# Expected improvements for the asynchronous parallel global optimization of expensive functions : potentials and challenges

J. Janusevskis[1], R. Le Riche[1,2], D. Ginsbourger[3], and R. Girdziusas[1]

[1] Ecole des Mines de Saint-Etienne, Saint-Etienne, France,
[2] CNRS, UMR 5146 Cl. Goux, France,
[3] Departement of Mathematics and Statistics, University of Bern, Switzerland.

**Abstract.** Sequential sampling strategies based on Gaussian processes are now widely used for the optimization of problems involving costly simulations. But Gaussian processes can also generate parallel optimization strategies. We focus here on a new, parameter free, parallel expected improvement criterion for asynchronous optimization. An estimation of the criterion, which mixes Monte Carlo sampling and analytical bounds, is proposed. Logarithmic speed-ups are measured on 1 and 9 dimensional functions.

## 1 Introduction to parallel expected improvements

The current technological solution for optimizing functions of numerically costly simulators is to rely on an increasing number of processing units (processors, cores, GPUs). Demanded features of new parallel optimization algorithms are not only high speed-ups but also the ability to work with heterogeneous processing units (e.g., computing grids) and fault tolerance. Evolutionary algorithms offer many opportunities for parallelization, including in heterogeneous computing networks, and in master-worker or island computing structures [2]. Master-worker parallel optimization algorithms are more common as they fit the costly simulator case: the optimizer is the master node, the workers evaluate the objective functions and constraints. In [6] for example, a parallel and asynchronous version of the local pattern search algorithm is described. [9] presents a deterministic global parallel algorithms which, contrarily to the forthcoming method is based on radial basis functions and has a synchronization step. [10] and [1] describe methods where Gaussian processes are used through expected improvement maximization with restarts and infill sampling, respectively, to provide the set of points to evaluate in parallel. Boths methods have a synchronisation step when the master optimizer iterates.

This article also proposes a sampling criterion based on Gaussian processes. Its originality relies on an asynchronous extension of the multi-points expected improvement of [3] which, itself, was a parallel extension of the expected improvement of [8]. In the sequel, the parallel asynchronous expected improvement will be denoted $EI^{(\mu,\lambda)}$.

A distinctive feature of $EI^{(\mu,\lambda)}$ is to provide a unified mathematical treatment of the already evaluated and the currently running points for optimization, therefore no additional parameter is introduced to parallelize the search. Initial empirical results on 1D and 9D functions show that logarithmic speed-ups are obtained. This work also illustrates that $EI^{(\mu,\lambda)}$ is difficult to compute.

## 2 A unified presentation of parallel expected improvements

Let us consider the optimization problem $\min_{x\in\mathbb{R}^d} f(x)$ where each evaluation of $f$ implies a call to a numerically intensive simulation program, and assume that a set of past observations $(\mathbb{X}, f(\mathbb{X}))$ has been gathered. In the Bayesian Global Optimization settings considered here, the unknown $f$ is represented *a priori* by a Gaussian Process $(Y(x))_{x\in\mathbb{R}^d}$, and is being approximated relying on the conditional distribution of $Y$ knowing that $Y(\mathbb{X}) = f(\mathbb{X})$ (*Kriging metamodel*). We focus here on cases where $\lambda$ computing nodes are available for starting new simulations while $\mu$ computing nodes are currently evaluating $f$ at a set of $\mu \geq 0$ "busy" points $\mathbb{X}_{\text{busy}} := \{x_{\text{b}}^1, \ldots, x_{\text{b}}^\mu\}$. We define the **asynchronous multi-points Expected Improvement** of $\mathbb{X}_{\text{asy}} := \{x_a^1, \ldots, x_a^\lambda\}$ as

$$EI^{(\mu,\lambda)}(\mathbb{X}_{\text{asy}}) := \mathbb{E}\left[(\min(Y(\mathbb{X}\cup\mathbb{X}_{\text{busy}})) - \min(Y(\mathbb{X}_{\text{asy}})))^+ \,|Y(\mathbb{X}) = f(\mathbb{X})\right],$$

where $[\bullet]^+ \equiv \max(0, \bullet)$. This criterion, which was initially introduced in [4], measures how much progress with respect to already calculated or currently calculating points $(\mathbb{X}\cup\mathbb{X}_{\text{busy}})$ will be made on average at $\mathbb{X}_{\text{asy}}$. The case $\mu = 0$, $\lambda = 1$ is the usual $EI$ defined in the EGO algorithm [8]. A parallel algorithm that works by maximizing $EI^{(\mu,\lambda)}$ can now be introduced:

**Asynchronous Parallel $EI$ algorithm**

1. Generate $\mathbb{X}$ through a space-filling design. Calculate $f(\mathbb{X})$.
2. While calculation budget not exhausted do
   (a) [non blocking] Retrieve new $x$'s and $f(x)$'s if any. Update the kriging model (optionally with parameter re-estimation). Update $\lambda$ and $\mu$.
   (b) Generate $\lambda$ points by $\max EI^{(\mu,\lambda)}(x_a^1, \ldots, x_a^\lambda)$ using a global optimizer (e.g. CMA-ES [5]). Send them to worker nodes for evaluation.

