Numerical Results for the Multi-objective Trust Region Algorithm MHT*

Jana Thomann† and Gabriele Eichfelder‡

August 20, 2018

Abstract

A set of 78 test examples is presented for the trust region method MHT described in [19]. It is designed for multi-objective heterogeneous optimization problems where one of the objective functions is an expensive black-box function, for example given by a time-consuming simulation. The presence of expensive functions is artificially introduced in the test problems by defining one of the objective functions as expensive. For these test problems, numerical results are presented. As an appendix, the computation of interpolation points for model functions used in MHT is described.

Key Words: multi-objective optimization, heterogeneous optimization, test set, test problems, trust region algorithm

1 Introduction

In many applications multi-objective optimization problems arise and the objective functions are not always given analytically, but sometimes only as a black-box, for example by a simulation. An additional difficulty arises if this black-box function is expensive, meaning that one function evaluation demands high computational effort. The trust region algorithm presented in [19] is designed to consider such multi-objective heterogeneous optimization problems where one of the objectives is an expensive black-box function, not given analytically but only by a time-consuming simulation. The simulation only gives function values. Derivative information is not available with reasonable effort and therefore not used. The other objective functions are so-called cheap functions which are analytically given, easy to compute and derivatives are easily available. The general optimization problem that is considered is described as

$$\min_{x\in\Omega} f(x) \qquad (MOP)$$

*This work was funded by DFG under no. GRK 1567.

†Institute for Mathematics, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany, jana.thomann@tu-ilmenau.de

‡Institute for Mathematics, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany, Gabriele.Eichfelder@tu-ilmenau.de

with $\Omega \subseteq \mathbb{R}^n$ and $f(x) = (f_1(x), ..., f_q(x))^\top$. The objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are assumed to be twice continuously differentiable for all $i = 1, ..., q$ and $\max_{i=1,...,q} f_i(x)$ is assumed to be bounded from below. The constraint set Ω contains either box constraints or is the whole domain.
The multi-objective heterogeneous trust region algorithm MHT as presented in [19] is based on the trust region approach and generalizes it to multi-objective problems. Following the trust region concept the computations are restricted to a local area in every iteration and the functions are replaced by local models. The search direction is generated in the image space and local ideal points of the model functions are used.
In section 2 an overview of the test problems used for the algorithm in [19] is presented followed by numerical results in section 3. In the appendix self-chosen test problems are described and the computation of the interpolation points used for building the interpolation models in the algorithm of [19] is described.

2 Test Problems

The following set of 78 test problems is based on test problems from the literature for general multiobjective and derivative-free algorithms [4, 12, 5, 20, 2, 1, 6, 13, 10, 11, 14, 15, 17, 21] and completed with some self-chosen problems. All considered problems are test problems and do not involve an actual expensive function. For these problems the efficient points can be computed which is necessary to compare the results of the algorithm to the actual efficient solutions. For evaluating the results one of the functions is declared as expensive and the amount of function evaluations for this function is counted.
Among the test examples are quadratic and nonquadratic functions, convex and nonconvex problems, either unconstrained or with box constraints. Table 1 shows an overview of all 78 considered test problems with information about the dimension of the domain (n), the constraints and the convexity of the problem. It also includes information about the geometry of the Pareto front, the set of all nondominated points (convex, nonconvex, disconnencted). We say the Pareto front PF is convex (nonconvex) if the set $PF + \mathbb{R}^2_+$ is convex (nonconvex). Besides, the table contains in its last column which of the objective functions is declared as expensive for the test runs of MHT. If there are significant differences regarding the difficulty of the functions, the more difficult function is declared as expensive.
Some of the test examples are scalable and different values for n are considered. They are listed in the table. For every test problem several randomly generated, but fixed, starting points were used. One test instance is defined as one test problem with one starting point. In total, 802 test instances have been considered, among them 348 convex and 454 nonconvex instances.

name	n	constraints	convexity	PF	exp.
		bi-objective test problems (q=2)			
BK1 [12, 4]	2	box	conv.	conv.	f_1
CL1 [1, 4]	4	box	nonc.	conv.	f_1
Deb41 [5, 4]	2	box	conv.	conv.	f_2
Deb53 [5, 4]	2	unc.	nonc.	nonc.	f_2
Deb513 [5, 4]	2	box	nonc.	discon.	f_2
Deb521b [5, 4]	2	box	nonc.	nonc.	f_2
DG01 [12, 4]	1	box	nonc.	conv.	f_1
DTLZ1 [6, 4]	2	box	nonc.	conv.	f_1
ex005 [13, 4]	2	box	nonc.	nonc.	f_2
Far1 [12, 4]	2	box	nonc.	nonc.	f_1
FF [10]	2,3,4,5	box	nonc.	nonc.	f_1
Fonseca [11]	2	box	nonc.	nonc.	f_1
IM1 [12, 4]	2	box	nonc.	conv.	f_1
Jin1 [14, 4]	2,3,4,5,10,20,30,40,50	box	conv.	conv.	f_1
Jin2 [14, 4]	2,3,4,5	box	nonc.	conv.	f_2
Jin3 [14, 4]	2,3,4,5	box	nonc.	nonc.	f_2
Jin4 [14, 4]	2,3,4,5	box	nonc.	nonc.	f_2
JOS3 [12]	3	box	conv.	conv.	f_1
Kursawe [15, 4]	3	box	nonc.	discon.	f_1
Laumanns [12]	2	box	conv.	conv.	f_1
LE1 [12, 4]	2	box	nonc.	nonc.	f_1
Lis [12, 2]	2	box	nonc.	nonc.	f_1
lovison1 [4, 16]	2	box, unc.	conv.	conv.	f_1
lovison2 [4, 16]	2	box, unc.	nonc.	nonc.	f_2
lovison3 [4, 16]	2	box, unc.	nonc.	conv.	f_1
lovison4 [4, 16]	2	box, unc.	nonc.	nonc.	f_1
MOP1 [12, 4]	1	unc.	conv.	conv.	f_1
Schaffer2 [17]	1	box	nonc.	discon.	f_1
T1	2	unc.	conv.	conv.	f_1
T2	2	unc.	nonc.	nonc.	f_2
T3	2	box	conv.	conv.	f_2
T4	2,3,4,5,10,20,30,40,50	box	conv.	conv.	f_1
T5	2	box	conv.	conv.	f_1
T6	2	box	conv.	conv.	f_1
T7	3	box	conv.	conv.	f_1
VU1 [12, 4]	2	box	nonc.	conv.	f_1
VU2 [12, 4]	2	box	conv.	conv.	f_2
ZDT1 [21, 4]	4	box	nonc.	conv.	f_2
ZDT2 [21, 4]	4	box	nonc.	nonc.	f_2
ZDT3 [21, 4]	4	box	nonc.	discon.	f_2
ZDT4 [21, 4]	2	box	nonc.	conv.	f_2
ZDT6 [21, 4]	4	box	nonc.	nonc.	f_2

test problems with q=3 objective functions					
FES2 [12, 4]	10	box	nonc.	nonc.	f_3
IKK1 [12, 4]	2	box	conv.	conv.	f_1
T8	3	box	conv.	conv.	f_3
ZLT1 [12, 4]	4	box	conv.	conv.	f_3

Table 1: Test Problems

3 Numerical Results

The multi-objective trust region algorithm MHT as presented in [19] has been implemented in Matlab (version 2017a) and tested for the multi-objective problems with two or three objective functions from section 2. We used the realization of the trust region update as described in [19, section 5.2] with the parameters $\eta_1 = 0.001, \eta_2 = 0.9$. The stopping criterion is implemented according to [19, section 5.1]. It uses a maximum number of function evaluations given by the user, the size of the trust region and a necessary condition for local weak efficiency. Furthermore, model information for the as expensive declared function is reused as often as possible to save function evaluations, see [19, section 5.4].
We compared MHT with two other methods. On the one hand, since MHT computes only one Pareto critical point and does not approximate the set of efficient points, we used the weighted sum approach with equal weights and apply EFOS (Expensive Function Optimization Solver) [18] to it with the predefined standard parameters. It is a solution method for expensive, simulation-based scalar optimization problems also using the trust region approach. As a stopping criterion a criticality measure using the gradients of the model functions is applied in conjunction with a validity criterion for the models. For convex multi-objective optimization problems every efficient point can be computed by a weighted sum of the objectives with suitable weights. For nonconvex problems only a subset of the efficient points can be computed. This needs to be regarded when comparing the results.
On the other hand, and to circumvent the disadvantages of the weighted sum approach, the multi-objective method DMS [4] is used as a comparative method. It is a direct search approach and therefore derivative-free and suitable for expensive functions. It approximates the whole set of efficient points, but offers also the option to compute only one efficient point. We used the latter option with the predefined standard parameters varying the maximum number of function evaluations. As a stopping criterion DMS uses a maximum number of function evaluations given by the user and the step size for the search step. If the step size is lower than the predefined value (10^{-3}), DMS stops. Furthermore, DMS includes a method to compute starting points on its own which is chosen by default. To use the starting point the user passes as input to the algorithm, the parameter 'list' in the parameter file needs to be changed from the predefined value 3 to 0.
Of course also the way of the implementation of the algorithms influence the performance for the test problems. In the currently available implementation of EFOS often internal

errors occur and runs are terminated without having computed a solution.
As a main comparison criterion we use the number of function evaluations until the considered method terminates and set the maximum number of allowed function evaluations for all algorithms to 2000. Firstly, we present some selected test instances to illustrate the procedure of MHT and to compare the results to the methods DMS and EFOS. In the end of this section, we present performance profiles for all considered test instances.

