

Ensemble of metamodels: extensions of the least squares approach to efficient global optimization

Wallace G. Ferreira¹ · Alberto L. Serpa²

Received: 27 September 2015 / Revised: 1 May 2017 / Accepted: 12 June 2017 / Published online: 5 July 2017
© Springer-Verlag GmbH Germany 2017

Abstract In this work we present LSEGO, an approach to drive efficient global optimization (EGO), based on LS (least squares) ensemble of metamodels. By means of LS ensemble of metamodels it is possible to estimate the uncertainty of the prediction with any kind of model (not only kriging) and provide an estimate for the expected improvement function. For the problems studied, the proposed LSEGO algorithm has shown to be able to find the global optimum with less number of optimization cycles than required by the classical EGO approach. As more infill points are added per cycle, the faster is the convergence to the global optimum (exploitation) and also the quality improvement of the metamodel in the design space (exploration), specially as the number of variables increases, when the standard single point EGO can be quite slow to reach the optimum. LSEGO has shown to be a feasible option to drive EGO with ensemble of metamodels as well as for constrained problems, and it is not restricted to kriging and to a single infill point per optimization cycle.

Keywords Ensemble of metamodels · Surrogate based optimization · Efficient global optimization · Least squares

✉ Wallace G. Ferreira
wferrei7@ford.com; wgferreira@yahoo.com

Alberto L. Serpa
serpa@fem.unicamp.br

¹ CAE & Optimization Engineering, Ford Motor Company
Brazil, Av. Taboão, 899, 09655-900 São Bernardo do Campo,
SP, Brazil

² School of Mechanical Engineering - FEM, Department
of Computational Mechanics - DMC, University of Campinas
- UNICAMP, 13083-970 Campinas, SP, Brazil

1 Introduction

In the last decades, the use of metamodeling methods (also known as *surrogate modeling* or *response surface methodology*) to replace expensive computer simulation models such as FE (Finite Elements) or CFD (Computational Fluid Dynamics) found in automotive, aerospace and oil-gas industries, for example, has become a common place in both research and practice in engineering design, analysis and optimization. A collection of engineering research and applications in this field has been recently published in Koziel and Leifesson (2013).

In this context, surrogate based (or response surface based, or metamodel based) optimization refers to the process of using fast running metamodels as surrogates of original complex and long time running computer simulation models to approximate the objectives and constraint functions in a standard optimization algorithm. This methodology has shown to be effective in both multidisciplinary and multiobjective optimization problems and it has been widely applied in research and industry. Refer to the reviews by Queipo and et al (2005), Simpson et al. (2008) and Forrester and Keane (2009) for a broad and detailed discussion on this subject.

The surrogate based optimization is in general an iterative (cyclic) process. At each cycle, instances of simulation models with different parameters are evaluated (sampling points from a Design of Experiments, DOE). The surrogate models are fit based on these sampling data and the resulting approximate functions are used in the search of optimum points (exploitation), analysis of the response behavior, sensitivity and trends in the design space (exploration). Once optimum design candidates and other extrema points are found, they are evaluated with the true simulation models and if necessary the new points are included in the

sampling space in order to improve the approximation and to restart the iterative process. See in Jones (2001) a review on different approaches (also known as sequential sampling approaches) used to drive the global surrogate based optimization.

Efficient global optimization (EGO) is an iterative surrogate based approach where, at each optimization cycle, one *infill* point is selected as the one that maximizes the *expected improvement* with respect to the optimum of the objective function. The EGO concept emerged after the work of Jones et al. (1998), that was based mainly on previous research on Bayesian Global Optimization (Schonlau 1997; Mockus 1994). Traditionally EGO algorithms are based on kriging approximation and a single infill point is generated per optimization cycle.

A question that arises is that the selection of only one infill point per optimization cycle can be quite slow to achieve the convergence to the optimum. If parallel computation is an available resource, then multiple infill points should be defined and therefore less cycles might be required for convergence. This aspect can be crucial specially if the computer models take several hours to run¹ and a single point EGO approach becomes prohibitive, specially in multidisciplinary optimization scenarios. In practical terms, when parallel resources are not an issue, it can be worthwhile to run more simulations per cycle in order to reach an optimum in a reasonable lead time, even if the total number of simulations is higher at the end of the optimization process.

The aspect of single versus multiple infill points per cycle is known and has been discussed since the origins of EGO-type algorithms at later 1990s. Schonlau (1997) proposed extensions to the standard EGO algorithm to deliver “*m* points per stage”, but he pointed out numerical difficulties in the evaluation of the derived expressions accurately at a reasonable computational cost. In addition, his results were limited to a two-variable function and indicate 18.9% penalty for the parallel approach in terms of function evaluations for the same accuracy.

Besides this apparent disadvantage in terms of function evaluations, it can be observed an increasing research interest on EGO approaches with multiple infill points per cycle (and other metamodel based parallelization strategies as well) published in the last ten years. This is because

parallel computation is nowadays a relatively easy resource and the potential penalty of parallel approaches in terms of function evaluations should be neglected in favor of quickly delivering optimization results.

Although it can be considered a relatively new research field surrogate based global optimization is gaining popularity as pointed out in Haftka et al. (2016). In this recent and broad survey they examined and discussed the publication focused on parallel surrogate-assisted global optimization with expensive models. According to the authors this area is not mature yet and it is not possible to conclude with respect to the comparative efficiency of different approaches or algorithms without further research.

As discussed by Haftka et al. (2016) different classes of algorithms or strategies can be defined with the objective of balancing exploitation and exploration of the design space during the optimization. In simple terms exploitation refers to deep diving (or zooming) in the candidate areas of feasible optimum points in order to improve the objective function and the constraints to deliver better optimization results. On the other hand, exploration means adding infill points in different areas of the design space in order to reduce the uncertainty and to improve the metamodel prediction capability.

The different classes of strategies involve the ones based on nature inspired algorithms like genetic, evolutionary, particle swarm, etc., that are naturally parallelized (the populations can be divided in different regions, or processors) and are commonly applied in global optimization. In addition surrogate based strategies (like EGO) can be used together with the parallelization to improve the exploitation and exploration features of the algorithms.

In this sense, specifically in the branch of metamodel-based multiple points per cycle algorithms (like EGO or other approaches with similar objectives), refer for instance to Sóbester et al. (2004), Henkenjohann and Kukert (2007), Ponweiser et al. (2008), Ginsbourger et al. (2010), Viana and Haftka (2010), Janusevskis et al. (2012), Viana et al. (2013), Desautels et al. (2012), Chaudhuri and Haftka (2012), Rehman et al. (2014), Mehari et al. (2015), Encisoa and Branke (2015) and other interesting and relevant works referenced and discussed in Haftka et al. (2016).

In our previous research work, Ferreira and Serpa (2016), we presented and discussed the concept of least squares (LS) regression, in order to find the optimal weights in ensemble of metamodels (or weighted average surrogates, WAS). We proposed the *augmented least squares ensemble of metamodels* (LS-a), a variation of the standard least squares regression to create ensemble of metamodels. In this way, the ensemble of metamodels constructed based on LS approach inherits the variance estimator, which can be used in the definition of the expected improvement function to be applied in EGO-type algorithms.

¹For instance, even with high end computers clusters used nowadays in automotive industry, one single full vehicle analysis of high fidelity safety crash FEM model takes up to 15 processing hours with 48 CPU in parallel. With respect to CFD analysis one single complete car aerodynamics model for drag calculation, by using 96 CPU, should take up 30 hours to finish. An interesting essay regarding this “never-ending” need of computer resources in structural optimization can be found in Venkataramanan and Haftka (2004).

Viana et al. (2013) proposed the MSEGO (multiple surrogates EGO). They devised a way export the uncertainty estimate for one kriging model to the other non-kriging models in a multiple surrogates set. With different uncertainty estimates, they generate different instances of expected improvement functions to be maximized and to provide multiple parallel infill points in each EGO cycle.

We suggested an alternative scheme to generate multiple infill points per cycle in the EGO algorithm by using a LS ensemble, in order to take advantage of multiple surrogates, as applied in Viana et al. (2013). In the present work we extend the results of Ferreira and Serpa (2016) by proposing an approach that is able to provide multiple points per cycle in a EGO-type algorithm by using least squares ensemble of metamodels.

We use the acronym LSEGO, that stands for “least squares ensemble efficient global optimization”. In the numerical experiments performed, LSEGO has shown to be a feasible alternative to drive EGO algorithms with ensemble of metamodels, and in addition it is not restricted to kriging nor a single infill point per optimization cycle. LSEGO was applied in the optimization of analytical benchmark functions (up to six variables) and including some examples of constrained optimization as well. In addition LSEGO produced competitive results as compared to MSEGO for the optimization of two benchmark functions.

The remainder of the present text will be divided as follows. In Section 2 we present the theoretical fundamentals of efficient global optimization and the standard EGO algorithm. In Section 3 we present and discuss our proposed LSEGO approach. In Section 4 we present the numerical experiments performed to validate and compare the LSEGO approach with the traditional EGO algorithm and the respective results and discussion are presented in Section 5. Finally, the concluding remarks are presented in Section 6.

2 Theoretical background

2.1 Metamodel based optimization

Let $\mathbf{x} = [x_1 \dots x_{n_v}]^T \in \mathfrak{N}^{n_v}$ be a vector of n_v parameters or design variables. Metamodels are nothing but methods that attempt to fit a function $\hat{y} = \hat{f}(\mathbf{x}) : \mathfrak{N}^{n_v} \rightarrow \mathfrak{R}$ to a set of known N data points $\chi : \{\mathbf{x}_i, y_i\}$, determined by a sampling plan (design of experiments, DOE).

In most of the models, the approximate function $\hat{y}(\mathbf{x}) \approx y(\mathbf{x})$ (response surface, surrogate or metamodel) can be given in the general form

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\psi}(\mathbf{x}), \tag{1}$$

where \mathbf{w} represents *weights* to be determined and $\boldsymbol{\psi}(\mathbf{x})$ are the associated *basis functions*.

Further details on formulation and different types of metamodels (e.g. polynomial response surfaces (PRS), radial basis functions (RBF), neural networks (NN), support vector regression (SVF), etc.) can be found in Forrester et al. (2008), Fang et al. (2006) and in the references therein. In the Appendix A it is presented the basic formulation for kriging metamodel (KRG), that will be referenced throughout this work.

The optimization process based on metamodels (i.e., metamodel based, response surface based or surrogate based optimization) can be stated as a standard optimization problem, by replacing the true objective function $y = f(\mathbf{x})$ and the n_c constraints $g_i(\mathbf{x})$, that are difficult and costly to compute, by their respective fast and cheap surrogates, i.e.,

$$\mathbf{x}_{opt} = \begin{cases} \min_{\mathbf{x}} \hat{y}(\mathbf{x}) \\ \text{s.t.}, \hat{g}_i(\mathbf{x}) - g_{max}^i \leq 0, i = 1 \dots n_c, \\ \mathbf{x}_{lb} \leq \mathbf{x} \leq \mathbf{x}_{ub} \end{cases}, \tag{2}$$

where g_{max}^i is the reference value (target) for i -th constraint² and \mathbf{x}_{lb} and \mathbf{x}_{ub} the bounds on the design space. The optimum point \mathbf{x}_{opt} can be found by any available optimization algorithm: gradient based, direct search, genetic algorithm, etc. Refer to Queipo and et al (2005) or Forrester and Keane (2009) for details.

2.2 Efficient global optimization (EGO)

As pointed out by Forrester and Keane (2009): “the Holy Grail of global optimization is finding the correct balance between exploitation and exploration”. Algorithms that favor exploitation (search of the optimum), can be quite slow to converge and in addition be trapped at a local minimum point and not be able to reach the global optimum. On the other hand, pure exploration (improvement of the surrogate and search in the whole design space) may lead to a waste of resources (function evaluations, simulations). In this sense, efficient global optimization (EGO) algorithms emerge as feasible tools to balance exploitation and exploration of the design space.

Since the data set χ is arbitrary, the determination of $\hat{y}(\mathbf{x})$ can be stated as a realization of an stochastic process. In this sense, the approximation can be modeled as Gaussian, i.e., normally distributed random variable $\hat{Y}(\mathbf{x})$, with mean $\hat{y}(\mathbf{x})$ and variance $\hat{s}^2(\mathbf{x})$.

Efficient global optimization algorithms are centered in the concept of maximum expected improvement. Let

²Only for notational convenience, without loss of generality, we will assume that all equality constraints $h(\mathbf{x})$ can be properly transformed into inequality constraints $g(\mathbf{x})$.

$y_{min} = \min(y_1, \dots, y_n)$ be the best current value for the function $y(\mathbf{x})$ in the sampling data set. Then, as described in Forrester et al. (2008), the probability of an improvement $I(\mathbf{x}) = y_{min} - Y(\mathbf{x})$ of a point \mathbf{x} with respect to y_{min} , can be calculated as

$$P[I(\mathbf{x})] = \frac{1}{\hat{s}\sqrt{2\pi}} \int_{-\infty}^0 \exp\left(-\frac{[I - \hat{y}]^2}{2\hat{s}^2}\right) dI, \quad (3)$$

by abbreviating the dependence of I , \hat{s} and \hat{y} on \mathbf{x} .

On the other hand, the amount of improvement expected can be obtained by taking the expectation $E[\cdot]$ of $\max(y_{min} - Y(\mathbf{x}), 0)$, which leads to

$$E[I(\mathbf{x})] = I(\mathbf{x})\Phi\left(\frac{I(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x})\phi\left(\frac{I(\mathbf{x})}{\hat{s}(\mathbf{x})}\right), \quad (4)$$

for $\hat{s}(\mathbf{x}) > 0$, and $E[I(\mathbf{x})] = 0$, otherwise. In this equation, $I(\mathbf{x}) \equiv (y_{min} - \hat{y}(\mathbf{x}))$, where \hat{y} is a realization of Y , and $\Phi(\cdot)$ and $\phi(\cdot)$ are respectively the normal cumulative distribution and normal probability density functions.

Equation (4) is known as the *expected improvement function* of any point \mathbf{x} in the design space, with respect to the current best value y_{min} . For details on the derivation see Forrester and Keane (2009) or the original publications of Schonlau (1997) and Jones et al. (1998).

By means of (4), the EGO algorithm can be defined for any metamodel $\hat{y}(\mathbf{x})$, that provides an uncertainty estimate $\hat{s}(\mathbf{x})$. EGO was originally devised and is traditionally applied with KRG (see the basic formulation in Appendix A), but other models like PRS, RBF and other Gaussian models can be used as well. See for example (Sóbester et al. 2004) in which RBF was applied.

2.3 The standard EGO algorithm

The standard EGO algorithm can be summarized in the following steps:

- i. Define a set χ of N sampling points and start the optimization cycles ($j \leftarrow 1$);
- ii. Evaluate the true response $y(\mathbf{x})$ at all data sites in χ , at the current cycle j , and set

$$y_{min} \leftarrow \min(y_1, \dots, y_N);$$

- iii. Generate the metamodel $\hat{y}(\mathbf{x})$ and estimate $E[I(\mathbf{x})]$ with all the data points available in the sampling space χ , at the current cycle j ;
- iv. Find the next infill point \mathbf{x}_{N+1} as the maximizer of $E[I(\mathbf{x})]$, $\chi_{infill} \leftarrow \mathbf{x}_{N+1} = \max E[I(\mathbf{x})]$; evaluate the true function $y(\mathbf{x})$ at \mathbf{x}_{N+1} and add this new point to the sampling space: $\chi \leftarrow \chi \cup \chi_{infill}$;
- v. If the stopping criteria is not met, set

$$y_{min} \leftarrow \min(y_1, \dots, y_{N+1}),$$

set ($N \leftarrow N + 1$), update cycle counter ($j \leftarrow j + 1$), and return to step iii. Otherwise, finish the EGO algorithm.

2.4 Extensions to the EGO algorithm

The EGO algorithm can be extended to handle constraints in the optimization, by using the concept of probability of improvement. The basis for this extension can be found in Schonlau (1997) and with details and applications in Forrester et al. (2008) and Han and Zhang (2012).

By following the derivation presented in Han and Zhang (2012), the idea is to find the probability of satisfying the constraints $g_i(\mathbf{x})$. In other words, when $P[G_i(\mathbf{x}) \leq 0] \rightarrow 1$, the constraint is satisfied; otherwise, when $P[G_i(\mathbf{x}) \leq 0] \rightarrow 0$, the constraint is violated. Analogously to (3), $P[G_i(\mathbf{x}) \leq 0]$ can be calculated as

$$P[G_i(\mathbf{x}) \leq 0] = \frac{1}{\hat{s}_i\sqrt{2\pi}} \int_{-\infty}^0 \exp\left(-\frac{[G_i - \hat{g}_i]^2}{2\hat{s}_i^2}\right) dG_i \quad (5)$$

where $G_i(\mathbf{x})$ is the random variable related to $\hat{g}_i(\mathbf{x})$ and $\hat{s}_i(\mathbf{x})$ the respective constraint uncertainty estimate.

In this way, the step iv. of the EGO standard algorithm presented at Section 2.3 can be modified as follows to accommodate n_c independent and uncorrelated constraints, i.e.,

$$\mathbf{x}_{N+j} = \begin{cases} \max_{\mathbf{x}} E[I(\mathbf{x})] \times \prod_{i=1}^{n_c} P[G_i(\mathbf{x}) \leq 0] \\ \text{s.t., } \mathbf{x}_{lb} \leq \mathbf{x} \leq \mathbf{x}_{ub} \end{cases} \quad (6)$$

The treatment of multiobjective optimization problems can be also extended by using the concept of probability and expected improvement. It requires a more elaborated derivation that is out of the scope of this text and the details can be found in Forrester et al. (2008). In addition, Jurecka (2007) has successfully extended and applied the EGO concept to treat robust optimization problems as well.

These and other possible extensions of EGO-type algorithms are of interest for research and practical applications and we intend to explore this field in our future work.