The classic $EI$ criterion has the desirable property that its maximum lies away from already sampled points of $\mathbb{X}$ and strikes a comprise between the exploration of unknown regions of the design space and the intensification of the search in known highly performing regions [8]. In addition, the multi-points asynchronous $EI^{(\mu,\lambda)}$ has its maximum away from any subset of already sampled and currently running points, $\mathbb{X}\cup\mathbb{X}_{\text{busy}}$ (it is null there while $I$ is a positive variable, [7]). Another advantage of $EI^{(\mu,\lambda)}$ is that it does not introduce extra parameters.

$EI^{(\mu,\lambda)}$ accounts for all ratios of optimizer over simulation computation times. If the simulations are much longer than any optimizer iteration (krig-

ing update and $EI$ maximization), points will be allocated to newly available nodes one at a time, in which case $EI^{(\mu,\lambda=1)}$ will be used. Vice versa, if the optimizer iteration takes longer than the simulations, no busy point occurs and $EI^{(\mu=0,\lambda)}$ is relevant. Intermediate cases call for general $EI^{(\mu,\lambda)}$'s.

## 3 Bounds and estimation of $EI^{(\mu,\lambda)}$

While the calculation of $EI^{(0,1)}$ and $EI^{(0,2)}$ is analytical [3], no general expression for $EI^{(\mu,\lambda)}$ is known. We propose to estimate $EI^{(\mu,\lambda)}$ through a Monte Carlo strategy augmented by bounds knowledge. In particular, the following bounds are established in [7] where $EI^*(x_b^i, x_a^j) := \mathbb{E}\left[(Y(x_b^i) - Y(x_a^j))^+ | Y(\mathbb{X}) = f(\mathbb{X})\right]$ :

$$\max_{i=1,\lambda} EI(x_a^i) \leq EI^{(0,\lambda)}(\mathbb{X}_{\text{asy}}) \leq \sum_{i=1}^{\lambda} EI(x_a^i)$$

$$0 \leq EI^{(\mu,\lambda)}(\mathbb{X}_{\text{asy}}) \leq \min\left(\sum_{i=j}^{\lambda} EI(x_a^j), \sum_{j=1}^{\lambda} EI^*(x_b^1, x_a^j), \ldots, \sum_{j=1}^{\lambda} EI^*(x_b^\mu, x_a^j)\right)$$

All the expressions in the bounds are analytical, including the $EI^*$ terms, because they are instances of the usual $EI$ formula [8].

The Monte Carlo estimator of the mean of $I^{(\mu,\lambda)}$ and its variance are calculated from $N$ samples of the conditional Gaussian process $Y$ (i.e., samples $i_j$ of the improvements) as follows:

$$EI_{MC}^{(\mu,\lambda)}(\mathbf{x}) = \frac{1}{N}\sum_{j=1}^{N} i_j^{(\mu,\lambda)}(\mathbf{x}) \ , \ \sigma_{MC}^2(\mathbf{x}) = \frac{1}{N(N-1)}\sum_{j=1}^{N}(i_j^{(\mu,\lambda)}(\mathbf{x}) - EI^{(\mu,\lambda)})^2$$

We now introduce the above bounds through Bayes law. The a priori density of $EI^{(\mu,\lambda)}$ is uniform between the lower and upper bounds, $\mathcal{U}(L,U)$. Since the likelihood of the expectation estimator, $p\left(EI_{MC}^{(\mu,\lambda)} | EI^{(\mu,\lambda)}\right)$, is Gaussian, the a posteriori density is a known truncated Gaussian as can be seen from $p\left(EI^{(\mu,\lambda)} | EI_{MC}^{(\mu,\lambda)}\right) = p\left(EI_{MC}^{(\mu,\lambda)} | EI^{(\mu,\lambda)}\right) \mathcal{U}(L,U)/\text{const}$. Calculation details are given in [7] and yield estimations of the mean and the variance of $EI^{(\mu,\lambda)}(\mathbf{x})|EI_{MC}^{(\mu,\lambda)}(\mathbf{x})$, denoted $M(\mathbf{x})$ and $V^2(\mathbf{x})$, respectively, which account for both Monte Carlo simulations and the bounds.

