ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

G. Di Pillo, S. Lucidi, L. Palagi, M. Roma

A CONTROLLED RANDOM SEARCH ALGORITHM WITH LOCAL NEWTON-TYPE SEARCH FOR GLOBAL OPTIMIZATION

R. 465 Maggio 1998

Gianni Di Pillo - Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Buonarroti 12 - 00185 Roma, Italy. Email : dipillo@dis.uniroma1.it.

Stefano Lucidi - Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Buonarroti 12 - 00185 Roma, Italy. Email : lucidi@dis.uniroma1.it.

Laura Palagi - Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Buonarroti 12 - 00185 Roma, Italy. Email : palagi@dis.uniroma1.it.

Massimo Roma - Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Buonarroti 12 - 00185 Roma, Italy and Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30 - 00185 Roma, Italy. Email : roma@dis.uniroma1.it.

Istituto di Analisi dei Sistemi ed Informatica, CNR
viale Manzoni 30
00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.rm.cnr.it
URL: http://www.iasi.rm.cnr.it

Abstract

In this work we deal with the problem of finding an unconstrained global minimizer of a multivariate twice continuosly differentiable function. In particular we propose an algorithm which combines a controlled random search procedure based on the modified Price algorithm described in [2] with a Newton-type unconstrained minimization algorithm proposed in [7]. More in particular, we exploit the skill of the Price strategy to examine the whole region of interest in order to locate the subregions "more promising" to contain a global minimizer. Then starting from a point in these regions, we use an effective Newton-type algorithm to compute very quickly the closest local minimizer. In this way we succeed in improving the efficiency of the Price approach. Numerical results on a set of standard test problems are reported with the aim to put in evidence the improvement in efficiency when dealing with large scale problems.

1. Introduction

In this paper, we consider the problem of finding a global solution of the unconstrained minimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$. Besides its own interest, the importance of this problem derives from the fact that several real world applications require the efficient solution of a global optimization problem; indeed, a large number and variety of decision problems arising in physics, engineering, economics can be modelled as global optimization problems. In this field many algorithms have been proposed by following both probabilistic and deterministic approaches. However, in the general case, global optimization problems remain hard to solve, in particular when the number of variables increases. We remark that a global optimization problem is considered large if the number of variables is greater than ten.

In this paper we focus our attention on the design of an efficient algorithm for large scale problems, by exploiting the information contained in the first and second order derivatives of the objective function. In particular, we consider some modifications of the Controlled Random Search (CRS) Price algorithm. A CRS method is a globalized local search method based on a random global phase and a local phase. The global search is used to locate the subregions "more promising" to contain a global minimizer; the local search is used for determining the global minimizer as soon as a "sufficiently small" neighbourhood of this point has been located. The basic idea of the method is that of iteratively contracting an initial set of sample points by substituting the worst point in terms of objective function value with a better one until a stopping condition is verified.

In the original version of CRS method proposed by Price [11], the new points are obtained by using a simplex type strategy. Recently some modifications of the method have been introduced in [1, 2, 9, 10], showing a growing interest in this class of algorithms. In particular, in [2] a new version of the CRS Price algorithm was proposed and its efficient computational behaviour was shown in solving both standard test problems and a difficult problem arising in astrophysics.

The efficiency of the Price approach seems to be mainly due to the CRS strategy, which can be considered a compromise between a pure random search strategy and a clustering strategy. Indeed, the Price algorithm is usually able to locate very quickly subregions containing a global minimizer. Moreover, within CRS methods, it is possible to define efficient heuristic stopping criteria.

However, CRS approach, even in the improved versions, presents some drawbacks:

– it is unable to perform fast local minimizations;

– it needs a large number of initial sample points (practical choices are $10(n+1)$ or $25n$ points).

The inefficiency of the local phase can be due to the fact that CRS methods use only function evaluations without exploiting information on the derivatives. Although this enables to use of CRS methods for solving a wide class of problems, it may be a serious drawback for the effectiveness of the method. In fact, in order to get convergence, a large number of sample points is needed and, as a consequence, these methods are effective in the small dimensional case only.

On the basis of these remarks, we propose in this paper a new local phase that incorporates a gradient-based minimization algorithm. In particular, we propose a new version of the modified Price method described in [2], where the main feature is the introduction of a local search performed from a trial point generated by the algorithm. This new strategy enables to generate

very quickly a local minimizer as soon as a small neighbourhood of this point has been determined. Furthermore, as is known, the use of gradient-related directions ensures that any global minimizer has a region of attraction and hence, starting the local search from a point in this region, we can insert quickly a global minimizer in the set of "promising" points.
As a local minimization procedure, we use the Newton-type method proposed in [7] which guarantees the convergence to second order critical points, i.e. to stationary points where the Hessian matrix is positive semidefinite. Any local method with second order convergence, is able to escape very quickly from the region where the objective function is nonconvex. In our opinion, this feature should have a beneficial effect in a CRS Price approach. In fact a Price-type method using such second order algorithm in the local phase, should avoid not promising points and cluster round a global minimizer more quickly.

