Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

A Memetic PSO Algorithm for Scalar Optimization Problems

Oliver Schütze and El-ghazali Talbi
INRIA Futurs, LIFL, CNRS Bât M3, Cité Scientifique,
59655 Villeneuve d'Ascq Cedex, FRANCE
Email: {schuetze,talbi}@lifl.fr

Carlos Coello Coello and Luis Vicente Santana-Quintero
CINVESTAV-IPN (Evolutionary Computation Group)
Computer Science Department
Av. IPN No. 2508
Col. San Pedro Zacatenco
México D.F. 07300, MEXICO
Email: ccoello@cs.cinvestav.mx
Email: lsantana@computacion.cs.cinvestav.mx

Gregorio Toscano Pulido
Universidad Autónoma de Nuevo León
Graduate Program in Systems Engineering
Monterrey, MEXICO
Email: gtoscano@gmail.com

Abstract— **In this paper we introduce line search strategies originating from continuous optimization for the realization of the guidance mechanism in particle swarm optimization for scalar optimization problems. Since these techniques are well-suited for—but not restricted to—local search the resulting algorithm can be considered to be memetic. Further, we will use the same techniques for the construction of a new variant of a hill climber. We will discuss possible realizations and will finally present some numerical results indicating the strength of the two algorithms.**

I. Introduction

The first use of the term *Memetic Algorithm* in the computing literature appeared in 1989 in a technical report by Moscato [15]. A memetic algorithm is a heuristic population-based optimization strategy which basically combines local search heuristics with crossover operators. By this reason, some researchers view them as *Hybrid Genetic Algorithms*.
Some real-coded memetic algorithms reported in literature are the following:

a) Hybrid Genetic Algorithms (HGAs): These are hybrid real-coded genetic algorithms which use local improvement procedures (LIPs) (e.g., gradient methods or random hill climbing) on continuous domains to refine the solutions. HGAs apply a LIP to every member of each population, the resulting solutions replace the population members and are used to generate the next population under selection and recombination. A different type of hybridization of LIPs and genetic algorithms concerns the construction of new classes of evolutionary algorithms designed to perform local improvements such as Hart [10], who uses an evolutionary pattern search algorithm.

b) Crossover local search algorithms (XLS): This crossover operator produces children in a neighborhood of the parents. Satoh [17] proposed an algorithm called MGG (minimal generation gap) with generation alternation through the crossover operator. The parents are replaced by (a) the best individual of the parents and their offspring, and (b) by a new individual which is chosen by roulette wheel techniques. In another variant of this algorithm – called G3 (generalized generation gap) and proposed by Deb [5] – the parents are replaced by the roulette-wheel selection with a block selection of the best two solutions. Once a XLS algorithm has found promising areas of the search space, it searches over only a small fraction of the neighborhood around each point.

c) Crossover Hill Climbing: Hill climbing is a local search algorithm that starts from a single solution. At each step, a candidate solution is generated using a move operator. Crossover hill climbing was first described by Jones [11] and O'Reilly [16]. So far, many different variants have been developed. The most representative among them is probably the algorithm proposed by Lozano [13] that maintains a pair of parents and performs repeatedly crossover on this pair until some number of offspring is reached. The best offspring is then selected and replaces the worst parent in case the former has a better fitness.

Line search strategies have been thoroughly studied since several decades and are well-known as a powerful tool for optimization ([2], [6]). Also in the field of Evolutionary Computation these techniques have been integrated since its pioneering days (here we refer to the work of H. Bremermann who already utilized line search strategies in the late 50ies, see [7] for an overview) and are beeing considered and

1-4244-0708-7/07/$20.00 ©2007 IEEE

adapted occasionaly time and again (e.g., [9]).

The update of the location of the particles in a PSO algorithm is typically realized by two mechanisms: a global, stochastic search strategy (the *craziness* which will not be investigated here) and a local search procedure (*guidance*). In the latter case the location of a current particle p is changed by a combination of movements from p towards both the local best position of p and the global best position. These directions can be viewed – in some general and natural sense – as descent directions for the system at the location of p. In this paper we propose to apply line search strategies to perform the guidance efficiently. In most PSO variants the movement is done toward particular points, but does not go beyond them. In these cases the particles surely have a bias to stay inside the convex hull $H(P)$ of the current population P with positions $x_i, i = 1, \ldots, N$:

$$H(P) = \left\{ \sum_{i=1}^{N} \lambda_i x_i \big| \lambda_i \geq 0, 1 = 1, \ldots, N, \text{and} \sum_{i=1}^{N} \alpha_i = 1 \right\},$$

or have to 'wait' for a suitable solution coming from the craziness – which can last very long, in particular in higher dimensional domains. By using line search strategies we aim at the following two benefits due to the adaptive guidance strategy: (a) an improvement of the coarse dynamics of the system and (b) a speedup of the local convergence.
Since in numerous test runs we have obtained particularly good results for small populations, we have also tested the extreme case (i.e., $|P| = 2$) leading to a new hill climber variant which we will also propose below.

