A Framework for Derivative Free Algorithm Hybridization

Jose Luis Espinosa-Aranda, Ricardo Garcia-Rodenas, and Eusebio Angulo

Universidad de Castilla-La-Mancha, Paseo de la Universidad 4, Ciudad Real, Spain
{JoseL.Espinosa,Ricardo.Garcia,Eusebio.Angulo}@uclm.es

Abstract. Column generation is a basic tool for the solution of large-scale mathematical programming problems. We present a class of column generation algorithms in which the columns are generated by derivative free algorithms, like population-based algorithms. This class can be viewed as a framework to define hybridization of free derivative algorithms. This framework has been illustrated in this article using the Simulated Annealing (SA) and Particle Swarm Optimization (PSO) algorithms, combining them with the Nelder-Mead (NM) method. Finally a set of computational experiments has been carried out to illustrate the potential of this framework.

Keywords: Column generation, derivative free methods, Particle Swarm Optimization, Nelder-Mead Simplex, Simulated Annealing.

1 Introduction

Derivative free algorithms is a key area with an increasing interest due to its versatility and broad use in optimization problems. The main motivation for this paper is to propose a means to accelerate the convergence of these algorithms in order to be able to apply them to large-scale problems in a reasonable computational time.

Column generation (CG) [1] is a classic means to attack the following (large-scale) constrained optimization problem:

$$\underset{x \in X}{\text{minimize}}\ f(x), \qquad \mathrm{P}(f, X)$$

where the feasible region $X \subseteq \mathbb{R}^N$ is non-empty and closed, and $f : X \mapsto \mathbb{R}$ is a continuous function on X. These algorithms follow two main steps:

1. Column Generation Problem (CGP). A relaxation of the original problem is constructed, based on current estimates $\widetilde{x}$ of the optimal solution, which provides a bound to the optimal value of the original problem, a new column and information that indicates if the column should be introduced into the solution or if the process should stop. This problem can be defined by $\mathrm{P}(\hat{f}, X)$ where $\hat{f}(x)$ is an approximation of $f(x)$ on $\tilde{x}$.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 80–89, 2013.

2. Restricted Master Problem (RMP). Previously generated columns define an inner approximation $\widehat{X}$ of the feasible region X and the new column $\widetilde{y}$ is used to expand it. The original problem is approximated by $\mathrm{P}(f, \widehat{X})$ and its solution $\widetilde{x}$ is used to define a new CG sub-problem which iterates the process.

Classic examples of this type of algorithm used for linearly and nonlinearly constrained problems are *simplicial decomposition* (SD) [2] and *Restricted Simplicial Decomposition* [3]. In these algorithms the relaxation is a linearization of the objective function while the feasible set is defined by the convex hull of the retained columns in the RMP. In this algorithm column generation is not based on pricing or dual information and is associated with multidimensional extensions of primal descent algorithms in NLP.

In [1] a CG algorithm is proposed based on closed descent algorithms. In this framework the CGP is defined by a given algorithm and the RMP as a phase which tries to accelerate the algorithm used in CGP. RMP is solved by an algorithm capable of using the advantages of RMP to its favour, like a reduced number of variables and a simple set of restrictions.

In this algorithm the SD is obtained as a column running only one iteration of the *Frank-Wolfe* method [4]. A key theoretical result is that the local rate of convergence of SD is governed by the local convergence rate of the method chosen for the solution of the RMP; thus a superlinear or quadratic convergence rate may be attained if a (projected) Newton method is used [5]. Note that the rate of convergence of the *Frank-Wolfe* algorithm is sublinear.

In [6] was carried out an experimental study focused on their computational efficiency of CG methods. Two types of test problems were considered, the first one is the nonlinear, capacitated single-commodity network flow problem, and the second one is a combined traffic assignment model. This paper validates this methodology in order to improve the performance of feasible direction and simplicial decomposition methods used in equilibrium assignment models.

The main contribution of this paper is to extend this framework to derivative free optimization methods for general optimization problems. To verify the goodness of the framework a set of computational tests have been performed with the algorithms: i) Nelder-Mead simplex method (NM) [7] ii) Bohachevsky et al. Simulated Annealing (SA) [8], and iii) Particle Swarm Optimization (PSO) [9], showing that the convergence rate and effectiveness of the heuristics algorithms can be improved by hybridization.

