VILNIUS UNIVERSITY

NERIJUS GALIAUSKAS

OPTIMIZATION ALGORITHMS FOR MULTIDIMENSIONAL SCALING WITH CITY-BLOCK DISTANCES AND THEIR PARALLELIZATION

Doctoral Dissertation
Technological Sciences, Informatics Engineering (07 T)

Vilnius, 2015

The dissertation was prepared during the period 2010–2014 at the Institute of Mathematics and Informatics of Vilnius University

Scientific supervisor:

Prof. Dr. Julius Žilinskas (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

VILNIAUS UNIVERSITETAS

NERIJUS GALIAUSKAS

OPTIMIZAVIMO ALGORITMAI DAUGIAMATĖMS SKALĖMS SU MIESTO KVARTALO ATSTUMAIS IR JŲ LYGIAGRETINIMAS

Daktaro disertacija
Technologijos mokslai, informatikos inžinerija (07 T)

Vilnius, 2015

Disertacija rengta 2010-2014 metais Vilniaus universiteto Matematikos ir informatikos institute

Mokslinis vadovas:
prof. dr. Julius Žilinskas (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Abstract

In this dissertation, a problem related to a visualization of elements of a multidimensional data set is considered. Here, for the sake of simplicity, an element of a multidimensional data set is called a multidimensional element. There are many techniques for visualizing multidimensional elements. Multidimensional scaling (MDS) is one of them. In applying MDS, a certain real function has to be constructed and minimized. In order to construct the function, a desired distance function has to be selected. If city-block distances are selected, the problem of minimizing such a function becomes a complicated optimization problem. In this work, this complicated optimization problem is the main research problem. Here, we formulate a new optimization problem and we show that it is equivalent to the original optimization problem arising in MDS with city-block distances. Also, we propose two sequential algorithms and one parallel algorithm for the newly formulated problem. Finally, we present the results of numerical investigations of the proposed algorithms.

Acknowledgments

While writing this dissertation I met a lot of wonderful people. I would like to give a big thank you to all of them separately. However, I will mention only a few of them here. Others will remain in my mind and heart forever.

First of all, I would like to thank my parents for *everything*.

I would like to express the deepest appreciation to my scientific supervisor Prof. Dr. Julius Žilinskas for his invaluable support, understanding, and patience.

I would also like to extend my appreciation to Prof. Dr. Roger Fletcher for his great ideas and suggestions for solving the research problem.

In addition, I would also like to thank Prof. Dr. Jonas Mockus and Dr. Algirdas Lančinskas for their deep review of the dissertation and their valuable advice on how to improve the quality of this work.

Last but not least, I would like to thank Prof. Dr. Gintautas Dzemyda, Prof. Dr. Antanas Žilinskas, Prof. Dr. Albertas Čaplinskas, Dr. Olga Kurasova, Danutė Rimeisienė, Prof. Dr. Valentina Dagienė, Prof. Dr. Eugenijus Stankus and Dr. Antanas Apynis for their direct and indirect help in writing this dissertation.

Contents

List of figures and tables

Notations

$c_{\mathbb{S}} = (c_{i_1} \ \ldots \ c_{i_{|\mathbb{S}|}}) \in \mathbb{R}^{q \times |\mathbb{S}|}$ – a matrix of vectors $c_i \in \mathbb{R}^q$, $i \in \mathbb{S} \subset \mathbb{N}$

$d_{\mathbb{S}} = (d_{i_1}, \ldots, d_{i_{|\mathbb{S}|}})^T \in \mathbb{R}^{|\mathbb{S}|}$ – a vector of elements $d_i \in \mathbb{R}$, $i \in \mathbb{S} \subset \mathbb{N}$

$\mathbb{U}_f^* = \operatorname{argmin}\{f(u) : u \in \mathbb{U}\}$ – a set of all minimizers of the objective function f on the feasible set $\mathbb{U}$

$n \in \mathbb{N}$ $(n > 1)$ – the number of given multidimensional elements

$m \in \{1, 2, 3\}$ – the dimension of the Cartesian coordinate system in which we desire to visualize the given multidimensional elements

$s = mn(n-1)/2$

$N = s/m = n(n-1)/2$

δ_{ij} – a dissimilarity between multidimensional elements i and j

$x = (x_1^T, x_2^T, \ldots, x_n^T)^T = (x_{11}, \ldots, x_{m1}, x_{12}, \ldots, x_{m2}, \ldots, x_{mn})^T \in \mathbb{R}^{mn}$ – an image of n multidimensional elements in the m-dimensional Cartesian coordinate system

$d^{\pm} = (d^+_{112}, d^-_{112}, \ldots, d^+_{m12}, d^-_{m12}, \ldots, d^+_{m(n-1)n}, d^-_{m(n-1)n})^T \in \mathbb{R}^{mn(n-1)}$

$y = (x^T, d^{\pm T})^T = (y_1, \ldots, y_{mn}, y_{mn+1}, \ldots, y_{mn^2})^T \in \mathbb{R}^{mn^2}$

$f_1(x) = \sum_{i<j}^{n} \left(\sum_{k=1}^{m} |x_{ki} - x_{kj}| - \delta_{ij}\right)^2$ – the raw Stress function with city-block distances, $x \in \mathbb{R}^{mn}$

$g(y) = \sum_{i<j}^{n} \left(\sum_{k=1}^{m} (d^+_{kij} + d^-_{kij}) - \delta_{ij}\right)^2$ – the objective function of the problem CMDS, $y = (x^T, d^{\pm T})^T \in \mathbb{R}^{mn^2}$

$\mathbb{Y}' = \{y = (x^T, d^{\pm T})^T \in \mathbb{R}^{mn^2} : \sum_{i=1}^{n} x_{ki} = 0,\ x_{ki} - x_{kj} = d^+_{kij} - d^-_{kij},\ d^+_{kij} d^-_{kij} = 0,\ d^+_{kij}, d^-_{kij} \geq 0,\ 1 \leq i < j \leq n, 1 \leq k \leq m\}$ – the feasible set of the problem CMDS

$\mathbb{Z} = \{z \in \{0,1\}^{2s} : z_{2i-1} + z_{2i} = 1,\ 1 \leq i \leq s\}$ – the upper-level feasible set of the problem CMDS, formulated as a two-level optimization problem

$\mu(k, i, j) = 2(k + m(j - i + (2n - i)(i-1)/2 - 1)) - 1$, k, i, $j \in \mathbb{N}$

$\mathbb{Y}'(z) = \{(x^T, d^{\pm T})^T \in \mathbb{R}^{mn^2} : \sum_{i=1}^{n} x_{ki} = 0,\ x_{ki} - x_{kj} = d^+_{kij} - d^-_{kij},\ (1 - z_{\mu(k,i,j)}) d^+_{kij} = 0,\ (1 - z_{\mu(k,i,j)+1}) d^-_{kij} = 0,\ d^+_{kij}, d^-_{kij} \geq 0,\ 1 \leq k \leq m,\ 1 \leq i < j \leq n\}$ – the lower-level feasible set of the problem CMDS, formulated as a two-level optimization problem, $z \in \mathbb{Z}$

Introduction

In this chapter, the research problem and its relevance are presented. The purpose, tasks, and novelty of the research are given here, too. Also, we present here propositions for defense. The results of the research were approved in a few publications and scientific conferences. A list of them is given here, too. The author of the dissertation took part in two scientific projects. The projects were closely related to the research described here. Thus, a list of the projects is given here, too.

The research problem and its relevance

The research problem is related to the problem of a visualization of multidimensional elements. A multidimensional element is perceived here as a vector in a real vector space of four dimensions or higher. A representation of such vectors in a lower-dimensional – i.e., one-, two-, or three-dimensional – Cartesian coordinate system is perceived here as a visualization of multidimensional elements. The set of points, which represent the multidimensional elements in the lower-dimensional Cartesian coordinate system, is called here the image of multidimensional elements. Note that an image of multidimensional elements is a subset of a lower-dimensional real vector space. Any image of multidimensional elements may give us some important information about them. For example, it may help us to find out if the elements have any regularities, outliers, or clusters [**26**].

There are a number of techniques that can be used to visualize multidimensional elements. One of them is called multidimensional scaling (MDS). MDS is used in areas such as psychometrics, market analysis, pharmacology, and others. In applying MDS, an image of multidimensional elements is obtained by constructing and minimizing a certain function. In order to construct the function, 1) dissimilarities between every pair of multidimensional elements have to be defined, and 2) a desired distance function, defined on a lower-dimensional real vector space, has to be selected. The obtained image has the following feature: the desired distances between points of the image are similar to, or even equal to, the defined dissimilarities.

Images of the same multidimensional elements are usually different if they are obtained by using MDS with different distance functions. Various applications of MDS show that images obtained by using MDS with the city-block distance function (also known as Minkowski distance of order one, L1 distance function) may give us more information about the multidimensional elements than by using other distance functions. However, if city-block distances are selected, the function to be minimized contains a set of absolute value (modulus) terms. In this case, minimization of such a function becomes a complicated optimization problem: the function may have more than one local minimizer and may even be non-differentiable at a minimizer. Thus, the research problem considered here is the minimization problem arising in MDS with city-block distances.

The purpose and tasks of the research

The research problem – i.e., the minimization problem arising in MDS with city-block distances – is equivalent to a certain optimization problem. Let us name the optimization problem, equivalent to the research problem, as problem CMDS. Currently there are no specific algorithms for the problem CMDS. Thus, the main purpose of this research is to offer several algorithms for the problem CMDS. In order to achieve this goal, the following tasks should be performed:

(1) To explore the main features of the objective function of the research problem and to become familiar with algorithms for the research problem.
(2) To construct a few sequential and parallel algorithms for the problem CMDS.
(3) To implement the algorithms and to perform a numerical investigation of them by using particular sets of multidimensional elements.
(4) To compare the results of the investigation with corresponding results obtained by using other algorithms.

The novelty of the research

(1) The equivalence between the research problem and the problem CMDS is proven.
(2) An algorithm is proposed that returns a local minimizer of the objective function of the problem CMDS.

(3) A two-level formulation of the problem CMDS is constructed.
(4) An algorithm is proposed that returns a global minimizer of the objective function of the problem CMDS.
(5) A parallel algorithm is proposed that returns a global minimizer of the objective function of the problem CMDS.

Propositions for defense

(1) The research problem is equivalent to an optimization problem with a convex quadratic objective function, and linear and complementarity (non-linear) constraints.
(2) The algorithm, based on the active-set method, returns a local minimizer of the objective function of the problem CMDS.
(3) The problem CMDS is a two-level optimization problem with a convex quadratic programming problem at the lower-level, and a combinatorial optimization problem at the upper-level.
(4) The algorithm, based on the branch-and-bound method, returns a global minimizer of the objective function of the problem CMDS.

Approbation of results

Results of the research were published in the following scientific papers: [**GŽ11, GŽ13, GŽ14, FGŽ14**]. Some of the results were presented at the following scientific conferences (the title of the presentation is written in italic letters):

(1) 4th Conference of Lithuanian Young Scientists on Operations Research in Business, Engineering and Computer Science. Kaunas, Lithuania. September 30, 2011.
Parallel algorithms for quadratic programming.
(2) Data Analysis Methods for Software Systems. Druskininkai, Lithuania. December 1–3, 2011.
Quadratic programming in data analysis.
(3) 2nd Conference of Young Scientists on Interdisciplinary Research in Physical and Technological Sciences. Vilnius, Lithuania. February 14, 2012.
Quadratic programming problems.
(4) 10th EUROPT Workshop on Advances in Continuous Optimization. Šiauliai, Lithuania. July 5–7, 2012.

Minimization of a stress function with city-block distances.

(5) 18th International Conference on Mathematical Modeling and Analysis. Tartu, Estonia. May 27–30, 2013.
Quadratic programming with complementarity constraints for multidimensional scaling with city-block distances.

(6) 5th Conference of Lithuanian Young Scientists on Operations Research in Business, Engineering and Computer Science. Šiauliai, Lithuania. September 19, 2013.
Parallelization of the active-set method for convex quadratic programming.

(7) 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Compiègne, France. October 28–30, 2013.
Parallel branch-and-bound for multidimensional scaling with L1 distances formulated as quadratic programming with complementarity constraints.

(8) 55th Conference of Lithuanian Mathematical Society. Vilnius, Lithuania. June 26–27, 2014.
Quadratic programming for multidimensional scaling with L1 distances.

(9) 8th International Workshop on Parallel Matrix Algorithms and Applications. Lugano, Switzerland. July 2–4, 2014.
Parallel solution of an active-set algorithm for convex quadratic programming.

Scientific projects

The author of the dissertation took part in the following scientific projects:

(1) "Nonconvex multiobjective optimization: methods and algorithms". Funded by Research Council of Lithuania. (No. MIP-063/2012/LSS-580000-444). 2012–2014.

(2) "Theoretical and engineering aspects of e-service technology development and application in high performance computing platforms". Funded by the European Social Fund. (No. VP1-3.1-ŠMM-08-K-01-010). 2012–2014.

1. Basic concepts and related works

In this chapter, we introduce the basic concepts of mathematical optimization and describe several optimization methods. Also, we present here a detailed description of the research problem, i.e., the minimization problem arising in multidimensional scaling with city-block distances. A few known algorithms for the research problem are given here, too. Certain parts of this chapter are published in [**GŽ11**].

1.1. Mathematical optimization

The research problem considered in this dissertation is equivalent to a certain optimization problem, here called the problem CMDS. In order to show the equivalence of these two problems and to reveal the main features of the problem CMDS, we have to know the basic concepts of mathematical optimization. In addition, in this dissertation we propose several algorithms for the problem CMDS. The algorithms are based on several well-known optimization methods. Thus, we have to be familiar with these methods, too. In this section, we introduce the basic concepts of mathematical optimization and describe several well-known optimization methods.