3.1 Convex Test Problems

At first we consider the quadratic, convex test problem (BK1) from [12, 4] given by

$$\min_{x\in\Omega}\begin{pmatrix} f_1(x)\\ f_2(x)\end{pmatrix} = \min_{x\in[-5,10]^2}\begin{pmatrix} x_1^2+x_2^2\\ (x_1-5)^2+(x_2-5)^2\end{pmatrix} \tag{BK1}$$

to illustrate the procedure of MHT. For this test problem function f_1 is declared as expensive function. For all instances of this test problem MHT and EFOS compute efficient points, DMS only for most of the instances. EFOS computes for different starting points always the same efficient point, whereas MHT and DMS generate different efficient points. MHT needs 12-13 expensive function evaluations and therefore significantly less than EFOS (57-73) and DMS (41-61).
Figure 1 shows one test result for MHT (domain top left, image space top right), EFOS (domain middle left, image space middle right) and DMS (domain bottom left, image space bottom right). The domain resp. image set is represented by scattered gray points, the starting point is marked black and the solution is marked orange. For MHT the iteration points are marked black and connected by a dotted line, the interpolation points that are evaluated to compute the model functions are marked as unfilled circles. In the domain the trust regions are depicted as gray shaded, transparent circles (the more areas overlap, the darker the gray shade). For EFOS and DMS it is not possible to distinguish between iteration points and further evaluated points during the iterations. Thus, all points evaluated for computing the solution are marked as unfilled circles and only the starting point and the solution are highlighted as for MHT.
For quadratic functions the quadratic interpolation model used for the expensive function in MHT is exact. Thus, the model built in the beginning of the algorithm is reused in all following iterations. Only in the last iteration, when it is checked if the iteration point is a Pareto critical point, the model function is recomputed in a local area. This can be seen in the top left part of Figure 1, where the interpolation points are situated in the first and in the last trust region. The interpolation points are also close to the iteration points in the image space which the top right figure shows. Both figures illustrate the local search strategy of MHT.
EFOS computes more points than MHT and they are more spread both over the image space and the domain. During the run even infeasible points are generated. This can be seen only in the domain (middle left) since for illustrative reasons we used the same range in the image space for all figures on the right.
DMS also computes more points than MHT, but they are not spread over the domain as for EFOS, but accumulate in a local area. Apart from this, the bottom left figure

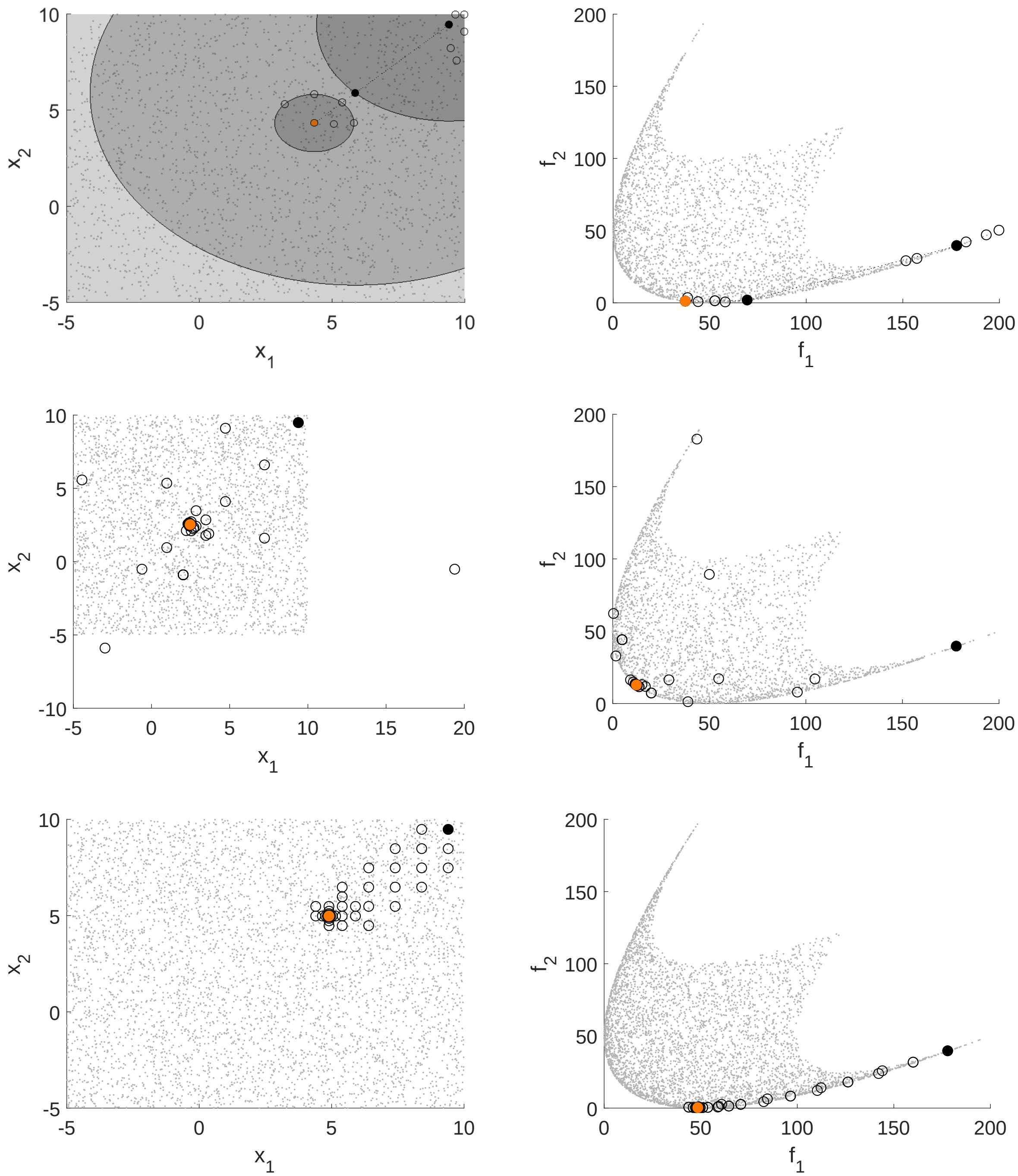


Figure 1: Test run for (BK1) for MHT (top), EFOS (middle) and DMS (bottom)

illustrates the search along the coordinate directions. In the last iterations the step size decreases.

In one run of MHT one Pareto critical point is computed. In general, different starting points generate different Pareto critical points due to the search strategy. A multistart approach with randomly generated starting points is illustrated in Figure 2. The starting points are marked as unfilled circles and the obtained points are marked black. Figure 2 shows that MHT (top left) and DMS (bottom) compute different nondominated points, whereas EFOS (top right) generates only one nondominated point. A reason for this is the weighted sum approach. Furthermore, not all resulting points from DMS are efficient points, some have still a large distance to the Pareto front.

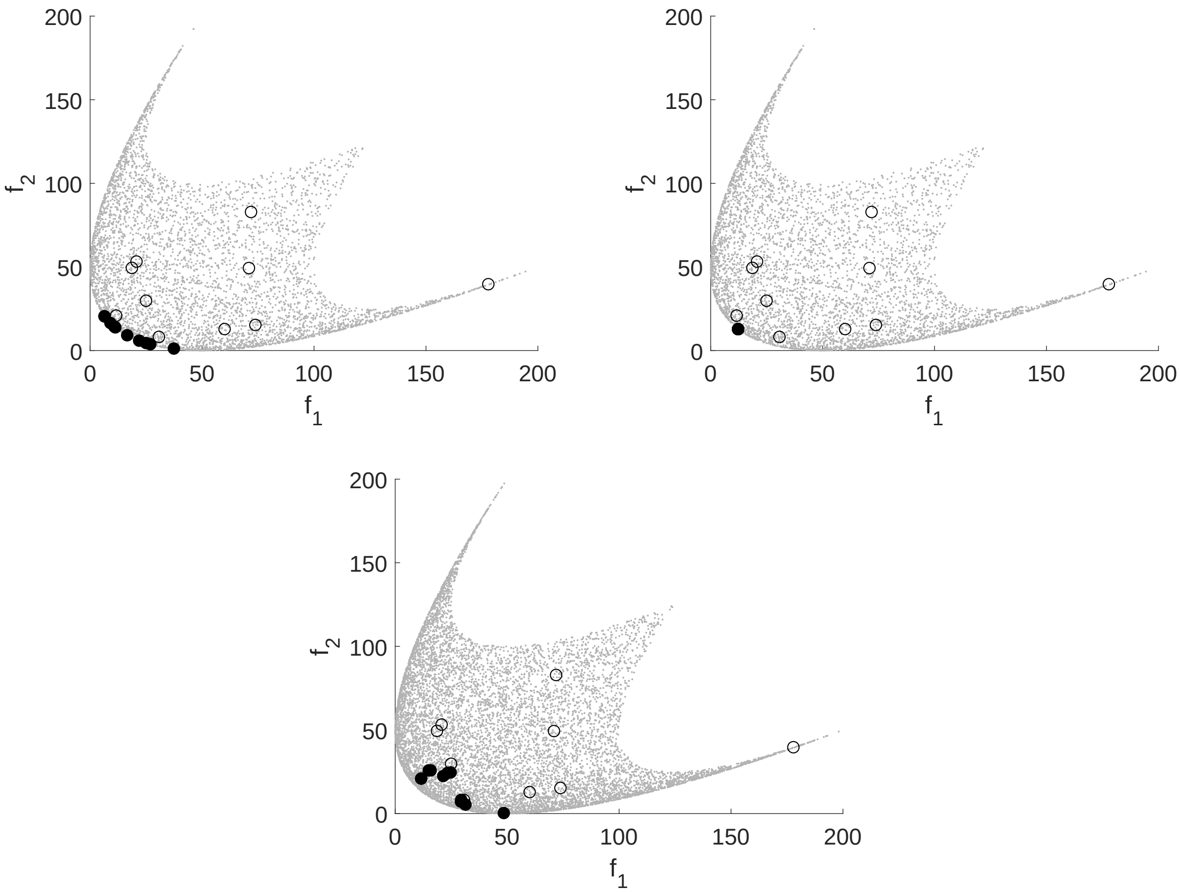


Figure 2: Multistart approach for (BK1) for MHT (top left), EFOS (top right) and DMS (bottom)

As second test example we consider the convex, but not quadratic problem (T6) given by

$$\min_{x\in\Omega} \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \min_{x\in(0,100]^2} \begin{pmatrix} -\ln(x_1) - \ln(x_2) \\ x_1^2 + x_2 \end{pmatrix}. \tag{T6}$$

For all test instances of this optimization problem EFOS is prematurely canceled due to an internal error. Consequently, we only show results for DMS and MHT. The as expensive declared function f_1 is not quadratic and therefore the interpolation model used in MHT is not exact. Though, the algorithm reuses old model information as often as possible which is illustrated for one specific instance, see the top left (domain) and top right image (image space) in Figure 3. The bottom left and right image show the result for DMS in the domain and the image space with the same starting point.

