Outi Wilppu | Napsu Karmitsa | Marko M. Mäkelä

New Multiple Subgradient Descent Bundle Method for Nonsmooth Multiobjective Optimization

Turku Centre *for* Computer Science

TUCS Technical Report

No 1126 version 2, last updated June 2017, original version December 2014

New Multiple Subgradient Descent Bundle Method for Nonsmooth Multiobjective Optimization

Outi Wilppu
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
outi.wilppu@utu.fi
Napsu Karmitsa
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
napsu@karmitsa.fi
Marko M. Mäkelä
University of Turku, Department of Mathematics and Statistics
FI-20014 Turku, Finland
makela@utu.fi

TUCS Technical Report
No 1126 version 2, last updated June 2017, original version December 2014

Abstract

The aim of this paper is to propose a new multiple subgradient descent bundle method for solving unconstrained convex nonsmooth multiobjective optimization problems. Contrary to many existing multiobjective optimization methods, our method treats the objective functions as they are without employing a scalarization in a classical sense. The main idea of this method is to find descent directions for every objective function separately by utilizing the proximal bundle approach, and then trying to form a common descent direction for every objective function. In addition, we prove that the method is convergent and it finds weakly Pareto optimal solutions. Finally, some numerical experiments are considered.

Keywords: Bundle methods, Descent methods, Multiobjective optimization, Nonsmooth optimization

TUCS Laboratory
Turku Optimization Group (TOpGroup)

1 Introduction

Multiobjective nature arises in many practical applications. There are several conflicting objectives to be optimized, and the aim is to find a compromise between these different goals being as good as possible for all the objectives at the same time. The compromise is optimal if we cannot improve any objective without deteriorating some other. This kind of problem is called multiobjective optimization problem and these problems exist in many areas, for example, in engineering [33], economics [36] and mechanics [34].

The most of the existing multiobjective optimization methods convert the multiple objectives to a single-objective problem and apply some single-objective method to solve it. This process is called scalarization (see e.g. [9, 30]). In this paper, instead of scalarization, we are focusing on descent methods employing the objectives as they are. The method is said to be descent if it moves to the direction where the values of all objectives improve at every iteration and a new iteration point produced gives better values for objective functions. For continuously differentiable (i.e. smooth) functions there are several methods of this type described in literature, for example, [8, 10, 11, 12, 13, 14, 35, 37].

In addition to multiobjective characteristic, many of the real-life problems have nonsmooth nature meaning that the functions are not necessarily differentiable in the classical sense. There are several reasons for the nonsmoothness of problems [2, 25]: first, an objective function itself can be nondifferentiable as are, for example, the piecewise linear tax models in economics [21]; second, some technical constraints may cause a nonsmooth dependence between the variables even though the objective functions are continuously differentiable as in, for instance, obstacle problems in optimal shape design [15]; third, some optimization methods for a constrained problem may lead to a nonsmooth problem as, for example, the exact penalty function method [2]; and fourth, the problem may be analytically smooth but numerically behave like a nonsmooth problem. Since nonsmooth problems may not be differentiable, gradient-based methods cannot be utilized.

Bundle methods [2, 16, 18, 19, 20, 22, 25, 29, 32, 40] are considered as one efficient way to solve general nonsmooth single-objective optimization problems. The idea in these methods is to approximate the subdifferential (i.e. a set of generalized gradients, the so-called subgradients [6]) of the objective function with a bundle including information from the neighborhood of the iteration point. The only assumptions needed are that one arbitrary subgradient and the value of the objective function can be evaluated at every point.

As noted before, there exist various methods for smooth multiobjective optimization. In addition, there exist many methods for nonsmooth single-objective optimization as well. However, only few methods are designed for nonsmooth multiobjective optimization. We are focusing on descent methods employing the objectives as they are. In literature, methods of this kind are described, for in-

stance, in [5, 4, 18, 28, 38, 43] from which the methods described in [18, 28, 43] utilize the bundle idea. These descent methods may be applied, for example, as a part of interactive methods [31, 35] to produce different weakly Pareto optimal solution candidates or to approximate the Pareto frontier efficiently.

In this paper, we propose a new multiple subgradient descent bundle method (MSGDB) for convex nonsmooth multiobjective optimization problems. MSGDB generalizes the ideas of smooth multiple-gradient descent algorithm (MGDA) [7, 8] which, in its turn, extends the well-known steepest descent method for the multiobjective problems. The nonsmoothness of the objectives is taken into account by using the proximal bundle idea. That is, in MSGDB, all the objective functions are first linearized separately and then the proximal bundle approach is used to find descent directions for each objective function. After that, a convex hull of the individual descent directions is formed and a minimum norm element is calculated to obtain a candidate for the common descent direction for all the objective functions. Only if we have not found a common descent direction after user specified number of so-called null steps, we approximate the union of the subdifferentials of the objectives instead of the individual subdifferentials to guarantee the convergence of the method to weakly Pareto optimal points.

In addition, we provide some numerical experiments, where the performance of MSGDB is compared with the multiobjective proximal bundle method (MPB) [28, 31]. Both of these methods form a common descent direction for every objective function, and they are based on the proximal bundle approach. MPB is a generalization of the proximal bundle method utilizing an improvement function taking all the objectives into account at the same time. The improvement function is then linearized in order to obtain the descent search direction [28, 31].

The paper is organized as follows. In Section 2, we have compiled some basic results from multiobjective and nonsmooth optimization. Section 3 is devoted to describe MSGDB. Some numerical experiments are given in Section 4. In conclusions, in Section 5, we give some final remarks. Additionally, a brief summary of the basic idea of MPB is recalled in Appendix A.

This is an updated version, where some major changes are made to correct an error of the first version.

2 Preliminaries

We will consider an *unconstrained multiobjective optimization problem* of the form

$$\begin{aligned} \min \quad & \{f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\} \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{R}^n, \end{aligned} \tag{1}$$

where the objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ are assumed to be convex but not necessarily differentiable. Since the objective functions are convex,

they are known to be *locally Lipschitz continuous* [2]. First, we compile some basic results from multiobjective and nonsmooth optimization. For more details we refer to [2, 6, 9, 27, 29, 30, 39] and references given there.

A solution $\boldsymbol{x}^* \in \mathbb{R}^n$ of the problem (1) is called *Pareto optimal* if there does not exist another point $\boldsymbol{x} \in \mathbb{R}^n$ such that $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*)$ for all $i = 1, \ldots, m$ and $f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}^*)$ for at least one index $j \in \{1, \ldots, m\}$. In other words, no objective can be improved without impairing some other objective at the same time. We say that a solution $\boldsymbol{x}^* \in \mathbb{R}^n$ of the problem (1) is *weakly Pareto optimal* if there does not exist another point $\boldsymbol{x} \in \mathbb{R}^n$ such that $f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*)$ for all $i = 1, \ldots, m$. This means that there does not exist any other point such that all objective functions f_i have better values. Usually, there exist several Pareto optimal solutions, and clearly, every Pareto optimal solution is also weakly Pareto optimal.

We briefly introduce the concept of an *improvement function* [19, 28, 43] giving a tool to handle several objectives simultaneously. In the unconstrained case, the improvement function $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined by

$$H(\boldsymbol{x}, \boldsymbol{y}) = \max_{i=1\ldots,m} \{f_i(\boldsymbol{x}) - f_i(\boldsymbol{y})\}. \tag{2}$$

The improvement function has the following property validating the use of it.

Lemma 2.1. [18] *If $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ satisfy $H(\boldsymbol{x}, \boldsymbol{y}) < H(\boldsymbol{y}, \boldsymbol{y})$, then $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y})$ for all $i = 1, \ldots, m$.*

Nonsmooth functions do not have a gradient at every point. Thus, instead of a classical gradient we utilize generalized gradients, and the set of these generalized gradients is called a subdifferential. The *subdifferential* of a convex function $f_i : \mathbb{R}^n \to \mathbb{R}$ at the point $\boldsymbol{x}$ is defined as a set

$$\partial f_i(\boldsymbol{x}) = \left\{\boldsymbol{\xi}_i \in \mathbb{R}^n \mid f_i(\boldsymbol{y}) \geq f_i(\boldsymbol{x}) + \boldsymbol{\xi}_i^T(\boldsymbol{y} - \boldsymbol{x}) \text{ for all } \boldsymbol{y} \in \mathbb{R}^n\right\}.$$

The subdifferential $\partial f_i(\boldsymbol{x})$ is a nonempty, convex and compact set [39]. An element $\boldsymbol{\xi}_i \in \partial f_i(\boldsymbol{x})$ is called a *subgradient* of the function f_i at the point $\boldsymbol{x}$. The subgradient of a differentiable convex function is unique and it is equal to its gradient, or in other words, $\partial f_i(\boldsymbol{x}) = \{\nabla f_i(\boldsymbol{x})\}$ [39]. Throughout the paper, we assume that at least one arbitrary subgradient can be evaluated at every point for every objective function.