The paper is organized as follows. In Section 2 the proposed framework for derivative free optimization methods is explained, in Section 3 we discuss some instances of the hybrid CG algorithm, in Section 4 several computational experiments are reported, and finally in Section 5 we conclude with a discussion of our findings and future work.

2 The Conceptual Framework of Hybridization

[1] dealt with a class of CG algorithms in which the approximated objective function $\hat{f}(y)$ coincides with the original $f(y)$. This point of view leads to a CG algorithm which can be defined according to the following three key items: i) the algorithm to solve the $\mathrm{P}(f, X)$ (denoted by $\mathcal{A}_{\mathrm{c}}$), ii) the algorithm applied to RMP (denoted by $\mathcal{A}_{\mathrm{r}}$) and iii) the means of stating the inner approximation $\widehat{X}$. The algorithms $\mathcal{A}_{\mathrm{c}}$ and $\mathcal{A}_{\mathrm{r}}$ analysed in [1] are descent primal methods and the $\mathrm{P}(f, X)$ is a differentiable optimization problem. In this paper we extend this framework to derivative free optimization methods and for general optimization problems $\mathrm{P}(f, X)$. We begin by stating the characteristics of the optimization algorithms $\mathcal{A}_{\mathrm{c}}$ and $\mathcal{A}_{\mathrm{r}}$ that can be used to solve the $\mathrm{CGP}(f, X)$ or $\mathrm{RMP}(f, \widehat{X})$.

A type of derivative free algorithm which works with populations generates an evolution of the population instead of generating a sequence of solutions (*particles* in Particle Swarm Optimization, an *atom* in Simulated Annealing, *chromosomes* in Genetic Algorithms, *ants* in Ant Colony Optimization, etc. [10]). For this reason we consider algorithms based on populations, and if the cardinality of this population is one the classical optimization methods appear.

Assumption 1 (Optimization Algorithm) *Let $P(f, Z)$ be an optimization problem and let $\mathcal{A}$ be an optimization algorithm. This algorithm is defined as an iterative procedure which will be assumed to fulfill the following two conditions.*

i) (Feasible population). This algorithm works on a population of particles $\widetilde{\mathcal{Z}} = \{z_1, \cdots, z_m\}$ which is modified iteratively. This algorithm is described by means of a point-to-set algorithmic mapping

$$\begin{aligned} \mathcal{A} : Z^m &\mapsto 2^{Z^m} \\ \widetilde{\mathcal{Z}} &\mapsto \mathcal{A}(\widetilde{\mathcal{Z}}) \end{aligned} \tag{1}$$

where 2^{Z^m} is the power set of Z^m. Also we denote

$$\mathcal{A}^t(\widetilde{\mathcal{Z}}) := [\mathcal{A} \overbrace{\circ \cdots \circ}^{t \text{ times}} \mathcal{A}]\,(\widetilde{\mathcal{Z}}) \tag{2}$$

and the realization of $t'-$iterations generate a sequence of populations $\widetilde{\mathcal{Z}}^1, \cdots, \widetilde{\mathcal{Z}}^{t'}$ such as $\widetilde{\mathcal{Z}}^t := \{z_1^t, \cdots, z_m^t\} \in \mathcal{A}(\mathcal{Z}^{t-1})$ of feasible points for any initialization $\widetilde{\mathcal{Z}} \in Z^m$.

ii) (Convergent property). This algorithm is convergent for every $\widetilde{\mathcal{Z}} \in Z^m$ in the following sense. Let $\mathcal{Z}^t \in \mathcal{A}^t(\mathcal{Z})$, $t = 1, 2, \cdots$ be the sequence of populations generated by $\mathcal{A}$ and let

$$z^t := \operatorname{Arg} \underset{z \in \mathcal{Z}^t}{\text{minimize}}\, f(z) \tag{3}$$

then

$$d_{SOL(f,Z)}(z^t) := \left\{ \min_{z^* \in SOL(f,Z)} \|z^t - z^*\| \right\} \to 0 \tag{4}$$

where $SOL(f, Z)$ denotes the set of global minimizers of $P(f, Z)$.

The methods used in [1] are descent closed algorithms which are convergent for differentiable convex programs. In this paper we generalize them to convergent algorithms for a given optimization problem. Properties i) and ii) guarantee the convergence of the algorithm in a finite number of iterations or a finite number of descents in the objective function. In the CG method the number of these descents is used to decide a change between the algorithms used. The following definition explains this concept.

