Meta-heuristics using the simplex algorithm for non-linear programming

João Pedro PEDROSO

Centro de Investigação Operacional
Faculdade de Ciências da Universidade de Lisboa
and
Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
R. Campo Alegre 823, 4150-180 Porto, Portugal
jpp@ncc.up.pt

Abstract— In this paper we present a meta-heuristic for non-linear programming, based on the Nelder and Mead simplex algorithm. The algorithm proposed is suitable for both unconstrained and constrained optimisation. We explore several possibilities for escaping local optima.

I. Introduction

The usage of the Nelder and Mead's simplex algorithm for non-linear optimisation as a part of a meta-heuristic is not new; several authors have suggested the inclusion of simulated annealing ideas in it[6, 2]. More recently, it has been used as a basis for applying the scatter search method [5]. The results reported in the literature are generally very encouraging.

We suggest further improvements to the simplex algorithm, in particular for tackling constraints, as most of the non-linear problems (NLP) with practical interest are constrained problems. Previously published works used penalties on infeasible solutions as a way of handling constrained problems.

In this work we propose a scheme for classifying the solutions of an NLP that takes into account both feasibility and the objective value. Based on it, we are able to sort the points of the simplex even in the case where some points are feasible and other points are not. This makes possible determining through which point of the simplex should reflection, expansion, or contraction be done, thus allowing straightforward application of the Nelder and Mead's algorithm.

The simplex algorithm is a local search method, and does not encompass a procedure for escaping local optima. Random-start iterated local search provides a simple way for avoiding local optima. We investigate on the performance of the modified simplex algorithm, as well as iterated local-search based on it. Finally, we propose additional procedures for trying to get away from local minima.

II. The simplex method

The simplex method for NLP is a very simple and elegant algorithm for finding the minimum of a function in a multi-dimensional space. It works based only on function evaluations, hence does not require information concerning the gradient of the function.

In an N-dimensional space, a simplex is a polyhedron with exactly $N + 1$ vertices (in general, we will be interested in non-degenerate simplexes, which have a finite volume in its space).

The algorithm starts with a simplex, and evaluates its vertices. It will then, at each iteration, try to get rid of it's worst point, by applying the following steps:

1. Reflect the simplex into the opposite side of the worst point.

2. If the reflection led to a point which is better than the simplex's best point, then try expanding further in that direction.

3. If the reflection led to a point which is worse than the second-worst point, then contract the simplex in one-dimension, moving the worst point in the direction of the simplex's centroid.

4. If this contraction did still not improve the simplex's second-worst point, then do a multi-dimensional contraction, by moving all the points in the direction of the best point.

These steps are repeated until a specified stopping criterion is satisfied. The more frequently used criteria are based on the slack between the best and the worst point, on the distance the simplex's centroid moved, or on the number of function evaluations.

For a more complete description of the method and a computational implementation, please refer to [6].

III. Classification of NLP solutions

Notice that the simplex algorithm does not require the actual value of function evaluation at each of the points; all that is required in to *order* the simplex's points, for determining through which vertex should reflection, expansion, or contraction be done. Generally, points are ordered according to the value of the objective function, possibly added to a penalty, if the problem is constrained and the solution is infeasible.

We propose that the comparison of solutions should be based on the deviation from feasibility, as well as on the objective value. For two different solutions x and y, x is said to improve y if and only if:

- x is closer to feasibility than y
- they are both feasible, or identically close to feasibility, and the objective value of x is better than that of y

Based on this classification scheme, we are able to order the points of the simplex, even if some points are feasible and other not, and directly apply the algorithm. The measure of infeasibility is given by the sum of constraint violations (including bound's violation).

IV. Escaping local optima

As the simplex method uses downhill movements, its solution will in general be a local optimum. If we wish to overcome this drawback, and be able to potentially obtain the global optimum of an NLP, we have to provide and escape mechanism.