We assume throughout the paper that both the gradient $\nabla f(x)$ and the Hessian matrix $\nabla^2 f(x)$ of f exist and are continuous.

2. The Price Algorithm with Local Optimizer

In this section the new version of the Price Algorithm with Local Optimizer (PALO) is described. We do not report here a detailed description of the Improved Price Algorithm (IPA) which constituted our basis, but we refer to [2] for the analysis of the method.
As mentioned earlier, the main distinguishing feature of our algorithmic scheme with respect to the IPA algorithm is the use of an unconstrained minimization phase in order to improve the performance of the method. This allows us to tackle large scale problems that both the original and the improved Price algorithms are unable to solve.
As usual, rather than Problem (1), we consider the problem

$$\min_{x \in D} f(x),$$

where $D \subseteq \mathbb{R}^n$ is a compact set and the minimization is essentially unconstrained in the sense that all the local minimizers of f are interior points of D. Usually the set D is defined by simple bounds on the variables; when there is no information on the magnitude of some components at the solution, we consider wide bounds.
The scheme of the Price Algorithm with Local Optimizer is reported in Table 1.

The algorithm works as follows. A set S^0 of sample points is chosen at random over D at Step 0, and among these, the "worst" and the "better" ones, x^k_{max} and x^k_{min}, in terms of objective function values are determined (Step 1). At each iteration k, a trial point $\tilde{x}^k$ is determined by a weighted reflection (Step 3) with respect to a centroid c^k_w computed over a subset of $n+1$ sample points selected at random in S^k (Step 2). In the original and the improved versions of the Price algorithm, whenever $f(\tilde{x}^k) < f(x^k_{max})$ the trial point $\tilde{x}^k$ is accepted and the set S^k is updated by replacing the point x^k_{max} with $\tilde{x}^k$. The algorithmic scheme defined here, differs from the Price scheme in the updating rule for the set S^k, namely in Step 5, where we introduce a "local phase". Whenever the new trial point $\tilde{x}^k$ obtained at Step 3 provides an acceptable value of the objective function (i.e. $f(\tilde{x}^k) < f(x^k_{max})$), we start from $\tilde{x}^k$ to determine a local minimizer, that is a better point $\bar{x}^k$ such that $f(\bar{x}^k) < f(\tilde{x}^k) < f(x^k_{max})$. Then we use the point $\bar{x}^k$ to update the set S^k.
We remark that at Step 2 there is a finite number of different choices for $n+1$ points over S^k, so that, if condition $\tilde{x}^k \notin D$ at Step 3 or condition $f(\tilde{x}^k) \geq f(x^k_{max})$ at Step 4 occur a number of times exceeding the number of possible choices, we take at random a point $\hat{x} \notin S^k$ over D and

Price Algorithm with Local Optimizer (PALO)

Data: *A positive integer $m \geq n+1$ and a positive real number ω.*

Step 0: *Set $k = 0$; determine a set $S^k = \{x_1^k, \ldots, x_m^k\}$ of points chosen at random over D and evaluate $f(x_i^k)$, $i = 1, \ldots, m$.*

Step 1: *Determine x_{max}^k, f_{max}^k, x_{min}^k, f_{min}^k such that*

$$f_{max}^k = f(x_{max}^k) = \max_{x \in S^k} f(x), \qquad \text{and} \qquad f_{min}^k = f(x_{min}^k) = \min_{x \in S^k} f(x).$$

If a stopping criterion is satisfied, then stop.

Step 2: *Choose at random $n+1$ points $x_{i_0}^k, x_{i_1}^k, \ldots, x_{i_n}^k$ over S^k and determine the weighted centroid c_w^k of the n points $x_{i_1}^k, \ldots, x_{i_n}^k$*

$$c_w^k = \sum_{j=1}^{n} w_j^k x_{i_j}^k \qquad \text{where}$$

$$w_j^k = \frac{\eta_j^k}{\sum_{j=1}^{n} \eta_j^k}, \quad \eta_j^k = \frac{1}{f(x_{i_j}^k) - f_{min}^k + \phi^k}, \quad \phi^k = \omega \frac{\left(f_{max}^k - f_{min}^k\right)^2}{f_{max}^0 - f_{min}^0}.$$

Step 3: *Compute $f_w^k = \sum_{j=1}^{n} w_j^k f(x_{i_j}^k)$ and determine the trial point $\tilde{x}^k$ by*

$$\tilde{x}^k = \begin{cases} c_w^k - \alpha^k (x_{i_0}^k - c_w^k), & \text{if} \quad f_w^k \leq f(x_{i_0}^k) \\ x_{i_0}^k - \alpha^k (c_w^k - x_{i_0}^k), & \text{if} \quad f_w^k > f(x_{i_0}^k) \end{cases}$$

$$\text{with} \qquad \alpha^k = 1 - \frac{|f(x_{i_0}^k) - f_w^k|}{f_{max}^k - f_{min}^k + \phi^k}.$$

If $\tilde{x}^k \notin D$ go to Step 2; otherwise compute $f(\tilde{x}^k)$.