1.1.1. Concepts

Here, we introduce the following concepts of mathematical optimization: optimization problem, equivalence of two optimization problems, two-level optimization problem, convexity, KKT conditions, quadratic programming (QP) problem, and equality constrained QP problem.

1.1.1.1. Optimization problem. Let us introduce the mathematical formulation of an optimization problem [**15, 32, 64, 71, 72, 79**]. Suppose that $q \in \mathbb{N}$, r_e, $r \in \mathbb{N} \cup \{0\}$ ($r_e \leq r$) and $\mathbb{E} = \{1, \ldots, r_e\}$, $\mathbb{I} = \{r_e + 1, \ldots, r\}$. Let c_i, $i \in \mathbb{E} \cup \mathbb{I}$, be real-valued functions, defined on space $\mathbb{R}^q$. Suppose that $\mathbb{U} \subseteq \mathbb{R}^q$ ($\mathbb{U} \neq \emptyset$), where

$$\begin{aligned} \mathbb{U} = \{u \in \mathbb{R}^q : \;& c_i(u) = 0, \; i \in \mathbb{E}, \\ & c_i(u) \geq 0, \; i \in \mathbb{I}\}, \end{aligned} \tag{1.1}$$

and $f : \mathbb{U} \to \mathbb{R}$. Let $u^* \in \mathbb{U}$ be a point such that

$$f(u^*) \leq f(u) \tag{1.2}$$

for all $u \in \mathbb{U}$. The problem of finding a point $u^* \in \mathbb{U}$ that satisfies inequality (1.2) is called an optimization problem (sometimes optimization program, or mathematical programming problem). The branch of mathematics that is concerned with optimization problems is called mathematical optimization (sometimes mathematical programming). In general, such a point u^* that satisfies inequality (1.2) may not exist at the set $\mathbb{U}$. The existence depends on features of the function f and the set $\mathbb{U}$. Here, we accept that such a point always exists.

The set $\mathbb{U}$, defined by (1.1), is usually called a feasible set (sometimes search space) and the function f, an objective (sometimes cost) function. The functions c_i, $i \in \mathbb{E} \cup \mathbb{I}$, are commonly referred to as constraints. Constraints c_i, $i \in \mathbb{E}$, are called equality constraints and c_i, $i \in \mathbb{I}$, inequality constraints. A set of indices of equality and inequality constraints that are active at $u \in \mathbb{U}$ is called an active set at the point u, and is usually defined by the symbol $\mathbb{A}(u)$, i.e.,

$$\mathbb{A}(u) = \{i \in \mathbb{E} \cup \mathbb{I} : c_i(u) = 0\}.$$

Equality constraints c_i, $i \in \mathbb{E}$, such that $c_i(u) = c_i'(u)c_i''(u)$, where c_i', $c_i'' : \mathbb{R}^q \to \mathbb{R}$, $i \in \mathbb{E}$, are called complementarity constraints. Let $\mathbb{S} \subseteq \mathbb{E} \cup \mathbb{I}$. Constraints c_i, $i \in \mathbb{S}$, such that $c_i(u) = c_i^T u - d_i$, where $c_i \in \mathbb{R}^q$, $d_i \in \mathbb{R}$, $i \in \mathbb{S}$, are called linear constraints. If $c_i \in \mathbb{R}^q$, $i \in \mathbb{S}$, are gradients of the linear constraints, matrix of these gradients $(c_{i_1} \ \ldots \ c_{i_{|\mathbb{S}|}}) \in \mathbb{R}^{q \times |\mathbb{S}|}$, $i_j \in \mathbb{S}$, $1 \leq j \leq |\mathbb{S}|$, we sometimes denote by the symbol $c_{\mathbb{S}}$. Similarly, vector $(d_{i_1}, \ldots, d_{i_{|\mathbb{S}|}})^T \in \mathbb{R}^q$, $i_j \in \mathbb{S}$, $1 \leq j \leq |\mathbb{S}|$, we sometimes denote by the symbol $d_{\mathbb{S}}$.

A point $u^* \in \mathbb{U}$ that satisfies inequality (1.2) for all $u \in \mathbb{U}$ is called a global minimizer of the function f on the set $\mathbb{U}$. In the text, instead of "a global minimizer" we usually write simply "a minimizer". A set of all minimizers of f on $\mathbb{U}$ we denote by the symbol $\mathbb{U}_f^*$ or $\operatorname{argmin}\{f(u) : u \in \mathbb{U}\}$. If $u^* \in \mathbb{U}_f^*$, then the objective value $f(u^*)$ is often called the minimum value of the function f on the feasible set $\mathbb{U}$.

Let $u' \in \mathbb{U}$ and $\mathbb{U}_\epsilon(u') = \{u \in \mathbb{U} : d(u, u') < \epsilon\}$, where $\epsilon > 0$ and $d : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ is a metric on $\mathbb{U}$. We say that $u^* \in \mathbb{U}$ is a local minimizer

of f on $\mathbb{U}$, if there is a point $u' \in \mathbb{U}$ and a number $\epsilon > 0$ such that u^* is a global minimizer of f on $\mathbb{U}_\epsilon(u')$, i.e., $u^* \in \operatorname{argmin}\{f(u) : u \in \mathbb{U}_\epsilon(u')\}$.

The problem of finding a minimizer of f on $\mathbb{U}$ is usually denoted as follows:

$$\begin{array}{ll} \underset{u \in \mathbb{R}^q}{\text{minimize}} & f(u) \\ \text{subject to} & c_i(u) = 0,\ i \in \mathbb{E}, \\ & c_i(u) \geq 0,\ i \in \mathbb{I}. \end{array} \tag{1.3}$$

According to the features of the objective function f and the feasible set $\mathbb{U}$, problem (1.3) may belong to one or another class of optimization problems. For example, if the set $\mathbb{U}$ is a finite set or countably infinite – i.e., there exists a bijective function $h : \mathbb{U} \to \mathbb{N}$ – then such a problem is often called a combinatorial optimization problem [**89**]. If the functions f and c_i, $i \in \mathbb{E} \cup \mathbb{I}$, are linear functions, problem (1.3) is referred to as a linear programming problem [**81**], etc.

1.1.1.2. Equivalence of two optimization problems. Let us introduce the concept of the equivalence of two optimization problems [**10, 52**]. Suppose that q', $q'' \in \mathbb{N}$ and $\mathbb{U}' \subseteq \mathbb{R}^{q'}$, $\mathbb{U}'' \subseteq \mathbb{R}^{q''}$ ($\mathbb{U}' \neq \emptyset$, $\mathbb{U}'' \neq \emptyset$). Let $f' : \mathbb{U}' \to \mathbb{R}$ and $f'' : \mathbb{U}'' \to \mathbb{R}$. We say that two optimization problems

$$\underset{u' \in \mathbb{U}'}{\text{minimize}} \quad f'(u') \quad \text{and} \quad \underset{u'' \in \mathbb{U}''}{\text{minimize}} \quad f''(u'')$$

are equivalent problems, if and only if, there exists a bijective function $h : \mathbb{U}' \to \mathbb{U}''$ such that

$$u'^* \in \mathbb{U}'^*_{f'} \text{ if and only if } h(u'^*) \in \mathbb{U}''^*_{f''}.$$

A few examples of equivalent optimization problems may be found in [**10, 22, 29, 80**].

1.1.1.3. Two-level optimization problem. Let us introduce the mathematical formulation of a two-level optimization problem [**17, 21, 24, 84**]. Suppose that q', $q'' \in \mathbb{N}$ and $\mathbb{U}' \subseteq \mathbb{R}^{q'}$, $\mathbb{U}'' \subseteq \mathbb{R}^{q''}$ ($\mathbb{U}' \neq \emptyset$, $\mathbb{U}'' \neq \emptyset$). Let r', $r'' \in \mathbb{N} \cup \{0\}$ and f', c'_i, f'', $c''_j : \mathbb{U}' \times \mathbb{U}'' \to \mathbb{R}$, $1 \leq i \leq r'$, $1 \leq j \leq r''$. The optimization problem

$$\begin{array}{ll} \underset{u' \in \mathbb{U},\ u''^* \in \mathbb{U}''^*_{f''}(u')}{\text{minimize}} & f'(u', u''^*) \\ \text{subject to} & c'_i(u', u''^*) \geq 0,\ 1 \leq i \leq r', \end{array} \tag{1.4}$$

where the set $\mathbb{U}''^{*}_{f''}(u') \subseteq \mathbb{U}''$ denotes a set of all global minimizers of the following problem

$$\begin{array}{ll} \underset{u'' \in \mathbb{U}''}{\text{minimize}} & f''(u', u'') \\ \text{subject to} & c''_i(u', u'') \geq 0,\ 1 \leq i \leq r'', \end{array} \tag{1.5}$$

is called a two-level (sometimes bilevel) optimization problem. Optimization problems (1.4) and (1.5) are called upper- and lower-level problems, respectively. In turn, variables u' and u'', objective functions f' and f'', and inequality constraints c'_i, $1 \leq i \leq r'$, and c''_j, $1 \leq j \leq r''$, are called upper- and lower-level variables, objective functions, and inequality constraints, respectively. It is clear that a set of equality constraints may be included in two-level optimization problems. However, for the sake of brevity, we have omitted them here.

1.1.1.4. Convexity. Let us define concepts of a convex set and a convex function **[9, 47, 69]**. Suppose that $q \in \mathbb{N}$ and $\mathbb{U} \subseteq \mathbb{R}^q$ ($\mathbb{U} \neq \emptyset$). The set $\mathbb{U}$ is called a convex set, if

$$\alpha u' + (1 - \alpha)u'' \in \mathbb{U}$$

for all u', $u'' \in \mathbb{U}$ and $\alpha \in [0, 1]$. For example, it is not hard to check that a feasible set, defined by linear constraints, is a convex set. Let $\mathbb{U}$ be a convex set and f be a real-valued function, defined on the set $\mathbb{U}$. The function f is called a non-strictly convex (sometimes just convex) function, if

$$f(\alpha u' + (1 - \alpha)u'') \leq \alpha f(u') + (1 - \alpha) f(u'')$$

for all u', $u'' \in \mathbb{U}$ and $\alpha \in [0, 1]$. A non-strictly convex function may have a lot of minimizers at its convex domain. The function f is called a strictly convex function, if

$$f(\alpha u' + (1 - \alpha)u'') < \alpha f(u') + (1 - \alpha) f(u'')$$

for all u', $u'' \in \mathbb{U}$ ($u' \neq u''$) and $\alpha \in (0, 1)$. Every minimizer of a strictly convex function on a convex set is unique. In addition, if a feasible point is a local minimizer of a convex function on a convex set, then that point is a global minimizer.

1.1.1.5. KKT conditions. Let us consider optimization problem (1.3). There is a set of necessary conditions for a feasible point to be a local minimizer of the objective function f on the feasible set $\mathbb{U}$. These conditions are sometimes referred to as the Karush-Kuhn-Tucker (KKT), or the first-order necessary conditions [**3, 10**]. Let $L : \mathbb{R}^q \times \mathbb{R}^r \to \mathbb{R}$ be a function such that

$$L(u, \lambda) = f(u) - \sum_{i \in \mathbb{E} \cup \mathbb{I}} \lambda_i c_i(u). \tag{1.6}$$

Function (1.6) is often called the Lagrangian function for problem (1.3) and $\lambda = (\lambda_1, \ldots, \lambda_r)^T$, a Lagrange (or Lagrangian) multiplier vector. If 1) $u^* \in \mathbb{U}$ is a local minimizer of f on $\mathbb{U}$, 2) functions f, c_i, $i \in \mathbb{E} \cup \mathbb{I}$, are continuously differentiable at the point u^*, and 3) vectors $\nabla c_i(u^*)$, $i \in \mathbb{A}(u^*)$, are linearly independent, then there is a Lagrange multiplier vector $\lambda^* \in \mathbb{R}^r$, such that:

$$\nabla_u L(u^*, \lambda^*) = 0, \tag{1.7a}$$

$$c_i(u^*) = 0, \ i \in \mathbb{E}, \tag{1.7b}$$

$$c_i(u^*) \geq 0, \ i \in \mathbb{I}, \tag{1.7c}$$

$$\lambda_i^* \geq 0, \ i \in \mathbb{I}, \tag{1.7d}$$

$$\lambda_i^* c_i(u^*) = 0, \ i \in \mathbb{E} \cup \mathbb{I}. \tag{1.7e}$$

Conditions (1.7a)–(1.7e) are called the KKT conditions for $u^* \in \mathbb{U}$ to be a local minimizer of the function f on the set $\mathbb{U}$. For the sake of simplicity, the conditions are sometimes referred to as the KKT conditions for problem (1.3). Separately, conditions (1.7a), (1.7b)–(1.7c), (1.7d), and (1.7e) are called stationary, primal feasibility, dual feasibility, and complementarity conditions, respectively. If the objective function f is convex, c_i, $i \in \mathbb{E}$, are affine functions, i.e., in our case, $c_i(u) = c_i^T u - d_i$, $c_i \in \mathbb{R}^q$, $d_i \in \mathbb{R}$, $i \in \mathbb{E}$, and c_i, $i \in \mathbb{I}$, are continuously differentiable convex functions, then the KKT conditions become sufficient conditions for $u^* \in \mathbb{U}$ to be a local (in this case global) minimizer of f on $\mathbb{U}$ [**10**].

