GP-DEMO: Differential Evolution for Multiobjective Optimization Based on Gaussian Process Models

Miha Mlakar, Dejan Petelin, Tea Tušar, Bogdan Filipič

Jožef Stefan Institute, and
Jožef Stefan International Postgraduate School
Jamova cesta 39, SI-1000 Ljubljana, Slovenia

Abstract

This paper proposes a novel surrogate-model-based multiobjective evolutionary algorithm called Differential Evolution for Multiobjective Optimization Based on Gaussian Process Models (GP-DEMO). The algorithm is based on the newly defined relations for comparing solutions under uncertainty. These relations minimize the possibility of wrongly performed comparisons of solutions due to inaccurate surrogate model approximations. The GP-DEMO algorithm was tested on several benchmark problems and two computationally expensive real-world problems. To be able to assess the results we compared them with another surrogate-model-based algorithm called Generational Evolution Control (GEC) and with the Differential Evolution for Multiobjective Optimization (DEMO). The quality of the results obtained with GP-DEMO was similar to the results obtained with DEMO, but with significantly fewer exactly evaluated solutions during the optimization process. The quality of the results obtained with GEC was lower compared to the quality gained with GP-DEMO and DEMO, mainly due to wrongly performed comparisons of the inaccurately approximated solutions.

Keywords: Multiple objective programming, Evolutionary algorithms, Differential evolution, Surrogate models, Gaussian process modeling, probable Pareto dominance

☆

Email addresses: miha.mlakar@ijs.si (Miha Mlakar), dejan.petelin@ijs.si (Dejan Petelin), tea.tusar@ijs.si (Tea Tušar), bogdan.filipic@ijs.si (Bogdan Filipič)

1. Introduction

Optimization problems are present in our everyday life and come in a variety of forms, e.g. the task to optimize certain properties of a system by correctly choosing the system parameters. Many of these optimization problems require the simultaneous optimization of multiple, often conflicting, criteria (or objectives). These problems are called multiobjective optimization problems. The solution to such problems is not a single point, but a family of points, known as the Pareto-optimal set. This set of solutions gives the decision maker an insight into the characteristics of the problem before a single solution is chosen.

One of the most effective ways to solve problems with more objectives is to use multiobjective evolutionary algorithms (MOEAs). MOEAs are population-based algorithms that draw inspiration from optimization processes that occur in nature. During the optimization process, in order to find a Pareto-optimal set, a lot of different solutions have to be assessed (evaluated). If these solution evaluations are computationally expensive, the whole optimization process can take a lot of time.

In order to obtain the results of such an optimization problem more quickly, we can use surrogate models in the optimization process to approximate the objective functions of the problem. To evaluate a solution, instead of using a time-consuming exact evaluation, a solution can be approximated with the surrogate model. Since one solution approximation is (much) faster, the whole optimization process can be accelerated. However, note that the time needed to create and update the surrogate models during the optimization process has to be considered and included in the whole duration of the optimization process. So, in the case where the exact solution evaluations are quick, it can happen that the surrogate-model-based optimization takes longer than the optimization without surrogates.

In surrogate-model-based multiobjective optimization, approximated values are often inappropriately used in the solution comparison. As a consequence, exactly evaluated good solutions can be discarded from the population because they appear to be dominated by the inaccurate and over-optimistic approximations. This can slow the optimization process or even prevent the algorithm from finding the best solutions.

Some surrogate models provide a distribution, from which the approximated value and also the confidence interval of the approximation can be calculated. Using this confidence interval, we define new dominance relations that take into account this uncertainty and propose a new concept for comparing solutions under uncertainty that requires exact evaluations only in cases where more certainty is needed. This minimizes the mistakes made in comparisons of inaccurately approximated solutions.

Based on this concept we propose a new surrogate-model-based multiobjective evolutionary algorithm, called Differential Evolution for Multiobjective Optimization Based on Gaussian Process Modeling (GP-DEMO). This algorithm is an extension of the Differential Evolution for Multiobjective Optimization (DEMO) algorithm [1], which uses differential evolution to effectively solve numerical multiobjective optimization problems. In addition, DEMO also emphasizes the variation operators and compared to for instance hypervolume-based search is comparably cheap in terms of computational effort. In the GP-DEMO, Gaussian Process (GP) modeling is employed to find approximate solution values together with their confidence intervals. Then, instead of comparing the solutions using the Pareto dominance relation, GP-DEMO uses the new uncertainty-based dominance relations, requiring exact evaluations of solutions as needed. The efficiency of GP-DEMO is assessed on several benchmark and two real-world optimization problems.

The structure of this paper is as follows. In Section 2, we overview the work done in the field of surrogate-model-based optimization, especially in multiobjective optimization. In Section 3, we describe the Gaussian Process modeling that is used to build the surrogate models in GP-DEMO. Then, in Section 4, we describe the new relations and methods for comparing solutions (presented with and without uncertainty). The GP-DEMO algorithm is presented in Section 5. In Section 6, we test and compare GP-DEMO with two other algorithms on benchmark and real-world multiobjective optimization problems. Finally, Section 7 concludes the paper with a summary of the work done and our ideas for future work.

2. Related Work

In the literature the term surrogate model (sometimes also meta-model) based optimization is used where, during the optimization processes, some solutions are not evaluated with the original objective function, but are approximated using a model of this function. Different modeling methods are used to build the surrogate models. For single and multiobjective optimization similar methods are used. These methods typically return only one approximated value, which is why in multiobjective problems several models have to be used, so that every model approximates one objective. Some of the most commonly used methods are the Response Surface Method [2], Radial Basis Function [3], Neural Network [4], Kriging [5] and Gaussian Process Modeling [6, 7, 8].

In single-objective optimization, the usage of surrogate models is well established and has proven to be successful. In the literature many different algorithms and various modeling techniques are used to solve benchmark and real-world problems [9, 10]. The results typically show that the surrogate-model-based optimization in comparison with optimization without surrogates provides comparable results in fewer objective function evaluations [11, 12]. The use of differential evolution in combination with surrogate models is mentioned in [9]. The authors presented an algorithm based on differential evolution that generates multiple offspring for each parent and chooses the promising one based on the confidence and the approximation of the current surrogate model.

In the field of surrogate-model-based multiobjective optimization, where the result is not just one solution but a non-dominated front of solutions, the problem of finding these solutions is even more challenging. There are many approaches that differ in terms of which solutions are approximated and how they use the approximations. Surrogate models can aim at either a global approximation of the objective function, or a local one, focusing on the neighborhood of the best current individuals. In [12], the authors used a combination of local and global surrogate models for solving optimization problem of Aerodynamic Shape Design.

Within surrogate-model-based optimization algorithms a mechanism is needed to

find a balance between the exact and approximate evaluations. In evolutionary algorithms this mechanism is called evolution control [13] and can be either fixed or adaptive.

In fixed evolution control, the surrogate model is trained from previously exactly evaluated solutions and then used directly instead of expensive objective function evaluations. In this approach the number of exact function evaluations that will be performed during the optimization is known in advance. Fixed evolution control can be further divided into generation-based control, where in some generations all solutions are approximated and in the others they are exactly evaluated [14], and individual based control, where in every generation some (usually the best) solutions are exactly evaluated and others approximated [15].

In adaptive evolution control, the number of exactly evaluated solutions is not known in advance, but depends on the accuracy of the model for the given problem. Adaptive evolution control can be used in one of two ways: as a part of a memetic search or to pre-select the promising individuals which are then exactly evaluated [16].

In a memetic algorithm, an additional algorithm (e.g., a gradient-based or an evolutionary algorithm) is used to find the optimal solutions using the surrogate model. Once this optimum is found, the best solutions are exactly evaluated and used for updating the model. In [17], aggregated surrogate models are used in a memetic algorithm. The model is based on the distance to the currently known, non-dominated set and is used to find new, non-dominated individuals using local search. In memetic algorithms, especially if the surrogate model is not very accurate, a local optimum is often found instead of the global optimum.

In the case of pre-selecting the promising individuals, the surrogate model is used to find the promising or drop the low-quality individuals even before they are exactly evaluated, thus reducing the number of exact evaluations. For example, OEGADO [18] creates a surrogate model for each of the objectives. The best solutions in every objective get also approximated on other objectives, which helps with finding trade-off individuals. The best individuals are then exactly evaluated and used to update the models. ParEGO [19] uses the weighted sum of the objective functions to perform a local search. The weights are generated randomly for each iteration. When a different model is used for each of the functions, the conversion from the multiobjective problem

to the single-objective one has to be performed (or a multiobjective optimizer has to be used on the models). Moreover, if there are more models, their errors can add up, as well as the time needed to train the models.

Surrogate models are also used to rank and filter out offspring according to Pareto-related indicators like the hypervolume [20], or a weighted sum of the objectives [21]. The problem with the methods that use hypervolume as a way of finding promising solutions is the calculation time needed to calculate the hypervolume, especially on many objectives. Another possibility is described in [22], where the authors present an algorithm that calculates only non-dominated solutions or solutions that can, because of variance, become non-dominated.

Some surrogate models, in addition to the approximation value, also return the certainty of the prediction. The use of this confidence information can help to increase the prediction accuracy of the surrogate model. In [10], the authors use confidence information to guide the search towards less explored regions in the search space. The confidence of the prediction with the approximated value can be used to calculate the criterion of expected improvement. Approaches to applying this criterion are analysed in [23], while an algorithm using this criterion to decide which solutions should be exactly evaluated is presented in [24].

During the comparison of solutions in the surrogate-model-based optimization, it can happen that an incorrectly approximated solution is presented as the better of two compared solutions. As multiobjective optimization algorithms usually discard dominated solutions, a good, exactly evaluated solution might be lost in such a case. Similarly, if a good solution is incorrectly approximated as worse, this solution is discarded.

To prevent these unwanted effects, we propose a new concept for comparing solutions under uncertainty, where in addition to the approximated value of a solution, its variance is considered. In [25], the authors tackle a noisy optimization problem with an algorithm that compares the solutions with uncertainty and, if necessary, performs additional evaluations of the same solution to minimize the uncertainty and, if possible, decide which solution is better. A theoretical presentation of the solution comparison under uncertainty was presented in [26] for optimization problems where the uncertainty of the solutions cannot be reduced by the sampling methods. The authors suggest

a strong Pareto dominance relation in cases when the dominance status can be determined, and weak Pareto dominance relation when, because of uncertainty, the domination status could not be determined. In this case the expected values for every solution are assumed and these values are then compared. In [27], a partial order approach is suggested to enable the comparison of solutions presented with confidence intervals. This approach does not differentiate between the cases where the upper border of the first interval dominates the lover border of the second interval and the cases where some part of intervals overlap. Very similar approach to handle solutions presented with intervals, called imprecise Pareto relations, was presented in [28]. In [29], the authors define bounding boxes to represent the solutions with confidence intervals. But the comparison of solutions is again simplified to the rejection of individuals with a small probability to be competitive, or to the exact evaluation of solutions, with a high probability to be better. In our paper, we adjust the comparison of solutions for surrogate-model-based multiobjective optimization and apply this comparison to the GP-DEMO algorithm to ensure that the best solutions are preserved in the optimization process.

3. Gaussian Process Models

The Gaussian process (GP) models are probabilistic, non-parametric, models based on the principles of Bayesian probability, which can be used for both regression and classification problems. The name GP models refers to the assumption that a prior on the function to be modeled is a stochastic process with a normal distribution, i.e., a Gaussian process (GP).