The asynchronous parallel EI algorithm introduced earlier has a step that maximizes $EI^{(\mu,\lambda)}(\bullet)$ with the CMA-ES algorithm ([5]). In CMA-ES, the objective function values are used to rank the explored points. We propose to modify this comparison of points (say $\mathbf{x}^i$ and $\mathbf{x}^j$) in order to control the Monte Carlo simulations:

**Pairwise ranking procedure**

1. Set confidence $k$ (e.g., over 60% confidence for $k = 1$), $N$, $\Delta N$,
2. While not stop do
   (a) If $M(\mathbf{x^i}) - kV(\mathbf{x^i}) \geq M(\mathbf{x^j}) + kV(\mathbf{x^j})$ then $EI^{(\mu,\lambda)}(\mathbf{x^i}) \geq EI^{(\mu,\lambda)}(\mathbf{x^j})$, stop. (And vice versa.)
   (b) Decrease the Monte Carlo variances by allocating extra samples: $\Delta N_i = V(\mathbf{x^i})\Delta N/(V(\mathbf{x^i}) + V(\mathbf{x^j}))$, $\Delta N_j = \Delta N - \Delta N_i$.

## 4 Test results

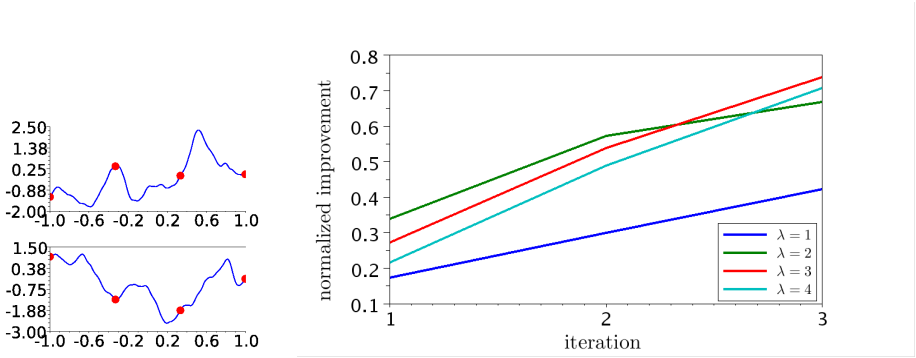


Fig. 1: Left: two of the ten sampled trajectories with Matern 5/2 kernel and scale $\theta = 0.15$. Right: mean normalized improvement as a function of iteration for $\lambda = 1$ to 4 ($\mu = 0$), averaged over 10 test functions (defined as 1D trajectories such as those on the left).

The tests reported in [7] are now summarized.

Firstly, 100 functions in 1D have been generated by sampling Gaussian process trajectories (see the 2 examples of Fig.1). For each function, a pair of points was randomly chosen and their $EI^{(\mu,\lambda)}$ compared, $\mu = 0, 1, 3$ and $\lambda = 1$ to 5. It is observed that, on average, accounting for the bounds divides by 7 the total number of Monte Carlo simulations necessary to discriminate the points. However, a first difficulty appeared. When $\mu > 0$, 25% of the pairs needed over $N = 100,000$ MC samples for the comparison, i.e., their points had very close $EI^{(\mu,\lambda)}$ values: the $EI^{(\mu,\lambda)}$ function has plateaus. This proportion increased when the pairs were generated by optimization because these plateaus correspond to high performance regions of the design space.

The second issue stems directly from $EI^{(\mu,\lambda)}$'s definition: its input is high dimensional ($\dim \mathbb{X}_{\text{asy}} = \lambda\, d$), making its maximization potentially costly.