Step 4: *If $f(\tilde{x}^k) \geq f_{max}^k$ then take $S^{k+1} = S^k$.*
Set $k = k+1$ and go to Step 2.

Step 5: *If $f(\tilde{x}^k) < f_{max}^k$ then perform an unconstrained local minimization starting from $\tilde{x}^k$ to obtain a local minimizer $\bar{x}^k$. Take $S^{k+1} = S^k \cup \{\bar{x}^k\} \setminus \{x_{max}^k\}$.*
Set $k = k+1$ and go to Step 1.

Table 1: Price Algorithm with Local Optimizer (PALO)

we set $S^{k+1} = S^k \cup \{\hat{x}\} \setminus \{x^k_{max}\}$. For the sake of simplicity this step is not explicitly reported in the scheme of the algorithm.

The use of this strategy should enable to overcome some of the main drawbacks of the Price approach. One of these is the fact that, once a region containing a global minimizer is located, a Price-type algorithm needs an expensive computational burden to get a poor approximation of the global minimizer. The use of a local phase allows us to obtain very quickly a good approximation of the global minimizer. A second important aspect concerns the "global phase". In fact, the new updating rule enables to include more "promising" points in the set of sample points S^k, and this should improve the skill of the procedure to cluster more and more round the subregions which are more likely to contain a global minimizer.

As regards the local minimization algorithm, we use a Newton-type method. Thus, we need information about derivatives of the objective function that are not needed in the original and in the improved Price algorithms. Whenever the objective function is provided in the form of a black box, we can obtain the derivatives numerically. Since the use of derivatives may be expensive, the computational saving obtained by using these information must be significant. Therefore, the local algorithm must be as efficient as possible to locate a good estimate of the local minimizer in few iterations so that the overall computational burden is not seriously affected. We use a local algorithmic model based on a search along a curvilinear path that is defined by a suitable combination of a truncated Newton direction and a negative curvature direction. In particular the local algorithm produces a sequence of points $\{x_k\}$ according to the rule

$$x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k \tag{2}$$

where s_k is a Newton-type direction, d_k is a negative curvature direction (i.e., a direction such that $d_k^T \nabla^2 f(x_k)) d_k < 0$) and α_k is a step-length computed by a nonmonotone Armijo-type curvilinear linesearch. We refer to [7, 8] for a detailed analysis of this local algorithm. We just mention here that the main feature of the method is to ensure global convergence towards points satisfying the second order necessary optimality condition, namely stationary points where the Hessian matrix is positive semidefinite. Another important aspect concerning the local algorithm is the use of a truncated scheme to compute the Newton-type direction still ensuring convergence to second order points. In particular the Lanczos method is used to compute both search directions and this allows us to efficiently solve even large scale problems. We recall that in truncated Newton schemes it is not necessary to store the Hessian matrix of f, but only matrix-vector products.

The use of this local minimization scheme enables to escape very quickly from the region where the objective function is locally nonconvex by the use of negative curvature directions. Besides of improving the efficiency of the local algorithm, the use of negative curvature directions allows us to explore subregions which could be missed by using the original or the improved Price iterates.

As regards the convergence properties, the CRS Price algorithms are clearly heuristic procedures. However, the algorithmic scheme can be easily modified in order to produce a sequence of points globally convergent in probability towards a global minimizer. In fact, it is sufficient to continue, once in a while, to choose at random points over D. Moreover, the use of a local gradient-related algorithm helps in improving also the theoretical properties of the CRS Price method. Indeed, the following proposition [6] states the existence of a neighbourhood of "attraction" of a global minimizer such that the sequence of points produced by the algorithm converges.

Proposition 2.1. *Let f be a twice differentiable function and x^* a minimizer such that $\nabla^2 f(x^*)$ is positive definite. Assume that there exist positive numbers a and c such that*

$$\alpha^k \leq a, \qquad \|s^k\| \leq c\|\nabla f(x^k)\|, \quad k = 1, 2, \ldots$$

Let $\{x^k\}$ be the sequence of points generated by (2) starting from x^0. Assume that for each k

$$f(x^k) \leq f(x^0).$$

Then there exists a neighbourhood $\mathcal{B}(x^*; \rho)$ such that, for every $x^0 \in \mathcal{B}(x^*; \rho)$ the sequence $\{x^k\}$ remains in $\mathcal{B}(x^*; \rho)$ and converges to x^*.