3 LS ensemble of metamodels EGO (LSEGO)

3.1 Definitions

As we presented in Ferreira and Serpa (2016), the linear ensemble of metamodels can be written as

$$\hat{y}_{ens} = \hat{\mathbf{Y}}\mathbf{w} \quad (7)$$

where $\mathbf{y} = [y_1 \cdots y_N]^T$, $\hat{\mathbf{Y}} = [\hat{y}_1(\mathbf{x}_1) \cdots \hat{y}_M(\mathbf{x}_i)]$ and $\mathbf{w} = [w_1 \cdots w_M]^T$, for N sampling points and M metamodels.

If the optimal weights \mathbf{w} are estimated by least squares methods (LS), as we discussed in detail in Ferreira and Serpa (2016), then the resulting ensemble of metamodels inherits the *least squares variance estimate* $V[\hat{y}_{ens}(\mathbf{x})] \equiv \hat{s}^2(\mathbf{x})$, for the prediction at each point \mathbf{x} , that can be written as

$$\hat{s}^2(\mathbf{x}) = \hat{\sigma}^2 [\hat{\mathbf{y}}(\mathbf{x})]^T \left(\hat{\mathbf{Y}}^T \hat{\mathbf{Y}} \right)^{-1} \hat{\mathbf{y}}(\mathbf{x}), \tag{8}$$

with $\hat{\mathbf{y}}(\mathbf{x}) = [\hat{y}_1(\mathbf{x}) \hat{y}_2(\mathbf{x}) \cdots \hat{y}_M(\mathbf{x})]^T$, and

$$\hat{\sigma}^2 = \frac{\hat{\mathbf{y}}_{ens}^T \hat{\mathbf{y}}_{ens} - \hat{\mathbf{w}}^T \hat{\mathbf{Y}} \hat{\mathbf{y}}_{ens}}{N - n_v} \tag{9}$$

where $\hat{\mathbf{y}}_{ens} = [\hat{y}_{ens}(\mathbf{x}_1) \hat{y}_{ens}(\mathbf{x}_2) \cdots \hat{y}_{ens}(\mathbf{x}_N)]^T$.

Therefore, if $\hat{s}^2(\mathbf{x})$ is a good estimate for the uncertainty of the LS ensemble of metamodel at the design space, then it can be used to derive the expected improvement function (4) in order to drive EGO algorithms, by using any kind of metamodels $\hat{y}_i(\mathbf{x})$ and not only kriging.

In this sense, we named this proposed approach as LSEGO (least squares ensemble efficient global optimization) and the intention in the present work is to verify and demonstrate the efficiency of this heuristic method as an alternative to drive EGO-type algorithms.

In fact, we cannot verify or prove *a priori* that $\hat{s}^2(\mathbf{x})$ is good or not for our purposes in the EGO context. Since it is a well-established and accepted estimate for general least squares regression, it will be applied in the present work without any proof. The main assumption is that if the LS ensemble of metamodel has reasonable prediction accuracy, therefore the variance prediction $\hat{s}^2(\mathbf{x})$ should be reasonable as well. Our numerical experiments showed that $\hat{s}^2(\mathbf{x})$ works well for generating the EI function and the optimization is convergent in several problems investigated. These results and conclusions will be presented and discussed in the next sections.

On the other hand, as mentioned by Viana et al. (2013), although there can exist several measures of quality for the variance estimate, it is possible to infer the behavior by using the coefficient of linear correlation $\rho(|e((x))|, \hat{s}((x)))$, where $e((x)) = y((x)) - \hat{y}((x))$ is the actual prediction error between the exact function and the approximation, in this case a LS-a ensemble. We will use the same approach here by means of a simple illustration presented in Fig. 1.

In this case, for a coarse sampling plan the quality of fit for LS-a is very poor as can be observed in Fig. 1. On the other hand, the expected improvement function is able to suggest infill points ($\max E[I((x))]$) where the vari-

ance is high for the approximation. The correlation in this case is very good, i.e., $\rho(|e((x))|, \hat{s}((x))) = 0.85$ with several points close to the 45 degrees line, and it is possible to observe that $\hat{s}((x))$ plays a good job estimating the behavior of the actual prediction error $e((x))$ through the design space.

As we observed in preliminary numerical experiments, as the optimization cycles evolve the quality of fit for the metamodel is naturally improved by the infill points added. As consequence the quality of the estimation of $\hat{s}((x))$ also increases and the convergence for LSEGO process is favored. In the next sections we will present and discuss the behavior of LSEGO for different benchmark problems.

3.2 Illustrations: one infill point per cycle

Figure 2 illustrates the evolution of LSEGO for a one variable function with one infill point per optimization cycle. At each optimization cycle is presented the true function $y((x))$ versus the approximation by LS-a ensemble (left plot) and the expected improvement function $E[I((x))]$ (right plot), calculated by means of (4), with $\hat{s}^2(\mathbf{x})$ as defined in (8).

At cycle 01 we have a very poor approximation with correlation coefficient $R^2 = 0.258$ and normalized root mean squared error $NRMSE = 31.9\%$. The $E[I((x))]$ presents a clearly defined peak close to the true minimum ($x_{opt}^{exact} = 5.624$). Note for this example in Fig. 2, at the first six optimization cycles, the maximum of expected improvement function works in the “exploitation mode” and prioritizes to add infill points around the optimum.

At this stage (cycle 06) the optimum found $x_{opt} = 5.600$ is quite close to the exact value (0.43% error), with a very good quality approximation for the metamodel: $R^2 = 0.975$ and $NRMSE = 4.2\%$. After cycle 06, the LSEGO algorithm automatically switches to the “exploration mode” and the infill points are selected in order to improve the quality of the approximation in the whole design space, instead of improving the minimum value found. The LSEGO algorithm was stopped at cycle 12 with $x_{opt} = 5.600$, $R^2 = 0.999$ and $NRMSE = 0.5\%$.

This behavior of LSEGO in one dimension was observed for other functions as well, with different levels of nonlinearity and multimodality. Based on these preliminary results we can conclude that LSEGO performed quite well in terms of exploitation and exploration of the design space.

On the other hand, for higher dimensional problems we noted a very slow convergence for the algorithm, associated to a high concentration of infill points around the global optimum, as observed in standard EGO algorithm as well. See for instance in Figs. 3 and 4 the behavior for LSEGO for Giunta-Watson function (see Appendix B, (20)) with two variables and one infill point per optimization cycle. The exact minimum is accurately found only at cycle 37 and all

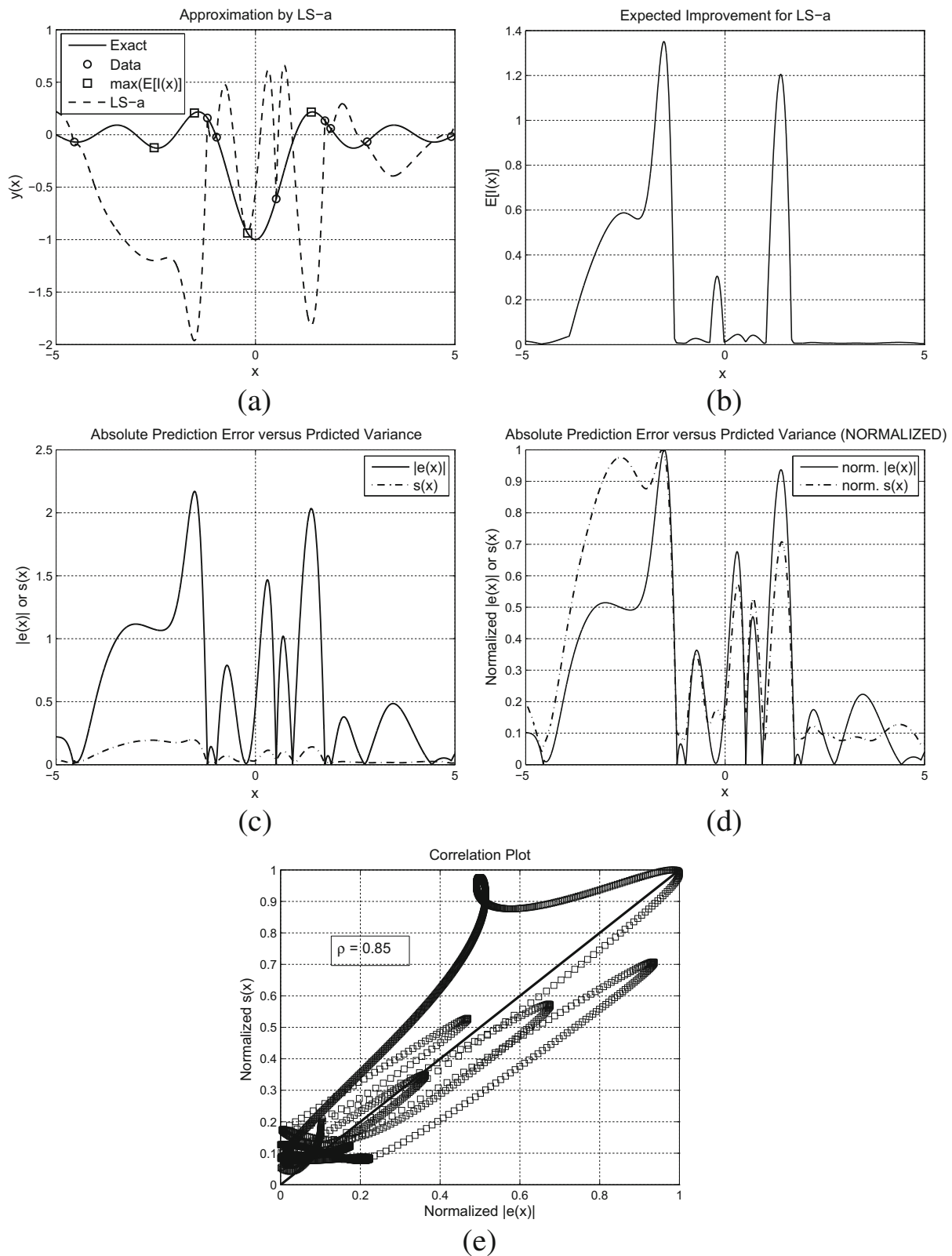


Fig. 1 Correlation between $|e(x)|$ and $\hat{s}(x)$ for the sinc function $y(x) = \frac{\sin(x)}{x}$. In (a), for a very coarse sampling plan the quality of fit for LS-a is quite poor in the beginning of the LSEGO optimization process. The expected improvement function $E[I((x))]$ in (b) is able to suggest reasonable infill points ($\max E[I((x))]$), labeled as *square*

markers in (a). Observe in (e) that $\hat{s}(x)$ is playing a good job estimating $|e(x)|$ in the design space, with correlation $\rho(|e(x)|, \hat{s}(x)) = 0.85$ and several points close to the 45 degrees line. Note in (c) and (d) the similarity of $|e(x)|$ and $\hat{s}(x)$. In (d) $|e(x)|$ and $\hat{s}(x)$ were normalized to remove the scale effects for a better visualization

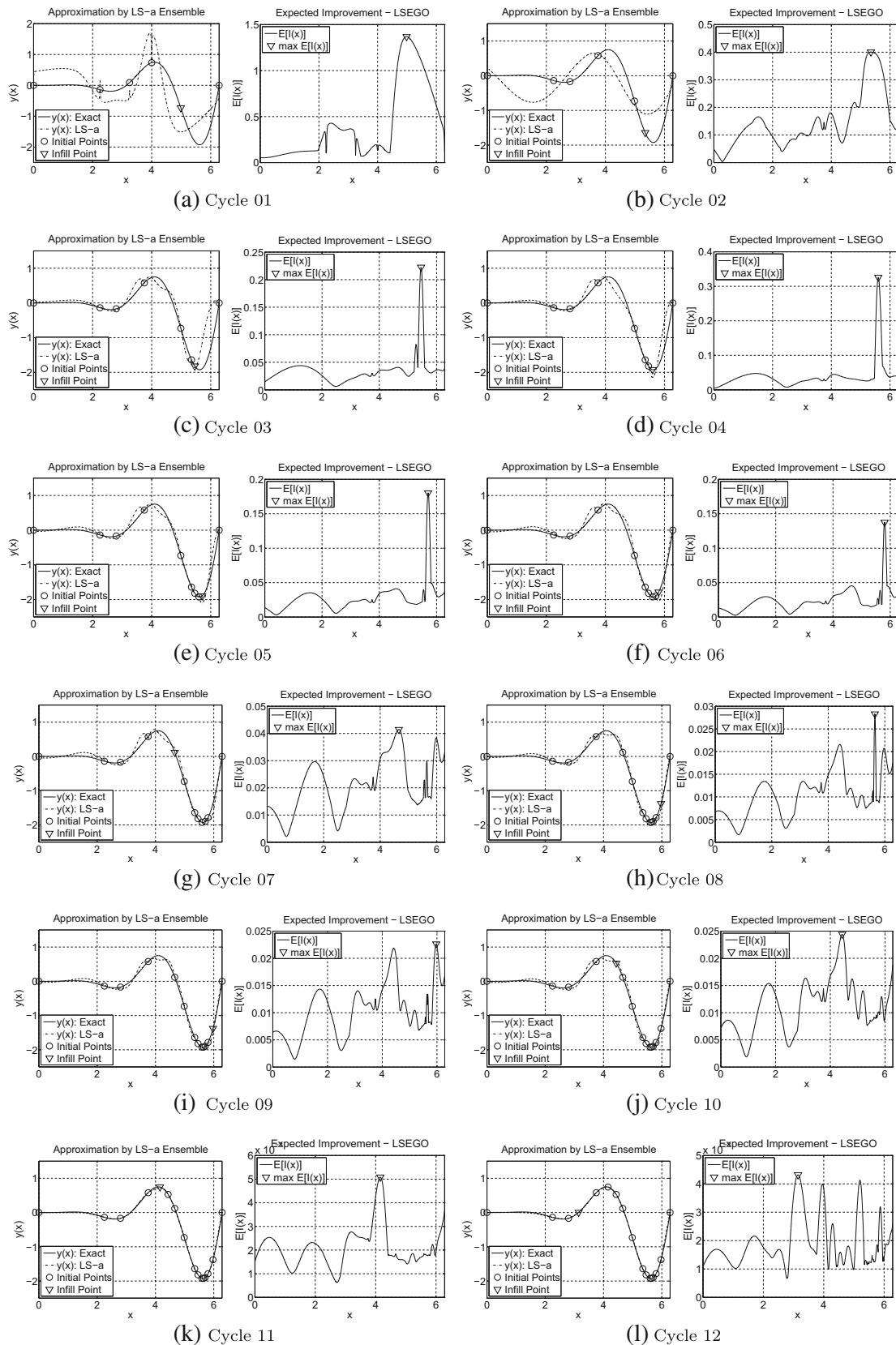
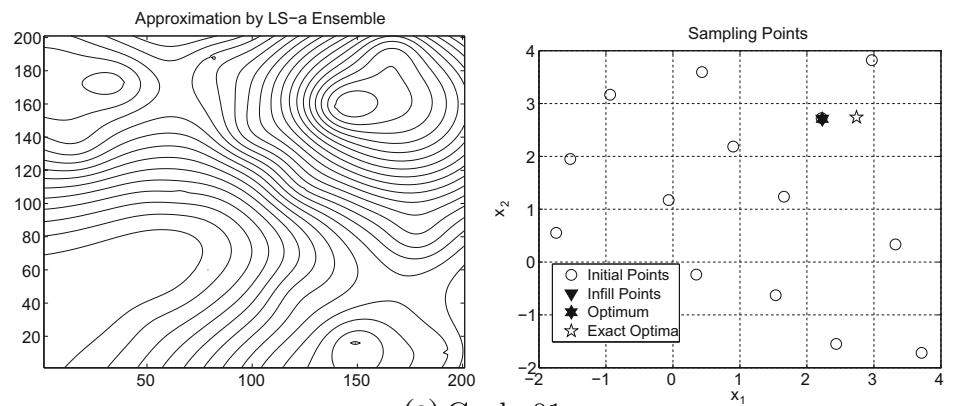


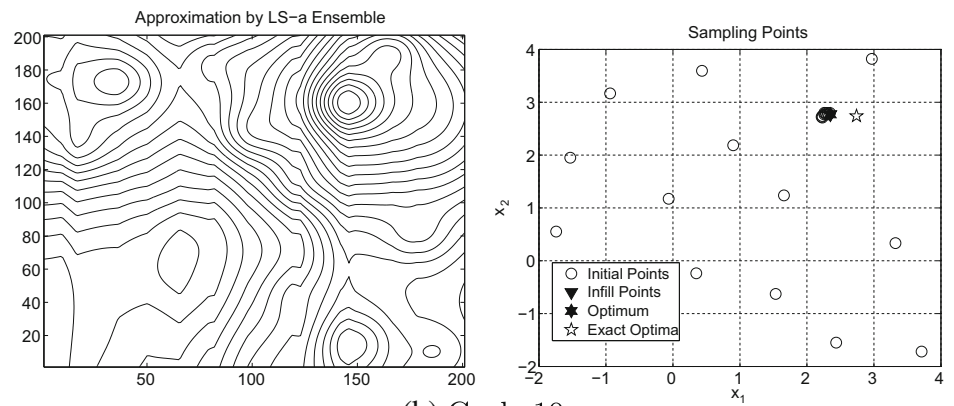
Fig. 2 Evolution of approximation $y(x)$ vs. $\hat{y}(x)$ and expected improvement $E[I(x)]$ for $y(x) = \frac{1}{700}(-2x + 5x^2 + 7x^3) \sin(2x)$ with $N_p = 1$ infill point per optimization cycle with the LSEGO

algorithm. The initial sampling points are at $\chi = [0, 2.25, 2.8, 3.75, 2\pi]$ and the LS-a ensemble is used with four metamodels: PRS ($ID = 1$), KRG ($ID = 2$), RBNN ($ID = 3$) and SVR ($ID = 4$), see Table 1