Definition 1 (*$n-$descent iteration).*
We say that an algorithm $\mathcal{A}$ achieves an $n-$descent if the algorithm is applied in a sufficient number of iterations to ensure n times a descent in the objective function. More formally, let n be a positive integer number and we denote

$$f(\widetilde{\mathcal{Z}}) := \underset{z\in\widetilde{\mathcal{Z}}}{\text{minimize}}\, f(z) \tag{5}$$

An $n-$ descent iteration of $\mathcal{A}$ consists of applying the following algorithm:

$$\begin{cases} \text{Let } \widetilde{\mathcal{Z}}' \text{ be the initial population and let } \ell = 0. \\ \text{Do While } (\ell \leq n) \\ \quad \widetilde{\mathcal{Z}} \in \mathcal{A}(\widetilde{\mathcal{Z}}') \\ \quad \text{If } f(\widetilde{\mathcal{Z}}') < f^* \text{then } f^* = f(\widetilde{\mathcal{Z}}') \text{ and } \ell = \ell + 1 \\ \quad \widetilde{\mathcal{Z}}' = \widetilde{\mathcal{Z}} \\ \text{End Do While} \end{cases}$$

The realization of a $n-$descent iteration is denoted by:

$$\widetilde{\mathcal{Z}} \in \mathcal{A}(\widetilde{\mathcal{Z}}', n) \tag{6}$$

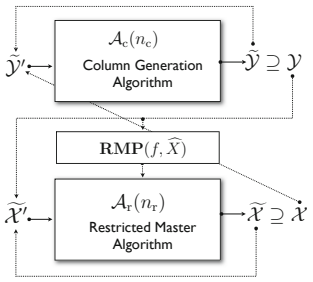

Fig. 1. Hybridization of Algorithms

In Table 1, we summarize the different steps of a CG algorithm belonging to the proposed framework. The resulting algorithm can be viewed as a hybrid algorithm of $\mathcal{A}_c$ and $\mathcal{A}_r$.

Table 1. Hybrid CG algorithm

0. (*Initialization*): Let $\{n_{\mathrm{c}}^t\}$ and $\{n_{\mathrm{r}}^t\}$ be two sequences of positive integer numbers. Let $\widetilde{\mathcal{Y}}'^1$ and $\widetilde{\mathcal{X}}'^1$ be respectively the initial populations for the algorithms $\mathcal{A}_{\mathrm{c}}$ and $\mathcal{A}_{\mathrm{r}}$. Let $\mathcal{X}^1 := \{\emptyset\}$ and $t := 1$.
1. (*Column Generation Algorithm*): Apply a n_{c}^t -descent with the algorithm $\mathcal{A}_{\mathrm{c}}$ on the problem $\mathrm{P}(f, X)$, starting from $\widetilde{\mathcal{Y}}'^t$. Let $\widetilde{\mathcal{Y}}^t$ be the resulting population i.e., $\widetilde{\mathcal{Y}}^t \in \mathcal{A}(\widetilde{\mathcal{Y}}'^t, n_{\mathrm{c}}^t)$ and let y^t be the current best solution. i.e.,
$$y^t := \underset{y \in \widetilde{\mathcal{Y}}^t}{\text{Arg minimize}}\ f(y), \tag{7}$$
2. (*Set augmentation*): Choose a set of columns $\mathcal{Y}^t$ satisfying $y^t \in \mathcal{Y}^t \subset \widetilde{\mathcal{Y}}^t$. Let $X^{t+1} \subset X$ a nonempty closed set containing $\{\mathcal{Y}^t, \mathcal{X}^t\}$.
3. (*Update of populations for RMP*$(f, \widehat{X})$): Let $\widetilde{\mathcal{X}'}^{t+1} \subseteq \mathcal{Y}^t \cup \widetilde{\mathcal{X}}^t$.
4. (*Restricted Master Algorithm*): Apply a n_{r}^t-descent with the algorithm $\mathcal{A}_{\mathrm{r}}$ on $\mathrm{RMP}(f, X^{t+1})$, starting from $\widetilde{\mathcal{X}'}^{t+1}$. Let $\widetilde{\mathcal{X}}^{t+1} \in \mathcal{A}(\widetilde{\mathcal{X}'}^{t+1}, n_{\mathrm{r}}^t)$ be the resulting population. Let x^{t+1} be the current best solution. i.e.,
$$x^{t+1} := \underset{x \in \widetilde{\mathcal{X}}^{t+1}}{\text{Arg minimize}}\ f(x), \tag{8}$$
5. (*Update of population for* $P(f, X)$): Choose a set of solutions $\mathcal{X}^{t+1}$ satisfying $x^{t+1} \in \mathcal{X}^{t+1} \subseteq \widetilde{\mathcal{X}}^{t+1}$ and update the population for solving $\mathrm{P}(f, X)$ as $\widetilde{\mathcal{Y}'}^{t+1} \subseteq \widetilde{\mathcal{Y}}^t \cup \mathcal{X}^{t+1}$.
6. (*Termination criterion*): If $x^{t+1} \in \mathrm{SOL}(f, X) \rightarrow$ Stop. Otherwise, let $t := t + 1$. Go to Step 1.

