Analysis of statistical model-based optimization enhancements in Generalized Self-Adapting Particle Swarm Optimization framework

Mateusz Zaborski, Michał Okulewicz, Jacek Mańdziuk [*]

Abstract. This paper presents characteristics of model-based optimization methods utilized within the Generalized Self-Adapting Particle Swarm Optimization (GA–PSO) – a hybrid global optimization framework proposed by the authors. GAPSO has been designed as a generalization of a Particle Swarm Optimization (PSO) algorithm on the foundations of a large degree of independence of individual particles. GAPSO serves as a platform for studying optimization algorithms in the context of the following research hypothesis: (1) it is possible to improve the performance of an optimization algorithm through utilization of more function samples than standard PSO sample-based memory, (2) combining specialized sampling methods (i.e. PSO, Differential Evolution, model-based optimization) will result in a better algorithm performance than using each of them separately. The inclusion of model-based enhancements resulted in the necessity of extending the GAPSO framework by means of an external samples memory - this enhanced model is referred to as M-GAPSO in the paper.

We investigate the features of two model-based optimizers: one utilizing a quadratic function and the other one utilizing a polynomial function. We analyze the conditions under which those model-based approaches provide an effective sampling strategy. Proposed model-based optimizers are evaluated on the functions from the COCO BBOB benchmark set.

Keywords: Particle Swarm Optimization, global optimization, metaheuristic

[*]Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland,{M.Zaborski,M.Okulewicz,J.Mandziuk}@mini.pw.edu.pl

1. Introduction

Non-gradient population-based optimization algorithms have grown popular over the last decades. Some of their features that attract the research community include their effectiveness, simplicity of implementation and ease of modification. In the domain of continuous optimization the mostly studied approaches include Particle Swarm Optimization (PSO) and Differential Evolution (DE).

PSO [11] is a well-known global optimization metaheuristic with many possible variants. For instance, Nepomuceno and Engelbrecht [13] proved that appropriate combination of heterogeneous versions of PSO can lead to significant performance improvement. DE [20] is another example of population-based optimization approach with large degree of independence between solutions (individuals).

Despite their effectiveness in solving numerous engineering and production problems [6, 19, 14, 15], those algorithms quite quickly "forget" the shape of optimized function due to their limited memory. Therefore, they loose information which might be useful in accelerating the convergence of the algorithm. One of the approaches, which might be worth looking into to amend this weakness, is surrogate modeling of optimized function [1]. While originally motivated by the research on expensive optimization, those surrogate optimization methods might be hybridized with population based approaches to make a good use of the function samples gathered during the optimization process.

There are many ways of enriching optimization algorithms by some surrogate modeling. In general, surrogate models of the original function are constructed to estimate its value at some point which is cheaper than making an evaluation of the true objective function. Kriging (Gaussian process) metamodel can be utilized as a standalone optimization technique [12]. The principle of its utilization is sequential approximation of the optimized function and designating the next sampling point. Iterative sampling rules lead to gradually decreasing metamodel uncertainty. Therefore, the performance of Kriging metamodel, against population-based approaches like PSO and DE, depends greatly on the computational costs of the objective function. Optimization process performed by Kriging will be more time and memory consuming than PSO's or DE's. However, this approach is profitable when the computational or financial cost of obtaining true objective function values is relatively high. Chung et al. [3] present that computationally expensive fitness function evaluations made by evolutionary algorithm (EA) can be successfully replaced by estimates from surrogate models such as the previously mentioned Kriging. However, Chung et al. also point out the computational complexity of prediction and training using Kriging is $O(n^3)$ for n data points due to necessary matrix inversions (or decompositions). Therefore, this approach can be inefficient for building model utilizing large dataset.

The CMA-ES [8] algorithm can be extended by surrogate models in a similar way. Only the most promising points designated by the metamodel (Gaussian process) can be evaluated by the original fitness [17]. The work showed that enriching CMA-ES with surrogate model reduces the number of necessary evaluations. It should be noted that experiments were conducted up to $100 \cdot D$ evaluations, where D is problem dimensionality, while our goal is to use the limit of 10^6 evaluations.

Cai et al. [2] utilize metamodel to modify PSO velocity updating mechanism and introduce two algorithms (ESPSO and PESPSO). Both utilize Radial Basis Function (RBF) as a surrogate model. In the case of ESPSO, PSO velocity update formula takes into consideration estimation of local and global minimum designated by metamodel. PESPSO additionally utilizes metamodel in the evaluations process. Real evaluation is made only by particles for which the function value estimation (after a position update) is better than their historical minimum. As in previously mentioned works, experiments were carried out relatively low number of evaluations.

Chung et al. [4] also prove that enhancing PSO with RBF metamodel utilizing samples archive can be beneficial. Presented SA-COSO algorithm is a hybrid of two PSO variants (FES-assisted PSO and RBF-assisted PSO). Both PSO variants use function value approximation to limit the number of real evaluations. Presented solution has been tested on 200-dimensional problems and outperformed the state-of-the-art algorithms. Real evaluation number in each problem has not exceed 1000.

Incorporating archive into an optimization algorithm is not always associated with strict surrogate modeling. JADE[24] is an example of DE extension that utilize archive for its parameters adaptation. The algorithm stores successful crossover probabilities and mutation factors and generates their current values based on the means of stored parameters.

Jin [10] surveys the research on fitness approximation in evolutionary computation. Except for the above-mentioned techniques, the work briefly describes the use of neural networks, support vector machines and polynomial models. Modeling function as a polynomial requires a set of fitness evaluations. Polynomial coefficients can be obtained with the least square method.

Generally speaking, in surrogate based optimization, statistical models themselves are subject to optimization by metaheuristics. This work investigates an approach where a PSO-DE hybrid is further enhanced with model-based optimization approaches. Within GAPSO setup they work in parallel with simple sampling approaches like PSO and DE, utilizing samples already gathered through those methods. Their principle is to sequentially construct a surrogate model, estimate local optimum and finally sample a designated point. Moreover, our solution is computationally efficient, i.e. an optimization run with relatively large number of real function evaluations can be executed in a reasonable time.

The rest of the paper is organized as follows. Section 2 introduces the M-GAPSO algorithm proposed by the authors including the method of constructing incorporated surrogate models. Section 3 gives insight into sampling properties of the model-based enhancements utilized in M-GAPSO. The next section presents the results on the COCO BBOB benchmark functions. Section 5 concludes the paper.

2. M-GAPSO framework description

This section summarizes the proposed Generalized Self-Adapting Particle Swarm Optimization framework with external samples memory (M-GAPSO), which is an enhancement of the GAPSO approach [21]. This work is an extension of the research presented in [23], while a detailed description of M-GAPSO can be also found in our working paper [16].

GAPSO optimization framework has been designed as a generalization of the PSO algorithm. It allows the use of virtually any optimization algorithm behavior. *Particles act independently and behave differently according to assigned optimization algorithm.* From the swarm's point of view internal behavior of a *particle* (i.e. function sampling scheme) is irrelevant. Each *particle* is only obliged to update its velocity and maintain its current and best positions. Therefore, the well-known algorithms, such as DE can be easily implemented within GAPSO by means of an appropriate scheme for updating a velocity vector.

