Sieves method in fuzzy control: logarithmically increase the number of rules

Vincent Berthier, Olivier Teytaud

Sieves method in fuzzy control: logarithmically increase the number of rules

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

***Abstract*—The Sieves method, in statistics, consists in extending a model progressively, as new data are made available. Typically, parameters are progressively added in a statistical estimation method while new samples are provided. We propose an adaptation of the Sieves method in optimization. Decision variables are progressively added while new fitness evaluations are received. We experiment the method on a simple set of noisy optimization problems, and then on a fuzzy control problem applied to unit commitment. The obtained algorithm is simple, applicable to various optimization algorithms (not only evolutionary optimization), and seemingly robust.**

I. Introduction

The Sieves method consists in progressively adding variable in a computational intelligence problem. While it is classical in statistics, both supported by a range of experiments and a body of theoretical works, the Sieves method is unusual in optimization. In this paper we (i) propose an adaptation of the Sieves method in optimization, (ii) perform artificial and (iii) perform real world experiments. Section I-A describes the classical Sieves method in statistics. Section I-B proposes a Sieves method in evolutionary optimization. Section II presents artificial experiments. Section III presents an application in fuzzy control.

A. Sieves method (SM)

Overfitting is the poor behavior of parametric decision tools when the parameters are tuned on a too small dataset. Typically, overfitting occurs when an optimization run is too short, compared to the number of parameters to be tuned. Overfitting is widely discussed in supervised machine learning[1], but not that much in optimization and control. We will see (Fig. 4) that overfitting does matter in Fuzzy control, and we propose a method for avoiding it. There is a long tradition of considering the impact of dimension in optimization, statistics, machine learning. In statistics, this has given birth to the Sieves method: instead of considering a huge problem at once, we work on a small sub-problem (with far less variables), and then we progressively add other variables[2]. In optimization, a different approach has been preferred, based on decomposition: instead of working on a high-dimensional optimization problem, we work on several subproblems, in a divide-and-conquer manner. For example, in mathematical programming, the Lagrangean decomposition[3] involves a master, coordinating the global optimization, and several subproblems. In evolutionary computation, some main successes are based on the decomposition of a big problem into several subproblems, with an evolutionary coordination, as in divide-and-evolve[4]. A particular form of noisy optimization is simple regret bandits[5]. In such a setting, there is a counterpart of the Sieves method, namely progressive widening ([6], [7] for the experimental part, [8] for a mathematical analysis). However, these works on progressive widening are limited to discrete sets of actions with little or no structure. There has been little effort on the application of Sieves methods to optimization. We here consider Sieves methods for noisy optimization.

B. Sieves method for evolutionary algorithms

Evolutionary algorithms are population-based metaheuristics. At each iteration, offspring are generated. Each individual (offspring) is evaluated. The best individuals become the parents and generate the next offspring. As the optimization algorithm is not the core of this paper, we refer to [9], [10] for more details on SAES; the Sieves method that we propose can be implemented for other evolution strategies without changing the principles. We use a Sieves index $SievesIndex(i,d)$ which decides, at iteration i and if the problem dimension is d, how many coordinates are considered. The pseudo-code of the Sieves method for (μ,λ)-SAES is presented in Alg. 1. A $(\mu/\mu,\lambda)$-SAES version is also used. It is recalled in Alg. 2.

Algorithm 1 Sieves method applied to SAES in dimension d. $\mathcal{N}$ denotes independent centered standard Gaussian (in dimension given by the context). The fitness function is obtained by averaging multiple reevaluations, as detailed in the experimental section. $Sieves(z,i,d) = (z_1,\dots,z_{SievesIndex(i,d)},0,0,\dots,0)$ for some $SievesIndex$ function (see text).

Parameters: parent population size μ, step-size mutation rate τ, population size λ, initial parent population Pop, each parent is $Pop_i = (x_i,\sigma_i)_{i\in\{1,\dots,\lambda\}}$ with σ_i initialized at some σ_0.
$i \leftarrow 0$
while Computation time not exhausted **do**
  $i \leftarrow i+1$
  for $j \in \{1,\dots,\mu\}$ **do**
    $\sigma_j \leftarrow \sigma_j \times \exp(\tau\mathcal{N})$
  end for
  for $j \in \{1,\dots,\lambda\}$ **do**
    $k \leftarrow mod(j,\mu)+1$
    $o_j = x_k + \sigma_j Sieves(\mathbb{N},i,d)$
    $f_j = fitness_{reevaluations}(o_j)$
  end for
  Sort individuals, so that $f_1 \le f_2 \le \dots \le f_\lambda$
  $\forall i \in \{1,\dots,\mu\}, Pop_i \leftarrow (o_i,\sigma_i)$
end while

Algorithm 2 Sieves method applied to SAES. $\mathcal{N}$ denotes independent centered standard Gaussian (in dimension given by the context). The fitness function is obtained by averaging multiple reevaluations, as detailed in the experimental section.

