Nonlinear Analysis: Modelling and Control, Vol. 31, No. 2, 463–476
https://doi.org/10.15388/namc.2026.31.45618

Vilnius University Press

An open-source parallel algorithm of Bayesian-based global search with Hooke–Jeeves local refinement for multi-objective optimization problems

Linas Litvinas

Institute of Computer Science, Vilnius University
Didlaukio str. 47, LT-08303 Vilnius, Lithuania
linas.litvinas@mif.vu.lt

Received: August 24, 2025 / **Revised:** November 22, 2025 / **Published online:** February 25, 2026

Abstract. Contemporary engineering and scientific problems often involve computationally intensive optimization tasks. This paper proposes a parallel version of the hybrid algorithm of the previously proposed Bayesian-based global search with Hooke–Jeeves local refinement for multi-objective optimization problems. The Bayesian-based hybrid algorithm has been complemented with multi-process data exchange using Open MPI to obtain a scalable parallel application. Each parallel process executes the Bayesian-based hybrid algorithm, and at the end, the Pareto optimal solutions of each process are merged into an aggregated set of Pareto optimal solutions. The master-slave pattern was used to parallelize the computations, where slave processes execute optimization algorithm and then send the obtained Pareto optimal solutions to the master process, which, in turn, also executes optimization algorithm and merges the Pareto optimal solutions of all processes. The developed parallel algorithm was tested under the same conditions previously used for testing other Bayesian algorithms to enable comparison of performance. Finally, the proposed parallel algorithm was published on the GitHub developer platform for code sharing.

Keywords: global optimization, Bayesian algorithm, Hooke–Jeeves algorithm, open-source parallel algorithm.

1 Introduction

Bayesian optimization is one of the most active areas of optimization research due to its well-founded theory and applications to complex design problems. In the development of methods for nonconvex black-box optimization, the Bayesian approach is one of the most actively used [1, 3, 19]. Although several algorithms based on Bayesian theory have been developed, the variety of modern applied problems requires new, appropriate modifications. Extending the application of Bayesian algorithms is difficult due to complexity related to matrix inversion. The partition-based implementation that was successful

in Lipschitz optimization [7, 12–16, 22] can be one way to reduce complexity. The complexity of hyperrectangle partition-based Bayesian algorithms has been shown to be lower than that of the standard implementations of the Bayesian algorithms since no matrix inversion is involved in partition-based algorithms [17, 18, 21]. However, hyperrectangle partition-based algorithms still have limitations in solving problems of higher dimensionality since the number of vertices of the hyperrectangle doubles with each new dimension. To address the referred issues, a hybrid multi-objective optimization algorithm has been presented that heuristically implements the Bayesian approach without using a formal model [9]. The proposed algorithm determines the next evaluation points using a bi-objective site-selection approach based on two heuristic criteria. One of these criteria is an estimate of the candidate point's distance to the Pareto front, and another is the uncertainty of this estimate [9]. These evaluations mimic the exploitation and exploration search strategy. As has already been shown (see, e.g., [5]), hybrid optimization algorithms are effective for complex optimal design problems; for example, in chemical engineering, hybrid algorithms incorporating the Hooke–Jeeves method have been successfully used for optimal design for at least the last 20 years [5]. At every iteration of proposed hybrid algorithm, the Hooke–Jeeves local search is used to refine the approximation of the Pareto front found by the global search algorithm phase [9]. Nevertheless, the ever-increasing computing intensity of computer models of various technologies (including chemical engineering and bioreactors) formulates higher requirements for optimization algorithms. One way to meet these requirements is through parallelization. This paper proposes a parallel version of Bayesian-based global search algorithm with Hooke–Jeeves local refinement for multi-objective optimization problems. The Bayesian-based hybrid algorithm is complemented by multi-process master-slave data exchange using Open MPI to provide a scalable parallel application [6]. The proposed parallel version of the hybrid algorithm is tested in conditions used to compare the performance of other Bayesian algorithms [20, 21]. The results were also compared with the results of the evolutionary optimization algorithm NSGA2 using test problems with up to 30 variables [2, 4].

2 The proposed parallel version of the hybrid algorithm

The following black-box multi-objective minimization problems will be considered:

$$\min F(x), \quad x \in A \subset \mathbb{R}^d,\ F(x) = \big(f_1(x), \ldots, f_m(x)\big)^{\mathrm{T}}.$$

The feasible region is the unit hypercube $A = [0,1]^d$ to which any hyperrectangular region is rescaled. Although the objective functions may have different ranges, the algorithm rescales them to a common range.

The proposed parallel algorithm consists of a sequential part complemented by a multi-process master-slave data exchange using Open MPI to provide a scalable parallel application [6]. The sequential section has two parts: a global search and a local refinement of the Pareto front approximation. The Bayesian global optimization strategy is preferred as it provides a rational balance between exploration and exploitation. The algorithm mimics the search strategy of the Bayesian algorithm without using a stochastic model

of the objective functions, thus avoiding the computational burden [9]. The global search phase is exchanged with the local refinement phase. The approximation of the Pareto front is refined by optimizing the surrogate function using the Hooke–Jeeves algorithm [9]. User-defined termination conditions terminate the sequential process of alternating global and local search. A detailed explanation of the sequential part of the algorithm and the pseudo-code is given in [9]. Only minor changes were made to the sequential part, while implementing the parallel algorithm. These changes include a refinement of each single-objective function from the optimized multi-objective function that occurs at each iteration of the algorithm, and stop conditions are checked after each phase of the algorithm. These changes can be analyzed in the GitHub repository, where the author of this paper gives an open-source C++ implementation of the proposed parallel algorithm [8]. The C++ programming language was used because it is highly suitable for high-performance computations [10]. In this paper, we call the sequential part of the algorithm HybAlg and use this name when explaining the parallel algorithm.