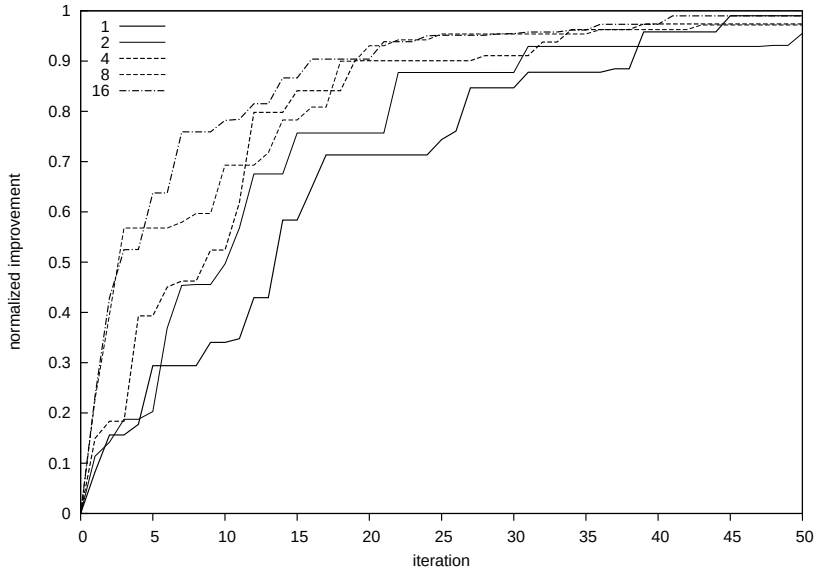


Fig. 2: Mean normalized improvement as a function of iteration for $\lambda = 1$ to 16 ($\mu = 0$), 9 dimensional test case.

Thirdly, on the positive side, the experiments summarized in Fig.1 and 2 show that logarithmic speed-ups are obtained: we have observed

$$\frac{\text{time to solve}, \lambda = 1}{\text{time to solve}, \lambda} \approx 1 + b \log(\lambda)$$

where $b = 0.85$ and 0.49 in 1 and 9 dimensions, respectively. The 9D test function is the approximation of a matrix by its rank 1 counter part:

$$(\mathbf{B}^*_{mr}, \mathbf{C}^*_{rn}) = \arg \min_{\mathbf{B}_{mr}, \mathbf{C}_{rn}} \|\mathbf{A}_{mn} - \mathbf{B}_{mr}\mathbf{C}_{rn}\|, \quad r < \text{rank}(\mathbf{A}_{mn}), \tag{1}$$

where $\|\cdot\|$ is the Frobenius norm, $\mathbf{A}$ is a $4 \times 5$ matrix of uniformly distributed elements in $[0, 1]$, $\mathbf{B}$ and $\mathbf{C}$ are a column and a row vector whose elements are constrained to be in $[-1, 1]$, respectively. This is a continuous nonconvex 9-dimensional box-constrained optimization problem whose solution is given by the first singular vectors. An instance of a typical run is shown for $\lambda = 1$ to 16, $\mu = 0$, in Fig. 2.

The observed speed-up is due to the kriging model which summarizes all of the information gathered by the worker nodes, including the currently running simulations. Further work addressing the aforementioned estimation and optimization difficulties of $EI^{(\mu,\lambda)}$ is needed.

## References


1. Berbecea, A.C., Kreuawan, S., Gillon, F. and Brochet, P.: A Parallel Multiobjective Efficient Global Optimization: The Finite Element Method in Optimal Design and Model Development. IEEE Transactions on Magnetics, 46(8), pp. 2868-2871, 2010.
2. Branke, J., Kamper, A., Schmeck, H.: Distribution of evolutionary algorithms in heterogeneous networks. In Deb K. et al., eds, GECCO 2004, LNCS 3102, Springer, pp. 923-934, 2004.
3. Ginsbourger, D., Le Riche, R., Carraro, L.: Kriging is well-suited to parallelize optimization. In Tenne Y. and Goh C.-K., eds., Computational Intelligence in Expensive Optimization Problems, Springer series in Evolutionary Learning and Optimization, pp. 131-162, 2009.
4. Ginsbourger, D., Janusevskis, J., Le Riche, R.: Dealing with asynchronicity in parallel Gaussian Process based global optimization. HAL technical report no. hal-00507632, `http://hal.archives-ouvertes.fr/hal-00507632/`, July 2010
5. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 9(2), pp. 159-195, 2001.
6. Kolda, T. G. : Revisiting asynchronous parallel pattern search for nonlinear optimization. SIAM J. Optimization, 16(2), pp. 563-586, 2005.
7. Janusevskis, J., Le Riche R., Ginsbourger, D.: Parallel expected improvements for global optimization: summary, bounds and speed-up. HAL technical report no. hal-00613971, `http://hal.archives-ouvertes.fr/hal-00613971_v1`, August 2011
8. Jones, D.R., Schonlau, M. and Welch, W. J.: Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4), pp. 455-492, 1998.
9. Regis, R. G., Shoemaker, C. A.: Parallel radial basis function methods for the global optimization of expensive functions. European J. of Operational Research, 182, pp. 514-535, 2007.
10. Sobester, A., Leary S.J. and Keane, A.J.: A parallel updating scheme for approximating and optimizing high fidelity computer simulations. J. of Structural and Multidisciplinary Optimization, 27, pp. 371-383, 2004.