This proposition justifies the use of a local gradient-related algorithm in the CRS methods framework. In fact by choosing at random points over D, a point in the neighbourhood of attraction of a global minimizer is found almost surely, in a finite number of trials [6]. When such a point is located, the local procedure converges towards the global minimizer.

3. Computational Results

In this section, we report the results of a numerical testing of the PALO algorithm. Our aim is to show the effectiveness of introducing a local minimization phase in the Price algorithm. We compare the results obtained by PALO with those obtained by the Improved Price Algorithm (IPA) proposed in [2].

In order to evaluate the behaviour of our algorithm we tested it on a set of standard test problems [2, 3, 4, 5, 6]. In Table 2 the features of the test problems are summarized. In particular we report for each problem the dimension, the region of interest D, the number of local minimizers, the optimal value of the objective function f^* and the source paper. Note that they are standard test problems which include some with an exponential number of local minimizers.

We tested first the PALO algorithm on problems of small dimension ($n = 2, 3, 4, 6$) in order to check the reliability of the approach. Then, we considered the subset of test functions with parametric dimension (L1, L2, L3, GW) and we set $n = 20, 50, 100$. We recall that, even if the definition "large scale" may be machine dependent, in the global optimization framework a problem with more than ten variables is usually considered of large size. All the computations were performed on an IBM RISC System/6000 375. The codes are double precision Fortran 90 compiled under xlf90 with the optimization compiling option. Random numbers were generated by using the XL Fortran90 intrinsic procedure `RANDOM_NUMBER` which returns a pseudorandom number from the uniform distribution over the range $(0, 1)$. Since at Step 0 and Step 2 of the algorithm it is required to select at random sample points, we perform, for each test problem, ten runs changing the seed of the pseudo-random number generator and we report the average results.

As regards the stopping criterion used at Step 2, we use a standard rule in CRS methods, namely:

$$f^k_{max} - f^k_{min} \leq 10^{-6}$$

whereas the stopping criterion for the local procedure in PALO is $\|\nabla f(x)\| \leq 10^{-5}$.

As regards the key parameter m which defines the number of sample points in the set S^k, in the CRS Price methods (included IPA) the value $25n$ is suggested. As mentioned earlier, one of the main points of this paper is to show that, thanks to the presence of the local algorithm, the

Problem	dim.	region of interest D	# min	f^*	Ref.
Camel (CL)	2	$-2.5 \leq x_1 \leq 2.5$ $-1.5 \leq x_2 \leq 1.5$	6	-1.0316285	[2]
Quartic (QC)	2	$-10 \leq x_{1,2} \leq 10$	2	-0.352386	[6]
Shubert (SH)	2	$-10 \leq x_{1,2} \leq 10$	760	-186.73091	[6]
Pen. Sh. (SHP)	2	$-10 \leq x_{1,2} \leq 10$	760	-186.73091	[6]
Pen. Sh. 2 (SHP2)	2	$-10 \leq x_{1,2} \leq 10$	760	-186.73091	[6]
Treccani (TR)	2	$-2.5 \leq x_1 \leq 2.5$ $-1.5 \leq x_2 \leq 1.5$	2	0.0	[3]
Hartman 3 (H3)	3	$0 \leq x_{1,2,3} \leq 1$	4	-3.8627	[2]
Shekel5 (SK5)	4	$0 \leq x_{1,\ldots,4} \leq 10$	5	-10.1532	[2]
Shekel7 (SK7)	4	$0 \leq x_{1,\ldots,4} \leq 10$	7	-10.4029	[2]
Shekel10 (SK10)	4	$0 \leq x_{1,\ldots,4} \leq 10$	10	-10.5364	[2]
Hartman 6 (H6)	6	$0 \leq x_{1,\ldots,6} \leq 1$	4	-3.3223	[2]
Levy 1 (L1)	n	$-10 \leq x_{1,\ldots,n} \leq 10$	$\approx 5^n$	0.0	[5]
Levy 2 (L2)	n	$-10 \leq x_{1,\ldots,n} \leq 10$	$\approx 10^n$	0.0	[5]
Levy 3 (L3)	n	$-10 \leq x_{1,\ldots,n} \leq 10$	$\approx 15^n$	0.0	[5]
Griewank (GW)	n	$-10 \leq x_{1,\ldots,n} \leq 10$	([1])	0.0	[4]

Table 2: The set of test problems.