3 Instances of Hybrid CG Algorithms

Classical algorithms used in differentiable optimization such as SD, RSD or NSD [1] are roughly obtained letting Z be a point instead of a population, X a polyhedral set and $\widehat{X}$ as the convex hull of retained columns. These methods can be interpreted as a hybridization of $\mathcal{A}_{\mathrm{c}}$=Frank-Wolfe and $\mathcal{A}_{\mathrm{r}}$=(projected) Newton method.

In free derivatives optimization the line-search based modification of the Hooke and Jeeves method [11] combine, in one iteration, a coordinate-wide search through each of the variables ($\mathcal{A}_{\mathrm{c}}$) with a pattern search ($\mathcal{A}_{\mathrm{r}}$). The $\mathcal{A}_{\mathrm{r}}$ produces an acceleration in the convergence of $\mathcal{A}_{\mathrm{c}}$.

In this paper we investigate numerically only the basic framework which consists of letting $X = \mathbb{R}^n$, $\widehat{X}^t = \mathbb{R}^N$, $\mathcal{Y}^t = \{y^t\}$, $\widetilde{\mathcal{X}}^{t+1} = \{y^t\}$ and $\widetilde{\mathcal{Y}'}^t = \widetilde{\mathcal{Y}}^{t-1}$. Roughly, this basic hybrid algorithm consists of interchanging both algorithms when an $n-$descent iteration is carried out. We have introduced the following improvement. In order to take into account the random walking of $\mathcal{A}_{\mathrm{c}}$, the

objective function $f(x)$ in RMP is represented by $f(d^t \cdot x^T)$ where T is the transpose of a vector, and $d^t = y^t - x^{t-1}$. This new function is a scalarization of the function $f(x)$.

In the numerical experiments, for $\mathcal{A}_{\mathrm{r}}$ the Nelder-Mead simplex search method has been used. This method is a direct search method that does not use numerical or analytic gradients and has local convergence. This algorithm described in [12] uses a simplex of $N+1$ points for N-dimensional vectors x. The algorithm first makes a simplex around the initial point y^t. Then, the algorithm modifies the simplex repeatedly generating at each step of the search a new point in or near the current simplex. The function value at the new point is compared with the function's values at the vertices of the simplex and, usually, one of the vertices is replaced by the new point, creating a new simplex.

On the other hand, for $\mathcal{A}_{\mathrm{c}}$ two algorithms with properties of global convergence have to be employed. The motivation is to combine global convergent methods with a local convergent method which is more computationally efficient. In particular in this paper SA and PSO algorithms have been used for $\mathcal{A}_{\mathrm{c}}$.

The SA algorithm used in this paper was proposed by [8] is an improved version of the classical SA [13]. It adds the parameters α and β, α is the maximum step size and β dictates the conditional probability that a worse solution can be accepted.

The second algorithm used is the original PSO algorithm using the global *gbest* model. It is a population-based algorithm and evolutionary in nature, introduced by [9]. PSO is a kind of random search algorithm based on the metaphors of social interaction and communications. This method has been shown to be effective in solving difficult and complex optimization problems in a wide range of fields [14] [15]. PSO maintains at each iteration a set of swarm particles, feasible points represented by $(\widetilde{\mathcal{Y}}')$ in our framework.