The GP regression is more or less identical to the Kriging method [30], which is widely used in the field of geostatistics. As a geostatistical method, it is mostly used for two- and three-dimensional input spaces and tends to ignore any probabilistic interpretations [31, 5]. Since the introduction of GPs in supervised learning [32], GP models have been used for modeling in various fields, e.g., biological systems [33, 34], environmental systems [35], chemical engineering [36] and many others.

The GP models differ from most of the other black-box identification approaches in that they do not try to approximate the modeled system by fitting the parameters of

the selected basis functions, but rather by searching for relationships among the measured data. The output of GP models is a normal distribution, expressed in terms of the mean and the variance. The mean value represents the most likely output and the variance can be interpreted as a measure of its confidence. The obtained variance, which depends on the amount and the quality of the available training data, provides important information when it comes to distinguishing GP models from other computational intelligence methods.

As GP models are, due to their probabilistic nature, suitable for interpolation, i.e., when data is missing, and in addition to the mean value also provide variance, they were already used in stochastic optimizations with surrogate models [37].

3.1. Gaussian Process Modeling

A GP is a collection of random variables that have a joint multivariate Gaussian distribution. Assuming a relationship of the form $y = f(\mathbf{x})$ between input $\mathbf{x}$ and output y, we have $y_1, \ldots, y_N \sim \mathcal{N}(0, \mathbf{K})$, where $\mathbf{K}_{pq} = Cov(y_p, y_q) = C(\mathbf{x}_p, \mathbf{x}_q)$ gives the covariance between the output points corresponding to the input points $\mathbf{x}_p$ and $\mathbf{x}_q$. Thus, the mean $\mu(\mathbf{x})$ and the covariance function $C(\mathbf{x}_p, \mathbf{x}_q)$ fully specify the GP.

The value of the covariance function $C(\mathbf{x}_p, \mathbf{x}_q)$ expresses the correlation between the individual outputs $f(\mathbf{x}_p)$ and $f(\mathbf{x}_q)$ with respect to inputs $\mathbf{x}_p$ and $\mathbf{x}_q$. It should be noted that the covariance function $C(\cdot, \cdot)$ can be any function that generates a positive semi-definite covariance matrix.

A commonly used covariance function is a composition of the square exponential covariance function with "automatic relevance determination" (ARD) hyperparameters [8] and the constant covariance function assuming white noise:

$$C(\mathbf{x}_p, \mathbf{x}_q) = v_1 \exp\left[-\frac{1}{2}\sum_{d=1}^{D} w_d (x_{dp} - x_{dq})^2\right] + \delta_{pq} v_0, \tag{1}$$

where w_d, v_1 and v_0 are the hyperparameters of the covariance function, D is the input dimension, and $\delta_{pq} = 1$ if $p = q$ and 0 otherwise. Other forms and combinations of covariance functions suitable for various applications can be found in [6]. The hyperparameters can be written as a vector $\mathbf{\Theta} = [w_1, \ldots, w_D, v_1, v_0]^T$. The hyperparameters

w_d indicate the importance of individual inputs. If w_d is zero or near zero, it means the inputs in dimension d contain little information and could possibly be neglected.

To accurately reflect the correlations presented in the training data, the hyperparameter values of the covariance function need to be optimized. Due to the probabilistic nature of the GP models, instead of minimizing the model error, the probability of the model is maximized.

Consider a set of N D-dimensional input vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$ and a vector of output data $\mathbf{y} = [y_1, y_2, \ldots, y_N]$. Based on the data $(\mathbf{X}, \mathbf{y})$, and given a new input vector $\mathbf{x}^*$, we wish to find the predictive distribution of the corresponding output y^*. From the training set $\mathbf{X}$, a covariance matrix $\mathbf{K}$ of size $N \times N$ is determined. The overall problem of learning unknown parameters from data corresponds to the predictive distribution $p(y^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*)$ of the new target y, given the training data $(\mathbf{y}, \mathbf{X})$ and a new input $\mathbf{x}^*$. In order to calculate this posterior distribution, a prior distribution over the hyperparameters $p(\mathbf{\Theta}|\mathbf{y}, \mathbf{X})$ can first be defined, followed by the integration of the model over the hyperparameters

$$p(y^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*) = \int p(y^*|\mathbf{\Theta}, \mathbf{y}, \mathbf{X}, \mathbf{x}^*) p(\mathbf{\Theta}|\mathbf{y}, \mathbf{X}) d\mathbf{\Theta}. \tag{2}$$

The computation of such integrals can be difficult due to the intractable nature of the non-linear functions, therefore, the general practice for estimating hyperparameter values is the maximum-likelihood estimation, i.e., minimizing the following negative log-likelihood function:

$$\mathcal{L}(\mathbf{\Theta}) = -\frac{1}{2} \log(\mid \mathbf{K} \mid) - \frac{1}{2} \mathbf{y}^T \mathbf{K}^{-1} \mathbf{y} - \frac{N}{2} \log(2\pi). \tag{3}$$

GP models can be easily utilized for regression calculation. Based on the training set $\mathbf{X}$, a covariance matrix $\mathbf{K}$ of size $N \times N$ is calculated. The aim is to find the distribution of the corresponding output y^* for some new input vector $\mathbf{x}^* = [x_1(N+1), x_2(N+1), \ldots, x_D(N+1)]$.

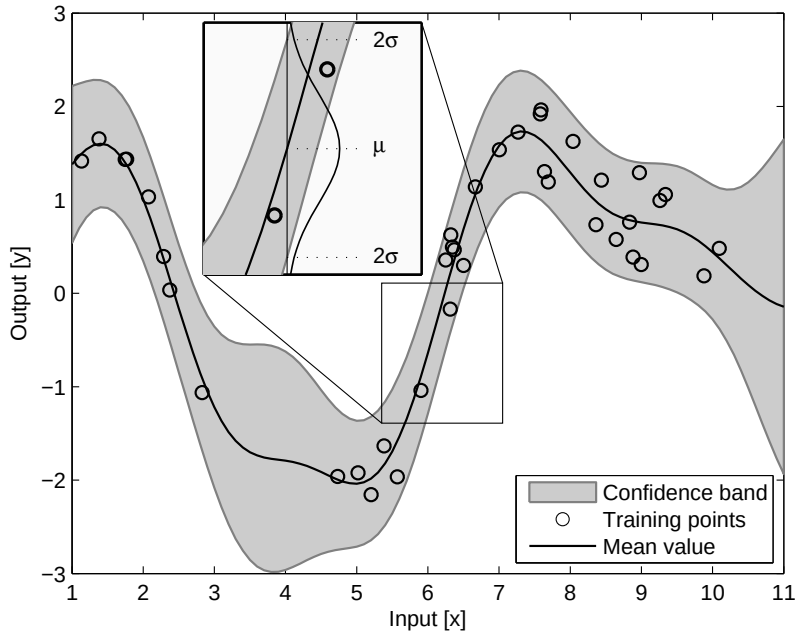
The predictive distribution of the output for a new test input has a normal probability distribution with a mean and variance

$$\mu(y^*) = \mathbf{k}(\mathbf{x}^*)^T \mathbf{K}^{-1} \mathbf{y}, \tag{4}$$

$$\sigma^2(y^*) = \kappa(\mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}^*), \tag{5}$$

where $\mathbf{k}(\mathbf{x}^*) = [C(\mathbf{x}_1, \mathbf{x}^*), \ldots, C(\mathbf{x}_N, \mathbf{x}^*)]^T$ is the $N \times 1$ vector of covariances between the test and the training cases, and $\kappa(x^*) = C(\mathbf{x}^*, \mathbf{x}^*)$ is the covariance between the test input itself.

As can be seen from (5) the GP model, in addition to a mean value, also provides information about the confidence of prediction using the variance. Usually, the confidence of the prediction is depicted with a 2σ interval, which corresponds to about 95% of the confidence interval. Considering the confidence intervals of all predictions, we obtain a confidence band, shown in grey in the example in Fig. 1. It highlights the areas of the input space where the prediction quality is poor, due to the lack of data or noisy data, by indicating a wider confidence band around the predicted mean.



Slika 1: Modeling with GP models: in addition to the mean value (prediction), we obtain a 95% confidence band for the underlying function f (shown in grey).

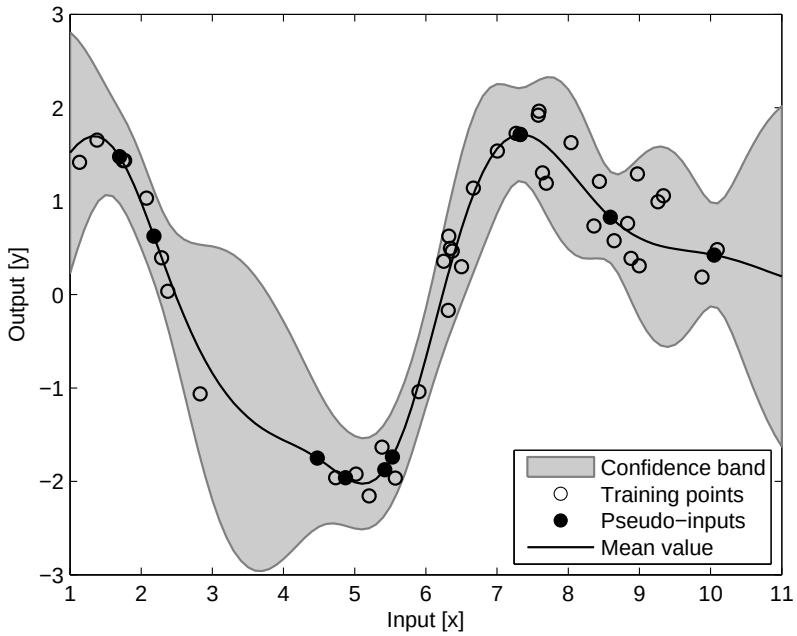
3.2. Sparse approximation

A noticeable drawback of "full" GP modeling is the computation load that increases with the third power of the amount of input data due to the calculation of the inverse of the covariance matrix. This computational complexity restricts the amount of training data to, at most, a few thousand cases. As multiobjective evolutionary algorithms usually require more than a few thousand evaluations, which serve as the training data for GP models, "full" GP modeling does not seem to be viable for our needs.

To overcome the computational-limitation issues and consequently to make the method viable for large-scale dataset applications, such as stochastic optimization, numerous authors have suggested various sparse approximations. A survey of such methods can be found in [38, 39]. A common property of all these sparse-approximation methods is that they try to retain the bulk of the information contained in the full training dataset, but reduce the size of the covariance matrix so as to facilitate a less computationally demanding implementation of the GP model. Usually, this subset of the training data is called the active set. The computational complexity of such algorithms is $O(NM^2)$, where N is the amount of training data and M is the size of the active set.

We decided to use a state-of-the-art, sparse-approximation method named Sparse Gaussian Processes using Pseudo-inputs (SPGP) [40], which is in general determined as a fully independent training conditional approximation [39, 38]. The idea of this method is that instead of selecting a subset of the training data, it rather optimizes the locations of M pseudo-inputs, as this seems to be easier to solve than the discrete subset selection problem. The pseudo-input locations are optimized based on the covariances between the training data points and the pseudo-inputs.

It should be noted that due to fewer data points being incorporated into the model (covariance matrix) and their arbitrary locations, the posterior of the SPGP model, especially the variance, can be somewhat different to the posterior of a "full" GP model. Such a case is illustrated in Fig. 2. It is clear that the mean value is very similar to the mean value obtained with the "full" GP model (Fig. 1), but the variance (95% confidence interval) is distinctly different from the variance obtained by the "full" GP model (Fig. 1).