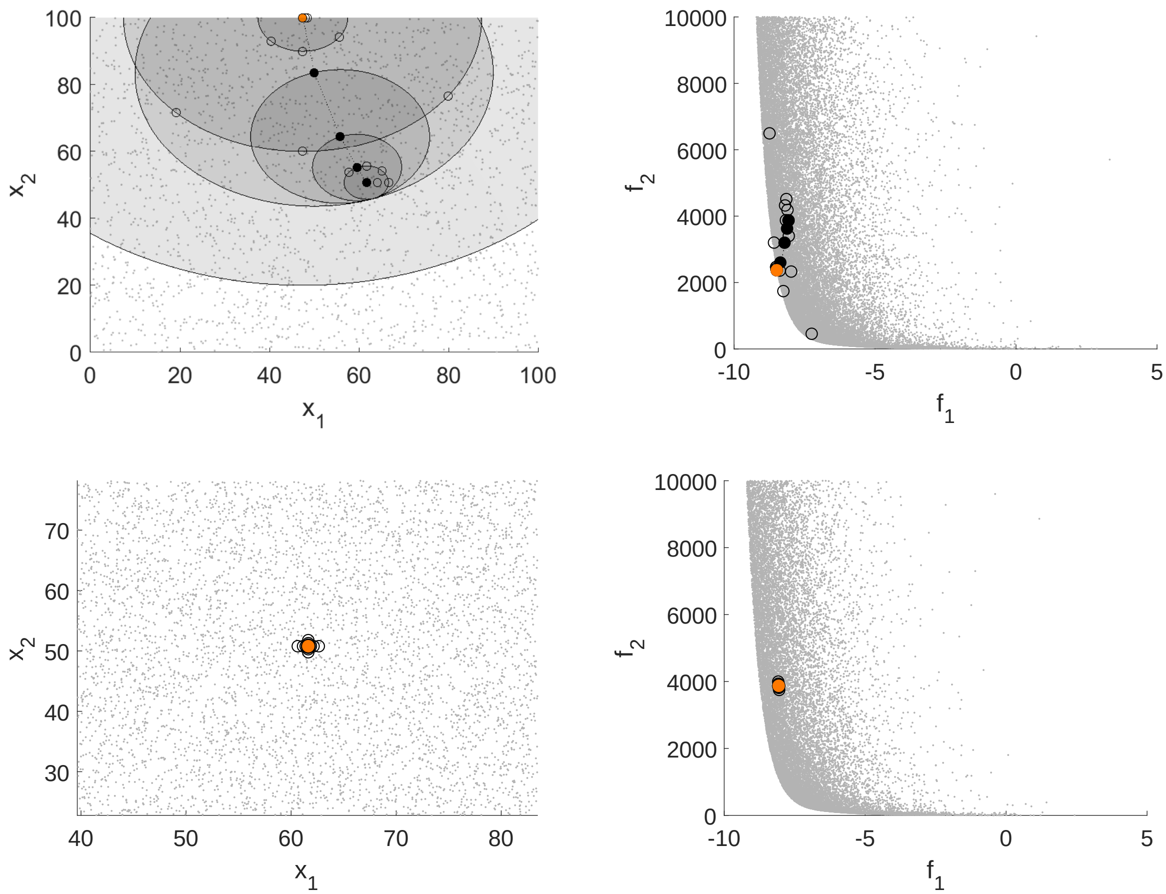


Figure 3: Test run for (T6) for MHT (top) and DMS (bottom)

For this test instance the iterates of MHT move towards an efficient point with few interpolation points that are mostly close to the iteration points. As the top left figure shows, the model was only updated in some iterations and already evaluated points could be reused. Within 19 function evaluations of f_1 an efficient point is generated. In contrast, DMS terminates after 41 function evaluations with a point close to the starting point which is not close to an efficient point. The results of runs with randomly generated starting points are depicted in Figure 4. MHT generates well distributed nondominated points within 8-114 function evaluations. DMS terminates for all starting points with points close to them. This is illustrated in the right figure as all the unfilled starting points are overlapped by the filled points computed by DMS. All of these points are computed within 41 function evaluations and most of them are not close to the Pareto front.

Another convex problem is (T7), but with a three-dimensional domain, given by

$$\min_{x\in\Omega} \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \min_{x\in[0,30]^3} \begin{pmatrix} \sum_{i=1}^n x_i^4 + \sum_{i=1}^n x_i^3 \\ \sum_{i=1}^n x_i \end{pmatrix} \tag{T7}$$

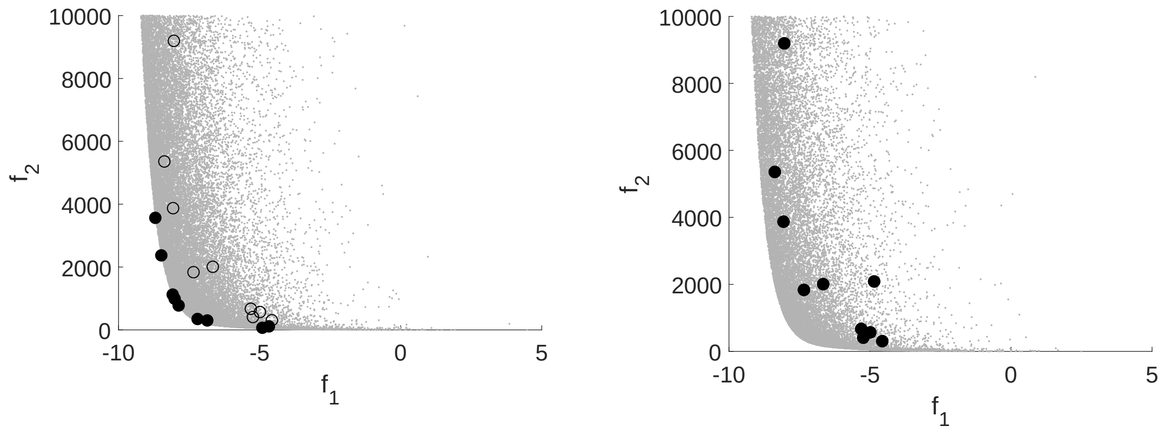


Figure 4: Multistart approach for (T6) for MHT (left) and DMS (right)

and with f_1 declared as expensive function. The unique efficient point for this optimization problem is $\bar{x} = (0, 0, 0)^\top$ with the function values $f_1(\bar{x}) = f_2(\bar{x}) = 0$. For all considered starting points all three algorithms compute this unique nondominated point respectively a point with vanishing distance to it. This is shown in Figure 5 for one instance. The

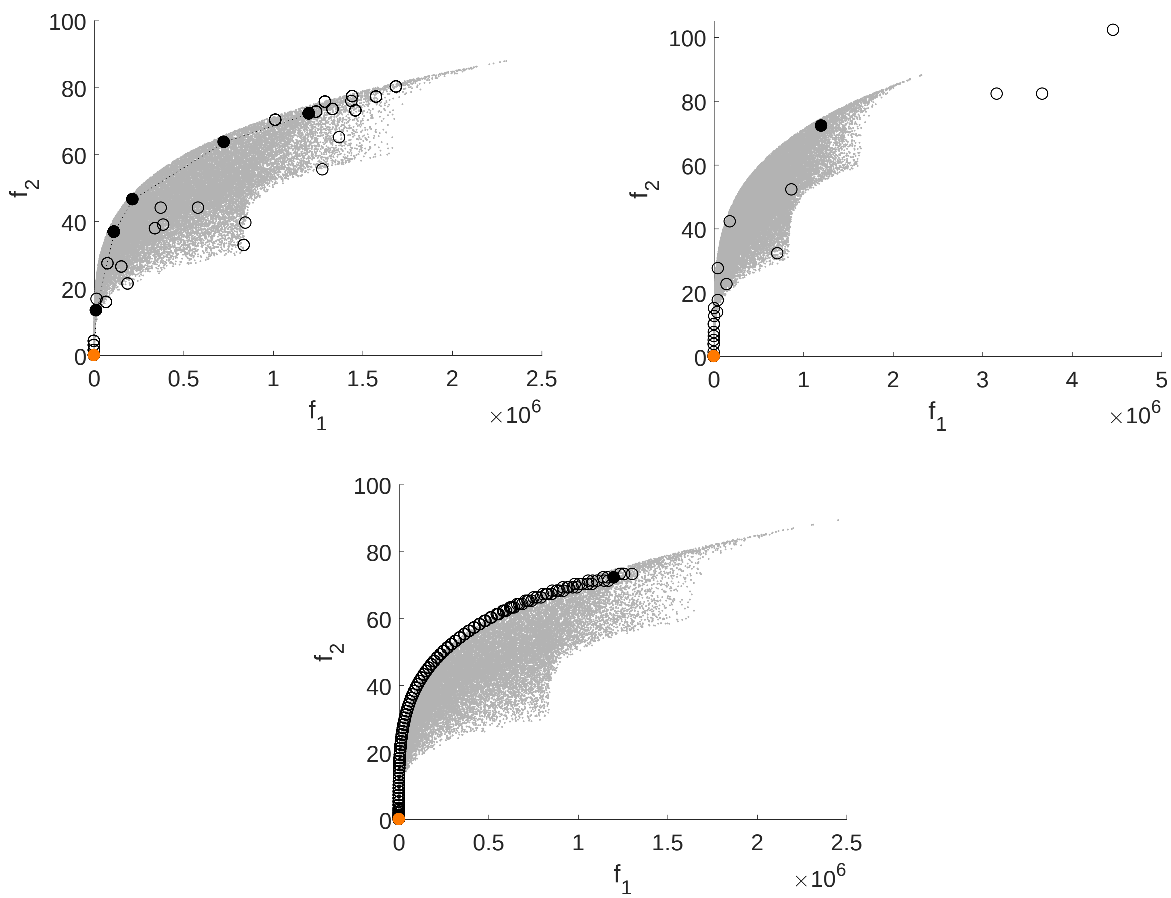


Figure 5: Test run for (T7) for MHT (top left), EFOS (top right) and DMS (bottom)