4 Computational Results

In this section the framework proposed in this paper is tested by the hybridization of SA with NM (SA+NM) and PSO with NM (PSO+NM) to test the improvement of the algorithms used. As mentioned above, the main motivation of this hybridization is to combine the capacity of the SA and PSO algorithms to find a global minimum with the capacity of the NM algorithm to obtain a more precise solution near a local minimum.

In the next sections, the design of experiments is explained, and the results are reported comparing our approach with the SA, NM and PSO algorithms.

4.1 Design of the Experiments

To compare the quality of the proposed modified algorithms it is necessary to test them with a large-scale test process on a variety of response functions, ignoring the possible effectiveness of the algorithm in a specific type of function.

The set of 20 deterministic functions used in the computational experiments are found in [16] and [17], and have been selected because they are a set of curvilinear functions for difficult unconstrained optimization problems with a variety of dimensions (N) and functional forms, which makes it possible to asses the robustness of the proposed approach.

To evaluate the algorithms' effectiveness each of them have been executed 100 times per test function. In each of the executions, the initial point in SA, NM and SA+NM is sampled from Uniform(-50,50) and initial particles in PSO and PSO+NM are randomly generated from Uniform(-50,50). Also, in PSO and PSO+NM algorithms the number of particles is defined as $5N$.

The stopping criterion selected for all the algorithms is that the current execution reaches $5000N^2$ function evaluations, which corresponds to $1000N$ iterations of the PSO algorithm. The sequence of number of descents was $\{n_{\mathrm{c}}^t\} = 5$ and $\{n_{\mathrm{r}}^t\} = 100$ in the numerical tests.

The criteria selected to compare the algorithm's robustness, effectiveness, efficiency and accuracy using the results obtained in the experiments described above are:

1. Rate of successful minimizations, considering that a successful minimization occurs when $|f(\mathcal{X}^t) - f^*| < 10^{-10}$ where $f^* = \min\{f(x) : x \in X\}$.
2. Average of function evaluations until a successful minimization occurs.
3. Gap between the best minimum found in the 100 executions and the optimum of the function.

4.2 Results

The results given in this section will show that the approach described is capable of improving the SA, NM and PSO algorithms.

Tables 2 and 3 show the complete results of the experiments performed. By comparing them, it can be seen that the modified version of the algorithms in general has a significantly higher rate of successful minimization and a lower average of objective function evaluation before reaching a successful minimization than the original algorithms.

To prove these statements the non-parametric statistical Wilcoxon test is carried out with a significance of 0.05 to compare the paired groups SA vs SA+NM, NM vs SA+NM, PSO vs PSO+NM and NM vs PSO+NM, and gives the results shown in Table 4. Taking into account these results, it can be stated that there exists a clear improvement in the number of successful minimizations for all the algorithms.

In the case of function evaluations until a successful minimization occurs, the results show that in NM and SA vs SA+NM there are no conclusive results because of the lack of successful minimizations in the NM and SA algorithms, but in PSO and NM vs PSO+NM it can be seen that there does exist an improvement, reducing the number of evaluations necessary to find the minimum of the function.

Table 2. NM, SA and SA+NM Results

	Successful minimizations			Function Evaluations			Gap with best minimum found		
Function	NM	SA	SA+NM	NM	SA	SA+NM	NM	SA	SA+NM
1	17	0	17	1.829e+02	-	1.479e+04	1.769e-13	2.190e-08	2.827e-15
2	0	0	24	-	-	1.816e+04	1.411e-09	1.035e-05	3.919e-12
3	5	0	46	2.912e+02	-	1.450e+04	4.300e-11	1.191e-06	1.943e-13
4	0	0	97	-	-	1.658e+04	3.756e-10	7.366e-07	9.437e-16
5	0	0	2	-	-	4.504e+04	2.181e-10	1.885e-04	4.288e-11
6	0	0	87	-	-	4.480e+04	3.941e-10	6.979e-06	2.553e-13
7	69	5	64	4.014e+02	2.757e+04	2.210e+04	0	2.208e-12	1.449e-16
8	0	0	0	-	-	-	3.169e-10	3.652e-04	1.167e-07
9	0	0	29	-	-	7.993e+04	4.480e-10	8.617e-05	2.130e-12
10	0	0	0	-	-	-	1.261e-10	1.809e-03	5.536e-09
11	0	0	23	-	-	8.001e+04	9.319e-10	6.986e-05	3.390e-13
12	0	0	0	-	-	-	6.552e-08	3.071e-06	1.173e-06
13	0	0	0	-	-	-	5.403e-07	3.247e-06	1.175e-06
14	0	0	29	-	-	3.200e+05	7.027e-10	1.916e-03	6.095e-12
15	78	0	0	3.499e+03	-	-	1.704e-14	6.399e-03	4.059e-06
16	0	0	0	-	-	-	1.009e-01	6.264e-01	5.793e-01
17	0	0	0	-	-	-	235.796	5.081	1.990e+00
18	0	0	0	-	-	-	2.949e-01	1.589	1.947e+00
19	0	0	0	-	-	-	2.544	1.406	1.686e+00
20	0	0	0	-	-	-	7.227e-01	4.031	4.348e+00