Slika 2: Modeling with SPGP models: pseudo-inputs (dots) are arbitrarily located, i.e., not a subset of the training data points (circles).

4. Relations in Multiobjective Optimization

A multiobjective optimization problem (MOP) consists of finding the minimum of the function:

$$f : X \to Z$$
$$f : (x_1, ..., x_n) \mapsto (f_1(x_1, ..., x_n), ..., f_m(x_1, ..., x_n)),$$

where n is the number of variables and m is the number of objectives, and where each solution $x = (x_1, ..., x_n) \in X$ is called a *decision vector*, while the corresponding element $z = f(x) \in Z$ is an *objective vector*. We use this problem formulation to describe the relations between the the solutions presented without and with uncertainty.

4.1. Relations without uncertainty

First, consider the case where all solutions of a MOP are exactly evaluated. As a rule, two solutions can be in exactly one of the following four relations.

Definition 4.1 (**Pareto dominance**). *The objective vector* z dominates *the objective vector* w, $z \prec w$, *iff* $z_j \leq w_j$ *for all* $j \in \{1, ..., m\}$ *and* $z_k < w_k$ *for at least one* $k \in \{1, ..., m\}$.

Definition 4.2 (**weak Pareto dominance**). *The objective vector* z weakly dominates *the objective vector* w, $z \preceq w$, *iff* $z_j \leq w_j$ *for all* $j \in \{1, ..., m\}$.

Definition 4.3 (**strict Pareto dominance**). *The objective vector* z strictly dominates *the objective vector* w, $z \prec\prec w$, *iff* $z_j < w_j$ *for all* $j \in \{1, ..., m\}$.

When $z = f(x), w = f(y)$ and z (weakly or strictly) dominates w, we say that solution x (weakly or strictly) dominates solution y. In other words, solution x is equal to or better than solution y. The weak Pareto dominance is a natural generalization of the $\leq$ relation, and the strict Pareto dominance is the natural generalization of the $<$ relation.

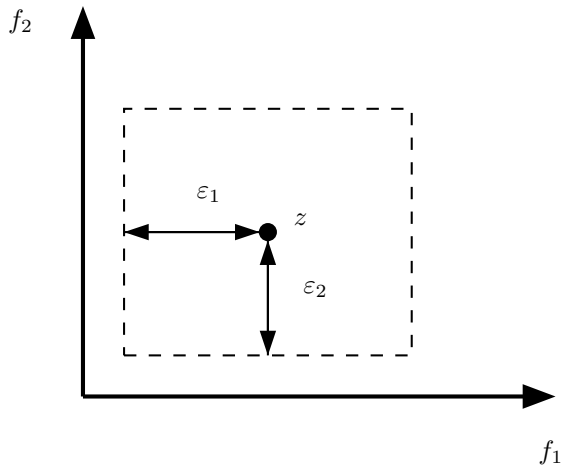
Definition 4.4 (**incomparability**). *The objective vectors* z *and* w *are* incomparable, $z||w$, *iff* $z \not\preceq w$ *and* $w \not\preceq z$.

Again, if z and w are incomparable, solutions x and y are incomparable.

4.2. Relations under uncertainty

Now consider the case where the objective values of the solutions are approximated by a surrogate model, e.g., a GP model, that is also able to provide the confidence interval for each approximated value. In such a case, the standard relations described previously are not suitable, but need to be adapted to accommodate the uncertainty. Every solution x is represented with a vector of approximated objective values $z = (z_1, z_2, ..., z_m)$ and a vector of confidence intervals in each objective $\varepsilon = (\varepsilon_1, \varepsilon_2, ..., \varepsilon_m)$. In order to be able to compare the solutions represented in this way, the relations between the solutions under uncertainty are defined on the *bounding boxes* of their objective values. From the vectors of the approximated values and the confidence intervals the *bounding box* of an objective vector z is designed as (Fig. 3):

$$\mathrm{BB(z, \varepsilon)} = [z_1 - \varepsilon_1, z_1 + \varepsilon_1] \times [z_2 - \varepsilon_2, z_2 - \varepsilon_2] \times ... \times [z_m - \varepsilon_m, z_m - \varepsilon_m].$$



Slika 3: The bounding box of an objective vector

Definition 4.5 (probable Pareto dominance). *The bounding box* $\mathrm{BB(z}, \varepsilon)$ *probably dominates the bounding box* $\mathrm{BB(w}, \delta)$*,* $\mathrm{BB(z}, \varepsilon) \prec_{\mathrm{u}} \mathrm{BB(w}, \delta)$*, iff for every* $z^i \in \mathrm{BB(z}, \varepsilon)$ *and every* $w^i \in \mathrm{BB(w}, \delta)$*:* $z^i \prec w^i$*.*

If $z = f(x)$ with confidence ε, $w = f(y)$ with confidence δ and $\mathrm{BB(z}, \varepsilon) \prec_{\mathrm{u}} \mathrm{BB(w}, \delta)$, then solution x probably dominates solution y. In other words, x dominates y with a (high) confidence (depending on ε and δ).

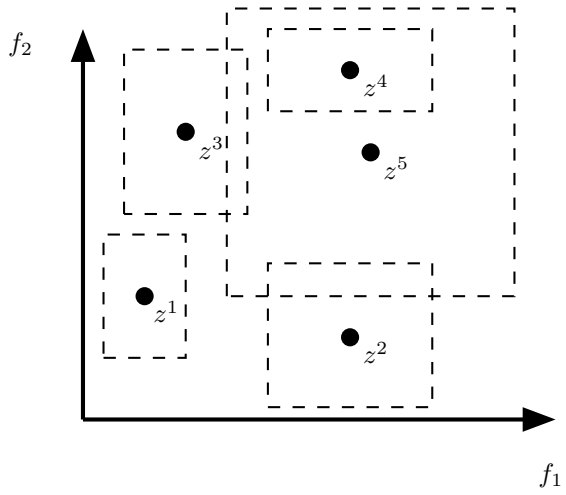
Fig. 4 presents the objective values $z^1, .., z^5$ and their bounding boxes. We can see that z^1 probably dominates solution z^4 ($z^1 \prec_{\mathrm{u}} z^4$).

Analogously, other relations can be defined.

Definition 4.6 (probable Pareto non-dominance). *The bounding box* $\mathrm{BB(z}, \varepsilon)$ *is probably non-dominated by the bounding box* $\mathrm{BB(w}, \delta)$*,* $\mathrm{BB(z}, \varepsilon) \not\prec_{\mathrm{u}} \mathrm{BB(w}, \delta)$*, iff for every* $z^i \in \mathrm{BB(z}, \varepsilon)$ *and* $w^i \in \mathrm{BB(w}, \delta)$*:* $z^i \prec w^i$ *or* $z^i || w^i$*.*

Several examples of probable Pareto non-dominance can be seen in Fig. 4: $z^1 \not\prec_{\mathrm{u}} z^2, z^1 \not\prec_{\mathrm{u}} z^3, z^1 \not\prec_{\mathrm{u}} z^4, z^1 \not\prec_{\mathrm{u}} z^5, z^2 \not\prec_{\mathrm{u}} z^4, z^3 \not\prec_{\mathrm{u}} z^4$.

When $z = f(x) \in \mathrm{BB(z}, \varepsilon), w = f(y) \in \mathrm{BB(w}, \delta)$ and $\mathrm{BB(z}, \varepsilon) \not\prec_{\mathrm{u}} \mathrm{BB(w}, \delta)$, we say that solution x is probably non-dominated by solution y. Only when the uncer-



Slika 4: Approximated solutions presented in the objective space using bounding boxes

tainty is finally removed, e.g. the solutions are exactly evaluated, we learn whether x dominates y or they are incomparable.

Definition 4.7 (probable incomparability). *The bounding box* $\mathrm{BB(z}, \varepsilon)$ *is probably incomparable with the bounding box* $\mathrm{BB(w}, \delta)$*,* $\mathrm{BB(z}, \varepsilon) \mid\mid_{\mathrm{u}} \mathrm{BB(w}, \delta)$*, iff for every* $z^i \in \mathrm{BB(z}, \varepsilon)$ *and* $w^i \in \mathrm{BB(w}, \delta)$*:* $z^i \mid\mid w^i$*.*

Again, two solutions are probably incomparable when the corresponding bounding boxes are probably incomparable.

Finally, when none of the presented relations under uncertainty apply, two solutions are in an uncertain relation.

In Fig. 4, z^2 is probably incomparable with z^3.

Definition 4.8 (uncertain relation). *The bounding box* $\mathrm{BB(z}, \varepsilon)$ *is in an uncertain relation with the bounding box* $\mathrm{BB(w}, \delta)$*,* $\mathrm{BB(z}, \varepsilon) \sim_{\mathrm{u}} \mathrm{BB(w}, \delta)$*, iff* $\mathrm{BB(z}, \varepsilon) \cap \mathrm{BB(w}, \delta) \neq 0$*.*

In Fig. 4, z^5 is in an uncertain relation with z^2, z^3 and z^4.

If solution x probably dominates solution y, then solution x is also probably non-dominated by solution y:

$x \prec_{\mathrm{u}} y \Rightarrow x \not\succ_{\mathrm{u}} y$.

The probable Pareto dominance as well as the probable incomparability imply probable Pareto non-dominance.

If all the solutions are exactly evaluated, i.e., all their corresponding confidence interval widths equal zero, the relations presented in this subsection directly translate to the relations described in Subsection 4.1.

Using these relations under uncertainty, a multiobjective optimization algorithm can often manage to compare two solutions without the need to exactly evaluate them first.

When in the comparison some uncertainty still remains, the solutions can be exactly evaluated to eliminate any doubt. This procedure can diminish the number of mistakes made due to wrong assessments of the solutions.

5. The GP-DEMO Algorithm

The GP-DEMO algorithm for surrogate-model-based optimization is, as the name suggests, built upon the DEMO algorithm [1]. DEMO is an multiobjective evolutionary algorithm based on Differential Evolution (DE) [41]. Like DE, DEMO is easy to understand and implement, and very effective on numerical problems. The main disadvantage of this algorithm is that it is not suited for solving combinatorial problems because the candidate creation uses vector addition and multiplication. DEMO is a steady-state evolutionary algorithm that adds candidate solutions to the existing population. Since they are immediately used for generating new solutions, the algorithm's convergence is accelerated. DEMO is also effective in uniformly spreading the solutions on the non-dominated front. This is done by removing the solutions from the extended population with the selection method taken from the NSGA-II algorithm [42]. GP-DEMO is very similar to DEMO. The difference is in that GP-DEMO approximates objective values and their confidence intervals with GP models and uses relations for comparing the solutions under uncertainty. After being approximated, solutions need to be exactly evaluated when 1) this is required to formalize a comparison (see subsection 5.1) and 2) they are the best found solutions (see subsection 5.2).

The GP-DEMO pseudocode is shown in Fig. 5.

GP-DEMO

1. Exactly evaluate the initial population $\mathcal{P}$ of random individuals.
2. Build initial GP model.
3. While stopping criterion not met, do:
   - 3.1. For each individual p_i ($i = 1, \ldots, popSize$) from $\mathcal{P}$ repeat:
     - (a) Create candidate c from parent p_i.
     - (b) Approximate the candidate with the GP model.
     - (c) Compare under uncertainty c and p_i and keep either the best one or both (see Subsection 5.1).
   - 3.2. If the population has more than $popSize$ individuals, use selection procedure under uncertainty.
   - 3.3. Update the GP model from the set of exactly evaluated solutions.
   - 3.4. Randomly enumerate the individuals in $\mathcal{P}$.
4. Exactly evaluate all approximated solutions on the front.