Parameters: parent population size μ, step-size mutation rate τ, population size λ, initial parent x, initial step-size $\sigma = \sigma_0$
$i \leftarrow 0$
while Computation time not exhausted **do**
  $i \leftarrow i + 1$
  for $j \in \{1, \dots, \lambda\}$ **do**
    $\sigma_j = \sigma \exp(\tau \mathcal{N})$
    $o_j = x + \sigma_j Sieves(\mathbb{N}, i, d)$
    $f_j = fitness_{reevaluations}(o_j)$
  end for
  Sort individuals, so that $f_1 \leq f_2 \leq \dots \leq f_\lambda$
  $x \leftarrow \frac{1}{\mu} \sum_{j=1}^{\mu} o_j$
  $\sigma \leftarrow \exp(\frac{1}{\mu} \sum_{j=1}^{\mu} \ln(\sigma_j))$
end while

II. Artificial experiments

In a noisy optimization problem, the objective function f depends on the chosen search point θ and on a noise component ω. We consider the optimization of the following simple noisy sphere problem: $f(\theta, \omega) = ||\theta - \theta^*||^2 + \omega$, where ω is an independent centered standard Gaussian noise. The optimization is performed by a self-adaptive evolution strategy (SAES[9]). The optimum is θ^*, which is randomly drawn as follows: $\forall j \in \{1, 2, \dots, 100\}, \theta^*_j = \mathcal{N}/j^\zeta$ for various ζ (discussion later) and the initial search point for SAES is 0. $\zeta > 0$ indicates that the first variable is (on average) more important than the second, which is more important than the third and so on. The initial step-size σ_0 is the same for all coordinates, and we test several values. Each experiment is reproduced 57 times. Each function evaluation is repeated several times in order to mitigate the level of noise. More precisely, each evaluation at iteration i is repeated i^2 times and the obtained fitness values are averaged[11]. We apply the Sieves method for optimization as follows. We compare several $sievesIndex(i, d)$ functions:

$$sievesIndex(i, d) = \lceil \frac{d}{1 + 9(k-1)} i^{(1+9(k-1))^{-\frac{1}{2}}} \rceil$$

for $k \in \{1, \dots, 9\}$. $k = 1$ means no Sieves method. k large means a strong Sieves method. With $\zeta = 1.5$ we get results as presented in Fig. 1(a), Fig. 1(b), depending on the initial step-size σ_0 (too small, too large, respectively), with isotropic-SAES. (Results with the correct initial step-size are not presented due to length constraints but lead to similar conclusions.) $k = 6$ is seemingly always good, leading to $sievesIndex(i, d) = \lceil \frac{20}{9} i^{1/\sqrt{46}} \rceil \simeq 2i^{.15}$. We reproduced the experiments with various values of ζ. Results were always positive with $\zeta > 1$ (i.e. the Sieves method worked better than the original method), but the optimal k was not always the same and $k = 6$ was not always better than no Sieves. With $\zeta < 1$, results were not good, whatever maybe k. We therefore conclude these preliminary experiments as follows:

- The Sieves method in optimization does not work when all coordinates are approximately equally important ($\zeta < 1$).
- The optimal parametrization of the $SievesIndex$ function is problem-dependent, so some suboptimal parametrizations seemingly cover a wide range of problems.

In the next section, we focus on fuzzy control, which is arguably a natural candidate for the Sieves method in optimization: there are many parameters, and the ranking of the parameters is easy: a fuzzy controller is a combination of rules, the ordering of which does not matter, and the first rules are more important than the next ones. We got in the experiments above a polynomial Sieves function with a very small exponent, indeed close to a logarithmic function; we will see that a single logarithmic function will work fine on several of our fuzzy control problems.

III. Applications to fuzzy control

Section III-A introduces direct policy search. Section III-B briefly presents fuzzy control. Section III-C presents our optimization problem. Section III-D presents our optimization algorithm. Section III-E presents preliminary experiments. Section III-F presents our parametrization of the SM. Section III-G shows our results.

A. Direct Policy Search and noisy optimization

Direct Policy Search (DPS) consists in optimizing a parametric policy directly on simulations, i.e. the objective function simulates the whole system with the parametric decision policy. There are many such works[12] with neural networks, and in fuzzy systems[13]. We here focus on fuzzy systems. Technically, DPS boils down to noisy optimization. Noisy optimization can be performed by various methods, including gradient-free gradient descent[14] (the gradient is estimated by differences) or evolutionary algorithms[15], [16]. We here use evolutionary algorithms.

B. Fuzzy control

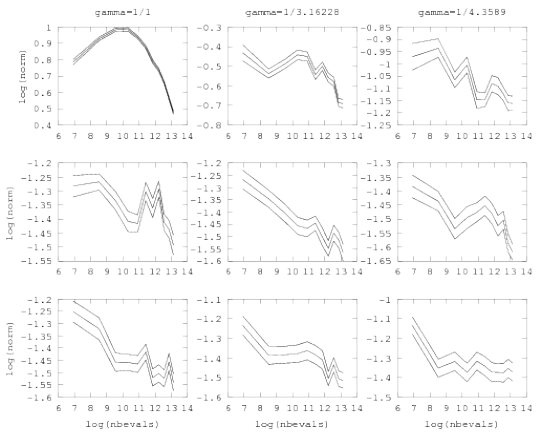
We consider fuzzy functions as follows:

$$\begin{aligned} action(s) &= \sum_{i=1}^{s} \frac{w_i(s)}{\sum_{j=1}^{s} w_j(s)} \theta_{i,1} \\ w_i(s) &= \\ I(\theta_{i,2}) &\times \prod_{j=1}^{d} \max\left(0, \frac{1 - |s_j - \theta_{i,3,j}|}{\exp(d + \theta_{i,4,j})}\right) \end{aligned} \quad (1)$$