An outline of this paper is as follows: in Section II we give the required background for the algorithms which are presented in Section III. In Section IV we present some numerical results. Finally, our conclusions and some possible paths for future research are presented in Section V.

II. Background

Here we present the required background for the algorithms which are presented in the next section. That is, we formulate the problem, address the basic idea of line search, and recall shortly a basic variant of both the hill climber and the PSO algorithm.

d) **Problem Description and Line Search***:* Throughout this article we consider the following *unconstrained optimization problem* (UOP): given a continuous function

$$f : \mathbb{R}^n \to \mathbb{R}$$

the task is to find a point $x^* \in \mathbb{R}^n$ such that

$$f(x^*) \leq f(y) \quad \forall y \in \mathbb{R}^n.$$

There exists a huge variety of very efficient point-wise iterative methods for the localization of (local) minimima of a given UOP. A widely used class of these methods are the so-called *line searchers*. The basic idea is rather simple and can be described as follows (see e.g. [6]):
starting with a point $x_0 \in \mathbb{R}^n$ the subsequent iterates are chosen by the two following steps:

for $k = 0, 1, ...$

- compute a descent direction ν_k
- compute $t_k \in \mathbb{R}_+$ such that $x_{k+1} := k_k + t_k \nu_k$ is an 'acceptable' next iterate

The descent direction can be e.g. chosen as $\nu_k^S = -\nabla f(x_k)$ leading to the steepest descent method or as the *Newton direction* $\nu_k^N - \nabla^2 f(x_k)^{-1} \nabla f(x_k)$ which leads to the (damped) Newton method. The method is called line search since in every step the UOP is replaced by a one-dimensional restriction of f, i.e. to the 'minimization' of

$$\begin{aligned} f_{\nu_k} &: \mathbb{R} \to \mathbb{R} \\ f_{\nu_k}(t) &= f(x_k + t\nu_k) \end{aligned} \tag{II.1}$$

In fact, it is widely accepted that it is not the most efficient way to find the exact minimum of f_{ν_k} in every step k in order to obtain the best overall performance. In practise, the minimization of f_{ν_k} is mostly replaced by the much weaker condition

$$f(x_{k+1}) = f(x_k + t_k \nu_k) < f(x_k), \tag{II.2}$$

which, in turn, does not guarantee convergence of the sequence of the x_k's.
A common way to obtain a good guess for the minimizer of a function f_ν without spending too much time by function calls is to approximate f_ν by a polynomial p which is typically of low degree. The minimum of p – which can be computed exactly without further function calls – is typically an acceptable next iterate in the sense that condition (II.2) is fulfilled, or can at least serve as a (hopefully better) starting point for the next guess. See [6] for a thorough discussion.

e) **Random Hill Climber***:* Here we present the Random Hill Climber (RHC) which is also known as the $(1+1)$-Evolution Strategy ([3]) and which serves as the basis for the algorithm which is presented in the next section.
Given a starting point $x_0 \in \mathbb{R}^n$ and $x_0^b := x_0$ the basic version of the algorithm reads as follows:

for $k = 1, 2. \ldots$

(a) set $x_k^1 := x_{k-1}^b$ and choose x_k^2 at random
(b) choose $x_k^b \in \{x_k^1, x_k^2\}$ such that $f(x_k^b) = \min(f(x_k^1), f(x_k^2))$

The RHC is definitely the simplest form of an evolutionary algorithm since in every step merely two points are taken into account. However, it can often perform competitively with more complex EAs ([14]) and is thus definitely worth to be investigated further on.

f) **Particle Swarm Optimization***:* In PSO, a population of particles is considered ([12]). These particles evaluate the search space by moving with a particular speed towards the best particle found so far (guide) by particular heuristics

including their experience from the past generations.
To be more precise, a general PSO method can be described as follows. A set of N particles is considered as a population P_k in generation $k \in \mathbb{N}_0$. Each particle i has a position $x_{i,k} \in \mathbb{R}^n$ and a velocity $v_{i,k} \in \mathbb{R}^n$ in generation k. These two values are updated in generation $k+1$ by the following two steps:

$$\begin{aligned} v_{i,k+1} &= \omega v_{i,k} + c_1 R_1 (p_{i,k} - x_{i,k}) + c_2 R_2 (p^g_{i,k} - x_{i,k}), \\ x_{i,k+1} &= x_{i,k} + v_{i,k+1}, \end{aligned} \tag{II.3}$$

where $i = 1, \ldots, N$, and

- ω is the *inertia weight* of the particle,
- c_1 and c_2 are positive constants,
- $R_1, R_2 \in [0,1]$ are chosen at random,
- $p_{i,k}$ is the best position found by particle i in the first k steps, and
- $p^g_{i,k}$ is the best position found by all particles in the first k steps.