2.1 Parallel algorithm

The Bayesian-based sequential algorithm HybAlg has been complemented with multi-process data exchange using Open MPI to obtain a scalable parallel application. Open MPI allows processes to run on different nodes (or processors) in a distributed system and communicate with each other [6]. In the case of the proposed algorithm, each process runs on its node (or processor). Each parallel process executes the sequential HybAlg algorithm with the same stopping conditions, and at the end, the Pareto optimal solutions of each process are merged into an aggregated set of Pareto optimal solutions. The master-slave pattern was used to parallelize the computations, where slave processes execute optimization HybAlg algorithms and finally send the obtained Pareto optimal solutions to the master process, which, in turn, also executes optimization HybAlg algorithm and merges the Pareto optimal solutions of all processes. In this work, setting a parameter value to accept suboptimal Pareto solutions reduces the time complexity of the master-slave data exchange. This is because the time complexity of merging the final k-sized set of Pareto solutions from all processes becomes linear, $O(k)$. If the parameter value to accepts suboptimal Pareto solutions is set to false, the master process will merge all data from the slave processes, while keeping track of the Pareto optimality of the received solutions. In this case, the order of time complexity increases, and this may become a bottleneck factor to the final work time amount of the parallel algorithm.

A visual representation of the proposed parallel algorithm is given in Fig. 1. The proposed parallel algorithm uses the master-slave computation distribution strategy of one master and $n-1$ slaves. Open MPI allows all n processes running on different nodes (or processors) in a distributed system to communicate with each other [6]. All n processes run the sequential part of the algorithm, i.e. HybAlg. Since HybAlg is stochastic, all of the n processes will get different optimization results and, therefore, different Pareto optimal solutions. Finally, the master receives all Pareto optimal solutions from $n-1$ slaves and merges them into a joint Pareto set.

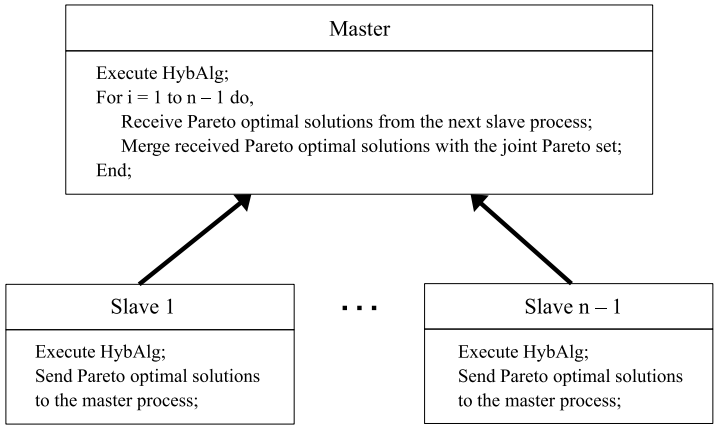

Figure 1. Flowchart of the proposed master-slave parallel algorithm of n processes with high-level pseudocode and arrows indicating data flow from $n-1$ slaves to a single master process.

2.2 Scalability analysis

The convergence analysis of the sequential part of the algorithm, i.e. HybAlg is given in [9]. The proposed parallel algorithm has the same properties since all n processes run HybAlg in parallel and independently. On the other hand, scalability analysis should be performed because modern engineering and scientific problems often involve computationally intensive optimization tasks, therefore, a large-scale optimization is important.

Scalability analysis involves analyzing the efficiency of parallel algorithms. The efficiency of the parallel algorithm running on p processors (E_p) is the speedup per processor:

$$E_p = \frac{S_p}{p},$$

where $S_p = T_{\text{sequential}}/T_p$ is the speedup with the number of processors p, where $T_{\text{sequential}}$ is the sequential execution time, while T_p is the parallel execution time with $p > 1$ processors or the sequential execution time with $p = 1$ processor. As already mentioned, all n processes run on different processors (or nodes) in the proposed parallel algorithm. Therefore, the equation $p = n$ holds, and the notations E_n and T_n will be used instead. Suppose the target workload is to run the stochastic sequential HybAlg algorithm n times with the same stopping conditions. In this case, the proposed parallel algorithm has an average runtime T_n. Having the average runtime T_1 of the stochastic sequential HybAlg algorithm with the same stopping conditions, the average execution time of the target workload in sequential mode can be evaluated: $T_{\text{sequential}} = T_1 \cdot n$. In this case, the mean efficiency (E_n) can be calculated:

$$E_n = \frac{T_{\text{sequential}}/T_n}{n} = \frac{(T_1 \cdot n)/T_n}{n} = \frac{T_1}{T_n}. \tag{1}$$

The efficiency of the proposed algorithm will be analyzed using the average runtime measurement values T_1 and T_n of the parallel algorithm from numerical experiments in the next section.