Our goal is to study the hybridization of the population based optimization algorithms with a search process guided by statistical models created from objective function samples. Therefore, the original GAPSO algorithm [21] has been extended by an additional memory module, for storing the already sampled function values. Apart from that, the original GAPSO's particle-by-particle restart method has been changed into a global JADE-like restart mechanism [18], as it was found to be more beneficial for the algorithm's performance [16].

A high-level pseudocode of the M-GAPSO is provided in Algorithm 1. The main concept of the algorithm is to apply various particle's location update formulas (behaviors), thus creating a hybrid approach. These behaviors are selected at random in a given iteration (line 10) according to the weight vector assigned to each behavior. It should be noted that the conditional statement starting in line 16 could be easily extended with additional clauses, as lines 15 and 25 provide a universal approach to the PSO-like velocity management for non-PSO algorithms.

The rest of this section provides information about the selection of optimization algorithms, external memory modules and implementation of the model-based behaviors.

2.1. Optimization algorithms within M-GAPSO

Currently the pool of M-GAPSO optimization algorithms contains the following methods: (1) Standard PSO-2007 [5], (2) DE/best/1/bin [20], (3) quadratic model based optimizer and (4) polynomial model based optimizer. Both model based optimizers are the key elements of this work, further discussed in Section 2.3. At every iteration each particle has one of the above optimization algorithms assigned. The assignment is carried out in accordance to established probability vector. Importantly, each behavior is assigned to at least one particle. Probability vector can be fixed during the whole optimization process or can be adaptable.

Adaptation module measures the performance of each particle. It observes the

Algorithm 1 M-GAPSO high–level pseudocode

```
1: $F$ is the objective function, $Swarm$ is a set of PSO particles
2: $Behavior$ is particle's velocity update rule, $BehaviorsPool$ is the set of available optimization algorithms
3: $BehaviorsWeights$ is a vector parameterizing the probability of selecting particular $Behavior$ from $BehaviorsPool$
4: $Initializer$ is particle's initial location sampler
5: $SamplesArchive$ is an RTree based samples' index
6: $RestartManager$ observes swarm state and performance
7: $Swarm \leftarrow Initializer.initializeSwarm(F.bounds)$
8: while Stopping criterion not met do
9:     for $Particle \in Swarm$ do
10:        $Behavior \leftarrow BehaviorsPool.select(BehaviorsWeights)$
11:        if $Behavior == PSO$ then
12:            $Particle.v \leftarrow Particle.applySPSOVelocityUpdate(Swarm)$
13:            $Particle.x \leftarrow Particle.x + Particle.v$
14:        else
15:            $Particle.x_{prev} \leftarrow Particle.x$
16:            if $Behavior == DE$ then
17:                $Particle.x \leftarrow Particle.applyDESampling(Swarm)$
18:            else if $Behavior == QuadraticModel$ then
19:                $QuadraticModel \leftarrow Particle.buildQuadraticModel(SamplesArchive)$
20:                $Particle.x \leftarrow QuadraticModel.selectModelMinimum()$
21:            else if $Behavior == PolynomialModel$ then
22:                $PolynomialModel \leftarrow Particle.buildPolynomialModel(SamplesArchive)$
23:                $Particle.x \leftarrow PolynomialModel.selectModelMinimum()$
24:            end if
25:            $Particle.v \leftarrow Particle.x - Particle.x_{prev}$    ▷ Management of the *velocity* feature for non-PSO optimization algorithms
26:        end if
27:        $Sample \leftarrow F.evaluate(Particle.x)$
28:        if $Sample.value < Particle.best_{value}$ then
29:            $Particle.best \leftarrow Sample.x$
30:            $Particle.best_{value} \leftarrow Sample.value$
31:        end if
32:        $SamplesArchive.store(Sample)$
33:        $RestartManager.register(Sample)$    ▷ Restart manager observes if there has been an improvement in global optimum estimation
34:    end for
35:    if $RestartManager.shouldReset(Swarm)$ then
36:        $Swarm \leftarrow Initializer.initializeSwarm(F.bounds)$
37:    end if
38: end while
```

contributions made by each of the optimization behaviors to the improvement of the global best value. The optimization algorithms that brought the highest average improvement over the last k iterations to the estimated optimum are used relatively more frequently according to the designated probability vector.

2.2. External memory

Gathering samples (understood as point coordinates and function values at these points) is a key enhancement compared to the initial work on GAPSO [21]. The main idea is to take advantage of already gathered samples and use them in the model-based optimization enhancements. In order to store and retrieve samples in an efficient way M-GAPSO utilizes a multidimensional R-Tree index [7]. It allows a quick access to the desired subset of samples, such as surroundings of a selected point. Due to performance reasons the number of collected samples is limited. When the number of samples reaches the limit, the archive is cleared and the gathering process begins to insert new samples into the empty archive.

As can be observed in Algorithm 1, function samples are stored in the external memory ($SamplesArchive$ object) after each function evaluation (line 32). The stored samples are utilized in the context of building statistical models (lines 19 and 22). The process of selecting samples from the archive and constructing those models is explained in details in the following subsection.

2.3. Model-based optimization

Model-based enhancements (quadratic or polynomial models) are applied in order to support quick convergence to local optima. In both cases the same principles of particle's behavior are considered. At the beginning, the model $\hat{f}(\boldsymbol{x})$ is fitted using specified sample collection $\mathcal{S}$. Then, the optimum ($\hat{\boldsymbol{x}^*}$) of model's function ($\hat{f^*}$) is selected in relation to the assumed boundaries. Finally, the particle is moved to coordinates $\hat{\boldsymbol{x}^*}$ that match the estimated optimum.

2.3.1. Quadratic model

Quadratic model is fitted on a data set $\mathcal{S}_Q$ composed of k nearest samples (in the sense of the Euclidean metric) to a particle $\boldsymbol{x}_{best}$ location for which the quadratic behavior has been selected (see Fig. 1 as an example). Quadratic function-based approach fits the following model:

$$\hat{f}_Q(\boldsymbol{x}) = \sum_{d=1}^{dim} \left(a_d x_d^2 + b_d x_d \right) + c \tag{1}$$

where $\boldsymbol{x} = [x_1, \ldots, x_{dim}]$ is a vector of coordinates.

Ordinary Least Squares method is applied to obtain linear regression coefficients: $a_1, \ldots, a_{dim}$ and $b_1, \ldots, b_{dim}$. Finding $\hat{f}_Q^*$ (minimum of $\hat{f}_Q$) is equivalent to finding dim independent minima $\hat{f}_{Q_d}^*$ ($d \in 1, \ldots, dim$) of dim independent parabolas $\hat{f}_{Q_d}(\boldsymbol{x}) = a_d x_d^2 + b_d x_d$. Coordinates of the bounded parabola peak are computed in