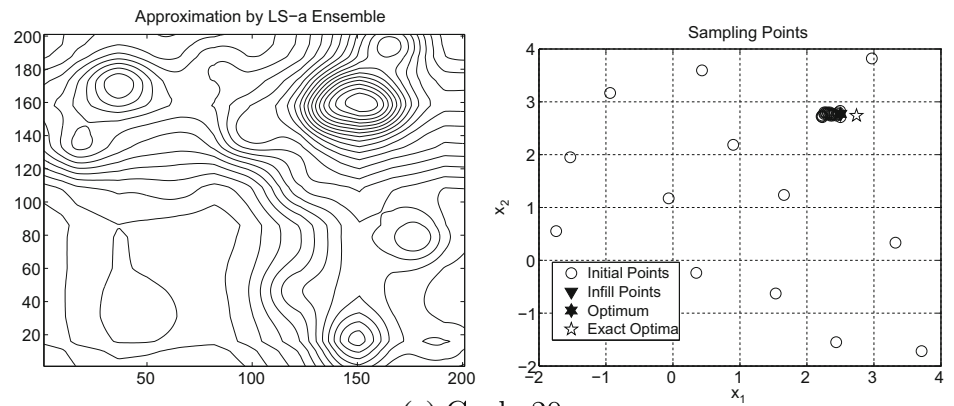
Fig. 3 Evolution of approximation of Giunta-Watson function (2 variables), with LSEGO and $N_p = 1$ infill point per optimization cycle. The 15 initial sampling points were generated with Latin Hypercube Matlab function `lhsdesign`, and the LS-a ensemble is used with four metamodels: PRS ($ID = 1$), KRG ($ID = 2$), RBNN ($ID = 3$) and SVR ($ID = 4$), see Table 1



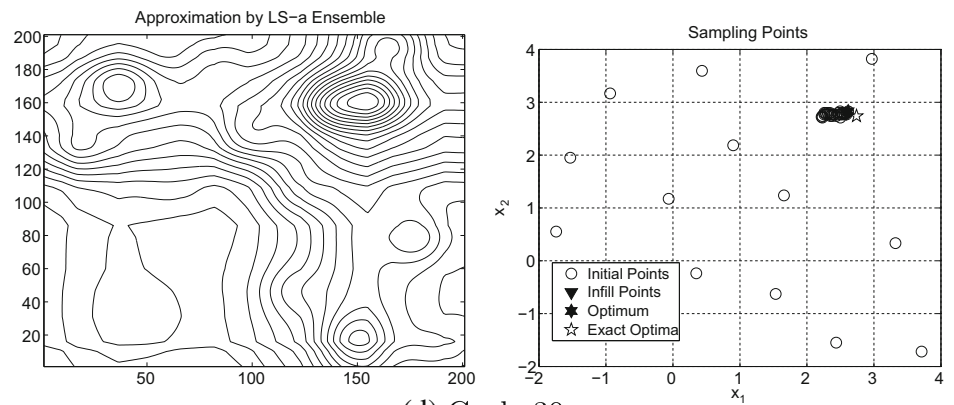
(a) Cycle 01



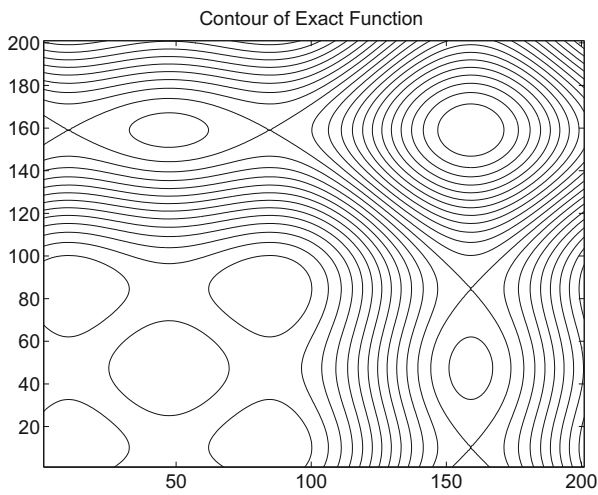
(b) Cycle 10



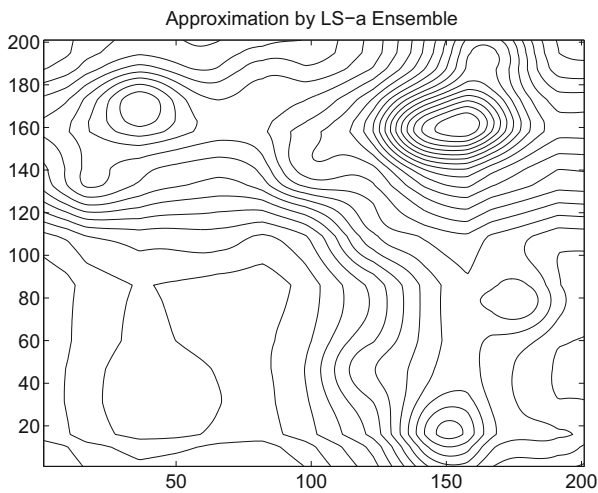
(c) Cycle 20



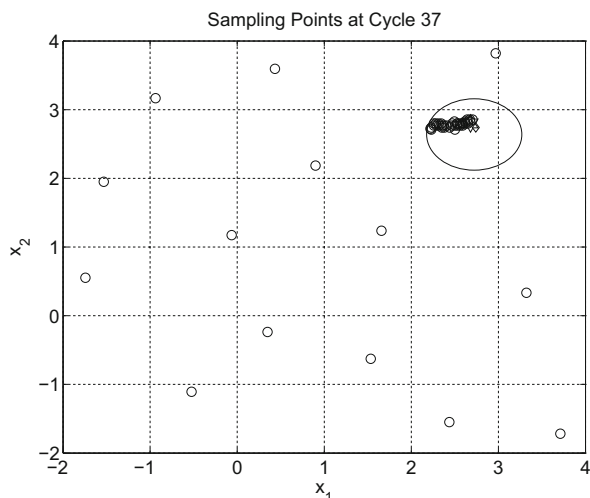
(d) Cycle 30



(a) Exact Contour



(b) Cycle 37: Approximation



(c) Cycle 37: Sampling Points

Fig. 4 Detail at Cycle 37 for the evolution of approximation of Giunta-Watson function (2 variables), with LSEGO and $N_p = 1$ infill point per optimization cycle

the infill points are located at this neighborhood. As a consequence, the quality of the approximation at cycle 37 is still poor outside the optimum vicinity (note the difference on the exact and approximate contours for the function).

It is well known that the expected improvement function can be extremely multimodal (i.e., with multiple peaks that lead to several locations with high probable improvement). This behavior was observed in most of the optimization cycles investigated for LSEGO as well (see for instance the $E[I((x))]$ curves in Fig. 2).

This multimodal behavior of the expected improvement function can be considered an advantage by selecting more than one infill point per optimization cycle in order to accelerate the whole optimization process, as discussed in Sóbester et al. (2004), for example.

3.3 LSEGO with parallel infill points

As we discussed in the Introduction (Section 1) the aspect of single versus multiple infill points per cycle is known and has been discussed since the origins of EGO-type algorithms (Schonlau 1997; Jones et al. 1998).

As presented in the review by Haftka et al. (2016), different classes of algorithms or strategies can be defined with the objective of balancing exploitation and exploration of the design space during the optimization. It is still an open question and active area of research to answer the question of how to add multiple infill points simultaneously in an efficient way.

Sóbester et al. (2004) used a multistart optimization algorithm to find multiple maximum points for the expected improvement function and take advantage of parallel processing resources. Their results indicated accelerated convergence for the optimization with significant reduction in processing time.

In the last years we can observe an increasing research interest in EGO approaches with multiple infill points per cycle. Henkenjohann and Kukert (2007), Ponweiser et al. (2008) and Ginsbourger et al. (2010), for instance, proposed different implementations of parallel EGO approaches by extending the concepts of *generalized expected improvement* and *m-step improvement* proposed in the work by Schonlau (1997).

Ginsbourger et al. (2010) proposed the multi-points or (q -points) expected improvement (q -EI). They derived an analytical expression for 2-EI and statistical estimates based on Monte-Carlo methods for the general case. Since Monte-Carlo methods can be computationally expensive, they also proposed two classes of heuristic strategies to obtain approximately q -EI-optimal infill points, i.e., the Kriging Believer (KB) and the Constant Liar (CL). Based on numerical experiments, they reported that CL appears to behave as reasonable heuristic optimizer of the q -EI criterion.

The use of probability of improvement (PI) to generate multiple infill points was discussed by Jones (2001) as a variant of EGO. In this case EGO uses PI beyond a given target as selection criterion. Maximizing PI based on different targets is a way to balance local (exploitation) and global (exploration) searches. Aggressive targets favor more exploitation than exploration and it is not clear how to set these targets properly.

Viana and Haftka (2010) proposed a multi-point algorithm based on an approximation of PI as infill criterion. Chaudhuri and Haftka (2012) proposed the algorithm EGO-AT by exploring the concept of targets for selecting multiple points (PI), discussed by Jones (2001). With EGO-AT (EGO with adaptive target) it is possible to adapt the targets for each optimization cycle based on the success of meeting the target in the previous cycle and generate multiple infill points.

In a different way, Viana et al. (2013) proposed the MSEGO (multiple surrogates EGO). In this case, they used multiple surrogates simultaneously at each cycle of EGO algorithm. With at least one kriging model available, they imported the uncertainty estimate for this model to the other non-kriging models in the set. By means of different uncertainty estimates, they generate different instances of expected improvement functions to be maximized and to provide multiple parallel infill points in each EGO cycle.

As discussed in Haftka et al. (2016), the seek of theoretical convergence proofs and rates that quantify the benefits of parallel computation constitutes important recent developments in the field. In this sense, several publications that investigate the theoretical bounds and rates of convergence and algorithm properties can be listed and it is remarkable the work of Desautels et al. (2012) in machine learning area. Although the relevant theoretical developments, these studies and proposed algorithms did not provide yet superior performance against the other heuristic approaches that do not have proof of convergence.

Finally, according to Haftka et al. (2016) the field of parallel surrogate-assisted global optimization with expensive models is a relatively new research field that is not mature yet and it is not possible to conclude with respect to the comparative efficiency of different approaches or algorithms. Further research is needed in order to take full advantage of additional improvements provided by parallelized surrogate based global optimization approaches.

In the present work, we suggest an alternative heuristic scheme to generate multiple infill points per cycle in the LSEGO algorithm by taking advantage of multiple surrogates in a form of a LS ensemble, in a similar direction of the multiple surrogates approach proposed by Viana et al. (2013).

Since we have an arbitrary set of M distinct metamodels, that are relatively fast to generate (as compared with the true

simulation model $y(\mathbf{x})$), it is possible to create an arbitrary number of N_p partial LS ensembles $\hat{y}_{ens}^k(\mathbf{x})$, by generating permutations of $\bar{M} < M$ metamodels. Therefore, by means of the N_p partial LS ensembles there are N_p respective expected improvement functions $E^k[I(\mathbf{x})]$ available to generate up to N_p infill points per cycle.

In this case, differently from Viana et al. (2013), it is not required to have at least one kriging model in the set to generate the uncertainty estimates, since in LSEGO $\hat{s}_k^2(\mathbf{x})$ are generated directly from the least squares definition for the partial ensembles.

Based on preliminary tests, we observed a good performance of LSEGO for the purpose of multiple infill points per cycle. The final LSEGO algorithm is summarized in Section 3.4 and in Section 3.5 we illustrate the application and behavior of LSEGO for one and two variables examples.

3.4 LSEGO algorithm with parallel infill points

The LSEGO algorithm with parallel infill points can be summarized in the following steps:

- i. Define a set χ of N sampling points and start the optimization cycles ($j \leftarrow 1$);
- ii. Evaluate the true response $y(\mathbf{x})$ at all data sites in χ , at the current cycle j , and set

$$y_{min} \leftarrow \min(y_1, \dots, y_N);$$

- iii. Generate the M metamodels $\hat{y}_i(\mathbf{x})$, the N_p partial LS ensembles $\hat{y}_{ens}^k(\mathbf{x})$ and the respective expected improvement functions $E^k[I(\mathbf{x})]$, with all data available at the current cycle j ;
- iv. Find the set of next distinct $N_p^* \leq N_p$ infill points $\chi_{infill} \leftarrow [\mathbf{x}_{N+1}, \dots, \mathbf{x}_{N+1+N_p^*}]$ as the respective maximizers of $E^k[I(\mathbf{x})]$; evaluate the true function $y(\mathbf{x})$ at the N_p^* infill points and add them to the sampling space: $\chi \leftarrow \chi \cup \chi_{infill}$;
- v. If the stopping criteria is not met, set

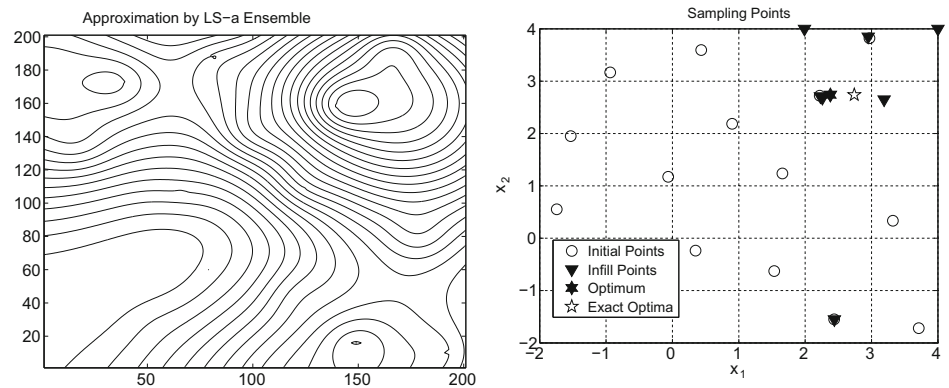
$$y_{min} \leftarrow \min(y_1, \dots, y_{N+N_p^*}),$$

set ($N \leftarrow N + N_p^*$), update cycle counter ($j \leftarrow j + 1$) and return to step iii. Otherwise, finish the LSEGO algorithm.

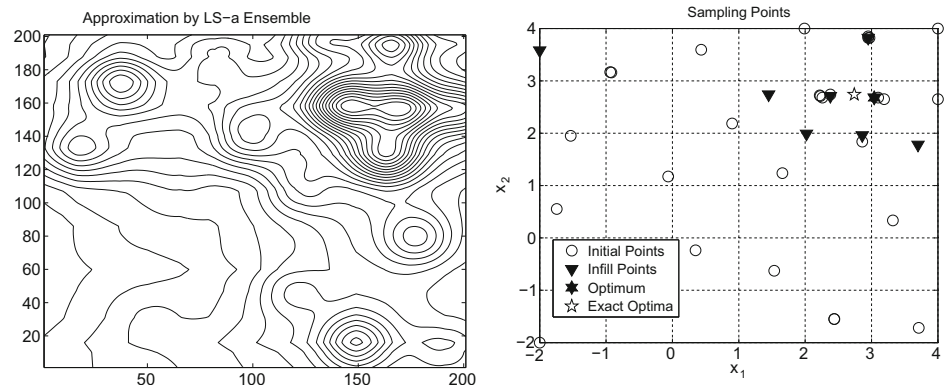
3.5 Illustrations: multiple infill points per cycle

We will illustrate the behavior of LSEGO with multiple infill points with functions of two variables. Figure 5 shows the same setup used in the case of Fig. 3 for Giunta-Watson function (2 variables), but now with $N_p = 8$ infill points per optimization cycle.

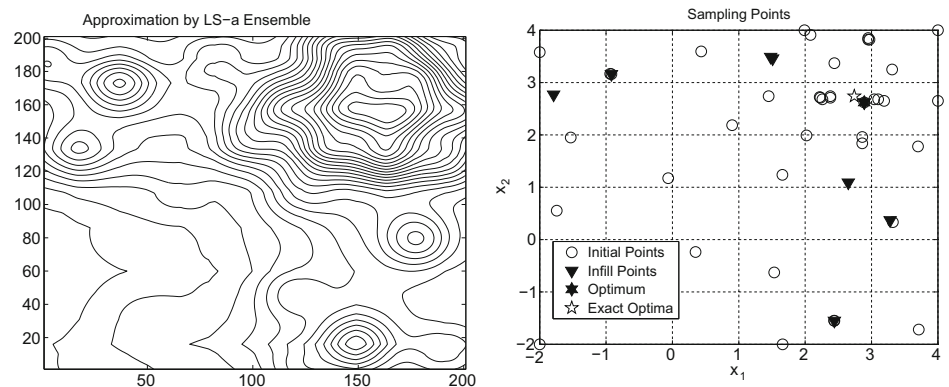
Fig. 5 Evolution of approximation of Giunta-Watson function (2 variables), with LSEGO-8: $N_p = 8$ infill points per optimization cycle and the LS-a ensemble is used with four metamodels: PRS ($ID = 1$), KRG ($ID = 2$), RBNN ($ID = 3$) and SVR ($ID = 4$), see Table 1



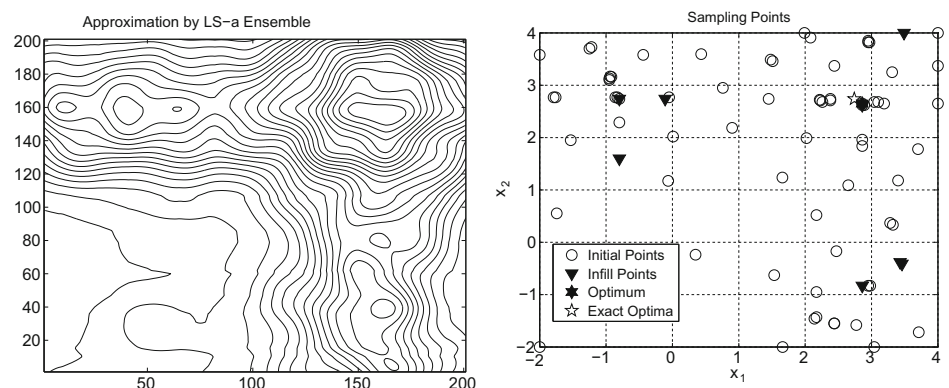
(a) Cycle 01



(b) Cycle 03



(c) Cycle 05



(d) Cycle 10

Note in Fig. 5 that, by allowing more infill points per cycle, LSEGO converges very quickly to the exact optimum at cycle 5 (as compared to cycle 37, for $N_p = 1$ in Fig. 3), with a reasonable correlation for the metamodel at this point ($R^2 = 0.803$ and $NRMSE = 8.1\%$). In addition, if we let LSEGO to continue with the exploration, the metamodel quality is continuously improved. Observe in Fig. 6 the results at cycle 12 ($R^2 = 0.998$ and $NRMSE = 0.6\%$), that can be explained by the increased spread of infill points (exploration), not only on the vicinity of the optimum (exploitation), as in the case for one infill point per cycle.