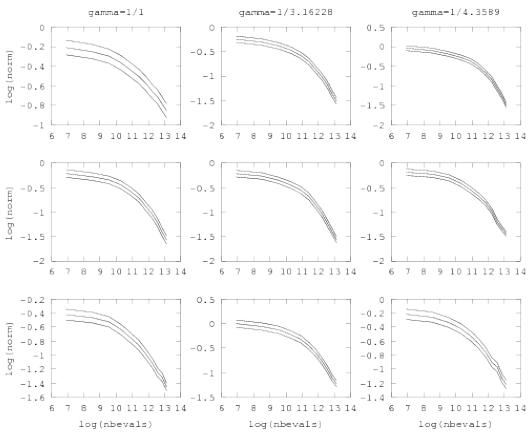
where $I : x \mapsto \frac{1}{2}(1 + x/\sqrt{x^2 + 1})$ is a mapping from $\mathbb{R}$ to $[0, 1]$, and where d is the dimension of the state space s and $action(s)$ is the chosen action for state s. As we consider methods adding progressively new rules, we also tested the replacement of Eq. 1 by Eq. 2, and also by Eq. 3; there are aimed at reducing the weights of newly added rules. We mention them for the sake of completeness, but they did not bring any clear improvement:

$$\begin{aligned} w_i(s) &= \\ I(\theta_{i,2} - 7) &\times \prod_{j=1}^{d} \max(0, 1 - \frac{|s_j - \theta_{i,3,j}|}{\exp(d + \theta_{i,4,j})}) \end{aligned} \quad (2)$$

$$\begin{aligned} w_i(s) &= \\ I(\theta_{i,2} - 2i) &\times \prod_{j=1}^{d} \max(0, 1 - \frac{|s_j - \theta_{i,3,j}|}{\exp(d + \theta_{i,4,j})}) \end{aligned} \quad (3)$$



(a) Results for $k = 1, 2, \ldots, 9$, for an initial step-size $\sigma_0 = 2.5$ i.e. way too large. $k = 1$ means no Sieves method. $k = 6$ is one of the good values. The three curves are average, average + standard deviation, average - standard deviation, respectively.



(b) Results for $k = 1, 2, \ldots, 9$, for an initial step-size $\sigma_0 = 5/54$ i.e. way too small. $k = 1$ means no Sieves method. $k = 6$ is one of the good values. The three curves are average, average + standard deviation, average - standard deviation, respectively.

Fig. 1. Impact of the k parameter on the Sieves method.

This means that the parameter is $\theta = (\theta_{i,1}, \theta_{i,2}, \theta_{i,3}, \theta_{i,4})_{i \in \{1,\ldots,NbRules\}}$ when the number of rules is $NbRules$. $\theta_{i,1}$ is the action chosen by the rule number i; $\theta_{i,2}$ is the a priori weight of the rule number i; $\theta_{i,3}$ is the typical state at which the rule applies; $\theta_{i,4}$ indicates the scope of the rule (the larger $\theta_{i,4,j}$, the wider the scope of the rule on axis j). We also tested a different membership function, i.e. $w_i(s)$ having no prior weight:

$$w_i(s) = \prod_{j=1}^{d} \max\left(0, \frac{1 - |s_j - \theta_{i,3,j}|}{\exp(d + \theta_{i,4,j})}\right). \quad (4)$$

Parameter	Value
Variant	DE/curr-to-best/1
	i.e. $x'_i = x_i + F1(x_a - x_b) + F2(x_{best} - x_i)$
	with x'_i the proposal, a and b random,
	x_{best} the best in the population.
Population size	30
Cr	0.5
F1	0.8
F2	0.8
Resampling	$10\sqrt{n}$ resamplings
	at iteration n

TABLE I
PARAMETERS OF OUR DE ALGORITHM WHEN OPTIMIZING N PARAMETERS, AT ITERATION i.

C. Objective function

The test case is a unit commitment problem. We have 21 time steps, 5 hydroelectric stocks, and thermal units. The demand is stochastic, and the point is to meet the demand (i.e. production should be equal to the demand at all time steps). The objective function is the production cost. The part of the demand which is not met by the hydroelectricity is produced by thermal power.

D. Optimization algorithm

The optimization algorithm is a Differential Evolution[17]. We use the same resampling policy as above, with coefficients optimized on the target problem, before including our Sieves method. The parameters used are presented in Table I.

E. Preliminary experiments

There are many fuzzy membership functions available in the literature[18], [19], so we conducted experiments for comparing some of them. Eq. 1 uses as a distance the product, over coordinates, of linearly decreasing member functions. We also tested Eq. 4, for checking the robustness of the method. We also tested two different problems, both however in the family of unit commitment.

F. Sieves parametrization

$$\text{Logarithmic Sieves:} SievesIndex(t) = nbParamsPerRule \times \lceil \log_2(1+t)/rate \rceil$$
$$\text{Linear Sieves:} SievesIndex(t) = nbParamsPerRule \times \lceil t/rate \rceil.$$

The parameter $rate$ is chosen so that for a given computational budget we reach approximately the given maximal number of rules at the end of the budget. $SievesIndex$ is also limited to the total number of rules considered in the function. We indeed compared many formulas logarithmically increasing and roughly converging to dozenz of rules within minutes; many where ok, V4 is a typical one and V2 is appproximately the best one. These formulas are anytime; we do not have to know in advance when the algorithm will be stopped. All these formulas were quite similar; we tested a hard limit depending on the total budget in the formula. This destroys the anytime nature of the formula; moreover, we did not have any positive impact - so we kept the original anytime formulas.

This leads to simple formulas, and the recommendation that, maybe, the number of rules should increase logarithmically with time, converging to dozen of rules within reasonable human time constants for the problem at hand. Importantly, many of the $SievesIndex$ functions that we have designed were performing better than the default method, suggesting that the method is somehow robust.