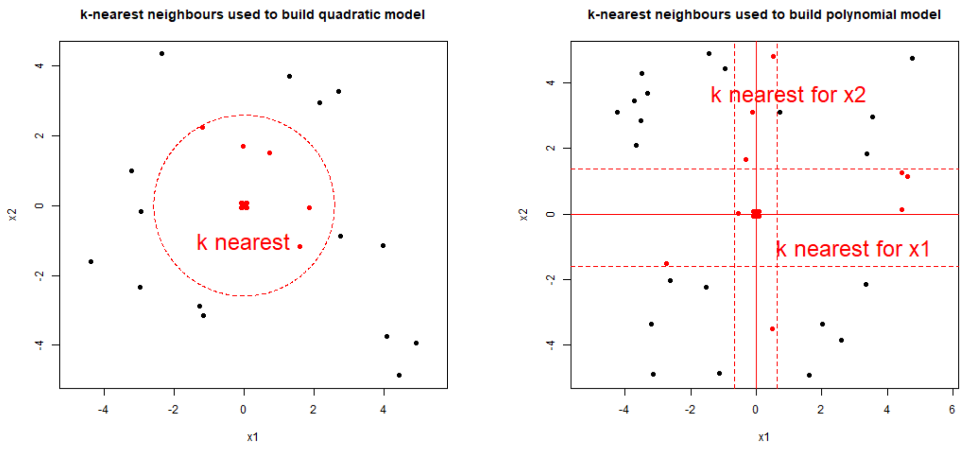

Figure 1. Comparison of samples data sets used for fitting quadratic and polynomial models.

the following way:

$$\hat{x}^*_{Q_d} = \begin{cases} -\dfrac{b_d}{2a_d}, & \text{if } a_d > 0 \wedge -\dfrac{b_d}{2a_d} \in [x_d^{lower}, x_d^{upper}] \\ x_d^{lower}, & \text{if } \hat{f}(x_d^{lower}) \le \hat{f}(x_d^{upper}) \\ x_d^{upper}, & \text{if } \hat{f}(x_d^{lower}) > \hat{f}(x_d^{upper}) \end{cases} \tag{2}$$

where x_d^{lower} and x_d^{upper} are lower and upper bounds of the function domain, respectively. Estimated optimum $\hat{\boldsymbol{x}}^*$ is a composition of dim independent peaks $\hat{\boldsymbol{x}}^*_d$:

$$\hat{\boldsymbol{x}}^* = [\hat{\boldsymbol{x}}^*_1, \ldots, \hat{\boldsymbol{x}}^*_{dim}] \tag{3}$$

2.3.2. Polynomial model

Polynomial model is fitted independently on dim data sets $\mathcal{S}_{P_d}$. Each of these data sets is composed of k samples closest to a line with coordinates fixed at the current location, except for dimension d for which the model is currently fitted. The differences between methods of gathering samples (quadratic vs. polynomial) are depicted in Fig. 1. Polynomial model function is an extension of quadratic one and can be expressed as follows:

$$\hat{f}_{P_d}(\boldsymbol{x}) = \sum_{i=1}^{p} a_{i,d} x_d^i + c_d \tag{4}$$

where p is a polynomial degree.

Similarly to the quadratic model, Ordinary Least Squares method is applied to obtain linear regression coefficients: $a_{i,d}, i \in \{1, \ldots, p\}, d \in \{1, \ldots, dim\}$. dim independent minima $\hat{f}^*_{P_d}, d \in \{1, \ldots, dim\}$ of dim independent polynomial functions $\hat{f}_{P_d}(\boldsymbol{x})$ are computed using grid search. For each dimension d, a space between x_d^{min} and x_d^{max} is divided into 1000 regular points, where x_d^{min} and x_d^{max} are respectively minimum and maximum coordinates in dimension d derived from the fitting data set $\mathcal{S}_{P_d}$. A value of $\hat{f}_{P_d}(\boldsymbol{x})$ in (4) is calculated in each of these 1000 points and the smallest one determines the optimal value $\hat{\boldsymbol{x}^*_d}$.

Finally, coordinates of optimum estimation $\hat{\boldsymbol{x}^*}$ are obtained in the same way as in the quadratic model.

3. Analysis of the model-based optimization enhancements

This section presents how model-based optimizers sample the function's search space. The goal is to observe conditions in which these optimizers are beneficial for the whole process, in order to utilize these observations in future improvements of the M-GAPSO algorithm.

3.1. Quadratic model sampling analysis

To analyze the effectiveness of the quadratic model-based optimizer, we have created contour plots of 2-dimensional instances of the selected benchmark functions. Figures 2-7 present locations of the function samples utilized to create the model (red dots) and the location indicated by the model as potential improvement (blue dot). Red square denotes the location of the function global optimum. The numbers in the plots present reference function values.

Introduction of quadratic model has been motivated by the possibility of achieving a quick convergence to local optima, while the population is still somewhat dispersed. As can be observed in Fig. 2 in certain cases this goal is achieved, while in others (Fig. 3) the model leads the search process away from the optimum. Currently, our working hypothesis is that this behavior could be improved through management of the samples set diversity in order to avoid numerical instabilities.

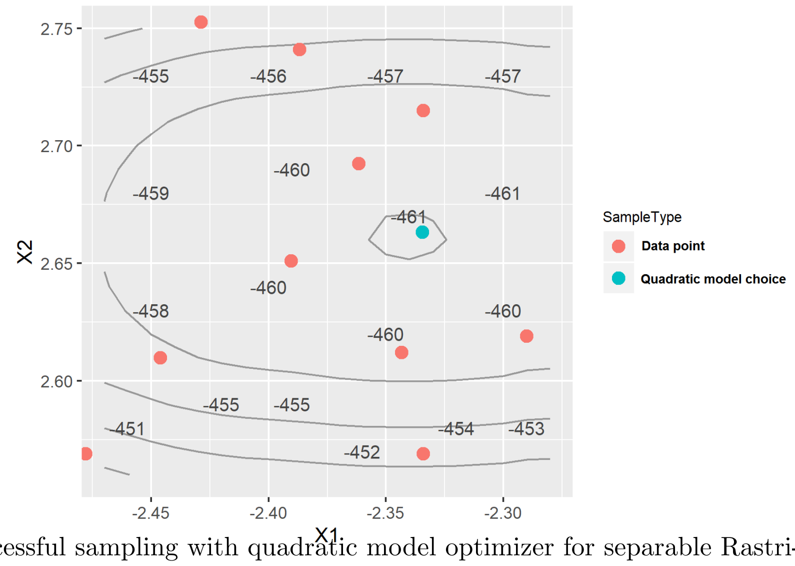

Figure 2. Successful sampling with quadratic model optimizer for separable Rastrigin's function (COCO BBOB f03).

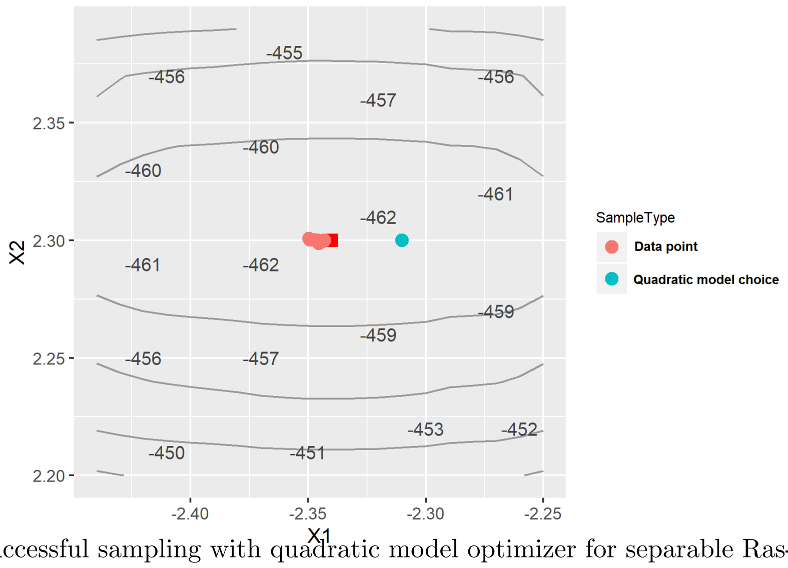

Figure 3. Unsuccessful sampling with quadratic model optimizer for separable Rastrigin's function (COCO BBOB f03).