range of function evaluations for all instances is similar for MHT (29-57) and EFOS (9-75). Compared to DMS with 206-348 evaluations, both MHT and EFOS save function evaluations. The reason for this large difference is the direct search approach of DMS which produces, as depicted in Figure 5 for one instance, dense evaluated points in the image space. Again, EFOS computes also infeasible points during the runs which is also the case for the instance depicted in Figure 5.

3.2 Nononvex Test Problems

MHT is a local method and as proved in [19] it produces Pareto critical points which fulfill a necessary condition for local optimality. For convex problems local and global optimality is identical. For nonconvex problems this is in general not the case. Still, the algorithm performs well for most of the nonconvex problems. Exemplarily, we want to consider the results of three nonconvex test problems. The first one is (Deb513) from [5, 4] defined by

$$\min_{x\in\Omega}\begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \min_{x\in[0,1]^2}\begin{pmatrix} x_1 \\ g(x)h(x) \end{pmatrix} \tag{Deb513}$$

with $g(x) = 1 + 10x_2$, $h(x) = 1 - (x_1/g(x))^2 - (x_1/g(x))\sin(8\pi x_1)$ and a disconnected Pareto front. For this test problem f_2 is declared as expensive function. All three algorithms are capable of computing a nondominated point for all instances of this test problem as Figure 6 shows. Furthermore, it shows that again MHT (top left) and DMS (bottom) generate several nondominated points whereas EFOS (top right) computes only one nondominated point.

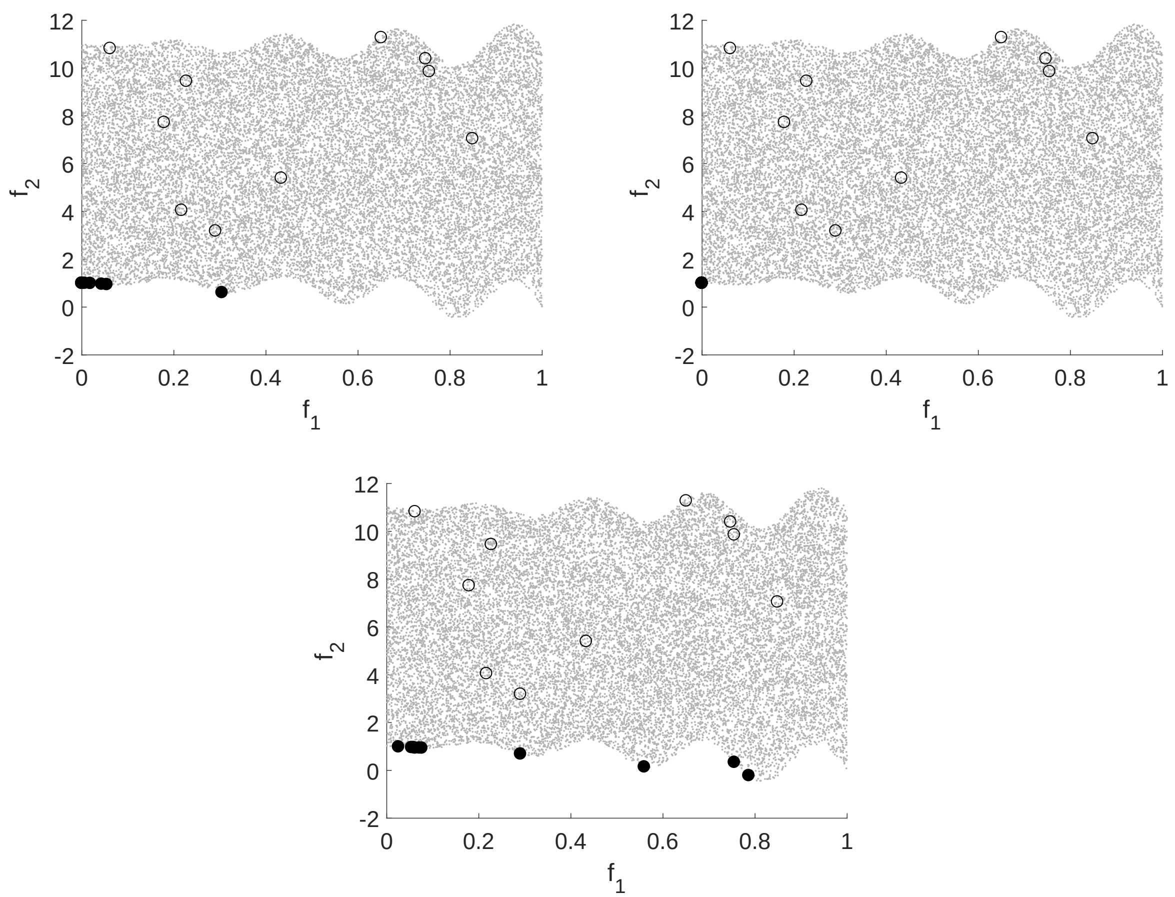


Figure 6: Multistart approach for (Deb513) for MHT (top left), EFOS (top right) and DMS (bottom)

The different search strategies are recognizable in Figure 7 which shows the result of one run with the same starting point for all three algorithms. EFOS (top right) computes the individual minimum of function f_1 within 7 function evaluations for all starting points. This is due to the weighted sum approach. The points that are evaluated during the run are mostly infeasible and situated far outside the pictured area. We zoomed in for reasons of illustration and comparison. For MHT (25 function evaluations) the top left figure illustrates the local search behavior controlled in the image space. Only for the initial model the interpolation points are spread broader over the image space, yet in the further iterations the interpolation points are close to the iteration points. The direct search approach of DMS (38 function evaluations) is visible in the bottom figure. It illustrates the search along the axes. DMS and MHT need a similar amount of function evaluations, yet both need significantly more than EFOS.

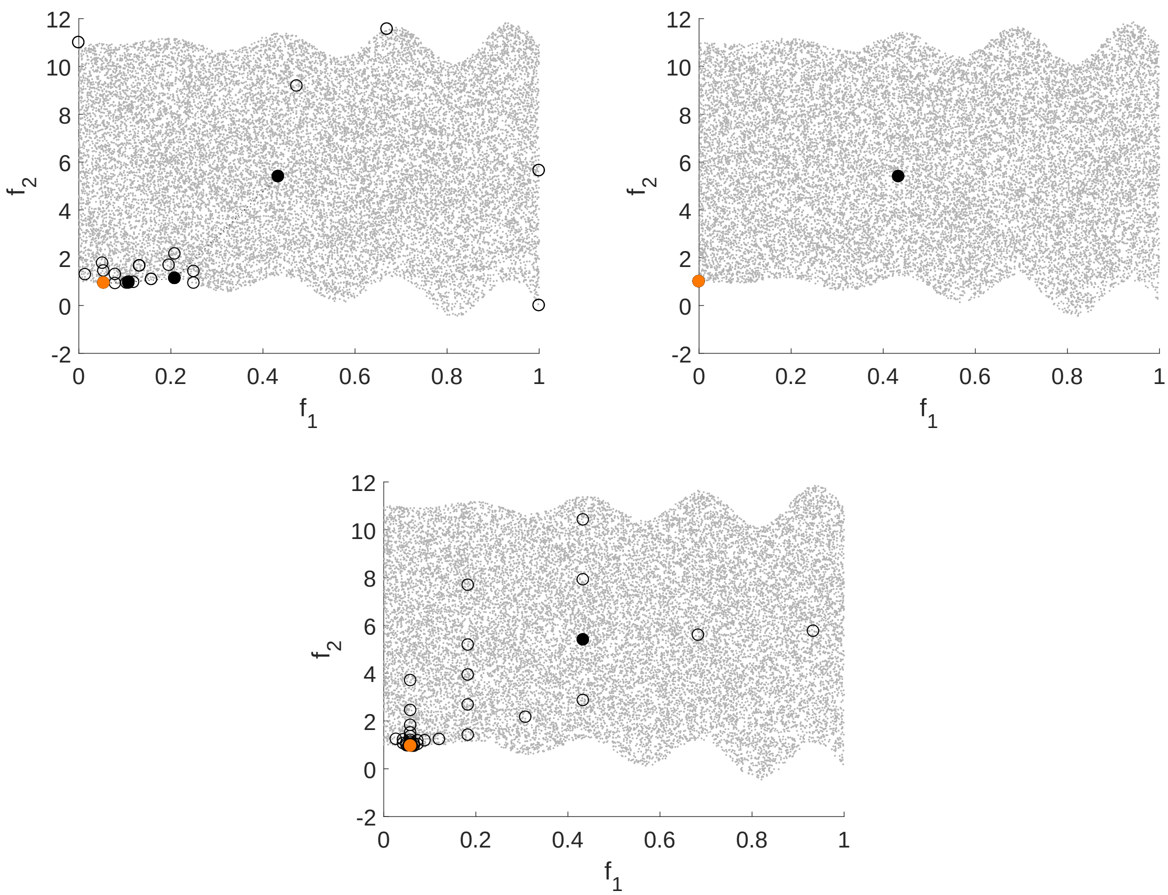


Figure 7: Test run for (Deb513) for MHT (top left), EFOS (top right) and DMS (bottom)

