Simple meta-heuristics using the simplex algorithm for non-linear programming

João Pedro PEDROSO

INESC - Porto and
DCC - Faculdade de Ciências, Universidade do Porto, Portugal
jpp@fc.up.pt

Abstract. In this paper we present an extension of the Nelder and Mead simplex algorithm for non-linear programming, which makes it suitable for both unconstrained and constrained optimisation[1]. We then explore several extensions of the method for escaping local optima, and which make it a simple, yet powerful tool for optimisation of nonlinear functions with many local optima.
A strategy which proved to be extremely robust was random start local search, with a correct, though unusual, setup. Actually, for some of the benchmarks, this simple meta-heuristic remained as the most effective one. The idea is to use a very large simplex at the begin; the initial movements of this simplex are very large, and therefore act as a kind of filter, which naturally drives the search into good areas.
We propose two more mechanisms for escaping local optima, which, still being very simple to implement, provide better results for some difficult problems.

1 Extensions to the simplex method

Nelder and Mead's algorithm for non-linear programming [2] is a local search method for finding a minimum of a function, based on the movements of a simplex in a multi-dimensional space. These movements rely on function evaluations, and do not require information concerning the gradient of the function. Points of the simplex 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. However, the algorithm does not require the actual value of function evaluation at each of the points; all that is required is to *order* the simplex's points, for determining through which vertex should reflection, expansion, or contraction be done. This is a common characteristic to all direct search methods [3].

The problem dealt with in this paper is characterised by a nonlinear function of a vector x, that we want to optimise. This function is usually multimodal, and in many cases non-smooth.

We assume that there are box constraints, i.e., for a problem of dimension N there will be constraints:

$$l_i \leq x_i \leq u_i \quad i = 1, \ldots, N. \tag{1}$$

[1] An extended version of this paper is available in [1].

If in addition to the box constraints there are P more general constraints in the form $g_p(x) \leq 0$, for $p = 1, \ldots, P$, then the total constraint violation for a solution x can be assessed by

$$\delta(x) = \sum_{p=1}^{P} \max(g_p(x), 0) + \sum_{i=1}^{N} [\max(x_i - u_i, 0) + \max(l_i - x_i, 0)]. \qquad (2)$$

The comparison of solutions can be based on this value, as well as on the objective value. For two different solutions x and y, x improves y if and only if $\delta(x) < \delta(y)$, or $\delta(x) = \delta(y)$ 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 simplex algorithm to constrained problems. Notice that equality constraints are poorly dealt by this modified method. The simplex will probably converge into a point which is on the surface defined by the equality, but will likely have much trouble for moving on that curve, in order to improve the value of the objective, without increasing infeasibilities. The classification system was used in [4]; a more elaborate method would be the filter method proposed in [5].

A random solution for a given instance is an N-dimensional vector that can be drawn as

$$x_i = \mathscr{U}(l_i, u_i) \quad i = 1, \ldots, N, \qquad (3)$$

where we denote a random number with uniform distribution in the interval $[a, b]$ as $\mathscr{U}(a, b)$. One possibility for using the Nelder and Mead algorithm is to start from a random point, as determined by Equation 3. Using the solution classification method described above, the search can start with a very large simplex, as this method tackles the possibility of getting out of bounds. Hence, we propose a setting were the initial step is very large: all the points except the initial random point will be out of bounds. Computational experiments have shown that this setup improves the overall performance of the simplex search; for smaller steps, the simplex would soon be trapped in a (generally poor) local optimum.

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

2.1 Random-start iterated simplex

This method consists of restarting the algorithm from a random solution every time there is convergence of the simplex according to the criterion ϵ, until the maximum number of evaluations M is reached. At each iteration, a random point is drawn and the simplex is reinitialised from that point (with a large step). Whenever the best found solution is improved, a new local search is performed with a smaller stopping criterion ϵ', for refining this local optimum.

The algorithm returns the best solution found on all the iterations.

2.2 Directional escape

Another possibility for escaping local optima 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 *pessimum*, and the subsequent 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.

After an escape point is determined, the simplex is reinitialised around it by adding a large step independently to each of its coordinates. We called this strategy *escape*.

This strategy has the nice property of requiring no additional parameters.

2.3 Tabu search

Tabu search for non-linear programming is not a precisely defined concept, as there is not a commonly used notion of *tabu* in this context. Actually, if the tabu concept is related to the kind of movement that is done, the escape mechanism described in the previous section can be considered as a tabu search: after a local optimum is reached, only expansion of the simplex is considered non-tabu.