1.1.1.6. QP problem. Let us define a quadratic programming (QP) problem [**25, 65, 68, 71**]. Suppose that $q \in \mathbb{N}$, r_e, $r \in \mathbb{N} \cup \{0\}$ ($r_e \leq q$, $r_e \leq r$) and $c_i \in \mathbb{R}^q$, $d_i \in \mathbb{R}$, $1 \leq i \leq r$. Let $\mathbb{E} = \{1, \ldots, r_e\}$,

$\mathbb{I} = \{r_e + 1, \ldots, r\}$ and $\mathbb{U} \subseteq \mathbb{R}^q$ be a set such that

$$\mathbb{U} = \{u \in \mathbb{R}^q : \; c_i^T u = d_i, \; i \in \mathbb{E}, \\ c_i^T u \geq d_i, \; i \in \mathbb{I}\}.$$

Suppose that vectors c_i, $i \in \mathbb{E}$, are linearly independent, i.e., $\text{rank}(c_{\mathbb{E}}) = |\mathbb{E}|$. Also, suppose that $A \in \mathbb{R}^{q \times q}$ ($A = A^T$, $A \neq 0$) and $b \in \mathbb{R}^q$. Let $f : \mathbb{U} \to \mathbb{R}$ be a function such that

$$f(u) = 0.5u^T Au + b^T u.$$

The function f is called a quadratic function. Optimization problem

$$\underset{u \in \mathbb{U}}{\text{minimize}} \quad f(u) \tag{1.8}$$

is called a QP problem. The quadratic function f may be convex or non-convex. The convexity depends on the matrix A, which is usually called the Hessian matrix [**35, 46**]. If the Hessian matrix is positive semidefinite, then the quadratic function is non-strictly convex. If the matrix A is positive definite, then the function f is strictly convex. If A is neither positive semidefinite nor positive definite, f is non-convex. There are a few characterizations for a matrix to be positive semidefinite or positive definite [**5, 21**]. For example, a symmetric matrix $M \in \mathbb{R}^{q \times q}$ is called 1) positive semidefinite, if there is a matrix $S \in \mathbb{R}^{q \times q}$ such that $M = S^T S$, 2) positive definite, if there is a nonsingular matrix $S \in \mathbb{R}^{q \times q}$, i.e., $\det(S) \neq 0$, such that $M = S^T S$. Positive semidefinite and positive definite matrices are usually denoted by the symbols $M \succeq 0$ and $M \succ 0$, respectively. If the quadratic function f is non-strictly or strictly convex, problem (1.8) is called a convex QP problem.

1.1.1.7. Equality constrained QP problem. Let us consider QP problem (1.8), where $r_e = r$, i.e., $\mathbb{I} = \emptyset$. Optimization problem

$$\begin{array}{ll} \underset{u \in \mathbb{R}^q}{\text{minimize}} & f(u) \\ \text{subject to} & c_i^T u = d_i, \; i \in \mathbb{E} \end{array} \tag{1.9}$$

is called an equality constrained QP problem. Let $u^* \in \mathbb{U}$ be a minimizer of f on $\{u \in \mathbb{R}^q : c_{\mathbb{E}}^T u = d_{\mathbb{E}}\}$. Then, according to KKT conditions (1.7a)–(1.7e) for problem (1.9), there is a Lagrange multiplier vector $\lambda^* \in \mathbb{R}^{|\mathbb{E}|}$,

such that:

$$\begin{pmatrix} A & -c_{\mathbb{E}} \\ c_{\mathbb{E}}^T & 0 \end{pmatrix} \begin{pmatrix} u^* \\ \lambda^* \end{pmatrix} = \begin{pmatrix} -b \\ d_{\mathbb{E}} \end{pmatrix}. \tag{1.10}$$

System (1.10) is called the KKT system, and the coefficient matrix of this system is called the KKT matrix.

Problem (1.9) occurs in a variety of optimization algorithms as a sub-problem, used to find a step direction [**14, 18, 36, 91**]. Let

$$H = \begin{pmatrix} A & -c_{\mathbb{E}} \\ c_{\mathbb{E}}^T & 0 \end{pmatrix}, w = \begin{pmatrix} u \\ \lambda \end{pmatrix} \text{ and } h = \begin{pmatrix} -b \\ d_{\mathbb{E}} \end{pmatrix}.$$

Depending on features of the KKT system $Hw = h$, the quadratic function f may have:

(1) No minimizer on $\mathbb{U}$, if the KKT system is inconsistent, i.e., $\text{rank}(H) \neq \text{rank}((H\ h))$.
(2) A unique minimizer, if the KKT system is consistent and the KKT matrix has full rank, i.e., $\text{rank}(H) = \text{rank}((H\ h)) = q + |\mathbb{E}|$.
(3) Infinitely many minimizers, if the KKT system is consistent and the KKT matrix is rank deficient, i.e., $\text{rank}(H) = \text{rank}((H\ h)) < q + |\mathbb{E}|$.

If the KKT system is inconsistent, the quadratic function f is unbounded below on the feasible set $\mathbb{U}$. In this case, a step direction is usually found by solving the following system of linear equations:

$$Hw = 0. \tag{1.11}$$

One of the ways to solve system (1.11) is to find an eigenvector $w^* = (u^*, \lambda^*)^T \in \mathbb{R}^{q+|\mathbb{E}|}$ of the KKT matrix H that corresponds to a zero eigenvalue. In addition, we require that $b^T u^* < 0$. This inequality ensures the descent direction.

If the KKT system is consistent, the solution of the equality constrained QP problem (1.9) is equivalent to the solution of the following linear least squares problem [**6**]:

$$\underset{w \in \mathbb{R}^{q+|\mathbb{E}|}}{\text{minimize}} \quad \|Hw - h\|_2 .$$

1.1.2. Methods

Here, we describe the following optimization methods: the active-set method for convex QP problems, and the branch-and-bound method. In

addition, here we present the null-space method for consistent KKT systems. Finally, a certain classification of parallel branch-and-bound methods is given here, too.

1.1.2.1. The active-set method. Let us describe the active-set method for QP problem (1.8) [**27, 33, 71**]. Suppose that the quadratic function f is convex. Let $u^* \in \mathbb{U}$ and $\mathbb{A}(u^*) = \{i \in \mathbb{E} \cup \mathbb{I} : c_i^T u^* = d_i\}$ be an active-set at the point u^*. Suppose that c_i, $i \in \mathbb{A}(u^*)$, are linearly independent vectors. It is not hard to check that, if the point u^* satisfies the following conditions:

$$\begin{array}{rl} Au^* + b - \sum_{i \in \mathbb{A}(u^*)} \lambda_i^* c_i & = 0, \\ c_i^T u^* & = d_i,\ i \in \mathbb{A}(u^*), \\ c_i^T u^* & \geq d_i,\ i \in \mathbb{I} \setminus \mathbb{A}(u^*), \\ \lambda_i^* & \geq 0,\ i \in \mathbb{I} \cap \mathbb{A}(u^*) \end{array} \tag{1.12}$$

for certain Lagrange multipliers $\lambda_i^* \in \mathbb{R}$, $i \in \mathbb{A}(u^*)$, then $u^* \in \mathbb{U}_f^*$, i.e., u^* is a minimizer of the quadratic function f on the feasible set $\mathbb{U}$ [**71**]. Therefore, in order to find a minimizer of f on $\mathbb{U}$, it is sufficient to solve the following equality constrained QP problem [**71**]:

$$\begin{array}{ll} \underset{u \in \mathbb{R}^q}{\text{minimize}} & f(u) \\ \text{subject to} & c_i^T u = d_i,\ i \in \mathbb{A}(u^*). \end{array} \tag{1.13}$$

If we would know the structure of the active set $\mathbb{A}(u^*)$ in advance, we would find u^* in one step, i.e., by solving problem (1.13). Unfortunately, we usually do not know anything about the set $\mathbb{A}(u^*)$ in advance. However, the sets $\mathbb{E}$ and $\mathbb{I}$ are finite sets. Hence, the total number of different active sets $\mathbb{A}(u)$, $u \in \mathbb{U}$, is finite, too. Let $\mathbb{A}$ be a set of active sets at points of the set $\mathbb{U}$, such that

$$\mathbb{A} = \{\mathbb{A}(u) : u \in \mathbb{U} \text{ and } \operatorname{rank}(c_{\mathbb{A}(u)}) = |\mathbb{A}(u)|\}.$$

Note that $|\mathbb{A}| \leq 2^{|\mathbb{I}|}$. Therefore, we may find the set $\mathbb{A}(u^*)$ and the corresponding minimizer u^* by using the following procedure: 1) select an active set $\mathbb{W} \in \mathbb{A}$ and remove it from $\mathbb{A}$, 2) find a minimizer of f on $\{u \in \mathbb{R}^q : c_i^T u = d_i,\ i \in \mathbb{W}\}$, say $u_{\mathbb{W}}^* \in \mathbb{R}^q$, and 3) check if $u_{\mathbb{W}}^*$ and the corresponding $\lambda_i^* \in \mathbb{R}$, $i \in \mathbb{W}$, satisfy conditions (1.12). If yes, then $u^* = u_{\mathbb{W}}^* \in \mathbb{U}_f^*$ and $\mathbb{A}(u^*) = \mathbb{W}$; otherwise repeat the first step. The active-set method is very similar to the above procedure. Figure 1.1 shows the

general scheme of the active-set method. Let us explain the method in a word.

First of all, an initial feasible point $u^0 \in \mathbb{U}$ and an initial active set $\mathbb{W}^0 \subseteq \mathbb{A}(u^0)$ are selected. In order to select a feasible point, various techniques may be used: for example, the Big-M method **[71, 87]**. Next, a minimizer of the function $f(u^k+p)$ on the set $\{p \in \mathbb{R}^q : c_i^T(u^k+p) - d_i = c_i^T p = 0,\ i \in \mathbb{W}^k\}$ is found, say $p^* \in \mathbb{R}^q$. The minimizer p^* is often called the step (or search) direction. If the point $u^k + p^* \in \mathbb{R}^q$ and the corresponding Lagrange multipliers $\lambda_i^* \in \mathbb{R}$, $i \in \mathbb{W}^k$, satisfy conditions (1.12), then $u^k + p^* \in \mathbb{U}_f^*$; otherwise $u^k + p^* \notin \mathbb{U}_f^*$. If $u^k + p^* \notin \mathbb{U}_f^*$, then the next feasible point u^{k+1} is constructed such that $f(u^{k+1}) \leq f(u^k)$. Also, the next active set $\mathbb{W}^{k+1}$ is obtained from the set $\mathbb{W}^k$ usually by adding or removing an index of a particular inequality constraint. Sometimes $\mathbb{W}^{k+1} = \mathbb{W}^k$. The process is continued until $u^K + p^* = u^K$ becomes a minimizer of f on $\mathbb{U}$ (here, $K \in \mathbb{N}$ denotes the total number of iterations required to find a minimizer of f on $\mathbb{U}$). How to select the value of $\alpha \in [0,1]$, how to obtain $\mathbb{W}^{k+1}$ from $\mathbb{W}^k$, etc., is described in Section 2.3.

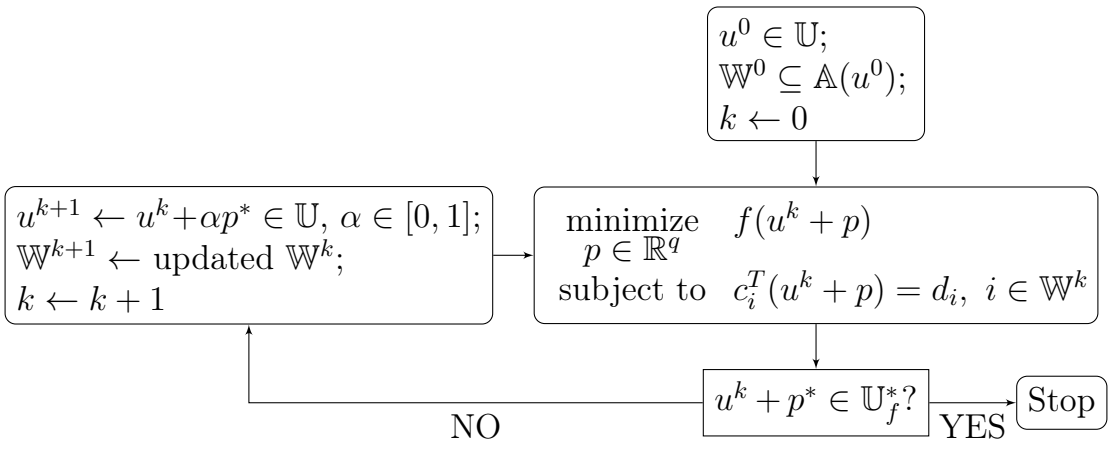


FIGURE 1.1. General scheme of the active-set method.

1.1.2.2. The null-space method. Let us consider equality constrained QP problem (1.9), where the quadratic function f is convex. The solution of such a problem is equivalent to the solution of a consistent KKT system (1.10). Here, we present the null-space method for system (1.10) **[28, 70]**. The method is based on the fact that the solution of a consistent linear system may be expressed as a certain linear combination. Let $(u^*, \lambda^*)^T \in \mathbb{R}^{q+|\mathbb{E}|}$ be a solution of system (1.10). Suppose that $Y \in \mathbb{R}^{q \times |\mathbb{E}|}$ and $Z \in \mathbb{R}^{q \times (q-|\mathbb{E}|)}$ are two matrices such that $c_{\mathbb{E}}^T Z = 0$ and $\det((Y\ Z)) \neq 0$.