2.3 Parameter values of the parallel algorithm

Before giving the results of numerical experiments, some parameter values of the parallel algorithm should be specified. A full list of the parameters of the HybAlg algorithm can be found and analyzed in [9]; here only the main parameters related to numerical experiments are given:

1. n – number of parallel processes.
2. $N_{\max}$ – maximum number of function $F(\cdot)$ evaluations.
3. $I_{\max}$ – maximum iteration number.
4. N – number of initial solutions with function $F(\cdot)$ evaluations.
5. $q \cdot N$ – number of randomly generated candidate points in the decision space for new function evaluations.
6. p – part of function evaluations get by local random generation of candidate points near the current Pareto solutions compared to the whole feasible region A.
7. h_0 and h_n – step size parameters of the Hooke–Jeeves optimization algorithm. Full set of step sizes from the largest to the smallest: $\{0.8 \cdot 2^{-i},\ h_0 \leqslant i \leqslant h_n\}$, where the largest step size is $0.8 \cdot 2^{-h_0}$, and the smallest step size is $0.8 \cdot 2^{-h_n}$.
8. $Update$ – if set to true, the step size parameters h_0 and h_n are updated on the second and subsequent iterations of the global search with local refinement. For every Pareto optimal solution $x_P \in P_A$, the value h_0 is updated so that the largest step size would be approximately equal to the minimal Euclidean distance to another Pareto optimal solution. The value h_n is also updated so that the smallest step size would be smaller than the largest. This paper uses only the $true$ value.

3 Numerical experiments and analysis

Many engineering and scientific optimization problems have very limited function evaluation budgets. The proposed optimization algorithm was tested under such conditions and compared with Bayesian-rooted optimization algorithms [21]. In the case of a high-dimensional feasible region, the optimization algorithm was tested with the ZDT test suite, which has many (up to 30) decision variables [23]. The results have been compared with the results of the evolutionary optimization algorithm NSGA2 [2]. The proposed parallel optimization algorithm has been tested in sequential mode with a single process ($n = 1$) and in parallel mode with multiple processes (up to $n = 1024$).

3.1 Experiments with a low budget for function evaluation

The results of the proposed algorithm have been compared with the results of the Bayesian-rooted optimization algorithms: standard and partition-based implementations of the

P-algorithm when experiments have a low budget for function evaluation [21]. A number of test problems will be solved to illustrate the performance of optimization algorithms. The multi-objective function is defined as follows:

$$\begin{aligned} f_1(x) &= 1 - \exp\left\{-\sum_{i=1}^{d}\left(x_i - \frac{1}{\sqrt{d}}\right)^2\right\}, \\ f_2(x) &= 1 - \exp\left\{-\sum_{i=1}^{d}\left(x_i + \frac{1}{\sqrt{d}}\right)^2\right\}, \end{aligned} \tag{2}$$

where the feasible region is A: $-4 \leqslant x_1, x_2 \leqslant 4$, and $d = 2$. The next test problem is the Shekel functions:

$$\begin{aligned} f_1(x) &= -\frac{0.1}{0.1 + (x_1 - 0.1)^2 + 2(x_2 - 0.1)^2} \\ &\quad - \frac{0.1}{0.14 + 20((x_1 - 0.45)^2 + (x_2 - 0.55)^2)}, \\ f_2(x) &= -\frac{0.1}{0.15 + 40((x_1 - 0.55)^2 + (x_2 - 0.45)^2)} \\ &\quad - \frac{0.1}{0.1 + (x_1 - 0.3)^2 + (x_2 - 0.95)^2}, \end{aligned} \tag{3}$$

where the feasible region is A: $0 \leqslant x_1, x_2 \leqslant 1$.

The metric used in a recent publication related to Bayesian-based optimization algorithms was applied to compare the performance of the proposed algorithm [21]. The Epsilon Indicator (EI) is a metric that integrates evaluation of precision and spread [24]. EI is the maximum Euclidean distance between the true Pareto front and its closest neighbors from the found nondominated solutions:

$$\mathrm{EI} = \max_{\mathrm{F_{P^*}} \in \mathrm{P}_*} \min_{\mathrm{F_P} \in \mathrm{P}} \|\mathrm{F_P} - \mathrm{F_{P^*}}\|,$$

where P is a set of nondominated solutions found by the algorithm under consideration, and P^* is a set of solutions that represent the Pareto front well, i.e. the solutions are sufficiently dense and uniformly distributed over the Pareto front.

The proposed algorithm has been tested with a 100 function $F(x)$ evaluation budget per process. The proposed algorithm has been carried out with the following parameters for every process running HybAlg:

$$(N_{\max}, I_{\max}, N, q, p, h_o, h_n)^{\mathrm{T}} = (90, \infty, 20, 10000, 0.8, 2, 4)^{\mathrm{T}}.$$

The value of the parameter $N_{\max}$ is chosen 10% below the function evaluation budget, since the proposed algorithm is stochastic and the real number of function evaluations varies, so the given parameter value ensures that, on average, about a 100 function $F(x)$ evaluation budget is obtained per process. The same strategy for choosing the $N_{\max}$

Table 1. Mean values and standard deviations of the performance criteria EI of the P-algorithm, partition-based implementation of the P-algorithm and the proposed algorithm in sequential mode with a single process ($n = 1$) for test problems (2) and (3).