We observed the same performance of LSEGO for other functions as well. See for instance the evolution for the Branin-Hoo function (ref. Appendix B, (18)) in Figs. 7 and 8. In this case, LSEGO has found the true optimum points within high accuracy at cycle 05 ($R^2 = 0.999$ and $NRMSE = 0.5\%$).

Based on these preliminary results with one and two dimensional functions, we can conclude that LSEGO has a good performance on driving EGO algorithm, with single and multiple infill points per cycle. As more infill points are added per cycle, the faster is the convergence to the global optimum (exploitation) and also the quality improvement (predictability) of the metamodel in the whole design domain (exploration). In the Section 4 we will show the numerical experiments performed with the objective to compare the performance of the proposed algorithm LSEGO versus the traditional EGO.

4 Numerical experiments

In this work the main objective is to compare the performance of the proposed algorithm LSEGO versus the traditional EGO. The approach followed for analysis here was based mainly on the work by Viana et al. (2013), with specific changes in in the overall setup for the numerical experiments. In addition we applied LSEGO in the optimization constrained optimization of analytical benchmark functions, with the approach outlined in Section 2.4, based on (6).

4.1 Computer implementation

We used the Matlab³ SURROGATES Toolbox v2.0 ref. Viana (2009), as platform for implementation and tests. See Appendix C for details.

³Matlab is a well known and widely used numerical programming platform and it is developed and distributed by The Mathworks Inc., see www.mathworks.com.

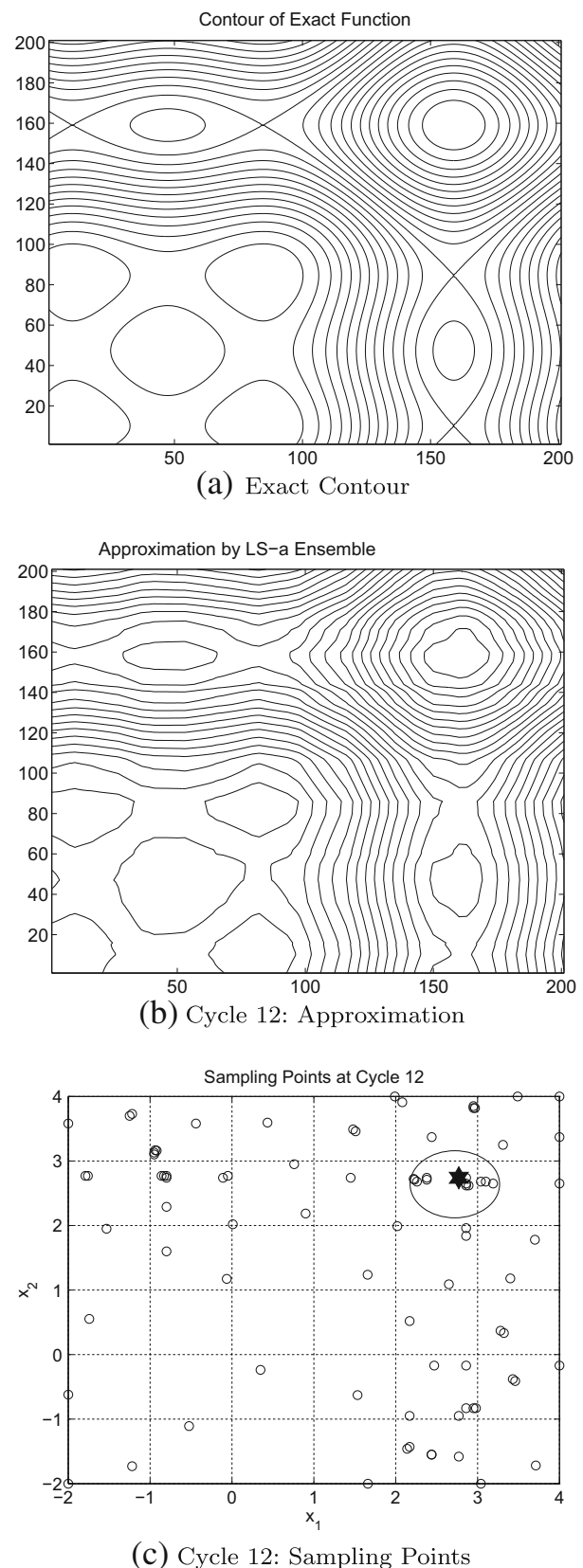
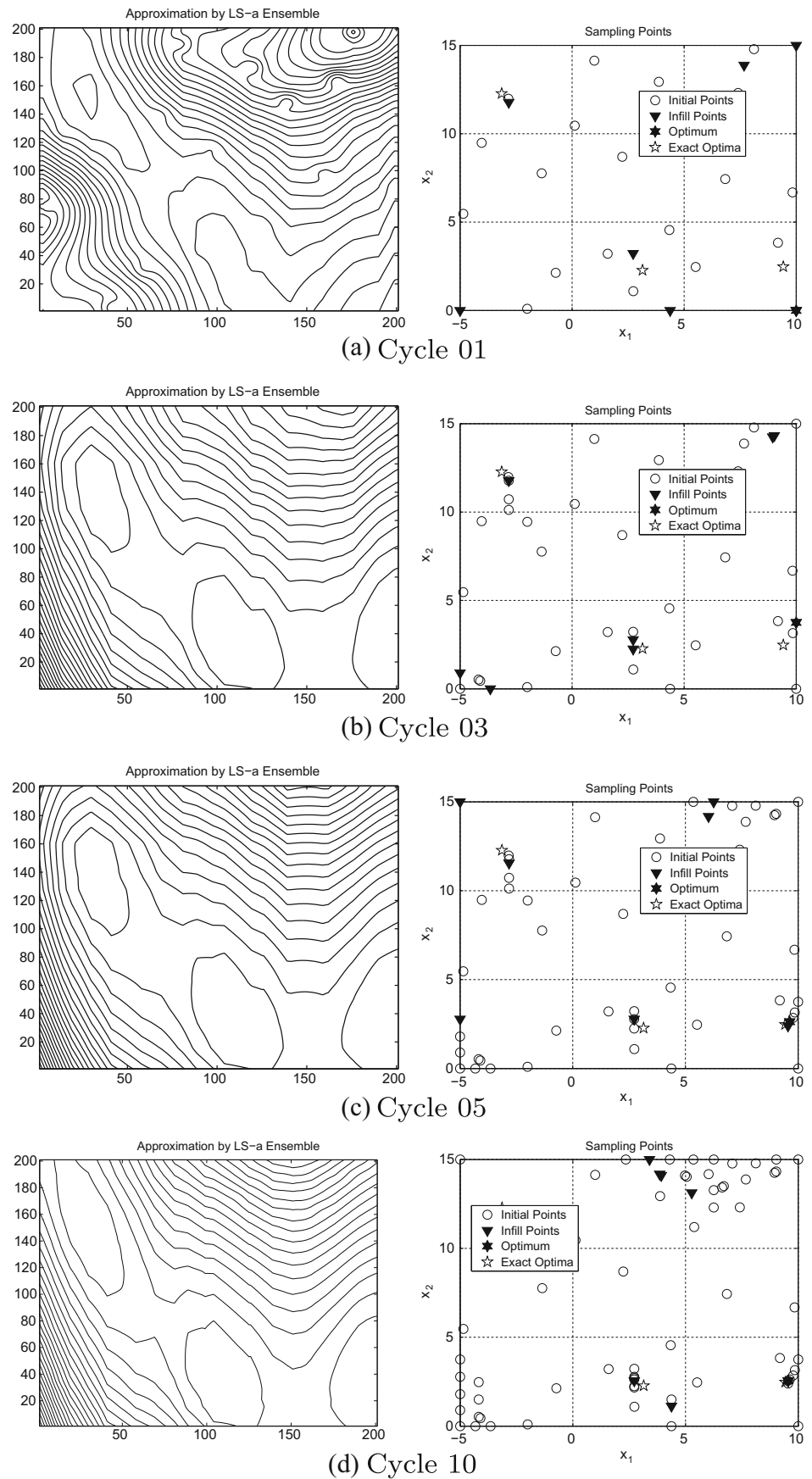


Fig. 6 Detail at Cycle 12 for the evolution of approximation of Giunta-Watson function (2 variables), with LSEGO-8: $N_p = 8$ infill points per optimization cycle

Fig. 7 Evolution of approximation of Branin-Hoo function (2 variables), with LSEGO-8: $N_p = 8$ infill points per optimization cycle and the LS-a ensemble is used with four metamodels: PRS ($ID = 1$), KRG ($ID = 2$), RBNN ($ID = 3$) and SVR ($ID = 4$), see Table 1



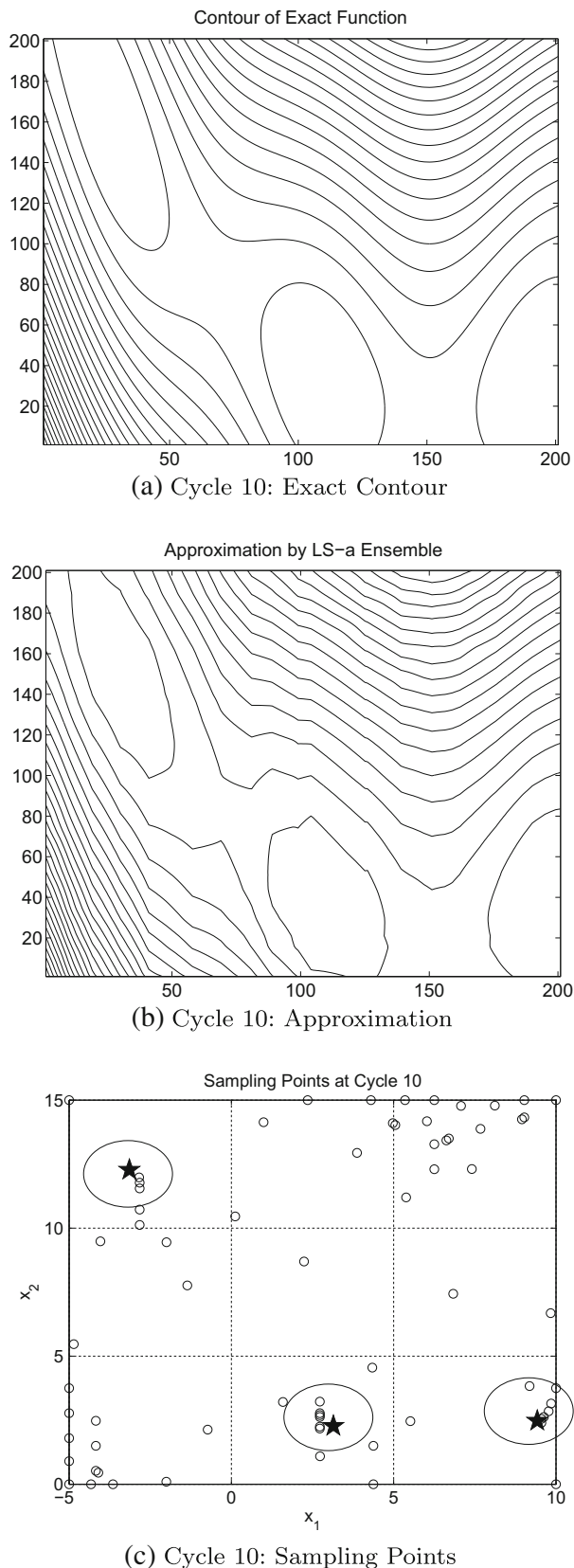


Fig. 8 Detail at Cycle 10 for the evolution in the approximation of Branin-Hoo function (2 variables) with LSEGO-8: $N_p = 8$ infill points per optimization cycle

In our previous work, Ferreira and Serpa (2016), we implemented routines for LS ensemble of metamodels. In the present work we extended the implementations to include the standard EGO and LSEGO algorithms as well, as described respectively in Sections 2.3 and 3.4.

The numerical implementation has been performed with Matlab v2009, on a computer Intel(R) Core(TM) i7-3610QM, CPU 2.30GHz, 8Gb RAM, 64bits, and the Windows 7 operating system.

4.2 Experimental setup

4.2.1 Analytical benchmark functions

We used three well known analytical functions with different number of variables (n_v) for testing the optimization algorithms: Branin-Hoo ($n_v = 2$), Hartman-3 ($n_v = 3$) and Hartman-6 ($n_v = 6$). See Appendix B for the respective equations.

For the constrained optimization experiments we generated the constraints by following the approach used in Forrester et al. (2008) to test the constrained expected improvement formulation.

That is, for Branin-Hoo function, let $x_1^* = 3\pi$ and $x_2^* = 2.475$ be the coordinates of one of the three local optima, then we write the normalized constraint as follows

$$g(x_1, x_2) = \frac{x_1 x_2}{x_1^* x_2^*} - 1 \geq 0. \quad (10)$$

In this way, by using this hyperbola function, at least one local optimum is forced to lie exactly at the constraint boundary.

The same idea was used for Hartman-3 and Hartman-6 functions, with their respective global optima, listed in Appendix B.

4.2.2 Ensembles of metamodels

The ensemble of metamodels were created with the augmented least squares approach LS-a (ref. Ferreira and Serpa 2016), with $\eta_{aug} \approx 33\%$ and nine distinct models of type PRS, KRG, RBNN and SVR, by considering the setup presented in Table 1. Refer to SURROGATES Toolbox manual (ref. Viana 2009) for details on the equations and tuning parameters for each of these metamodeling methods.

The EGO algorithm was implemented with the KRG model $ID = 2$ presented on Table 1. In case of LSEGO, we used ten permutations of the nine models displayed on Table 1 as follows. The first (full) ensemble used all the nine metamodels. For the second partial ensemble we removed the model with $ID = 9$ from the full ensemble. For the third one, the model with $ID = 8$ was removed from the full ensemble and we continue this way up to ten ensembles,

Table 1 Basic metamodels setup for creating the ensembles

ID	Type	Details
1	PRS	Full quadratic model
2	KRG	Quadratic regression, exponential correlation, $\theta_0 = 10$ and $10^{-2} \leq \theta_i \leq 200$
3	RBNN	$Goal = (0.05\bar{y})^2$, $Spread = 2/5$ and $MN = N$
4	SVR	$C = 100max(\bar{y} + 3\sigma_y , \bar{y} - 3\sigma_y)$ and $\epsilon = \sigma_y/\sqrt{N}$
5	PRS	Linear model
6	KRG	Linear regression, Gaussian correlation, $\theta_0 = 10$ and $10^{-2} \leq \theta_i \leq 200$
7	RBNN	$Goal = (0.05\bar{y})^2$, $Spread = 1/3$ and $MN = N/2$
8	SVR	$C = 100max(\bar{y} + 3\sigma_y , \bar{y} - 3\sigma_y)$, $\epsilon = \sigma_y/\sqrt{N}$, $KernelOptions = 1/2$ and $Loss = Quadratic$
9	KRG	Constant regression, Gaussian correlation, $\theta_0 = 10$ and $10^{-2} \leq \theta_i \leq 200$

Obs.1: KRG: kriging; PRS: polynomial response surface; RBNN: radial basis neural network; SVR: support vector regression

Obs.2: All other parameters not mentioned are kept with default values

Obs.3: \bar{y} , σ_y and N are respectively: mean and standard deviation of y and number of sampling points

Obs.4: No attempt has been made in order to fine tune the surrogate’s shape parameters

i.e., one full LS ensemble ($M = 9$) and nine partial LS ensembles ($\bar{M} = 8$).

4.2.3 Design of experiments

The quality of the approximation by metamodels and the rate of convergence to the optimum is strongly dependent on the number and distribution of the initial sampling points defined in the design space (i.e., DOE). In the cases investigated, as a common practice in comparative studies of metamodeling performance, we repeated each experiment with 100 different DOE, in order to average out the influence of random data on the quality of fit. The detailed setup for the initial DOE for each test problem is presented in Table 2. We used the same number of initial points N_{tr} in the DOE as used in Jones et al. (1998).

The 100 different initial DOE with N points ($N = N_{tr} + N_{add}$) were created by using the Latin Hypercube Matlab function `lhsdesign`, optimized with `maximin` criterion set to 1000 iterations. At each cycle of LSEGO, the augmenting points N_{add} are chosen randomly for the

full sampling set (N) to generate the LS-a ensemble with constant rate $\eta_{aug} \approx 33\%$.

4.2.4 Setup for EGO algorithms

In the optimization of each test problem (i.e., Branin-Hoo, Hartman-3 and Hartman-6), we repeated EGO and LSEGO $N_{rep} = 100$ times, in order to average out the effect of different initial DOE on the convergence.

In case of EGO we used the standard case ($N_p = 1$) infill point per cycle. In case of LSEGO we used for Branin-Hoo ($N_p = 2, 5$ and 10) points per cycle and for Hartman functions ($N_p = 10$). The acronym LSEGO varies as function of N_p , e.g., LSEGO-1 stands for LSEGO with $N_p = 1$ and LSEGO-10 stands for LSEGO with $N_p = 10$.