G. Experimental results: Sieves method for fuzzy control

We perform experiments with membership functions as in Eq. 4 in Figs. 2 and 4. We then perform experiments with membership functions as in Eq. 1 in Figs. 3 and 5. Figs. 2 and 4 correspond to 5 stocks and 25 time steps. Figs. 3 and 5 correspond to 15 stocks and 50 time steps. Figs. 6, 7, 8, 9 reproduce the results of Figs. 2, 3, 4, 5, with logarithmic Sieves instead of linear Sieves. They are just aimed at validating the methodology on different settings. We get positive results in all cases; in some cases the improvement is only in asymptotic cases, with seemingly less local minima than without Sieves method. One might ask whether the improvement is because of the limited number of rules used during the optimization run, or if the SM of the number of rules provides an improvement. Actually, the SM helps, and this is shown in additional experiments in Figs. 2, 3, 4, 5, where we compare what happens with several hard limits on the number of rules. We get an improvement in all cases, asymptotically.

IV. Conclusions and further work

Fuzzy control is a wide research area. The Sieves method is classical in machine learning. We extended the Sieves method to noisy optimization, including experiments in fuzzy control. Results are clearly positive. The simple recommendation is that the number of rules should increase somewhere between logarithmically and linearly with the number of iterations. Results for early stages of the optimization are sometimes positive and sometimes not, but asymptotically in all cases the Sieves method provided better results. Seemingly, it avoids local minima. The classical mathematical results for justifying the method of Sieves are based on VC-dimension arguments or other statistical complexity measure (see e.g. [20], [2]). They do not apply here, because it is hard to compute the VC-dimension or covering numbers of level sets of fitness functions involved in control problems[21]. Based on our experiments, we conjecture that a logarithmically increasing number of rules is a reasonable solution. The parameters had little impact on our results, provided the rule was logarithmic, starting at 1, and converging to a few dozen of rules within a reasonable time for the problem at hand. Using this method, we can work without any hard constraint on the number of rules; just add them progressively during the optimization run, in an anytime manner. Our work is limited to DE. Our choice of parameters for the Sieves index might have to be made adaptive - though we tested many rules and all of them were satisfactory under this "start with one rule and concavely add new rules, reaching a few dozen within time constants of the optimization problem". The artificial setting was tested with various initial step-sizes and several values of the problem parameter ζ. The optimal parametrization is not always the same, but with a slow increase we almost always outperform the baseline if $\zeta > 1$ - which means that some parameters are more imporant than others and that we know which ones. We consider as a main goal the construction of "universal" rules for choosing the rate at which rules should be added; maybe validating our rule above on more testcases of refining it. Sieves method are classical in many fields; there is room for them in optimization.

References

[1] V. N. Vapnik, *The Nature of Statistical Learning*. Springer Verlag, 1995.

[2] X. Shen and W.-H. Wong, "Convergence rate of sieve estimates," *The Annals of Statistics*, vol. 22, no. 2, pp. 580–615, 1994.

[3] M. P. Nowak and W. Rmisch, "Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty," *Annals of Operations Research*, vol. 100, pp. 251–272, 1998.

[4] M. Schoenauer, P. Savéant, and V. Vidal, "Divide-and-evolve : une nouvelle méta-heuristique pour la planification temporelle indépendante du domaine," in *Journées Francophones Planification, Décision, Apprentissage*, F. G. et Gérard Verfaillie, Ed. Toulouse: GDR I3 groupe PDMIA, 2006. [Online]. Available: http://hal.inria.fr/inria-00121779/en/

[5] S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in finitely-armed and continuous-armed bandits," *Theor. Comput. Sci.*, vol. 412, no. 19, pp. 1832–1852, 2011.

[6] R. Coulom, "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," *In P. Ciancarini and H. J. van den Herik, editors, Proceedings of the 5th International Conference on Computers and Games, Turin, Italy*, 2006.

[7] G. Chaslot, M. Winands, J. Uiterwijk, H. van den Herik, and B. Bouzy, "Progressive Strategies for Monte-Carlo Tree Search," in *Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007)*, P. Wang *et al.*, Eds. World Scientific Publishing Co. Pte. Ltd., 2007, pp. 655–661.

[8] Y. Wang, J.-Y. Audibert, and R. Munos, "Algorithms for infinitely many-armed bandits," in *Advances in Neural Information Processing Systems*, vol. 21, 2008.

[9] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[10] H.-G. Beyer, *The Theory of Evolution Strategies*, ser. Natural Computing Series. Springer, Heideberg, 2001.

[11] S. Astete-Morales, J. Liu, and O. Teytaud, "log-log convergence for noisy optimization," in *Proceedings of EA 2013*, ser. LLNCS. Springer, 2013, p. accepted.

[12] Y. Bengio, "Using a financial training criterion rather than a prediction criterion," CIRANO, CIRANO Working Papers 98s-21, 1998. [Online]. Available: http://ideas.repec.org/p/cir/cirwor/98s-21.html

[13] L. A. Zadeh, "The birth and evolution of fuzzy logic," *Int. J. of General Systems*, pp. 95–105, 1990.

[14] V. Fabian, "Stochastic Approximation of Minima with Improved Asymptotic Speed," *Annals of Mathematical statistics*, vol. 38, pp. 191–200, 1967.

[15] V. Heidrich-Meisner and C. Igel, "Hoeffding and bernstein races for selecting policies in evolutionary direct policy search," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 401–408.

[16] D. V. Arnold and H.-G. Beyer, "A general noise model and its effects on evolution strategy performance," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 380–391, 2006.