Then there are vectors $u_Y^* \in \mathbb{R}^{|\mathbb{E}|}$ and $u_Z^* \in \mathbb{R}^{q-|\mathbb{E}|}$ such that

$$u^* = Yx_Y^* + Zx_Z^*. \tag{1.14}$$

By substituting (1.14) into the system $c_{\mathbb{E}}^T u^* = d_{\mathbb{E}}$, we have that

$$(c_{\mathbb{E}}^T Y)u_Y^* = d_{\mathbb{E}}. \tag{1.15}$$

Note that $\operatorname{rank}(c_{\mathbb{E}}^T) = |\mathbb{E}|$. Thus, $\operatorname{rank}(c_{\mathbb{E}}^T(Y\ Z)) = \operatorname{rank}((c_{\mathbb{E}}^T Y\ 0)) = |\mathbb{E}|$ and, consequently, $\det(c_{\mathbb{E}}^T Y) \neq 0$. Therefore, system (1.15) has a unique solution. By substituting (1.14) into the system $Au^* + b = c_{\mathbb{E}}\lambda^*$, we have that

$$AZx_Z^* = c_{\mathbb{E}}\lambda^* - b - AYx_Y^*. \tag{1.16}$$

It follows from (1.16) that

$$(Z^T AZ)x_Z^* = -Z^T(AYx_Y^* + b) \text{ and} \tag{1.17a}$$

$$(c_{\mathbb{E}}^T Y)^T \lambda^* = Y^T(Au^* + b). \tag{1.17b}$$

System (1.17a) is sometimes referred to as the reduced KKT system and the matrix $Z^T AZ$ – the reduced Hessian matrix of the quadratic function f. Therefore, in order to apply the null-space method for KKT system (1.10), we have to select certain matrices Y and Z and to solve certain systems of linear equations. The matrix Z contains the basis vectors of the null-space (kernel) of the matrix $c_{\mathbb{E}}^T$, i.e., the basis vectors of the set $\ker(c_{\mathbb{E}}^T) = \{u \in \mathbb{R}^q : c_{\mathbb{E}}^T u = 0\}$. There are a few known ways to select matrices Y and Z [**1, 71**]. For example, it is not hard to check that $c_{\mathbb{E}}^T Z = 0$ and $\det((Y\ Z)) \neq 0$, if:

(1) $Y = \begin{pmatrix} M^{-1} \\ O \end{pmatrix}$ and $Z = \begin{pmatrix} -M^{-1}N \\ I \end{pmatrix}$, where $(M\ N) = c_{\mathbb{E}}^T$, $M \in \mathbb{R}^{|\mathbb{E}|\times|\mathbb{E}|}$, $N \in \mathbb{R}^{|\mathbb{E}|\times(q-|\mathbb{E}|)}$ (here we assume that the first $|\mathbb{E}|$ columns of the matrix $c_{\mathbb{E}}^T$ are linearly independent vectors; if $c_{\mathbb{E}}^T$ does not satisfy the assumption, we have to sort columns of the matrix $c_{\mathbb{E}}^T$ and corresponding elements of the vector $d_{\mathbb{E}}$ such that the assumption would be satisfied), $O \in \{0\}^{(q-|\mathbb{E}|)\times|\mathbb{E}|}$ is a zero matrix, and $I \in \{0,1\}^{(q-|\mathbb{E}|)\times(q-|\mathbb{E}|)}$ is an identity matrix.

(2) $Y = Q_1$ and $Z = Q_2$, where $(Q_1\ Q_2)\begin{pmatrix} R \\ O \end{pmatrix} = c_{\mathbb{E}}$ is the QR decomposition of the matrix $c_{\mathbb{E}}$. Here $Q_1 \in \mathbb{R}^{q\times|\mathbb{E}|}$, $Q_2 \in \mathbb{R}^{q\times(q-|\mathbb{E}|)}$, $R \in \mathbb{R}^{|\mathbb{E}|\times|\mathbb{E}|}$, and $O \in \{0\}^{(q-|\mathbb{E}|)\times|\mathbb{E}|}$ is a zero matrix.

1.1.2.3. The branch-and-bound method. In applying the branch-and-bound method for an optimization problem, the feasible set is divided into particular subsets [**11, 13, 16**]. This operation is often called branching. All divided feasible subsets are stored in a certain set. Such a set is usually depicted as a tree and called the search tree. The lower, and sometimes the upper, bounds for the minimum value of the objective function on every subset are then evaluated. This operation is often called bounding. The division and evaluation (branching and bounding) operations are performed recursively on every subset until 1) the subset becomes indivisible, or 2) the division is meaningless. The subset becomes indivisible if it contains only one element. In this case, the element is accepted as a current minimizer and the objective value at the current minimizer is accepted as a current objective minimum, if the previous objective minimum is worse (greater) than the current one. An initial objective minimum is usually determined in advance. Such an initial objective minimum is usually equal to the positive infinity, or to the objective value at a feasible point. The division is meaningless if we determine that the subset does not contain the minimizer of the objective function on the feasible set. For example, if the lower bound for the minimum objective value on the subset is worse (greater) than the current objective minimum, then such subset does not contain the minimizer definitely. If the division of the subset is meaningless, we eliminate the subset from further processing. The elimination of subsets is often called pruning. According to the features of the feasible set and the objective function, we may introduce problem-dependent pruning rules. After some time, the process stops. The final minimizer is then accepted as a minimizer of the objective function on the feasible set.

1.1.2.4. Parallel branch-and-bound methods. An algorithm for a particular problem may be sequential or parallel [**54, 57, 62**]. Let us perceive an algorithm as a finite sequence of steps, devoted to achieving a certain goal. We say that the algorithm is a sequential algorithm if all the steps are performed one after another sequentially. If one or more of the

steps are performed in parallel, we say that the algorithm is a parallel algorithm. Suppose that $\mathcal{A}$ is a sequential algorithm for a certain optimization problem and it is based on the branch-and-bound method. There are many strategies to perform the algorithm $\mathcal{A}$ in parallel [**4, 20, 31, 56**]. All of these strategies are usually called parallel branch-and-bound methods. In literature, we may find various classifications of the parallel branch-and-bound methods [**85**]. Let $\mathcal{A}_P$ be a parallel version of the sequential algorithm $\mathcal{A}$. According to the classification given in [**31**], we say that the parallel algorithm $\mathcal{A}_P$ has the parallelism of type

1, if operations such as branching, bounding, pruning, etc., are performed in parallel in order to accelerate the execution of the sequential algorithm $\mathcal{A}$. A parallel algorithm, proposed in this dissertation, has the parallelism of type 1.
2, if the search tree is constructed in parallel. Examples of algorithms that have the parallelism of type 2 may be found in [**61, 73**].
3, if several search trees are constructed in parallel. A few parallel algorithms that have the parallelism of type 3 are proposed in [**55, 67**].

It is clear that a parallel algorithm may have the parallelism of several types. For example, see [**67, 73**].

1.2. Multidimensional scaling

Multidimensional scaling (MDS) is a technique of visualization of multidimensional elements [**7, 19, 26, 60, 82**]. Let us repeat that 1) a multidimensional element is perceived here as a vector in a real vector space of four dimensions or higher, 2) a visualization of multidimensional elements is realized here as a representation of the corresponding vectors in a lower-dimensional Cartesian coordinate system, and 3) an image of multidimensional elements is defined here as a set of points which represent the multidimensional elements in the lower-dimensional Cartesian coordinate system. In applying MDS, the image of multidimensional elements is obtained by specifying and solving a certain optimization problem. A separate case of this optimization problem is the main problem considered in this dissertation. In this section, the optimization problem arising in MDS is introduced and the separate case is presented. A few known algorithms for this separate case are given here, too.

1.2.1. Optimization problem arising in MDS

Suppose that we have n ($n > 1$) multidimensional elements, i.e., n vectors with k ($k > 3$) real components. For the given set of multidimensional elements, let us find a set of corresponding n points in m-dimensional ($m \in \{1, 2, 3\}$) real vector space by using MDS. As a technique, MDS consists of three basic steps: 1) to evaluate relationships between every pair of the given multidimensional elements, 2) to select a desired distance function, defined on the m-dimensional real vector space, and 3) to specify and minimize a particular real function, defined on the space $\mathbb{R}^{mn}$.

A relationship between multidimensional elements i and j is usually called a dissimilarity (sometimes similarity) and is denoted by δ_{ij}. Here, we restrict our attention to the case where $\delta_{ij} \in \mathbb{R}$ and $\delta_{ij} = \delta_{ji} > 0$, $\delta_{ii} = 0$, $1 \leq i, j \leq n$. The dissimilarities between the elements define a square matrix of size n. This matrix is called the dissimilarity matrix and is denoted usually by the symbol Δ^n. Note that multidimensional elements are k-dimensional vectors with real components. Therefore, dissimilarities between them are often evaluated by using any distance function defined on the vector space $\mathbb{R}^k$.

At the second step, the Minkowski distance of order p ($p \geq 1$) is usually selected:

$$d_p(x_i, x_j) = \left(\sum_{k=1}^{m} |x_{ki} - x_{kj}|^p \right)^{1/p}, \tag{1.18}$$

where $x_i = (x_{1i}, \ldots, x_{mi})^T$, $x_j = (x_{1j}, \ldots, x_{mj})^T \in \mathbb{R}^m$ [**90**]. A number p is sometimes referred to as the power of the Minkowski distance.

Any image of the given multidimensional elements obtained by using MDS has the following feature: the desired distances between points of the image are equal to or similar to the dissimilarities between corresponding elements. Such an image is obtained by specifying a certain real function – defined on the space $\mathbb{R}^{mn}$ – and minimizing it on that space. This function is usually called MDS loss function. There are several known loss functions. For example (for the sake of brevity, here we present unweighted loss functions):

(1) Raw Stress [**59**]:

$$f_p(x) = \sum_{i<j}^{n} \left(d_p(x_i, x_j) - \delta_{ij} \right)^2. \tag{1.19}$$

(2) Normalized Stress **[7, 8, 26]**:

$$f_p^{SN}(x) = \frac{f_p(x)}{\sum_{i<j}^{n} \delta_{ij}^2}. \tag{1.20}$$

(3) Stress-1 **[8, 58]**:

$$f_p^{S1}(x) = \sqrt{f_p^{SN}(x)}. \tag{1.21}$$

Here, $x = (x_1^T, \ldots, x_n^T)^T \in \mathbb{R}^{mn}$. Therefore, if $x^* = (x_1^{*T}, \ldots, x_n^{*T})^T \in \mathbb{R}^{mn}$ is a minimizer of a particular loss function on $\mathbb{R}^{mn}$, then $\{x_1^*, \ldots, x_n^*\} \subset \mathbb{R}^m$ is perceived as an image of the given multidimensional elements. In addition, $d_p(x_i^*, x_j^*) \simeq \delta_{ij}$ for all $1 \leq i < j \leq n$, where the function d_p is defined by (1.18). Let us consider loss function (1.19).

Function (1.19) is commonly referred to as the raw Stress (or just Stress) with distance function d_p. Images of the same multidimensional elements are usually different when using different distance functions d_p. Diverse applications of MDS show that images obtained by using city-block distances – i.e., the distance function d_1 (Minkowski distance (1.18) of order $p = 1$) – are sometimes more informative than by using other distances **[2, 30, 92, 93, 95]**. However, if city-block distances are used, the Stress function f_1 contains a set of absolute value (modulus) terms:

$$f_1(x) = \sum_{i<j}^{n} \left(\sum_{k=1}^{m} |x_{ki} - x_{kj}| - \delta_{ij} \right)^2. \tag{1.22}$$

Minimization of such a function

$$\underset{x \in \mathbb{R}^{mn}}{\text{minimize}} \quad f_1(x) \tag{1.23}$$

becomes a complicated optimization problem. The function f_1 has many local minimizers on the set $\mathbb{R}^{mn}$ **[2, 7, 38, 45, 49]**. In addition, it may even be non-differentiable at a minimizer **[93]**. Thus, optimization problem (1.23) is the main problem considered in this dissertation.

1.2.2. Known algorithms for MDS with city-block distances

There are a few known algorithms for finding a local or a global minimizer of the Stress function with city-block distances on the feasible set. Some of them are devoted to minimizing more general function f_p ($p \geq 1$), i.e., the raw Stress with the distance function d_p (see (1.19)). Others are

devoted only to minimizing the function f_1. A brief review of algorithms for minimizing f_p and f_1 may be found in [**40, 95**]. Thus, optimization problem (1.23) may be solved by applying 1) the simulated annealing method [**12, 23, 53**], 2) a two-stage approach with applications of the least-squares regression in the first stage and the simulated annealing in the second [**66**], 3) combinatorial methods [**45, 49, 94**], 4) the tunneling method [**38, 42**], etc. In this dissertation, we propose a few new algorithms for problem (1.23) and reveal features of them. The features of the proposed algorithms are compared with corresponding features of two known algorithms. One of these known algorithms, called SMOOTH, is described in [**39, 40**]. The other one is presented in [**94**]. We called it BB2009. In the rest of this subsection, we give short descriptions of these two known algorithms.

1.2.2.1. SMOOTH. The Stress function with city-block distances has many local minimizers on the feasible set $\mathbb{R}^{mn}$. Optimization algorithms – based, for example, on the steepest descent strategy [**78**] – may "land" at a local minimizer instead of a global [**59**]. Groenen et al. [**40**] suggested an iterative algorithm, called distance smoothing or simply SMOOTH, that tries to avoid local minimizers while looking for a global one. Unfortunately, the final result of the algorithm is sometimes a local minimizer, too. Here, we present the basic steps of this algorithm. In order to better understand the steps, let us introduce 1) a sequence of auxiliary functions that converges pointwise on $\mathbb{R}^{mn}$ to the function f_1, 2) a concept of a function that majorizes another function, and 3) a pseudocode of an MM algorithm.