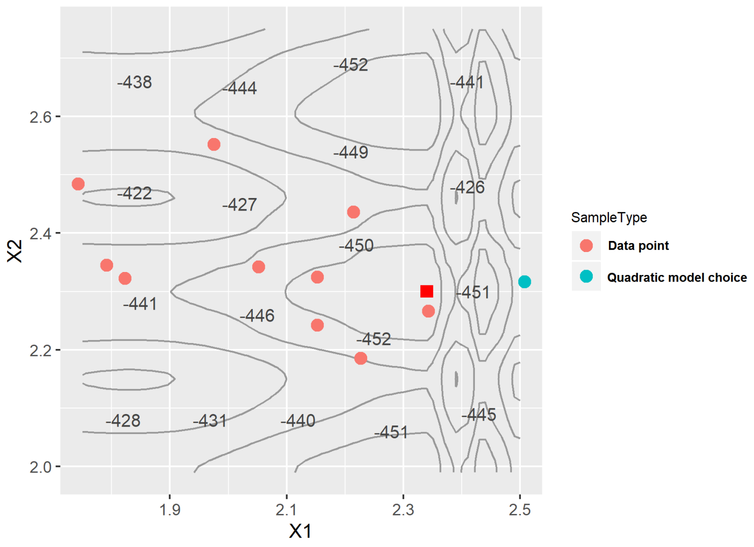

Figure 4. Quadratic model optimizer behavior for Büche-Rastrigin's function (COCO BBOB f04).

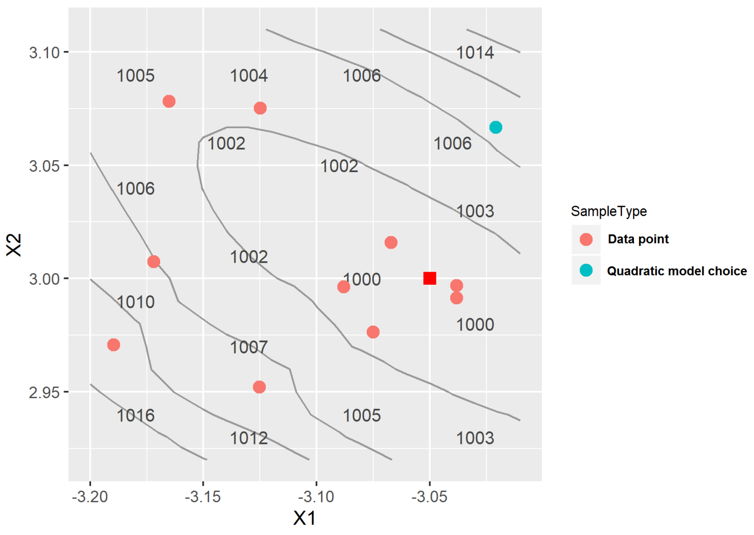

Figure 5. Quadratic model optimizer behavior for non-separable Rastrigin's function (COCO BBOB f15).

As expected, this simple quadratic model-based optimizer has not been able to guide the search process correctly in the case of asymmetrical shapes of the optima (Fig. 4) or non-separable functions (Fig. 5). While samples preselection might help in the case of asymmetrical optima, improving quality for non-separable function, without enlarging the model, could be achieved through some variation of Principal Components Analysis approach.

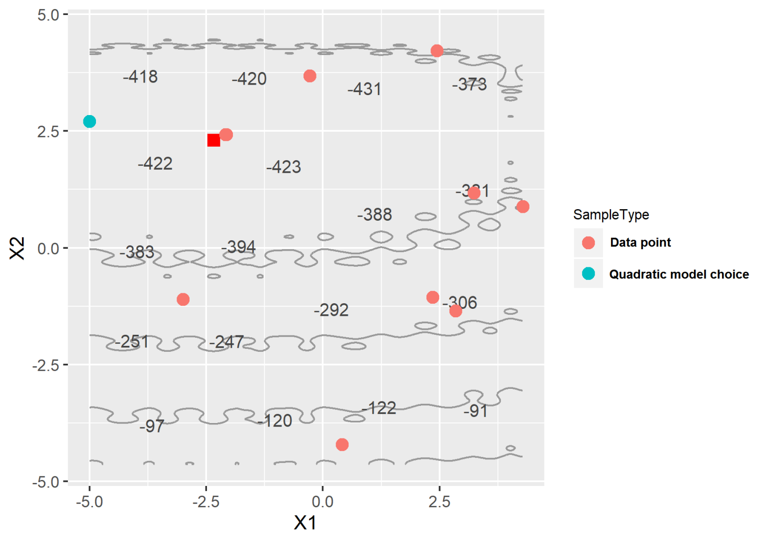

Figure 6. Quadratic model optimizer behavior with standard sample set size ($5 \cdot DIM$) in the initial iterations of optimization process for a separable Rastrigin's function (COCO BBOB f03).

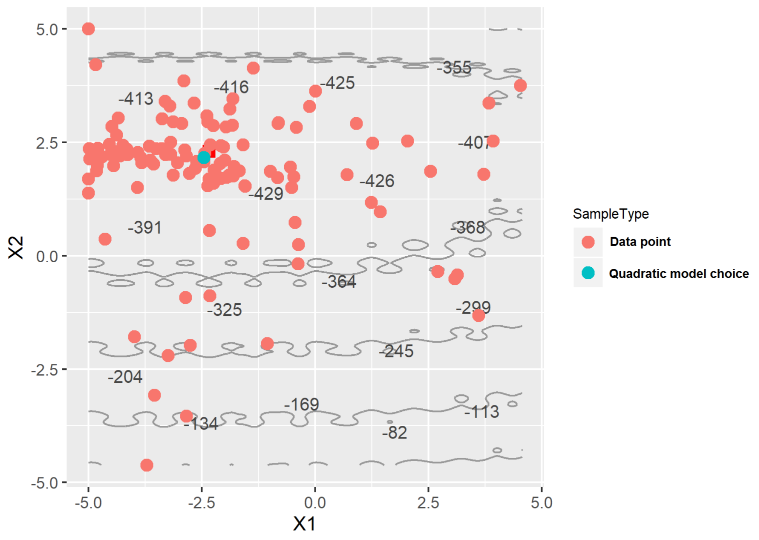

Figure 7. Quadratic model optimizer behavior with increased sample set size ($100 \cdot DIM$) in the initial iterations of optimization process for a separable Rastrigin's function (COCO BBOB f03).

Finally, quadratic model optimizer could be better utilized in initial phases of the optimization process for functions with the so-called general structure, simply by enlarging the set of samples on which the model is constructed. A comparison between sample set sizes of $5 \cdot DIM$ and $100 \cdot DIM$ is depicted in Figures 6 and 7.