[17] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1008202821328

[18] J. G. Monicka, D. N. Sekhar, and K. R. Kumar, "Article: Performance evaluation of membership functions on fuzzy logic controlled ac voltage controller for speed control of induction motor drive," *International*

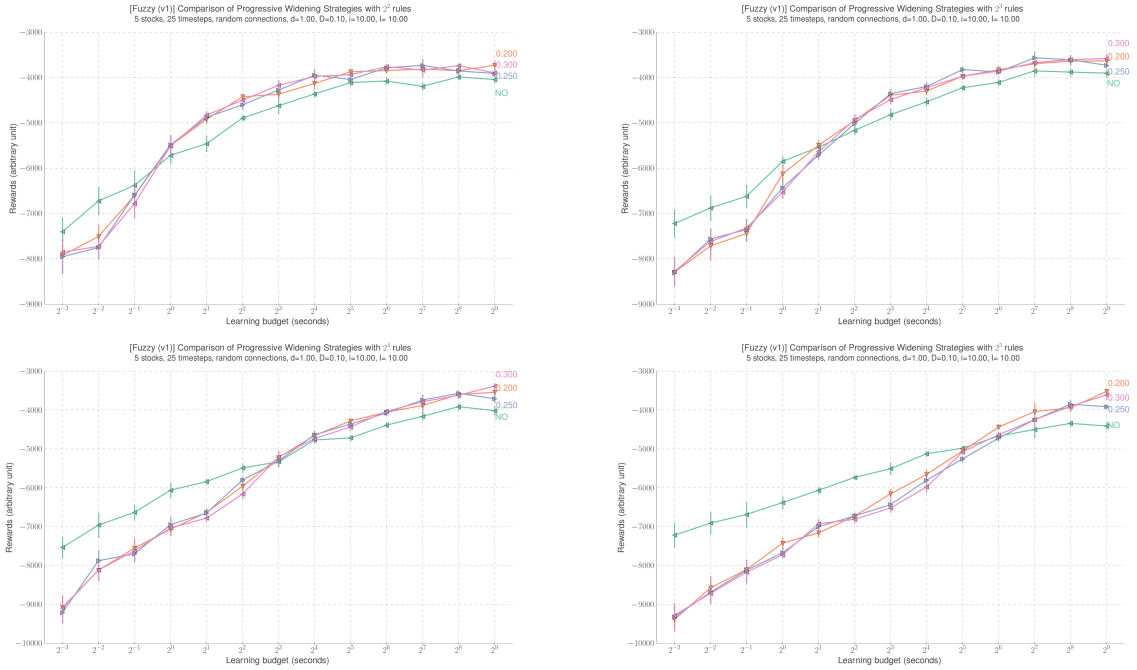


Fig. 2. Results with Eq. 4 and the smaller testcase, with linear Sieves. Circles and "NO" legend: no Sieves method. Triangles correspond to Sieves methods with various rates (see legend), in the case of a linear rate. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works asymptotically.

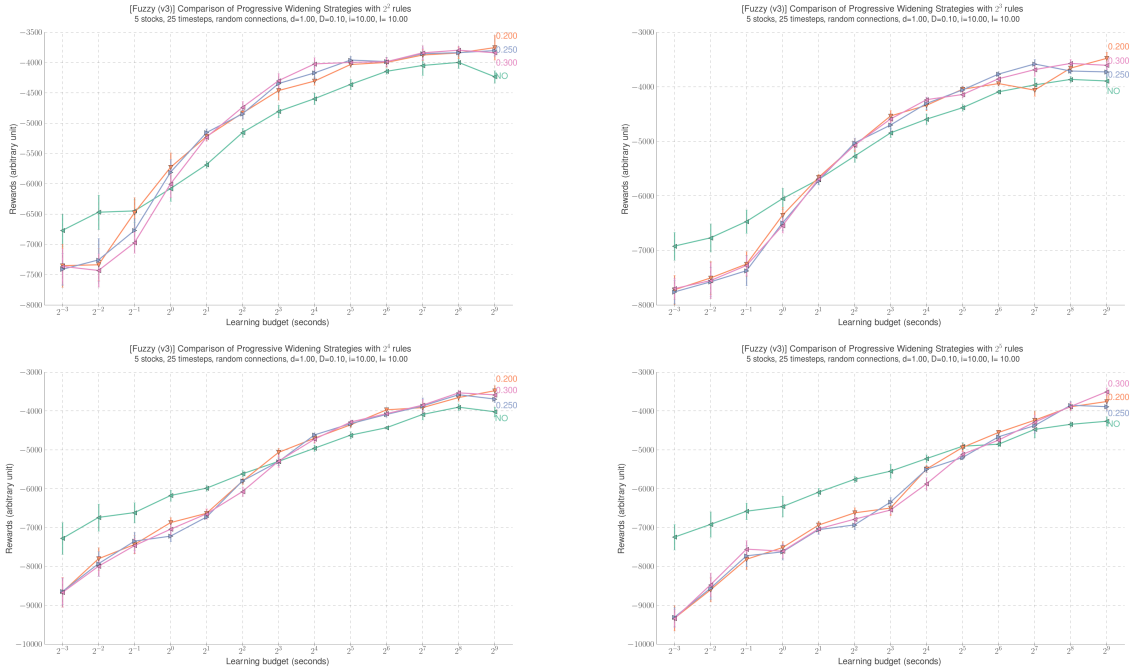


Fig. 3. Case of Eq. 1, with the smaller test case, and linear Sieves. "NO" stand for the default version, with no Sieves method. Triangles correspond to Sieves methods, with rate given in the legend. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works.