An optimization method is called *descent* if it produces a better solution at every iteration. In order to find this better solution, the concept of a descent direction is required. The direction $\boldsymbol{d} \in \mathbb{R}^n$ is said to be a *descent direction* for a function $f_i : \mathbb{R}^n \to \mathbb{R}$ at the point $\boldsymbol{x}$ if there exists $\varepsilon > 0$ such that

$$f_i(\boldsymbol{x} + t\boldsymbol{d}) < f_i(\boldsymbol{x}) \quad \text{for all } t \in (0, \varepsilon].$$

The direction $\boldsymbol{d} \in \mathbb{R}^n$ is a descent direction for a function f_i at the point $\boldsymbol{x}$ if (see e.g. [2])

$$\boldsymbol{\xi}_i^T \boldsymbol{d} < 0 \text{ for all } \boldsymbol{\xi}_i \in \partial f_i(\boldsymbol{x}). \tag{3}$$

A well-known necessary and sufficient condition for global optimality in the convex single-objective case is that the objective function attains its global minimum at the point $\boldsymbol{x}$ if and only if [39]

$$\mathbf{0} \in \partial f_i(\boldsymbol{x}). \tag{4}$$

The multiobjective counterpart for this result in Theorem 2.2 gives us necessary and sufficient conditions for a solution $\boldsymbol{x}^*$ to be weakly Pareto optimal for the problem (1). To simplify the notation, we denote $F(\boldsymbol{x}) = \bigcup_{i=1}^m \partial f_i(\boldsymbol{x})$ throughout the paper.

Theorem 2.2. [27] *Let f_i be a convex function for all $i = 1, \ldots, m$ and $\boldsymbol{x}^* \in \mathbb{R}^n$. Then $\boldsymbol{x}^*$ is a weakly Pareto optimal solution of the problem* (1) *if and only if*

$$\mathbf{0} \in \text{conv } F(\boldsymbol{x}^*),$$

where conv *denotes the convex hull of the set.*

3 Multiple subgradient descent bundle method

The multiple subgradient descent bundle method (MSGDB) extends the ideas of the multiple-gradient descent algorithm (MGDA) [7, 8] for the convex nonsmooth case. MGDA, in its turn, is a generalization of the classical steepest descent method for smooth multiobjective optimization. In MGDA, the direction of the negative gradient is utilized. This direction is known to be descent for a smooth function [3, 29]. By knowing the gradients giving descent directions for every objective separately, the convex hull of these gradients can be formulated. Then the minimum norm element of that convex hull can be found. The negative direction of this minimum norm element gives a common descent direction for all the objective functions [7, 8].

3.1 The sketch of the method

In the following, we assume that the objective functions are nonsmooth, and thus classical gradients cannot be employed. Instead of gradients, we utilize subgradients in MSGDB. However, if we only replace gradients with subgradients, the descent direction cannot be ensured. This is due to the fact that there is no guarantee that the opposite direction of an arbitrary subgradient would be a descent direction [29, 41]. If the whole subdifferentials of the objective functions were

known, the steepest descent directions could be calculated but this is, in the most cases, too a demanding requirement in practice. This justifies the use of the bundle approach in order to find a descent direction.

In MSGDB, we first use the proximal bundle idea to find descent directions separately for every objective. Then, we form the convex hull of these directions, and seek the minimum norm element of that hull. In the smooth case, it can be proved that the direction obtained is a common descent direction. However, in the nonsmooth case, we cannot guarantee that the direction obtained is a common descent direction, but we get a good candidate for that. Thus, compared with MGDA, we have to make extra null step and optimality checking to be sure that the common descent direction is obtained. The simplified flowchart of MSGDB is given in Figure 1.

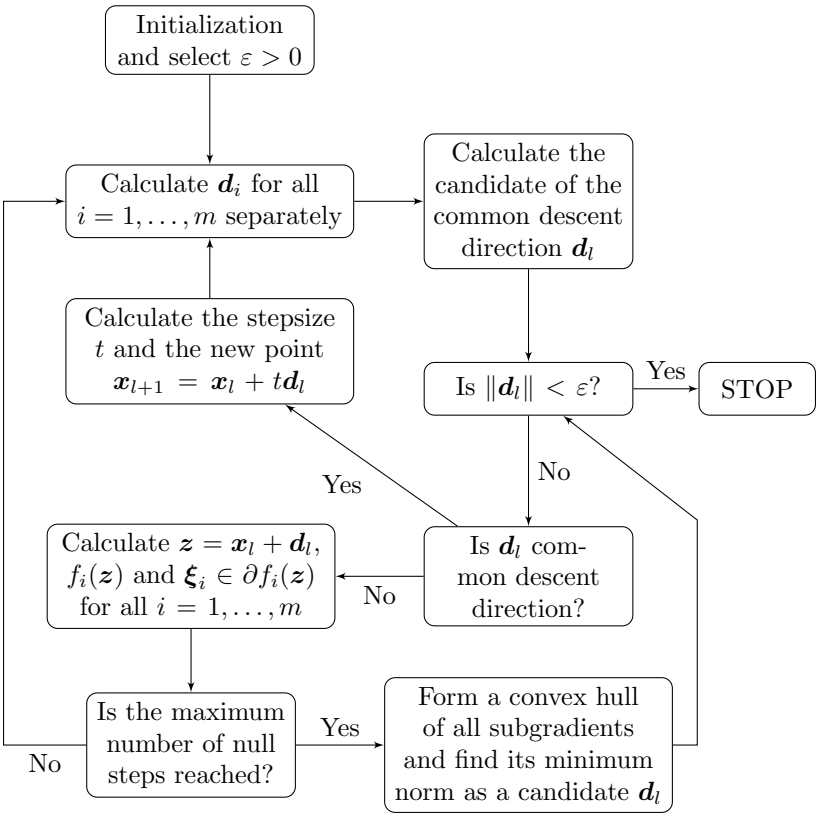

Figure 1: Flowchart of MSGDB

Due to the formulation of MSGDB, if the objective functions are smooth, then the method reduces back to MGDA, since the subdifferential of a smooth function consist of only its gradient. Moreover, if we have only one objective, we would obtain the same search direction with MSGDB than with the proximal bundle method [20].

We start our overview on MSGDB by presenting the following theorem giving the theoretical base for the algorithm.

Theorem 3.1. *Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be a convex function for all $i = 1, \ldots, m$ and let $\boldsymbol{p}^* = \operatorname{argmin} \|\boldsymbol{p}\|$, where $\boldsymbol{p} \in \operatorname{conv} F(\boldsymbol{x})$ and $\boldsymbol{d}^* = -\boldsymbol{p}^*$. Either we have*

i) $\boldsymbol{d}^* = \boldsymbol{0}$ and the point $\boldsymbol{x}$ is weakly Pareto optimal, or

ii) $\boldsymbol{d}^* \neq \boldsymbol{0}$ and the vector $\boldsymbol{d}^*$ is a common descent direction for every objective function.

Proof. Consider the first case. Now $\boldsymbol{0} \in \operatorname{conv} F(\boldsymbol{x})$ implying that $\boldsymbol{x}$ is weakly Pareto optimal according to Theorem 2.2.

Consider then the second case, where $\boldsymbol{d}^* \neq \boldsymbol{0}$ implying that $\boldsymbol{0} \notin \operatorname{conv} F(\boldsymbol{x})$. Thus, $\boldsymbol{0} \notin \partial f_i(\boldsymbol{x})$ for all $i = 1, \ldots, m$. Since the subdifferential is a convex and compact set, it follows from Lemma 2 in [27] that the set conv $F(\boldsymbol{x})$ is compact. Then, Lemma 5.2.6 in [29] implies that $\boldsymbol{p}^T\boldsymbol{p}^* \geq \|\boldsymbol{p}\|^2$ for all $\boldsymbol{p} \in \operatorname{conv} F(\boldsymbol{x})$. Let $\boldsymbol{\xi}_i \in \partial f_i(\boldsymbol{x}) \subset \operatorname{conv} F(\boldsymbol{x})$ be an arbitrary subgradient. Now

$$\boldsymbol{\xi}_i^T \boldsymbol{d}^* = -\boldsymbol{\xi}_i^T \boldsymbol{p}^* \leq -\|\boldsymbol{\xi}_i\|^2 < 0.$$

Then, according to (3), $\boldsymbol{d}^*$ is a descent direction for f_i. The same holds for every f_i, $i = 1, \ldots, m$. Therefore, $\boldsymbol{d}^*$ is a common descent direction. □

Theorem 3.1 states that we either obtain a common descent direction or a zero vector by taking the minimum norm element of the convex hull of $F(\boldsymbol{x})$. If the zero vector is obtained, the point is weakly Pareto optimal. However, in our algorithm we approximate the convex hull of $F(\boldsymbol{x})$ with the convex hull of individual descent directions. Thus, Theorem 3.1 does not guarantee that we necessarily obtain a common descent direction. Nevertheless, we obtain at least a good candidate for it.

Next, we consider more specific how these individual directions are generated with the proximal bundle approach. The basic idea behind the proximal bundle method is to approximate the whole subdifferential of the objective function by gathering information from the neighborhood of the iteration point into the bundle. Thus, only one arbitrary subgradient from the subdifferential and the value of the function at each iteration point need to be evaluated. We refer to [2, 18, 19, 20, 25, 29, 40] for more details about the bundle methods.