Let $\epsilon \in \mathbb{R}$ ($\epsilon > 0$) and $\{f_1^\epsilon\}$ denotes a sequence of functions $f_1^\epsilon : \mathbb{R}^{mn} \to \mathbb{R}$ such that

$$f_1^\epsilon(x) = \sum_{i<j}^{n} \left(\sum_{k=1}^{m} h_\epsilon(x_{ki} - x_{kj}) - \delta_{ij} \right)^2,$$

where

$$h_\epsilon(x_{ki} - x_{kj}) = \begin{cases} (x_{ki} - x_{kj})^2/(2\epsilon) + \epsilon/2, & \text{if } |x_{ki} - x_{kj}| < \epsilon, \\ |x_{ki} - x_{kj}|, & \text{if } |x_{ki} - x_{kj}| \geq \epsilon. \end{cases}$$

Function $h_\epsilon(x_{ki} - x_{kj})$ is sometimes referred to as the Huber function [**40, 48**]. It smoothes city-block distances $|x_{ki} - x_{kj}|$ in the function f_1, if they are

optionally small. The function f_1^ϵ is sometimes called the distance smoothing raw Stress. Note that

$$\lim_{\epsilon \to 0} \left(t^2/(2\epsilon) + \epsilon/2 \right) = 0$$

for all $t \in \mathbb{R}$ such that $|t| < \epsilon$. Therefore,

$$\lim_{\epsilon \to 0} f_1^\epsilon(x) = f_1(x)$$

for all $x \in \mathbb{R}^{mn}$, i.e., a sequence of functions $\{f_1^\epsilon\}$ converges pointwise on $\mathbb{R}^{mn}$ to f_1 whenever ϵ tends to zero.

Suppose that $q \in \mathbb{N}$ and $\mathbb{U} \subseteq \mathbb{R}^q$. Let $f : \mathbb{U} \to \mathbb{R}$ be a particular function. We say that function $f^{(M)} : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ majorizes function f at the point $u' \in \mathbb{U}$, if and only if,

$$f(u) \leq f^{(M)}(u, u') \text{ and } f(u') = f^{(M)}(u', u')$$

for all $u \in \mathbb{U}$. The function $f^{(M)}$ is sometimes called a surrogate function of the function f **[50]**. The point u' is sometimes referred to as a supporting point. Surrogate functions are used for optimization algorithms, based on an MM algorithm **[50, 63]**. In literature, we often may find a note that the MM algorithm is just a strategy, and not an algorithm in and of itself. The MM algorithm is based on the fact that, if $u' \in \mathbb{U}$ and $u^* \in \operatorname{argmin}\{f^{(M)}(u, u') : u \in \mathbb{U}\}$, then

$$f(u^*) \leq f^{(M)}(u^*, u') \leq f^{(M)}(u', u') = f(u').$$

A pseudocode of the MM algorithm is shown in Algorithm 1.1. In applying the MM algorithm, usually simple and easily minimized functions are selected as the surrogate functions.

Algorithm 1.1 MM algorithm

Input: $\mathbb{U} \subseteq \mathbb{R}^q$, $f : \mathbb{U} \to \mathbb{R}$, $f^{(M)} : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$, $u^0 \in \mathbb{U}$, $\gamma > 0$
Output: u^* – a local minimizer of f on $\mathbb{U}$

```
u' ← u^0; next ← 1
while next = 1 do
    u* ∈ argmin{f^(M)(u, u') : u ∈ U}
    if f(u') − f(u*) < γ then
        next ← 0
    u' ← u*
```

The basic steps of the algorithm SMOOTH are shown in Algorithm 1.2. First of all, an initial $\epsilon_0 > 0$ and a random $x^0 \in \mathbb{R}^{mn}$ are selected (line 1 of the algorithm). In literature, ϵ_0 is usually equal to $2\max\{n^{-1}\sum_{j=1}^{n}\delta_{ij} : 1 \leq i \leq n\}$ [**74**]. Later, optimization problems

$$\underset{x \in \mathbb{R}^{mn}}{\text{minimize}} \quad f_1^{\epsilon}(x) \tag{1.24}$$

are solved by choosing decreasing values of ϵ (lines 2–4). Problems (1.24) are solved by using the MM algorithm, presented in Algorithm 1.1 (line 4 of the algorithm SMOOTH). Function $f_1^{\epsilon(M)}$ that majorizes function f_1^{ϵ} is described in detail in [**40**]. A minimizer of the previous distance smoothing raw Stress is used as an initial feasible point to start Algorithm 1.1. Finally, after $i_{\max}$ iterations, problem (1.23) is solved by using the same MM algorithm (line 5). A detailed description of the function $f_1^{(M)}$ is given in [**40, 41**]. Note that in order to start the algorithm SMOOTH, an initial feasible point $x^0 \in \mathbb{R}^{mn}$ has to be selected (line 1). Hence, the algorithm returns a minimizer of f_1 on $\mathbb{R}^{mn}$ which depends on the selected feasible point. Thus, it may return a local minimizer instead of a global one.

Algorithm 1.2 SMOOTH

Input: $m \in \{1, 2, 3\}$, $n \in \mathbb{N}$ $(n > 1)$, $\delta_{ij} \in \mathbb{R}$ $(\delta_{ij} = \delta_{ji} > 0,\ \delta_{ii} = 0)$, $1 \leq i, j \leq n$, $\gamma > 0$, $i_{\max} \in \mathbb{N}$

Output: x^* – a local minimizer of f_1 on $\mathbb{R}^{mn}$

1: $\epsilon \leftarrow \epsilon_0 > 0$; $x^0 \in \mathbb{R}^{mn}$
2: **for** $i = 1, i_{\max}$ **do**
3: $\quad \epsilon \leftarrow \epsilon_0(i_{\max} - i + 1)/i_{\max}$
4: $\quad x^i \leftarrow \mathrm{MM}(\mathbb{R}^{mn}, f_1^{\epsilon}, f_1^{\epsilon(M)}, x^{i-1}, \gamma)$
5: $x^* \leftarrow \mathrm{MM}(\mathbb{R}^{mn}, f_1, f_1^{(M)}, x^{i_{\max}}, \gamma)$

1.2.2.2. BB2009. This algorithm is a combinatorial algorithm for finding a global minimizer of the Stress function with city-block distances f_1 on the feasible set $\mathbb{R}^{mn}$. It is based on the fact that the set $\mathbb{R}^{mn}$ may be divided into a finite number of subsets and a certain quadratic function may be defined on every subset separately. From here it follows that problem (1.23) may be formulated as a two-level optimization problem with a combinatorial optimization problem at the upper-level, and a QP problem at the lower-level. In this algorithm, the upper-level problem is solved by using the branch-and-bound method. The lower-level problem is solved by

using a method for QP problems. Let us show that problem (1.23) may be formulated as a two-level optimization problem.

It is not hard to understand that for every $x = (x_1^T, \ldots, x_n^T)^T = (x_{11}, \ldots, x_{m1}, x_{12}, \ldots, x_{m2}, \ldots, x_{1n}, \ldots, x_{mn})^T \in \mathbb{R}^{mn}$, we may assign at least one matrix

$$P = \begin{pmatrix} p_{11} & p_{12} & \ldots & p_{1n} \\ & & \ldots & \\ p_{m1} & p_{m2} & \ldots & p_{mn} \end{pmatrix} \in \{1, \ldots, n\}^{m \times n},$$

such that $(p_{k1}, p_{k2}, \ldots, p_{kn})$ is a permutation of the set $\{1, \ldots, n\}$ and $x_{kp_{ki}} \leq x_{kp_{kj}}$ for all $1 \leq i < j \leq n$, $1 \leq k \leq m$ (see Example 1.1). There are $(n!)^m$ such different matrices. For every matrix P let us define a vector $t = (t_{112}, \ldots, t_{m12}, t_{113}, \ldots, t_{m13}, \ldots, t_{1(n-1)n}, \ldots, t_{m(n-1)n})^T \in \{-1, 1\}^{mn(n-1)/2}$ such that

$$t_{kij} = \begin{cases} -1, & \text{if } q < r, \text{ i.e., } x_{ki} \leq x_{kj}, \\ 1, & \text{if } q > r, \text{ i.e., } x_{ki} \geq x_{kj}, \end{cases}$$

where $p_{kq} = i$ and $p_{kr} = j$, for all $1 \leq i < j \leq n$, $1 \leq k \leq m$ (see Example 1.1). Suppose that $\mathbb{T} \subset \{-1, 1\}^{mn(n-1)/2}$ denotes a set of all such t ($|\mathbb{T}| = (n!)^m$). In turn, for every vector $t \in \mathbb{T}$ we may define a subset of the feasible set $\mathbb{R}^{mn}$, say $\mathbb{X}(t)$, such that $\bigcup_{t \in \mathbb{T}} \mathbb{X}(t) = \mathbb{R}^{mn}$ and

$$\begin{aligned} \mathbb{X}(t) = \; & \{x \in \mathbb{R}^{mn} : \; x_{ki} \leq x_{kj}, \text{ if } t_{kij} = -1 \text{ and } x_{ki} \geq x_{kj}, \text{ if } t_{kij} = 1, \\ & \qquad 1 \leq i < j \leq n, \; 1 \leq k \leq m\} = \\ = \; & \{x \in \mathbb{R}^{mn} : \; (x_{ki} - x_{kj}) t_{kij} \geq 0, \; 1 \leq i < j \leq n, \; 1 \leq k \leq m\}. \end{aligned}$$

Therefore, the feasible set $\mathbb{R}^{mn}$ may be divided into a finite number of subsets $\mathbb{X}(t)$, $t \in \mathbb{T}$.

Let $f : \mathbb{T} \times \mathbb{R}^{mn} \to \mathbb{R}$ be a function such that

$$f(t, x) = \sum_{i<j}^{n} \left(\sum_{k=1}^{m} (x_{ki} - x_{kj}) t_{kij} - \delta_{ij} \right)^2. \tag{1.25}$$

It is not hard to check that $|x_{ki} - x_{kj}| = (x_{ki} - x_{kj}) t_{kij}$ for all $1 \leq i < j \leq n$, $1 \leq k \leq m$, $t \in \mathbb{T}$ and $x \in \mathbb{X}(t)$. Hence, $f_1(x) = f(t, x)$ whenever $t \in \mathbb{T}$, $x \in \mathbb{X}(t)$. For every $t \in \mathbb{T}$, the function f is a convex quadratic function on the subset $\mathbb{X}(t)$. In turn, the function f_1 is a piecewise quadratic function on the set $\mathbb{R}^{mn}$. Therefore, optimization problem (1.23) may be formulated

as the following two-level optimization problem

$$\underset{t \in \mathbb{T},\, x^* \in \mathbb{X}_f^*(t)}{\text{minimize}} \quad f(t, x^*), \tag{1.26}$$

where the set $\mathbb{X}_f^*(t) \subset \mathbb{R}^{mn}$ denotes a set of all minimizers of the following problem

$$\underset{x \in \mathbb{X}(t)}{\text{minimize}} \quad f(t, x). \tag{1.27}$$

Upper-level problem (1.26) is a combinatorial optimization problem and lower-level problem (1.27) is a convex QP problem.

The algorithm BB2009 has a parallel version, described in [**98**]. Let PBB2012 be the parallel version of the algorithm BB2009. Then the parallel algorithm PBB2012 has the parallelism of type 2 (see Section 1.1.2.4).

EXAMPLE 1.1. *For vector* $x = (x_{11}, \ldots, x_{16})^T = (7, -1, 3, -4, 3, 8)^T$, *we may assign two matrices* $P_1 = (p_{11}^1 \ldots p_{16}^1) = (4\,2\,3\,5\,1\,6)$ *and* $P_2 = (p_{11}^2 \ldots p_{16}^2) = (4\,2\,5\,3\,1\,6)$ *such that* $x_{1p_{1i}^1} \leq x_{1p_{1j}^1}$ *and* $x_{1p_{1i}^2} \leq x_{1p_{1j}^2}$ *for all* $1 \leq i < j \leq 6$. *Indeed,* $p_{11}^1 = 4$, $p_{12}^1 = 2$ *and* $x_{14} = -4 < -1 = x_{12}$, $p_{13}^1 = 3$ *and* $x_{14} = -4 < 3 = x_{13}$, *etc.*

Matrix P_1 *corresponds to vector* $t^1 = (1,\ 1,\ 1,\ 1,\ -1,\ -1,\ 1,\ -1,\ -1,\ 1,\ \mathbf{-1},\ -1,\ -1,\ -1,\ -1)^T \in \{-1, 1\}^{15}$. *Matrix* P_2, *to vector* $t^2 = (1,\ 1,\ 1,\ 1,\ -1,\ -1,\ 1,\ -1,\ -1,\ 1,\ \mathbf{1},\ -1,\ -1,\ -1,\ -1)^T \in \{-1, 1\}^{15}$. *All components of the vectors* t^1 *and* t^2 *are identical, except the component at the position* 135, *i.e.,* $t_{135}^1 = -1$, $t_{135}^2 = 1$.

2. Algorithms for MDS with city-block distances

The research problem – i.e., the problem of minimizing the Stress function with city-block distances – is equivalent to a certain optimization problem. The optimization problem, equivalent to the research problem, is called here the problem CMDS. In this chapter, first of all, the main features of the Stress function with city-block distances are listed. Next, the problem CMDS is presented, and the equivalence between the research problem and the problem CMDS is proven. Finally, three algorithms for the problem CMDS are proposed and described. Certain parts of this chapter are published in [**FGŽ14, GŽ14, GŽ13**].

