Evolutionary Cutting Planes

Jérémie Decock, David L. Saint-Pierre, Olivier Teytaud

Evolutionary Cutting Planes

Jérémie Decock, David L. Saint-Pierre, Olivier Teytaud

TAO (Inria), LRI, UMR 8623 (CNRS - Univ. Paris-Sud)
Bat 660 Claude Shannon Univ. Paris-Sud, 91190 Gif-sur-Yvette, France
Email: {*firstname.lastname*}*@inria.fr*

Abstract. The Cutting Plane method is a simple and efficient method for optimizing convex functions in which subgradients are available. This paper proposes several methods for parallelizing it, in particular using a typically evolutionary method, and compares them experimentally in a well-conditioned and ill-conditioned settings.

1 Introduction

Various information levels can help for optimization: the objective function values are usually available; sometimes, the gradient is also provided[3], and in some cases the Hessian is available (either directly, in Newton's method, or approximated by successive gradients, as in quasi-Newton methods[2, 4, 6, 14]; there are also cases in which both the gradient and Hessian are approximated by objective function values without direct computations[13]). We here consider cases in which we have access to *subgradients*.

A traditional method for cases in which subgradients are available is the cutting-plane method[8]: at each search point $\boldsymbol{x}_n$, a linear approximation of the objective function f, tangent to f at $\boldsymbol{x}_n$, is computed[1]; this linear approximation is termed a cut c. This cutting-plane method assumes that the objective function f is convex, so that the cuts are all *below* f. The $\tilde{f}_n$ function – corresponding to the maximum of cuts $\{c_1, ..., c_n\}$ computed in previous iterations $\{1, ..., n\}$ – can therefore be used as an heuristic. Thus, $\tilde{f}$ is an approximate model of the objective function and it determines the search point of the next iteration $n+1$: $\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x}} \tilde{f}_n$. As $\tilde{f}$ is piecewise linear, $\arg\min_{\boldsymbol{x}} \tilde{f}_n$ can be quickly computed using linear programming solvers.

The parallelization of optimization algorithms is critical for many applications. To the best of our knowledge, whereas many methods using linear cuts have been parallelized (e.g. dual dynamic programming[12]), there is no published work on parallel cutting plane methods.

In this paper we propose variants of parallel cutting plane methods. We analyse these variant to validate the approach both theoretically and experimentally. Section 2 formalizes the problem. Section 3 introduces several parallel cutting plane methods. Section 4 provides experimental results.

[1] This cutting-plane method is not the one which is used in integer linear programming but the version for continuous non-linear optimization with gradients.

2 Problem Statement

Let $f : \mathcal{C} \subseteq \mathbb{R}^d \to \mathbb{R}$ be a convex function with domain $\mathcal{C}$, where d is the dimension of the problem. The generic problem at hand is defined by

$$\min f(\boldsymbol{x}) \text{ , subject to } \boldsymbol{x} \in \mathcal{C},$$

where $\mathcal{C}$ is a convex set. Assuming that we have access, for a given $\boldsymbol{x}$, to both $f(\boldsymbol{x})$ and a subgradient $\nabla f_{\boldsymbol{x}}$, then the cutting plane method can be defined as follow for the iteration $n \in \{0, 1, 2, \dots\}$:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x}} \tilde{f}_n \ , \tag{1}$$

$$c_{n+1} = \boldsymbol{x} \mapsto c_{n+1}(\boldsymbol{x}) = f(\boldsymbol{x}_{n+1}) + \nabla f_{\boldsymbol{x}_{n+1}}.(\boldsymbol{x} - \boldsymbol{x}_{n+1}) \ , \tag{2}$$

$$\tilde{f}_{n+1} = \max(\tilde{f}_n, c_{n+1}) \ , \tag{3}$$

where $\tilde{f}_0$ is some heuristic, and where Eq. (1) defines the sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$ of search points. Eq. (2) defines the cuts, i.e. linear functions tangent to the objective function f. Eq. (3) defines the sequence of approximate models $\tilde{f}_1, \tilde{f}_2, \dots, \tilde{f}_n$. $\tilde{f}_0$ can be $-\infty$ or some heuristically defined functions. However, if $\tilde{f}_0$ can be encoded in linear programming, then Eq. (1) can be solved by linear programming for any $n \geq 0$. Usually x_1 is randomly drawn in $\mathcal{C}$ if $\tilde{f}_0$ is a constant function.