In our algorithm (see Step 2 of Algorithm 1), we calculate the descent directions for individual objective functions f_i, $i = 1, \ldots, m$. These calculations are performed for every objective separately. Consider an iteration point $\boldsymbol{x}_k$ at iteration k and some auxiliary points $\boldsymbol{y}_{i,j}$, $j \in J_{i,k}$ from the past iterations where $J_{i,k}$ is a nonempty subset of $\{1, \ldots, k\}$. In addition, some arbitrary subgradients $\boldsymbol{\xi}_{i,j} \in \partial f_i(\boldsymbol{y}_{i,j})$ for $i = 1, \ldots, m$ and $j \in J_{i,k}$ are supposed to be known.

The following piecewise linear model, also called a *cutting plane model* [20, 29, 40], is formed at the k-th iteration to approximate the function f_i:

$$\hat{f}_i^k(\boldsymbol{x}) = \max_{j \in J_{i,k}} \left\{ f_i(\boldsymbol{x}_k) + \boldsymbol{\xi}_{i,j}^T(\boldsymbol{x} - \boldsymbol{x}_k) - \alpha_{i,j}^k \right\} \text{ for all } i = 1, \ldots, m, \tag{5}$$

where the linearization error $\alpha_{i,j}^k$ is defined by

$$\alpha_{i,j}^k = f_i(\boldsymbol{x}_k) - f_i(\boldsymbol{y}_{i,j}) - \boldsymbol{\xi}_{i,j}^T(\boldsymbol{x}_k - \boldsymbol{y}_{i,j}) \text{ for all } j \in J_{i,k}. \tag{6}$$

Since the objective function f_i is convex, $\alpha_{i,j}^k \geq 0$ by the definition of subdifferential. From this it follows that $\hat{f}_i^k$ is a lower approximation for f_i.

The search direction $\boldsymbol{d}_{i,k}$ can then be calculated from the formula

$$\boldsymbol{d}_{i,k} = \underset{\boldsymbol{d}_i \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \hat{f}_i^k(\boldsymbol{x}_k + \boldsymbol{d}_i) + \frac{1}{2} u_{i,k} \left\| \boldsymbol{d}_i \right\|^2 \right\} \text{ for all } i = 1, \ldots, m, \tag{7}$$

where $u_{i,k}$ is a positive weighting parameter and $\frac{1}{2} u_{i,k} \left\| \boldsymbol{d}_i \right\|^2$ is a stabilizing term guaranteeing the existence and the uniqueness of the solution and keeping the approximation local enough.

It is possible to rewrite the nonsmooth direction finding problem (7) for each objective function f_i in the following smooth form

$$\begin{aligned} \min \quad & v_i + \frac{1}{2} u_{i,k} \left\| \boldsymbol{d}_i \right\|^2 \\ \text{s. t.} \quad & \boldsymbol{\xi}_{i,j}^T \boldsymbol{d}_i - \alpha_{i,j}^k \leq v_i \quad \text{for all } j \in J_{i,k} \\ & \boldsymbol{d}_i \in \mathbb{R}^n, v_i \in \mathbb{R}. \end{aligned} \tag{8}$$

The problem (8) can be made easier by solving its quadratic dual problem

$$\begin{aligned} \min \quad & \frac{1}{2u_{i,k}} \Big\| \sum_{j \in J_{i,k}} \lambda_{i,j} \boldsymbol{\xi}_{i,j} \Big\|^2 + \sum_{j \in J_{i,k}} \lambda_{i,j} \alpha_{i,j}^k \\ \text{s. t.} \quad & \sum_{j \in J_{i,k}} \lambda_{i,j} = 1 \\ & \lambda_{i,j} \geq 0, \quad \text{for all } j \in J_{i,k}. \end{aligned} \tag{9}$$

If $\lambda_{i,j}$ for all $j \in J_{i,k}$ solve the problem (9), then a unique solution of the problem (8) is obtained in the form [29]

$$\begin{aligned} \boldsymbol{d}_{i,k} &= -\frac{1}{u_{i,k}} \sum_{j \in J_{i,k}} \lambda_{i,j} \boldsymbol{\xi}_{i,j} \\ v_{i,k} &= -\Big(\frac{1}{u_{i,k}} \Big\| \sum_{j \in J_{i,k}} \lambda_{i,j} \boldsymbol{\xi}_{i,j} \Big\|^2 + \sum_{j \in J_{i,k}} \lambda_{i,j} \alpha_{i,j}^k \Big). \end{aligned} \tag{10}$$

After that, a new auxiliary point $\boldsymbol{y}_{i,k+1} = \boldsymbol{x}_k + \boldsymbol{d}_{i,k}$ and the function value $f_i(\boldsymbol{y}_{i,k+1})$ are calculated. The procedure can be stopped and set $\boldsymbol{d}_i = \boldsymbol{d}_{i,k}$ if

$$f_i(\boldsymbol{y}_{i,k+1}) \leq f_i(\boldsymbol{x}_k) + m_L v_{i,k}, \tag{11}$$

where $m_L \in (0, \frac{1}{2})$ is a line search parameter. Note that $v_{i,k}$ has the following form [29]

$$v_{i,k} = \hat{f}_i^k(\boldsymbol{y}_{i,k+1}) - f_i(\boldsymbol{x}_k)$$

being a predicted descent of the function f_i at the point $\boldsymbol{x}_k$. Since $\hat{f}_i^k \leq f_i$ and the condition (11) is satisfied, $v_{i,k} < 0$. This implies that the function value obtained at the new iteration point is significantly better than the function value at the previous iteration point. If the condition (11) does not hold, we perform a *null step* where the model will be improved by adding new information to the bundle. This is done by updating the bundle such that a new index is added to the set $J_{i,k+1} = J_{i,k} \bigcup \{k+1\}$. Additionally, the subgradient $\boldsymbol{\xi}_{i,k+1} \in \partial f_i(\boldsymbol{y}_{i,k+1})$, the auxiliary point $\boldsymbol{y}_{i,k+1}$ and the function value $f_i(\boldsymbol{y}_{i,k+1})$ are added to the bundle. The iteration point is not updated. That is, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$. After that, a new value for the direction $\boldsymbol{d}_{i,k+1}$ can be calculated.

In standard single-objective bundle methods, if the condition (11) holds, a new iteration point $\boldsymbol{x}_{k+1} = \boldsymbol{y}_{i,k+1}$ is calculated. This step is called a *serious step*. In our method, we utilize this idea by performing null steps until the condition (11) is satisfied, and the sufficient descent is reached. Then, the direction obtained is denoted with $\boldsymbol{d}_i$, and the bundle procedure is stopped. Nevertheless, the new iteration point is not yet calculated.

Above we have shown how to calculate descent directions for individual objective functions f_i, $i = 1, \ldots, m$. In Step 3 of Algorithm 1, we find a candidate for the common descent direction for all the objective functions. This candidate is calculated as a minimum norm element of the convex hull of the descent directions similar to MGDA in the smooth case. We emphasize that Theorem 3.1 does not guarantee that we actually find a common descent direction as in the smooth case, but at least we obtain a good candidate for that.

In our nonsmooth case, we consider the following set C:

$$C = \text{conv}\,\{\boldsymbol{d}_i \,|\, i = 1, \ldots, m\}. \tag{12}$$

The minimum norm element of C can be found by solving

$$\begin{aligned} \min \quad & \left\| \sum_{i=1}^{m} \lambda_i \boldsymbol{d}_i \right\|^2 \\ \text{s.t.} \quad & \sum_{i=1}^{m} \lambda_i = 1 \\ & \lambda_i \geq 0, \quad \text{for all } i = 1, \ldots, m, \end{aligned} \tag{13}$$

which has a unique solution λ_i for all $i = 1, \ldots, m$, since the objective function of (13) is strictly convex. Once we have calculated $\boldsymbol{d}_l^* = -\sum_{i=1}^m \lambda_i \boldsymbol{d}_i$, we have to test whether the candidate $\boldsymbol{d}_l^*$ improves the values of all the objectives or not. This is done in Step 4 of Algorithm 1 among the test of the stopping condition.

If we have found a common descent direction for all the objective functions, we calculate a new stepsize. The stepsize t is calculated similarly to MGDA [7, 8] by applying some line search strategy in order to find a stepsize $t_i \geq \tau > 0$ for all $i = 1, \ldots, m$, where τ is a stepsize tolerance. The stepsize t is then selected by $t = \min\{t_i \,|\, i = 1, \ldots, m\}$, and the new iteration point $\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + t\boldsymbol{d}_l$ is calculated. Note that, for example, with the line search procedure in [42], it can be guaranteed that we either find the stepsize $t_i \geq \tau$, or we perform a null step (see Lemma 2.1 in [42]). By performing a null step for individual objectives, we obtain new directions $\boldsymbol{d}_{i,k}$ for all $i = 1, \ldots, m$ [19].