2.1. The Stress function with city-block distances

Here, we present the main features of the Stress function with city-block distances, defined in (1.22). Namely, here we show that the function f_1 is invariant under translation and under mirroring, and is not differentiable everywhere over its domain. Let $L \in \mathbb{R}^{l_1 \times l_2}$ and

$$\mathrm{diag}_k(L) = \mathrm{diag}(\underbrace{L, \ldots, L}_{k}) = \begin{pmatrix} L & & \\ & \ddots & \\ & & L \end{pmatrix} \in \mathbb{R}^{l_1 k \times l_2 k} \tag{2.1}$$

denote a rectangular block diagonal matrix.

2.1.1. Invariance under translation

If $c \in \mathbb{R}$, then $|(x_{ki} - c) - (x_{kj} - c)| = |x_{ki} - x_{kj}|$ for all $x \in \mathbb{R}^{mn}$. Therefore:

$$f_1(x) = f_1(x + v)$$

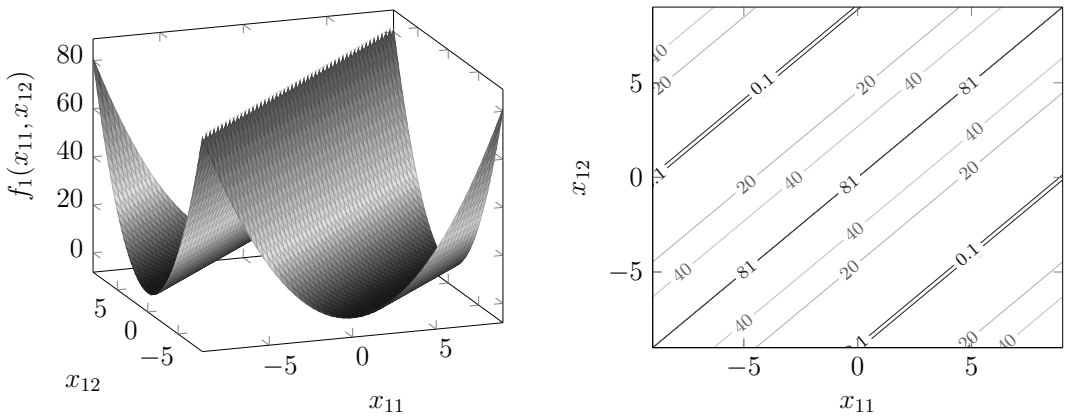
for all $x \in \mathbb{R}^{mn}$ and $v = (c_1, \ldots, c_m, \ldots, c_1, \ldots, c_m)^T \in \mathbb{R}^{mn}$. Because of this feature, we say that the function f_1 is invariant under translation. Due to this feature, we may look for a minimizer of f_1 on a subset of the set $\mathbb{R}^{mn}$, instead of on the whole set $\mathbb{R}^{mn}$. For example, on the subset $\mathbb{X}' = \{x \in \mathbb{R}^{mn} : \sum_{i=1}^{n} x_{ki} = 0,\ 1 \leq k \leq m\} \subset \mathbb{R}^{mn}$. In addition, if $x =$

$(x_1^T, \ldots, x_n^T)^T \in \mathbb{X}'$, then points of the set $\{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ are located around the origin of the m-dimensional Cartesian coordinate system.

EXAMPLE 2.1. *If $m = 1$ and $n = 2$, then*

$$f_1(x_{11}, x_{12}) = f_1(x_{11} + c, x_{12} + c)$$

for all $(x_{11}, x_{12})^T \in \mathbb{R}^2$ and $c \in \mathbb{R}$ (see Figure 2.1).



(A) Graph. (B) Contour lines.

FIGURE 2.1. Graph and contour lines of function $f_1(x) = (|x_{11} - x_{12}| - 9)^2$, $x = (x_{11}, x_{12})^T \in \mathbb{R}^2$.

2.1.2. Invariance under mirroring I

If $p = (p_1, \ldots, p_m)^T \in \mathbb{P}(m)$ – where $\mathbb{P}(m)$ denotes a set of all possible permutations of the set $\{1, \ldots, m\}$ – then $\sum_{k=1}^m |x_{ki} - x_{kj}| = \sum_{k=1}^m |x_{p_k i} - x_{p_k j}|$ for all $x \in \mathbb{R}^{mn}$. Therefore:

$$f_1(x) = f_1(\Pi(p)x) \tag{2.2}$$

for all $x \in \mathbb{R}^{mn}$ and $p \in \mathbb{P}(m)$, where $\Pi(p) = \mathrm{diag}_n(I(p)) \in \{0,1\}^{mn \times mn}$, $I(p) = (e_{p_1} \ldots e_{p_m}) \in \{0,1\}^{m \times m}$ and $e_{p_k} \in \{0,1\}^m$, $\|e_{p_k}\| = 1$, $e_{p_k p_k} = 1$, $1 \leq k \leq m$. In this case, we say that the function f_1 is invariant under mirroring when the mirrored points are obtained by exchanging coordinate axes in the m-dimensional Cartesian coordinate system.

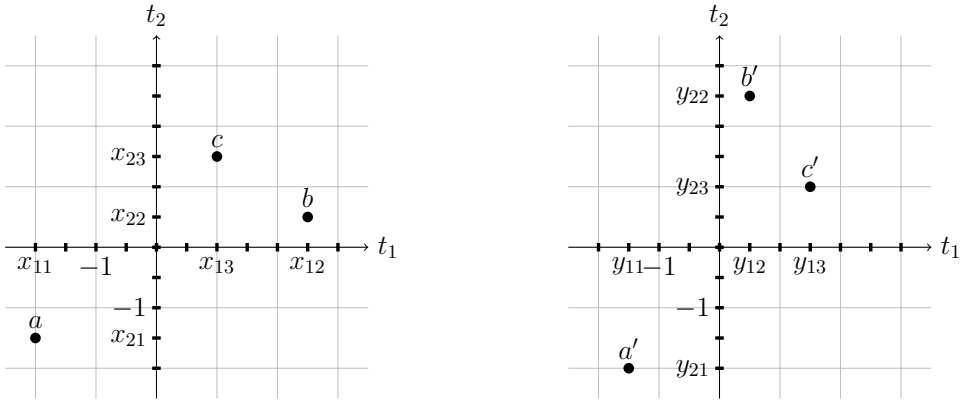
EXAMPLE 2.2. *If $m = 2$, $n = 3$ and $x = (-2, -1.5, 2.5, 0.5, 1, 1.5)^T \in \mathbb{R}^6$, then*

$$f_1(x) = f_1(\Pi(1,2)x) = f_1(\Pi(2,1)x) = f_1(y),$$

where

$$\Pi(1,2) = \left(\begin{array}{cc|cccc} \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{array}\right), \Pi(2,1) = \left(\begin{array}{cc|cccc} 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 \end{array}\right)$$

and $\Pi(2,1)x = y = (-1.5, -2, 0.5, 2.5, 1.5, 1)^T \in \mathbb{R}^6$. *Mirrored points, defined by the vectors* x *and* y*, are shown in Figure 2.2.*



(A) Points, defined by the vector x. (B) Points, defined by the vector $y = \Pi(2,1)x$.

FIGURE 2.2. Mirrored points, defined by vectors x and y.

2.1.3. Invariance under mirroring II

Note that $|x_{ki} - x_{kj}| = |(-x_{ki}) - (-x_{kj})| = |x_{kj} - x_{ki}|$ for all $x \in \mathbb{R}^{mn}$. Therefore:

$$f_1(x) = f_1(P(u)x)$$

for all $x \in \mathbb{R}^{mn}$, where $P(u) = \mathrm{diag}_n(T(u)) \in \{-1,0,1\}^{mn \times mn}$, $T(u) = \mathrm{diag}(u_1, \ldots, u_m) \in \{-1,0,1\}^{m \times m}$ and $u \in \{-1,1\}^m$. We say that f_1 is invariant under mirroring when the mirrored points are obtained by exchanging directions of coordinate axes in the m-dimensional Cartesian coordinate system.

EXAMPLE 2.3. *Suppose that* $m = 2$, $n = 3$ *and* $x = (2.5, 1, 1.5, 3.5, 4, 2)^T \in \mathbb{R}^6$. *Then*

$$f_1(x) = f_1(P(u_1, u_2)x),$$

where

$$P(u_1,u_2) = \left(\begin{array}{cc|cccc} \mathbf{u_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{u_2} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \mathbf{u_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{u_2} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \mathbf{u_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{u_2} \end{array}\right)$$

and $(u_1, u_2)^T \in \{(1,1)^T, (-1,1)^T, (-1,-1)^T, (1,-1)^T\}$. *Mirrored points, defined by vectors* $P(u_1, u_2)x$, $(u_1, u_2)^T \in \{-1, 1\}^2$, *are shown in Figure 2.3.*

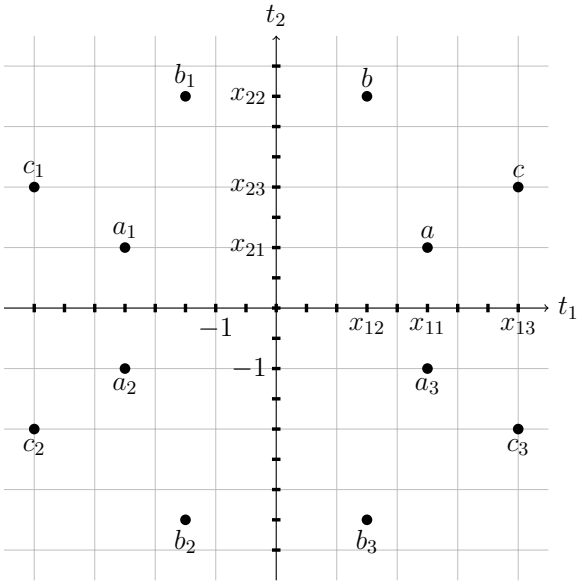


FIGURE 2.3. Mirrored points, defined by vectors $P(1,1)x = x$ (points a, b, c), $P(-1,1)x$ (points a_1, b_1, c_1), $P(-1,-1)x$ (points a_2, b_2, c_2) and $P(1,-1)x$ (points a_3, b_3, c_3).

2.1.4. Non-differentiability everywhere over its domain

Note that

$$\frac{\partial f_1}{\partial x_{qr}}(x) = 2\sum_{i<j}^{n}\left[\left(\sum_{k=1}^{m}|x_{ki} - x_{kj}| - \delta_{ij}\right)\cdot\frac{\partial|x_{qi} - x_{qj}|}{\partial x_{qr}}\right],$$

where

$$\frac{\partial|x_{qi} - x_{qj}|}{\partial x_{qr}} = \frac{(x_{qi} - x_{qj})}{|x_{qi} - x_{qj}|}\cdot\frac{\partial(x_{qi} - x_{qj})}{\partial x_{qr}} = \begin{cases} \text{sign}(x_{qi} - x_{qj}), & \text{if } i = r, \\ -\text{sign}(x_{qi} - x_{qj}), & \text{if } j = r, \\ 0, & \text{otherwise,} \end{cases}$$

for all $1 \le q \le m$, $1 \le r \le n$. Therefore, the Stress function with city-block distances is not differentiable at a point $x \in \mathbb{R}^{mn}$, when $x_{qi} = x_{qj}$ with certain $q \in \{1, \ldots, m\}$ and $i, j \in \{1, \ldots, n\}$ $(i < j)$. Thus, the function f_1 is not differentiable everywhere over its domain $\mathbb{R}^{mn}$. The function may be non-differentiable even at a local minimizer **[93]**.

EXAMPLE 2.4. *Suppose that* $m = 2$, $n = 6$ *and matrix*

$$\Delta^6 = \begin{pmatrix} 0 & 1.21 & 0.81 & 1.12 & 0.98 & 0.69 \\ 1.21 & 0 & 1.04 & 1.18 & 0.75 & 1.33 \\ 0.81 & 1.04 & 0 & 1.24 & 0.58 & 0.92 \\ 1.12 & 1.18 & 1.24 & 0 & 0.95 & 0.89 \\ 0.98 & 0.75 & 0.58 & 0.95 & 0 & 0.98 \\ 0.69 & 1.33 & 0.92 & 0.89 & 0.98 & 0 \end{pmatrix}$$

contains dissimilarities between certain multidimensional elements $\{v_1, \ldots, v_6\} \subset \mathbb{R}^k$ *(*$k > 3$*). Then* $x^* = (x_1^{*T}, \ldots, x_6^{*T})^T = (-0.062, -0.547, -0.062, 0.664, -0.442, -0.104, 0.574, 0.102, -0.259, 0.202, 0.251, -0.317)^T \in \mathbb{R}^{12}$ *is a minimizer of* f_1 *on* $\mathbb{R}^{12}$*. However, the function* f_1 *is not differentiable at the point* x^**, because* $x_{11}^* = x_{12}^*$*. The corresponding image of these vectors in a 2-dimensional Cartesian coordinate system is shown in Figure 2.4.*

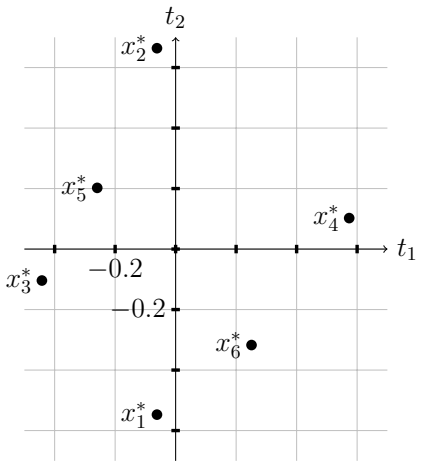


FIGURE 2.4. An image of multidimensional elements $v_i \in \mathbb{R}^k$ $(k > 3)$, $1 \leq i \leq 6$, obtained by specifying and minimizing the loss function f_1. Here, $x_1^* = (-0.062, -0.547)^T$, $x_2^* = (-0.062, 0.664)^T$.