3 Parallel cutting plane methods

The classic cutting plane method is a sequential approach that generates only one cut per iteration n. The idea put forth in this paper is to generate several per iteration. Assuming we have access p processors, then we can potentially generate p cuts per iteration. In this section, we define parallel cutting plane (PCP) methods. First Section 3.1 describes a naive parallelization of the cutting plane method. Then Section 3.2 explains a gaussian approach, using random perturbation; a variant is also proposed. Third, Section 3.3 explains the billard method to generate the cuts and also develops a variant.

3.1 Regular PCP

The idea of a naive parallelization of the cutting plane method is to evaluate several points between $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n+1}$ rather than only $\boldsymbol{x}_{n+1}$. For some constant $C > 0$, we define a regular PCP as follows for the iteration $n \in \{0, 1, 2, \dots\}$:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x}} \tilde{f}_n(\boldsymbol{x}) \ , \tag{4}$$

$$\forall i \in \{1, \dots, p\}, \boldsymbol{x}_{n+1,i} = \boldsymbol{x}_n + \left(C\frac{i}{p}\right)(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n) \ , \tag{5}$$

$$c_{n+1,i} = \boldsymbol{x} \mapsto c_{n+1,i}(\boldsymbol{x}) = f(\boldsymbol{x}_{n+1,i}) + \nabla f_{\boldsymbol{x}_{n+1,i}}.(\boldsymbol{x} - \boldsymbol{x}_{n+1,i}) \ , \tag{6}$$

$$\tilde{f}_{n+1} = \max(\tilde{f}_n, \max_i c_{n+1,i}) \ . \tag{7}$$

where $\tilde{f}_0$ is some heuristic. Obviously Eq.(6) can be computed in parallel. In the worst case scenario the only cut of relevance is from the point $\boldsymbol{x}_{n+1}$ in which case it boils down to the sequential cutting plane method. When $C = 1$, it means that we evaluate cuts regularly distributed from $\boldsymbol{x}_n$ to $\boldsymbol{x}_{n+1}$; this is termed the regular PCP. When $C > 1$ (e.g. $C = 2$), we evaluate cuts also farther than $\boldsymbol{x}_{n+1}$; this is termed extended regular (or *e-regular* in the following figures).

3.2 Gaussian PCP

Evolutionary algorithms are usually not the fastest optimization methods, but they have advantages in terms of robustness and parallelization. Combining cutting planes and evolutionary mutations is a natural idea for parallelizing the cutting planes method. The naive PCP from Section 3.1 has the drawback that it generates points distributed along a line. We propose a gaussian variant of the form $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, where cuts are generated using $\boldsymbol{\mu} = \boldsymbol{x}_{n+1}$ and $\boldsymbol{\sigma} = ||\boldsymbol{x}_{n+1} - \boldsymbol{x}_n||/\sqrt{d}$. Note the division by the square root of the dimension in order to keep the distance normalized. As such we propose the following variant:

$$\tilde{f}_0 = \textit{some heuristic} \tag{8}$$

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x}} \tilde{f}_n(\boldsymbol{x}) \ , \tag{9}$$

$$\boldsymbol{x}_{n+1,1} = \boldsymbol{x}_{n+1} \tag{10}$$

$$\boldsymbol{\mu} = \boldsymbol{x}_{n+1} \quad \boldsymbol{\sigma} = ||\boldsymbol{x}_{n+1} - \boldsymbol{x}_n||/\sqrt{d} \tag{11}$$

$$\forall i \in \{2, \ldots, p\}, \boldsymbol{x}_{n+1,i} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \ , \tag{12}$$

$$\forall i \in \{2, \ldots, p\}, c_{n+1,i}(\boldsymbol{x}) = f(\boldsymbol{x}_{n+1,i}) + \nabla f_{\boldsymbol{x}_{n+1,i}}.(\boldsymbol{x} - \boldsymbol{x}_{n+1,i}) \ , \tag{13}$$