In order not to restrict the search to the lines which are given by the locations of the particles of the initial generation a stochastic variable called *craziness*[1] is introduced in addition to the movement of the particles (*flight*) described above. One common method is to exchange the current location of the particle with the best position – which is stored separately in $p^g_{i,k}$ – with a randomly chosen location in each iteration step.

III. The Algorithms

In this section we propose a hill climber as well as a PSO variant which involve line search strategies. The common situation in these (and other) algorithms is that in every step there are points $x_0, x_1 \in \mathbb{R}^n$ considered where $f(x_1) < f(x_0)$. Thus, $\nu := x_1 - x_0$ can be viewed as a descent direction[2] for f at the point x_0 and hence in principle line search strategies can be applied. In the following we will present the two algorithms and will then go into detail for a particular realization of the line search.

A. Hill Climber with Line Search

The underlying idea of the classical RHC is to compare two points in every step and to archive the best solution found b_k during the run of the algorithm. In order to apply line search in a reasonable way, we have to avoid too large values for $||\nu||$ and have thus to choose further candidates 'near' b_k. For this, we define the following neighborhood: given a point $c \in \mathbb{R}^n$ and a vector $r \in \mathbb{R}^n_+$ with positive entries we define

$$B(c,r) := \{x \in \mathbb{R}^n \, : \, c_i - r_i \leq x_i \leq c_i + r_i \forall i = 1,..,n\},$$

which can be viewed as an n-dimensional box with center c and radius r.

Given an initial point $x_0 \in \mathbb{R}^n$, a vector of radii $r \in \mathbb{R}^n_+$, and $x^b_0 := x_0$ the *Hill Climber with Line Search* reads as follows:

g) Hill Climber with Line Search:

for $k = 1, 2 \ldots$

(a) set $x^1_k := x^b_{k-1}$ and choose $x^2_k \in B(x^1_k, r)$ at random

(b) set $\tilde{x}^b_k \in \{x^1_k, x^2_k\}$ such that $f(\tilde{x}^b_k) = \min(f(x^1_k), f(x^2_k))$ and the other point as $\tilde{x}^s_k$. Define $\nu_k := \tilde{x}^b_k - \tilde{x}^s_k$.

(c) compute $t_k \in \mathbb{R}_+$ and set $\tilde{x}^n_k := \tilde{x}^s_k + t_k \nu_k$.

(d) choose $x^b_k \in \{\tilde{x}^b_k, \tilde{x}^n_k\}$ such that $f(x^b_k) = \min(f(\tilde{x}^b_k), f(\tilde{x}^n_k))$

The algorithm represents a possible alternative to the PSO algorithm (described below) in particular for local search problems (see e.g. the last example in this paper) or in case the function evaluation is expensive. Possible strategies for the choice of the t_k's in step (c) will be discussed in Section 3.3.

B. PSO with Line Search

Using the notations stated above, the position and the velocity of each particle in generation $k+1$ are updated by the following steps:

compute $t_{i,k,1}, t_{i,k,2} \in \mathbb{R}_+$

$$v_{i,k+1} = \omega v_{i,k} + t_{i,k,1}(p_{i,k} - x_{i,k}) + t_{i,k,2}(p^g_{i,k} - x_{i,k})$$
$$x_{i,k+1} = x_{i,k} + v_{i,k+1}$$

The general formulation of this algorithm is indeed very close to the formulation of the basic variant. A particular realization of the algorithm which includes the following discussion can be found in Algorithm 1.