2.2. Equivalent problem to MDS with city-block distances

Let $m \in \{1, 2, 3\}$, $n \in \mathbb{N}$ $(n > 1)$ and $\delta_{ij} \in \mathbb{R}$ $(\delta_{ij} = \delta_{ji} > 0, \delta_{ii} = 0)$, $1 \leq i, j \leq n$. Suppose that $x = (x_{11}, \ldots, x_{m1}, x_{12}, \ldots, x_{m2}, \ldots, x_{mn})^T \in \mathbb{R}^{mn}$ and $d^{\pm} = (d_{112}^+, d_{112}^-, \ldots, d_{m12}^+, d_{m12}^-, \ldots, d_{m(n-1)n}^+, d_{m(n-1)n}^-)^T \in \mathbb{R}^{mn(n-1)}$. Let $\mathbb{Y}$ be a subset of $\mathbb{R}^{mn^2}$ such that

$$\begin{aligned} \mathbb{Y} = \{y = (x^T, d^{\pm T})^T \in \mathbb{R}^{mn^2} : \; & x_{ki} - x_{kj} = d_{kij}^+ - d_{kij}^-, \\ & d_{kij}^+ d_{kij}^- = 0, \\ & d_{kij}^+, d_{kij}^- \geq 0, \\ & 1 \leq i < j \leq n, 1 \leq k \leq m\}, \end{aligned} \tag{2.3}$$

and let $g : \mathbb{Y} \to \mathbb{R}$ be a real function such that

$$g(y) = \sum_{i<j}^{n} \left(\sum_{k=1}^{m} (d_{kij}^{+} + d_{kij}^{-}) - \delta_{ij} \right)^2. \tag{2.4}$$

We state that optimization problem (1.23) is equivalent to the following optimization problem:

$$\begin{array}{ll} \text{minimize} & g(y). \\ y \in \mathbb{Y} & \end{array} \tag{2.5}$$

Let us show that problems (1.23) and (2.5) are equivalent problems.

THEOREM 2.1. *Optimization problems (1.23) and (2.5) are equivalent problems.*

PROOF. First of all, let us assume that $\mathbb{X} = \mathbb{R}^{mn}$ denotes the feasible set of problem (1.23), and $\mathbb{X}_{f_1}^*$, $\mathbb{Y}_g^*$ denote sets of all global minimizers of functions f_1, g on $\mathbb{X}$, $\mathbb{Y}$, respectively. According to the definition of the equivalence of two optimization problems, let us show that there exists a bijective (injective and surjective) function $h : \mathbb{X} \to \mathbb{Y}$ such that

$$x^* \in \mathbb{X}_{f_1}^*, \text{ if and only if } h(x^*) = y^* \in \mathbb{Y}_g^*. \tag{2.6}$$

Let us consider the following function:

$$h(x) = \begin{pmatrix} x \\ z^{\pm}(x) \end{pmatrix},$$

where $z^{\pm}(x) = (z_{112}^{+}(x),\, z_{112}^{-}(x), \ldots,\, z_{m12}^{+}(x),\, z_{m12}^{-}(x), \ldots,\, z_{m(n-1)n}^{+}(x),\, z_{m(n-1)n}^{-}(x))^T \in \mathbb{R}^{mn(n-1)}$ and

$$z_{kij}^{+}(x) = \begin{cases} x_{ki} - x_{kj}, & \text{if } x_{ki} > x_{kj}, \\ 0, & \text{otherwise}, \end{cases}$$
$$z_{kij}^{-}(x) = \begin{cases} -(x_{ki} - x_{kj}), & \text{if } x_{ki} < x_{kj}, \\ 0, & \text{otherwise}. \end{cases}$$

Note that

$$\begin{aligned} z_{kij}^{+}(x) - z_{kij}^{-}(x) &= x_{ki} - x_{kj}, \\ z_{kij}^{+}(x) z_{kij}^{-}(x) &= 0, \\ z_{kij}^{+}(x), z_{kij}^{-}(x) &\geq 0, \end{aligned}$$

for all $1 \le i < j \le n$, $1 \le k \le m$ whenever $x \in \mathbb{X}$. Therefore, $h(x) \in \mathbb{Y}$ for all $x \in \mathbb{X}$.

It follows directly from the definition of the function h that it is injective, i.e., if $h(x^1) = h(x^2)$, then $x^1 = x^2$ for all x^1, $x^2 \in \mathbb{X}$.

The function h is surjective, i.e., for every $y = (x^T, d^{\pm T})^T \in \mathbb{Y}$ there exists a certain $x \in \mathbb{X}$ such that $y = h(x)$. Indeed, suppose that $(x^T, d^{\pm T})^T \in \mathbb{Y}$. Let us show that $d^{\pm} = z^{\pm}(x)$. According to the definition of the set $\mathbb{Y}$, we have that

$$\begin{array}{l} \text{if } d^{+}_{kij} = 0, \text{ then } d^{-}_{kij} = -(x_{ki} - x_{kj}) \ge 0 \text{ and} \\ \text{if } d^{-}_{kij} = 0, \text{ then } d^{+}_{kij} = x_{ki} - x_{kj} \ge 0. \end{array}$$

Suppose that $z^{+}_{kij}(x) = 0$. Then $x_{ki} \le x_{kj}$ and $z^{-}_{kij}(x) = -(x_{ki} - x_{kj}) \ge 0$. If $z^{-}_{kij}(x) = 0$, then $x_{ki} \ge x_{kj}$ and $z^{+}_{kij}(x) = x_{ki} - x_{kj} \ge 0$. Thus, $d^{\pm} = z^{\pm}(x)$ whenever $(x^T, d^{\pm T})^T = y \in \mathbb{Y}$. From here it follows that for every $y \in \mathbb{Y}$ there exists $x \in \mathbb{X}$ such that $y = h(x)$. Therefore, the function h is surjective.

From the fact that, if $(x^T, d^{\pm T})^T \in \mathbb{Y}$, then $d^{+}_{kij} + d^{-}_{kij} = |x_{ki} - x_{kj}|$ for all $1 \le i < j \le n$, $1 \le k \le m$, it follows that

$$f_1(x) = g(y) \text{ whenever } y = (x^T, d^{\pm T})^T \in \mathbb{Y}. \tag{2.7}$$

Thus, $x^* \in \mathbb{X}^*_{f_1}$, i.e., $f_1(x^*) \le f_1(x)$ for all $x \in \mathbb{X}$, if and only if $g(h(x^*)) \le g(h(x))$ for all $h(x) \in \mathbb{Y}$, i.e., $h(x^*) = y^* \in \mathbb{Y}^*_g$. Therefore, the bijective function h satisfies biconditional statement (2.6). From here it follows that problems (1.23) and (2.5) are equivalent problems. □

The function f_1 is invariant under translation (see Section 2.1.1). Due to the fact that $f_1(x) = g(y)$ whenever $y = (x^T, d^{\pm T})^T \in \mathbb{Y}$, it follows that the function g is invariant under translation, too. Let

$$\mathbb{Y}' = \mathbb{Y} \cap \{y \in (x^T, d^{\pm T})^T \in \mathbb{R}^{mn^2} : \sum_{i=1}^{n} x_{ki} = 0,\ 1 \le k \le m\}.$$

Let us consider the following optimization problem, which is called here the problem CMDS:

$$\underset{y \in \mathbb{Y}'}{\text{minimize}} \quad g(y). \tag{2.8}$$

The objective function of problem (2.8) is a quadratic function, and the feasible set is defined by linear and complementarity constraints. Thus,

the problem may be expressed into a certain matrix form. Let us write problem (2.8) into this form. Suppose that $y = (x^T, d^{\pm T})^T = (y_1, \ldots, y_{mn}, y_{mn+1}, \ldots, y_{mn^2})^T \in \mathbb{R}^{mn^2}$, $s = mn(n-1)/2$ and $\mathrm{diag}_k(L) \in \mathbb{R}^{l_1 k \times l_2 k}$, where $L \in \mathbb{R}^{l_1 \times l_2}$, denotes a rectangular block diagonal matrix, defined by (2.1). For the sake of simplicity, let $I^l \in \{0,1\}^{l \times l}$ denote an identity matrix and $O^{l_1 \times l_2} \in \{0\}^{l_1 \times l_2}$, a zero matrix. Then problem (2.8) can be written into the following matrix form:

$$\begin{array}{ll} \underset{y \in \mathbb{R}^{mn^2}}{\text{minimize}} & 0.5 y^T A y - b^T y + \text{const.} \\ \text{subject to} & c_i^T y = 0,\ i \in \mathbb{E} = \{1, \ldots, m+s\}, \\ & c_i^T y \geq 0,\ i \in \mathbb{I} = \{m+s+1, \ldots, m+3s\}, \\ & y_{(mn+i)} y_{(mn+1+i)} = 0,\ i = 1, 3, 5, \ldots, 2s-1. \end{array} \tag{2.9}$$

where:

$$\begin{aligned} A &= \begin{pmatrix} O^{mn \times mn} & O^{mn \times 2s} \\ O^{2s \times mn} & \mathrm{diag}_{n(n-1)/2}(E) \end{pmatrix} \in \{0,1\}^{mn^2 \times mn^2}, \\ b &= (O^{1 \times mn}, \Delta_{12}^T, \Delta_{13}^T, \ldots, \Delta_{(n-1)n}^T)^T \in \mathbb{R}^{mn^2}, \\ c_{\mathbb{E}} &= \begin{pmatrix} C^1 & C^2 \\ O^{2s \times m} & \mathrm{diag}_s\left(\begin{smallmatrix} -1 \\ 1 \end{smallmatrix}\right) \end{pmatrix} \in \{-1,0,1\}^{mn^2 \times m+s}, \\ c_{\mathbb{I}} &= \begin{pmatrix} O^{mn \times 2s} \\ I^{2s} \end{pmatrix} \in \{0,1\}^{mn^2 \times 2s}. \end{aligned}$$

Here, $E \in \{1\}^{2m \times 2m}$, $\Delta_{ij} = (\delta_{ij}, \ldots, \delta_{ij})^T \in \mathbb{R}^{2m}$, $1 \leq i < j \leq n$, $C^1 \in \{0,1\}^{mn \times m}$, $C^2 \in \{-1,0,1\}^{mn \times s}$ and

$$C^1 = \begin{pmatrix} I^m \\ \vdots \\ I^m \end{pmatrix},$$

$$C^2 = \begin{pmatrix} I^m & I^m & \ldots & I^m & & & & & & \\ -I^m & & & & I^m & I^m & \ldots & I^m & & \\ & -I^m & & & -I^m & & & & & \\ & & & & & -I^m & & & & \\ & & \ddots & & & & \ddots & & \ldots & \\ & & & & & & & & & I^m \\ & & & -I^m & & & & -I^m & & -I^m \end{pmatrix}.$$

Let us pay attention to the fact that $\det(A) = 0$. In addition, $A = ((2m)^{-1/2}A)^T\ ((2m)^{-1/2}A)$. Therefore, $A \succeq 0$, i.e., the Hessian matrix A of the quadratic function g is positive semidefinite. Hence, the function g is a non-strictly convex quadratic function. In order to better understand the structure of the matrices A, $(c_{\mathbb{E}}\, c_{\mathbb{I}})$ and the vector b, let us show an example.