Slika 5: Outline of the GP-DEMO algorithm.

As already mentioned in Subsection 3.2, we decided to use the SPGP sparse approximation for GP modeling due to it having a much lower computational complexity than "full" GP modeling. Although the SPGP is a sparse approximation method, updating the model is a relatively slow operation. Therefore, we decided not to update the model after every new, exact evaluation becomes available, but only after every new generation. Such an approach seems natural for evolutionary algorithms and can be interpreted as batch learning. It should be noted that only exact evaluations are included in updating the model. This means that in the worst case, the number of solutions evaluated exactly is the same as the population size, and in the best case, no solution is evaluated exactly and as a result there is no need to update the model. The number of

exactly evaluated solutions depends on the quality of the model. In general, it is true that the better the model, the lower the number of exactly evaluated solutions. In some cases, when the fitness function is very complex and the GP model smooths it, even though the approximation mean value is accurate enough, the variance is too big, so solutions are evaluated exactly.

The update of the SPGP model is implemented as some kind of windowing technique. After every generation, the n last exactly evaluated solutions are used to update the model, where n is the window size. As we use the SPGP sparse approximation for GP modeling, m data points (exactly evaluated solutions) are obtained as the active set. Both parameters, n and m, are design parameters. It should be noted that the active set is calculated from scratch during every update, which means that the active set of the previous model is not used as the initial active set in the model update. Due to the nature of the optimization process, we do not want to keep the solutions from the whole decision space, but the solutions near the Pareto optimal front. Nevertheless, the GP model's hyperparameters are preserved, as their values are not supposed to change much with each generation/update, and therefore the model optimization converges much more quickly.

The creation of the candidate solution from the parent the solution is done in the same way as in DEMO. The comparison of the candidate and parent solutions and the decision on the outcome are described in Subsection 5.1. The selection process where the number of solutions is reduced to the population size is also adapted for surrogate-model-based optimization with confidence intervals. This procedure is described in Subsection 5.2.

5.1. Comparing solutions under uncertainty

A comparison of the candidate and parent solutions in the GP-DEMO algorithm is based on the relations under uncertainty described in Subsection 4.2. There are six possible situations that can happen when comparing candidate c with the vector of confidence intervals ε and the parent p with the vector of confidence intervals δ:

1. If $c \,||_{\mathrm{u}}\, p$, both solutions are added to the population.

In this case, solutions c and p are certainly incomparable. Even if both solutions would be exactly evaluated and thus with confidence interval widths equal to zero, they would still be incomparable and the algorithm would still add both solutions to the population. Hence, no additional evaluations are needed in this case.

2. If $c \prec_{\mathrm{u}} p$, solution c is added to the population and solution p is discarded.
   Here the solution c is definitely better than the solution p, therefore no additional evaluations are necessary as they would not change the dominance relation.
3. If $p \prec_{\mathrm{u}} c$, solution p is added to the population and solution c is discarded.
   This case is similar to the previous one.
4. If $c \not\prec_{\mathrm{u}} p$, the algorithm checks ε. If $\varepsilon \neq 0$, the algorithm exactly evaluates c and compares the solutions again. If $\varepsilon = 0$, the algorithm exactly evaluates p and compares the solutions again.
   In this case, solution p is better in at least one objective and not worse in the others. In order to determine if either the solution c dominates solution p or they are incomparable, (at least) one solution needs to be exactly evaluated. Because c looks more promising, its confidence interval is checked. If its width is different from zero, meaning that the solution is approximated, the algorithm exactly evaluates solution c and then compares the solutions again. If the confidence interval width is equal to zero, meaning that c is exactly evaluated, then, in order to be able to compare the solutions, the algorithm exactly evaluates solution p and compares the solutions again.
5. If $p \not\prec_{\mathrm{u}} c$, the algorithm checks δ. If $\delta \neq 0$, the algorithm exactly evaluates p and compares the solutions again. If $\delta = 0$, the algorithm exactly evaluates c and compares the solutions again.
   This case is similar to the previous one, except that the solution p is now more promising.
6. If $c \sim_{\mathrm{u}} p$, the algorithm checks ε. If $\varepsilon \neq 0$, the algorithm exactly evaluates c and compares the solutions again. If $\varepsilon = 0$, the algorithm exactly evaluates p and compares the solutions again.

In this case, the only way to see which solution is better is to exactly evaluate (at least) one solution. Because the candidate (offspring) has the potential to be better then the parent, the algorithm first checks if it is exactly evaluated. If it is not, the algorithm exactly evaluates it. If it is, the algorithm exactly evaluates the parent and than compares the solutions again.

5.2. Selection under uncertainty

The function of the selection procedure in a multiobjective evolutionary algorithm is to limit the size of the population and to uniformly spread the solutions on the front. The selection procedure in GP-DEMO is based on the selection procedure proposed in the NSGA-II algorithm. This selection procedure involves nondominated sorting and ranking using the crowding distance metric. In nondominated sorting all the nondominated individuals are allocated into the first front and the nondominated sorting is applied again to the remaining individuals. In this way, a sequence of fronts is obtained, where the individuals from the preceding fronts are preferred to those from the subsequent fronts. The new population is filled in turn with the individuals from the best fronts. If a front cannot fit into the population entirely, the individuals from this front are further ranked according to the crowding distance metric. Sorting based on crowding distance prefers individuals from less crowded regions of the objective space to ensure a good spread of solutions.

The problem that can occur when performing nondominated sorting with approximated solutions is that some solutions can wrongly dominate other solutions. For this reason those solutions are then dominated and can be discarded. As a result the process of finding the best nondominated solutions is misled. To prevent this from happening, the original selection procedure is modified so that the solutions that are either certainly or possibly (because of confidence intervals) on the first front are exactly evaluated. With this approach we ensure that the front of nondominated solutions is always accurate, there are no deficiencies in the optimization process, and the possibility of getting stuck in the local optima because of inaccurate approximations is reduced.

To calculate the crowding distance metric, the approximated objective values are used. In this step, the algorithm does not exactly evaluate any more approximated

solutions, to spare as many exact evaluations as possible.

6. Numerical Evaluation

This section describes the test problems that were used in this study, presents the settings used for testing, shows the results of the optimization, and compares the results gained with different algorithms. The section concludes with the discussion and explanation of the results.

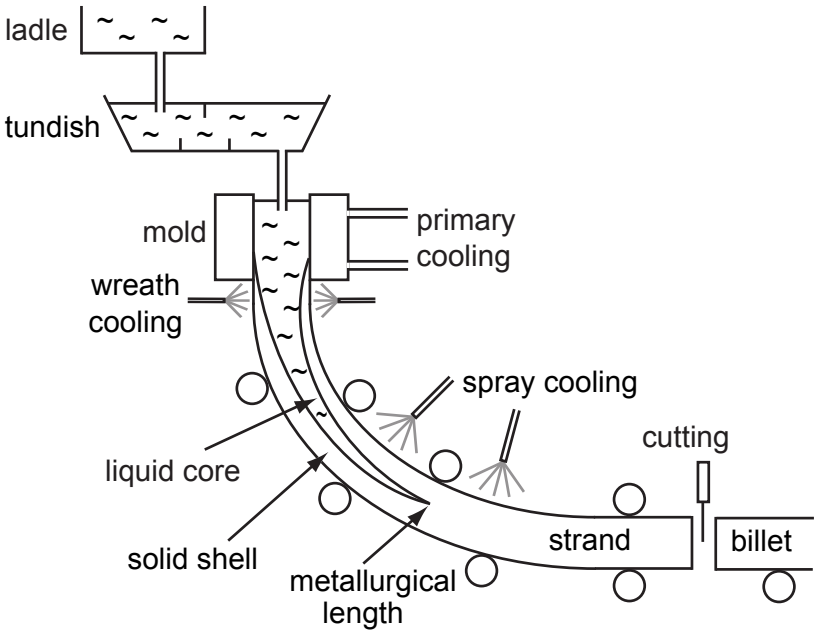
6.1. Test problems

The test problems used in our study are all minimization problems and can be divided into two groups. The first group consists of known benchmark problems and the second group consists of two real-world problems. The benchmark problems were divided into easier and more complex problems to test all the aspects of the algorithms. The first of the two real-world problems is the optimization of a continuous steel casting process and the second one is the problem of finding the best correlation between a simulated and a measured electrocardiogram (ECG).

6.1.1. Benchmark problems

The benchmark problems are further divided into two subgroups. The first subgroup consists of three problems from [43] called BNH, OSY and SRN. All the problems are constrained and have two objectives. These three problems are relatively simple and are used to measure how many exact evaluations can be saved with surrogate-model-based algorithms in comparison to DEMO.

The second subgroup consists of the WFG test problems introduced in [44]. The WFG toolkit is used to construct a suite of problems that provides a thorough test for optimizers. The nine WFG problems, WFG1–WFG9, are formulated in such a manner that each poses a different type of challenge to the optimizers. The WFG toolkit tests the abilities of surrogate-model-based algorithms to find solutions comparable to the ones gained with DEMO on simple as well as complex problems.



Slika 6: Scheme of the steel casting process

6.1.2. The continuous steel casting problem

The continuous casting of steel is a very complex metallurgical process where molten steel is cooled and shaped into semi-manufactures of desired dimensions. The main components of the casting system (schematically shown in Fig. 6) are the ladle, tundish, mold and cooling subsystems [45].

The process of steel casting starts with molten steel being poured into a ladle from an electric furnace and then led through the tundish that acts as a buffer for the liquid metal, which is then drained into an open-base copper mold. The water-cooling inside the mold cools the mold and the hot steel starts solidifying in contact with it. The water flowing through the channels built into the walls of the mold cools the steel. The channels represent the primary cooling subsystem.

Molten steel with a thin solid shell, now called the strand, exits the base of the mold into a spray chamber where it is immediately supported by closely spaced water-cooled rollers. The strand is sprayed with water in the wreath and spray cooling areas in order to increase the rate of solidification. Together, the wreath and spray cooling

areas represent the secondary cooling subsystem.

When the steel exits the casting system, it is cut into billets of the desired length. The length of the liquid core in the strand is called the metallurgical length. The metallurgical length, the thickness of the solid shell at the mold exit, and the strand surface temperature at the unbending point have a large effect on the quality of the cast steel.

The optimization problem involves input variables (process parameters), output variables, and the desired output values, determined by domain experts. The task is to find the input variable settings resulting in values of the output variables as close as possible to the desired values. Based on empirical knowledge in the steel production domain, such settings result in high-quality steel.

Since the process of steel casting is expensive, time consuming and could also be dangerous, it is necessary to have a model to make the optimization of the parameters of the steel casting possible. To model the casting, the numerical model of steel casting [46] was used. The four input variables of this numerical model that are being optimized are the casting speed, the mold outlet coolant temperature, the wreath system coolant flow, and the spray system coolant flow. The lower and the upper bounds for those variables were also determined by experts.

Given the input parameters, the simulator computes the three output variables that are essential for the quality of cast steel. The output variables are the metallurgical length, the shell thickness and the surface temperature at the unbending point. As an optimization criterion, the difference between the output variable produced by the numerical simulator and its desired value is considered. Thus, the goal is to find such values of the input variables that all the criteria would be 0 or as close to 0 as possible.

The time needed to exactly evaluate one simulation of the steel casting process is approximately 2 minutes on a 3.4-GHz Intel Core i7 computer with 8 GB RAM.

6.1.3. The ECG problem

The second real-world test problem is the problem of finding the best correlation between a simulated and a measured ECG. An ECG is a diagnostic and monitoring tool that records heart activity by measuring, on the body surface, the electrical currents originating in the heart. Modeling the electrical activity of a human heart provides

useful insight into the ECG generating mechanisms that can in turn be used to further the understanding of the ECG and improve its diagnostic benefits.