If we have not found a common descent direction, then we try to find another candidate, and test whether that is descent or not. This is done in Steps 5 and 2B of the Algorithm 1. These steps form a cycle being like a null step in the single-objective proximal bundle method. This null step can be repeated a couple of times and the number of null steps is denoted with n_{null}. In practice, null step is performed by calculating an auxiliary point $\boldsymbol{z} = \boldsymbol{x}_l + \boldsymbol{d}_l$ and subgradients $\boldsymbol{\xi}_i \in \partial f_i(\boldsymbol{z})$, and adding those to the corresponding bundle. After that, we return to Step 2B to calculate new individual descent directions separately, and then calculate a new candidate. However, if a reasonable number of these null steps, marked with n_{max}, do not pay dividends, we have a hint that we might have reached a weakly Pareto optimal point. Due to the formulation of the algorithm and the nonsmooth nature of the problem, we might have lost some crucial information needed to find the direction equalizing to zero vector, even if we already are in a weakly Pareto optimal point. In the following, we give an example when this kind of situation may happen.

Example 1. Consider the problem (1) with $n = m = 2$ and the objective functions $f_1(\boldsymbol{x}) = |x_1| + |x_2| + 2x_1$ and $f_2(\boldsymbol{x}) = |x_1| + |x_2| + 2x_2$. If we select $\boldsymbol{x}_1 = (0, 0)^T$ as a starting point, we can take, for instance, subgradients $\boldsymbol{\xi}_{1,1} = (0, 1)^T$ and $\boldsymbol{\xi}_{2,1} = (1, 0)^T$. If we now use the described method to find the minimum norm element, we get a candidate for the search direction $\boldsymbol{d}_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Now $\|\boldsymbol{d}_1\| = 1 > \varepsilon > 0$. The stopping condition $\|\boldsymbol{d}_l\| < \varepsilon$ does not hold and $\boldsymbol{d}_1$ is not a common descent direction. However, the point $\boldsymbol{x}_1 = (0, 0)^T$ is weakly Pareto optimal since the convex hull of $\partial f_1((0, 0)) = [1, 3] \times [-1, 1]$ and $\partial f_2((0, 0)) = [-1, 1] \times [1, 3]$ contains a zero vector.

That is why we test (see Step 6 in Algorithm 1) whether the optimality condition holds at this point. In addition, this way we can guarantee the convergence of the method. If the null step is already performed the maximum number of times, or $n_{null} > n_{max}$, we sacrifice more computational effort and we try to improve our approximation of $F(\boldsymbol{x}_l)$. That is, we test the optimality by taking a convex hull of

all the subgradients stored, and find the minimum norm element of that hull. That is, we are solving the problem

$$\begin{aligned} \min \quad & \left\| \sum_{i=1}^{m} \sum_{j \in J_{i,k}} \lambda_{i,j} \boldsymbol{\xi}_{i,j} \right\|^2 + \sum_{i=1}^{m} \sum_{j \in J_{i,k}} \lambda_{i,j} \alpha_{i,j}^k \\ \text{s.t.} \quad & \sum_{i=1}^{m} \sum_{j \in J_{i,k}} \lambda_{i,j} = 1 \\ & \lambda_{i,j} \geq 0, \quad \text{for all } i = 1, \ldots, m,\ j \in J_{i,k}. \end{aligned} \tag{14}$$

Now we have the more accurate approximation of $F(\boldsymbol{x}_l)$ in the problem (14) than in the problem (13). While the problem (14) is used to ensure that we do not lose any crucial information about weak Pareto optimality, we obtain a common descent direction if our current solution is not optimal. This is due to the fact that the problem (14) corresponds to the one where we seek the descent direction for the improvement function $H(\,\cdot\,, \boldsymbol{x}_l)$ defined in (2) (see (22) in Appendix A). By Lemma 2.1, a descent direction for $H(\,\cdot\,, \boldsymbol{x}_l)$ is a common descent direction for every objective. We are now in a position to give a detailed description of the algorithm for MSGDB in Algorithm 1.

In practice, the size of the bundles need to be limited. The easiest way to do this is to choose some maximal size for the bundles, for example, $J_{max} = n + 5$. The set $J_{i,k+1}$ is updated as in Step 2C if $|J_{i,k}| < J_{max}$, and if $|J_{i,k}| = J_{max}$, then a set

$$J_{i,k+1} = J_{i,k} \bigcup \{k+1\} \setminus \{k - J_{max}\} \tag{15}$$

is used. Another possible limiting strategy is the subgradient aggregation strategy [17]. Additionally, the parameter $u_{i,k+1}$ is updated in Step 2C and this can be done, for example, with a weight updating algorithm presented in [20].

Algorithm 1 Multiple subgradient descent bundle method (MSGDB)

Step 1: (*Initialization*) Select the starting point $\boldsymbol{x}_1$, the line search parameter $m_L \in (0, \frac{1}{2})$, the stopping parameter $\varepsilon > 0$, the maximum number of null steps n_{max} and the stepsize tolerance $\tau > 0$. Set an outer iteration index $l = 1$ and the null step counter $n_{null} = 0$.

Step 2: (*Direction finding*) Do the following steps A–C for all $i = 1, \ldots, m$ to calculate directions $\boldsymbol{d}_i$.

Step A: (*Initialization*) Select the weighting parameter $u_{i,1}$. Set $\boldsymbol{y}_{i,1} = \boldsymbol{x}_1$ and $J_{i,1} = \{1\}$. Initialize an inner iteration index $k = 1$.

Step B: (*Direction finding*) Calculate a direction $\boldsymbol{d}_{i,k}$ from formula (7) and set $\boldsymbol{y}_{i,k+1} = \boldsymbol{x}_k + \boldsymbol{d}_{i,k}$. If the descent condition (11) holds, then set $\boldsymbol{d}_i = \boldsymbol{d}_{i,k}$. Otherwise go to Step C.

Step C: (*Update*) Set $J_{i,k+1} = J_{i,k} \bigcup \{k+1\}$, calculate $\boldsymbol{\xi}_{i,k+1} \in \partial f_i(\boldsymbol{y}_{i,k+1})$ and update $u_{i,k+1}$. Set $k = k + 1$ and go to Step B.

Step 3: (*Search direction candidate finding*) Calculate a minimum norm element $\boldsymbol{p}^*$ of the set C (see (12)) by solving the problem (13). Set $\boldsymbol{d}_l = -\boldsymbol{p}^*$.

Step 4: (*Stopping criterion and descent test*) If $\|\boldsymbol{d}_l\| < \varepsilon$, then stop. If $f_i(\boldsymbol{x}_l + \tau \boldsymbol{d}_l) < f_i(\boldsymbol{x}_l)$, for all $i = 1, \ldots, m$, go to Step 7. Otherwise, go to Step 5.

Step 5: (*Null step*) Calculate $\boldsymbol{\xi}_i \in \partial f_i(\boldsymbol{x}_l + \boldsymbol{d}_l)$ for all $i = 1, \ldots, m$, and add it to the corresponding bundle $J_{i,k}$ with a suitable index. Set $n_{null} = n_{null} + 1$. If $n_{null} > n_{max}$, go to Step 6. Otherwise go to Step 2B.

Step 6: (*Search direction candidate finding*) Solve the problem (14). Calculate the search direction candidate $\boldsymbol{d}_l = -\boldsymbol{p}$ and go to Step 4.

Step 7: (*Line search*) Calculate a stepsize $t \geq \tau > 0$. Set $\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + t\boldsymbol{d}_l$, $l = l + 1$ and $n_{null} = 0$. Go to Step 2B.

3.2 Convergence

The remainder of this section is devoted to prove that the solution generated by MSGDB is weakly Pareto optimal. In the following convergence analysis, we assume that the level set $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}_1), \text{ for all } i = 1, \ldots, m\}$ is bounded and the stopping parameter ε at the stopping condition in Step 4 is zero. We state three lemmas concerning infinite cycles in Algorithm 1. The proofs of the Lemmas 3.2 and 3.3 base on the convergence of the proximal bundle method [20, 29, 40] in the situation where after a finite number of serious steps the consecutive null steps are performed infinitely many times. As for Lemma 3.4, it considers the case where Algorithm 1 generates an infinite sequence of solutions $\{\boldsymbol{x}_l\}$ by cycling between Steps 2–4 and 7. After that, we can formulate the main convergence result.

Lemma 3.2. *If Step 2 of Algorithm 1 is performed infinitely with some i, $i = 1, \ldots, m$ after a finite number l of iterations, then the current solution $\boldsymbol{x}_l$ is weakly Pareto optimal solution for the problem* (1)*.*

Proof. Lemma 3.4 in [20] implies that if for some i, $i = 1, \ldots, m$ Step 2 is performed infinitely, then $\mathbf{0} \in \partial f_i(\boldsymbol{x}_l)$. Thus, $\mathbf{0} \in$ conv $F(\boldsymbol{x}_l)$ meaning that $\boldsymbol{x}_l$ is weakly Pareto optimal solution for the problem (1) by Theorem 2.2. □

Lemma 3.3. *Assume that the stopping parameter $\varepsilon = 0$. Let us consider the $(l+1)$-th round of Algorithm 1 and the cycle between Steps 4 and 6.*

i) If the cycle is passed a finite number of times, then we either obtain a search direction improving all the objectives or the current solution $\boldsymbol{x}_l$ is weakly Pareto optimal.

ii) If the cycle is infinite, then the current solution $\boldsymbol{x}_l$ is weakly Pareto optimal solution for the problem (1)*.*

Proof. If Steps from 4 to 6 are executed finitely, we obtain from Step 4 that either $\|\boldsymbol{d}_l\| = 0$ and the solution $\boldsymbol{x}_l$ is weakly Pareto optimal by Theorem 3.1, or the direction $\boldsymbol{d}_l$ improves all the objectives f_i, $i = 1, \ldots, m$ by the descent test made in Step 4.