In this section we propose a different concept: that of tabu based on the region of space being searched (as proposed also, for example, in [6]).

As we will shortly see, for tabu search to work in our setting it will have to be significantly modified, compared to the more usual tabu search in combinatorial optimisation.

Tabu solutions: classification. A trivial extension of the method devised for solution classification described on section 1 consists of associating a tabu value to each solution, and then using this value as the primary key for solution sorting. In this context, for two different solutions x and y, x is said to improve y if and only if:

– x has a smaller tabu value than y;
– both have similar tabu values, and x has a smaller sum of constraint violations than y (i.e., $\delta(x) < \delta(y)$);
– both are feasible and not tabu, and the objective value of x is better than that of y.

Tabu regions. The most straightforward way of implementing a tabu search for continuous, non-linear problems is that of making the region around a local optimum (obtained by the Nelder and Mead algorithm) a tabu region. This way, for the tenure of the tabu status, we are sure that the search will not fall into the same local optimum.

This strategy, however, did not lead to good results, for the benchmarks used in this paper. We have tested many different approaches on this method, all of them with no success. The main reasons for this are related to the size of the tabu region: if it is too narrow, the search tends to find local optima on the border between tabu and non-tabu regions; on the other hand, if the region is too large, good local optima around the current solution are missed. This difficulty in parameterisation, and the observation that search around local optima is frequently essential to find better solutions, lead us to give up true tabu search, and try the opposite strategy: non-tabu search.

2.4 Inverting tabu regions: non-tabu search

As all the strategies that assigned a tabu status to the region of the last found local optima failed, we deduced that this tabu status barred the search from good regions, resulting in poor performance.

It is therefore expectable that for a good performance, the search has to be driven into the areas of previous local optima, instead of avoiding them. The rationale is that good local optima are often close to other local optima; hence, it might make sense to reinforce the search around previous optima, instead of avoiding regions close to them. Of course, the limit of this reasoning occurs when search cannot escape some particular local optimum.

In the algorithm that we devised for this, which could be named *non-tabu search*, the region around a local optimum is exploited by drawing a random solution on its vicinity, and restarting local search from it. A parameter σ controls the distance from the current base solution, used to draw new starting solutions. Another parameter, R, controls the number of attempts to do around each base solution. After R attempts are made, the base solution moves into the best solution found in theses tentatives.

These two parameters give a way for controlling the search, and to adapt it to the problem being tackled. Good parameters for a particular problem are generally easy to devise; but we could find no parameters that are simultaneously good for all the benchmarks.

3 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 several escaping mechanisms. The simplest of them is random-start local search. Another one was 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 other possibility presented is based on the exploitation of the area of a the previous local optimum, by drawing starting points for local search in its vicinity: we called it non-tabu search.

Computational experiments (available in [1]) have shown that all the escaping mechanisms were effective for avoiding stagnation in local optima.

Due to the simplicity of its implementation, random start iterated local search is a highly attractive method. However, for some problems it is not able to find truly good solutions (though the average solution is generally of high quality).

The simplex expansion escaping mechanism is for most of the test cases slightly superior to random start local search, but in general the non-tabu search provides the best results.

Test and improvement of the escape methods for problems with equality constraints, and other possibilities of dealing with these constraints, remain as topics for future research. More research topics are their incorporation in more elaborate strategies, like strategic oscillation or population based methods.

References

1. Pedroso, J.P.: Simple meta-heuristics using the simplex algorithm for non-linear programming. Technical Report DCC-2007-06, DCC, FC, Universidade do Porto (2007)
2. Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal **7** (1965) 308–313
3. Lewis, R.M., Torczon, V., Trosset, M.W.: Direct search methods: then and now. Journal of Computational and Applied Mathematics **124**(1-2) (2000) 191–207
4. Pedroso, J.P.: Meta-heuristics using the simplex algorithm for nonlinear programming. In: Proceedings of the 2001 International Symposium on Nonlinear Theory and its Applications, Miyagi, Japan (2001) 315–318
5. Audet, C., Dennis Jr, J.: A pattern search filter method for nonlinear programming without derivatives. SIAM Journal on Optimization **14**(4) (2004) 980–1010
6. Chelouah, R., Siarry, P.: A hybrid method combining continuous tabu search and nelder-mead simplex algorithms for the global optimization of multiminima functions. European Journal of Operational Research **161** (2005) 636–654