For this problem the ECG simulator presented in [47] was used. The simulator uses a simplified heart cell model consisting of the action potential (AP), a function which defines the heart cells' electrical activity. Since we focus only on the difference between the T waves of the ECG traces, the full resolution and complexity of this simulator is not needed. Thus, a coarse model consisting of eight times fewer heart cells than the original model was used, enabling a faster simulation.

The input parameters (variables of the optimization problem) of the simulator consist of two groups of four parameters. Every group defines the AP of the heart's cell layer.

This optimization problem consists of two objectives. For every objective, the objective function first calculates the Pearson correlation coefficient [48] between the simulated ECG and the ECG measured on one of the two location points on the body. Then, in order to get a minimization problem, the function calculates the objective value (f) by subtracting the Pearson correlation coefficient (PCC) from 1:

$$f = (1 - PCC). \tag{6}$$

When the simulated and the measured ECG are fully correlated, their Pearson correlation coefficient is equal to 1 and the objective value is equal to 0.

Solving this optimization problem consists of finding the right combination of variable values for setting the APs of the simulated heart in such a way that the simulated ECG is as close as possible to the measured ECG.

The time needed to exactly evaluate one solution of the ECG problem is approximately 15 seconds on a 3.4-GHz Intel Core i7 computer with 8 GB RAM.

6.2. Experimental setup

For the purpose of determining the quality of the results obtained with the GP-DEMO algorithm, a comparison was made with the DEMO algorithm and with the surrogate-model-based algorithm called Generational Evolution Control (GEC). We implemented GEC based on the algorithm NSGA-II-ANN from [14]. The basic idea

of this algorithm is that during the optimization process in some generations all the solutions are exactly evaluated, while in others, all the solutions are approximated with a surrogate model. In their paper, the authors describe different versions of the NSGA-II-ANN algorithm. For the purpose of this research we chose the version of the NSGA-II-ANN algorithm that the authors claimed to be better than other versions. In this version the number of generations for exact evaluations is three, followed by seven generations of approximated solutions. This combination is then repeated during the whole optimization process.

To be able to perform a fair comparison of the algorithms, some modifications had to be made to the NSGA-II-ANN algorithm, and this modified algorithm is called GEC. In the GEC algorithm, GP modeling was used for the modeling instead of the artificial neural network (ANN), because the use of a different surrogate model would influence the results. Instead of the NSGA-II algorithm, the DEMO algorithm was used to ensure that the creation and combination of candidate solutions did not affect the results. At the end of the optimization process, in order to get a comparable hypervolume and nondominated solutions, all the approximated nondominated solutions get exactly evaluated. This ensures that the front of nondominated solutions is accurate and not approximated.

The algorithm parameter values used for the testing were the same for all three algorithms. They were set as follows:

- maximum number of solution evaluations: 10000,
- population size: 100,
- weight: 0.5,
- crossover probability: 0.3,
- selection method: as in NSGA-II.

The maximum number of solution evaluations for the continuous steel casting problem was 3000 instead of 10000 in order to save time and because after 3000 solution evaluations the hypervolume did not increase any more.

The GP-DEMO and the GEC algorithms used GP modeling to create the surrogate models. The modeling technique and the parameter values used were the same in both algorithms (see Section 3). The width of the confidence interval was equal to two standard deviations. This means that the probability that the exactly evaluated solution is within the confidence interval of the approximated solution is 95%. The sizes of the active set and the window set were determined after trying different settings and were chosen as the best compromise between the time needed to build the model and the precision of this model for the approximation.The parameter values are following:

- active set size: 350,
- window set size: 500.

The window set is relatively small. This ensures that during the optimization process, when approaching the optimum, the surrogate model uses only exactly evaluated solutions close to the optimum. This locality enables the surrogate model to be more precise, thus making the confidence interval narrower during the optimization.

The experiments were run 30 times for each of the benchmark problems and 10 times for each of the real-world problems because of similar results and in order to save time.

6.3. Results

For every problem and for every algorithm four different measures were obtained:

- number of exact evaluations performed during the optimization process,
- final hypervolume,
- duration of the optimization process,
- number of nondominated solutions on the final front.

The results averaged over all the runs are presented in Tables 1–3.

For every problem the final fronts of all the tested algorithms are shown in Fig. 7–20.

Tabela 1: Results on BNH, OSY and SRN test problems

Problem	Algoritem	Število eksaktno ovrednotenih rešitev	Hipervolumen	Optimizacijski čas	Nedominirane rešitve
BNH	DEMO	10000	0.6880	00:00:05	100
	GP-DEMO	401	0.6880	00:05:47	100
	GEC	3080	0.6878	00:14:18	100
OSY	DEMO	10000	0.9643	00:00:05	100
	GP-DEMO	1485	0.9646	00:24:36	100
	GEC	3078	0.9643	00:23:08	99
SRN	DEMO	10000	0.9547	00:00:05	100
	GP-DEMO	267	0.9550	00:02:49	100
	GEC	3091	0.9553	00:14:53	100

6.4. Discussion

The evaluation of the GP-DEMO algorithm was done by comparing it with two other multiobjective evolutionary algorithms. The first one is DEMO, which is known to be very effective and was used for comparing the quality of the results. The second one is GEC, a surrogate-model-based multiobjective evolutionary algorithm that was used to compare both the quality of the results and the number of exactly evaluated solutions performed during the optimization process.

The first set of benchmark optimization problems that was used for the testing (BNH, OSY and SRN) is composed of relatively simple optimization problems. Because of this simplicity, the GP modeling technique is able to create very precise models of their objective functions. Therefore, the approximated solutions have very narrow confidence intervals and are rarely required to be exactly evaluated. The comparison of the solutions with very narrow confidence intervals can usually be done without exactly evaluating them. This results in the GP-DEMO and GEC getting almost the same hypervolume as DEMO, but with fewer exactly evaluated solutions. On these problems, GP-DEMO exactly evaluated only between 3 % (BNH and SRN) and 15 % (OSY) of all the solution evaluations. The particularity of the GEC algorithm is that the number of exact evaluations performed during the optimization process differs

Tabela 2: Results on WFG test problems

Problem	Algoritem	Število eksaktno ovrednotenih rešitev	Hipervolumen	Optimizacijski čas	Nedominirane rešitve
WFG1	DEMO	10000	0.9376	00:00:10	72
	GP-DEMO	8291	0.8965	02:42:01	51
	GEC	3100	0.7512	00:38:08	11
WFG2	DEMO	10000	0.9643	00:00:09	100
	GP-DEMO	3991	0.9640	02:22:54	100
	GEC	3003	0.9605	00:38:11	44
WFG3	DEMO	10000	0.9594	00:00:10	100
	GP-DEMO	4864	0.9594	02:21:26	100
	GEC	3023	0.9579	00:37:26	85
WFG4	DEMO	10000	0.9340	00:00:10	100
	GP-DEMO	3508	0.9288	02:33:52	94
	GEC	3025	0.9248	00:37:13	25
WFG5	DEMO	10000	0.9154	00:00:09	100
	GP-DEMO	6710	0.9143	02:10:04	100
	GEC	3083	0.9135	00:38:56	73
WFG6	DEMO	10000	0.9365	00:00:09	100
	GP-DEMO	2003	0.9216	02:24:44	71
	GEC	3019	0.9309	00:37:33	70
WFG7	DEMO	10000	0.9368	00:00:09	100
	GP-DEMO	7897	0.9368	02:52:01	100
	GEC	3080	0.9155	00:38:42	23
WFG8	DEMO	10000	0.8653	00:00:10	98
	GP-DEMO	3273	0.8649	02:22:14	89
	GEC	3022	0.8558	00:36:33	57
WFG9	DEMO	10000	0.9207	00:00:09	100
	GP-DEMO	8988	0.9204	02:42:01	100
	GEC	3077	0.8911	00:35:18	13

Tabela 3: Results on real-world test problems

Problem	Algoritem	Število eksaktno ovrednotenih rešitev	Hipervolumen	Optimizacijski čas	Nedominirane rešitve
Kontinuirno ulivanje	DEMO	3000	0.5078	112:22:37	100
	GP-DEMO	950	0.5078	36:20:05	100
	GEC	1090	0.5074	39:28:39	97
EKG	DEMO	10000	0.9987	39:21:19	100
	GP-DEMO	8135	0.9986	36:35:18	100
	GEC	3100	0.9937	13:16:04	22

very little from one problem to another. Since in three out of ten generations the newly created solutions are exactly evaluated, in our case, with the maximum number of evaluated solutions equal to 10000, the number of exactly evaluated solutions is at least 3000. Additional exact evaluations are performed at the end of the optimization process where all the approximated solutions are exactly evaluated. Because of this strategy the number of exact evaluations performed with GEC varies between 3000 and 3100 on all the problems, which is a little more than 30 % of all the evaluated solutions.

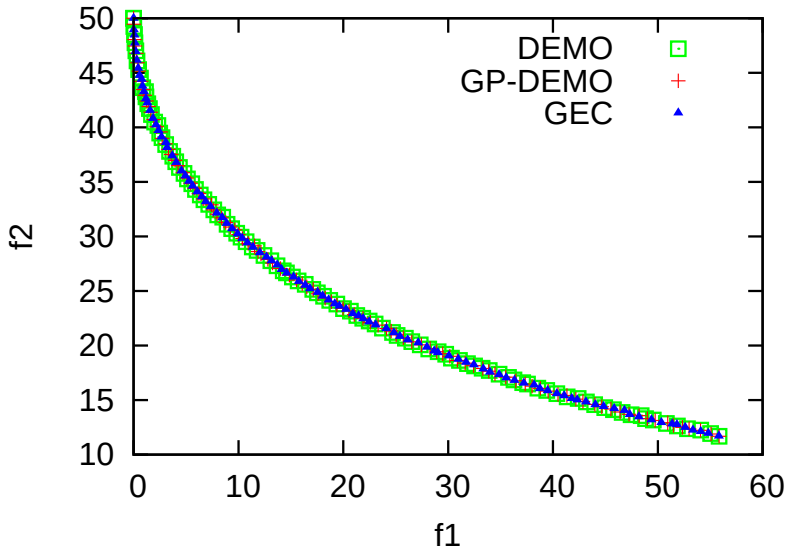
The WFG test problems were the second set of benchmark problems. These problems are known to be hard optimization problems designed to thoroughly test any algorithm. Because of their complexity, the WFG objective functions are very difficult to model. This results in two possible problems. The first problem is that the confidence intervals of the approximated solutions are wider than on the previous set of test problems. Therefore, many solutions need to be exactly evaluated and the reduction in the number of exactly evaluated solutions was only between 10 % and 80 %. Secondly, the GP model could be in some cases overconfident in what are actually wrongly approximated predictions. Such cases occur when the modeled function has sudden changes in values and the GP model identifies these changes as outliers or noise. As the GP model, due to the probabilistic nature, smooths them, the approximation is wrong. When such changes are relatively large compared to the “usual” changes, the variance of the approximation is small and therefore the exact value is not inside the confidence interval. In order to prevent solutions from falling out of the

confidence interval, we could make the confidence interval wider, e.g., three standard deviations, but since this happens rarely and the final results are still good, we did not change the width of the confidence interval. This case occurred mainly on the WFG1 problem and the hypervolume obtained with GP-DEMO was slightly worse compared to DEMO (Fig. 10). The hypervolume values on the other WFG problems are very similar. Although the solutions get closer together during the optimization, the number of exact evaluations does not increase. The approximations get more precise, resulting in narrower confidence intervals, which rarely overlap.

