n - 1)^2\}$$

with $n = 100, y_i = 1 + \frac{1}{4}(x_i - 1),\ S \equiv \{x \in R^n \mid -10 \leq x_i \leq 10, \quad i = 1, ..., n\}$;

$$(2) min\ f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$$

with $n = 25,\ S \equiv \{x \in R^n \mid -10 \leq x_i \leq 10, \quad i = 1, ..., n\}$;

$$(3) min\ f(x) = 10n + \sum_{i=1}^{n}(x_j^2 - 10cos(2\pi x_j));$$

with $n = 8,\ S \equiv \{x \in R^n \mid -2.56 \leq x_i \leq 2.56, \quad i = 1, ..., n\};$

$$(4) min\ f(x) = \begin{cases} \left(\frac{2}{\rho_i^2}\frac{\langle x-m_i, x_t-m_i\rangle}{\|x-m_i\|} - \frac{2}{\rho_i^3}(\|x_t - m_i\|^2 + 4 - f_i)\right)\|x - m_i\|^3 + & (x \in B_i) \\ \left(1 - \frac{4}{\rho_i}\frac{\langle x-m_i, x_t-m_i\rangle}{\|x-m_i\|} + \frac{3}{\rho_i^2}(\|x_t - m_i\|^2 + 4 - f_i)\right)\|x - m_i\|^2 + f_i, & \\ \|x - x_t\|^2 + 4, \quad (x \notin B_i), & \end{cases}$$

with $n = 20$, $B_i \equiv \{x \in \mathbb{R}^n \mid \|x - m_i\| \leq \rho_i\}$, for $i = 1, ..., 9$,
$S \equiv \{x \in R^n \mid -1 \leq x_j \leq 1,\ j = 1, \ldots, n\}$, $m_i, (i = 1, ..., 9)$, and x_t denoting ten points uniformly chosen in S such that the B_i balls do not overlap each other, f_i real values to be taken as the values of $f(\cdot)$ at m_i.

. Local Search

The local minimization has been carried out by a code, called *cgtrust*, written by C.T. Kelley [7].

This code implements a *trust region* type algorithm that uses a polynomial procedure to compute the step size along a search direction.

. Software and Hardware

The parallel algorithm version of Glob has been tested in a parallel MatLab environment under the Linux operating system.

Two computers have been used; the first equipped with an Intel Quad CPU Q9400 based on four processors, the second with a AMD PHENOM II X6 1090T based on six processors.

Experiments have been carried out both on each single computer and on the two connected to a local network.

In the table we report the fourteen configurations of the computing resources used in each of our experiments.

config. no.	computer 1	computer 2	workers in 1	workers in 2
1	Intel Quad		1	
2	Intel Quad		2	
3	Intel Quad		3	
4	Intel Quad		4	
5	Amd phenom 6		1	
6	Amd phenom 6		2	
7	Amd phenom 6		4	
8	Amd phenom 6		6	
9	Intel Quad	Amd phenom 6	2	2
10	Intel Quad	Amd phenom 6	4	4
11	Intel Quad	Amd phenom 6	4	6
12	Amd phenom 6	Intel Quad	2	2
13	Amd phenom 6	Intel Quad	4	4
14	Amd phenom 6	Intel Quad	6	4

Table: The configurations of the computing resources

- The code used with one worker is largely simpler than that with more than one worker.
- 100 runs of the algorithm have been done on each problem. The averages are reported.
- The computational cost of local searches is the sum of function and gradient evaluations.
- The algorithm stops whenever the global minimum has been found within a fixed accuracy.
- To evaluate the performance of our algorithm speedup and the efficiency are given.

. Speedup

$$S_p = \frac{T_1}{T_p}$$

where T_1 is the time of execution of the algorithm with 1 processor and T_p is the time of execution of the algorithm with p processors.
$S_p \quad ideal = p$

Efficiency

$$E_p = \frac{S_p}{p}$$

estimates how much the parallel execution exploit the computer resources.
$E_p \quad ideal = 1$

	fun_1			fun_2			fun_3			fun_4		
processors	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*
1	0.0574			472.2093			478.1111			84.5026		
2	0.0819	0.70	0.35	618.5813	0.76	0.38	685.4575	0.70	0.35	90.8027	0.93	0.47
4	0.0252	2.28	0.57	196.6817	2.40	0.60	225.4558	2.12	0.53	34.7754	2.43	0.61
6	0.0222	2.59	0.43	122.9062	3.84	0.64	146.722	3.26	0.54	19.8663	4.25	0.71