In the case of the infinite cycle between Steps 4 and 6, we begin the proof by observing that solving the problem (14) in Step 6 is a similar problem than a dual search direction problem (22) solved in MPB (see Appendix A). Thus, by solving the problem (14), we obtain Lagrange multipliers $\lambda_{i,j} \geq 0$ for all $i = 1, \ldots, m$, $j \in J_{i,k}$ such that $\lambda_{i,j} = 0$ for those indices $i \notin I(\boldsymbol{x})$, where $I(\boldsymbol{x}) = \{i \,|\, f_i(\boldsymbol{x}) - f_i(\boldsymbol{x}_l) = H_i(\boldsymbol{x}, \boldsymbol{x}_l)\}$ and $\sum_{i=1}^{m} \sum_{j \in J_{i,k}} \lambda_{i,j} = 1$. Therefore, while we improve our approximation about $F(\boldsymbol{x}_l)$, we end up to solve the same problem which is solved when we minimize $H(\boldsymbol{x}_l + \boldsymbol{d}_l, \boldsymbol{x}_l)$ with proximal bundle approach.

That is why the infinite cycle between Steps 4 and 6 corresponds to the infinite sequence of null steps executed in MPB. By the convergence analysis of MPB in [28], the situation reduces back to the convergence of the single-objective proximal bundle method [20]. By Lemma 3.4 in [20], if we perform the null step of the proximal bundle method infinitely, then $v_l \to 0$, where

$$v_l = -\Big(u_l \|\boldsymbol{d}_l\|^2 + \sum_{i=1}^{m} \sum_{j \in J_{i,k}} \alpha_{i,j}^k\Big).$$

Now $u_l = \frac{1}{2}$, and since the objective functions f_i for all $i = 1, \ldots, m$ are convex, the linearization errors $\alpha_{i,j}^k \geq 0$ for all $i = 1, \ldots, m$, $j \in J_{i,k}$. Hence, if $v_l \to 0$, then also $\|\boldsymbol{d}_l\| \to 0$ and the sequence of directions $\{\boldsymbol{d}_l\} \to 0$. By Theorem 3.1, $\boldsymbol{x}_l$ is weakly Pareto optimal for the problem (1). □

Lemma 3.4. *Let us consider the problem* (1). *Assume that the stopping parameter ε is selected to be zero and the level set $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}_1)$, for all $i = 1, \ldots, m\}$ is bounded. The accumulation point of the infinite sequence of the solutions generated by cycle between Steps 2 – 4 and 7 in Algorithm 1 is weakly Pareto optimal.*

Proof. Suppose, that Steps 2–4 and 7 in Algorithm 1 generate the infinite sequence of the solutions

$$\{\boldsymbol{x}_l\}, \text{ where } \boldsymbol{x}_l = \boldsymbol{x}_{l-1} + t\boldsymbol{d}_l \tag{16}$$

and $\boldsymbol{x}^*$ is the accumulation point of this sequence. This accumulation point exists, since the level set is assumed to be bounded. When $\{\boldsymbol{x}_l\} \to \boldsymbol{x}^*$, from the formulation of the $\boldsymbol{x}_l$ in (16) and the fact that $t \geq \tau > 0$, it follows that the sequence of directions $\{\boldsymbol{d}_l\} \to \mathbf{0}$. Thus, $\boldsymbol{x}^*$ is weakly Pareto optimal solution for the problem (1) by Theorem 3.1. □

Summarizing, we have now considered all the cases where an infinite cycle may happen. We can now establish the convergence of Algorithm 1.

Theorem 3.5. *Let us consider the problem* (1). *Assume that the stopping parameter ε is selected to be zero and the level set $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}_1)$, for all $i = 1, \ldots, m\}$ is bounded. If MSGDB stops with a finite number of iterations, then the solution is weakly Pareto optimal. On the other hand, if there exists an infinite cycle, the accumulation point generated by MSGDB is weakly Pareto optimal.*

Proof. Assume first that Algorithm 1 stops with a finite number of iterations. Then, $\|\boldsymbol{d}_l\| = 0$, and thus $\boldsymbol{d}_l = \mathbf{0}$. According to Theorem 3.1, the solution $\boldsymbol{x}_l$ is weakly Pareto optimal.

Assume then that during the execution of Algorithm 1 there exists an infinite cycle after a finite number of cycles between Steps 2–4 and 7. If that infinite cycle exists in Step 2, then according to Lemma 3.2, the current solution is weakly Pareto optimal for the problem (1). If that cycle exists between Steps 4-6, then Lemma 3.3 implies that the current solution is weakly Pareto optimal for the problem (1). In the case where the infinite cycle exists between Steps 2–4 and 7, the accumulation point of the sequence of the solutions generated by MSGDB is weakly Pareto optimal solution for the problem (1) by Lemma 3.4. □

4 Computational experiments

In this section, we numerically compare MSGBD with MPB described in Appendix A. At first, we compare the search directions generated by the methods in order to show that the search directions obtained with different methods are not necessarily the same direction. After that, we describe the implementations of the methods and give some computational examples and analyze the results.

4.1 Comparing search directions

At first, we consider two simple examples where we calculate the search directions which we obtain at the first iteration round. One search direction is calculated with MSGDB and one with MPB. We apply two different types of weighting parameters u_k, one with $u_k = 2u_{k-1}$, where $u_1 = 1$ and the other with $u_k = 1$ for all k. After that, we calculate stepsizes. In these examples in Section 4.1, we use the exact line search.

Both example problems are of the form

$$\begin{aligned} \min \quad & \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\} \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{R}^2. \end{aligned} \tag{17}$$

In the first problem, the convex objective functions in the problem (17) are

$$\begin{aligned} f_1(\boldsymbol{x}) &= \max\left\{x_1^2 + (x_2 - 1)^2,\ (x_1 + 1)^2\right\} \\ f_2(\boldsymbol{x}) &= \max\left\{2x_1 + 2x_2,\ x_1^4 + x_2^2\right\} \end{aligned}$$

and the function values at the starting point $\boldsymbol{x}_0 = (0, 2)^T$ are $f_1(\boldsymbol{x}_0) = 1.00$ and $f_2(\boldsymbol{x}_0) = 4.00$. We get the results shown in Table 1.

In the second problem, the convex objective functions in the problem (17) are

$$\begin{aligned} f_1(\boldsymbol{x}) &= \max\left\{(x_1 - 2)^2 + (x_2 + 2)^2, x_1^2 + 8x_2\right\} \\ f_2(\boldsymbol{x}) &= \max\left\{5x_1 + x_2, x_1^2 + x_2^2\right\} \end{aligned}$$

and the function values at the starting point $\boldsymbol{x}_0 = (1, 2)^T$ are $f_1(\boldsymbol{x}_0) = 17.00$ and $f_2(\boldsymbol{x}_0) = 7.00$. We obtain the results shown in Table 2.

Table 1: The first example

	$u_k = 2u_{k-1}$		$u_k = 1$	
	MSGDB	MPB	MSGDB	MPB
$\boldsymbol{d}$	$(0.5000, 0.5000)$	$(0.3824, 0.5294)$	$(0.4000, 0.8000)$	$(0.4000, 0.8000)$
t	1.0000	1.0064	1.1335	1.1335
$\boldsymbol{x}_2$	$(-0.5000, 0.5000)$	$(-0.3848, 0.5294)$	$(-0.4534, 1.0932)$	$(-0.4534, 1.0932)$
$f_1(\boldsymbol{x}_2)$	0.5000	0.3785	0.2988	0.2988
$f_2(\boldsymbol{x}_2)$	2.3125	0.2923	1.2796	1.2796

Table 2: The second example

	$u_k = 2u_{k-1}$		$u_k = 1$	
	MSGDB	MPB	MSGDB	MPB
$\boldsymbol{d}$	$(2.0558, 3.0107)$	$(0.4122, 0.1765)$	$(1.9527, 2.9901)$	$(1.8459, 1.5986)$
t	0.6077	1.4612	0.6220	0.7628
$\boldsymbol{x}_2$	$(-0.2493, 0.1704)$	$(0.3977, 1.7421)$	$(-0.2146, 0.1402)$	$(-0.4081, 0.7806)$
$f_1(\boldsymbol{x}_2)$	9.7700	16.5700	9.4849	13.5307
$f_2(\boldsymbol{x}_2)$	0.2326	3.7306	0.0657	0.7759

From these two examples, we notice that with MSGDB and MPB we do not necessarily obtain the same search directions. For example, in Table 1, we have two cases with different weighting parameters. In the first case, directions are different and in the second case we obtain the same directions.