In order to compare the rate of convergence (y_{min} vs. cycles), the total number of cycles allowed to run was set 5 for Branin-Hoo, 10 for Hartman-3 and 15 for Hartman-6. For comparison of variability of y_{min} at each cycle, as function of different DOE, we used `boxplots`.⁴

In order to compare the *level of improvement* (L_{imp}) versus number of function evaluations (f_{eval}), for all test problems, it was considered $f_{eval}^{max} = 50$, with $N_{rep} = 25$

Table 2 Basic specifications for the initial DOE

Test Problem	n_v	N_{tr}	N_{add}
Branin-Hoo	2	21	10
Hartman-3	3	33	16
Hartman-6	6	65	32

n_v : number of variables, N_{tr} : training points and N_{add} : augmenting points for LS-a ensemble

⁴Boxplot is a common statistical graph used for visual comparison of the distribution of different variables in a same plane. The box is defined by lines at the lower quartile (25%), median (50%) and upper quartile (75%) of the data. The lines extending above and upper each box (named as *whiskers*) indicate the spread for the rest of the data out of the quartiles definition. If existent, outliers are represented by plus signs “+”, above/below the whiskers. We used the Matlab function `boxplot` (with default parameters) to create the plots.

repetitions. In this case L_{imp} is calculated for each cycle k as:

$$L_{imp} = 100\% \times \left| \frac{y_{min}^k - y_{min}^0}{y_{min}^{exact} - y_{min}^0} \right|, \quad (11)$$

where y_{min}^0 is the minimum for the initial sample, y_{min}^k is the minimum value at cycle k and y_{min}^{exact} is the exact minimum value.

4.2.5 Maximization of the expected improvement functions

In case of standard EGO (with one point per cycle) we used the Matlab built-in genetic algorithm optimizer `ga` with both `PopulationSize` and `Generations` set to 100. The `InitialPopulation` was set with size $10n_v$ individuals, chosen by using the function `lhsdesign`, optimized with `maxmin` criterion set to 1000 iterations.

In case of LSEGO with multiple infill points there are many expected improvement functions to maximize per cycle. The use of a genetic algorithm can be quite time consuming in this case. Based on preliminary numerical experiments, we found a good balance in accuracy and computation time by using the Matlab pattern search algorithm `patternsearch`, with $10n_v$ initial points (X_0), chosen by using the function `lhsdesign`, optimized with `maxmin` criterion set to 1000 iterations.

As discussed in Jones (2001) the EGO-type algorithms tends to generate infill points quite close to each other in several cycles. Sampling points too close can degenerate the approximation of many metamodels, in special KRG and RBF. In addition, infill points too close have low contribution to the exploration of the design space, they can be a waste of resources and in addition lead to a slower convergence for the optimization. We verified this fact in several preliminary numerical experiments (ref. for instance the illustration example of Fig. 3).

In this case, in order to avoid approximation issues during EGO and LSEGO cycles, we generated exceeding infill points per cycle and selected *distinct* N_p with highest expected improvement value. In our tests we found a good balance by generating $2N_p$ candidate infill points and then we used a *clustering* procedure to remove points too close, or even equal to each other. After the clustering selection, the N_p distinct points are added to the sampling space for the next optimization cycle.

In our first tests we used the Matlab function `unique` to remove equal points from the sampling space, but this procedure has shown to be not effective. Then we implemented a clustering selection procedure by using the Matlab function `cluster` with the `distance` criterion. The `cutoff` was based on the maximum distance (d_{max}) among points/clusters in the whole sampling space, by using

the Matlab `linkage` function. In preliminary numerical tests we found that `cutoff` around to 10% of d_{max} is effective to remove too close points.

5 Results and discussion

5.1 Comparative study: LSEGO versus EGO

The objective in this test set is to compare the performance of the proposed algorithm LSEGO versus the traditional EGO.

In Fig. 9 is presented the optimization results (median over 100 runs starting with different experimental designs) for the classical efficient global optimization algorithm EGO (only with kriging and one infill point per cycle) and LSEGO (with LS ensemble of surrogates/metamodels and different number of points per cycle, N_p).

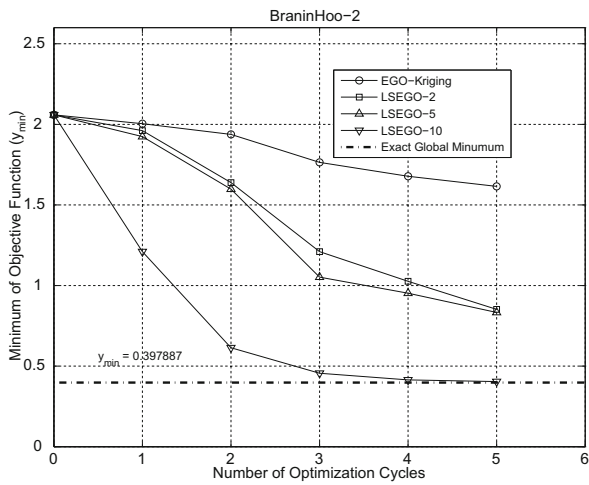
For the three functions studied here, it was observed significant improvement on the reduction of number of cycles for convergence by adding more infill points per cycle. The higher the number of points per cycle, faster the convergence. In addition this effect of accelerated convergence per cycle is more evident as the number of variables increases.

In case of Branin-Hoo function, we tested LSEGO for $N_p = 2, 5$ and 10. For Hartman-3 and Hartman-6 it is presented only for $N_p = 10$. In case of Hartman functions we found a similar trend, as presented in Viana et al. (2013), with little difference in convergence for $N_p = 5$ and $N_p = 10$. In this way we removed $N_p < 10$ from the scope for these cases.

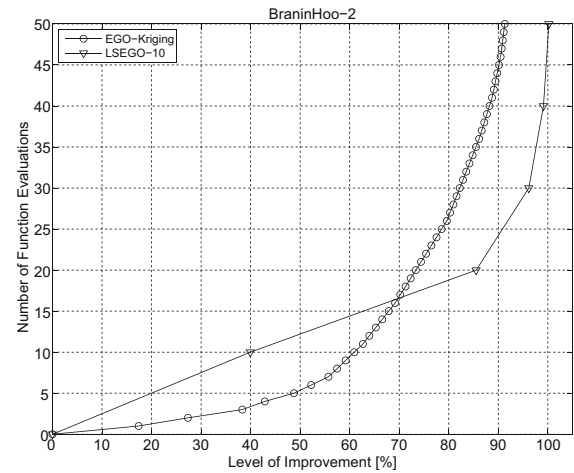
Figure 10 is the counterpart of Fig. 9, by plotting the results versus function evaluations instead of optimization cycles. The idea is to compare LSEGO and EGO in terms of computational costs, measured by the number of simulations (function evaluations) required for the same level of improvement on the objective function at each optimization cycle.

As showed in the curves of Fig. 10, we can observe that for Branin-Hoo and Hartman-3 (low to mid dimensions) at the beginning of the optimization cycles EGO-Kriging has lower cost in terms of number of function evaluations than LSEGO-10 for the same level of improvement, but at some point (e.g., around 70% for Branin-Hoo) the situation is more favorable to LSEGO-10, which is also observed of Hartman-3. In case of the Hartman-6 function, the convergence of EGO-Kriging is quite slow and LSEGO-10 always outperforms EGO-Kriging in terms of function evaluations for the same level of improvement.

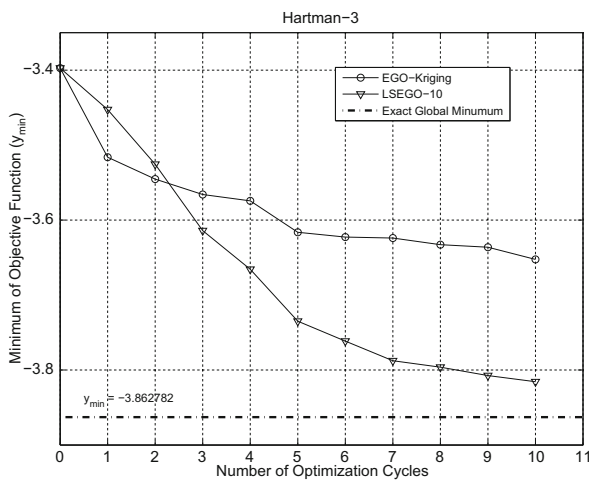
These results confirm the fact that by using multiple infill points per cycle with EGO algorithms is, in general, beneficial in terms of delivering results in reasonable processing time (when parallel computation is available), but on the



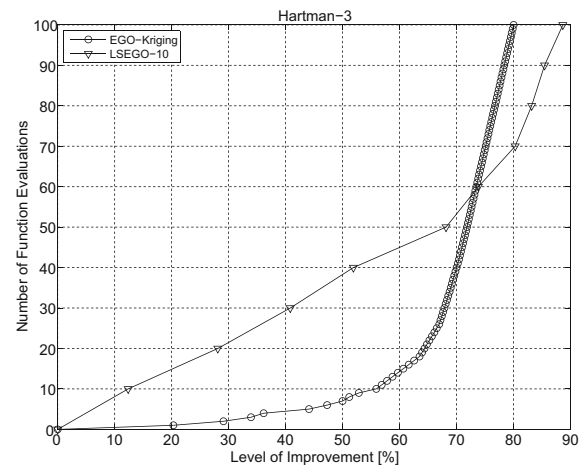
(a) Branin-Hoo, $n_v = 2$



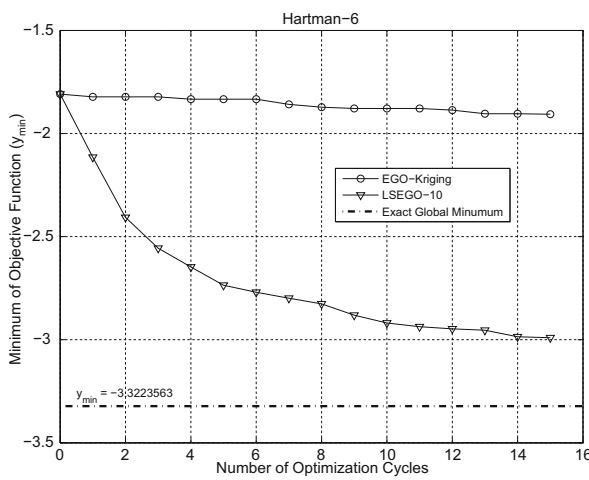
(a) Branin-Hoo, $n_v = 2$



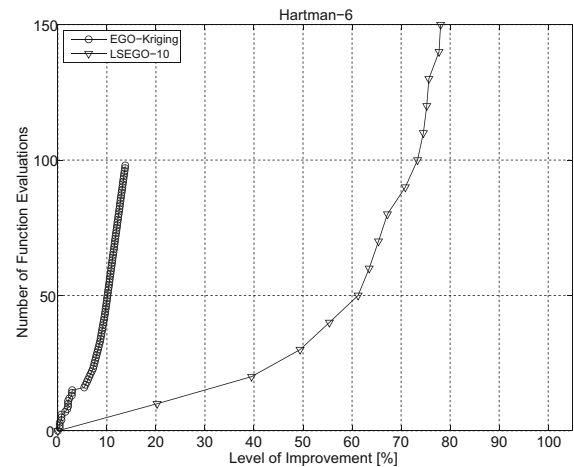
(b) Hartman-3, $n_v = 3$



(b) Hartman-3, $n_v = 3$



(c) Hartman-6, $n_v = 6$



(c) Hartman-6, $n_v = 6$

Fig. 9 Comparison EGO-Kriging versus LSEGO variants for the functions Branin-Hoo, Hartman-3 and Hartman-6. Median (over 100 different initial sampling, DOE) for the efficient global optimization results as function of the number of cycles. The convergence to the exact global minimum is accelerated by adding more points per cycle with LSEGO in all the cases studied

Fig. 10 Comparison EGO-Kriging versus LSEGO-10. Median (over 100 runs with different initial samples, DOE) for the number of function evaluations versus level of improvement. In case of Branin-Hoo and Hartman-3, LSEGO required more function evaluations to achieve the same level of improvement at the beginning of the optimization. In case of Hartman-6 LSEGO required less function evaluations for the same improvement in the whole the process

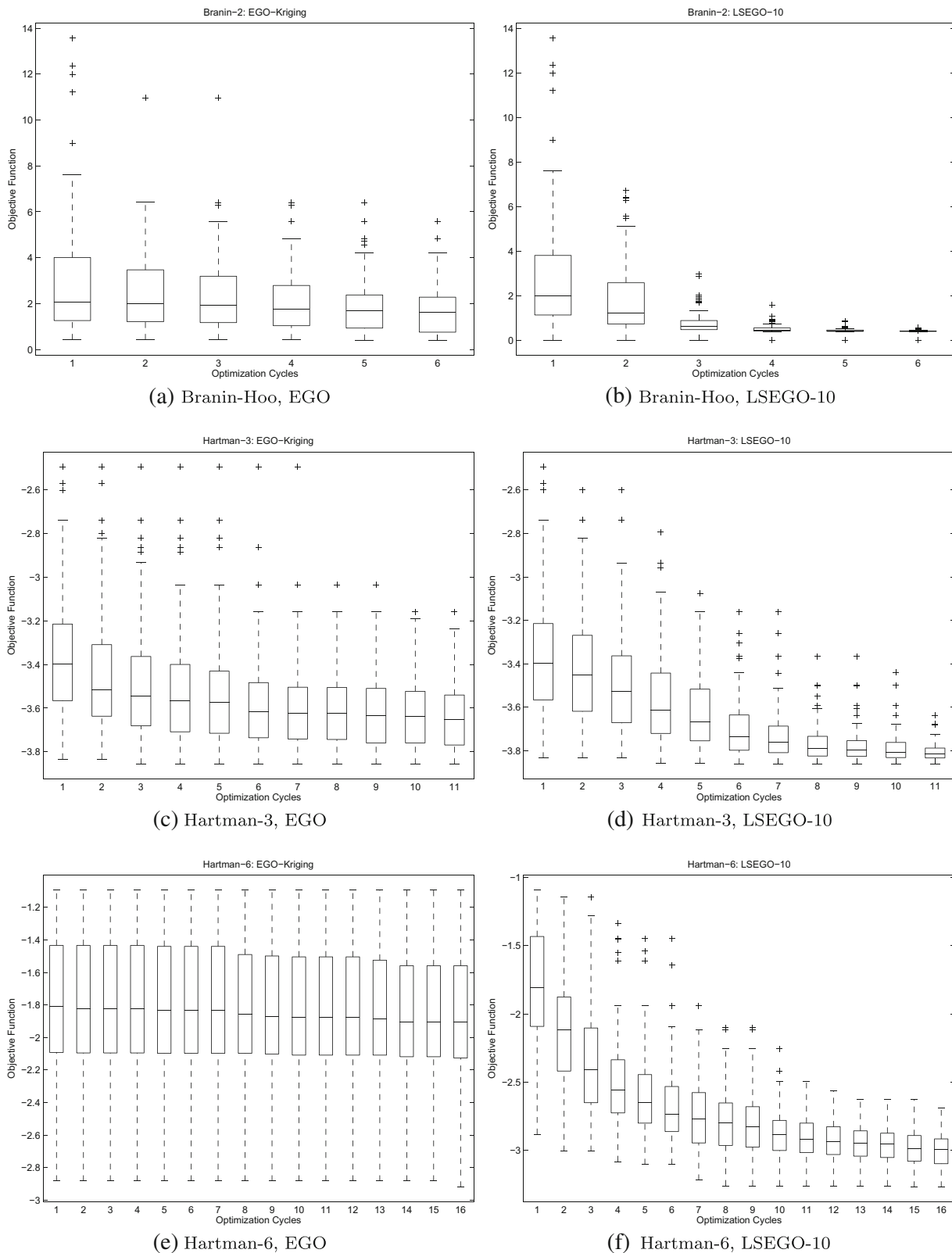


Fig. 11 Comparison of the convergence of EGO-Kriging (left boxplots) vs. LSEGO-10 (right boxplots), over 100 different initial DOE in each case for the functions Branin-Hoo, Hartman-3 and Hartman-6. The variability of the results is higher as the number of variables increases and the convergence to the optimum is very slow with one

infill point and EGO-Kriging. In all the cases, the addition of multiple infill points per optimization cycle with LSEGO-10 accelerates the convergence and also reduce significantly the dispersion of the results as the optimization cycles evolve

other hand it is not guaranteed that the total computational cost of the optimization process (by the total number of parallel simulations) will be reduced at the end of the day. The trend on the results indicate that this effect is problem dependent on the number of variables and the level of nonlinearity of the problem at hand.

Figure 11 presents boxplots of the optimization results for 100 experimental designs for LSEGO vs. EGO. Note that at the beginning of the optimization (cycle 1) the minimum solution found has a high dispersion as function on the initial DOE and this dispersion is rapidly decreasing for LSEGO, what is observed in EGO at much lower rate.

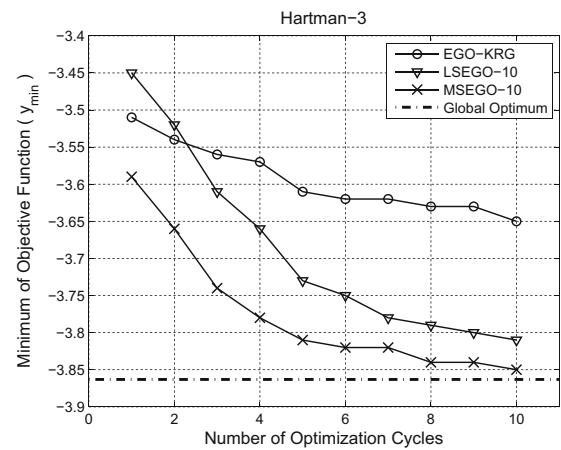
Again, in the same direction as the results presented by Viana et al. (2013), we found that by using LSEGO with multiples points per cycle a significant reduction in the dispersion of the results (minimum of objective function) can be achieved as long as the optimization cycles evolve. This trend confirms that by using LSEGO with multiple points per cycle reduces the variability and dependence of the optimization results on the initial DOE, what is not assured by using EGO with only one infill point per cycle.