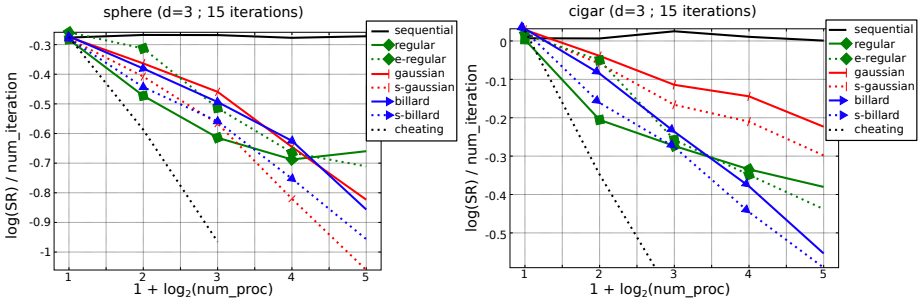
$$\tilde{f}_{n+1} = \max(f_n, \max_i c_{n+1,i}) \tag{14}$$

Our aim is to have points less regularly distributed, hopefully with a better diversity in the cuts. Admittedly, a drawback of this method is that points are generated isotropically, which might be problematic for ill-conditioned problems.When the domain of f is constrained ($\mathcal{C} \subset \mathbb{R}^d$) Eq. 12 might draw some points outside $\mathcal{C}$; another instruction is then required to filter these wrong points.

Special-Gaussian PCP We here propose a "special" variant of the Gaussian version above. The pseudo-code is the same, but $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are such that the Gaussian distribution is centered on the average $\boldsymbol{\mu} = \frac{1}{2}(\boldsymbol{x}_n + \boldsymbol{x}_{n+1})$ and $\boldsymbol{\sigma} = ||\boldsymbol{x}_{n+1} - \boldsymbol{x}_n||/2\sqrt{d}$.

3.3 Billiard PCP

In a convex setting, each (non-null) cut provide a half-space in which the optimum lies. With multiple cuts, we have a polyhedron in which the optimum necessarily lies. The billiard algorithm is a classical approach[10, 1, 7, 5] for sampling a polyhedron. We apply it, here and get $p-1$ points $\boldsymbol{x}_{n,2}, \ldots, \boldsymbol{x}_{n,p}$ approximately uniformly distributed in the polyhedron. $\boldsymbol{x}_{n,1}$ is set to $\boldsymbol{x}_n$.



(a) Sphere (b) Cigar

Fig. 1: **Dimension $d = 3$, 15 iterations,** $p = 5$. Comparison of Simple Regret (SR) based on the number of processors p. Interpretation: for the sphere, s-Billiard and s-Gaussian outperform the other variants when $p > 3$; for the cigar, billiard and s-Billiard outperform the other variants when $p > 3$.