The strategies that we describe for escaping are based on a restart criterion ϵ, and a stopping criterion M. Both of these are user-supplied values.

Restart will occur if the vertices of the simplex have all evaluations which are feasible (or all infeasible), and the deviation between the objective (resp., the infeasibility) of the best and worst vertices is bellow ϵ. We use absolute deviations if the objectives are close to zero (less than ϵ), relative deviations otherwise.

All of the methods will stop if the number of evaluations has reached the limit M.

A. Iterated simplex

This method consists of restarting the algorithm from a random solutions every time there is convergence of the simplex according to the criterion ϵ, until the maximum number of evaluations M is reached.

B. Directional escape

Another possibility for escaping is the following. When the simplex has converged according to the criterion ϵ, start expanding the simplex through its best vertex (always updating the ranking among the vertices). Expansion will initially decrease the quality of the point; but after a certain number or repetitions, we will reach a local *maximum*, and the expansion will lead to an improvement. We propose to expand until the worst point of the simplex has been *improved*. At that point, we expect to be on the other side of the hill; hence, if we restart the simplex algorithm from that point, we expect to reach a different local optimum. We also restart if the bound has been crossed, using the first point outside bounds to initialise the simplex.

We have tested two possibilities for reinitialisation. The first, corresponds to the usual way of initialising the simplex: it consists of adding a step λ, independently, to each of the coordinates of x^0:

$$x^i = x^0 + \lambda e^i (u - l) \tag{1}$$

where e^i are the unit vectors of dimension N ($e^i_i = 1$, $e^i_j = 0 \quad \forall j \neq i$), and u and l are the upper and lower bound vectors. We called this strategy *escape*.

The other possibility is to complete the simplex with points x^i, for $i > 0$, drawn randomly with uniform distribution between l and u. This strategy was called *escape+random*.

V. Computational results

For the evaluation of the strategies that we proposed in this paper, we have relied on a set of multi-modal test functions which include constrained and unconstrained problems. (Benchmarks which were maximisation problems were converted into minimisations.)

Problem 1: Griewank's function ($d = 4000$).

$$f_1(x) = \frac{1}{d}\sum_{i=1}^{N}(x_i - 100)^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1$$
$$x_i \in [-600, 600] \quad i = 1, \ldots, N$$

Problem 2: Shekel's foxholes ($m = 30$; c_i, $A(i)$ available in [1]).

$$f_2(x) = -\sum_{j=1}^{m}\frac{1}{||x - A(j)||^2 + c_j}$$
$$x_i \in [0, 10] \quad i = 1, \ldots, N$$

Problem 3: Michalewicz' function ($m = 10$).

$$f_3(x) = -\sum_{i=1}^{N}\sin(x_i)\cdot\sin^{2m}\left(\frac{i \cdot x_i^2}{\pi}\right)$$
$$x_i \in [0, \pi] \quad i = 1, \ldots, N$$

Problem 4: Langerman's function. ($m = 30$; c_i, $A(i)$ available in [1]).

$$f_4(x) = -\sum_{j=1}^{m} c_j \cdot e^{-\frac{||x - A(j)||^2}{\pi}} \cdot \cos(\pi \cdot ||x - A(j)||^2)$$
$$x_i \in [0, 10] \quad i = 1, \ldots, N$$

Problem 5: Crescent function ($N = 2$) [4].

$$f_5(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

$$\text{subject to :} \quad (x_1-5)^2+(x_2-5)^2 \geq 100$$
$$(x_1-6)^2+(x_2-5)^2 \leq 82.81$$
$$x_1 \in [13,100], x_2 \in [0,100]$$

Problem 6: Luus's function ($N=3$) [2].