	P-algorithm		Partition-based	Proposed algorithm ($n = 1$)	
Problem (2)	0.2	0.034	0.092	0.145	0.02
Problem (3)	0.13	0.053	0.25	0.229	0.074

parameter value is used in this paper. The number of iterations $I_{\max}$ is chosen to be the maximum number that cannot be reached, so that it can be noted as infinite $I_{\max} = \infty$. The parameter $N = 20$ is the number of initial random evaluations of the function $F(x)$, and $q = 10000$ is the coefficient value, where $q \cdot N$ gives the number of randomly generated candidate points in the decision space for new evaluations of the function $F(x)$. The number of randomly generated candidate points should be high if the budget for function evaluation is small. That is why such a high value of q is chosen. The value of $p = 0.8$ is part of the function evaluations obtained by local generation of candidate points in the vicinity of the current Pareto optimal solutions, compared to the function evaluations obtained by global generation of candidate points in all feasible region. The Hooke–Jeeves optimization algorithm's step size parameters have the values of $h_0 = 2$ and $h_n = 4$. The same parameters were used for both of the test problems, except that the value of the parameter h_o was $h_o = 4$ for problem (3).

Since the proposed algorithm and P-algorithm are stochastic, the test problems were solved 100 times [11, 20]. Mean values and standard deviations of the performance criteria EI metric are presented in Table 1. Otherwise, the hyperrectangle partition-based P-algorithm is deterministic; therefore, its results are calculated from a single run [21]. The proposed algorithm in sequential mode with a single process ($n = 1$) shows creditable performance. When dealing with problem (2), the proposed algorithm gives better EI values than the P-algorithm. In the case of problem (3), the proposed algorithm gives better EI values than the hyperrectangle partition-based P-algorithm. As can be seen in Table 1, the proposed parallel algorithm in sequential mode gives similar results as in the case of the previous sequential implementation, although minor changes have been applied [9].

Table 2 gives the results of the proposed algorithm's parallel runs. Mean values and standard deviations of the performance criteria EI and mean values of the number of function evaluations, runtime T_n and efficiency E_n of the proposed parallel algorithm are given in Table 2. The proposed parallel algorithm runs HybAlg n times on different parallel processes with the same stopping conditions, so that for each n value, a single process performs approximately the same number of function evaluations as in the $n = 1$ case. The mean value of the performance criteria EI in Table 2 suggests that the precise Pareto front representation is obtained with $n = 256$ parallel processes, and only a small refinement is obtained in the case of $n = 512$ and $n = 1024$. In the case of the parallel algorithm with different number n of parallel processes, the runtime difference can be neglected from a practical point of view because for $n = 1024$ parallel processes, the runtime of the algorithm is within a few seconds. However, a scalability analysis should

Table 2. Mean values and standard deviations of the performance criteria EI and mean values of the number of function evaluations, runtime T_n and efficiency E_n of the proposed parallel algorithm with n processes for test problems (2) and (3).

n	EI	EI std	Function evaluations	T_n [s]	E_n
			Problem (2)		
1	0.145	0.02	106	0.186	1
2	0.112	0.018	211	0.244	0.762
4	0.078	0.013	424	0.293	0.634
8	0.053	0.009	851	0.367	0.506
16	0.038	0.007	1687	0.463	0.401
32	0.026	0.004	3391	0.535	0.347
64	0.017	0.002	6774	0.553	0.336
128	0.012	0.001	13595	0.636	0.292
256	0.007	0.0009	27175	0.684	0.271
512	0.005	0.0006	54336	0.743	0.25
1024	0.003	0.0003	108645	0.778	0.239
			Problem (3)		
1	0.229	0.074	104	0.387	1
2	0.158	0.069	213	0.491	0.788
4	0.103	0.047	415	0.594	0.651
8	0.067	0.018	832	0.697	0.555
16	0.045	0.01	1669	0.897	0.431
32	0.03	0.005	3337	1.099	0.352
64	0.022	0.004	6690	1.136	0.34
128	0.014	0.002	13342	1.249	0.309
256	0.009	0.001	26713	1.375	0.281
512	0.006	0.001	53355	1.455	0.265
1024	0.004	0.0005	106804	1.516	0.255

be performed. As already mentioned, the target workload is to run HybAlg n times with the same stopping conditions. The mean values of the efficiency E_n are calculated using Eq. (1) and are given in Table 2. As can be seen, a creditable efficiency $E_n \geqslant 0.5$ is observed with $n \leqslant 8$ parallel processes.

3.2 Experiments using test problems with a large number of decision variables

The results of the proposed algorithm were compared with the results of the evolutionary optimization algorithm NSGA2 in the case of high dimensionality of the feasible region [2]. The biobjective test suite with many decision variables was used to test the performance of the optimization algorithm: ZDT1, ZDT2, ZDT3 have 30 decision variables, while ZDT4 and ZDT6 have 10 decision variables [23].