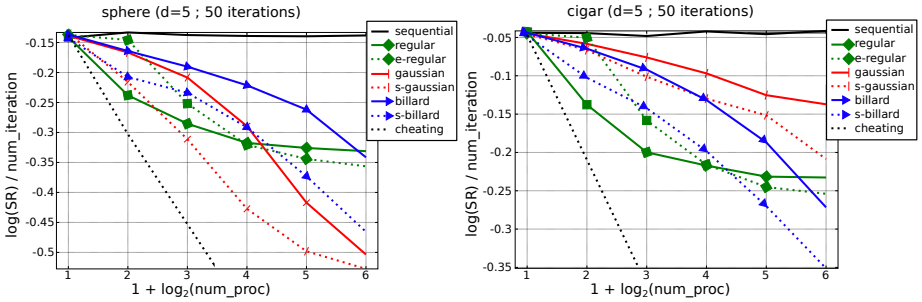
Special-billiard PCP The center of mass might be an interesting point; therefore, for this special billiard, we keep one point for the average of the $\boldsymbol{x}_{n,i}$ for $i \in \{2, \ldots, p-1\}$: $\boldsymbol{x}_{n,p} = \frac{1}{p-2} \sum_{i=2}^{p-1} \boldsymbol{x}_{n,i}$. This method is inspired by [9, 11]. It approximates the center of mass of the polyhedron of possible optima; a computationally faster method is also proposed in [15] (using an approximation of the centre of gravity by the center of the biggest ellipsoid).

4 Experiments

In this section we compare 2 regular parallel cutting plane methods (C=1 and C=1.5, from Section 3.1), 2 gaussian approaches (defined in Section 3.2) and 2 billiards methods from Section 3.3 to 2 baseline: *sequential* and *cheat*. The sequential method is the vanilla version where there are no parallelization. The cheat method can execute each cut in parallel as if it was in the sequential scenario, thus showing the absolute best possible outcome.

Simple regret We consider in our results the Simple Regret (SR). The simple regret of an optimization run is the difference $f(\hat{\boldsymbol{x}}) - \inf_{\boldsymbol{x}} f(\boldsymbol{x})$, where $\hat{\boldsymbol{x}}$ is the approximation of the optimum provided by the optimization algorithm at the end of the optimization run.

Comparison We compare our methods, as a function of the number p of processors. For the sake of comparison, we include a "cheating" method. The cheating method is in fact the sequential classical cutting-plane method, but with each group of p iterations being considered as one iteration only. This is not a real parallel method: it is cheating in the sense that processor #2 can use the result of the computation of processor #1, and so on. It is just aimed at showing the ideal



(a) Sphere (b) Cigar

Fig. 2: **Dimension $d = 5$, 50 iterations,** $p = 6$. Comparison of Simple Regret (SR) based on the number of processors p. Interpretation: for the sphere, Gaussian and s-Gaussian outperform the other variants when $p > 4$; for the cigar, billiard and s-Billiard outperform the other variants when $p > 5$.

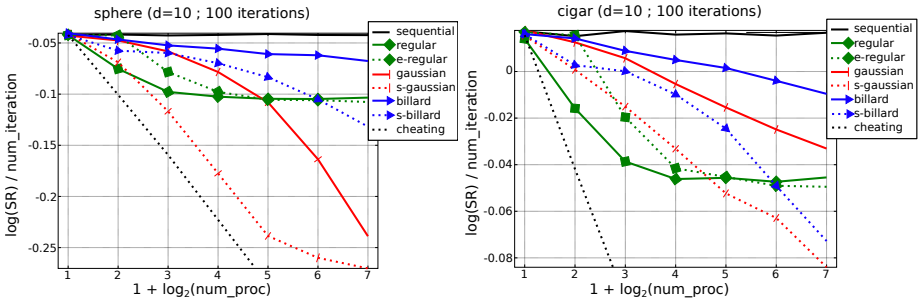
case, i.e. the performance that would be obtained if we could fully parallelize the cutting plane method so that p iterations on a sequential machine could be run simultaneously on a machine with p processors. For the sake of comparison, we also include a "sequential" method, in which we do not parallelize at all; only one processor is used, so that this is indeed the sequential method.

We consider the sphere function $f(\boldsymbol{x}) = ||\boldsymbol{x}||^2$ for $\boldsymbol{x} \in [0,1]^d$, and the cigar function $f(\boldsymbol{x}) = \boldsymbol{x}_1^2 + 10^3 \sum_{i=2}^{i=d} \boldsymbol{x}_i^2$. All cutting plane methods start with a point randomly uniformly drawn in $\{\boldsymbol{x} \in \mathbb{R}^d; ||\boldsymbol{x}|| = 1\}$. Results are averaged over 20 runs; each point in each curve is computed independently, so that superiority over each abscissa shows statistical significance.