number of points m needed to get convergence can be set to a smaller value than the standard one. Of course, in the small scale case a minimum number of random points must be generated in order to ensure a sufficient sample of the set D. To this aim, in our numerical experiments, we choose the value

$$m = \max\{3(n+1), \overline{m}\} \tag{3}$$

where $\overline{m}$ is a prefixed number of sample points.
We report in Table 3 and Table 4 the average results over 10 runs for PALO and IPA for the small scale problems as $\overline{m}$ ranges from 20 to 50. In particular, we report the dimension n, the value of m, the average number of iterations (# it), the average number of iterations until a global minimizer is found for the first time (# it_{opt}), the overall number of failures (# fail) (i.e. the number of runs terminated without attaining the global minimum value f^* within the tolerance of 10^{-6}). As regards PALO, we report also the average number of calls to the local optimizer (# local). We do not report in these tables the computational effort in terms of function, gradient evaluations and matrix-vector products (see the Appendix for the detailed results), since, in the small scale case, we are interested in verifying that the use of a local optimizer reduces the computational burden of the global phase of the Price method, namely

[1] For $n = 2$ this function has 500 minimizers

			PALO				IPA		
Problem	n	m	# it	# it_{opt}	#local	# fail	# it	# it_{opt}	#fail
CL	2	20	48.9	2.5	19.5	0	445.1	404.2	0
		30	68.6	1.9	28.6	0	646.9	544.4	0
		40	85.7	1.3	37.2	0	861.6	725.3	0
		50	123	2.5	50.6	0	1118.6	949.6	0
QC	2	20	29.2	1.3	19.2	0	417	372.8	1
		30	47.8	2.1	33.7	0	510	429.3	1
		40	61.4	2.1	43.7	0	711.4	573.3	0
		50	77.7	2	56.5	0	899.3	712.2	0
SH	2	20	179.8	179.8	49.9	0	830.7	805	0
		30	280.5	280.5	73.8	0	1683.8	1683.8	0
		40	420.4	420.4	104	0	3196.7	3179.9	0
		50	499	499	126.9	0	4023	3923.1	0
SHP	2	20	334.1	334.1	76.3	2	963.6	963.6	7
		30	534.1	534.1	117.3	0	2376.1	2376.1	2
		40	701.1	701.1	164.8	0	2729.4	2729.4	0
		50	846.8	846.8	191.9	0	3201.1	3201.1	1
SHP2	2	20	291.1	291.1	71.6	1	1061.2	1061.2	3
		30	607	607	116.7	1	1640.7	1640.7	3
		40	906.1	906.1	148.7	0	2502.3	2502.3	1
		50	1235.7	1235.7	206	0	3094.3	3094.3	0
TR	2	20	29	1	15.9	0	369	328.5	0
		30	48.3	1.1	22.3	0	535.9	456.3	0
		40	60.5	1	29.8	0	769	673.7	0
		50	73.3	1	37.5	0	922.5	738.7	0
L1	2	20	62.8	3.7	26.4	0	434.2	380.9	0
		30	97.3	4.7	40.5	0	665.6	570.7	0
		40	112.4	2.4	49	0	900	790.1	0
		50	144	5.7	64.2	0	1115.2	879.5	0
L2	2	20	106.8	12.3	44.6	0	525.4	475.9	0
		30	140.5	9	63	0	805.8	716.3	0
		40	205.1	5.5	83.5	0	1043.1	890.3	0
		50	249.8	9.8	106.8	0	1352.8	1189.2	0
L3	2	20	153.5	56.5	60.8	0	541.2	492.1	0
		30	225.2	55.5	92.7	0	809.8	716	0
		40	308.2	57.1	128.2	0	1041.6	850	0
		50	354.8	56.2	153.8	0	1285.7	1112	0
GW	2	20	120.5	18.6	33	0	647.7	587.9	0
		30	167.6	8	50.3	0	862	769.9	0
		40	240.1	16.5	67.7	0	1192.4	1060.5	0
		50	277.7	16.2	83.8	0	1591.6	1374.8	0
H3	3	20	43.8	43.8	19.1	0	530.2	496	0
		30	58.2	58.2	27.5	0	812.9	812.9	0
		40	65.2	65.2	34.9	0	1115.9	1115.9	0
		50	87.7	87.7	44.9	0	1484.4	1484.4	0

Table 3: Average results over 10 runs for small scale problems. Part 1

			PALO				IPA		
Problem	n	m	# it	# it_{opt}	# local	# fail	# it	# it_{opt}	# fail
SK5	4	20	93.4	3.6	28.1	0	1973.5	1965.4	2
		30	157.1	6.9	43.2	0	3347.2	3299.8	0
		40	113.8	3	45.2	0	4698.4	4605.6	0
		50	135.8	3.2	57.4	0	5795.7	5663.3	0
SK7	4	20	106.3	106.3	23.7	0	2095.8	2077.4	0
		30	129.2	129.2	36.6	0	2899.3	2880	1
		40	123.9	123.9	45.2	0	4231.4	4195	0
		50	129	129	54.8	0	5294.2	5137.1	0
SK10	4	20	106.3	106.3	25.1	0	2032	2013.6	0
		30	120.7	120.7	35.4	0	3331.1	3277.4	0
		40	169.2	169.2	48.3	0	4220	4158	0
		50	188.9	188.9	60.9	0	5260.8	5095.2	0
H6	6	21	38.1	38.1	21.6	0	1626.9	1626.9	6
		30	38.4	38.4	27.5	0	2682.2	2682.2	0
		40	50.6	50.6	36.4	0	3876.1	3781.	0
		50	67.7	67.7	47.8	0	4936.3	4857.3	0

Table 4: Average results over 10 runs for small scale problems. Part 2

the number of iterations needed.