C. Realization of the Algorithms

As stated above, the situation for the line search is that we are given two points $x_0, x_1 \in \mathbb{R}^n$ where $f(x_1) < f(x_0)$ and the associated 'descent direction' $\nu := x_1 - x_0$ (see Fig. 1). We propose to realize the line search in the following way: choose $e \in (1, 2]$ and compute $f_\nu(e)$. If $f_\nu(e) < f_\nu(1)$ then accept $x_{new} = x_0 + e\nu$ as the next iterate. If the above condition does not hold we have collected enough information to approximate f_ν by a quadratic polynomial $p = at^2 + bt + c$ with coefficients $a, b, c \in \mathbb{R}$. By solving the system of linear equations given by the interpolation conditions

$$\begin{array}{rclcl} \text{I} &:& p(0) &=& 0 \cdot a + 0 \cdot b + 1 \cdot c = f_\nu(0) \\ \text{II} &:& p(1) &=& 1 \cdot a + 1 \cdot b + 1 \cdot c = f_\nu(1) \\ \text{III} &:& p(e) &=& e^2 \cdot a + e \cdot b + 1 \cdot c = f_\nu(e) \end{array}$$

we obtain for the coefficients of p:

$$\begin{aligned} a &= \frac{f_\nu(e) - f_\nu(0) - e(f_\nu(1) - f_\nu(0))}{e^2 - e}, \\ b &= \frac{-f_\nu(e) + f_\nu(0) + e^2(f_\nu(1) - f_\nu(0))}{e^2 - e}, \\ c &= f_\nu(0). \end{aligned}$$

Since $p(1) < p(0)$ and $p(e) \geq p(1)$ and since p is a quadratic polynomial the function contains exactly one minimum at

$$t^* = \frac{-b}{2a} = 2\frac{e^2(f_\nu(1) - f_\nu(0)) - f_\nu(e) + f_\nu(0)}{e(f_\nu(1) - f_\nu(0)) - f_\nu(e) + f_\nu(0)} \in (0, e). \tag{III.4}$$

[1] Also referred as *turbulence* in the specialized literature.

[2] In the sense that there exists a $\bar{t} \in \mathbb{R}_+$ such that $f(x_0 + \bar{t}\nu) < f(x_0)$. Note that this property does not have to be fulfilled initially, i.e. for continuous differentiable functions the condition $\nabla f(x_0)^T \nu < 0$ is not guaranteed.

The interpolants typically serve as a good approximation of f_ν locally, i.e. around $t = 0$ and if $||\nu||$ is small. However, this does not hold globally, in particular for multimodal functions. In order to add a stochastic component to the line search and not to destroy the local property of the interpolants described above, we propose to add a perturbation around t^* where the maximal distance to t^* should be proportional to $||\nu||$, i.e.

$$x_{new} = x_0 + t^*\nu + \frac{||\nu||}{C} r\nu, \qquad \text{(III.5)}$$

where $C \in \mathbb{R}$ is a positive constant and $r \in [-1, 1]$ is chosen at random. Hence, the perturbation vanishes for $||\nu|| \to 0$. Further, we suggest also to choose the value $e \in (1, 2]$ at random in order not to obtain the same setting for the construction of p in every step.

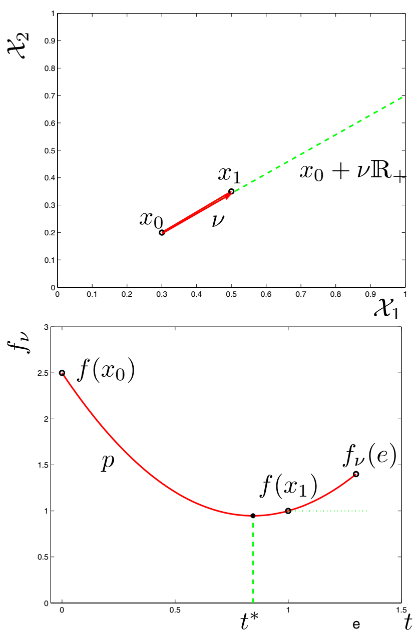

Fig. 1. Approximation of the one-dimensional restriction f_μ of the underlying optimization problem by a quadratic polynomial (see text).

Remark 1. Another possibility for the determination of the quadratic polynomial p is to use the value $p'(1) = f'_\nu(x_1)$ (respectively e.g. an approximation like the forward difference $p'(1) \approx \frac{f_\nu(1+h) - f_\nu(1)}{h}$) as the required third piece of information which leads to

$$t^* = \frac{2(f_\nu(1) - f_\nu(0)) - f'_\nu(1)}{2(f_\nu(1) - f_\nu(0) - f'_\nu(1))}.$$

Using this approach we have obtained even a slightly better performance on some differentiable UOPs compared to the line search described above. However, we decided to propose a gradient free version since this seems to be more natural for the construction of both a hill climber as well as a PSO algorithm.