$f_6(x) = x_1^2 + x_2^2 + x_3^2$

subject to :

$4(x_1-0.5)^2 + 2(x_2-0.2)^2 + x_3^2 + 0.1x_1x_2 + 0.2x_2x_3 \leq 16$

$2x_1^2 + x_2^2 - 2x_3^2 \geq 2$

$x_i \in [-2.3, 2.7] \quad i = 1,2,3$

Problem 7: Keane's function.

$$f_7(x) = -\frac{\left|\sum_{i=1}^{N}\cos^4(x_i) - 2\prod_{i=1}^{N}\cos^2(x_i)\right|}{\sqrt{(\sum_{i=1}^{N} i x_i^2)}}$$

$$\text{subject to :} \quad \prod_{i=1}^{N} x_i \geq 0.75$$
$$\sum_{i=1}^{n} x_i \leq 15N/2$$
$$x_i \in [0,10] \quad i = 1,\ldots,N$$

Problem 8: Polygon model [3]. (The actual number of variables is $2N$.)

$$f_8(x,y) = -\frac{1}{2}\sum_{i=1}^{N-1} x_{i+1}x_i \sin(y_{i+1}-y_i)$$

subject to :

$x_i^2 + x_j^2 - 2x_ix_j\cos(y_i - y_j) \leq 1,\ i = 1,...,N, j = i,...,N$

$y_i \leq y_{i+1}, \quad i = 1,\ldots,N$

$x_i \in [0,1], \quad y_i \in [0,\pi] \quad i = 1,\ldots,N$

For the benchmarks which admit choosing the dimension of the problem, we have set N=10. In all the runs a random solution x^0 in the box defined by the problem bounds was used as the first vertex of the initial simplex. The remaining vertices x^i, $i = 1,\ldots,N$, were obtained by equation 1. In this experiment we have set $\lambda = 1$. This implies that all the points except x^0 will be infeasible, as they will be out of the bounding boxes. Computational experiments have shown that this improves the overall performance of all the tested methods. For smaller steps, the simplex would soon be trapped in a (generally poor) local optimum.

For each of the methods, the initial solution is different from run to run; but for a given run, all the methods will start on the same solution. This explains why the performance curves presented below are all identical during the initial steps (until the first restart).

In this experiment the maximum number of function evaluations allotted to each method were $M = 100000$. For all the methods except the pure simplex method, we established a criterion $\epsilon = 10^{-4}$ for stopping the current downhill search. This implies that when the deviation between the objective of the best and the worst point of the simplex is less than that value, the iterated simplex and the escape strategies will restart on a different solution. The pure simplex method will continue exploiting that local optimum, until reaching M evaluations.

The only performance index considered is the value of the best evaluation as a function of the number of evaluations. That value, averaged on 100 independent runs, is plotted on figures 1 and 2. For constrained problems, lines were plot after *all* the 100 runs obtained feasible solutions (and hence it was possible to average the objective values).

VI. Conclusions

In this paper we presented an extension of the simplex method for non-linear programming which allows its straightforward application to constrained problems.

For avoiding stagnation in local optima, we analysed the behaviour of iterated application of the simplex method, as well as other escape mechanisms. These were based on expanding the simplex from the local minimum, going uphill, until the expansion goes downhill again. At that point, we expect to be on the other side of the hill, and restarting simplex descent will likely lead to a different local optimum.

The numerical experiments have shown that all the escaping mechanisms were effective for avoiding stagnation in local optima. Due to the simplicity of its implementation, iterated local search might be the preferred method.

References

[1] H. Bersini, M. Dorigo, L. Gambardella, S. Langerman, and G. Seront. First international contest on evolutionary optimization, 1996. In IEEE International Conference on Evolutionary Computation.