In addition, we cannot say which one yields better directions. As we see, in the first case of the first example the direction calculated with MPB gives a better point than the direction calculated with MSGDB. In this case, the better point refers to the point where both objective functions f_1 and f_2 obtain smaller value. However, in the second example MSGDB gives a better point than MPB in both cases. Thus, based on the way to calculate the search direction we cannot say that one method is always better than another.

4.2 Implementation and numerical results

In numerical experiments, we have used single-objective convex test problems CB3, DEM, QL, LQ, Mifflin1 and Wolfe described in [24] and combined these functions in order to obtain twenty multiobjective problems. The combinations used are described in Table 3. All our test problems are nonsmooth and convex. The dimension of all the test problems is two. In the first 15 problems, we have

two objectives and the last five problems have three objectives.

Table 3: Test problems

Problem	Objectives	$\boldsymbol{x}_0$	Problem	Objectives	$\boldsymbol{x}_0$
1.	CB3 & DEM	$(2, 2)$	11.	QL & Mifflin 1	$(2, 4)$
2.	CB3 & QL	$(-1, -1)$	12.	QL & Wolfe	$(2, 2)$
3.	CB3 & LQ	$(2, 2)$	13.	LQ & Mifflin 1	$(-0.5, -0.5)$
4.	CB3 & Mifflin 1	$(2, 2)$	14.	LQ & Wolfe	$(-2, -2)$
5.	CB3 & Wolfe	$(2, 2)$	15.	Mifflin 1 & Wolfe	$(-0.5, -0.5)$
6.	DEM & QL	$(2, 4)$	16.	CB3, DEM & QL	$(0.8, 0.6)$
7.	DEM & LQ	$(1, 1)$	17.	LQ, Mifflin 1 & Wolfe	$(-0.5, -0.5)$
8.	DEM & Mifflin 1	$(-2, -2)$	18.	DEM, QL & LQ	$(0.8, 0.6)$
9.	DEM & Wolfe	$(1, 1)$	19.	CB3, Mifflin 1 & Wolfe	$(2, 2)$
10.	QL & LQ	$(2, 4)$	20.	DEM, LQ & Wolfe	$(1, 1)$

We have used the implementation of MPB described in [26] where the two-point line search algorithm [29] is employed. In MSGDB, we apply line search which is based on Armijo type rule [1] due to its simplicity. Both the methods are implemented in Fortran, MPB in Fortran77 and MSGDB in Fortran95. To make the methods more comparable, we have applied the same quadratic solver by Lukšan described in [23] with both implementations. Moreover, the weighting parameter u_k is updated with the weight updating algorithm described in [20]. In both methods, the size of the bundle is bounded by choosing the set (15) and the value of J_{max} to be $n + 5$. Other parameters for MSGDB are selected as follows: the stopping parameter $\varepsilon = 10^{-5}$, the line search parameter $m_L = 0.25$, the maximum number of null steps $n_{max} = 2$ and the stepsize tolerance $\tau = 0.001$. The parameters for MPB are default values [26] such that the stopping criteria are the same in both methods and the locality measure is zero in MPB due to the convexity of the objectives. In the following, we consider one test problem closer and after that we analyze the results of several tests.

Let us take a closer look at the test problem number 3. In that problem, the objective functions are combined from test problems CB3 and LQ [24]. Thus objective functions of the problem (17) are now

$$\begin{aligned} f_1(\boldsymbol{x}) &= \max\{x_1^4 + x_2^2, (2 - x_1)^2 + (2 - x_2)^2, 2e^{x_2 - x_1}\} \\ f_2(\boldsymbol{x}) &= \max\{-x_1 - x_2, -x_1 - x_2 + x_1^2 + x_2^2 - 1\}. \end{aligned}$$

The starting point is chosen to be $\boldsymbol{x}_0 = (2, 2)^T$. The performance of MSGDB is described in Table 4 where current points and the function values at those points are listed at every iteration. As we see, the value of both objective functions decreases at every iteration.

Table 4: The performance of the MSGDB algorithm with test problem 3

Iteration	$\boldsymbol{x}$	$f(\boldsymbol{x})$
0	$(2, 2)$	$(20.0000, 3.0000)$
1	$(1.1503, 1.5844)$	$(4.2613, 0.0989)$
2	$(0.9615, 1.4862)$	$(3.3797, -0.3145)$
3	$(0.9592, 1.3838)$	$(3.0577, -0.5081)$
4	$(0.9604, 1.2878)$	$(2.7747, -0.6674)$
5	$(0.9644, 1.1981)$	$(2.5166, -0.7970)$
6	$(1.0076, 1.0476)$	$(2.1280, -0.9425)$
7	$(1.0040, 0.9927)$	$(2.0067, -1.0033)$

The performance of MSGDB in this test problem is also illustrated in Figure 2. In this figure, gray dashed contours correspond the contours of the first objective function and gray dotted contours correspond the contours of the second objective function. The optimal points of the first and second objective are marked with white and black square, respectively. The value of the function at the optimal point of the first objective is $f = (2, -1)$ and at the optimal point of the second objective $f = (3.34, -1.41)$. The white circles are iteration points of MSGDB. Now we can see that the solution obtained $f^* = (2.0067, -1.0033)$ is very close to the optimal point of the first objective function.

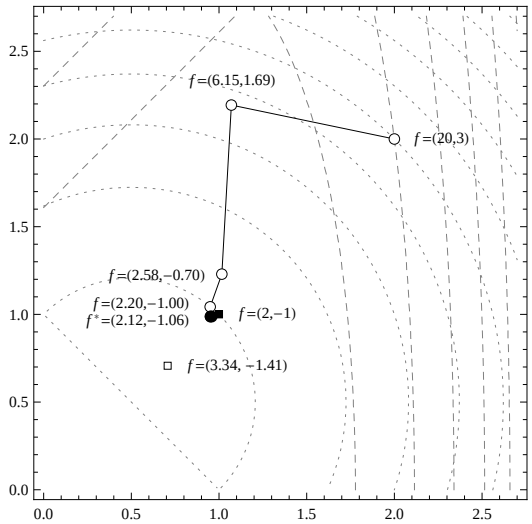

Figure 2: The performance of the algorithm in decision space for test problem 3

In Figures 3 and 4, we illustrate the situation where MSGDB is run ten times with different starting points. In Figure 3, we have solutions obtained in the decision space marked with black points. Additionally, three paths of the algorithm are illustrated with black lines and white circles in order to demonstrate the performance of MSGDB. In Figure 4, these solutions are depicted in the objective space.

Again, in Figures 3 and 4, squares represent single-objective optimal points.

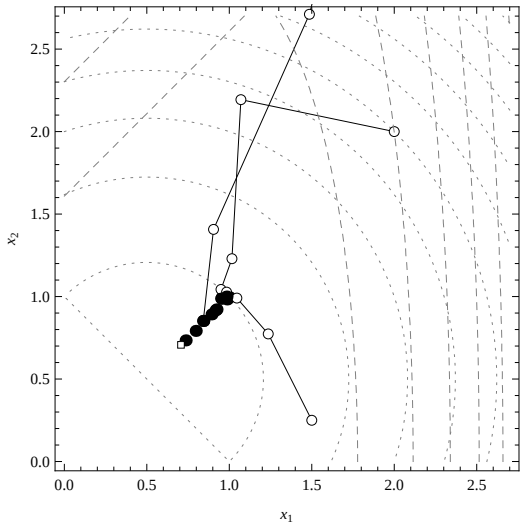

Figure 3: The solutions obtained in decision space for test problem 3 with several starting points

From Figures 3 and 4, we can observe that the solution obtained is quite sensitive for the selection of the starting point. With different starting points we can generate different (weakly) Pareto optimal solutions and obtain an approximation of the Pareto optimal set.

In Table 5, the obtained function values for all the presented test problems obtained with MSGDB and MPB are described. In addition, the number of iterations and subgradient calls are listed. In the implementation of MPB, the subgradients are called at the same time for all the objectives, and in the implementation of MSGDB, the subgradients of objective functions are called separately. Thus, there are three subgradient call columns ($\boldsymbol{\xi}_1$, $\boldsymbol{\xi}_2$ and $\boldsymbol{\xi}_3$) for MSGDB and only one column ($\boldsymbol{\xi}$) for MPB in Table 5.