While it is straightforward to apply the line search on the hill climber, this task needs more consideration for the PSO algorithm. In the latter algorithm we are given in fact *three* descent directions for each particle and for every generation, namely

$$\nu_{i,1} = p_{i,k} - x_{i,k},$$
$$\nu_{i,2} = p^g_{i,k} - x_{i,k},$$
$$\nu_{i,3} = p^g_{i,k} - p_{i,k}.$$

The most greedy search can certainly be obtained by taking only one search direction into account. The inclusion of more search directions for the update of the location of the particles will on one hand surely lead to more diversity among them but will on the other hand lead to more function calls in every step. For the computation of the results which are presented in the next section we have solely used the first strategy, i.e. we have set $t_{i,k,1} = 0$ and have computed $t_{i,k,2}$ via the line search method described above (see also Algorithm 1).

Algorithm 1 Memetic-PSO

$\vec{gbest} \leftarrow \vec{x_0}$
for $i \leftarrow 0, nParticles$ **do** ▷ *Initialize Population and update* g_{best}
  $\vec{gbest} \leftarrow \vec{x_0} \leftarrow initialize\_randomly()$
  $fitness_i \leftarrow f(\vec{x_i})$
  if $fitness_i < f(\vec{g_{best}})$ **then**
    $\vec{g_{best}} \leftarrow \vec{x_i}$
  end if
end for
repeat
  $e \leftarrow U(1.1, 1.7)$ ▷ *Uniformly random number generated in (1.1, 1.7)*
  for $i \leftarrow 0, nParticles$ **do**
    $\vec{p} \leftarrow (\vec{g_{best}} - \vec{x_i})$
    for $k \leftarrow 0, nObjectives$ **do**
      $aux_k \leftarrow (x_{i,k} + e \cdot p_k)$
      $aux_k \leftarrow Check\_bounds(aux_k)$
    end for
    if $f(\vec{aux}) < f(\vec{g_{best}})$ **then**
      $\vec{g_{best}} \leftarrow \vec{aux}$ ▷ *Accept* $\vec{aux}$
    else ▷ *Interpolate*
      $a \leftarrow \frac{(f(\vec{aux}) - fitness_i) - e \cdot (f(\vec{g_{best}}) - fitness_i)}{(e^2 - e)}$
      $b \leftarrow f(\vec{g_{best}}) - fitness_i - a$
      $t \leftarrow -\frac{b}{2 \cdot a}$
      for $k \leftarrow 0, nObjectives$ **do**
        $aux_k \leftarrow x_{i,k}$
        $x_{i,k} \leftarrow x_{i,k} + U\{(t - 0.5 \cdot e), (t + 0.5 \cdot e)\}$
        $x_{i,k} \leftarrow Check\_bounds(x_{i,k})$
      end for
      $fitness_i \leftarrow f(\vec{x_i})$
      if $fitness_i = f(\vec{g_{best}})$ **then**
        $x_i \leftarrow turbulence(x_i)$ ▷ *Create new variables for* x_i
        $fitness_i \leftarrow f(\vec{x_i})$
      end if
    end if
    if $fitness_i < f(\vec{g_{best}})$ **then**
      $\vec{g_{best}} \leftarrow \vec{x_i}$
    end if
  end for
until Termination Criteria

IV. Numerical Results

In this section we illustrate the efficiency of the two algorithms by validating them on several examples.

A. Results for Memetic-PSO

First we turn our attention to the Memetic-PSO. In order to compare its performance we have chosen thirteen different test problems taken from [8] with different geometrical characteristics: functions f_1 to f_5 are unimodal functions, f_6 is a step function and thus dicontinuous, f_7 is a noisy quartic function and functions f_8 to f_{13} are multimodal functions (see Table I). In order to validate our proposed approach, our results are compared with respect to those generated by the *FEP* (Fast Evolutionary Programming) proposed in [19], which is an algorithm representative of the state-of-the-art in the area. The average results of 50 independent runs are shown in Table II.

B. Results for the Hill Climber with Line Search

Next we want to evaluate the perfomance of the novel hill climber. The choice of the appropriate set of test functions is not too easy in this case: if multimodal functions are taken, the result of the optimization will be highly dependent on the initial guess, and presumably be worse than results coming from population-based methods. If on the other hand functions are taken which are 'easy' in the context of optimization (e.g., convex functions), the outcome can also in this case be predicted quite easily. We have chosen for two unimodal functions which are not too easy to handle in order to test the hill climber for its primal task: black box local search. To be more precise, we consider the following two UOPs:

$$f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$$

$$f_1(x) := \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \qquad \text{(Schwefel's function)}$$

$$f_2(x) := \sum_{i=1}^{n} (x_i^2 + \text{random}[-0.01, 0.01]) \text{ (Quadratic + Noise)} \tag{IV.6}$$