Table 3. NM, PSO and PSO+NM Results

	Successful minimizations			Function Evaluations			Gap with best minimum found		
Function	NM	PSO	PSO+NM	NM	PSO	PSO+NM	NM	PSO	PSO+NM
1	17	48	64	1.829e+02	5.408e+03	6.862e+02	1.769e-13	0	0
2	0	64	76	-	2.962e+03	2.159e+03	1.411e-09	0	0
3	5	58	69	2.912e+02	2.778e+03	9.795e+02	4.300e-11	0	0
4	0	57	71	-	3.023e+03	1.677e+03	3.756e-10	0	0
5	0	61	80	-	1.112e+04	5.231e+03	2.181e-10	1.523e-109	3.641e-99
6	0	62	80	-	3.925e+03	1.956e+03	3.941e-10	0	6.043e-283
7	69	57	96	4.014e+02	7.442e+03	1.307e+03	0	0	1.733e-238
8	0	0	63	-	-	8.959e+03	3.169e-10	1.675e-09	3.748e-23
9	0	24	34	-	9.891e+03	4.012e+03	4.480e-10	1.868e-32	1.805e-31
10	0	25	64	-	5.543e+04	2.197e+04	1.261e-10	7.500e-22	0
11	0	57	73	-	9.798e+03	6.319e+03	9.319e-10	0	0
12	0	43	43	-	1.284e+05	8.431e+04	6.552e-08	6.626e-12	6.626e-12
13	0	1	23	-	3.151e+05	2.354e+05	5.403e-07	9.539e-11	7.986e-11
14	0	30	37	-	4.586e+04	2.528e+04	7.027e-10	1.032e-30	5.311e-31
15	78	0	43	3.499e+03	-	8.342e+04	1.704e-14	1.728e-09	5.276e-18
16	0	0	0	-	-	-	1.009e-01	7.411e-03	1.478e-02
17	0	13	10	-	2.495e+05	2.193e+05	2.358e+02	0	0
18	0	0	3	-	-	3.383e+05	2.949e-01	1.865e-08	5.417e-18
19	0	44	53	-	5.563e+05	4.949e+05	2.544e+00	0	0
20	0	36	22	-	1.371e+06	8.114e+05	7.227e-01	0	0

Table 4. Significances of Wilcoxon tests

Algorithms	S. minimizations	F. Evaluations	Gap best minimum
NM vs SA+NM	0.037	0.0545*	0.3005
SA vs SA+NM	0.0025	-*	0.0285
NM vs PSO+NM	0.00015	0.034	0.0001
PSO vs PSO+NM	0.001	0.0001	0.2345

* Not enough data

To illustrate the general performance of the approaches in terms of evaluations of the objective function, Figure 2 show the 2nd function of the experiments, (B2 function), plotting the best evaluation of the function vs the number of evaluations of the objective function. It can be seen from the figures that the SA+NM and PSO+NM algorithms converge more quickly than the classical algorithms, also breaking the local minimum which cannot be reduced by the NM algorithm.