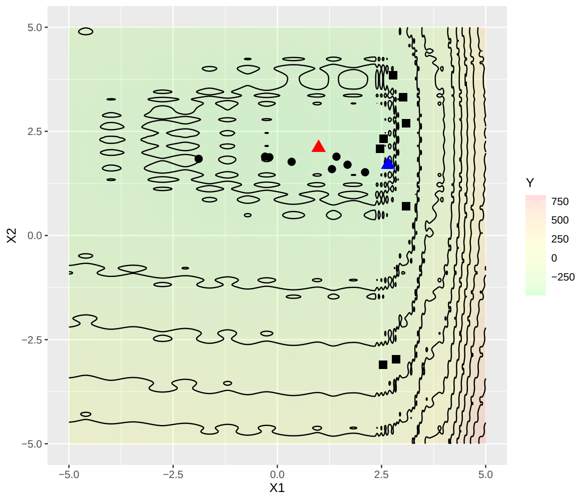

Figure 8. Example of polynomial model next sample calculation during Büche-Rastrigin's function (COCO BBOB f04) optimization run.

3.2. Estimated optimum trajectory of polynomial model

Since the polynomial model is fitted independently in each dimension, it is expected that it will perform satisfactorily on separable functions. Its behavior was empirically examined using Büche-Rastrigin's function (COCO BBOB f04) since we hypothesized that the polynomial model should perform better for an asymmetrical function, while quadratic model was unable to do so, due to its inherent properties. We presented example of its next sample calculation and trajectory of all evaluations made during the optimization process. The goal was to observe if model tends to choose promising points for the evaluation.

Figure 8 presents an example of the next sample calculation of polynomial model. The snapshot was made during 2D separable Büche-Rastrigin's function (COCO BBOB f04) optimization run and concerns the fifth polynomial model-based behavior call. The blue triangle represents the current particle position. Black squares represent gathered samples which are closest to x_1 coordinate. Black circles gathered samples which are closest to x_2 coordinate. The next sample position is marked with red triangle. Early phase of the optimization run explains why the samples are not strictly concentrated along lines parallel to the axis. Nonetheless, the model has designated a promising point that is significantly distant from a vertical dataset.

Figure 9 presents trajectory of all evaluations of polynomial model-based behavior during the separable function f4 (instance 1, 2D) optimization run. There are many evaluation points close to the perpendicular lines designated by optimum coordinates. This is a consequence of the polynomial model construction and the samples gathering method. The presented characteristics of making subsequent evaluations is suitable for separated functions, which is confirmed by the results presented in section 4.2.

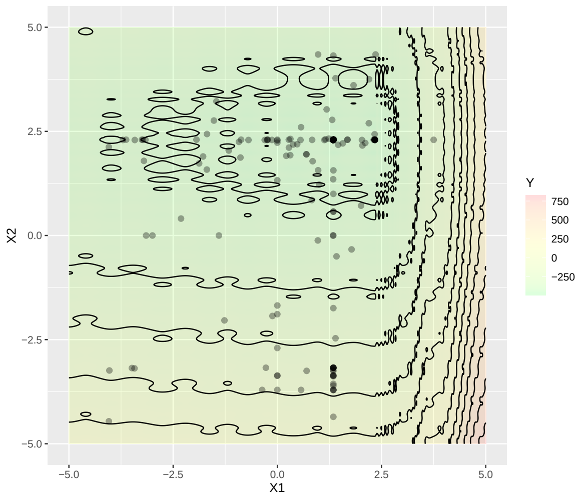

Figure 9. Example of polynomial model trajectory during Büche-Rastrigin's function (COCO BBOB f04) optimization run.

3.3. Quality of samples gathered with polynomial model behavior

A complementary experiment was carried out in which in each dimension k-nearest samples gathered through polynomial model-based behaviors were collected. The dispersion of all gathered samples was calculated independently for each dimension $d \in \{1, \ldots, dim\}$.

For each d, dim dispersion measures $disp_{d,d'}$ are obtain independently for each of dim dimensions $d' \in \{1, \ldots, dim\}$. The goal is to check how k-nearest samples in dimension d are dispersed through a line (hyperplane for dimensions greater than 2D) designated by dimension d and how close they are in other dimensions $d' \neq d$. The assumption is that the model will perform better if $disp_{d,d} >> disp_{d,d'}$ for $d' \neq d$.

$d' \setminus d$	1	2	3	4	5
1	0.488	0.145	0.166	0.167	0.176
2	0.149	0.594	0.159	0.159	0.172
3	0.159	0.146	0.463	0.166	0.179
4	0.175	0.165	0.178	0.416	0.185
5	0.137	0.132	0.135	0.137	0.304

Table 1. Dispersion (standard deviation) by dimension of samples gathered during f4 (instance 1, 5D) optimization run.

Let $\boldsymbol{x}^P = [x_1^P, \ldots, x_{dim}^P]$ be the current position of model-based behavior particle. The data set $\mathcal{S}_{P_d}$ (cf. Figure 1) contains k coordinate vectors $\boldsymbol{x_{i,d}} = [x_{i,1,d}, \ldots, x_{i,dim,d}]$, $i \in \{1, \ldots, k\}$. The dispersion $disp_{d,d'}$ is calculated as standard deviation of $(x_d^P - x_{i,d',d})$.

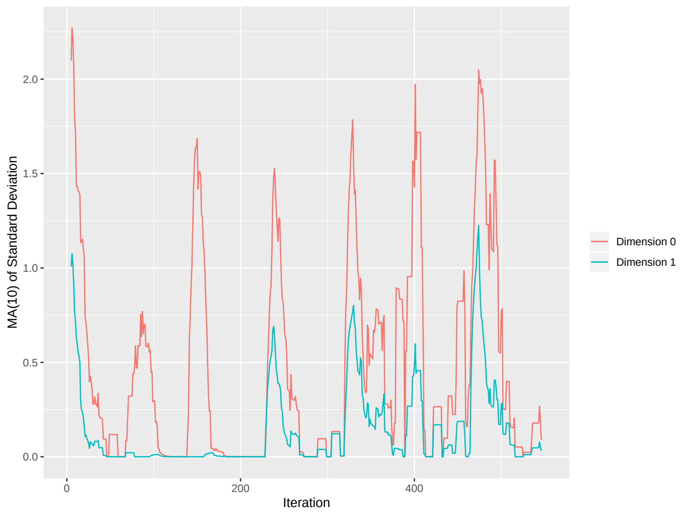

Figure 10. Dispersion by dimension of gathered samples in polynomial model during f4 (instance 1, 5D) optimization run.

Table 1 presents mean values of all $disp_{d,d'}$ obtained during f4 (instance 1, 5D) optimization run. As expected $disp_{d,d}$ is higher than $disp_{d,d'}$ for $d' \neq d$. Nonetheless, the differences are not significant.

Based on the results presented in Table 1, all subsequent values $disp_{d,d'}$ were stored. Figure 10 shows how were the moving averages of $disp_{1,1}$ and $disp_{1,2}$ changing during f4 (instance 1, 5D) optimization run. The repetitive rapid increases of dispersion are mainly caused by archive clearing after reaching the limit of stored samples. Another factors that cause sudden changes in dispersion value are algorithm restarts and cyclic reassigning of behaviors to each particle. Moreover, polynomial model-based behavior is susceptible to relatively extensive jumps within the function domain.

4. Experimental results

All experiments were conducted using COCO BBOB benchmark data set [9]. Sections 4.1 and 4.2 present the base experiments and Sections 4.3 and 4.4 are devoted to additional tests investigating how changing the key parameters of polynomial model-based behavior affects its performance.