As second noncovex test problem we consider (Jin2) with $n = 4$ from [14, 4] defined by

$$\min_{x\in\Omega} \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \min_{x\in[0,1]^4} \begin{pmatrix} x_1 \\ g(x)\left(1 - \sqrt{\frac{x_1}{g(x)}}\right) \end{pmatrix} \tag{Jin2}$$

with $g(x) = 1 + 3\sum_{i=2}^{4} x_i$ and f_2 declared as expensive. EFOS could not compute an efficient solution for this test problem, but stopped with an internal error for the considered starting points. DMS and MHT compute nondominated points for all instances of this problem. The multistart approach with randomly chosen starting points in Figure 8 shows that both compute different nondominated points given different starting points. However, the points computed by DMS are better spread than the points computed by MHT. Regarding the required function evaluations no clear statement can be made which algorithm needs less (DMS 92-126, MHT 30-169). In all runs MHT needs many function

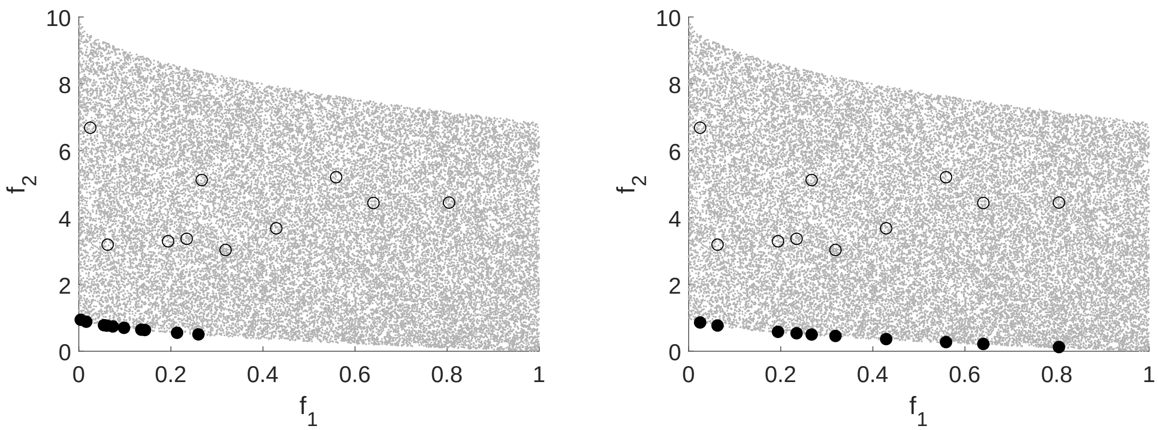


Figure 8: Multistart approach for (Jin2) for MHT (left) and DMS (right)

evaluations in the end of the procedure. Due to the nonconvexity and the local search strategy this number of function evaluations is needed to ensure the stopping criterion being fulfilled. This is exemplarily shown for one specific run in Figure 9. This instance again illustrates the coordinate search of DMS.

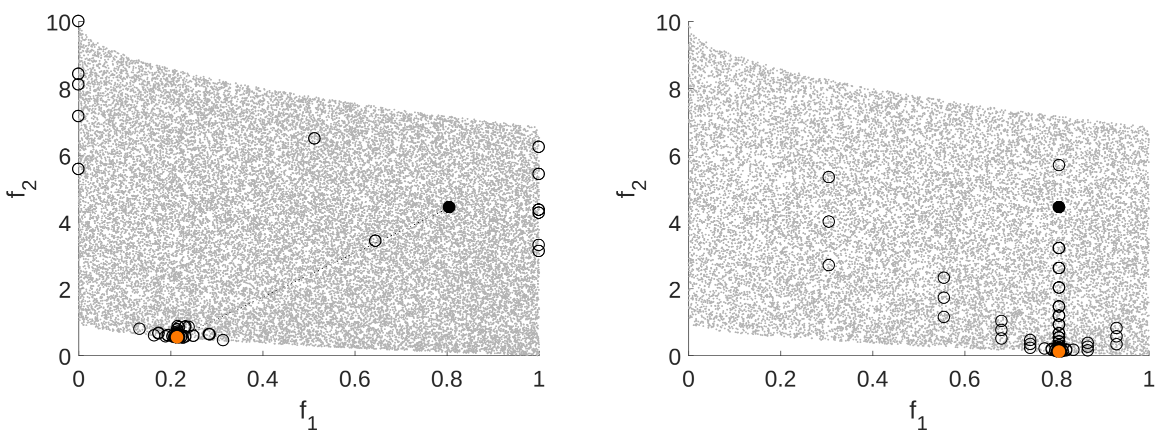


Figure 9: Test run for (Jin2) for MHT (left) and DMS (right)

As last nonconvex test problem we consider 'FF' with $n = 3$ from [10] given by

$$\min_{x \in \Omega} \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \min_{x \in [-4,4]^3} \begin{pmatrix} 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \\ 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \end{pmatrix} \tag{FF}$$

with f_1 declared as expensive function. This test problem illustrates that Pareto criticality is only a necessary condition for local efficiency and that MHT does not necessarily generate an efficient point if it stops legitimately according to the stopping criterion.
Figure 10 shows the result of MHT for one test instance of (FF). The algorithm terminates with the point $\bar{x} = (-0.0604, -1.2138, -0.8433)^\top$ with the function values $f_1(\bar{x}) = 0.9964$ and $f_2(\bar{x}) = 0.5234$. The stopping criterion indicates that $\bar{x}$ is Pareto critical, since the step size $\bar{t}$ of the auxiliary Pascelotti-Serafini problem that is used in MHT is small enough ($\bar{t} = 3.319810^{-7}$). For further details see [19, sections 3.2, 5.1]. Additionally, another criterion using an auxiliary function denoted by ω and described later in subsection 3.4 confirms that $\bar{x}$ is a Pareto critical point ($\omega(\bar{x}) = 8.260410^{-3}$). Though, this point is not an efficient point as the illustration in the image space in Figure 10 shows.

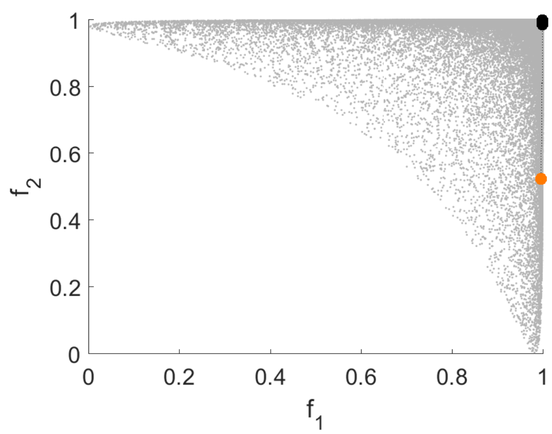


Figure 10: Test run for (FF) for MHT

3.3 Scalable Test Problems

We considered test problems with different dimensions for the domain and did also runs for MHT with two scalable test problems up to dimension 50. As expected the numerical effort rises when the dimension rises. The higher the dimension is, the more function evaluations are needed to build up a model. If in every iteration a complete new set of interpolation points would be computed, a total number of $(n + 1)(n + 2)/2$ function evaluations would be needed for the expensive function in every iteration. However, in MHT the model is not updated in every iteration, but only if necessary and if it is updated, the former interpolation points are reused if possible. Thus, the number of new function evaluations necessary is kept to a minimum.
Furthermore, we suggest in [19] to use for higher dimensions a linear interpolation model also based on Lagrange polynomials. This needs only $n + 1$ interpolation points and therefore the number of function evaluations also reduces. Though, a linear model needs to be updated more often since it is less accurate. Table 2 gives an overview of how many function evaluations are needed to compute one model function (quadratic/linear interpolation with Lagrange polynomials) depending on the dimension of the domain.

n	2	3	4	5	10	20	30	40	50
quadratic model	6	10	15	21	66	231	496	861	1326
linear model	3	4	5	6	11	21	31	41	51

Table 2: Function evaluations for computing one model function

In the numerical tests we used the linear models for all instances with dimension $n \geq 10$. To illustrate the behavior of MHT with rising dimension we consider the scalable test problem (Jin1) from [14, 4] defined by

$$\min_{x\in\Omega} \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \min_{x\in[0,1]^n} \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} x_i^2 \\ \frac{1}{n}\sum_{i=1}^{n}(x_i-2)^2 \end{pmatrix} \tag{Jin1}$$

with f_1 declared as expensive. It has been tested with MHT for $n \in \{2, 3, 4, 5, 10, 20, 30, 40, 50\}$. Figure 11 shows runs with MHT for $n = 5$ on the left and $n = 10$ on the right. For $n = 10$