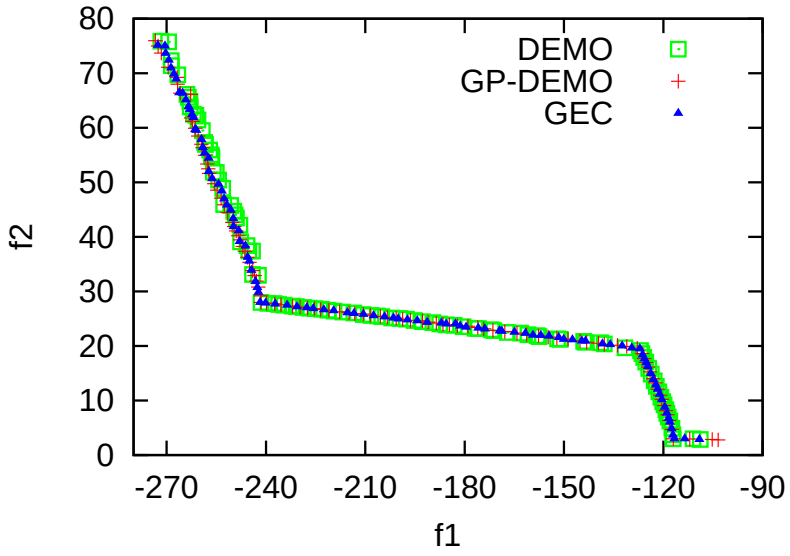
The results gained with GEC on the WFG test problems are the worst on almost all the problems. The final fronts have fewer solutions and lower hypervolumes. Because the GEC algorithm does not use the confidence interval when comparing solutions, the mistakes where an approximated solution wrongly dominates the other solutions occur more often. Because the surrogate models for the WFG test problems are not very accurate, the approximation errors are larger and more frequent. On all the WFG test problems, just before the end of the optimization process, the final fronts gained with GEC consisted of 100 solutions and their hypervolumes were competitive. After exactly evaluating these solutions, the results got worse.The reason why algorithms like GEC face this dificulty is that early in the optimization process a wrongly approximated solution appears to be very good. This solution then prevents other high-quality solutions from staying in the population because it dominates them. At the end of the optimization process this solution is exactly evaluated as a low-quality solution, lowering the hypervolume since a part of the decision space gets weakly covered.

The analysis of the results from the real-world test problems show similar findings. The difference with real-world problems is that the whole optimization process takes longer because every exact solution evaluation is computationally expensive.

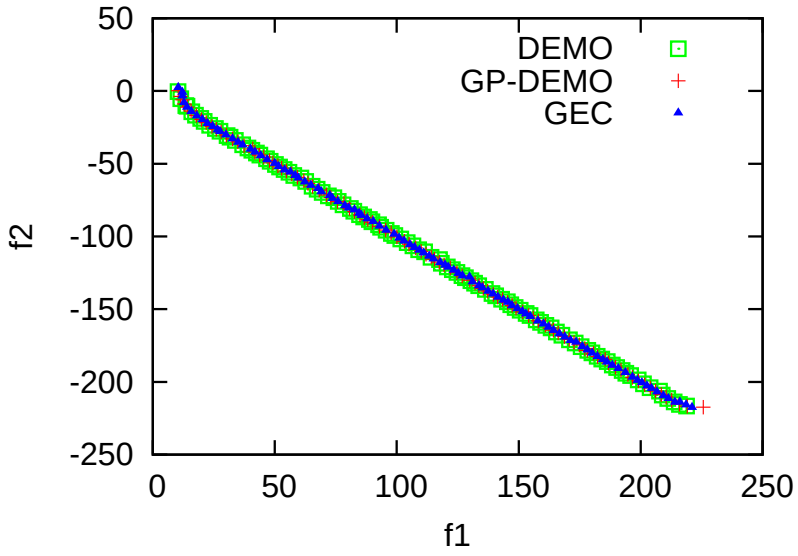
The first real-world problem is the problem of optimizing the continuous casting process in order to get the best possible quality of cast steel. The GP modeling technique is able to create an accurate surrogate model of the continuous steel casting process so the hypervolume and the size of the final front were almost identical for all three algorithms. The GP-DEMO algorithm exactly evaluated just a little over 30 % of all the solution evaluations. This percentage would be even higher if the stopping



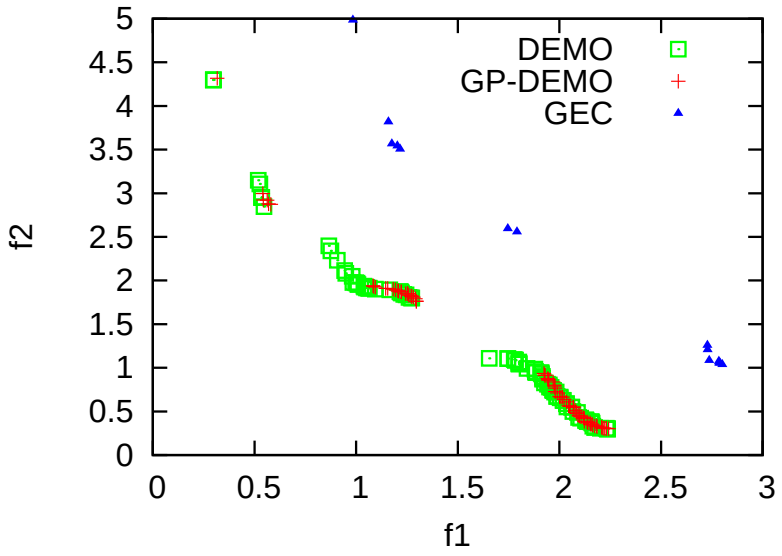
Slika 7: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the BNH problem



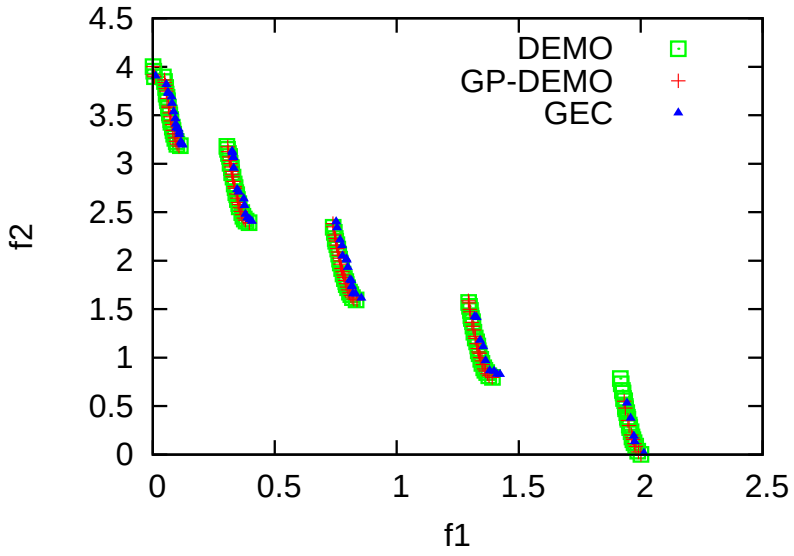
Slika 8: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the OSY problem



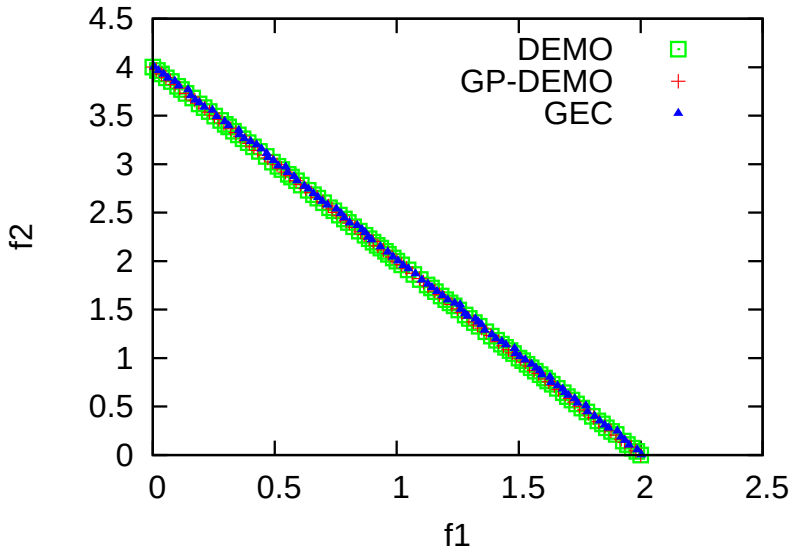
Slika 9: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the SRN problem



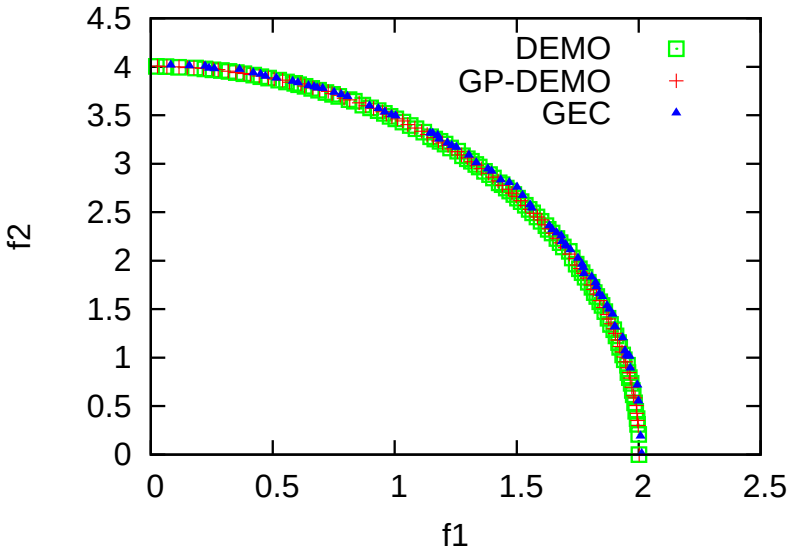
Slika 10: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG1 problem



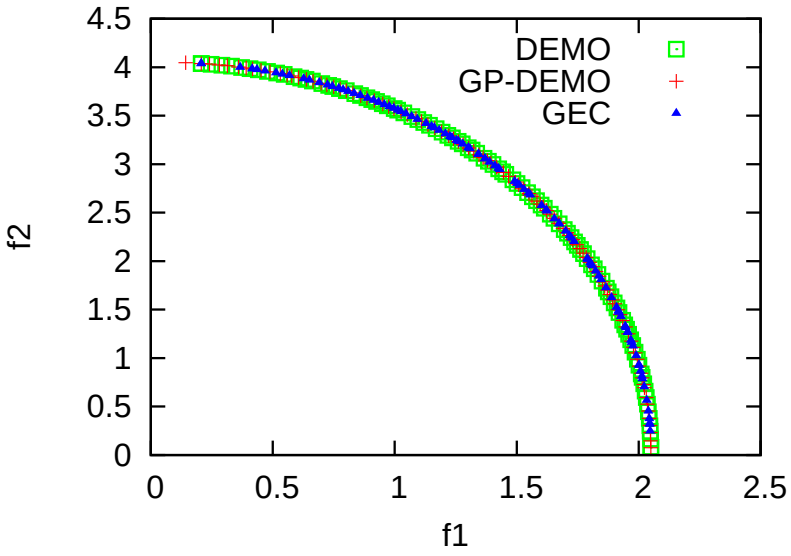
Slika 11: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG2 problem