In summary the main objective of the experiments in this set, compiled in Figs. 8, 9 and 10, is to highlight that even by paying more function evaluations per cycle, the behavior of parallel EGO-type algorithms can be more stable with higher chance of convergence and real improvement of the objective function with few optimization cycles/iterations. In other words, as we discussed in the Introduction (Section 1), these results confirm that it can be worthwhile to spend more evaluations in a controlled way in order to improve the overall performance of the optimization process.

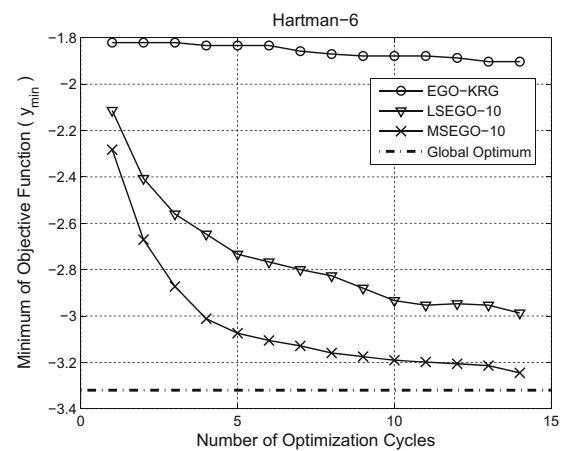
Figure 12 presents a comparison among EGO-Kriging, LSEGO-10 and MSEGO-10 for Hartman-3 and Hartman-6 functions. For both cases MSEGO-10 was faster in convergence rate than LSEGO-10. The maximum difference found for y_{min} at each optimization cycle for MSEGO versus LSEGO is around 4% for Hartman-3 and 14% for Hartman-6.

Our first hypothesis to explain the superior behavior of MSEGO against LSEGO is on the quality/accuracy of the variance estimate for each of these methods. Since the variance estimate of MSEGO is based on the Kriging metamodel in the set, then we performed two more LSEGO runs for Hartman-3 function. The first one with all Kriging metamodels in the set and the other one with no Kriging metamodel at all, for the same set of 100 DOE used to get the results in Fig. 9. These new results are presented in Fig. 12c.

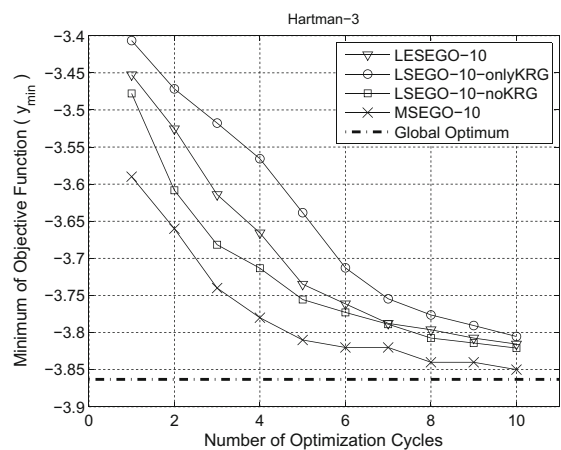
As can be observed in Fig. 12c, surprisingly the best LSEGO performance was for the case with no Kriging metamodel, especially for the first five optimization cycles. These new results indicate that the better performance of



(a) Hartman-3, $n_v = 3$



(b) Hartman-6, $n_v = 6$



(c) Hartman-3, $n_v = 3$

Fig. 12 Comparison among EGO-Kriging, LSEGO-10 and MSEGO-10. Median (over 100 runs with different initial samples, DOE) for the efficient global optimization results as function of the number of cycles. For both cases MSEGO-10 was faster in convergence rate than LSEGO-10. The maximum difference for y_{min} between MSEGO and LSEGO at each optimization cycle is around 4% for Hartman-3 and 14% for Hartman-6. MSEGO-10 data gathered by digitizing the respective figures from Viana et al. (2013)

MSEGO is not due to Kriging variance estimate in MSEGO (as our first assumption). Then, other factors are on the source of this behavior. First of all, the 100 DOE used for LSEGO and MSEGO were not the same. Although in average (or median sense) we can compare both methods, we are not comparing exactly in the same controlled experiments or set of data and these results can vary in some extent especially for different sampling points. In addition, LSEGO and MSEGO have other different aspects in the implementation, e.g., the maximization of expected improvement function (EI), that can contribute for these differences in performance.

In order to answer accurately these differences in the convergence rates we suggest for future work a more comprehensive comparative study involving LSEGO, MSEGO and other similar parallel EGO approaches, in terms of accuracy, variance distribution/breakdown and computational cost in a way to put some light on all these questions.

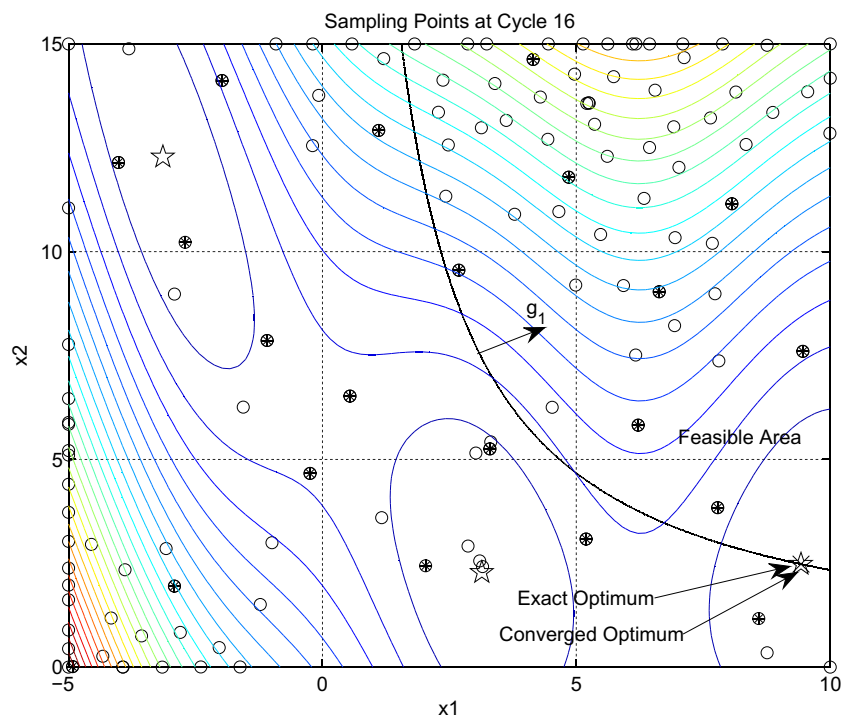
5.2 Results with constrained optimization

5.2.1 Braninh-Hoo

For the Braninh-Hoo function, the results of the constrained optimization with LSEGO-10 are presented in Fig. 13. In this case, with the constraint defined with (10), there is only one global optimum in the feasible area, exactly at the constraint boundary at

$$\mathbf{x}_{exact}^* = (9.425, 2.475).$$

Fig. 13 Optimization results for the constrained Braninh-Hoo function. LSEGO-10 converged exactly to the constrained optimum at $\mathbf{x}^* = (9.425, 2.475)$, after 15 cycles. Note the higher density and uniformity of infill points inside than outside the feasible area. The circles “o” are the infill sampling points and the initial DOE points are labeled with asterisks “*”. The three unconstrained local optima of Braninh-Hoo are labeled as “stars”



Note that the algorithm added infill points at the whole design space, but the density and uniformity of infill points inside is higher than outside the feasible area.

The evolution of the objective function $y(\mathbf{x})$ and the normalized constraint $g(\mathbf{x})$ during the optimization cycles with LSEGO-10 for Braninh-Hoo function are presented in Fig. 14. Note that LSEGO-10 started very far from the global minimum $y^* = 16.4266$ and reached fast the neighborhood of \mathbf{x}_{exact}^* , at the second optimization cycle, with $y^* = 0.5987$, although no further improvement on $y(\mathbf{x})$ was observed until cycle 14. LSEGO-10 converged exactly to the global optimum at cycle 15.

5.2.2 Hartman-3

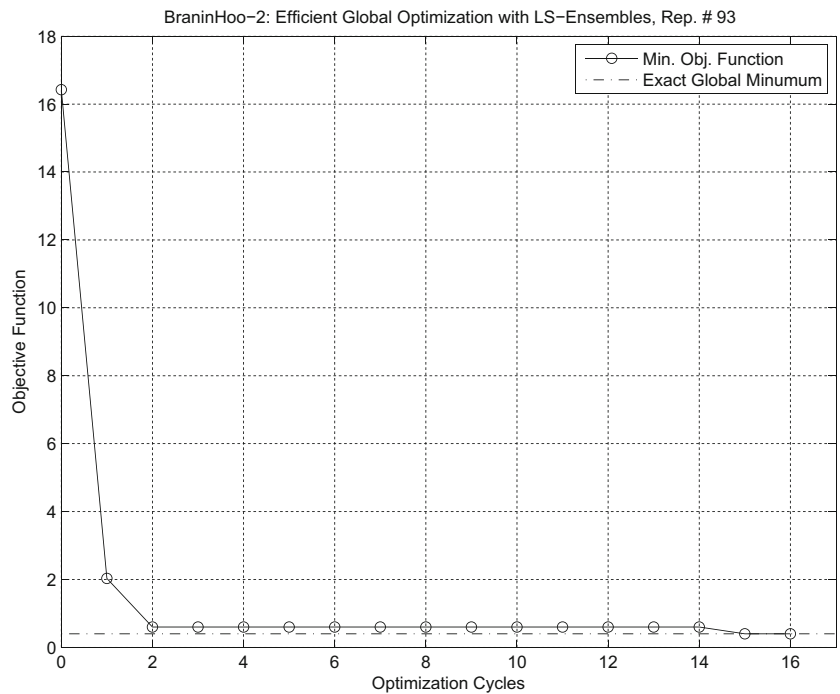
In the same way, the results for Hartman-3 function is presented in Fig. 15. LSEGO-10 reached the neighborhood of \mathbf{x}_{exact}^* at cycle 5, with $y^* = -3.7899$, i.e., 1.89% error from y_{exact}^* .

The algorithm evolved in the cycles, by reducing the value of value of $g(\mathbf{x})$ and trying to reach the constraint boundary (i.e., $g(\mathbf{x}) = 0$). At cycle 15, it was found $y^* = -3.8371$, or 0.66% error from y_{exact}^* . At cycle 31, the algorithm reached

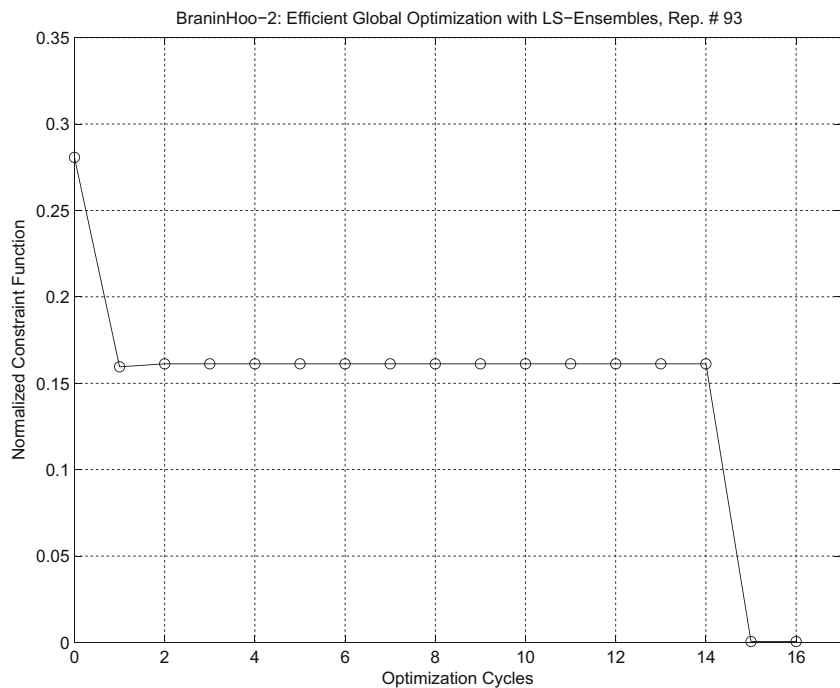
$$\mathbf{x}^* = (0.1444, 0.5553, 0.8537),$$

with $y^* = -3.8621$ or 0.02% error from y_{exact}^* , and no further improvement in $y(\mathbf{x})$ or $g(\mathbf{x})$ was found up to 40 cycles.

Fig. 14 Evolution objective function $y(\mathbf{x})$ and the normalized constraint $g(\mathbf{x})$ during the optimization cycles with LSEGO-10 for Branin-Hoo function. LSEGO-10 started with $y^* = 16.4266$ and reached $y^* = 0.5987$ at cycle 2. LSEGO-10 converged the global optimum with $y^* = 0.3979$ at cycle 15



(a) Objective Function $y(\mathbf{x})$



(b) Normalized Constraint $g(\mathbf{x})$

5.2.3 Hartman-6

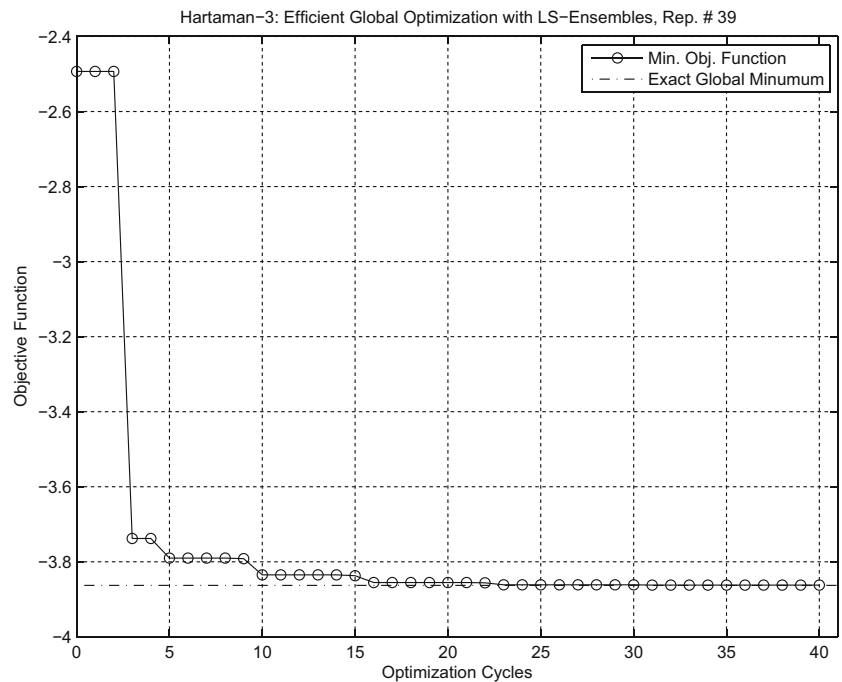
In case of Hartman-6 function, see Fig. 16, the algorithm converged at a slower rate to the neighborhood of \mathbf{x}_{exact}^* . At cycle 35, $y^* = -3.100329$, or 6.68% error from y_{exact}^* . At cycle 38, $y^* = -3.177094$, or 4.37% error from y_{exact}^* . We

stopped the algorithm at cycle 50 with little improvement with respect to cycle 38, i.e.,

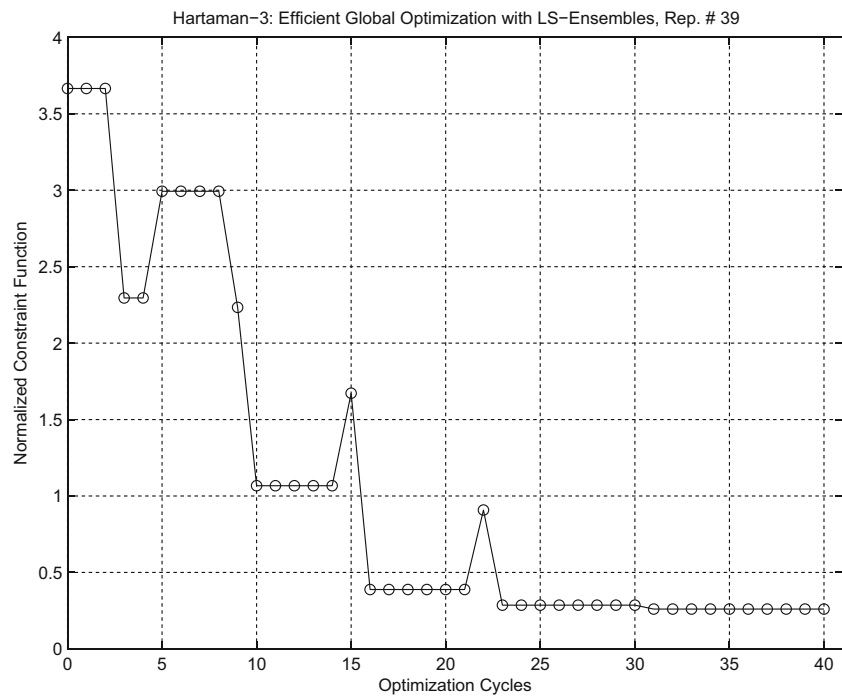
$$\mathbf{x}^* = (0.2688, 0.1547, 0.4508, 0.2890, 0.3433, 0.6541),$$

and $y^* = -3.2082$, or 3.44% error from y_{exact}^* .