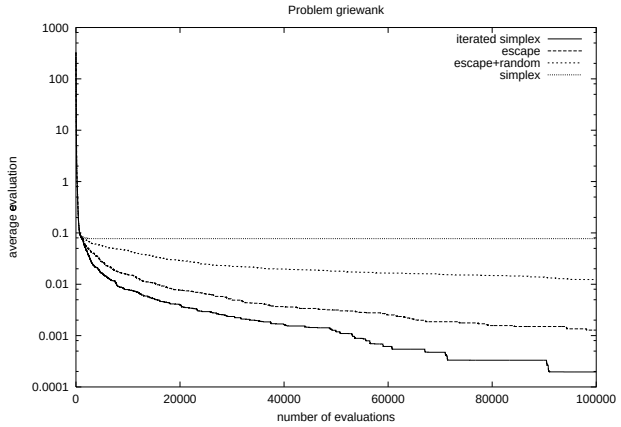
[2] M. F. Cardoso, R. L. Salcedo, and S. F. Azevedo. The simplex-simulated annealing approach to continuous non-linear optimization. *Computers Chemical Engineering*, 20(9):1065–1080, 1996.

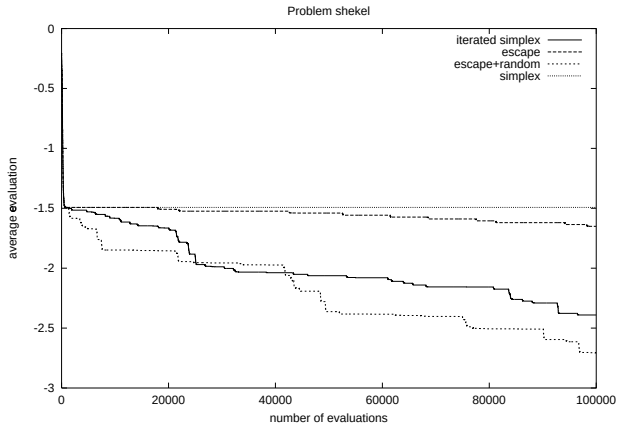
[3] E. D. Dolan and J. J. Moré. Benchmarking optimization software with COPS. Technical Report ANL/MCS-246, Argonne National Laboratory, 2001.

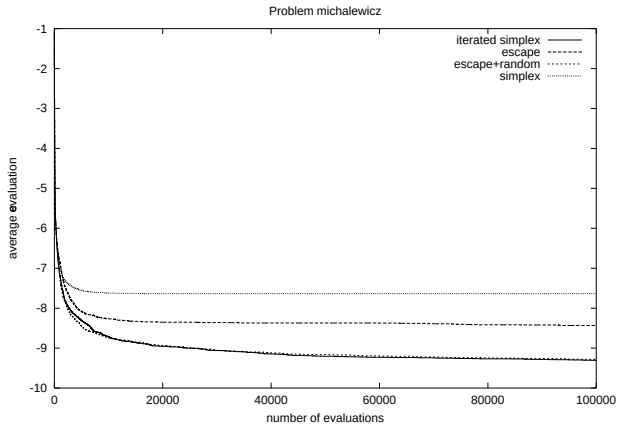
[4] C. A. Floudas and P. M. Pardalos. *Recent Advances in Global Optimization.* Princeton University Press, 1992.

[5] F. Glover, M. Laguna, and R. Martí. Scatter search. Technical report, Graduate School of Business Administration, University of Colorado, 2000.

[6] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipies in C: the Art of Scientific Computing.* Cambridge University Press, second edition, 1997.