To compare the performance of the proposed algorithm in the case of a high-dimensional feasible region, the same measure was used as in the publication on the evolutionary optimization algorithm NSGA2 [2]. The Inverted Generational Distance ($\mathrm{IGD}_{\mathrm{avg}}$) is a metric that integrates measures of approximation accuracy and spread. It is calculated as the average of the Euclidean distances between the true Pareto front and its nearest neighbor of the nondominated solutions found. The following equation can be used to

Table 3. Mean values and standard deviations of function evaluation count and performance criteria $\mathrm{IGD}_{\mathrm{avg}}$ of evolutionary optimization algorithm NSGA2 and the proposed algorithm in sequential mode with a single process ($n = 1$) for ZDT test problems.

Problem	μ_{eval}	σ_{eval}	$\mu_{\mathrm{IGD}_{\mathrm{avg}}}$	$\sigma_{\mathrm{IGD}_{\mathrm{avg}}}$
		NSGA2		
ZDT1	17098	2393.78	0.006	0.0007
ZDT2	17657	1528.04	0.006	0.0008
ZDT3	16559	1899.07	0.007	0.0078
ZDT4	24451	2864.01	0.006	0.0013
ZDT6	24833	1084.74	0.004	0.0002
		Proposed algorithm ($n = 1$)		
ZDT1	15432	33.02	0.003	0.0016
ZDT2	15921	31.7	0.006	0.0034
ZDT3	14951	36	0.002	0.0013
ZDT4	22054	41.67	0.108	0.0284
ZDT6	22362	11.56	0.003	0.0013

express the metric:

$$\mathrm{IGD}_{\mathrm{avg}} = \frac{1}{|P^*|} \sum_{F_{P^*} \in P^*} \min_{F_P \in P} \|F_P - F_{P^*}\|,$$

where P is a set of nondominated solutions that are found by the algorithm under consideration, and P^* is a set of solutions that well represent the Pareto front. That is, the solutions are sufficiently dense and uniformly distributed over the Pareto front [9].

The proposed algorithm was tested with a fixed function $F(x)$ evaluation budget per process. The parameter $N_{\max}$ value is selected to be 10% below the function evaluation budget. Consequently, the function evaluation budget is not exceeded on average per process. The following parameters were utilized in the execution of the proposed algorithm for each process running HybAlg:

$$(I_{\max}, N, q, p, h_o, h_n)^{\mathrm{T}} = (\infty, 100, 1, 0.8, 2, 8)^{\mathrm{T}}.$$

The number of iterations $I_{\max} = \infty$ is chosen as the maximum number that cannot be reached. The number of random function evaluations at the start is $N = 100$. The value of $q = 1$ is the coefficient value for the number $q \cdot N$ of randomly generated candidate points in decision space for new function evaluations. A low coefficient value of q is chosen because the number of randomly generated candidate points cannot be high in the case of a large function evaluation budget, as large number of candidate points would place a burden on the proposed algorithm. The value $p = 0.8$ is part of function evaluations of local candidate points generation near the current Pareto optimal solutions, compared to function evaluations from generation of candidate points across the whole feasible region. The Hooke–Jeeves optimization algorithm's step size parameters have the values $h_0 = 2$ and $h_n = 8$.

The proposed algorithm and the evolutionary optimization algorithm NSGA2 are stochastic. Therefore, the test problems were solved several times [2, 4]. The means and standard deviations of $\mathrm{IGD}_{\mathrm{avg}}$ metric are presented in Table 3. Note that the normalization

of function values is performed to calculate $\mathrm{IGD}_{\mathrm{avg}}$ [2,9]. Table 3 shows the results of the $\mathrm{IGD}_{\mathrm{avg}}$ metric of the proposed algorithm in sequential mode with a single process ($n = 1$) in comparison with the results of the evolutionary optimization algorithm NSGA2 after 51 experiments [2]. The proposed algorithm demonstrates good performance. Table 3 illustrates that for the test problems ZDT1, ZDT2, ZDT3, and ZDT6, the proposed algorithm has a better or equal performance compared to the results of the evolutionary optimization algorithm NSGA2. In the case of the ZDT4 test problem, the proposed algorithm's

Table 4. Mean values and standard deviations of the performance criteria $\mathrm{IGD}_{\mathrm{avg}}$ and mean values of the number of function evaluations, runtime T_n and efficiency E_n of the proposed parallel algorithm with n processes for ZDT test problems.