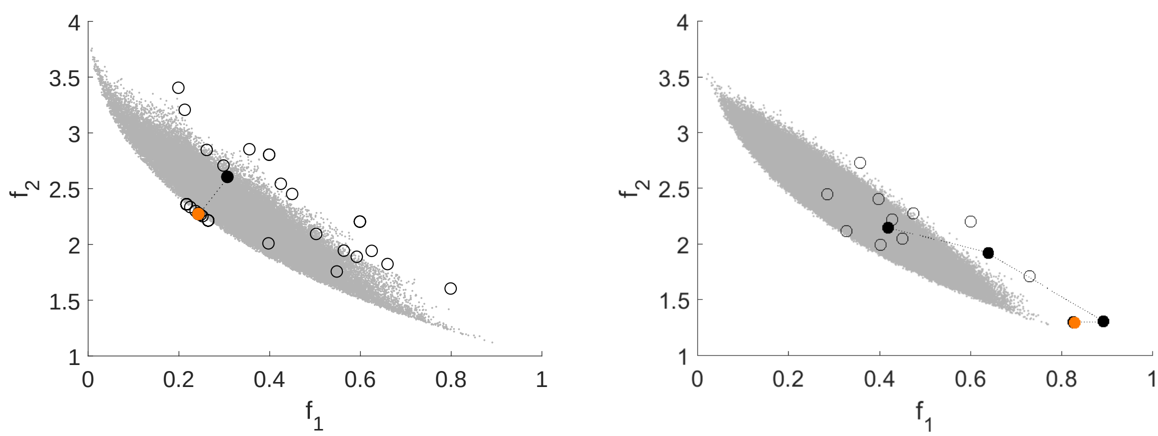
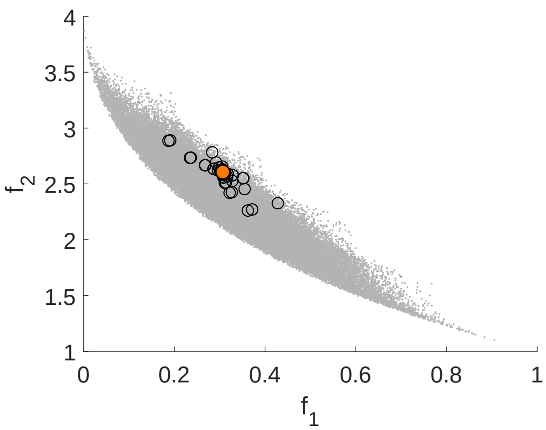


Figure 11: Test run for (Jin1) for MHT with $n = 5$ (left) and $n = 10$ (right)

a linear model is used which as expected worsens the predictions of the model functions. Since the trial point acceptance test in MHT does not demand a strict decrease in every component, but is instead a weaker formulation, also points that increase one of the objective functions can be accepted. Figure 12 shows the results for DMS applied to (Jin1) with $n = 5$ and $n = 10$ with the same starting points as used in the runs depicted in Figure 11. MHT needs 42 function evaluations for $n = 5$ (21 for $n = 10$) and therefore significantly less than DMS which needs 78 evaluations (152 for $n = 10$). Even though DMS computes many function values, it explores only the area close to the starting point and terminates with a point close to the starting point.



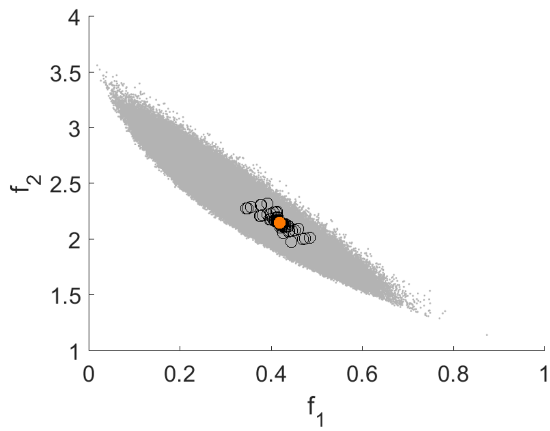


Figure 12: Test run for (Jin1) for DMS with $n = 5$ (left) and $n = 10$ (right)

In the following we give a short overview of the number of function evaluations (ranges, R, and mean values, M) required by MHT and DMS for the scalable test examples (Jin1) and T4, see Test Problem 4 in Appendix A. Table 3 gives an overview of the range (R) and mean value (M) of used function evaluations for MHT.

n	2	3	4	5	10	20	30	40	50
Jin1 (R)	11-12	20-32	15-46	42-82	15-667	46-621	39-322	63-244	109-316
Jin1 (M)	11.4	24.3	29.9	52.4	107.3	174.4	120.2	121.3	165
T4 (R)	12-14	21-22	31-32	43-44	68-206	252-433	399-561	609-1013	1023-1459
T4 (M)	13.2	21.4	31.4	43.7	152.1	338	483.7	794.4	1246.9

Table 3: Function evaluations (range and mean value) per dimension for MHT

As already seen in the instance of (Jin1) presented above, the ranges in Table 3 show that there are instances for which a higher dimension required less function evaluations than a lower dimension, e.g. dimension 5 and 10. This is due to the choice of starting points and due to the different kinds of model functions. From dimension 10 onwards linear model functions are used for the as expensive declared objective function. Along with the choice of a starting point and the local search strategy this can cause a lower total number of function evaluations for single instances. However, in general, the tendency of rising function evaluations with rising dimension is apparent.
The comparison method DMS is a direct search approach from [4]. For the general direct search approach $2n$ function evaluations are needed in every iteration. Table 4 gives an overview of this number up to dimension 50.

n	2	3	4	5	10	20	30	40	50
eval. per it.	4	6	8	10	20	40	60	80	100

Table 4: Function evaluations per iteration for general direct search

For DMS as it is implemented and available, Table 5 shows the the range (R) and mean value (M) of function evaluations needed for the two scalable test problems. We set the maximum number of allowed function evaluations to 2000. This is not enough for some instances of T4 as Table 5 shows. For these instances DMS terminated with the maximum number of function evaluations reached without having computed an efficient point. We believe that with further function evaluations, DMS will solve these test instances.

n	2	3	4	5	10	20	30	40	50
Jin1 (R)	30-35	47-52	63-69	78-85	152-165	307-328	469-488	622-648	767-818
Jin1 (M)	33.1	49.8	66.2	81.7	160.7	321.4	479.7	637.2	800.5
T4 (R)	54-152	71-213	117-424	181-510	706-1128	1849-2000	2000	2000	2000
T4 (M)	83.8	133.2	212.1	303.5	834.1	1998.6	2000	2000	2000

Table 5: Function evaluations (range and mean value) per dimension for DMS

Table 3 and Table 5 give a first impression of how the dimension n influences MHT in comparison to DMS. As expected, the direct search approach needs significantly more function evaluations with rising dimension than MHT. The mean values of function evaluations per dimension for all instances of the scalable test problems FF, Jin1, Jin2, Jin3, Jin4, T4 (see Table 1 in section 2) are listed in Table 6.

n	2	3	4	5	10	20	30	40	50
MHT	16.09	39.97	72.63	109.22	129.7	256.2	301.95	457.85	705.95
DMS	46.40	79	122.38	174.52	834.10	1153.2	1239.8	1318.6	1400.3

Table 6: Mean value of function evaluations per dimension for all scalable problems for MHT and DMS

This indicates that, compared to the direct search approach of DMS, the algorithm introduced in [19] can also save function evaluations in higher dimensions. However, it is important to note that DMS does not make use of any derivative information, also not of the cheap function. Therefore, it must be expected that DMS needs more function evaluations than any method that does use such information.

3.4 Performance Profiles

For classifiying the test runs as successful or not successful the distance to the Pareto front, the set of nondominated points, is used. If it falls below a problem dependent constant, the test run for an instance is classified as solved. To compare the performance of the algorithms the number of function evaluations for the as expensive declared function is counted until the algorithm terminates. Figure 13 shows a performance profile for all 348 convex test instances in full range on the left and zoomed in on the right.
If up to 480 function evaluations are allowed for the expensive function, DMS and EFOS behave similar. With further function evaluations DMS is capable of solving further test instances whereas EFOS stagnates and cannot solve more instances. In general, Figure 13 shows that MHT needs less function evaluations than EFOS and DMS to solve the convex test problems. It solves all 348 convex test instances within at most 1459 expensive

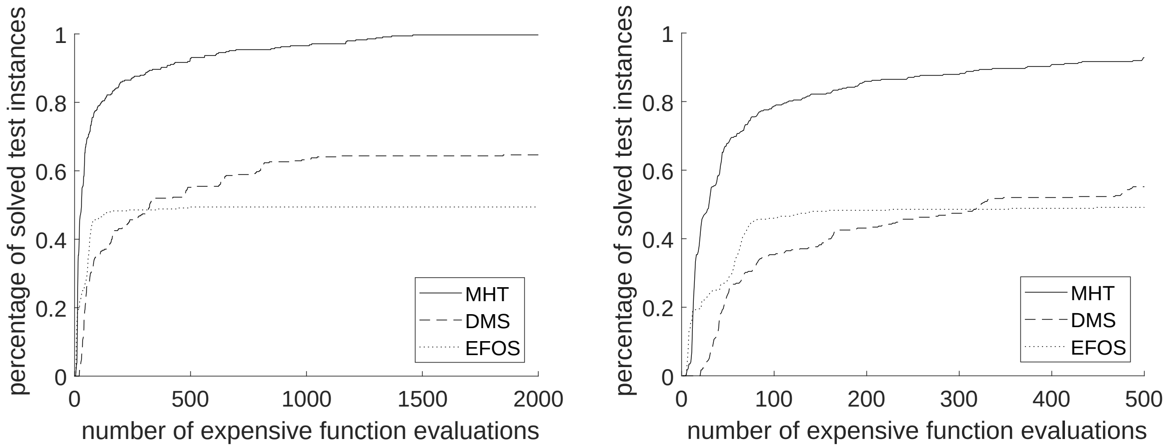


Figure 13: Performance Profile for MHT, DMS and EFOS for 348 convex instances

function evaluations. This high number is due to the high dimensional test instances included. If considering only test instances up to dimension 10 all convex instances are solved by MHT after 667 expensive function evaluations as the performance profile in Figure 14 shows (full range left, zoomed in right). With the same amount of function evaluations (667) DMS solves 64.93% and EFOS 62.31 % of the convex test problems up to dimension 10.