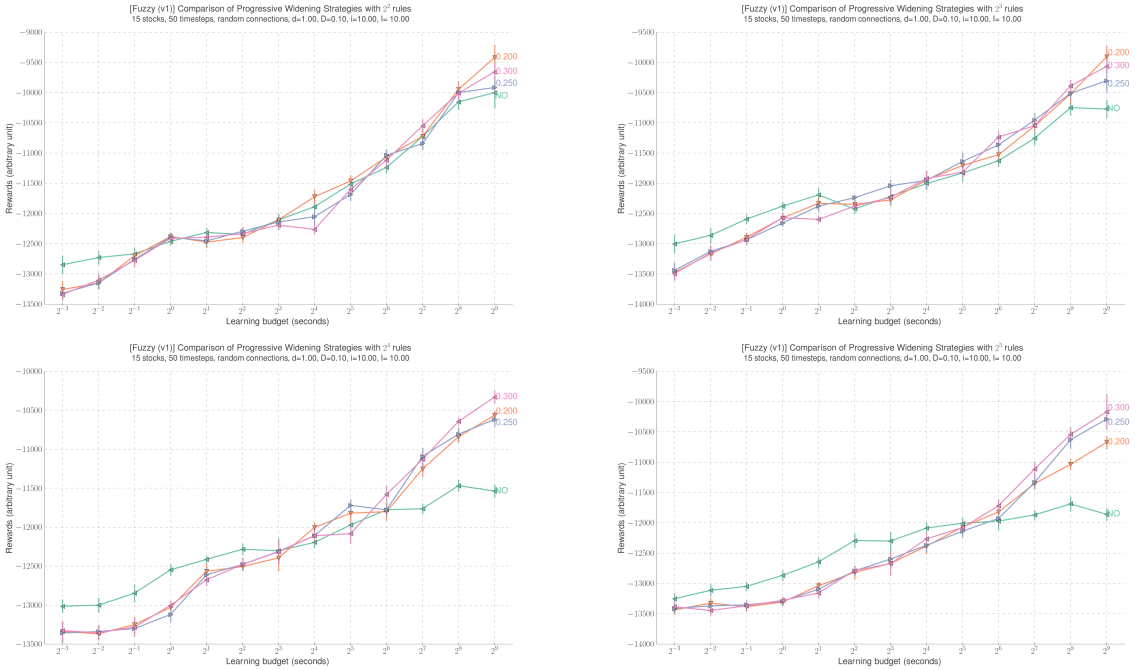


Fig. 4. Case of Eq. 4, with the large test case, linear Sieves. "NO" stands for the default version, with no Sieves method. Triangles correspond to Sieves methods, with rate given in the legend. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works.

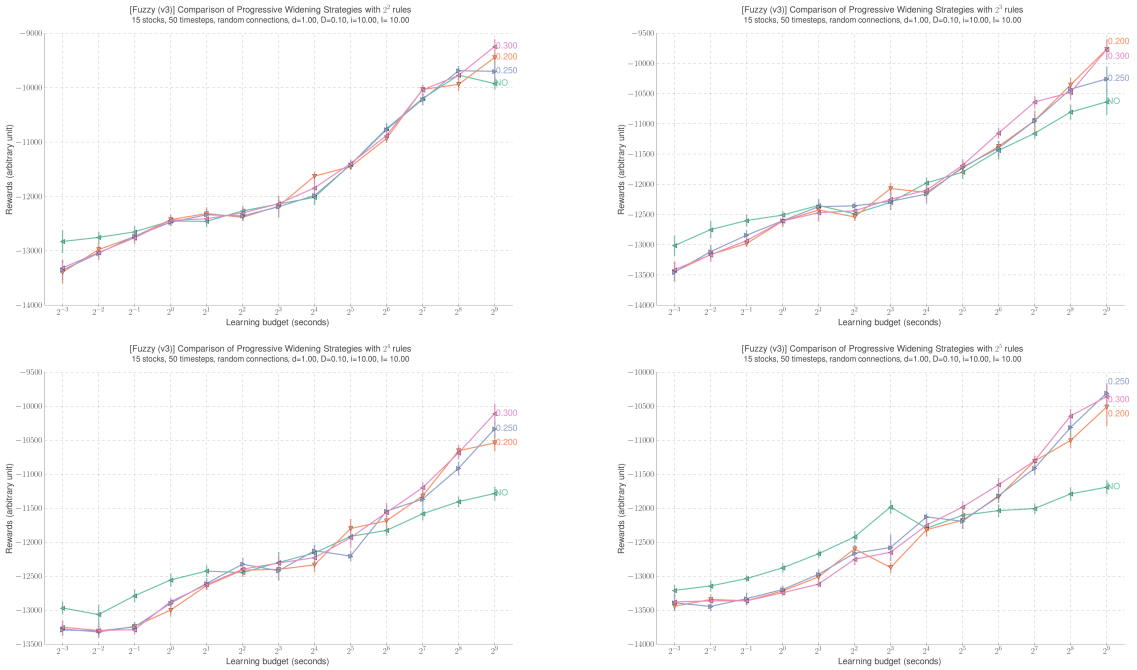


Fig. 5. Case of Eq. 1, large test case, linear Sieves. "NO" stands for the default version, with no Sieves method. Triangles correspond to Sieves methods, with rate given in the legend. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works.