The minimum of both functions is $x^* = (0, \ldots, 0) \in \mathbb{R}^n$. We have compared the performance of the Hill Climber with Line Search (HCLS) with the Random Hill Climber (RHC), the downhill simplex method of Nelder and Mead (NM, the function `fminsearch` of MATLAB[3]), and a derivative-free Quasi-Newton method (QN, the function `E04JYF` of the NAG[4] library) on these two functions. As the starting point we have chosen $x_0 = (2, 3, 2, 3, \ldots) \in \mathbb{R}^n$ and have set $Q = [-5, 5]^n$ as the domain (for RHC). Every computation was terminated as successful when a point x_k with $|f_i(x_k)| < 0.1$ was found and terminated as unsuccessful if such a point was not found within 10^7 function calls. Tables III and IV show the average result of 20 test runs. The results indicate that the new solver can compete with the other well-known and widely accepted black box optimizer at least on this (small) set of benchmark functions.

TABLE III
PERFORMANCE OF THE HILL CLIMBER WITH LINE SEARCH ON FUNCTION f_1 (SEE (IV.6)) AND COMPARISON TO OTHER ALGORITHMS: THE DOWNHILL SIMPLEX METHOD OF NELDER AND MEAD (NM), A DERIVATIVE-FREE QUASI-NEWTON METHOD (QN) AND THE RANDOM HILL CLIMBER (RHC). # FC DENOTES THE NUMBER OF FUNCTION CALLS AND $|f(x_{end})|$ THE FUNCTION VALUE OF THE BEST FOUND SOLUTION (AVERAGE OF 20 TEST RUNS).

Method		QN	NM	RHC	HCLS
$n = 5$	# FC	659	502	$1.0 \cdot 10^5$	188
	$\|f(x_{end})\|$	0	6.5	0.084	0.084
$n = 10$	# FC	2465	448	$8.8 \cdot 10^5$	638
	$\|f(x_{end})\|$	0	21.8	0.094	0.089
$n = 20$	# FC	3847	1471	$7.4 \cdot 10^6$	2551
	$\|f(x_{end})\|$	0	$45,4$	0.098	0.095
$n = 50$	# FC	2222	8336	$1.0 \cdot 10^7$	$2.5 \cdot 10^5$
	$\|f(x_{end})\|$	0	124.6	1.15	0.095
$n = 100$	# FC	$n.a.$	$2.0 \cdot 10^5$	$1.0 \cdot 10^7$	$1.4 \cdot 10^6$
	$\|f(x_{end})\|$	$n.a.$	$2.7 \cdot 10^8$	8.13	0.099

TABLE IV
PERFORMANCE OF THE HILL CLIMBER WITH LINE SEARCH ON FUNCTION f_2 (SEE (IV.6)) AND COMPARISON TO OTHER ALGORITHMS. THE NOTATION IS THE SAME AS IN TABLE III.

Method		QN	NM	RHC	HCLS
$n = 5$	# FC	970	358	1226	115
	$\|f(x_{end})\|$	2.3	0.013	0.058	0.069
$n = 10$	# FC	2678	$5.8 \cdot 10^6$	$1.2 \cdot 10^5$	241
	$\|f(x_{end})\|$	1.81	0.3797	0.087	0.085
$n = 20$	# FC	8049	$1.0 \cdot 10^7$	$6.8 \cdot 10^5$	460
	$\|f(x_{end})\|$	0.28	11.39	1.24	0.095
$n = 50$	# FC	6546	$1.0 \cdot 10^7$	$9.2 \cdot 10^7$	1371
	$\|f(x_{end})\|$	0.34	17.38	0.108	0.096
$n = 100$	# FC	$1.2 \cdot 10^5$	$1.0 \cdot 10^7$	$1.0 \cdot 10^7$	3361
	$\|f(x_{end})\|$	0.14	39.67	1.35	0.097

C. An Application: Computing Solution Sets of Nonlinear Equations

Finally, we consider a problem where the Hill Climber with Line Search can be very helpful, namely the computation of the solution sets $H^{-1}(0)$ of a given (non-differentiable) function

$$H : \mathbb{R}^{N+K} \to \mathbb{R}^N.$$

Problems of this kind can e.g. arise in multi-objective optimization.