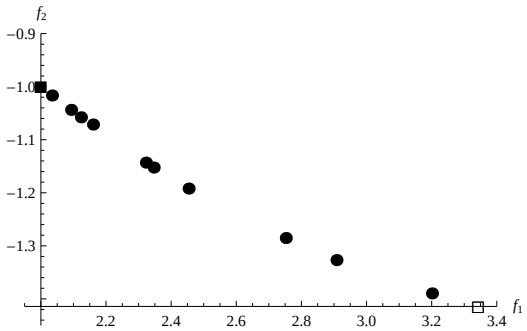

Figure 4: The solutions obtained in objective space for test problem 3 with several starting points

From results in Table 5, we can conclude that the number of iterations are approximately the same order, since according to the results, the average iterations needed for MSGDB is 5.35 and for MPB is 9.75. Even if the average number of iterations is slightly smaller with MSGDB, the average numbers of subgradient calls in this implementation 21.70, 21.85 and 18.20 are larger than the average number of subgradient calls 11.10 needed with MPB, but they are still on the same magnitude. Additionally, we observe that the methods produce mainly different weakly Pareto optimal solutions since the median relative distance of solutions in the objective space is 1.1409 varying in the interval from 0.0003 to 89.1013.

In tests performed with MSGDB, we do not find a direction improving all the objectives by solving the problem (13) in six test problems performed. Thus, the problem (14) was solved once to stop the execution of the algorithm in these six cases. To compare the computational efforts of MSGDB and MPB, we observe that in MSGDB we solve more quadratic problems than in MPB. However, the quadratic problems in MSGDB besides the problem (14) are smaller. In addition, it is worth noting that the implementation of MPB is the result of long development and testing process contrary to the implementation of MSGDB being only the first implementation.

Table 5: Results of numerical tests

	MSGDB					MPB		
Problem	ξ_1	ξ_2	ξ_3	Iteration	$f(\boldsymbol{x}^*)$	ξ	Iteration	$f(\boldsymbol{x}^*)$
1.	29	26		8	$(2.0028, 6.0096)$	8	7	$(3.5601, 3.7825)$
2.	22	20		6	$(2.4373, 28.8838)$	13	12	$(4.3879, 17.8486)$
3.	25	23		7	$(2.0067, -1.0033)$	5	4	$(2.0298, -1.0147)$
4.	25	23		7	$(2.0071, 18.8567)$	18	17	$(2.0503, 18.0151)$
5.	28	28		8	$(2.0090, 24.9150)$	9	8	$(2.0065, 24.9059)$
6.	21	33		5	$(16.7246, 7.2019)$	8	7	$(16.8000, 7.2000)$
7.	11	15		3	$(4.2427, -1.4142)$	6	5	$(4.1156, -1.4129)$
8.	27	24		6	$(-1.0492, 4.3114)$	8	7	$(-2.3812, 93.4027)$
9.	12	12		2	$(1.1712, -0.2522)$	5	4	$(-0.0334, 0.5350)$
10.	43	27		7	$(7.2039, 2.5761)$	7	6	$(7.3908, 2.5145)$
11.	17	15		3	$(7.2001, 122.7955)$	13	12	$(11.1826, 99.1489)$
12.	15	13		2	$(12.3669, 47.9562)$	12	11	$(7.4091, 48.7891)$
13.	20	37		6	$(-1.0115, -0.9998)$	14	13	$(-1.0682, -0.9975)$
14.	44	43		13	$(1.0214, -7.4496)$	20	19	$(-1.0392, 12.4706)$
15.	13	14		3	$(0.2407, -1.8086)$	13	12	$(-0.2198, 3.2972)$
16.	11	11	10	2	$(3.2831, 4.2314, 38.6799)$	5	4	$(3.1316, 4.3116, 38.1779)$
17.	13	13	14	3	$(0.2183, 0.2407, -1.8086)$	25	17	$(-0.1612, -0.1620, 2.4294)$
18.	19	19	28	5	$(4.2341, 39.7545, -1.4140)$	9	8	$(4.2318, 39.7598, -1.4142)$
19.	28	26	28	8	$(2.0090, 18.8204, 24.9150)$	18	17	$(2.7168, 7.0171, 20.8194)$
20.	11	15	11	3	$(4.2427, -1.4142, 17.6777)$	6	5	$(4.2427, -1.4142, 17.6780)$
Average	21.70	21.85	18.20	5.35		11.10	9.75	

5 Conclusions

We have proposed a new descent bundle based method for convex unconstrained multiobjective optimization. The method generalizes ideas from multiple-gradient descent algorithm (MGDA) and combines them with the proximal bundle method. In order to find a common descent direction for all the objectives, the idea of MGDA is utilized, and in order to obtain the descent directions for each objective separately, the idea of the proximal bundle is used. In addition, we have added null step and optimality condition checking in order to guarantee that our candidate truly is a descent direction before using it or that we can stop in weakly Pareto optimal point. Thus, in the case of a single-objective function, the search direction generated with multiple subgradient descent bundle method (MSGDB) is similar to the search direction generated with the proximal bundle method and in the case of differentiable objective functions MSGDB is similar to MGDA.

We have considered the basic idea of multiobjective proximal bundle method (MPB) as well and compared our MSGDB with it. MPB is chosen for the reference method since it is also a descent method for nonsmooth multiobjective optimization utilizing the proximal bundle idea. When in MSGDB the proximal bundle approach is used in order to find the descent direction for all objectives separately and after that to find one common descent direction, in MPB all objectives are taken into consideration at the same time with the improvement function and the proximal bundle idea is used in order to find a descent direction for this improvement function.

We have seen that the methods described may produce different directions and we cannot say that one would always be better than another. According to numerical experiments, we have shown that the number of iterations needed with MSGDB is small and the same order that is needed with MPB. However, the number of subgradient calls is larger but still same magnitude. It's worth of noting, that the implementation of MSGDB is only the first implementation and it could be improved.

In addition, we observed that the methods produce different weakly Pareto optimal solutions. Usually, it is useful to have several different solutions produced from the same starting point in interactive methods [30, 31, 35]. Thus, these kind of different descent methods are needed.

In order to extend MSGDB in future, the aim is to design a method which is able to solve also nonconvex and constrained multiobjective problems. Another possible development could be the invocation of the separately calculated search directions. Since every objective function has its own search direction, those might be used, for instance, in interactive methods by scaling directions according to the decision maker's preferences.

Acknowledgements

The research has been financially supported by the Finnish Academy of Science and Letters (the Vilho, Yrjö and Kalle Väisälä Foundation), Emil Aaltonen Foundation, University of Turku Graduate School UTUGS Matti programme, Academy of Finland Project No. 289500 and University of Turku.

References

[1] L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[2] A. Bagirov, N. Karmitsa, and M. M. Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, Cham Heidelberg, 2014.

[3] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, Inc., New Jersey, third edition, 2006.

[4] J. Y. Bello Cruz and A. N. Iusem. A strongly convergent method for nonsmooth convex minimization in Hilbert spaces. *Numerical Functional Analysis and Optimization*, 32(10):1009–1018, 2011.

[5] H. Bonnel, A. N. Iusem, and B. F. Svaiter. Proximal methods in vector optimization. *SIAM Journal on Optimization*, 15(4):953–970, 2005.

[6] F. H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, Inc., New York, 1983.

[7] J.-A. Désidéri. Multiple-Gradient Descent Algorithm (MGDA). Technical Report 6953, INRIA Research Report, 2009.

[8] J.-A. Désidéri. Multiple-Gradient Descent Algorithm (MGDA) for multiobjective optimization. *Compte rendus de l'Académie des sciences, Ser. I*, 350:313–318, 2012.

[9] M. Ehrgott. *Multicriteria Optimization*. Springer, Berlin, second edition, 2005.

[10] J. Fliege, L. M. Graña Drummond, and B. F. Svaiter. Newton's method for multiobjective optimization. *SIAM Journal on Optimization*, 20(2):602–626, 2009.

[11] J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[12] E. H. Fukuda and L.M. Graña Drummond. Inexact projected gradient method for vector optimization. *Computational Optimization and Applications*, 54(3):473–493, 2013.

[13] L. M. Graña Drummond and B. F. Svaiter. A steepest descent method for vector optimization. *Journal of Computational and Applied Mathematics*, 175:395–414, 2005.

[14] Ken Harada, Jun Sakuma, and Shigenobu Kobayashi. Local search for multiobjective function optimization: Pareto descent method. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 659–666, New York, NY, USA, 2006. ACM.

[15] J. Haslinger and P. Neittaanmäki. *Finite Element Approximation for Optimal Shape, Material and Topology Design*. J. Wiley & Sons, Chichester, 1996.

[16] J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms, I and II*. Springer Verlag, Berlin, 1993.

[17] K. C. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27(3):320–341, 1983.

[18] K. C. Kiwiel. A descent method for nonsmooth convex multiobjective minimization. *Large Scale Systems*, 8(2):119–129, 1985.

[19] K. C. Kiwiel. *Methods of descent for nondifferentiable optimization*. Springer-Verlag, Berlin, 1985.

[20] K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable optimization. *Mathematical Programming*, 46:105–122, 1990.

[21] C. Lemaréchal. Nondifferentiable Optimization. In G. G. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, pages 529–572. Elsevier North-Holland, Amsterdam, 1989.

[22] C. Lemaréchal, J. S. Strodiot, and A. Bihain. On a Bundle Algorithm for Nonsmooth Optimization. In O. L. Mangasarian, G. L. Meyer, and S. M. Robinson, editors, *Nonlinear Programming, Computational Methods in Applied Sciences vol. 4*, pages 245–282. Academic Press, New York, 1981.