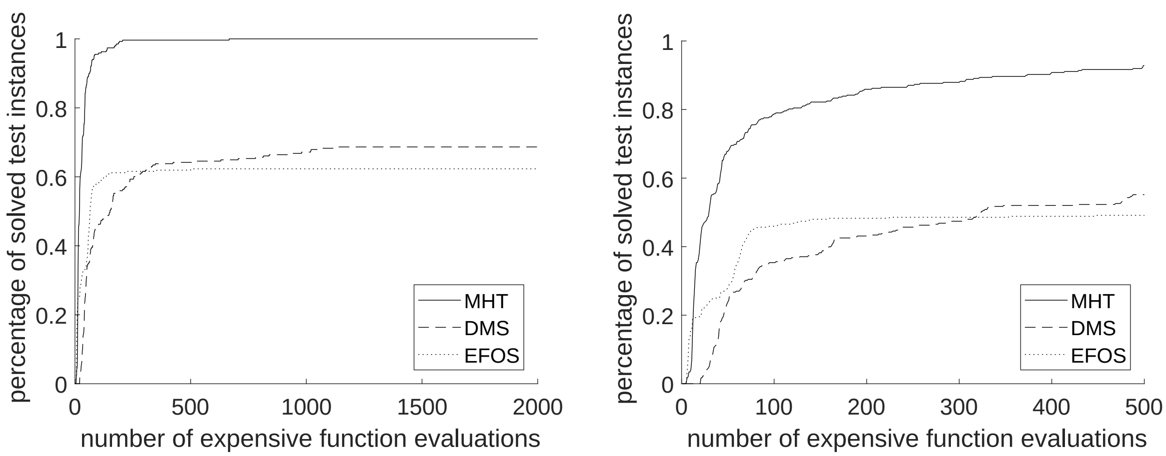


Figure 14: Performance Profile for MHT, DMS and EFOS for 268 convex instances, $n \leq 10$

As already noted, MHT is a local method and generates Pareto critical points, see [19]. Pareto criticality is a necessary condition for local optimality. This must be regarded when considering nonconvex problems since local optimality is in general not synonymous to global optimality. If the algorithm stops legitimately in a Pareto critical point, the distance to the Pareto front does not need to converge to zero. Thus, for a performance profile over all considered test examples we do not only use this distance to calssify test instances as solved, but complement it with a measure for Pareto criticality. The auxiliary function $\omega : \mathbb{R}^n \to \mathbb{R}$ defined by

$$\omega(x) := - \min_{\|d\| \leq 1} \max_{i=1,...,q} \nabla_x f_i(x)^\top d$$

for $f_i : \mathbb{R}^n \to \mathbb{R}$ continuously differentiable functions, $i = 1, ..., q$, characterizes Pareto criticality. According to [7, 8, 9] ω is a continuous function, it holds $\omega(x) \geq 0$ for all $x \in \mathbb{R}^n$ and a point $x \in \mathbb{R}^n$ is Pareto critical for (MOP) if and only if it holds $\omega(x) = 0$. Consequently, given $\bar{x}$ the solution generated by one of the considered algorithms (MHT, DMS, EFOS), we classify a test instance as solved if either the distance of $f(\bar{x})$ to the Pareto front is small enough or if it holds $\omega(\bar{x}) \leq \varepsilon$. We chose $\varepsilon = 0.1$ for the data analysis. Using these classifications, the performance profile in Figure 15 shows how many of all 802 considered test instances are solved depending on the required function evaluations for MHT, DMS and EFOS. The full range is shown on the left and on the right the performance profile is zoomed in to 500 function evaluations.

Figure 15 illustrates that by applying MHT 98.13% of all test instances are solved with either an efficent or a Pareto critical point. Within the same number of function evaluations (1459) EFOS solved 57.98% and DMS sovled 84.04% of all considered test instances. Thus, MHT solved more test problems than both comparison methods and needs less function evaluations. Although the behavior of DMS is similar to our method, still MHT saves computation time and solves more instances in terms of distance to the Pareto front or Pareto criticality.

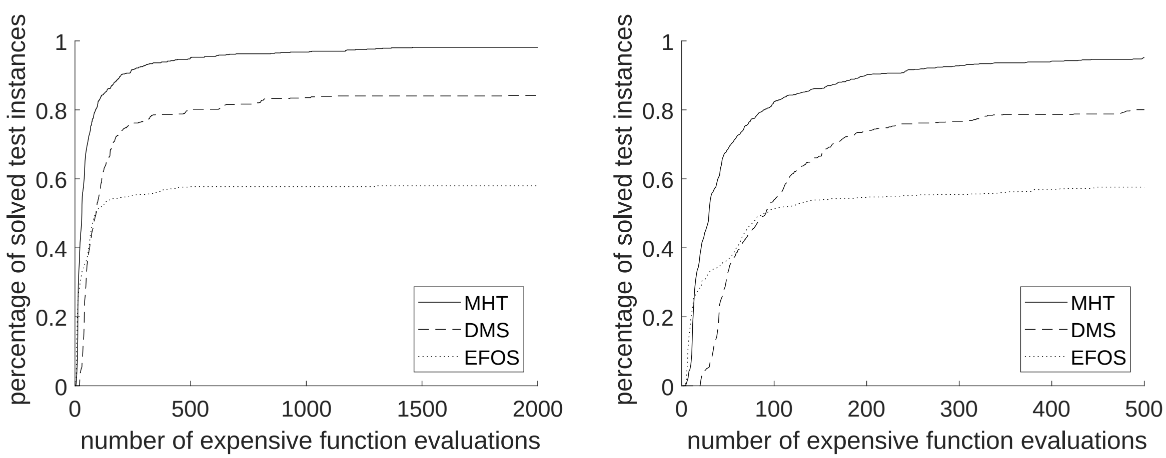


Figure 15: Performance Profile for MHT, DMS and EFOS for all 802 instances

Appendix

A Self-chosen Test Problems

In the following we formulate the objective functions and the constraint sets of the self-chosen test examples T1-T8 listed in Table 1.

Test Problem 1 (T1) (MOP) with $n = 2$, $q = 2$, constraint set $\Omega = \mathbb{R}^2$,

$$f_1(x) = \frac{1}{2}x_1^2 + x_2^2 - 10x_1 - 100$$
$$f_2(x) = x_1^2 + \frac{1}{2}x_2^2 - 10x_2 - 100$$

and f_1 declared as expensive.

Test Problem 2 (T2) (MOP) with $n = 2$, $q = 2$, constraint set $\Omega = \mathbb{R}^2$,

$$f_1(x) = \sin x_2$$
$$f_2(x) = 1 - \exp\left(-\left(x_1 - \frac{1}{\sqrt{2}}\right)^2 - \left(x_2 - \frac{1}{\sqrt{2}}\right)^2\right)$$

and f_2 declared as expensive.

Test Problem 3 (T3) (MOP) with $n = 2$, $q = 2$, constraint set $\Omega = [-2, 2]^2$,

$$f_1(x) = x_1 + 2$$
$$f_2(x) = x_1 - 2 + x_2$$

and f_2 declared as expensive.

Test Problem 4 (T4) (MOP) with $n \in \{2, 3, 4, 5, 10, 20, 30, 40, 50\}$, $q = 2$, constraint set $\Omega = [-10, 10]^n$,

$$f_1(x) = \sum_{i=1}^{n-1} x_i^2 + 2$$
$$f_2(x) = \sum_{i=1}^{n} x_i - 2$$

and f_1 declared as expensive.

Test Problem 5 (T5) (MOP) with $n = 2$, $q = 2$, constraint set $\Omega = (0, 30] \times [0, 30]$,

$$f_1(x) = x_1 \ln(x_1) + x_2^2$$
$$f_2(x) = x_1^2 + x_2^4$$

and f_1 declared as expensive.

Test Problem 6 (T6) (MOP) with $n = 2, q = 2$, constraint set $\Omega = (0, 100]^2$,

$$f_1(x) = -ln(x_1) - \ln(x_2)$$
$$f_2(x) = x_1^2 + x_2$$

and f_1 declared as expensive.

Test Problem 7 (T7) (MOP) with $n = 3, q = 2$, constraint set $\Omega = [0, 30]^3$,

$$f_1(x) = \sum_{i=1}^{n} x_i^4 + \sum_{i=1}^{n} x_i^3$$
$$f_2(x) = \sum_{i=1}^{n} x_i$$

and f_1 declared as expensive.

Test Problem 8 (T8) (MOP) with $n = 3, q = 3$, constraint set $\Omega = (0, 10] \times [0, 10] \times [0, 10]$

$$f_1(x) = \sum_{i=1}^{n} x_i^3$$
$$f_2(x) = \sum_{i=1}^{n-1} (x_i - 4)^2 + x_n^2$$
$$f_3(x) = -\ln(x_1) + 5 \sum_{i=2}^{n} x_i^2$$

and f_3 declared as expensive.

B Computation of Interpolation Points

In the algorithm MHT [19] an interpolation model based on Lagrange polynomials is used for the expensive function. As for every interpolation model, the quality of the model depends on the quality of the interpolation points. We follow the theory presented in [3] and use the concept of poisedness. For explaining this concept we need to give a general overview of the interpolation model used.