Given a point $x_0 \in H^{-1}(0)$ one possibility to find further solutions in the neighborhood of x_0 is to use *continuation methods* (see [1] for an overview of existing methods), e.g. the one proposed in [18]. This method transforms the original problem via so-called predictor-corrector strategies into a sequence of UOPs of the form

$$\min_x \|H(x)\|. \tag{IV.7}$$

[3] `http://www.mathworks.com`
[4] `http://www.nag.com`

TABLE I

TEST FUNCTIONS

Name	Test Problem	n	Search Space	optimum
Sphere model	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	0
Schwefel's Problem 2.22	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]^n$	0
Schwefel's Problem 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} x_j \right)^2$	30	$[-100, 100]^n$	0
Schwefel's Problem 2.21	$f_4(x) = max_i(\lvert x_i \rvert, 1 \leq i \leq n)$	30	$[-100, 100]^n$	0
Rosenbrock's Function	$f_5(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \right]$	30	$[-30, 30]^n$	0
Step Function	$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]^n$	0
Quartic Function (noise)	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^n$	0
Schwefel's Problem 2.26	$f_8(x) = \sum_{i=1}^{n} -x_i \sin \sqrt{\lvert x_1 \rvert}$	30	$[-500, 500]^n$	-12569.5
Rastrigin's Function	$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	30	$[-32, 32]^n$	0
Ackley's Function	$f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos 2\pi x_i \right)$ $+20 + e$	30	$[-32, 32]^n$	0
Griewank Function	$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	$[-600, 600]^n$	0
Penalized Function	$f_{12}(x) = \frac{\pi}{n} \{10 \sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})]$ $+(y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4).$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	30	$[-50, 50]^n$	0
Penalized Function	$f_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_1 - 1)^2 [1 + \sin^2(3\pi x_{i+1})]$ $+(x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4).$	30	$[-50, 50]^n$	0

TABLE II

COMPARISON OF MEMETIC-PSO AND FEP ON SEVERAL TEST FUNCTIONS (SEE TABLE I). THE RESULTS OBTAINED BY THE LATTER ALGORITHM ARE TAKEN FROM [8]. THE MEMETIC-PSO STOPPED IF THE OPTIMUM WAS REACHED OR A MAXIMUM NUMBER OF FUNCTION CALLS PERFORMED BY THE FEP ALGORITHM WAS REACHED. THE NUMBERS IN BOLDFACE MARK THE BEST RESULT.

		Memetic - PSO				F E P		
Function	Optimum	Mean Evaluations	Reach optima 50 runs	Mean Best	Std Dev	Eval	Mean Best	Std Dev
f_1	0	**7,917**	**50**	**9.1e-5**	**1.27e-5**	150,000	5.7e-4	1.3e-4
f_2	0	**15,462**	**50**	**9.51e-5**	**6.2e-6**	200,000	8.1e-3	7.7e-4
f_3	0	**63,599**	**50**	**9.91e-3**	**1.12e-4**	500,000	1.6e-2	1.4e-2
f_4	0	**72,869**	**50**	**9.84e-3**	**2.36e-4**	500,000	0.3	0.5
f_5	0	**1,076,309**	**40**	**1.18**	**3.054**	2'000,000	5.06	5.87
f_6	0	**53,072**	**50**	**0**	**0**	150,000	0	0
f_7	0	**1,952**	**50**	**8.9e-5**	**1.18e-5**	300,000	7.6e-3	2.6e-3
f_8	-12569.5	855,000	0	-10,056	430.7	**900,000**	**-12,554**	**52.6**
f_9	0	500,000	0	16.23	4.54	**500,000**	**4.6e-2**	**1.2e-2**
f_{10}	0	**27,205**	**50**	**9.6e-5**	**5.19e-5**	150,000	1.8e-2	2.1e-3
f_{11}	0	**114,403**	**24**	**2.72e-2**	**2.12e-2**	200,000	1.6e-2	2.2e-2
f_{12}	0	**17,751**	**50**	**9.21e-7**	**1.0e-7**	150,000	9.2e-6	3.6e-6
f_{13}	0	**9,814**	**50**	**8.89e-5**	**1.28e-5**	150,000	1.6e-4	7.3e-5

In case H is not differentiable, e.g. the Hill Climber with Line Search (as well as in principle every other derivative-free minimization algorithm) can be used in the corrector step. Note that in this context a local solver is required for a good performance of the continuation method.

As an (academic) example we consider the problem of finding all the points $x \in \mathbb{R}^3$ where $\|x\|_\infty = 1$ and $x_3 \leq 0.5 \sin(2\pi \min(|x_1|, |x_2|))$ holds. Thus, we are interested in the set $H^{-1}(0)$, where

$$H : \mathbb{R}^4 \to \mathbb{R}^2$$

$$H(x, t) = \begin{pmatrix} \|x\|_\infty - 1 \\ x_3 - 0.5 \cos(2\pi \min(|x_1|, |x_2|)) + t^2 \end{pmatrix} \tag{IV.8}$$

Figure 2 shows the result of the continuation method. Here, the entire solution set was obtained by starting with one single point $(x_0, t_0) = (-1, -1, 0.5, 0) \in H^{-1}(0)$. During the run of the algorithm, an amount of 11484 solutions was produced, i.e. all in all the total number of 68904 UOPs of the form (IV.7) were solved successfully. The computations have been done on an Intel Xeon 3.2 GHz processor and have taken approximately 30 seconds. This results indicates that the Hill Climber with Line Search is well-suited to be used in combination with a continuation method.