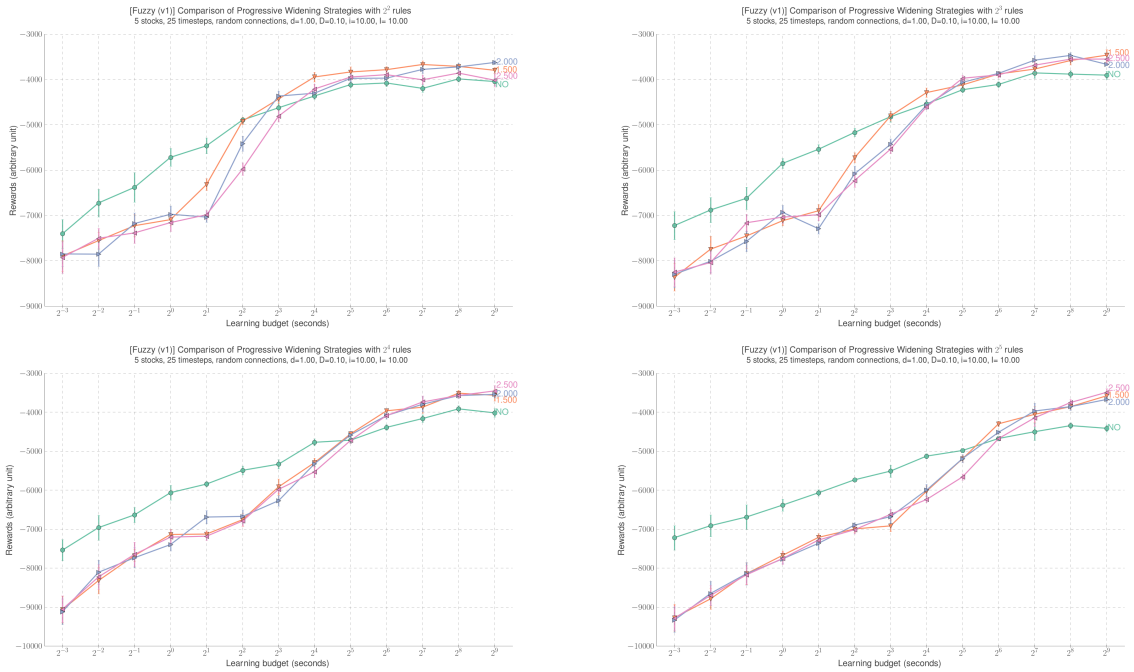


Fig. 6. Results with Eq. 4, smaller test case, and the smaller testcase, logarithmic Sieves. "NO": no Sieves method. Triangles correspond to Sieves methods with various rates, in the case of a linear rate. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works asymptotically.

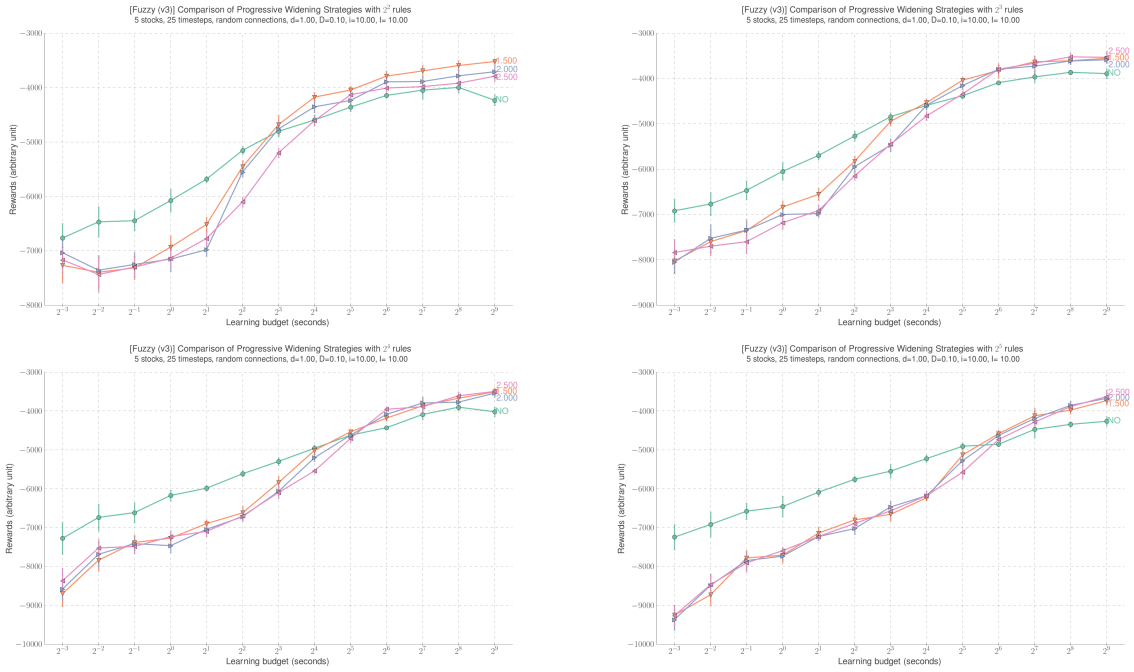


Fig. 7. Case of Eq. 1, smaller test case, logarithmic Sieves. "NO" stands for the default version, with no Sieves method. Triangles correspond to Sieves methods, with rate given in the legend. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works.