Consider the space of polynomials of degree less than or equal to 2 in $\mathbb{R}^n$ denoted by $\mathcal{P}_n^2$. It is known that the dimension p of this space is given by $p = (n + 1)(n + 2)/2$, see [3]. Furthermore, let $\psi = \{\psi_1, \psi_2, ..., \psi_p\}$ be an arbitrary basis of $\mathcal{P}_n^2$. Then every polynomial $m(x) \in \mathcal{P}_n^2$ can be expressed as

$$m(x) = \sum_{j=1}^{p} \alpha_j \psi_j(x) \tag{1}$$

with $\alpha \in \mathbb{R}^p$ some suitable coefficients. By determining these coefficients the model function is determined. This can be done by using the interpolation conditions

$$m(y^i) = f(y^i) \text{ for all } i = 1, 2, ..., p$$

which form a linear system in terms of the interpolation coefficients. This can be written in matrix form as

$$M(\psi, Y)\alpha_\psi = f(Y)$$

with $(M(\psi, Y))_{ij} = \psi_j(y^i)$ for $i, j = 1, ..., p$, $f(Y) = (f(y^1), f(y^2), ..., f(y^p))^\top$ and $\alpha_\psi = (\alpha_1, \alpha_2, ..., \alpha_p)^\top$. This system of equations has a unique solution if the matrix $M(\psi, Y)$ is nonsingular. This matrix is used to define poisedness.

Definition B.1 *The set $Y = \{y^1, y^2, ..., y^p\}$ is poised for polynomial interpolation in $\mathbb{R}^n$ if the corresponding matrix $M(\psi, Y)$ is nonsingular for some basis ψ in $\mathcal{P}_n^2$.*

In the literature, poised sets are also referred to as d-unisolvent sets. It is easy to see that if $M(\psi, Y)$ is nonsingular for some basis ψ, then it is nonsingular for any basis of $\mathcal{P}_n^d$. Under these circumstances, the interpolating polynomial m exists and is unique.

Lemma B.2 *[3, Lem. 3.2] Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and a poised set $Y \subset \mathbb{R}^n$, the interpolating polynomial m defined by (1) exists and is unique.*

To approximate the expensive function, we use the basis given by the Lagrange polynomials $L = \{l_1, l_2, ..., l_p\}$, because it is commonly used for measuring poisedness. Given a set of interpolation points $Y = \{y^1, y^2, ..., y^p\} \subset \mathbb{R}^n$, they are defined by

$$l_i(y^j) = \begin{cases} 1 & \text{, if } i = j \\ 0 & \text{, else} \end{cases} .$$

The Lagrange polynomials can be computed by Algorithm 1 from [3, Alg. 6.1] using the monomial basis

$$\bar{\psi} := \{\bar{\psi}_1, ..., \bar{\psi}_p\} = \left\{1, x_1, x_2, ..., x_n, \frac{x_1^2}{2}, x_1x_2, ..., x_{n-1}x_n, \frac{x_n^2}{2}\right\}.$$

as initial basis.

Algorithm 1 Computing Lagrange polynomials [3, Alg. 6.1]

Initialization: polynomials $\bar{\psi}_i$ of monomial basis as initial polynomials $l_i, i = 1, 2, ..., p$, set of interpolation points Y, $|Y| = p$
for $i = 1, 2, ..., p$ **do**
  Point selection: Compute $j_i = \text{argmax}_{i \le j \le p} |l_i(y^j)|$.
    If $|l_i(y^{j_i})| = 0$, then **stop** (Y is not poised)
    Otherwise exchange points y^i and y^{j_i} in set Y.
  Normalization: $l_i(x) = l_i(x)/l_i(y^i)$
  Orthogonalization: $l_j(x) = l_j(x) - l_j(y^i)l_i(x)$ for $j = 1, 2, ..., p, j \neq i$
end for

Given a set of interpolation points $Y = \{y^1, y^2, ..., y^p\} \subset \mathbb{R}^n$, the model function using Lagrange polynomials can be formulated as $m(x) = \sum_{i=1}^{p} f(y^i)l_i(x)$ and is referred to as Lagrange model. If the set of interpolation points is poised, it can be proved, see for example [3], that the Lagrange polynomials exist, are uniquely defined and actually form a basis of $\mathcal{P}_n^2$. Furthermore, Lemma B.2 holds for a general basis and therefore also for the basis of the Lagrange polynomials. Thus, the Lagrange model exists and is unique. This implies the following statement about exactness of the Lagrange model for a quadratic function.

Lemma B.3 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function and $Y = \{y^1, y^2, ..., y^p\} \subset \mathbb{R}^n$ a poised set of interpolation points. Then the Lagrange model $m_L : \mathbb{R}^n \to \mathbb{R}$ given by $m_L(x) = \sum_{i=1}^{p} f(y^i) l_i(x)$ is exact, that is it holds $f = m_L$.*

Proof. Since Y is a poised set, $f \in \mathcal{P}_n^2$ and the Lagrange polynomials are a basis of $\mathcal{P}_n^2$, there exist scalars $\alpha_i \in \mathbb{R}$ for $i \in \{1, ..., n\}$ such that it holds $f(x) = \sum_{i=1}^{p} \alpha_i l_i(x)$ for all $x \in \mathbb{R}^n$. Furthermore, it holds $f(y^j) = \sum_{i=1}^{p} \alpha_i l_i(y^j) = \alpha_j$ and $f(y^j) = m_L(y^j) = \sum_{i=1}^{p} f(y^i) l_i(y^j) = f(y^j)$ for all interpolation points $y^j \in Y, j = 1, ..., p$. This implies $\alpha_j = f(y^j)$ for all $j \in \{1, ..., p\}$ and therefore $f = m_L$. □

The basis of Lagrange polynomials can be computed and updated reasonably efficiently which makes them suitable for maintaining interpolation models. In Algorithm 2 a realization of computing a poised set using Lagrange polynomials is given according to [3, Alg. 6.2]. As input a set of interpolation points Y is needed. This set can contain only one point or several points. The algorithm completes this set to a poised set and computes new points if necessary. These points are contained in a closed ball denoted by B.

Algorithm 2 Completing a nonpoised set by Lagrange polynomials [3, Alg. 6.2]

Initialization: polynomials $\bar{\psi}_i$ of monomial basis as initial polynomials $l_i, i = 1, 2, ..., p$, initial set of interpolation points Y, $|Y| = p_{ini}$, closed ball B

for $i = 1, 2, ..., p$ **do**

  Point selection: Compute $j_i = \text{argmax}_{i \le j \le p_{ini}} |l_i(y^j)|$.

    If $|l_i(y^{j_i})| > 0$ and $i \le p_{ini}$, exchange points y^i and y^{j_i} in set Y.

    Otherwise compute (or recompute if $i \le p_{ini}$) y^i as

$$y^i \in \underset{x \in B}{\text{argmax}} \, |l_i(x)|$$

  Normalization: $l_i(x) = l_i(x)/l_i(y^i)$

  Orthogonalization: $l_j(x) = l_j(x) - l_j(y^i) l_i(x)$ for $j = 1, 2, ..., p, j \ne i$

end for

For the usage of Algorithm 2 within MHT B is chosen as the current trust region B_k and Y as the current iteration point x_k and all former evaluated points that are contained in B_k.

References

[1] F. Cheng and X. Li. Generalized center method for multiobjective engineering optimization. *Eng. Opt.*, 31:641–661, 1999.

[2] C. Coello, D. V. Veldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/Plenum Publishers, 2002.

[3] A. Conn, K. Scheinberg, and L. Vicente. *Introduction to Derivative-Free Optimization.* MPS-SIAM Series on Optimization, 2009.

[4] A. Custodio, J. Madeira, A. Vaz, and L. Vicente. Direct multisearch for multiobjective optimization. *SIAM J. Optim.*, 21(3):1109–1140, 2011.

[5] K. Deb. Multiobjective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[6] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, volume 1, pages 825–830, 2002.

[7] L. Drummond and B. Svaiter. A steepest descent method for vector optimization. In *J. Comput. Appl. Math.*, volume 175, pages 395–414, 2005.

[8] J. Fliege, L. Drummond, and B. Svaiter. Newton‘s method for multiobjective optimization. *SIAM J. Optim.*, 20(2):602–626, 2009.

[9] J. Fliege and B. Svaiter. Steepest descent methods for multicriteria optimization. *Math. Methods of Oper. Res.*, 51(3):479–494, 2000.

[10] C. Fonseca and P. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[11] C. Fonseca and P. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28(1):26–37, 1998.

[12] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evolutionary Computation*, 10(5):477–506, 2006.

[13] C.-L. Hwang and A. Masud. *Multiple Objective Decision Making Methods and Applications: A State-of-the-Art Survey*, volume 164 of *Lecture Notes in Economics and Mathematical Systems*. Springer Berlin Heidelberg, 1979.

[14] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 1042–1049. Morgan Kaufmann Publishers Inc., 2001.

[15] F. Kursawe. A variant of evolution strategies for vector optimization. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, PPSN I, pages 193–197, Berlin, Heidelberg, 1991. Springer.

[16] A. Lovison. Singular continuation: Generating piecewise linear approximations to pareto sets via global analysis. *SIAM J: Optim.*, 21(2):463–490, 2011.

[17] J. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100. L. Erlbaum Associates Inc., 1985.

[18] A. Thekale. *Trust-Region Methods for Simulation Based Nonlinear Optimization*. PhD thesis, Friedrich-Alexander-Universität Erlangen Nürnberg, 2011.

[19] J. Thomann and G. Eichfelder. A trust region algorithm for heterogeneous multi-objective optimization. *Preprint-Series of the Institute for Mathematics, Technische Universität Ilmenau, Germany*, 2018.

[20] D. V. Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations*. PhD thesis, Air Force Institute of Technology, Wright Patterson AFB, OH, USA, 1999.

[21] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: empirical results. *Evol. Comput.*, 8(2):173–195, 2000.