Fig. 15 Evolution of the objective function $y(\mathbf{x})$ and the normalized constraint $g(\mathbf{x})$, during the optimization cycles with LSEGO-10 for Hartman-3 function. LSEGO-10 found $y^* = -3.7899$, i.e., 1.89% error from y_{exact}^* at cycle 5. The algorithm converged at cycle 31 with $y^* = -3.8621$ or 0.02% error from y_{exact}^*



(a) Objective Function $y(\mathbf{x})$



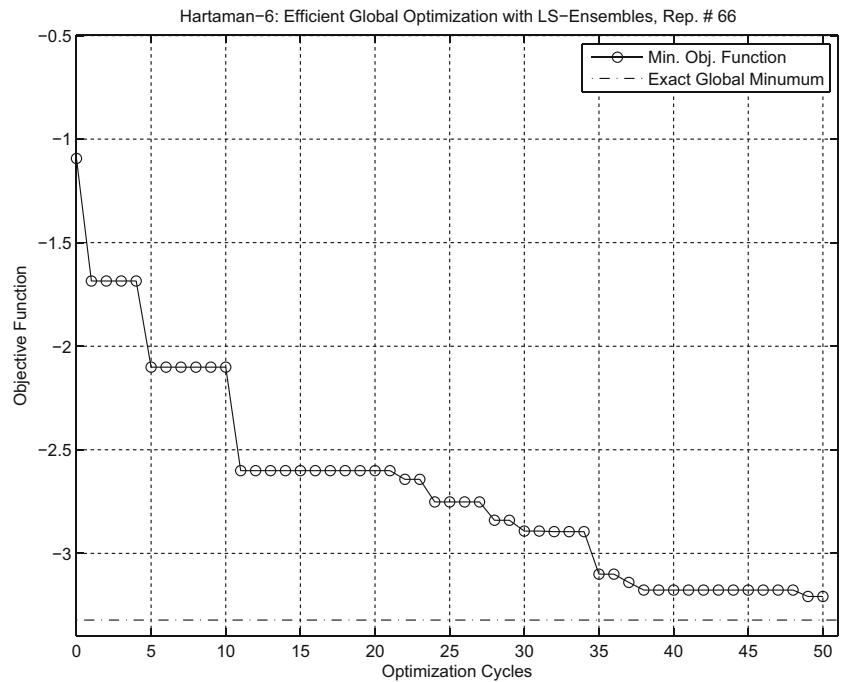
(b) Normalized Constraint $g(\mathbf{x})$

We repeated these numerical experiments with the analytical benchmark functions and the convergence pattern was nearly the same for different initial sampling points. It is worth noting that in all the cases, the optimization algorithm presented the convergence behavior in steps. Observe this fact in Fig. 14 for Branin-Hoo and with more pronounced

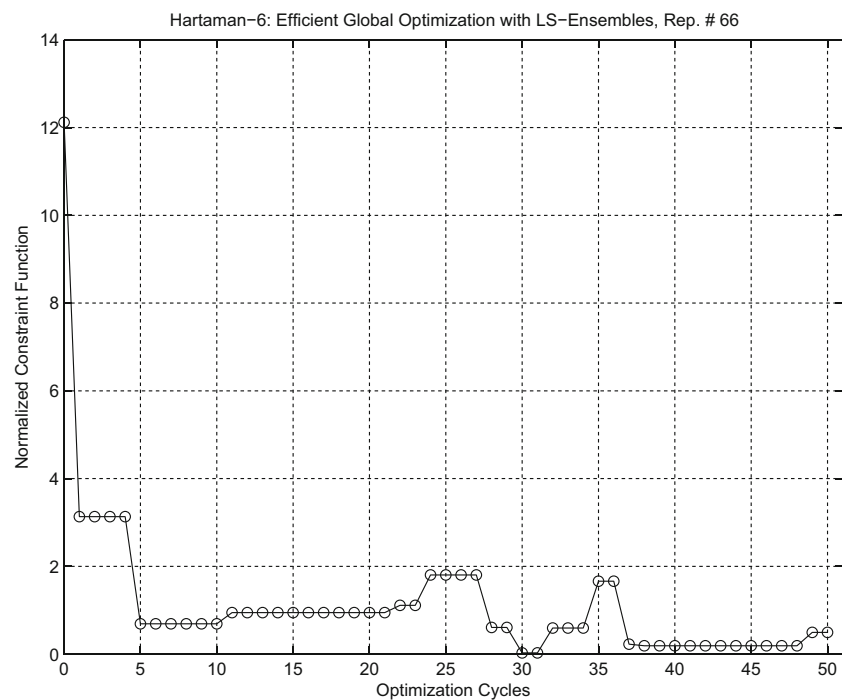
effect in Fig. 15 for Hartman-3 and in Fig. 16 for Hartman-6 function.

As discussed in Forrester and Keane (2009), this stepwise behavior can be understood as the algorithm “switching” from the exploitation to exploration modes, during the optimization cycles. In other words, after adding some infill

Fig. 16 Evolution of the objective function $y(\mathbf{x})$ and the normalized constraint $g(\mathbf{x})$, during the optimization cycles with LSEGO-10 for Hartman-6 function. At cycle 35, $y^* = -3.100329$, or 6.68% error from y_{exact}^* . The algorithm was stopped at cycle 50 with little improvement with respect to cycle 38. At this point $y^* = -3.2082$, or 3.44% error from y_{exact}^*



(a) Objective Function $y(\mathbf{x})$



(b) Normalized Constraint $g(\mathbf{x})$

points in the beginning cycles, the quality of fit of the metamodels increase and the algorithm is able to find some improvement in the objective function (exploitation mode).

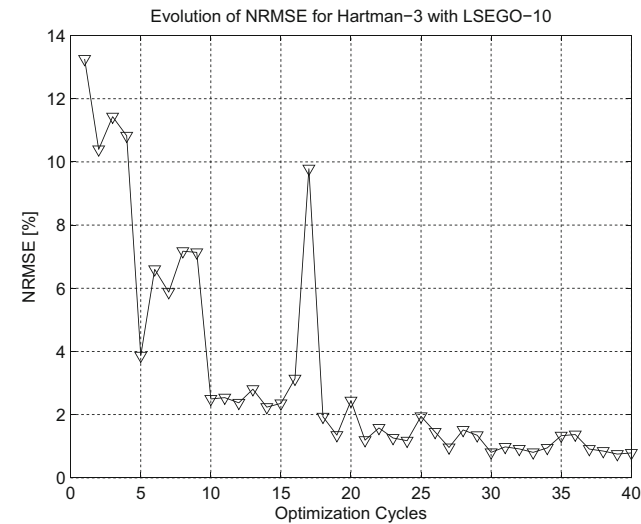
In the sequence, the algorithm switches to the exploration mode at some cycles, and the next “jump” downhill in direction to the optimum is only achieved after the quality of fit of

the metamodels of $y(\mathbf{x})$ and $g(\mathbf{x})$ is enough to promote the next improvement. If we track the quality of fit of the objective function during the optimization cycles, this behavior can be observed.

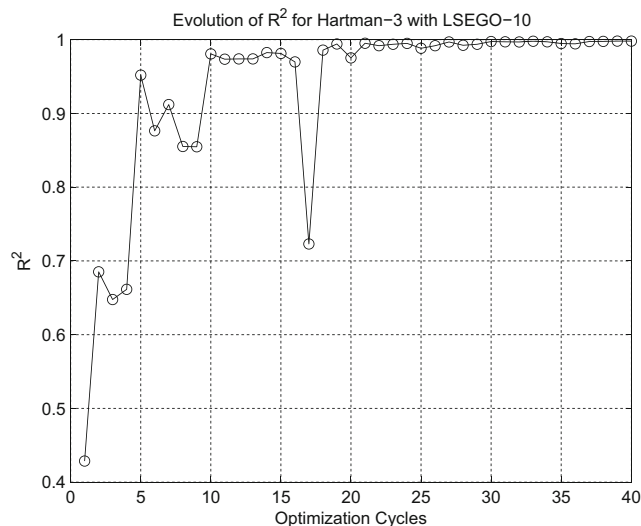
See for instance the stepwise evolution of the quality of approximation of $y(\mathbf{x})$ by means of NRMSE and R^2 , during

the optimization cycles for Hartman-3 function in Fig. 17. In this case, at the initial cycles, the quality of fit is erratic at some extent, with jumps at each three or five steps (cycles 3, 5, 10, 15...).

Recall Fig. 15 and note that these jumps occur simultaneously as the ones at it is observed the main improvements in objective function and for the constraint. When the accuracy of the metamodel reaches stable levels (i.e., $R^2 > 0.9$



(a) NRMSE, Hartman-3



(b) R^2 , Hartman-3

Fig. 17 Evolution of the quality of fit of $y(\mathbf{x})$, by means of NRMSE and R^2 , during the optimization cycles with LSEGO-10 for Hartman-3 function. The optimization algorithm switches from exploitation to the exploration mode, during the cycles and the convergence is achieved in steps. The “jumps” downhill in direction of the optimum occur in steps, after the quality of fit of the metamodels of $y(\mathbf{x})$ and $g(\mathbf{x})$ is enough to promote the improvement

and $NRMSE < 2\%$, around cycle 20, the algorithm is quite close to the global optimum and it converges at cycle 25, when the quality of fit is very good (i.e., $R^2 \approx 1$).

In this sense, based on this observed behavior for the algorithm, it is recommended in practical applications to monitor the quality of fit for the metamodels, in parallel to the evolution of the objective and constraint functions, in order to avoid premature or false convergence at suboptimal points. In practical situations, this balance between quality of fit, improvement of objective function and constraints versus total number of sampling points, must be observed for each problem.

In addition, in several practical problems, improvements in the objective or constraints in the order of 10% to 25% are very hard to meet. In such situations, finding one or a set of truly improved designs is much more important than finding the “global” optimum for the problem, and the decision on when to stop the optimization cycles must be taken based on these considerations as well.

These results with the analytical benchmarks showed that LSEGO algorithm works to handle constrained optimization problems as well. Also, it can be noted that the constraint function is not active at the end of optimization for Hartman-3 function (Fig. 15) and it is “fluctuating” for Hartman-6 function (Fig. 16), although in both cases the optimum result found is close to the constraint boundary. We also observed this behavior in few other numerical tests performed in Ferreira (2016).

As discussed by Viana (2011) in constrained optimization (with constraints being metamodels) or in reliability-based optimization, it can happen that after running the optimization the solution found should be infeasible due to metamodel errors. In order to try to avoid this kind of “pathology”, the first thing that can be done is the correct choice of constraints to be included in the optimization, specially the redundant ones and those that are unlikely to be active (at the constraint boundary), as discussed by Forrester et al. (2008).

In these cases, some sort of penalization approaches can be applied to the inactive or violated constraints to force the optimum to the boundary or feasible region. In addition, other strategies by managing the samples to favor the boundary or the feasible region can be applied, as remarked by Forrester et al. (2008). Viana (2011) extended this discussion and possible directions can be (i) use of conservative constraints based on margin of safety parameters or targets to push the optimum to the feasible region or (ii) use adaptive sampling methods to improve the prediction capability of the constraints in the boundary between feasible and unfeasible domains.

In the present research we did not implement any kind of strategy or control for constraint boundary prediction improvement and feasibility assurance. Although it is still

an open question, since it is a required feature for any optimization algorithm it is strongly recommended to be studied and implemented in future developments of LSEGO and other metamodel based optimization algorithms.

In this sense, a deeper numerical investigation must be performed, with different functions (number of variables, nonlinearity, multimodality, etc.) and increasing number of constraints, in order to understand in detail the behavior, convergence properties, advantages and limitations of LSEGO algorithm.

Finally, there is still some controversy regarding the effectiveness of sequential sampling versus one-stage approaches, see for instance Jin et al. (2002) and Viana et al. (2010). The question is: “what is the advantage of sequential sampling optimization? On the other hand, is it worthwhile to start the optimization with a single shot of a dense DOE with random sampling points in the design space (one-stage) instead of adding few points per cycle?”. In order to try to answer this question in Ferreira (2016) we developed some tests and a brief description is presented in Appendix D.

6 Concluding remarks

In this work we presented LSEGO, an approach to drive efficient global optimization (EGO), based on LS (least squares) ensemble of metamodels. By means of LS ensemble of metamodels it is possible to estimate the uncertainty of the prediction by using any kind of metamodels (not only kriging) and provide an estimate for the expected improvement function. In this way, LSEGO is an alternative to find multiple infill points at each cycle of EGO and improve both convergence and prediction quality during the whole optimization process.

At first, we demonstrated the performance of the proposed LSEGO approach with one dimensional and two dimensional analytical functions. The algorithm has been tested with increasing number of infill points per optimization cycle. As more infill points are added per cycle, the faster is the convergence to the global optimum (exploitation) and also the quality improvement (predictability) of the metamodel in the whole design domain (exploration).

In a second test set, we compared the proposed LSEGO approach with the traditional EGO (with kriging and a single infill point per cycle). For this intent, we used well known analytical benchmark functions to test optimization algorithms, from two to six variables.

For the problems studied, the proposed LSEGO algorithm was shown to be able to find the global optimum with much less number of optimization cycles required by the classical EGO approach. This accelerated convergence was specially observed as the number of variables increased,

when the standard EGO can be quite slow to reach the global optimum.

The results also showed that, by using multiple infill points per optimization cycle, driven by LSEGO, the confidence of metamodels prediction in the whole optimization process is improved. It was observed in the boxplots for all cases investigated a variability reduction with respect to the initial sampling space (initial DOE), as more infill points are added during the optimization cycles.

We also compared LSEGO versus standard EGO in terms of the number of function evaluations, which translates directly to computational cost (i.e., number of simulations required). We can observe that for Branin-Hoo and Hartman-3 (low to mid dimensions) at the beginning of the optimization cycles EGO-Kriging has lower cost in terms of number of function evaluations than LSEGO for the same level of improvement, but as the optimization cycle evolve the situation was more favorable to LSEGO, which is also observed of Hartman-3. In case of the Hartman-6 function, the convergence of EGO-Kriging was quite slow and LSEGO outperformed EGO-Kriging in terms of function evaluations for the same level of improvement in the whole optimization process.

In a comparison with one competitor (MSEGO, from Viana et al. 2013), LSEGO-10 performed with lower convergence rate than MSEGO-10 (about 4% lower for Hartman-3 and 14% lower in case of Hartman-6 function).

In addition we observed that the LSEGO algorithm works well to handle constrained optimization problems, in a feasible number of optimization cycles. In these constrained problems investigated, LSEGO presented a stepwise convergence pattern, which is common to EGO-type algorithms. In this sense, it is recommended in practical applications to monitor the quality of fit for the metamodels, in parallel to the evolution of the objective and constraint functions, in order to avoid premature or false convergence at suboptimal points.

Although LSEGO converged quite well to the optimum region in the constraint optimization of Hartman-3 and Hartman-6 functions, we observed some difficulty to reach exactly the constraint boundary. In this work we did not implement any kind of control method for constraint boundary prediction improvement and feasibility assurance. This is still an open question in the literature and there is no “silver bullet” to solve this kind of issue that is intrinsic to surrogate based constrained optimization. It is a topic to be studied and implemented in future developments of LSEGO. In this way, deeper numerical investigations must be done, with different functions and increasing number of constraints, in order to understand in detail the behavior, convergence properties, advantages and limitations of LSEGO algorithm.

The results achieved in the present work are in accordance with previous work published in the related research area. In this way, the LSEGO approach has shown to be a feasible alternative to drive efficient global optimization by using multiple or ensemble of metamodels, not restricted to a kriging approximation or single infill point per optimization cycles.

In summary we observed that the behavior of parallel EGO-type algorithms can be more stable with higher chance of convergence and real improvement of the objective function with few optimization cycles/iterations. In other words, it can be worthwhile to spend more evaluations in a controlled way and improve the overall performance of the optimization process.

It is worth noting that EGO-type algorithms should be extended to treat properly more general constrained, multi-objective and also robust optimization problems. As future research work, we intend to extend the application of LSEGO approach presented here within the context of constrained and multidisciplinary optimization of a broader set of analytical benchmarks and real world engineering applications. We are already working on this front with good preliminary results, and our intention is to publish them soon.

At last but not least, there are some other open questions and trends for future research and applications of EGO methods (specially with multiple metamodels) raised by the anonymous reviewers, that are:

- How to treat problems with multi-fidelity surrogates in the optimization or even metamodels defined in different regions or clusters of the sampling space?
- Another consideration is that: should the weights be input-space-dependent to deal with heteroskedastic situations or data from different sources (e.g., physical and numerical)?
- Finally, EGO methods are driven by the expected improvement function the requires accurate error/variance estimates to work properly. In this sense, more detailed studies are required to help to understand the key factors that affect the expected improvement function and how to assure the error prediction quality in a broader sense.

Acknowledgements The authors would like to thank Dr. F.A.C. Viana for the prompt help with the SURROGATES Toolbox and also for the useful comments and discussions about the preliminary results of this work.

W.G. Ferreira would like to thank Ford Motor Company and also the support of his colleagues at the MDO group and Product Development department that helped in the development of this work, which is part of his doctoral research concluded at UNICAMP by the end of 2016.

Finally, the authors are grateful for the questions and comments from the journal editors and reviewers. Undoubtedly their valuable suggestions helped to improve the clarity and consistency of the present text.

Appendix A: The kriging metamodel

The Kriging model, originally proposed by Krige (1951), is an interpolating metamodel in which the basis functions, as stated in (1), are of the form

$$\psi^{(i)} = \psi \left(\left\| \mathbf{x}^{(i)} - \mathbf{x} \right\| \right) = \exp \left(- \sum_{j=1}^k \theta_j \left| x^{(i)} - x_j \right|^{p_j} \right), \quad (12)$$

with tuning parameters θ_j and p_j normally determined by maximum likelihood estimates.