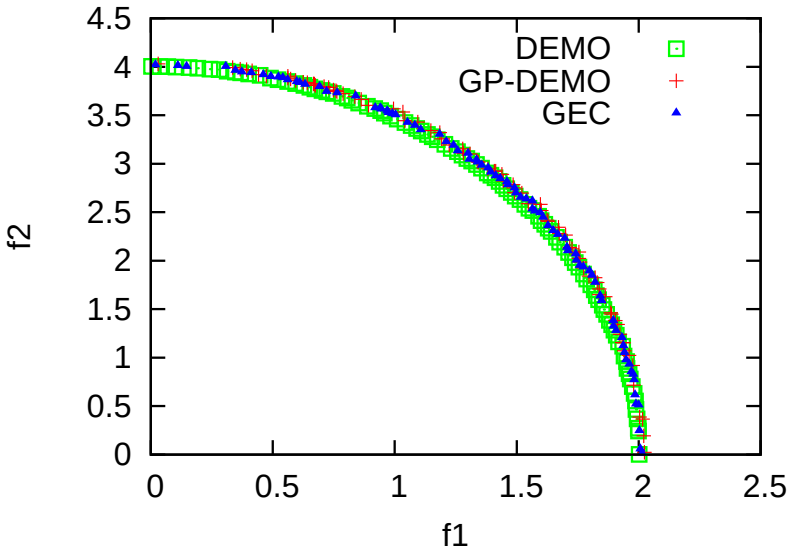
Slika 12: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG3 problem



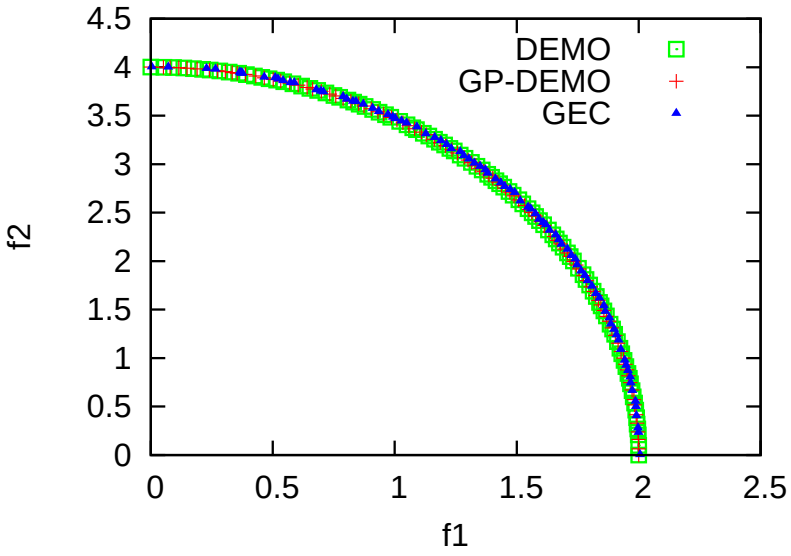
Slika 13: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG4 problem



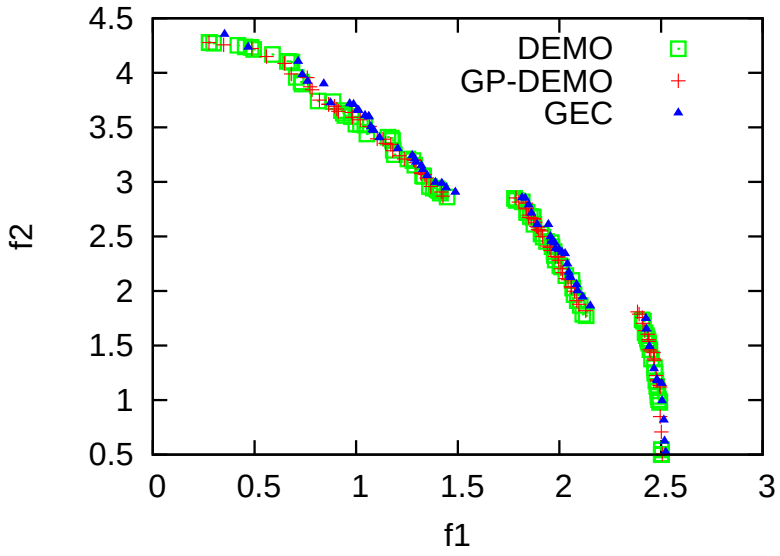
Slika 14: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG5 problem



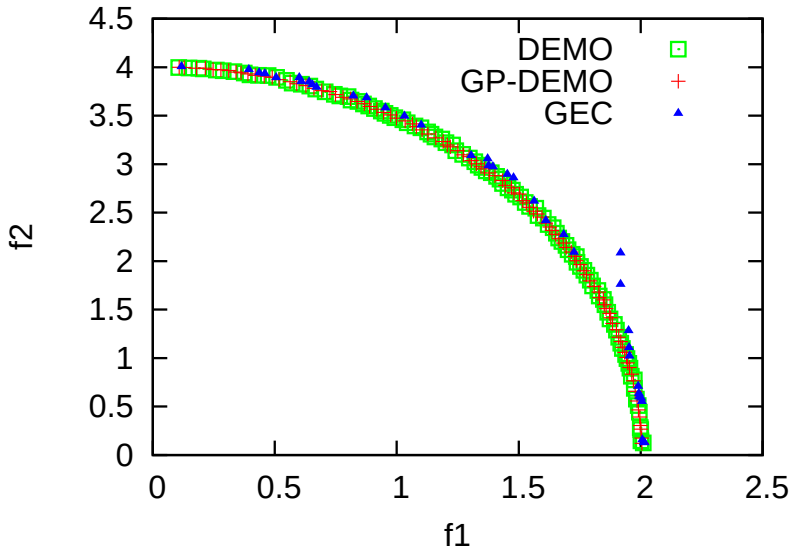
Slika 15: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG6 problem



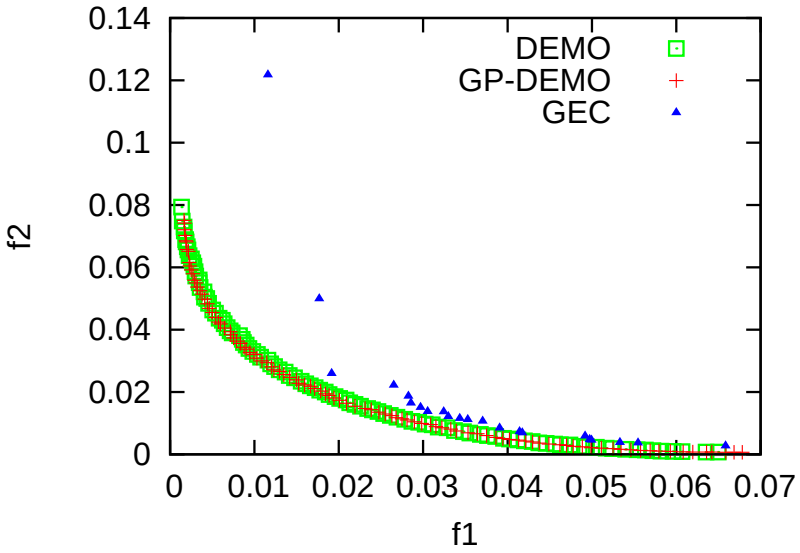
Slika 16: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG7 problem



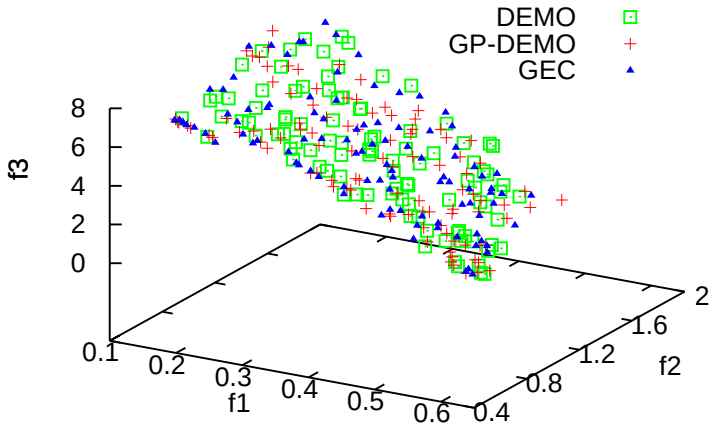
Slika 17: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG8 problem



Slika 18: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the WFG9 problem



Slika 19: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the ECG problem



Slika 20: Fronts of non-dominated solutions found by DEMO, GP-DEMO and GEC on the continuous steel casting problem

criterion would allow more than 3000 solution evaluations, because at the beginning the surrogate model is not as accurate as it becomes after updates. With more precise surrogate models, the confidence intervals become narrower and less exact evaluations are needed. Because one exact evaluation takes more than two minutes, the time needed for optimization with GP-DEMO on this problem is three days shorter than with DEMO.

The analysis of the results on the ECG problem shows similar algorithm behavior as was seen on the WFG test problems. The surrogate models created during the optimization process return quite narrow confidence intervals. However, because the solutions on the front are very close to zero, and also very close to one another, this confidence interval still turns out to be relatively wide. This then results in GP-DEMO achieving a small saving in the number of exactly evaluated solutions, but still managing to get results that are as good as the ones gained with the DEMO algorithm. The GEC algorithm has difficulties once again in reaching the same quality of results, and the number of nondominated solutions on the final front is small.

If we compare the times needed for optimization we can see that for the benchmark problems, where one exact evaluation is almost instant, DEMO finds final fronts in a few seconds. With GP-DEMO and GEC the optimization times are longer because of the time spent building the surrogate models. The time needed for updating the surrogate models during the optimization process depends on the number of exactly evaluated solutions that are used for training, the complexity of the objective function, and also of the number of objectives, because for every objective a separate surrogate model is created. Since the exact evaluation of 10000 solutions on benchmark optimization problems takes just a few seconds, the time needed to update the surrogate models is approximately the same as the optimization time (see Tables 1 and 2). If the solution evaluations are very fast, the DEMO algorithm is the best, but on real-world problems, where exact solution evaluations take more time, GP-DEMO becomes more suitable because the optimization times are shorter and the quality of the results is comparable to the quality obtained with DEMO. In comparison with the other two algorithms, GEC sometimes needs fewer exact function evaluations, but its disadvantage is that the quality of the results is worse (see Tables 2 and 3).

Tabela 4: Border time (in seconds) for one exact solution evaluation where optimization time for GP-DEMO and DEMO take the same amount of time.

BNH	OSY	SRN	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
0.04	0.2	0.02	5.7	1.4	1.6	1.4	3.4	1.1	4.9	1.3	9.5

For the benchmark problems we calculated how long one exact solution evaluation should take, so that the optimization times of GP-DEMO and DEMO would be equal. These border times are shown in Table 4. If the time needed for one exact solution evaluation is known, this helps us choose which algorithm to use in order to get good results as fast as possible. On the test problems that can be accurately modeled the border times are very short and on problems with less accurate surrogate models these times are slightly longer. These times could be further reduced by optimizing the GP modeling procedure. For example, the update of the surrogate models could be made parallel for every objective and also the update of the GP models could perhaps be performed less often.