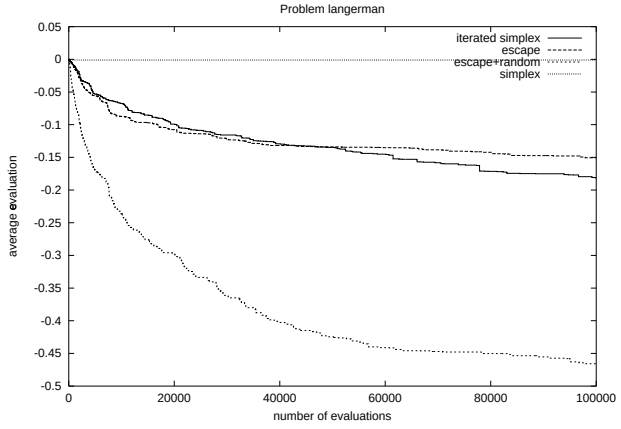
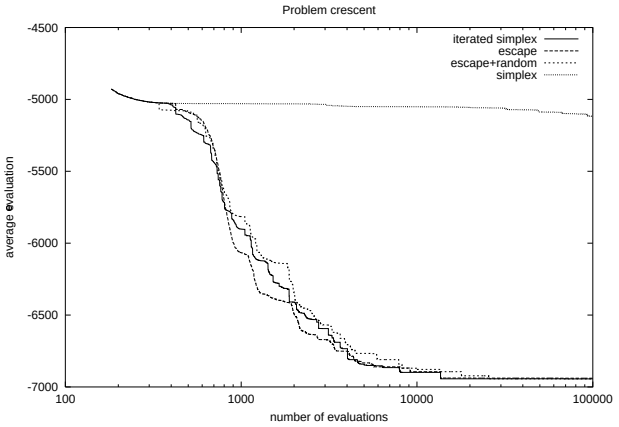
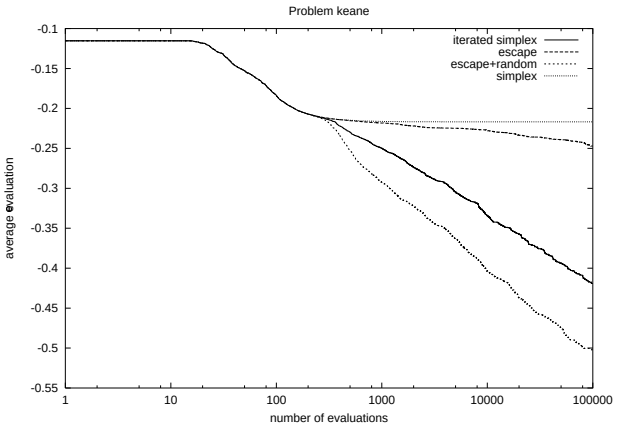
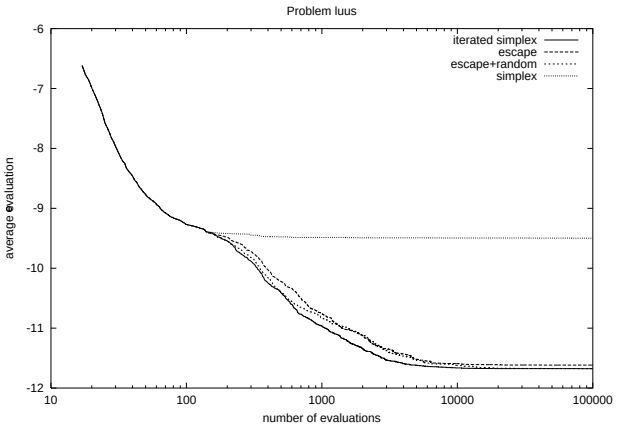


Figure 1: Performance on unconstrained benchmarks.







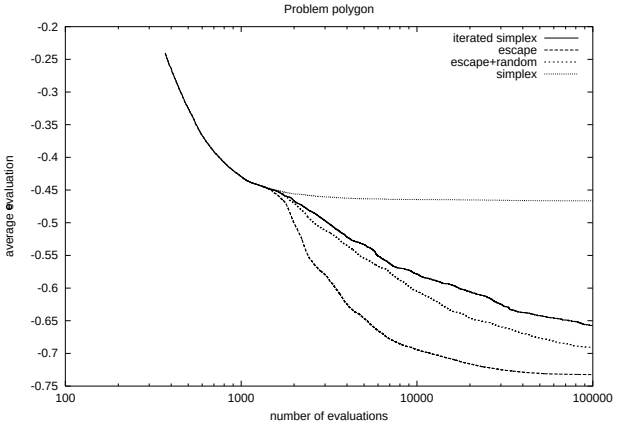


Figure 2: Performance on constrained benchmarks.