With the parameters estimated, the final kriging predictor is of the form

$$\hat{f}(\mathbf{x}) = \hat{\mu} + \boldsymbol{\psi}^T \boldsymbol{\Psi}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu}), \quad (13)$$

where $\mathbf{y} = [y^{(1)} \dots y^{(N)}]^T$, $\mathbf{1}$ is a vector of ones, $\boldsymbol{\Psi} = \psi^{(r)(s)}$ is the so called $N \times N$ matrix of correlations between the sample data, calculated by means of (12) as

$$\boldsymbol{\Psi} = \psi \left(\left\| \mathbf{x}^{(r)} - \mathbf{x}^{(s)} \right\| \right) \quad (14)$$

and $\hat{\mu}$ is given by

$$\hat{\mu} = \frac{\mathbf{1}^T \boldsymbol{\Psi}^{-1} \mathbf{y}}{\mathbf{1}^T \boldsymbol{\Psi}^{-1} \mathbf{1}}. \quad (15)$$

One of the key benefits of kriging models is the provision of uncertainty estimate for the prediction (mean squared error, MSE) at each point \mathbf{x} , given by

$$\hat{\sigma}^2(\mathbf{x}) = \hat{\sigma}^2 \left[1 - \boldsymbol{\psi}^T \boldsymbol{\Psi}^{-1} \boldsymbol{\psi} + \frac{1 - \mathbf{1}^T \boldsymbol{\Psi}^{-1} \mathbf{y}}{\mathbf{1}^T \boldsymbol{\Psi}^{-1} \mathbf{1}} \right], \quad (16)$$

with variance estimated by

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^T \boldsymbol{\Psi}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu})}{N}. \quad (17)$$

Refer to Forrester et al. (2008) or Fang et al. (2006) for further details on metamodel formulation.

Appendix B: Analytical benchmark functions

These functions were chosen since they are widely used to validate both metamodeling and optimization methods, as for example in and Jones et al. (1998) and Viana et al. (2013).

Branin-Hoo

$$y(\mathbf{x}) = \left(x_2 + \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10, \quad (18)$$

Table 3 Data for Hartman-3 function

i	c _i	a _{ij}			p _{ij}		
		j = 1	2	3	j = 1	2	3
1	1	3	10	30	0.3689	0.117	0.2673
2	1.2	0.1	10	35	0.4699	0.4387	0.747
3	3	3	10	30	0.1091	0.8732	0.5547
4	3.2	0.1	10	35	0.03815	0.5743	0.8828

for the region $-5 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 15$. There are 3 minima in this region, i.e., $\mathbf{x}^* \approx (-\pi, 12.275), (\pi, 2.275), (3\pi, 2.475)$ with $f(\mathbf{x}^*) = \frac{5}{4\pi}$.

Hartman

$$y(\mathbf{x}) = - \sum_{i=1}^4 c_i \exp \left[- \sum_{j=1}^{n_v} a_{ij} (x_j - p_{ij})^2 \right], \quad (19)$$

where $x_i \in [0, 1]^{n_v}$, with constants c_i, a_{ij} and p_{ij} given in Table 3, for the case $n_v = 3$ (Hartman-3); and in Tables 4 and 5, for the case $n_v = 6$ (Hartman-6).

In case of Hartman-3, there are four local minima,

$$\mathbf{x}_{local} \approx (p_{i1}, p_{i2}, p_{i3}),$$

with $f_{local} \approx -c_i$ and the global minimum is located at

$$\mathbf{x}^* \approx (0.114614, 0.555649, 0.852547),$$

with $f(\mathbf{x}^*) \approx -3.862782$.

In case of Hartman-6, there are four local minima,

$$\mathbf{x}_{local} \approx (p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5}, p_{i6}),$$

with $f_{local} \approx -c_i$ and the global minimum is located at

$$\mathbf{x}^* \approx (0.201690, 0.150011, 0.476874, 0.275332, 0.3111652, 0.657301),$$

with $f(\mathbf{x}^*) \approx -3.322368$.

Giunta-Watson This is the “noise-free” version of the function used by Giunta and Watson (1998)

$$y(\mathbf{x}) = \sum_{i=1}^{n_v} \left[\frac{3}{10} + \sin \left(\frac{16}{15} x_i - 1 \right) + \sin^2 \left(\frac{16}{15} x_i - 1 \right) \right], \quad (20)$$

Table 4 Data for Hartman-6 function, c_i and a_{ij}

i	c _i	a _{ij}					
		j = 1	2	3	4	5	6
1	1	10	3	1	3.5	1.7	8
2	1.2	0.05	10	17	0.1	8	14
3	3	3	3.5	1.7	10	17	8
4	3.2	17	8	0.05	10	0.1	14

Table 5 Data for Hartman-6 function, p_{ij}

i	p _{ij}					
	j = 1	2	3	4	5	6
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.665
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

where $\mathbf{x} \in [-2, 4]^{n_v}$.

Appendix C: SURROGATES Toolbox

The SURROGATES Toolbox (ref. Viana 2009) is a Matlab based toolbox that aggregates and extends several open-source tools previously developed in the literature for design and analysis of computer experiments, i.e., metamodeling and optimization. We used the version v2.0, but v3.0 already includes EGO variants.⁵

The SURROGATES Toolbox uses the following collection of third party software published: SVM by Gunn (1997), DACE by Lophaven et al. (2002), GPML by Rasmussen and Williams (2006), RBF by Jakabsons (2009), and SHEPPACK by Thacker et al. (2010). The compilation in a single framework has been implemented and applied in previous research by Viana and co-workers, as for example Viana et al. (2009) and Viana (2011).

Appendix D: A note on sequential sampling vs. one-stage approach

In Ferreira (2016) we investigated some examples with analytical engineering functions. We repeated the one-stage optimization ten times, with different initial DOE, at a very large rate of number of sampling points in terms of number of variables,⁶ i.e., for $f_1(\mathbf{x})$ of *Three-Bar Truss* $N = 120$ ($60n_v$); for *Cantilever Beam* $N = 120$ ($60n_v$); for *Helical Spring* $N = 360$ ($120n_v$) and for *Pressure Vessel* $N = 460$ ($120n_v$).

The results for this experiment are presented in Fig. 18. For the cases investigated, the results showed that there is no guarantee to achieve the exact optimum with a one-stage approach, even starting the optimization with a high density of sampling points in the design space. Probably, the

⁵For further details and recent updates on SURROGATES Toolbox refer to the website: <https://sites.google.com/site/srgtstoolbox/>.

⁶As a common practice for metamodel based optimization purposes, the number of points in initial DOE is often in the range $5n_v$ to $10n_v$.

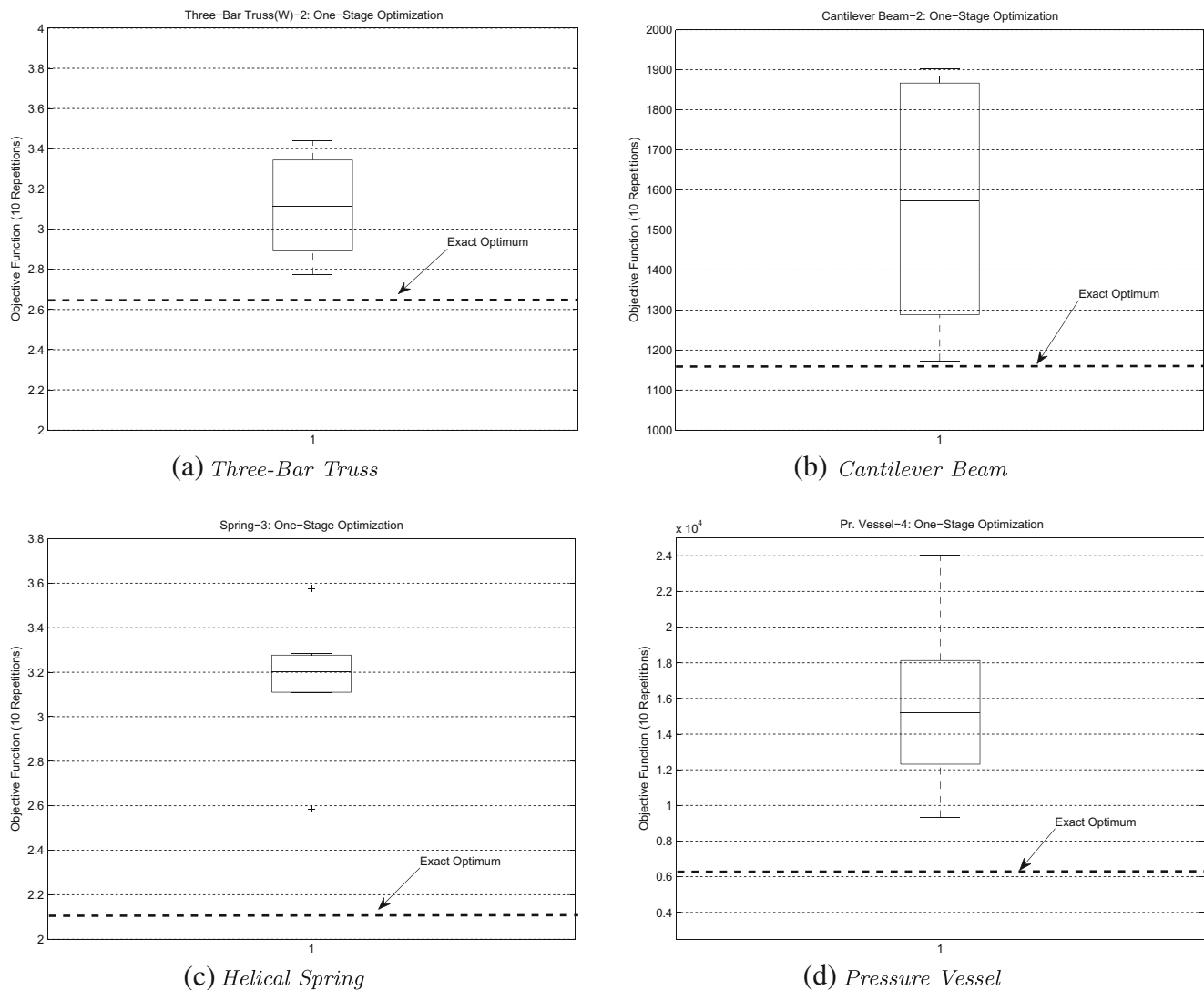


Fig. 18 Boxplots with the converged results for analytical engineering functions with one-stage optimization. Each problem was repeated 10 times with different initial DOE, i.e., for $f_1(\mathbf{x})$ of *Three-Bar Truss* $N = 120$ ($60n_v$); for *Cantilever Beam* $N = 120$ ($60n_v$); for *Helical*

Spring $N = 360$ ($120n_v$) and for *Pressure Vessel* $N = 460$ ($120n_v$). Even with very dense number of initial sampling points, there is no guarantee of achieving the exact optimum. For details refer to Ferreira (2016)

majority of these points are working only for improving the overall quality of the metamodels (exploration) and these points are not being effective to help finding the exact minimum (exploitation), what is clearly a waste of resources for optimization objectives in mind.

These results confirm our beliefs that it is worthwhile to apply sequential sampling approaches like EGO-type algorithms, or some hybrid approach (allied to clustering, for instance), in order to increase the number of points slowly and “surgically” at regions of the design space, with real chance or expectation of improvement in the objective and constraint responses.

In this sense, we reinforce the comments of Forrester and Keane (2009), that the metamodel based optimization must always include some form of iterative search and repetitive

infill process to ensure the accuracy in the areas of interest in the design space. In this direction, we agree on the recommendations that a reasonable number of points for starting the sequential sampling metamodel based optimization is about one third (33%) of the available budget in terms of true function/model evaluations (or processing time) to be spent in the whole optimization cycle.

References

- Chaudhuri A, Haftka RT (2012) Efficient global optimization with adaptive target setting. *AIAA J* 52(7):1573–1578
- Desautels T, Krause A, Burdick J (2012) Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *J Mach Learn Res* 15(1):13,873–3923

- Encisoa SM, Branke J (2015) Tracking global optima in dynamic environments with efficient global optimization. *Eur J Oper Res* 242:744–755
- Fang KT, Li R, Sudjianto A (2006) Design and modeling for computer experiments. Computer science and data analysis series. Chapman & Hall/CRC, Boca Raton
- Ferreira WG (2016) Efficient global optimization driven by ensemble of metamodels: new directions opened by least squares approximation. PhD thesis, Faculty of Mechanical Engineering, University of Campinas (UNICAMP), Campinas, Brazil
- Ferreira WG, Serpa AL (2016) Ensemble of metamodels: the augmented least squares approach. *Struct Multidiscip Optim* 53(5):1019–1046
- Forrester A, Keane A (2009) Recent advances in surrogate-based optimization. *Prog Aerosp Sci* 45:50–79
- Forrester A, Sóbester A, Keane A (2008) Engineering design via surrogate modelling—a practical guide. Wiley, United Kingdom
- Ginsbourger D, Riche RL, Carraro L (2010) Kriging is well-suited to parallelize optimization. In: Computational intelligence in expensive optimization problems - adaptation learning and optimization, Springer, vol 2, pp 131–162
- Giunta AA, Watson LT (1998) Comparison of approximation modeling techniques: polynomial versus interpolating models. In: 7th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization, AIAA-98-4758, pp 392–404
- Gunn SR (1997) Support vector machines for classification and regression. Technical Report. Image, Speech and Intelligent Systems Research Group, University of Southampton, UK
- Haftka RT, Villanueva D, Chaudhuri A (2016) Parallel surrogate-assisted global optimization with expensive functions - a survey. *Struct Multidiscip Optim* 54(1):3–13
- Han ZH, Zhang KS (2012) Surrogate-based optimization - real-world application of genetic algorithms, ISBN 978-953-51-0146-8 edn. InTech, Dr. Olympia Roeva - Editor, Shanghai, China
- Henkenjohann N, Kukert J (2007) An efficient sequential optimization approach based on the multivariate expected improvement criterion. *Qual Eng* 19(4):267–280
- Janusevskis J, Riche RL, Ginsbourger D, Girdziusis R (2012) Expected improvements for the asynchronous parallel global optimization of expensive functions: potentials and challenges. *Learning and Intelligent Optimization* 7219:413–418
- Jekabsons G (2009) RBF: radial basis function interpolation for matlab/octave. Riga Technical University, Latvia version 1.1 ed
- Jin R, Chen W, Sudjianto A (2002) On sequential sampling for global metamodeling in engineering design. In: Engineering technical conferences and computers and information in engineering conference, DETC2002/DAC-34092, ASME 2002 Design, Montreal-Canada
- Jones DR (2001) A taxonomy of global optimization methods based on response surfaces. *J Glob Optim* 21:345–383
- Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. *J Glob Optim* 13:455–492
- Jurecka F (2007) Optimization based on metamodeling techniques. PhD thesis, Technische Universität München, München-Germany
- Koziel S, Leifesson L (2013) Surrogate-based modeling and optimization - applications in engineering. Springer, New York, USA
- Krige DG (1951) A statistical approach to some mine valuations and Allied problems at the Witwatersrand. Master's thesis, University of Witwatersrand, Witwatersrand
- Lophaven SN, Nielsen HB, Sondergaard J (2002) DACE - a matlab kriging toolbox. Tech. Rep. IMM-TR-2002-12, Technical University of Denmark
- Mehari MT, Poorter E, Couckuyt I, Deschrijver D, Gerwen JV, Pareit D, Dhaene T, Moerman I (2015) Efficient global optimization of multi-parameter network problems on wireless testbeds. *Ad Hoc Netw* 29:15–31
- Mockus J (1994) Application of bayesian approach to numerical methods of global and stochastic optimization. *J Glob Optim* 4:347–365
- Ponweiser W, Wagner T, Vincze M (2008) Clustered multiple generalized expected improvement: a novel infill sampling criterion for surrogate models. In: Wang J (ed) 2008 IEEE World congress on computational intelligence, IEEE computational intelligence society. IEEE Press, Hong Kong, pp 3514–3521
- Queipo NV et al (2005) Surrogate-based analysis and optimization. *Prog Aerosp Sci* 41:1–28
- Rasmussen CE, Williams CK (2006) Gaussian processes for machine learning. The MIT Press
- Rehman SU, Langelaar M, Keulen FV (2014) Efficient kriging-based robust optimization of unconstrained problems. *Journal of Computational Science* 5:872–881
- Schonlau M (1997) Computer experiments and global optimization. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada
- Simpson TW, Toropov V, Balabanov V, Viana FAC (2008) Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come - or not. In: 12th AIAA/ISSMO multidisciplinary analysis and optimization conference, Victoria, British Columbia
- Sóbester A, Leary SJ, Keane A (2004) A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Struct Multidiscip Optim* 27:371–383
- Thacker WI, Zhang J, Watson LT, Birch JB, Iyer MA, Berry MW (2010) Algorithm 905: SHEPPACK: modified shepard algorithm for interpolation of scattered multivariate data. *ACM Trans Math Softw* 37(3):1–20
- Venkataraman S, Haftka RT (2004) Structural optimization complexity: what has moore's law done for us? *Struct Multidiscip Optim* 28:375–287
- Viana FAC (2009) SURROGATES Toolbox user's guide version 2.0 (release 3). Available at website: <http://fchegury.googlepages.com>
- Viana FAC (2011) Multiples surrogates for prediction and optimization. PhD thesis, University of Florida, Gainesville, FL, USA
- Viana FAC, Haftka RT (2010) Surrogate-based optimization with parallel simulations using probability of improvement. In: Proceedings of the 13th AIAA/ISSMO multidisciplinary analysis optimization conference, Forth Worth, Texas, USA
- Viana FAC, Haftka RT, Steffen V (2009) Multiple surrogates: how cross-validation error can help us to obtain the best predictor. *Struct Multidiscip Optim* 39(4):439–457
- Viana FAC, Cogu C, Haftka RT (2010) Making the most out of surrogate models: tricks of the trade. In: Proceedings of the ASME 2010 international design engineering technical conferences & computers and information in engineering conference IDETC/CIE 2010, Montreal, Quebec, Canada
- Viana FAC, Haftka RT, Watson LT (2013) Efficient global optimization algorithm assisted by multiple surrogates techniques. *J Glob Optim* 56:669–689