4.1. Comparison with GAPSO and state-of-the art version of CMA-ES

Comparisons with GAPSO and state-of-the art CMA-ES[22] (*KL-BIPOP*) are shown in Fig. 11. Evaluation was made on 24 noiseless continuous functions. Two variants of M-GAPSO are included in this comparison: *PDLPr* and *PDr*. The first one includes model-based behaviors. The other one, does not, i.e. its behavior pool consists only of *PSO-2007* and *DE/best/1/bin* particles. Both variants do not utilize adaptation module. The comparison of *PDr* and *PDLPr* was made for 2, 3, 5, 10, 20 and 40 dimensions. Results of the GAPSO are available for 5 and 20 dimensions. The *KL-BIPOP* results are presented for all dimensions except 40. Parameter values of M-GAPSO *PDLPr* variant are presented in Table 2 in the appendix.

The enhancements implemented in M-GAPSO improved the performance over GAPSO for both 5 and 20 dimension experiments. Both variants are significantly better than GAPSO, specifically in the case of 5 dimensions. Compared to CMA-ES, M-GAPSO performs better for 2, 3 and 5 dimension experiments. For higher dimensions both M-GAPSO variants are slightly worse than CMA-ES. For 2,3 and 5 dimension experiments both M-GAPSO variants perform similarly. For higher dimensions *PDLPr* tends to outperform *PDr*. Moreover, in all experiments *PDLPr* distinctly exceeds *PDr* in the early phase of an optimization run which suggests that in the early phase model-based behaviors approximate function well enough to help other particles to move faster to more promising areas.

4.2. Comparison for separable functions

Considering the principle of model-based behaviors and analysis presented in section 2.3 we expected that the *PDLPr* variant should perform distinctly better than the *PDr* for separable functions. Complementary comparisons for both variants are presented in Figure 12. The experiment was conducted with $DIM \cdot 10^6$ optimization budget, for 2, 3, 5, 10, 20 and 40 dimensions using separable benchmark functions.

For all dimensions *PDLPr* performs distinctly better than *PDr*. For 2 and 3 dimensions the advantage is noticeable especially in the early stages of an optimization run. For higher dimensions *PDLPr* performs indisputably better during the whole optimization run. Furthermore, for 20 and 40 dimensions it performs relatively better for separable functions than for the remaining ones from the 24 function benchmark considered.

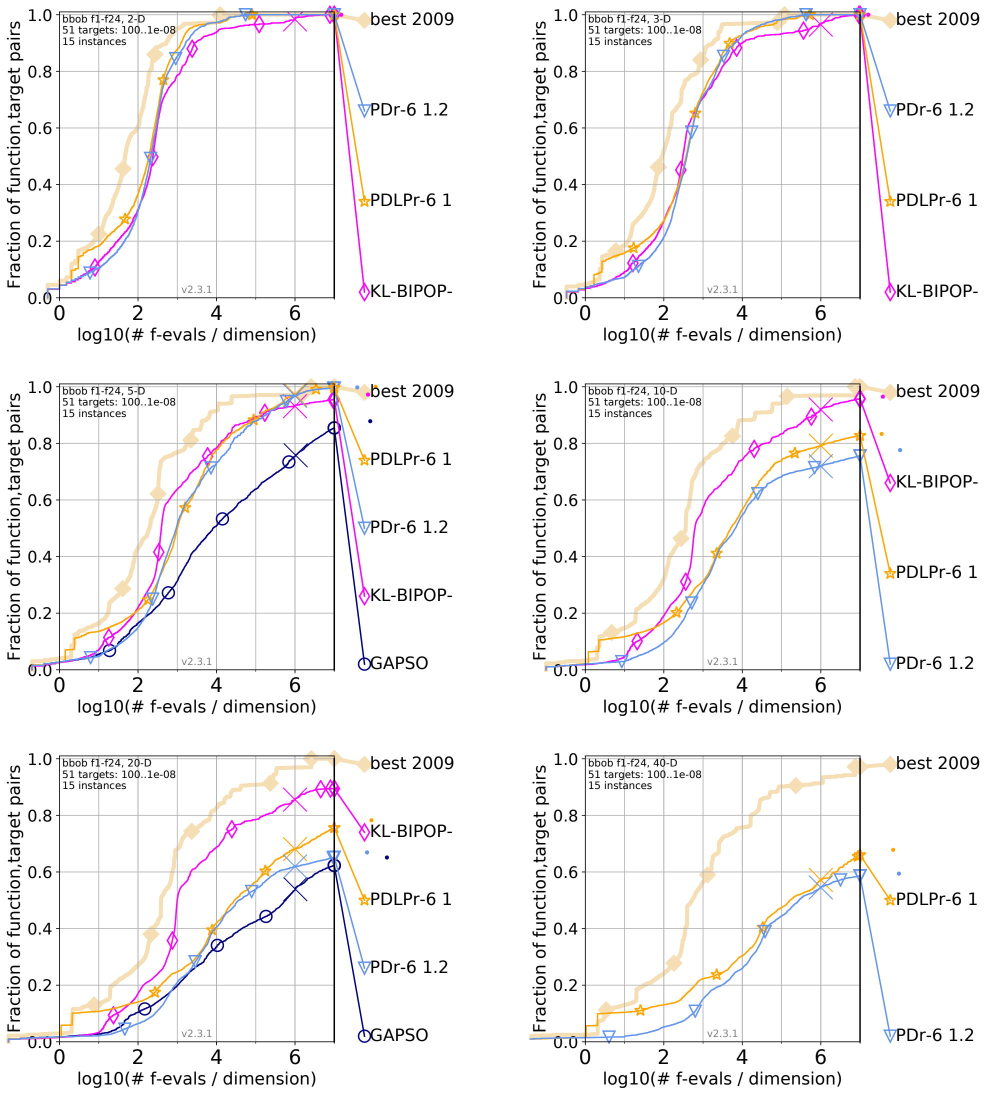

Figure 11. Results of M-GAPSO configurations with model-based optimization (PDLPr) and without model-based optimization (PDr) against GAPSO and state-of-the-art variant of CMA-ES, for $DIM * 10^6$ optimization budget.

4.3. Experiments with various polynomial degrees

In the experiments presented in Figs. 11 and 12 a constant degree (equal to 4) of polynomial model-based behavior was assumed. In Fig. 13 performance for various degrees of polynomial-based model is presented with $DIM \cdot 10^6$ optimization budget