n	$\mu_{\mathrm{IGD}_{\mathrm{avg}}}$	$\sigma_{\mathrm{IGD}_{\mathrm{avg}}}$	Function evaluations	T_n [s]	E_n
			ZDT1		
1	0.003	0.001	15432	9.4	1
2	0.001	0.0002	30870	12.6	0.74
4	0.0005	0.0001	61731	14	0.67
8	0.0002	$2.5 \cdot 10^{-5}$	123461	18.1	0.51
16	0.0001	$8.4 \cdot 10^{-6}$	246956	17.9	0.52
32	$5.2 \cdot 10^{-5}$	$3.3 \cdot 10^{-6}$	493863	20.5	0.45
			ZDT2		
1	0.006	0.003	15921	9.1	1
2	0.002	0.001	31837	10.2	0.89
4	0.001	0.0004	63666	13.3	0.68
8	0.0004	0.0001	127292	16.3	0.55
16	0.0002	$4.8 \cdot 10^{-5}$	254653	18.8	0.48
32	0.0001	$1.4 \cdot 10^{-5}$	509232	23.3	0.39
			ZDT3		
1	0.002	0.001	14951	7.3	1
2	0.001	0.0003	29907	7.3	1
4	0.0005	$8.3 \cdot 10^{-5}$	59822	9.1	0.8
8	0.0002	$3.1 \cdot 10^{-5}$	119686	11.2	0.65
16	0.0001	$1.5 \cdot 10^{-5}$	239317	13	0.56
32	$6.7 \cdot 10^{-5}$	$5.5 \cdot 10^{-6}$	478633	14.4	0.5
			ZDT4		
1	0.108	0.028	22054	7.2	1
2	0.079	0.02	44117	8.1	0.88
4	0.05	0.016	88213	9.8	0.73
8	0.028	0.007	176460	10.3	0.69
16	0.016	0.004	352878	10.9	0.66
32	0.008	0.001	705767	13	0.55
			ZDT6		
1	0.003	0.001	22362	11	1
2	0.001	0.0002	44735	13.7	0.8
4	0.0006	0.0001	89456	17.2	0.63
8	0.0002	$4.6 \cdot 10^{-5}$	178916	19.2	0.57
16	0.0001	$1.7 \cdot 10^{-5}$	357839	20.1	0.54
32	$6.9 \cdot 10^{-5}$	$5.3 \cdot 10^{-6}$	715683	24.2	0.45

Continued on next page

Table 4 (*Continued from previous page*)

n	$\mu_{\mathrm{IGD}_{\mathrm{avg}}}$	$\sigma_{\mathrm{IGD}_{\mathrm{avg}}}$	Function evaluations	T_n [s]	E_n
		ZDT1			
64	$2.5 \cdot 10^{-5}$	$1.7 \cdot 10^{-6}$	987768	24	0.39
128	$1.3 \cdot 10^{-5}$	$6.4 \cdot 10^{-7}$	$1.9 \cdot 10^{6}$	25.4	0.37
256	$6.5 \cdot 10^{-6}$	$3 \cdot 10^{-7}$	$3.9 \cdot 10^{6}$	26.9	0.34
512	$3.2 \cdot 10^{-6}$	$1.7 \cdot 10^{-7}$	$7.9 \cdot 10^{6}$	27.6	0.34
1024	$1.6 \cdot 10^{-6}$	$8.8 \cdot 10^{-8}$	$1.5 \cdot 10^{7}$	32.3	0.29
		ZDT2			
64	$5.8 \cdot 10^{-5}$	$5.8 \cdot 10^{-6}$	$1 \cdot 10^{6}$	27.7	0.32
128	$3 \cdot 10^{-5}$	$2.8 \cdot 10^{-6}$	$2 \cdot 10^{6}$	29.9	0.3
256	$1.5 \cdot 10^{-5}$	$1.1 \cdot 10^{-6}$	$4 \cdot 10^{6}$	32.3	0.28
512	$7.5 \cdot 10^{-6}$	$4.6 \cdot 10^{-7}$	$8.1 \cdot 10^{6}$	33.2	0.27
1024	$3.7 \cdot 10^{-6}$	$2.3 \cdot 10^{-7}$	$1.6 \cdot 10^{7}$	38.3	0.23
		ZDT3			
64	$3.3 \cdot 10^{-5}$	$2.3 \cdot 10^{-6}$	957248	16.5	0.44
128	$1.6 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	$1.9 \cdot 10^{6}$	18.1	0.4
256	$8.2 \cdot 10^{-6}$	$4.9 \cdot 10^{-7}$	$3.8 \cdot 10^{6}$	19.5	0.37
512	$4 \cdot 10^{-6}$	$2.3 \cdot 10^{-7}$	$7.6 \cdot 10^{6}$	20.3	0.35
1024	$2 \cdot 10^{-6}$	$9.9 \cdot 10^{-8}$	$1.5 \cdot 10^{7}$	24.1	0.3
		ZDT4			
64	0.004	0.0009	$1.4 \cdot 10^{6}$	15	0.48
128	0.002	0.0004	$2.8 \cdot 10^{6}$	16.4	0.43
256	0.0008	0.0001	$5.6 \cdot 10^{6}$	19.2	0.37
512	0.0003	$4.1 \cdot 10^{-5}$	$1.1 \cdot 10^{7}$	22.5	0.32
1024	0.0001	$1.5 \cdot 10^{-5}$	$2.2 \cdot 10^{7}$	27	0.26
		ZDT6			
64	$3.4 \cdot 10^{-5}$	$2.5 \cdot 10^{-6}$	$1.4 \cdot 10^{6}$	27.4	0.4
128	$1.7 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	$2.8 \cdot 10^{6}$	29.9	0.36
256	$8.6 \cdot 10^{-6}$	$4.1 \cdot 10^{-7}$	$5.7 \cdot 10^{6}$	31.6	0.34
512	$4.3 \cdot 10^{-6}$	$2.4 \cdot 10^{-7}$	$1.1 \cdot 10^{7}$	32.5	0.33
1024	$2.1 \cdot 10^{-6}$	$1.1 \cdot 10^{-7}$	$2.2 \cdot 10^{7}$	37.4	0.29

performance was worse compared to the results of the evolutionary optimization algorithm NSGA2. As can be seen in Table 3, the proposed parallel algorithm in sequential mode gives similar results as in the case of the previous sequential implementation, although minor changes have been applied [9].