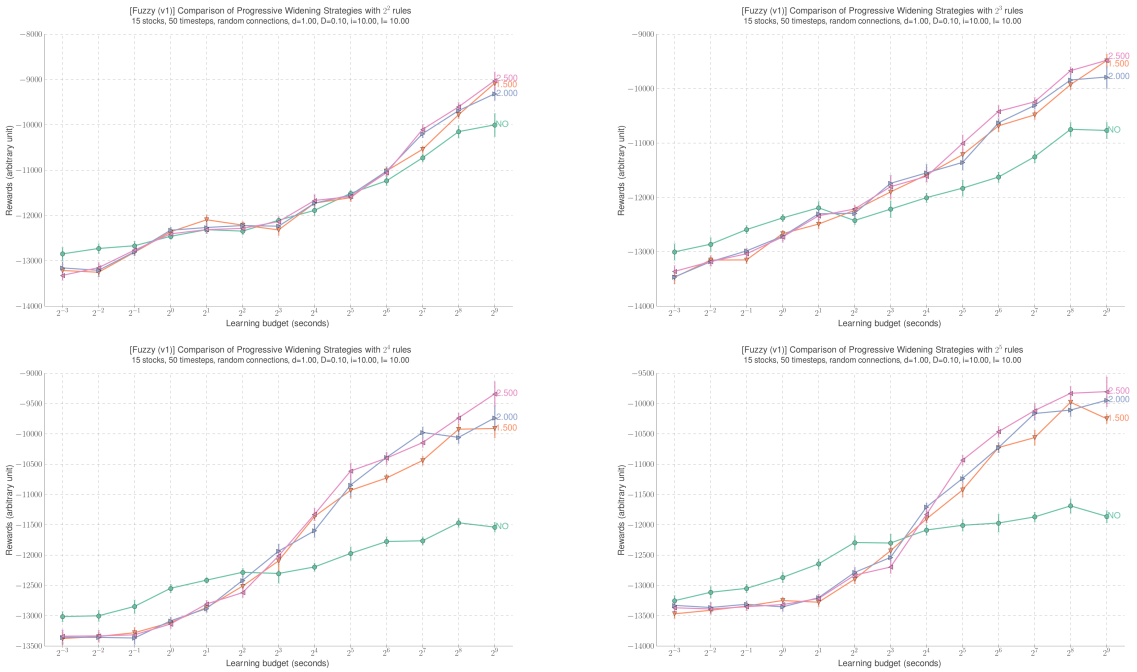


Fig. 8. Case of Eq. 4, large test case, logarithmic Sieves. "NO" stands for the default version, with no Sieves method. Triangles correspond to Sieves methods, with rate given in the legend. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works.

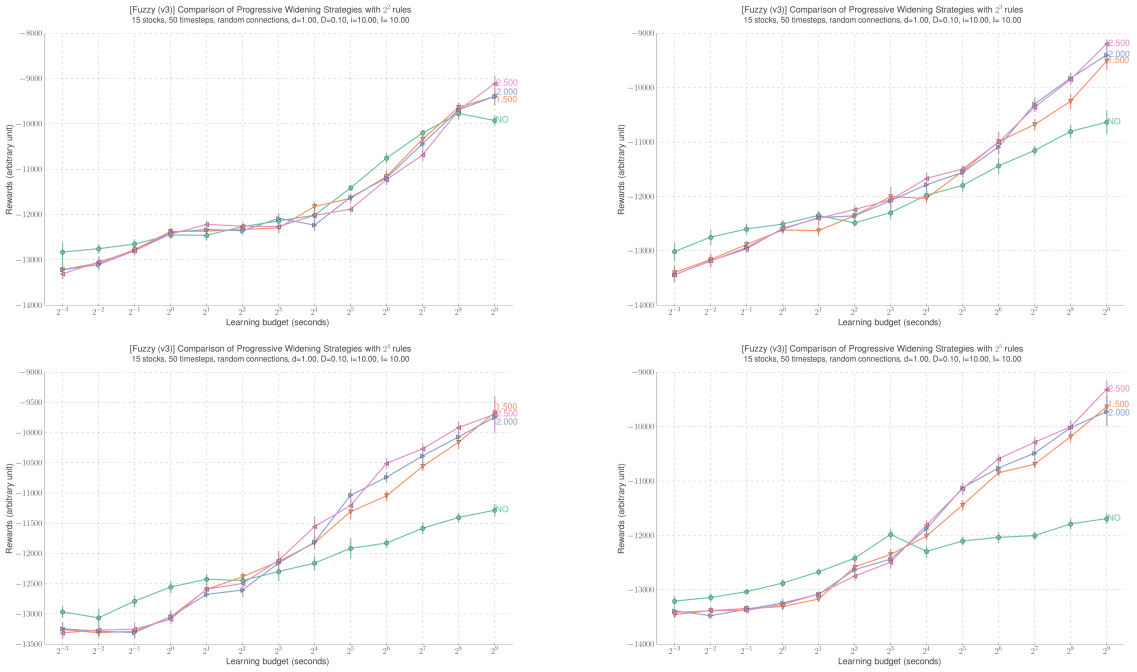


Fig. 9. Case of Eq. 1, large test case, logarithmic Sieves. "NO" stands for the default version, with no Sieves method. Triangles correspond to Sieves methods, with rate given in the legend. We compare our Sieves method to cases in which we restrict a priori the number of rules to 1, 2, 4, 8, 16 rules. More precisely, in each case, there is the default method (fixed number of rules) and our method (variable number of rules, reaching the same number for the largest budget). Basically, the Sieves method always works.

Journal of Computer Applications, vol. 13, no. 5, pp. 8–12, January 2011, full text available.

[19] J. Zhao and B. Bose, "Evaluation of membership functions for fuzzy logic controlled induction motor drive," in *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, vol. 1, Nov 2002, pp. 229–234 vol.1.

[20] L. Devroye, L. Györfi, and G. Lugosi, *A probabilistic Theory of Pattern Recognition*. Springer, 1997.

[21] M. Vidyasagar, *A Theory of Learning and Generalization*. New York, New York: Springer-Verlag, 1997.