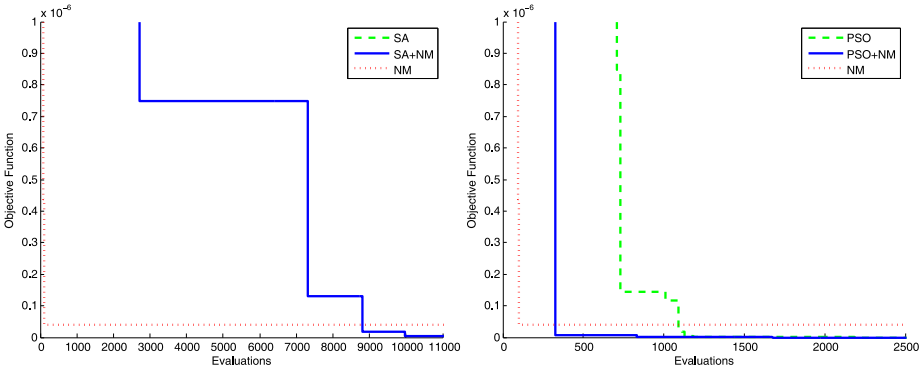

Fig. 2. Objective Function *vs* Evaluations for SA, NM and SA+NM

Finally, for the gap with the best found minimum results it can be shown that the SA is improved by the SA+NM algorithm and the NM is improved by the PSO+NM algorithm.

5 Conclusions

In this paper a new framework for hybridization of derivative free algorithms is presented. This approach includes the hybridization of two methods in an attempt to combine the specific capacity of exploitation and exploration of each algorithm, increasing the robustness, effectiveness, efficiency and accuracy.

This paper investigates numerically the basic CG algorithm consisting of letting $\widehat{X} = X = \mathbb{R}^N$ and the algorithms NM, SA and PSO. Also a set of computational tests has been done using 20 curvilinear functions for difficult unconstrained optimization, in which the original algorithms are compared with the modified versions. The results show that the hybrid algorithms have a higher rate of successful minimizations and an acceleration of the algorithms, which are capable of giving the optimum using fewer evaluations of the function than the original algorithm.

Future work will research new instances of the algorithm based on the initialization [16] or augmentation rules for defining $\widehat{X}$. The idea is to extend the computational results by trying to apply the framework to other algorithms and more complex optimization problems like [18].

Acknowledgements. This research has been financed by the *Ministerio de Ciencia e Innovación* of Spain with the TRA-2011-27791-C03-03 research project.

References

1. García, R., Marín, A., Patriksson, M.: Column generation algorithms for nonlinear optimization, I: Convergence analysis. Optimization 52(2), 171–200 (2003)
2. Von Hohenbalken, B.: Simplicial decomposition in nonlinear programming algorithms. Mathematical Programming 13(1), 49–68 (1977)
3. Ventura, J.A., Hearn, D.W.: Restricted simplicial decomposition for convex constrained problems. Mathematical Programming 59, 71–85 (1993)
4. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quarterly, 95–110 (1956)
5. Hearn, D.W., Lawphongpanich, S., Ventura, J.A.: Restricted simplicial decomposition: Computation and extensions. Mathematical Programming Study 31, 99–118 (1987)
6. García-Ródenas, R., Marín, A., Patriksson, M.: Column generation algorithms for nonlinear optimization, II: Numerical investigations. Computers and Operations Research 38(3), 591–604 (2011)
7. Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal 7, 308–313 (1965)
8. Bohachevsky, I.O., Johnson, M.E., Myron, L.S.: Generalized Simulated Annealing for Function Optimization. Technometrics 28(3) (1986)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings IEEE of the International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
10. Kennedy, J.F., Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)
11. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal of the Association for Computing Machinery, 212–229 (1961)
12. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the nelder-mead simplex method in low dimensions. SIAM Journal on Optimization 9(1), 112–147 (1999)
13. Laarhoven, P., Aarts, E.: Simulated annealing: Theory and Applications, 3rd edn. Kluwer Academic Publishers, Dordrecht (1987)
14. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. Journal of Artificial Evolution and Applications, 1–10 (2008)
15. Angulo, E., Castillo, E., García-Ródenas, R., Sánchez-Vizcaíno, J.: Determining Highway Corridors. Journal of Transportation Engineering 138(5), 557–570 (2012)
16. Fan, S.-K.S., Zahara, E.: A hybrid simplex search and particle swarm optimization for unconstrained optimization. European Journal of Operational Research, 527–548 (2006)
17. Functions definition, `http://bit.ly/UQlMTB` (last access: January 15th, 2013)
18. Espinosa-Aranda, J.L., García-Ródenas, R.: A discrete event-based simulation model for real-time traffic management in railways. Journal of Intelligent Transportation Systems 16(2), 94–107 (2012)