Table 4 gives the results of the proposed algorithm's parallel runs. The mean values and standard deviations of the performance criteria $\mathrm{IGD}_{\mathrm{avg}}$ and the mean values of the number of function evaluations, runtime T_n and efficiency E_n of the proposed parallel algorithm are given in Table 4. As mentioned, the proposed parallel algorithm runs HybAlg n times on different parallel processes with the same stopping conditions; therefore, for each n value, a single process performs approximately the same number of function evaluations as in the $n = 1$ case. The mean values of the performance criteria $\mathrm{IGD}_{\mathrm{avg}}$ in Table 4 suggest that the performance criteria $\mathrm{IGD}_{\mathrm{avg}}$ is approximately halved by doubling the number of parallel processes n, and in this way, a precise Pareto front representation can be obtained. For the ZDT4 problem, the sequential run of the proposed algorithm

gives an insufficient result, i.e. the Pareto front representation has insufficient precision (Table 3). By running the algorithm in parallel mode with $n = 256$ processes, the Pareto front representation with sufficient precision is obtained (Table 4). Next, a scalability analysis needs to be performed. As already mentioned, the target workload is to run HybAlg n times with the same stopping conditions. The mean values of the efficiency E_n are calculated using Eq. (1) and are presented in Table 4. As can be observed, a creditable efficiency of $E_n \geqslant 0.5$ is achieved with up to $n \leqslant 16$ parallel processes for ZDT1 and ZDT6 test problems; up to $n \leqslant 8$ parallel processes for the ZDT2 test problem; and up to $n \leqslant 32$ parallel processes for ZDT3 and ZDT4 test problems.

4 Conclusions

The parallel algorithm of Bayesian-based global search with Hooke–Jeeves local refinement for multi-objective optimization problems was proposed. The master-slave pattern, utilizing Open MPI, was employed to parallelize the computations, where slave processes execute optimization HybAlg algorithm and subsequently send the obtained Pareto optimal solutions to the master process. The parallel algorithm was tested under an extremely low function evaluation budget, where the algorithm in sequential mode delivered a creditable performance when compared to the results of Bayesian rooted optimization algorithms. It was also tested on the ZDT test suite with many decision variables, and the results of numerical experiments demonstrated a good performance of the algorithm in sequential mode compared to the results of the widely used evolutionary optimization algorithm NSGA2. In the case of an extremely low function evaluation budget, the precise Pareto front representation is obtained with $n = 256$ parallel processes. With many decision variables, the performance criteria $\mathrm{IGD}_{\mathrm{avg}}$ is approximately halved by doubling the number of parallel processes n to achieve the necessary precision of the Pareto front representation. In the case of an extremely low function evaluation budget, a creditable efficiency $E_n \geqslant 0.5$ is observed with $n \leqslant 8$ parallel processes, and the runtime is less than one second in sequential mode. Dealing with many decision variables, a creditable efficiency is observed with $n \leqslant 16$ or $n \leqslant 32$ parallel processes for almost all ZDT test problems, and the runtime is about several seconds in sequential mode. The proposed parallel algorithm is more efficient when the sequential mode takes a long time to execute. This suggests that the algorithm is well suited for computationally intensive optimization tasks that are commonly encountered in contemporary engineering and scientific problems. Overall, the proposed parallel algorithm has the desired features of creditable performance with an extremely low function evaluation budget and with many decision variables, as well as scalability using parallel processes. These features make it a universal and suitable option for many multi-objective optimization tasks. Moreover, the algorithm was published on the GitHub developer platform for sharing code and creating new branches with advanced features. Further plans include researching the efficiency of the algorithm in the case of multi-objective optimization of the characteristics of a batch-stirred bioreactor modeled by expensive black-box functions for which the execution time of the proposed algorithm in sequential mode can be up to several hundred hours.

Conflicts of interest. The authors declares no conflicts of interest.

Acknowledgment. The author would like to thank Prof. Habil. Dr. Antanas Žilinskas for providing guidance, feedback and support throughout the research and preparation of the paper. The author would also like to thank the Faculty of Mathematics and Informatics at Vilnius University and its Digital Science and Computing Center for providing access to their supercomputing resources.