[23] L. Lukšan. Dual method for solving a special problem of quadratic programming as a subproblem at linearly constrained nonlinear minimax approximation. *Kybernetika*, 20:445–457, 1984.

[24] L. Lukšan and J. Vlček. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report 798, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, 2000.

[25] M. M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17(1):1–29, 2002.

[26] M. M. Mäkelä. Multiobjective proximal bundle method for nonconvex nonsmooth optimization: Fortran subroutine MPBNGC 2.0. Technical Report B 13/2003, Reports of the Department of Mathematical Information Technology, Series B, Scientific computing, University of Jyväskylä, Jyväskylä, 2003.

[27] M. M. Mäkelä, V.-P. Eronen, and N. Karmitsa. On nonsmooth multiobjective optimality conditions with generalized convexities. In Th. M. Rassias, C. A. Floudas, and S. Butenko, editors, *Optimization in Science and Engineering*, pages 333–357. Springer, 2014.

[28] M. M. Mäkelä, N. Karmitsa, and O. Wilppu. Proximal bundle method for nonsmooth and nonconvex multiobjective optimization. In T. Tuovinen, S. Repin, and P. Neittaanmäki, editors, *Mathematical Modeling and Optimization of Complex Structures*, volume 40 of *Computational Methods in Applied Sciences*, pages 191–204. Springer, 2016.

[29] M. M. Mäkelä and P. Neittaanmäki. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific Publishing Co., Singapore, 1992.

[30] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.

[31] K. Miettinen and M. M. Mäkelä. Interactive bundle-based method for nondifferentiable multiobjective optimization: NIMBUS. *Optimization*, 34:231–246, 1995.

[32] R. Mifflin. An algorithm for constrained optimization with semismooth functions. *Mathematics of Operations Research*, 2(2):191–207, 1977.

[33] E. S. Mistakidis and G. E. Stavroulakis. *Nonconvex Optimization in Mechanics. Smooth and Nonsmooth Algorithms, Heuristics and Engineering Applications by the F.E.M.* Kluwer Academic Publisher, Dordrecht, 1998.

[34] J. J. Moreau, P. D. Panagiotopoulos, and G. Strang (Eds.). *Topics in Nonsmooth Mechanics*. Birkhäuser, Basel, 1988.

[35] H. Mukai. Algorithms for Multicriterion Optimization. *IEEE Transactions on Automatic Control*, ac-25(2):177–186, 1979.

[36] J. Outrata, M. Koĉvara, and J. Zowe. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints. Theory, Applications and Numerical Results*. Kluwer Academic Publishers, Dordrecht, 1998.

[37] Ž. Povalej. Quasi-Newton's method for multiobjective optimization. *Journal of Computational and Applied Mathematics*, 255:765 – 777, 2014.

[38] S. Qu, C Liu, M. Goh, Y. Li, and Y. Ji. Nonsmooth multiobjective programming with quasi-Newton methods. *European Journal of Operational Research*, 235(3):503 – 510, 2014.

[39] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[40] H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2(1):121–152, 1992.

[41] N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag, Berlin, 1985.

[42] J. Vlček and L. Lukšan. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications*, 111(2):407–430, 2001.

[43] S. Wang. Algorithms for multiobjective and nonsmooth optimization. In P. Kleinschmidt, F.J. Radermacher, W. Sweitzer, and H. Wildermann, editors, *Methods of Operations Research, 58*, pages 131–142. Athenaum Verlag, 1989.

Appendix A. Multiobjective proximal bundle method

We recall the multiobjective proximal bundle method (MPB) [28, 31] which is a generalization of the single-objective proximal bundle method [20, 29]. It combines the ideas of the proximal bundle and the multiobjective linearization technique presented in [18, 43].

In MSGDB, the problem (1) was approached by calculating descent directions for every objective function separately by utilizing the bundle idea, and then by combining this information, a common descent direction was concluded. In MPB, the bundle idea is also used but in a different way. A common descent search direction for all the objectives is formed straight with a different linearization technique. This linearization technique is based on [18, 43].

In MPB, the improvement function $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined in (2) is utilized. According to [19, 28, 43] the problem (1) attains a weakly Pareto optimal solution

at the point $\boldsymbol{x}^*$ if and only if

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} H(\boldsymbol{x}, \boldsymbol{x}^*). \tag{18}$$

Thus, at the k-th iteration we are looking for a direction $\boldsymbol{d}_k$ being a solution of the problem

$$\begin{aligned} \min \quad & H(\boldsymbol{x}_k + \boldsymbol{d}, \boldsymbol{x}_k) \\ \text{s. t.} \quad & \boldsymbol{d} \in \mathbb{R}^n. \end{aligned} \tag{19}$$

The problem (19) can be approximated by defining a convex piecewise linear approximation for the improvement function (2). This approximation can be defined by

$$\hat{H}_k(\boldsymbol{x}) = \max_{i=1,\ldots,m} \left\{ \hat{f}_i^k(\boldsymbol{x}) - f_i(\boldsymbol{x}_k) \right\},$$

where the function $\hat{f}_i$ is the same cutting plane model than in (5). Hence, an approximation for the problem (19) is obtained, and a search direction can be calculated by solving the problem

$$\boldsymbol{d}_k = \underset{\boldsymbol{d}\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \hat{H}_k(\boldsymbol{x}_k + \boldsymbol{d}) + \frac{1}{2} u_k \left\| \boldsymbol{d} \right\|^2 \right\}, \tag{20}$$

where $u_k > 0$ is a weighting parameter as in (7). This nonsmooth problem can also be written as a smooth quadratic problem like (8) for MSGDB in the form

$$\begin{aligned} \min \quad & v + \frac{1}{2} u_k \left\| \boldsymbol{d} \right\|^2 \\ \text{s. t.} \quad & \boldsymbol{\xi}_{i,j}^T \boldsymbol{d} - \alpha_{i,j}^k \leq v, \quad \text{for all } i = 1, \ldots, m, \text{ for all } j \in J_k \\ & \boldsymbol{d} \in \mathbb{R}^n, v \in \mathbb{R}, \end{aligned} \tag{21}$$

where the linearization error $\alpha_{i,j}^k$ is defined as in (6). The difference between problems (8) and (21) is that in (8) there exist constraints only for the current value of i, while in (21) there exist constraints for every index i. Likewise the problem (8), the problem (21) can also be dualized to make it easier to solve

$$\begin{aligned} \min \quad & \frac{1}{2u_k} \Big\| \sum_{i=1}^{m} \sum_{j\in J_k} \lambda_{i,j} \boldsymbol{\xi}_{i,j} \Big\|^2 + \sum_{i=1}^{m} \sum_{j\in J_k} \lambda_{i,j} \alpha_{i,j}^k \\ \text{s. t.} \quad & \sum_{j\in J_k} \lambda_{i,j} = 1 \\ & \lambda_{i,j} \geq 0, \text{ for all } i = 1, \ldots, m, \; j \in J_k. \end{aligned} \tag{22}$$

In MBP, the *two-point line search* strategy is utilized to calculate a stepsize. The aim of the two-point line search strategy is to find a stepsize $0 < t_k \leq 1$ such

Algorithm 2 Multiobjective proximal bundle method (MPB)

Step 1: (*Initialization*) Select the starting point $\boldsymbol{x}_1$, the final accuracy tolerance $\varepsilon > 0$, the weight $u_1 > 0$ and line search parameters. Set an iteration index $k = 1$.

Step 2: (*Direction finding*) Calculate the search direction $\boldsymbol{d}_k$ from the problem (20).

Step 3: (*Stopping criterion*) Stop, if stopping criteria $-\frac{1}{2}v_k < \varepsilon$ is met.

Step 4: (*Line search*) Calculate a stepsize t_k by using the two-point line search strategy and calculate the new point $\boldsymbol{x}_{k+1}$ and the trial point $\boldsymbol{y}_{k+1}$.

Step 5: (*Update*) Add more information to the bundle by evaluating $\boldsymbol{\xi}_{i,k+1} \in f_i(\boldsymbol{y}_{i,k+1})$ for $i = 1, \ldots, m$ and adding those elements to the set J_k with suitable indices to improve the approximation. Update the weight parameter u_{k+1}. Go to Step 2.

---

that the value of $H(\boldsymbol{x}_k + t_k\boldsymbol{d}_k, \boldsymbol{x}_k)$ is minimal when $\boldsymbol{x}_k + t_k\boldsymbol{d}_k \in \mathbb{R}^n$. This stepsize is produced by the line search algorithm in [29] (pp. 126–130).

The general description of the algorithm of MPB is given below. Like in MSGDB, the weight updating algorithm presented in [20] is used.

The solutions of MPB are weakly Pareto optimal as we see in the next theorem.

Theorem 5.1. [28] *Let us consider the problem* (1). *If MPB stops with a finite number of iterations, then the solution is weakly Pareto optimal. On the other hand, any accumulation point of the infinite sequence of solutions generated by MPB is weakly Pareto optimal.*

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

ISBN 978-952-12-3154-4
ISSN 1239-1891