From Tables 3 and 4 it can be easily observed that PALO is much more efficient than IPA in terms of number of iterations. Moreover, it is also evident that a global minimizer is usually found more quickly by PALO rather than by IPA. This is due to the fact that, as mentioned earlier, by choosing at random points over D, a point in a neighbourhood of attraction of a global minimizer is almost surely found and by using a local gradient-related algorithm we get it. We observe also that the PALO algorithm seems to be more reliable in the sense that, with a smaller number of sample points, it succeeds in finding a global minimizer.

The results obtained for the small dimensional case are very encouraging. For these problems, the comparison among CPU time of PALO and IPA is meaningless because in both cases the time needed to solve the problems is very small. Indeed, the presence of the local optimizer increases the computational burden, but this increase is not significant in the small dimensional case whereas it could be significant in the large scale case. However, as the dimension increases the effectiveness of the local algorithm increases too and hence we believe that the use of the local phase could have a crucial role in the large scale case. To verify this point, we compare PALO and IPA on the subset of test functions (L1, L2, L3, GW) with parametric dimension n where we set $n = 20, 50, 100$. As concerns the parameter m, we use the value given by (3) that results to be $m = 3(n+1)$ for PALO, whereas in order to allow IPA to solve these problems we increase the value of m up to $m = 25n$, which is the value suggested for the original CRS Price algorithm.

The results obtained by PALO and IPA on the large scale test problems are reported in Table 5. In particular, we have summarized the overall computational burden of the two algorithms in terms of average CPU time needed for solving each problem. Moreover we report the best objective function optimal value (f^*) obtained by both algorithms and the number of failures over the ten runs (# fail). The detailed results obtained by both algorithms are reported in the Appendix. We observe that, even if we allow IPA to use a larger number of sample points, whenever the dimension of the problem is greater than 20, IPA is almost always unable to solve

the problem within the tolerance of 10^{-6}. Moreover, also in the cases where both algorithms get a global minimizer, the CPU time required by PALO is much smaller than the CPU time required by IPA. This shows that there is a significant computational saving using PALO rather than IPA in spite of the burden due to the local optimizer. Furthermore, the results reported in Table 5 clearly show that even the accuracy of the solution is much higher by using PALO rather than IPA.

		PALO				IPA			
Problem	n	m	time	f^*	# fail	m	time	f^*	# fail
L1	20	63	0.95	$2 \cdot 10^{-32}$	0	500	45.78	$9 \cdot 10^{-8}$	0
	50	153	5.41	$9 \cdot 10^{-31}$	0	1250	647.18	$7 \cdot 10^{-6}$	7
	100	303	26.69	$6 \cdot 10^{-29}$	0	2500	7466.23	$9 \cdot 10^{-3}$	10
L2	20	63	4.73	$2 \cdot 10^{-32}$	0	500	48.75	$3 \cdot 10^{-7}$	0
	50	153	35.58	$9 \cdot 10^{-33}$	0	1250	738.61	$1 \cdot 10^{-2}$	10
	100	303	141.52	$4 \cdot 10^{-33}$	0	2500	8166.21	$3 \cdot 10^{-1}$	10
L3	20	63	5.82	$6 \cdot 10^{-28}$	0	500	55.59	$1 \cdot 10^{-7}$	0
	50	153	36.08	$6 \cdot 10^{-28}$	0	1250	837.04	$2 \cdot 10^{-5}$	5
	100	303	157.62	$6 \cdot 10^{-28}$	0	2500	9159.59	$2 \cdot 10^{-2}$	10
GW	20	63	0.97	$1 \cdot 10^{-20}$	0	500	78.53	$1 \cdot 10^{-7}$	0
	50	153	14.02	$4 \cdot 10^{-32}$	0	1250	1153.35	$8 \cdot 10^{-7}$	0
	100	303	126.82	$7 \cdot 10^{-32}$	0	2500	10478.2	$1 \cdot 10^{-4}$	10

Table 5: Comparison between PALO and IPA for large scale problems

Appendix

In this appendix we report the tables of the detailed results of our numerical experiments. All the results are average results over ten runs. In particular in Tables 6–7 we report the comparative results for the small scale problems obtained by using PALO and IPA. For each problem we report the dimension n, the value of m, the average number of iterations (# it), the average number of function evaluations (# f). As regards PALO we report also the average number of gradient evaluations (# g) and the average number of matrix-vector products (# Hd).
In Tables 8–9–10 we report the detailed results obtained for the large scale problems by using PALO and IPA. For each problem we report the dimension n, the value of m, the average number of iterations (# it), the average number of function evaluations (# f). As regards PALO we report also the average number of gradient evaluations (# g) and the average number of matrix-vector products (# Hd); moreover in Table 9 we report the average results obtained until a global minimizer is found for the first time (# it_{opt}, # f_{opt}, # g_{opt}, # Hd_{opt}).