V. CONCLUSIONS AND FUTURE WORK

We have presented new variants of a PSO algorithm and a hill climber by involving line search strategies. These techniques allow both for the improvement of the coarse dynamics of the system (of particles) as well as for a speedup of its local convergence. We have demonstrated the strength of the algorithms by several numerical results.

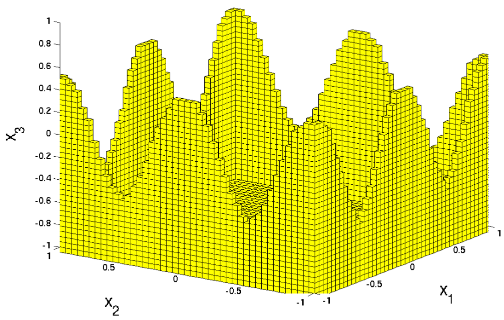

Fig. 2. Computation of an implicitly defined set of an underlying non-differentiable function by a continuation method where the Hill Climber with Line Search was taken for both the predictor step and for the corrector step (for details of the algorithm we refer to [18]).

In the future, we intend to extend the techniques presented in this paper. One particularly intersting extension would be the development of adaptive constraint handling techniques since so far the treatment of those problems – even optimization problems with box constraints – with the methods proposed above is not satisfacory. Further, we think of using and adapting the method proposed in this paper for the construction of multi-objective particle swarm optimization algorithms. In particular the example in Section IV-C motivates for further research in this direction. In this context, a combination with the procedure described in [4] seems to be very promising.

h) Acknowledgements: The third author gratefully acknowledges support from CONACyT through project 45683-Y.

REFERENCES

[1] E. L. Allgower and K. Georg. *Numerical Continuation Methods*. Springer, 1990.

[2] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[3] H.-G. Beyer. Toward a theory of evolution strategies: The (μ, λ)-theory. *Evolutionary Computation*, 2 (4):381–407, 1994.

[4] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.

[5] Kalyanmoy Deb, Ashish Anand, and Dhiraj Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, 10(4):371–395, 2002.

[6] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[7] David B. Fogel. *Evolutionary Computation: The Fossil Record.* Wiley-IEEE Press, 1998.

[8] Ashish Ghosh and Shigeyoshi Tsutsui, editors. *Advances in Evolutionary Computing (Theory and Applications)*. Springer-Verlag, Berlin, 2003. ISBN 3-540-43330-9.

[9] A. Ghozeil and D. B. Fogel. A preliminary investigation into directed mutation in evolutionary algorithms. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 329–335. Springer, Berlin, 1996.

[10] W. E. Hart. Evolutionary pattern search algorithms for unconstrained and linearly constrained optimization. *Evolutionary Computation*, 5:388–397, August 2001.

[11] Terry Jones. Crossover, macromutation, and population-based search. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 73–80, San Francisco, CA, 1995. Morgan Kaufmann.

[12] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia*, pages 1942–1948. IEEE Service Center, Piscataway, NJ, 1996.

[13] Manuel Lozano, Francisco Herrera, Natalio Krasnogor, and Daniel Molina. Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(3):273–302, September 2004.

[14] M. Mitchell, J. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing? In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 51–58. Morgan Kaufman, San Mateo, CA, 1994.

[15] Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P 826, California Institute of Technology, Pasadena, CA, 1989.

[16] Una-May O'Reilly and Franz Oppacher. Hybridized crossover-based search techniques for program discovery. In *Proceedings of the 1995 World Conference on Evolutionary Computation*, volume 2, pages 573–578, Perth, Australia, 29 - 1 1995. IEEE Press.

[17] Hiroshi Satoh, Masayuki Yamamura, and Shigenobu Kobayashi. Minimal generation gap model for gas considering both exploration and exploitation. In *Iizuka '96*, volume 2, pages 494–497, Fukuoka, Japan, 1996.

[18] O. Schütze, A. Dell'Aere, and M. Dellnitz. On continuation methods for the numerical treatment of multi-objective optimization problems. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Ralph E. Steuer, editors, *Practical Approaches to Multi-Objective Optimization*, number 04461 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/349>.

[19] Xin Yao, Yong Liu, and Guangming Liu. Evolutionary programming made faster. *IEEE Trans. on Evolutionary Computation*, 3(2):82–102, 1999.