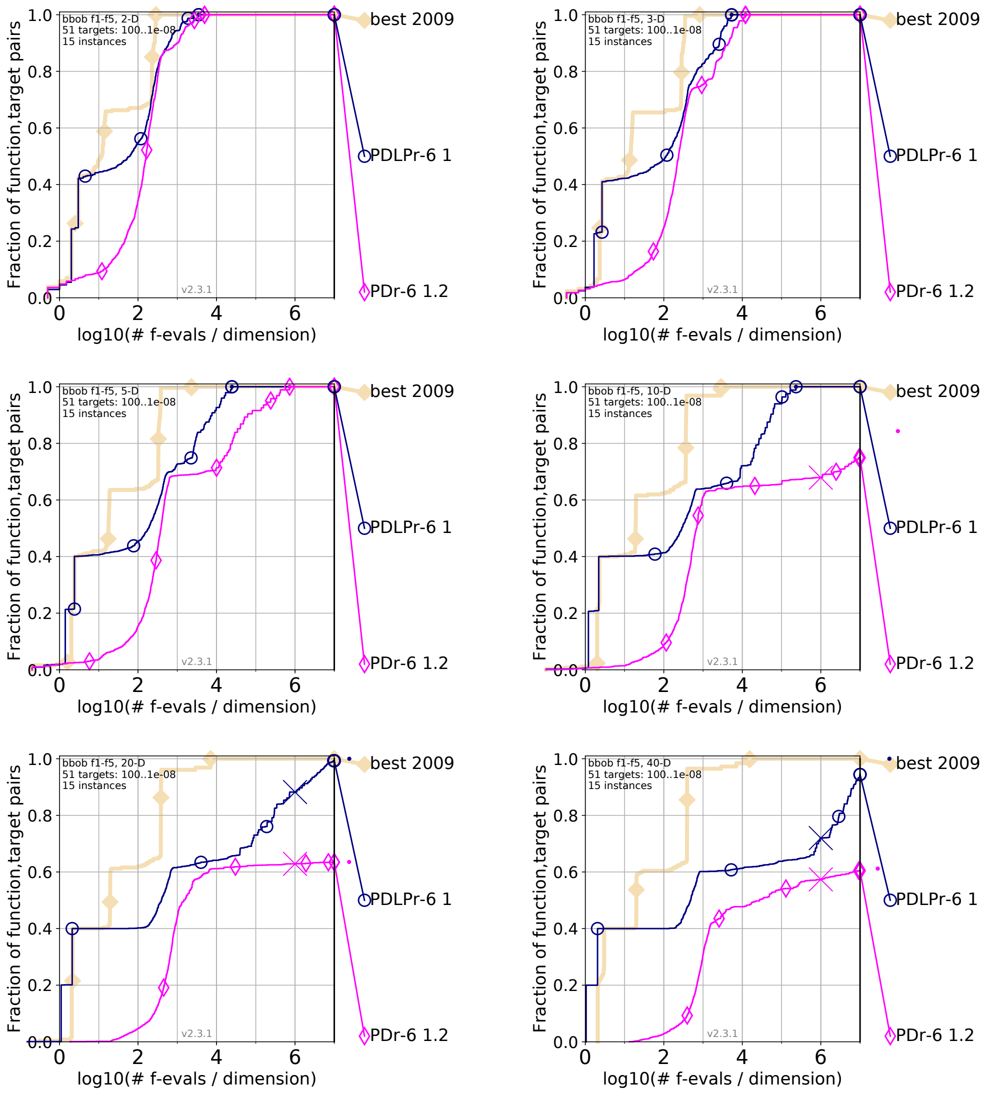

Figure 12. Results for separable functions of M-GAPSO configurations with model-based optimization (PDLPr) and without model-based optimization (PDr), for $DIM \cdot 10^6$ optimization budget.

on 2D, 3D, 5D and 10D separable benchmark functions. The following polynomial degrees were examined: 2, 4, 5, 6 and 10. Additionally, a variant with no polynomial model-based behavior was also included in the behavior pool.

For 2D experiment the performance of all variants was similar. As the number of dimensions increases, inclusion of polynomial model-based behavior in the pool re-

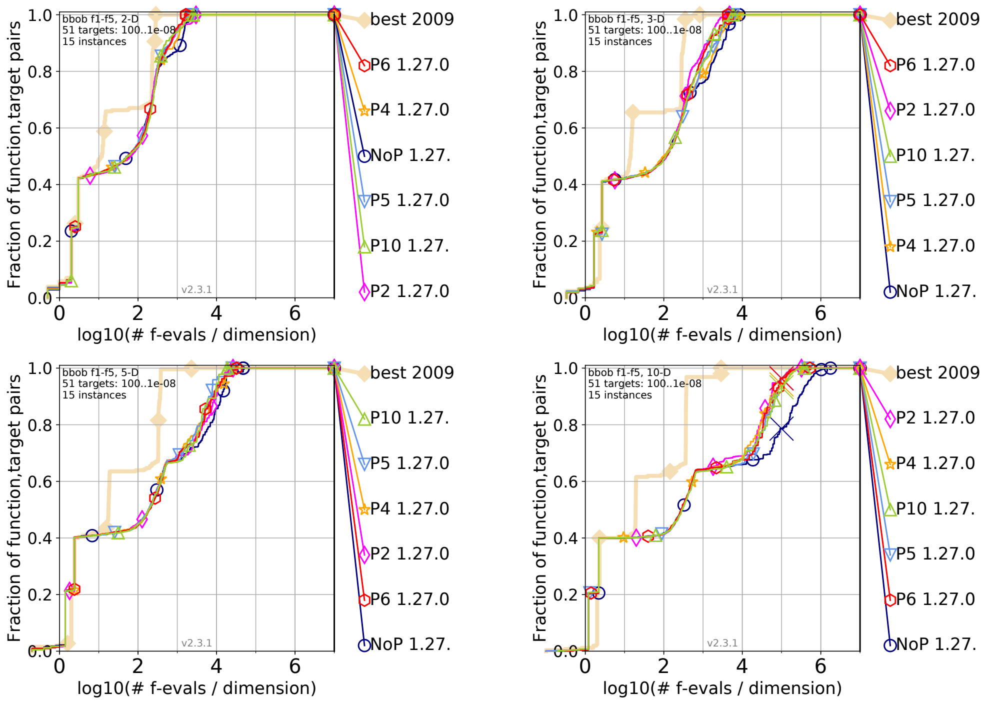

Figure 13. Results of M-GAPSO configurations for different degrees of the polynomial model on separable functions, for $DIM \cdot 10^6$ optimization budget.

sults in improved performance. M-GAPSO with no polynomial model-based behavior performed worst in all trials with dimension greater than 2. For 10D the lack of the polynomial model-based behavior causes significant performance decrease. None of the specific polynomial degrees performed clearly better or worse than the others. Adaptation of polynomial-based model degree is one of our future research directions.

4.4. Experiments with other samples gathering methods

A possibility of mixing polynomial and quadratic model-based behavior was also investigated. Mixing is understood as incorporating samples gathering method as in the quadratic model-based behavior into polynomial model-based behavior. Figure 14 presents performance of both gathering methods (cf. Figure 1) incorporated into polynomial model-based behavior. The experiment was conducted with $DIM \cdot 10^6$ optimization budget, in 2D, 3D, 5D and 10D using separable benchmark functions.