			PALO				IPA	
Problem	n	m	# it	# f	# g	# Hd	# it	# f
CL	2	20	48.9	161.9	109.5	108.8	445.1	465.1
		30	68.6	235.9	159.3	261.6	646.9	676.9
		40	85.7	306.3	211.8	350.6	861.6	901.6
		50	123	413.8	272.9	445.8	1118.6	1168.6
QC	2	20	29.2	126.9	126	133.9	417	437
		30	47.8	207.8	203	212.1	510	540
		40	61.4	275.5	271	284.5	711.4	751.4
		50	77.7	351.6	352.8	368.6	899.3	949.3
SH	2	20	179.8	390.3	232.4	369.2	830.7	850.7
		30	280.5	600.6	347.4	551.8	1683.8	1713.8
		40	420.4	858.8	482.2	763.2	3196.7	3236.7
		50	499	1037.5	587.4	930.2	4023	4073
SHP	2	20	334.1	646.5	335.4	523	963.6	983.6
		30	534.1	1010.3	506.9	786.4	2376.1	2406.1
		40	701.1	1356.3	695.7	1071.6	2729.4	2769.4
		50	846.8	1627.7	841.7	1310	3201.1	3251.1

Table 6: Detailed average results over 10 runs for small scale problems. Part 1

			PALO				IPA	
Problem	n	m	# it	# f	# g	# Hd	# it	# f
SHP2	2	20	291.1	596.7	325.7	511.4	1061.2	1081.2
		30	607	1109.7	535	847.8	1640.7	1670.7
		40	906.1	1538.1	673.5	1060	2502.3	2542.3
		50	1235.7	2109.8	920.9	1445.8	3094.3	3144.3
TR	2	20	29	115.3	80.4	95.6	369	389
		30	48.3	168.6	112.5	134.2	535.9	565.9
		40	60.5	225.3	146.2	177	769	809
		50	73.3	278.7	188.2	227.6	922.5	972.5
L1	2	20	62.28	199.7	124.4	197.4	434.2	454.2
		30	97.3	306.9	192	306.8	665.6	695.6
		40	112.4	374.9	240.8	391.2	900	940
		50	144	481.1	309.1	495	1115.2	1165.2
L2	2	20	106.8	358.7	260.1	434.6	525.4	545.4
		30	140.5	490	362.9	604.4	805.8	835.8
		40	205.1	660.9	471.6	782.8	1043.1	1083.1
		50	249.8	838.7	609.3	1013.6	1352.8	1402.8
L3	2	20	153.5	675.3	345.5	574.6	541.2	561.2
		30	225.2	1022.4	526	872.7	809.8	839.8
		40	308.2	1391.8	729.1	1210.7	1041.6	1081.6
		50	354.8	1686.9	876.7	1458.6	1285.7	1345.7
GW	2	20	120.5	252.4	119.9	175.0	647.7	667.7
		30	167.6	369.6	183.6	268	862	892
		40	240.1	511.1	245.8	360.2	1192.4	1232.4
		50	277.7	618.3	313.5	462.8	1591.6	1641.6
H3	3	20	43.8	236.5	147.1	311.9	530.2	550.2
		30	58.2	337.8	213.8	454.3	812.9	842.9
		40	65.2	407.9	264.4	559.1	1115.9	1155.9
		50	87.7	529.9	343.3	728.4	1484.4	1534.4
SK5	4	20	93.4	432.8	297.6	555.1	1973.5	1993.5
		30	157.1	674	451.9	845.2	3347.2	3377.2
		40	113.8	751.3	554.8	1052.8	4698.4	4738.4
		50	135.8	960.8	715.3	1360.4	5795.7	5845.7
SK7	4	20	106.3	421.8	273.4	519.7	2095.8	2115.8
		30	129.2	612.1	414.3	782.4	2899.3	2929.3
		40	123.9	781.9	554.5	1051.8	4231.4	4271.4
		50	129	1014.4	748.9	1430.9	5294.2	5344.2
SK10	4	20	106.3	434.8	285.8	542.8	2032	20520
		30	120.7	619.6	427.6	816.7	3331.1	3361.1
		40	169.2	832	572.3	1086.2	4220	4260
		50	188.9	1037.3	728.5	1384.7	5260.8	5310.8
H6	6	21	38.1	281.6	197	514.7	1626.9	1647.9
		30	38.4	350.6	255.4	646.2	2682.2	2712.2
		40	50.6	469.4	338	853.5	3876.1	3916.1
		50	67.7	599.1	434.5	118.5	4936.3	4986.3