7. Conclusion

In this paper, the newly developed surrogate-model-based multiobjective evolutionary algorithm GP-DEMO is presented. GP-DEMO is based on the algorithm DEMO using the GP model as a surrogate model for approximating solutions. To prevent wrongly performed comparisons of the solutions due to inaccurate approximated solutions, new relations for comparing solutions under uncertainty is suggested. These relations, also considering the confidence intervals of the approximation, are used for every comparison during the optimization process instead of the Pareto dominance relations. This prevents cases where an inaccurately approximated solution wrongly dominates a better solution, which results in slowing the optimization process or even in not finding the best solutions. The comparison of the candidate and parent solutions and the selection process for maintaining the population size are modified in such a way that we determine which solutions should be exactly evaluated to minimize the possibility of a wrongly performed comparison and still find optimal results with as few exact evaluations as possible.

We tested GP-DEMO on 12 different benchmark and two computationally expensive real-world problems. To assess the quality of the result, we solved those problems with another surrogate-model-based multiobjective evolutionary algorithm called GEC and with DEMO. The assessment was based on the number of exactly evaluated solutions, hypervolume value, optimization time and the number of nondominated solutions on the final front.

Comparing the hypervolume and the highest number on nondominated solutions on the final fronts, DEMO, as expected, obtained the best results, but the number of exactly evaluated solutions was a maximum as there is no solution approximations. On the other hand, the hypervolume values and the number of solutions on the final fronts obtained by GP-DEMO were almost identical to those obtained by DEMO on almost all the problems, but the number of exactly evaluated solutions performed by GP-DEMO was much lower, between 4% and 80% of all the exact evaluations, depending on the problem. In comparison to GP-DEMO and DEMO, the GEC algorithm obtained similar hypervolume values and the number of solutions on the final fronts on less complex problems, but worse on more complex problems due to the previously described problem of wrongly performed comparisons. As GEC has fixed evolution control, the number of exactly evaluated solutions was around 30% of all the evaluations.

A comparison of the optimization time shows that for benchmark problems DEMO is the most suitable of the compared algorithms, but it is not suitable for computationally expensive real-world problems. In contrast, both surrogate-model-based evolutionary multiobjective algorithms are more suitable for computationally expensive real-world problems, because the time needed for training and updating the surrogate models on benchmark problems takes up most of the whole optimization time.

As the quality of the results obtained by GP-DEMO and DEMO was very similar, we calculated the border time of one exact solution evaluation. If the exact solution evaluation is longer than the border time, then the optimization time of DEMO is longer than the optimization time of GP-DEMO. Therefore, we can say that in the case of a multiobjective optimization problem where one solution evaluation takes more than the border time and the quality of the results is important, GP-DEMO should be used.

Our future work will focus on improving the GP-DEMO algorithm. The first pos-

sibility is to speed up the optimization time by updating all the surrogate models after each generation in parallel. Later on we will work on GP modeling in order to speed up the update process and achieve approximations more accurately. Finally, instead of using a single model the algorithm will be extended to exploit two surrogate models, a global one, covering the whole space, and a local one, concentrating only on the current best solutions, but in greater detail.

Acknowledgments

The work presented in this paper was carried out under young researcher grant M090248, research programme P2-0209, and research projects L2-3651 and J2-4120, all funded by the Slovenian Research Agency. The authors are grateful to Dr. Robert Vertnik for providing a numerical simulator of the continuous casting process.

Literatura

[1] T. Robič, B. Filipič, DEMO: Differential evolution for multiobjective optimization, in: Evolutionary Multi-Criterion Optimization – EMO 2005, Vol. 3410 of Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 520–533.

[2] R. H. Myers, D. C. Montgomery, Response Surface Methodology: Process and Product Optimization Using Designed Experiments, Wiley, New York, 1995.

[3] R. L. Hardy, Multiquadric equations of topography and other irregular surfaces, Journal of Geophysical Research 76 (8) (1971) 1905–1915.

[4] D. F. Specht, Probabilistic neural networks, Neural Networks 3 (1) (1990) 109–118.

[5] M. L. Stein, Interpolation of Spatial Data: Some Theory for Kriging, Springer Series in Statistics, Springer, New York, 1999.

[6] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.

[7] M. Seeger, Gaussian processes for machine learning, International Journal of Neural Systems 14 (2) (2004) 69–106.

[8] D. J. C. MacKay, Introduction to Gaussian processes, in: C. M. Bishop (Ed.), Neural Networks and Machine Learning, NATO ASI Series, Kluwer Academic Press, 1998, pp. 133–166.

[9] J. Zhang, A. C. Sanderson, DE-AEC: A differential evolution algorithm based on adaptive evolution control, in: 2007 IEEE Congress on Evolutionary Computation (CEC), IEEE, Piscataway, 2007, pp. 3824–3830.

[10] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, K. Giannakoglou, Metamodel-assisted evolution strategies, in: Parallel Problem Solving from Nature – PPSN VII, Vol. 2439 of Lecture Notes in Computer Science, Springer, Berlin, 2002, pp. 361–370.

[11] Y. Jin, M. Olhofer, B. Sendhoff, Managing approximate models in evolutionary aerodynamic design optimization, in: 2001 IEEE Congress on Evolutionary Computation (CEC), Vol. 1, IEEE, Piscataway, 2001, pp. 592–599.

[12] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, K. Y. Lum, Combining global and local surrogate models to accelerate evolutionary optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications 37 (1) (2007) 66–76.

[13] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Computing 9 (1) (2003) 3–12.

[14] K. Deb, P. Nain, An evolutionary multi-objective adaptive meta-modeling procedure using artificial neural networks, in: S. Yang, Y.-S. Ong, Y. Jin (Eds.), Evolutionary Computation in Dynamic and Uncertain Environments, Vol. 51 of Studies in Computational Intelligence, Springer, Berlin, 2007, pp. 297–322.

[15] D. E. Grierson, W. H. Pak, Optimal sizing, geometrical and topological design using a genetic algorithm, Structural Optimization 6 (3) (1993) 151–159.

[16] M. Pilat, R. Neruda, An evolutionary strategy for surrogate-based multiobjective optimization, in: 2012 IEEE Congress on Evolutionary Computation (CEC), IEEE, Piscataway, 2012, pp. 1–7.

[17] M. Pilat, R. Neruda, ASM-MOMA: Multiobjective memetic algorithm with aggregate surrogate model, in: 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE, Piscataway, 2011, pp. 1202–1208.

[18] D. Chafekar, L. Shi, K. Rasheed, J. Xuan, Multiobjective GA optimization using reduced models, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications 35 (2) (2005) 261–265.

[19] J. Knowles, ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Transactions on Evolutionary Computation 10 (1) (2006) 50–66.

[20] M. Emmerich, K. C. Giannakoglou, B. Naujoks, Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels, IEEE Transactions on Evolutionary Computation 10 (4) (2006) 421–439.

[21] H. A. Taboada, F. Baheranwala, D. W. Coit, N. Wattanapongsakorn, Practical solutions for multi-objective optimization: An application to system reliability design problems, Reliability Engineering & System Safety 92 (3) (2007) 314–322.

[22] G. Li, M. Li, S. Azarm, S. A. Hashimi, T. A. Ameri, N. A. Qasas, Improving multi-objective genetic algorithms with adaptive design of experiments and online metamodeling, Structural and Multidisciplinary Optimization 37 (5) (2009) 447–461.

[23] T. Wagner, M. Emmerich, A. Deutz, W. Ponweiser, On expected-improvement criteria for model-based multi-objective optimization, in: Parallel Problem Solving from Nature – PPSN XI, Vol. 6238 of Lecture Notes in Computer Science, Springer, Berlin, 2010, pp. 718–727.

[24] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global Optimization 13 (4) (1998) 455–492.

[25] T. Voß, H. Trautmann, C. Igel, New uncertainty handling strategies in multi-objective evolutionary optimization, in: Parallel Problem Solving from Nature – PPSN XI, Vol. 6239 of Lecture Notes in Computer Science, Springer, Berlin, 2010, pp. 260–269.

[26] P. Limbourg, Multi-objective optimization of problems with epistemic uncertainty, in: Evolutionary Multi-Criterion Optimization – EMO 2005, Vol. 3410 of Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 413–427.

[27] R. Gunter, A partial order approach to noisy fitness functions, in: 2001 IEEE Congress on Evolutionary Computation (CEC), Vol. 1, IEEE, Piscataway, 2001, pp. 318–325.

[28] P. Limbourg, D. Aponte, An optimization algorithm for imprecise multi-objective problem functions, in: 2005 IEEE Congress on Evolutionary Computation (CEC), Vol. 1, IEEE, Piscataway, 2005, pp. 459–466.

[29] M. Emmerich, B. Naujoks, Metamodel assisted multiobjective optimisation strategies and their application in airfoil design, in: Adaptive Computing in Design and Manufacture VI, Springer, London, 2004, pp. 249–260.

[30] G. Matheron, The intrinsic random functions and their applications, Advances in Applied Probability 5 (1973) 439–468.

[31] N. A. Cressie, Statistics for Spatial Data, Wiley, New York, 1993.

[32] C. K. I. Williams, C. E. Rasmussen, Gaussian processes for regression, in: Advances in Neural Information Processing Systems 8, MIT Press, 1996, pp. 514–520.

[33] K. Ažman, J. Kocijan, Application of Gaussian processes for black-box modelling of biosystems, ISA Transactions 46 (4) (2007) 443–457.

[34] Ž. Južnič-Zonta, J. Kocijan, X. Flotats, D. Vrečko, Multi-criteria analyses of wastewater treatment bio-processes under an uncertainty and a multiplicity of steady states, Water Research 46 (18) (2012) 6121–6131.

[35] B. Grašič, P. Mlakar, M. Z. Božnar, Ozone prediction based on neural networks and Gaussian processes, Nuovo Cimento C 29 (2006) 651–661.

[36] B. Likar, J. Kocijan, Predictive control of a gas-liquid separation plant based on a Gaussian process model, Computers & Chemical Engineering 31 (3) (2007) 142–152.

[37] F. A. Viana, R. T. Haftka, L. T. Watson, Efficient global optimization algorithm assisted by multiple surrogate techniques, Journal of Global Optimization (2012) 1–21.

[38] J. Quinonero-Candela, C. E. Rasmussen, C. K. I. Williams, Approximation methods for Gaussian process regression, Tech. Rep. MSR-TR-2007-124, Microsoft Research (2007).

[39] J. Quinonero-Candela, C. E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, The Journal of Machine Learning Research 6 (2005) 1939–1959.

[40] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems 18, MIT Press, 2006, pp. 1257–1264.

[41] K. V. Price, R. Storn, Differential evolution – A simple evolution strategy for fast optimization, Dr. Dobb's Journal 22 (4) (1997) 18–24.

[42] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[43] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, 1st Edition, Wiley, New York, 2001.

[44] S. Huband, L. Barone, L. While, P. Hingston, A scalable multi-objective test problem toolkit, in: Evolutionary Multi-Criterion Optimization – EMO 2005, Vol. 3410 of Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 280–295.

[45] T. Robič, B. Filipič, In search for an efficient parameter tuning method for steel casting, in: Proceedings of the International Conference on Bioinspired Optimization Methods and their Applications – BIOMA 2004, Jožef Stefan Institute, Ljubljana, 2004, pp. 83–94.

[46] R. Vertnik, B. Šarler, Simulation of continuous casting of steel by a meshless technique, International Journal of Cast Metals Research 22 (1–4) (2009) 311–313.

[47] M. Depolli, V. Avbelj, R. Trobec, Computer-simulated alternative modes of U-wave genesis, Journal of Cardiovascular Electrophysiology 19 (1) (2008) 84–89.

[48] J. L. Rodgers, W. A. Nicewander, Thirteen ways to look at the correlation coefficient, The American Statistician 42 (1) (1988) 59–66.