Figure 1 explores the different variants with a fixed dimension $d = 3$, a number of iterations $n = 15$ and a number of processors $p = 5$. For the sphere, s-Billiard and s-Gaussian outperform the other variants when $p > 3$. For the cigar, billiard and s-Billiard outperform the other variants when $p > 3$. Our variants (s-Gaussian and s-Billiard) outperform their respective initial version (respectively Gaussian and Billiard).

Figure 2 evaluates the different variants with a fixed dimension $d = 5$, a number of iterations $n = 50$ and a number of processors $p = 6$. For the sphere, Gaussian and s-Gaussian outperform the other variants when $p > 4$. For the cigar, Billiard and s-Billiard outperform the other variants when $p > 5$. Once again, our variants (s-Gaussian and s-Billiard) outperform their respective initial version (respectively Gaussian and Billiard).

Figure 3 evaluates the different variants with a fixed dimension $d = 10$, a number of iterations $n = 100$ and a number of processors $p = 7$. For the sphere, Gaussian and s-Gaussian outperform the other variants when $p > 4$. For the cigar, s-Gaussian and s-Billiard outperform the other variants when $p > 5$. Once again, our variants (s-Gaussian and s-Billiard) outperform their respective initial version (respectively Gaussian and Billiard).



(a) Sphere (b) Cigar

Fig. 3: **Dimension $d = 10$, 100 iterations, $p = 7$.** Comparison of Simple Regret (SR) based on the number of processors p. Interpretation: for the sphere, Gaussian and s-Gaussian outperform the other variants when $p > 4$; for the cigar, s-Gaussian and s-Billiard outperform the other variants when $p > 5$.

Figure 4 presents the impact of the number of iterations upon the Simple Regret (SR) for the 10 dimensions sphere and cigar objective functions. A good variant typically would require a number of iteration as low as possible since each iteration represents a sequential step. The s-Gaussian is clearly the best variant when $p > 4$, whatever the number of iteration.

5 Conclusion

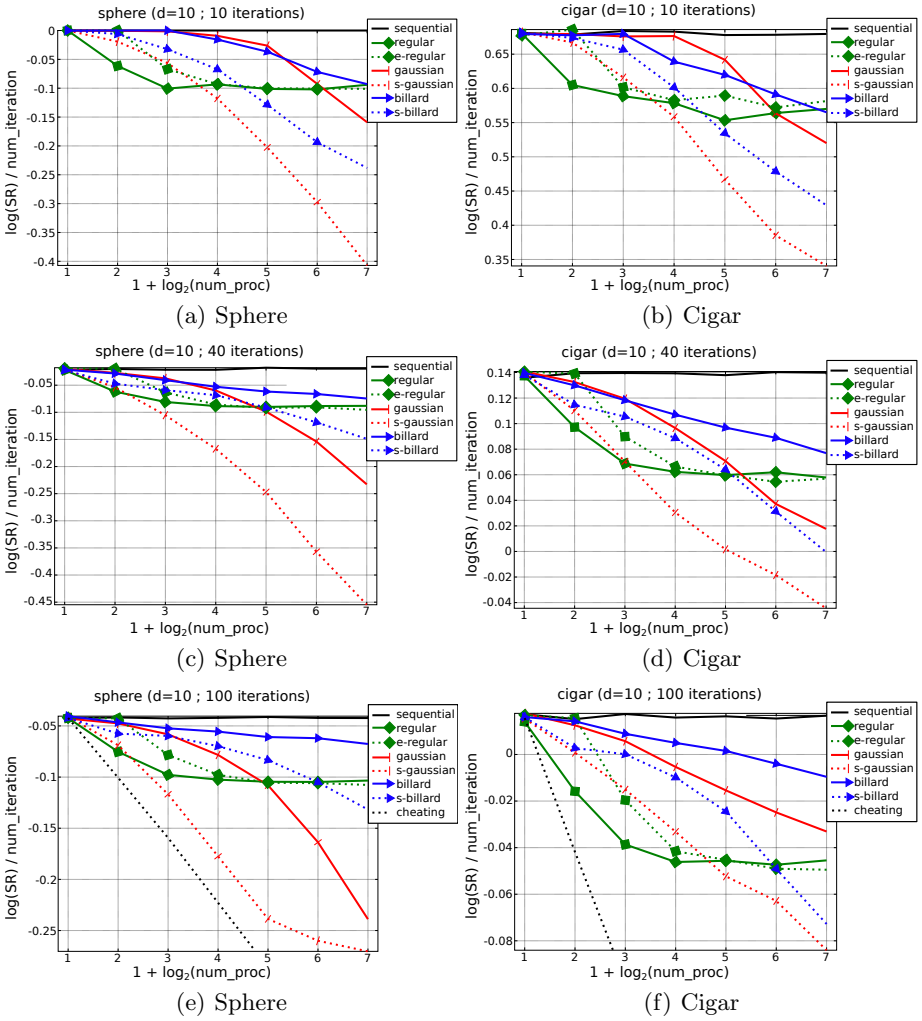
Our conclusions are as follows:

- The s-Billiard method is better than the billiard method. This is consistent with the superiority of the ellipsoid method. This might make sense even in the sequential case.
- Comparison between methods: the s-Gaussian method is the best method in the well conditioned case ("sphere"). The s-Billiard method is the best only in the ill-conditioned case ("cigar") when the dimension is small.
- Speed-up: the speed-up is reasonably good; we have nearly a half or a third of a linear speed-up (i.e. we need twice or three times more processors than in the "cheating" case).

References

1. P. Baldwin. The billiard algorithm and ks entropy. *Journal of Physics A: Mathematical and General*, 24(16):L941, 1991.
2. C. G. Broyden. The convergence of a class of double-rank minimization algorithms 2. *The New Algorithm. J. of the Inst. for Math. and Applications*, 6:222–231, 1970.
3. A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Compte-rendus hebdomadaires de l'académie des sciences*, pages 536–538, 1847.

4. R. Fletcher. A new approach to variable-metric algorithms. *Computer Journal*, 13:317–322, 1970.
5. T. Gensane. Dense packings of equal spheres in a cube. *the electronic journal of combinatorics*, 11(1):R33, 2004.
6. D. Goldfarb. A family of variable-metric algorithms derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.
7. R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines: Estimating the bayes point in kernel space. In *IJCAI Workshop SVMs*, pages 23–27, 1999.
8. J. E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.
9. A. Levin. An algorithm for the minimization of convex functions. *Soviet Math. Doklady*, 6:286–290, 1965.
10. B. D. Lubachevsky. How to simulate billiards and similar systems. *Journal of Computational Physics*, 94(2):255–283, 1991.
11. D. J. Newman. Location of the maximum on unimodal surfaces. 12(3):395–398, July 1965.
12. M. V. F. Pereira and L. M. V. G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Math. Program.*, 52(2):359–375, Oct. 1991.
13. M. J. D. Powell. Developments of newuoa for minimization without derivatives. *IMA J Numer Anal*, pages drm047+, February 2008.
14. D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24:647–656, 1970.
15. S. Tarasov, L. Khachiyan, and I. Erlikh. The method of inscribed ellipsoids. *Soviet Mathematics Doklady*, 37(1):226–230, 1988.



(a) Sphere (b) Cigar

(c) Sphere (d) Cigar

(e) Sphere (f) Cigar

Fig. 4: **Dimension $d = 10$; 10, 40 and 100 iterations;** $p = 7$. Comparison of Simple Regret (SR) based on the number of processors p. Interpretation: for the sphere and the cigar, s-Gaussian outperform the other variants when $p > 4$, whatever the number of iteration.