Table 7: Detailed average results over 10 runs for small scale problems. Part 2

Problem	n	m	#it	# f	# g	# Hd
L1	20	63	128.5	1244.4	1275.1	3628.4
	50	153	222.4	2725.9	2742	8384.6
	100	303	468	3950.4	3806.2	10946.1
L2	20	63	282.5	4713.8	6098.8	21480.4
	50	153	614.7	15792.6	18819.2	83431.3
	100	303	1109.0	31501.2	36405.4	154430.4
L3	20	63	454.2	8430.1	7186.2	25615.9
	50	153	861.3	23678.2	21063	81358
	100	303	1627.8	53570.5	49772.9	170360.9
GW	20	63	69.4	417.1	410.5	1362.2
	50	153	157.3	999.2	1029	4092.4
	100	303	303.7	1930.4	2090.9	9750.8

Table 8: Detailed average results over 10 runs for PALO for large scale problems

Problem	n	m	$\#\text{it}_{opt}$	$\#\text{f}_{opt}$	$\#\ \text{g}_{opt}$	$\#\ \text{Hd}_{opt}$
L1	20	63	7	274.8	233.6	777.7
	50	153	6.8	553	427.4	1603.4
	100	303	5.7	754.4	442.9	1817.3
L2	20	63	24.2	1884.8	2360	9686.8
	50	153	35.5	6811.8	7606.6	41337.9
	100	303	26.1	9371.6	9487.9	55833.2
L3	20	63	100.3	4949.5	4644	18295.5
	50	153	65.9	8087.5	7635.1	34850.1
	100	303	94.3	23435.6	22961.8	94624.7
GW	20	63	1	67	3.5	12.2
	50	153	1	156	2	1.6
	100	303	1	306	2	1

Table 9: Detailed average results obtained until a global minimizer is found for the first time, over 10 runs for PALO for large scale problems

Problem	n	m	# it	# f	$\#\ \text{it}_{opt}$	$\#\ \text{f}_{opt}$
L1	20	500	96625.2	97125.2	95625.2	96125.2
	50	1250	260160	261410	260160	261410
	100	2500	539902.5	542402.5	*	*
L2	20	500	107184.3	107684.3	106469.2	106969.2
	50	1250	315078.9	316328.9	*	*
	100	2500	696716.7	699216.7	*	*
L3	20	500	122472.4	122972.4	121448.6	121948.6
	50	1250	344299.1	345549.1	344299.1	345549.1
	100	2500	700340.7	702840.7	*	*
GW	20	500	181045.2	181545.2	179533.3	180033.3
	50	1250	546569.4	547819.4	546444.6	547694.6
	100	2500	948964.9	951464.9	*	*

Table 10: Detailed average results over 10 runs for IPA for large scale problems

References

[1] M. M. Ali, A. Torn, and S. Viitanen, "A numerical comparison of some modified controlled random search algorithms," *Journal of Global Optimization*, vol. 11, pp. 377–385, 1997.

[2] P. Brachetti, M. De Felice Ciccoli, G. Di Pillo, and S. Lucidi, "A new version of the Price's algorithm for global optimization," *Journal of Global Optimization*, vol. 10, pp. 165–184, 1997.

[3] R. Ge, "A filled function method for finding a global minimizer of a function of several variables," *Mathematical Programming*, vol. 46, pp. 191–204, 1990.

[4] A. Griewank, "Generalized descent for global optimization," *Journal of Optimization Theory and Applications*, vol. 34, pp. 11–39, 1981.

[5] A. V. Levy and A. Montalvo, "The tunneling algorithm for the global minimization of functions," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, pp. 15–29, 1985.

[6] S. Lucidi and M. Piccioni, "Random tunneling by means of acceptance-rejection sampling for global optimization," *Journal of Optimization Theory and Applications*, vol. 62, no. 2, 1989.

[7] S. Lucidi, F. Rochetich, and M. Roma, "Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization," *SIAM Journal on Optimization*, vol. 8, pp. 916–939, 1998.

[8] S. Lucidi and M. Roma, "Numerical experiences with new truncated Newton methods in large scale unconstrained optimization," *Computational Optimization and Applications*, vol. 7, pp. 71–87, 1997.

[9] W. L. Price, "Global optimization by controlled random search," *Journal of Optimization Theory and Applications*, vol. 40, pp. 333–348, 1983.

[10] W. L. Price, "Global optimization algorithms for a CAD workstation," *Journal of Optimization Theory and Applications*, vol. 55, pp. 133–146, 1987.

[11] W. Price, "A controlled random search procedure for global optimization," in *Towards Global Optimization 2* (L. Dixon and G. Szego, eds.), Amsterdam: North-Holland, 1978.