Table: Results working with Amd Phenom 6

	fun_1			fun_2			fun_3			fun_4		
processors	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*
1	0.150493			803.5487			820.9142			143.0092		
2	0.117475	1.28	0.64	996.9096	0.81	0.40	855.5212	0.96	0.48	154.8207	0.92	0.46
3	0.027	5.57	1.86	520.4818	1.54	0.51	482.2071	1.70	0.57	84.1986	1.70	0.57
4	0.0262	5.74	1.44	280.6494	2.86	0.72	337.9634	2.43	0.61	59.7636	2.39	0.60

Table: Results working with Intel Quad

	fun_1			fun_2			fun_3			fun_4		
processors	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*
1	0.1039			637.879			649.51265			113.7559		
2+2	0.0263	3.95	0.99	307.2342	2.08	0.52	275.5479	2.36	0.59	54.6751	2.08	0.52
4+4	0.027	3.85	0.48	131.8754	4.84	0.60	134.5788	4.83	0.60	24.226	4.70	0.59
4+6	0.0253	4.11	0.41	83.555	7.63	0.76	96.1157	6.76	0.68	18.9221	6.01	0.60

Table: Results working with Intel Quad and Amd Phenom 6

	fun_1			fun_2			fun_3			fun_4		
processors	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*
1	0.1039			637.879			649.51265			113.7559		
2+2	0.0265	3.92	0.98	255.9299	2.49	0.62	287.6572	2.26	0.56	48.0993	2.37	0.59
4+4	0.0263	3.95	0.49	133.6477	4.77	0.60	116.5528	5.57	0.70	19.179	5.93	0.74
6+4	.0256	4.06	0.41	90.5088	7.05	0.70	109.5831	5.93	0.59	13.689	8.31	0.83

Table: Results working with Amd Phenom 6 and Intel Quad

Remarks

- In almost all the experiments the parallel algorithm improves largely the speedup of the computation. The efficiency in many experiments is above 0.70 although the task of a worker is to start the process and to collect and to distribute the intermediate and final data.
- The speedup becomes less than one only when two workers are employed. Clearly this has to be related to the fact that the complexity of the parallel code is not balanced by the use of just one additional worker.

	fun_1			fun_2			fun_3			fun_4		
processors	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*	*secs*	*speed*	*eff*
1	0.09			14.96			22.12			19.74		
2+2	0.07	1.29	0.32	9.15	1.63	0.41	27.26	0.81	0.20	6.3	3.13	0.78
4+4	0.07	1.29	0.16	4.75	3.15	0.39	11.06	2.00	0.25	3.36	5.88	0.73
4+6	0.07	1.29	0.13	4.54	3.30	0.33	11.49	1.92	0.19	2.52	7.83	0.78

Table: Results working with Intel Quad and Amd Phenom 6, codes in C

In this table we give for the configurations 9-11 the results with the codes written in C . The speedup is smaller; the function evaluations are carried out in a more efficient way, hence the advantages of the parallel computation is less clear.

C.G.E. Boender and A.H.G. Rinooy Kan.

Bayesian stopping rules for a class of stochastic global optimization methods.
Technical Report Report 8319/0, Erasmus University Rotterdam, 1985.

B. C. Cetin, J. Barhen, and J. W. Burdick.

Terminal repeller unconstrained subenergy tunnelling (trust) for fast global optimization.
Journal of Optimization Theory and Applications, 77 (1):97–126, 1993.

R. Desai and R. Patil.

Salo: combining simulated annealing and local optimization for efficient global optimization.
In *Proceedings of the 9th Florida AI Research Simposium,FLAIRS-96'*, pages 233–237. 1996.

M. Gaviano, D. Lera, and Steri A.M.

A local search method for continuous global optimization.
Journal of Global Optimization, 48: 73–85, 2010.

M. Gaviano, D. Lera, and Mereu E.

A parallel algorithm for global optimization prpoblems in a distribuited computing environment.
Thesis, ??:?, 2010.

Abdel-Rahman Hedar and Masao Fukushima.

Tabu search directed by direct search methods for non linear global optimization.
European Journal of Operational Research, 170 (2):329–349, 2006.

C. T. Kelley.

Iterative Methods for Optimization.
SIAM, Philadelphia, 1999.

A. Levy and A. Montalvo.

The tunneling algorithm for the global minimization of functions.
SIAM Journal on Scientific and Statistical Comuting, 6:15–29, 1985.

S. Lucidi and M. Piccioni.

Random tunneling by means of acceptance-rejection sampling for global optimization.
Journal of Optimization Theory and Applications, 62 (2):255–277, 1989.