References

1. F. Archetti, A. Candelieri, *Bayesian Optimization and Data Science*, Springer, Cham, 2019, https://doi.org/10.1007/978-3-030-24494-1.
2. J. Blank, K. Deb, A running performance metric and termination criterion for evaluating evolutionary multi- and many-objective optimization algorithms, in *2020 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, Piscataway, 2020, pp. 1–8, https://doi.org/10.1109/CEC48606.2020.9185546.
3. J. Cui, B. Yang, Survey on Bayesian optimization methodology and applications, *J. Softw.*, **29**(10):3068–3090, 2007, https://doi.org/10.13328/j.cnki.jos.005607.
4. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.*, **6**(2):182–197, 2002, https://doi.org/10.1109/4235.996017.
5. E.S. Fraga, A. Žilinskas, Evaluation of hybrid optimization methods for the optimal design of heat integrated distillation sequences, *Adv. Eng. Softw.*, **34**(2):73–86, 2003, https://doi.org/10.1016/S0965-9978(02)00125-4.
6. R. Graham, G. Shipman, B. Barrett, R. Castain, G. Bosilca, A. Lumsdaine, Open MPI: A high-performance, heterogeneous MPI, in *2006 IEEE International Conference on Cluster Computing*, IEEE, Piscataway, 2006, pp. 1–9, https://doi.org/10.1109/CLUSTR.2006.311904.
7. D.R. Jones, C.D. Perttunen, B.E. Stuckman, Lipschitzian optimization without the Lipschitz constant, *J. Optim. Theory Appl.*, **79**:157–181, 1993, https://doi.org/10.1007/BF00941892.
8. L. Litvinas, An open-source parallel algorithm of Bayesian-based global search with Hooke–Jeeves local refinement for multi-objective optimization problems; implementation source code, https://github.com/ok9/parallel.git.
9. L. Litvinas, A hybrid of Bayesian-based global search with Hooke–Jeeves local refinement for multi-objective optimization problems, *Nonlinear Anal. Model. Control*, **27**(3):534–555, 2022, https://doi.org/10.15388/namc.2022.27.26558.
10. S. Meyers, *Effective Modern C++*: 42 *Specific Ways to Improve Your Use of C++11 and C++14*, O'Reilly, Sebastopol, CA, 2014.
11. P.M. Pardalos, A. Žilinskas, J. Žilinskas, *Non-Convex Multi-Objective Optimization*, Springer, Cham, 2017, https://doi.org/10.1007/978-3-319-61007-8.
12. R. Paulavičius, Ya.D. Sergeyev, D.E. Kvasov, J. Žilinskas, Globally-biased DISIMPL algorithm for expensive global optimization, *J. Glob. Optim.*, **59**:545–567, 2014, https://doi.org/10.1007/s10898-014-0180-4.

13. R. Paulavičius, J. Žilinskas, *Simplicial Global Optimization*, SpringerBriefs Optim., Springer, New York, 2014, https://doi.org/10.1007/978-1-4614-9093-7.

14. J. Pintér, *Global Optimization in Action. Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications*, Nonconvex Optim. Appl., Vol. 6, Springer, New York, 1996, https://doi.org/10.1007/978-1-4757-2502-5.

15. Ya.D. Sergeyev, D.E. Kvasov, Global search based on efficient diagonal partitions and a set of Lipschitz constants, *SIAM J. Optim.*, **16**(3):910–937, 2006, https://doi.org/10.1137/040621132.

16. L. Stripinis, R. Paulavičius, Review and computational study on practicality of derivative-free DIRECT-type methods, *Informatica*, **36**(1):141–174, 2025, https://doi.org/10.15388/24-INFOR548.

17. A. Žilinskas, R. Baronas, L. Litvinas, L. Petkevičius, Multi-objective optimization and decision visualization of batch stirred tank reactor based on spherical catalyst particles, *Nonlinear Anal. Model. Control*, **24**(6):1019–1033, 2019, https://doi.org/10.15388/NA.2019.6.10.

18. A. Žilinskas, G. Gimbutienė, A hybrid of Bayesian approach based global search with clustering aided local refinement, *Commun. Nonlinear Sci. Numer. Simul.*, **78**:104857, 2019, https://doi.org/10.1016/j.cnsns.2019.104857.

19. A. Zhigljavsky, A. Žilinskas, *Bayesian and High-Dimensional Stochastic Optimization*, Springer, Cham, 2021, https://doi.org/10.1007/978-3-030-64712-4.

20. A. Žilinskas, A statistical model-based algorithm for 'black-box' multi-objective optimisation, *Int. J. Syst. Sci.*, **45**(1):82–93, 2014, https://doi.org/10.1080/00207721.2012.702244.

21. A. Žilinskas, L. Litvinas, A partition based bayesian multi-objective optimization algorithm, in Ya.D. Sergeyev, D.E. Kvasov (Eds.), *Numerical Computations: Theory and Algorithms*, Springer, Cham, 2020, pp. 511–518, https://doi.org/10.1007/978-3-030-40616-5_50.

22. A. Žilinskas, J. Žilinskas, Adaptation of a one-step worst-case optimal univariate algorithm of bi-objective Lipschitz optimization to multidimensional problems, *Commun. Nonlinear Sci. Numer. Simul.*, **21**(1–3):89–98, 2015, https://doi.org/10.1016/j.cnsns.2014.08.025.

23. E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evol. Comput.*, **8**(2):173–195, 2000, https://doi.org/10.1162/106365600568202.

24. E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, *IEEE Trans. Evol. Comput.*, **7**(2):117–132, 2003, https://doi.org/10.1109/TEVC.2003.810758.