The results show that for lower dimensions (2D and 3D) there are no significant performance differences between both variants of gathering samples. The differences occur in 5D and 10D experiments. Therefore, the use of more advanced samples gathering algorithm within polynomial model-based behavior has proven to be a rea-

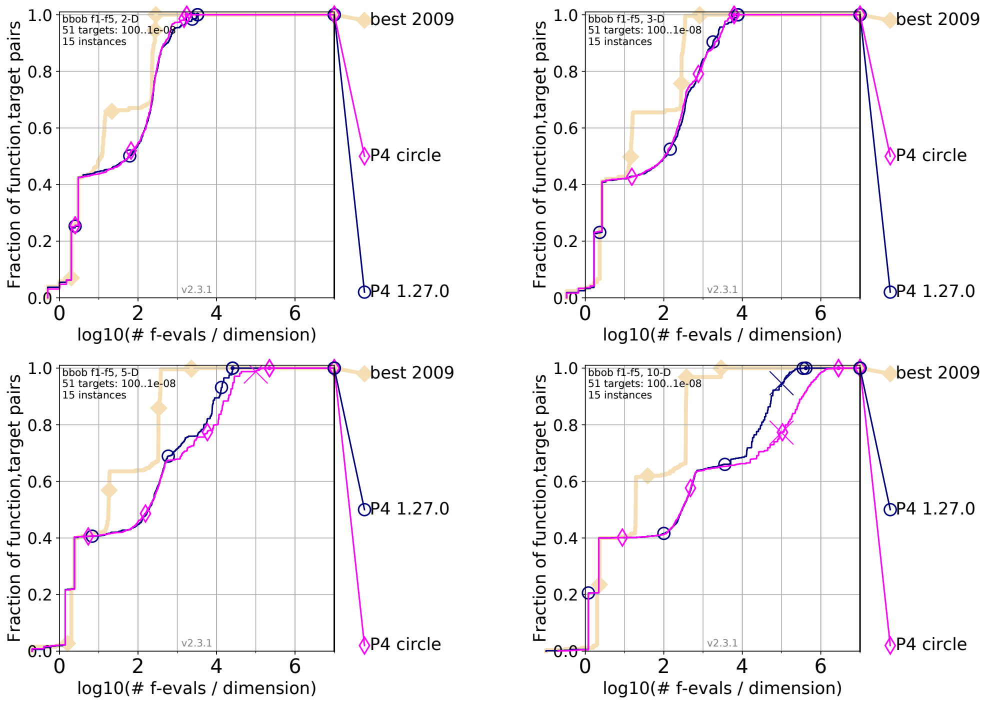

Figure 14. Results of M-GAPSO configurations for various samples gathering methods in polynomial model, for $DIM \cdot 10^6$ optimization budget.

sonable approach.

5. Summary

In this paper quadratic and polynomial model-based behaviors are introduced. Experimental evaluation confirms the reasonability of incorporating them into M-GAPSO. The analysis of samples gathered by quadratic and polynomial model-based behaviors reveals the strong and weak points of the proposed solution and points directions for further improvement of results. Presented models can be parameterized to act more globally, e.g. by gathering a wider range of samples.

It is important to note that various other models can also be incorporated into M-GAPSO. For instance, the RBF or Gaussian process may be included in the behavior pool. In the current version of the algorithm, model-based behavior designates the next sampling point regardless of the model quality. In future work we plan to select models depending on their fitness measure.

In summary, we believe that M-GAPSO results are promising, although there is still room for improvement of the method. In particular other local methods for handling samples gathered in external memory, as well as global modeling schemes can be considered as an alternative to the current M-GAPSO setting.

Appendix

Table 2. Settings of the M-GAPSO *PDLPr* variant.

Core M-GAPSO parameters	
Population size	$10D$
PSO behavior weight	1000
DE behavior weight	1000
Quadratic model–based behavior weight	1
Polynomial model–based behavior weight	1
Adaptation module	off
Optimization algorithms parameters	
PSO cognitive factor c_1	1.4
PSO social factor c_2	1.4
PSO velocity inertia factor ω	0.64
DE cross-over probability	0.9
DE mutation scaling factor F	0.0 - 1.4
Samples archive size	$2 \cdot 10^5$
Quadratic model nearest samples count	$5D$
Polynomial model degree	4
Polynomial model nearest samples count	$4D + 1$

References

[1] Bartz-Beielstein T. and Zaefferer M. Model-based methods for continuous and discrete global optimization. *Applied Soft Computing*, 55:154–167, 2017.

[2] Cai X., Qiu H., Gao L., Jiang C., and Shao X. An efficient surrogate-assisted particle swarm optimization algorithm for high-dimensional expensive problems. *Knowledge-Based Systems*, 184:104901, nov 2019.

[3] Chugh T., Rahat A., Volz V., and Zaefferer M. Towards Better Integration of Surrogate Models and Optimizers. In *High-Performance Simulation-Based Optimization*, pages 137–163. 2020.

[4] Chugh T., Sun C., Wang H., and Jin Y. *Surrogate-Assisted Evolutionary Optimization of Large Problems*, pages 165–187. Springer International Publishing, Cham, 2020.

[5] Clerc M. Standard particle swarm optimisation. 2012.

[6] Das S., Abraham A., and Konar A. Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In *Advances of computational intelligence in industrial systems*, pages 1–38. Springer, 2008.

[7] Guttman A. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.

[8] Hansen N. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, pages 75–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[9] Hansen N., Brockhoff D., Mersmann O., Tusar T., Tusar D., ElHara O. A., Sampaio P. R., Atamna A., Varelas K., Batu U., Nguyen D. M., Matzner F., and Auger A. COmparing Continuous Optimizers: numbbo/COCO on Github, 2019.

[10] Jin Y. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005.

[11] Kennedy J. and Eberhart R. C. Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks. IV*, pages 1942–1948, 1995.

[12] Kleijnen J. P. C. Simulation Optimization Through Regression or Kriging Metamodels. In *High-Performance Simulation-Based Optimization*, pages 115–135. 2020.

[13] Nepomuceno F. V. and Engelbrecht A. P. A Self-adaptive Heterogeneous PSO Inspired by Ants. In *International Conference on Swarm Intelligence*, pages 188–195. Springer, 2012.

[14] Okulewicz M. and Mańdziuk J. Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7895:547–558, 2013.

[15] Okulewicz M. and Mandziuk J. Two-phase multi-swarm PSO and the dynamic vehicle routing problem. In *2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI)*, pages 1–8, Orlando, Fl, USA, dec 2014. IEEE.

[16] Okulewicz M., Zaborski M., and Mańdziuk J. Generalized Self-Adapting Particle Swarm Optimization algorithm with archive of samples, 2020. preprint, available at: https://arxiv.org/pdf/2002.12485.

[17] Pitra Z., Bajer L., and Holeňa M. Doubly Trained Evolution Control for the Surrogate CMA-ES. In *Parallel Problem Solving from Nature – PPSN XIV*, pages 59–68. Springer International Publishing, Cham, 2016.

[18] Poaík P. and Klema V. JADE, an adaptive differential evolution algorithm, benchmarked on the BBOB noiseless testbed. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12*, page 197, New York, New York, USA, 2012. ACM Press.

[19] Poli R. An analysis of publications on particle swarm optimization applications. Technical report, Technical Report CSM-469, Department of Computer Science, University of Essex, 2007.

[20] Storn R. and Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[21] Uliński M., Żychowski A., Okulewicz M., Zaborski M., and Kordulewski H. Generalized Self-adapting Particle Swarm Optimization Algorithm. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3242, pages 29–40. Springer, Cham, 2018.

[22] Yamaguchi T. and Akimoto Y. Benchmarking the novel CMA-ES restart strategy using the search history on the BBOB noiseless testbed. In *GECCO '17 Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1780–1787, 2017.

[23] Zaborski M., Okulewicz M., and Mańdziuk J. Generalized Self-Adapting Particle Swarm Optimization algorithm with model-based optimization enhancements. In *2nd PP-RAI Conference (PPRAI-19)*, pages 380–383, Wrocław, Poland, 2019. Wrocław University of Science and Technology.

[24] Zhang J. and Sanderson A. C. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.