EXAMPLE 2.5. *Suppose that* $n = 3$, $m = 2$ *and* $\Delta^3 = \begin{pmatrix} 0 & 7 & 12 \\ 7 & 0 & 3 \\ 12 & 3 & 0 \end{pmatrix}$. *In this case,* $s = 6$, $A \in \{0,1\}^{18\times 18}$, $b \in \mathbb{R}^{18}$, $\mathbb{E} = \{1,\dots,8\}$, $\mathbb{I} = \{9,\dots,20\}$, $(c_{\mathbb{E}}\, c_{\mathbb{I}}) \in \{-1,0,1\}^{18\times 20}$ *and:*

$$A = \left(\begin{array}{cccccc|cccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&\mathbf{1}&\mathbf{1}&\mathbf{1}&\mathbf{1}
\end{array}\right),$$

$$b = \begin{pmatrix} 0&0&0&0&0&0&\mathbf{7}&\mathbf{7}&\mathbf{7}&\mathbf{7}&\mathbf{12}&\mathbf{12}&\mathbf{12}&\mathbf{12}&\mathbf{3}&\mathbf{3}&\mathbf{3}&\mathbf{3} \end{pmatrix}^T,$$

$$(c_{\mathbb{E}}\, c_{\mathbb{I}}) = \left(\begin{array}{cccccccc|cccccccccccc}
\mathbf{1}&0&\mathbf{1}&0&\mathbf{1}&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&\mathbf{1}&0&\mathbf{1}&0&\mathbf{1}&0&0&0&0&0&0&0&0&0&0&0&0\\
\mathbf{1}&0&\mathbf{-1}&0&0&0&\mathbf{1}&0&0&0&0&0&0&0&0&0&0&0\\
0&\mathbf{1}&0&\mathbf{-1}&0&0&0&\mathbf{1}&0&0&0&0&0&0&0&0&0&0\\
\mathbf{1}&0&0&0&\mathbf{-1}&0&\mathbf{-1}&0&0&0&0&0&0&0&0&0&0&0\\
0&\mathbf{1}&0&0&0&\mathbf{-1}&0&\mathbf{-1}&0&0&0&0&0&0&0&0&0&0\\
\hline
0&0&\mathbf{-1}&0&0&0&0&0&\mathbf{1}&0&0&0&0&0&0&0&0&0\\
0&0&\mathbf{1}&0&0&0&0&0&0&\mathbf{1}&0&0&0&0&0&0&0&0\\
0&0&0&\mathbf{-1}&0&0&0&0&0&0&\mathbf{1}&0&0&0&0&0&0&0\\
0&0&0&\mathbf{1}&0&0&0&0&0&0&0&\mathbf{1}&0&0&0&0&0&0\\
0&0&0&0&\mathbf{-1}&0&0&0&0&0&0&0&\mathbf{1}&0&0&0&0&0\\
0&0&0&0&\mathbf{1}&0&0&0&0&0&0&0&0&\mathbf{1}&0&0&0&0\\
0&0&0&0&0&\mathbf{-1}&0&0&0&0&0&0&0&0&\mathbf{1}&0&0&0\\
0&0&0&0&0&\mathbf{1}&0&0&0&0&0&0&0&0&0&\mathbf{1}&0&0\\
0&0&0&0&0&0&\mathbf{-1}&0&0&0&0&0&0&0&0&0&\mathbf{1}&0\\
0&0&0&0&0&0&\mathbf{1}&0&0&0&0&0&0&0&0&0&0&\mathbf{1}
\end{array}\right).$$

2.3. Algorithm, based on the active-set method

Here, we propose and describe an algorithm for problem (2.9). The proposed algorithm is based on the active-set method for linearly constrained

convex QP problems (see Section 1.1.2.1). Let us remember that the active-set method constructs a sequence of feasible points which converges to a minimizer of a convex quadratic function on a feasible set, defined by linear constraints. The feasible set of problem (2.9) is defined by both linear and complementarity (non-linear) constraints. Therefore, we cannot apply the active-set method directly to problem (2.9). We modified the method so that every point of the sequence would satisfy both linear and complementarity constraints. Unfortunately, the proposed algorithm usually returns a local minimizer of g on $\mathbb{Y}'$ instead of a global one. Let us describe the algorithm.

A pseudocode of the algorithm, based on the active-set method, is presented in Algorithm 2.1. We called the algorithm "Modified Active-Set" (MAS). Let us describe the main steps of MAS in detail. A number after the word "line" or "lines" indicates the line number in the pseudocode.

Lines 1–10 of Algorithm 2.1 define the part of initialization. Here, an initial feasible point $y^0 \in \mathbb{Y}'$ is selected (line 1) and an initial active set $\mathbb{W}^0 \subset \{i \in \mathbb{E} \cup \mathbb{I} : c_i^T y^0 = 0\}$ is constructed (lines 2–9). A feasible point $y^0 \in \mathbb{Y}'$ may be selected by using various techniques. One of the techniques is presented in Section 3.1. The set $\mathbb{W}^0$ contains indices of all equality constraints (line 2) and indices of some inequality constraints (lines 3–9). Because of $y^0_{(mn+i)} y^0_{(mn+1+i)} = 0$ for all $i \in \{1, 3, 5, \ldots, 2s-1\}$, it follows that

$$c^T_{(m+s+i)} y^0 = 0 \text{ and/or } c^T_{(m+s+1+i)} y^0 = 0 \tag{2.13}$$

for all $i \in \{1, 3, 5, \ldots, 2s-1\}$. If only one of two equalities (2.13) holds, either index $(m+s+i)$ or index $(m+s+1+i)$ is added to the set $\mathbb{W}^0$ (lines 4–6, 9). If both of two equalities (2.13) hold, a randomly selected index is added (lines 7–9). Note that c_i, $i \in \mathbb{W}^0$, are linearly independent vectors.

Lines 11–37 of the algorithm define the part of calculations. Here, a finite sequence of feasible points $\{y^1, y^2, \ldots, y^K = y^*\} \subset \mathbb{Y}'$ is constructed such that $g(y^{k+1}) \leq g(y^k)$, $0 \leq k \leq K-1$. Every element of the sequence is defined by the following formula: $y^{k+1} = y^k + \alpha^k p^k$, $0 \leq k \leq K-1$. The vector $p^k \in \mathbb{R}^{mn^2}$ is called a step direction and the number $\alpha^k \in [0, 1]$ is called a step length.

The step direction p^k is a minimizer of

$$\phi(p) = 0.5(y^k + p)^T A(y^k + p) - b^T(y^k + p)$$

Algorithm 2.1 Modified Active-Set (MAS)

Input: $m \in \{1,2,3\}$, $n \in \mathbb{N}$ $(n > 1)$, $\delta_{ij} \in \mathbb{R}$ $(\delta_{ij} = \delta_{ji} > 0,\ \delta_{ii} = 0)$, $1 \le i, j \le n$
Output: y^* – a local minimizer of g on $\mathbb{Y}'$

$y^0 \leftarrow$ any feasible point
$\mathbb{W}^0 \leftarrow \mathbb{E}$
for all $i \in \{1, 3, \ldots, 2s-1\}$ **do**
    $j \leftarrow 0$
    if $y^0_{(mn+1+i)} = 0$ **then**
        $j \leftarrow 1$
        if $y^0_{(mn+i)} = 0$ **then**
            $j \leftarrow$ random number from the set $\{0, 1\}$
    $\mathbb{W}^0 \leftarrow \mathbb{W}^0 \cup \{m + s + i + j\}$
$stop \leftarrow 0$; $k \leftarrow 0$
while $stop \neq 1$ **do**
    $\begin{pmatrix} p^k \\ \lambda^k \end{pmatrix} \in \left\{ \begin{pmatrix} p \\ \lambda \end{pmatrix} \in \mathbb{R}^{mn^2+|\mathbb{W}^k|} : \begin{pmatrix} -A & c_{\mathbb{W}^k} \\ c^T_{\mathbb{W}^k} & 0 \end{pmatrix} \begin{pmatrix} p \\ \lambda \end{pmatrix} = \begin{pmatrix} Ay^k - b \\ 0 \end{pmatrix} \right\}$
    if $p^k \neq 0$ **then**
        $\tilde{\mathbb{I}} \leftarrow \{i \in \mathbb{I} : i \notin \mathbb{W}^k \text{ and } c_i^T p^k < 0\}$
        $\alpha^k \leftarrow \min\{1, \min\{-(c_i^T y^k)/(c_i^T p^k) : i \in \tilde{\mathbb{I}}\}\}$
        if $\alpha^k = -(c_{i^*}^T y^k)/(c_{i^*}^T p^k) \le 1$ with some $i^* \in \tilde{\mathbb{I}}$ **then**
            $\mathbb{W}^{k+1} \leftarrow \mathbb{W}^k \cup \{i^*\}$
        $y^{k+1} \leftarrow y^k + \alpha^k p^k$
    else
        $\lambda \leftarrow 0 \in \mathbb{R}^{m+3s}$; $j \leftarrow 0$
        for all $i \in \mathbb{W}^k$ **do**
            $j \leftarrow j + 1$; $\lambda_i \leftarrow \lambda_j^k$
        $\tilde{\mathbb{I}} \leftarrow \{i \in \mathbb{W}^k \cap \mathbb{I} : \lambda_i < 0\}$
        $next \leftarrow 1$
        while $next = 1$ **do**
            if $\tilde{\mathbb{I}} \neq \emptyset$ **then**
                $i^* \leftarrow \operatorname{argmin}\{\lambda_i : i \in \tilde{\mathbb{I}}\}$
                $j \leftarrow 1$
                if $(i^* - (m+s))\%2 = 0$ **then**
                    $j \leftarrow -1$
                if $(i^* + j) \in \mathbb{W}^k$ **then**
                    $\mathbb{W}^{k+1} \leftarrow \mathbb{W}^k \setminus \{i^*\}$; $next \leftarrow 0$
                else
                    $\tilde{\mathbb{I}} \leftarrow \tilde{\mathbb{I}} \setminus \{i^*\}$
            else
                $y^* \leftarrow y^k$; $next \leftarrow 0$; $stop \leftarrow 1$
    $k \leftarrow k + 1$

on

$$\Phi = \{p \in \mathbb{R}^{mn^2} : c_i^T(y^k + p) = c_i^T p = 0,\ i \in \mathbb{W}^k\}.$$

There always exists a minimizer of ϕ on Φ. Indeed, the Hessian matrix A of the function ϕ is a positive semidefinite matrix. Hence, ϕ is a non-strictly

convex quadratic function. Because of the structure of the matrix A and the vector b, the system $\nabla\phi(p) = Ap + (Ay^k - b) = 0$ has at least one solution for all $y^k \in \mathbb{R}^{mn^2}$. Therefore, the quadratic function ϕ is bounded below on $\mathbb{R}^{mn^2}$ [**88**] and it has a minimizer on the set Φ. According to the KKT conditions for an equality constrained QP problem, if p^k is a minimizer of ϕ on Φ, then there is a Lagrange multiplier vector $\lambda^k \in \mathbb{R}^{|\mathbb{W}^k|}$ such that the following system (KKT system) is satisfied:

$$\begin{pmatrix} -A & c_{\mathbb{W}^k} \\ c_{\mathbb{W}^k}^T & 0 \end{pmatrix} \begin{pmatrix} p^k \\ \lambda^k \end{pmatrix} = \begin{pmatrix} Ay^k - b \\ 0 \end{pmatrix}. \tag{2.14}$$

KKT system (2.14) has at least one solution. Thus, we choose any of them (line 12).

Suppose that the step direction p^k is not equal to zero (line 13). In this case, a new feasible point $y^{k+1} = y^k + \alpha^k p^k$ is constructed along this direction (lines 14–18). The step length α^k is defined by the following formula:

$$\alpha^k = \min\left\{1, \min\left\{(-c_i^T y^k)/(c_i^T p^k) : i \notin \mathbb{W}^k \text{ and } c_i^T p^k < 0\right\}\right\}.$$

The definition of the step length ensures that the point y^{k+1} belongs to the feasible set $\mathbb{Y}'$ [**71**]. If the new point touches an inequality constraint that is not active at the point y^k (the index of that constraint does not belong to the set $\mathbb{W}^k$), the index of that constraint is added to the set $\mathbb{W}^{k+1}$ (lines 16–17).

Next, suppose that the step direction p^k is equal to zero (line 19). In this case, we verify whether the point y^k is a minimizer of g on $\mathbb{Y}'$ or not. It is done by checking the KKT dual feasibility condition (1.7d). In other words, on the set $\mathbb{W}^k$ we select indices of those inequality constraints which have negative Lagrange multipliers (lines 20–23). Let $\tilde{\mathbb{I}}$ denote a set of these selected indices. If the set $\tilde{\mathbb{I}}$ is empty, then the Lagrange multipliers corresponding to inequality constraints active at y^k are equal to zero or positive. Hence, the point y^k is a minimizer of g on $\mathbb{Y}'$ and calculations are stopped. If $\tilde{\mathbb{I}}$ is not empty, the set $\mathbb{W}^k$ contains at least one index of an inequality constraint with a negative Lagrange multiplier. In this case, a regular active-set method from the set $\mathbb{W}^k$ removes an index of inequality constraint with the most negative Lagrange multiplier (lines 27, 32). Let us remember that we are considering an optimization problem with a number of complementarity constraints $y_{(mn+i)}y_{(mn+1+i)} = 0$, $i = 1, 3, \ldots, 2s-1$.

Every complementarity constraint i corresponds to one (or both) of these equalities:

$$c^T_{(m+s+i)}y = 0,\ c^T_{(m+s+1+i)}y = 0.$$

Thus, one (or both) of indices $(m+s+i)$ or $(m+s+1+i)$ belongs to the set $\mathbb{W}^k$ (because $y^k \in \mathbb{Y}'$). If both of these indices would be removed from the set $\mathbb{W}^k$, at the next iteration it might be the following situation (but not necessarily): $p^{k+1}_{(mn+i)} \neq 0$, $p^{k+1}_{(mn+1+i)} \neq 0$ and $\alpha^{k+1} > 0$. It is clear that then $y^{k+1}_{(mn+i)}y^{k+1}_{(mn+1+i)} \neq 0$ and $y^{k+1} \notin \mathbb{Y}'$. If after some number of iterations one of the above indices would appear on the set $\mathbb{W}^l$ with some $l > k$, there is no guarantee that the complementarity constraint i would be satisfied again. Hence, in order to preserve all the complementarity constraints at every iteration, we forbid the removal of an index that does not have a pair on the set $\mathbb{W}^k$ (lines 28–34). If the removal of index $(m+s+i)$ or $(m+s+1+i)$ is forbidden, the index is eliminated from the set $\tilde{\mathbb{I}}$. Then the analysis of $\tilde{\mathbb{I}}$ is continued until $\tilde{\mathbb{I}}$ becomes empty (a minimizer of g on $\mathbb{Y}'$ is found) or the removal is allowed (lines 31–32).

2.4. Algorithm, based on the branch-and-bound method

In this section, we propose and describe another algorithm for problem (2.9). In this case, the algorithm is based on the branch-and-bound method (see Section 1.1.2.3) and it returns a global minimizer of g on $\mathbb{Y}'$. Problem (2.9) may be expressed as a two-level optimization problem with a convex QP problem at the lower-level, and a combinatorial optimization problem at the upper-level. We applied the branch-and-bound method, i.e., we defined the branching, bounding, and pruning operations for the upper-level combinatorial optimization problem. Here, first of all, we express problem (2.9) as a two-level optimization problem. Next, we define the branching, bounding, and pruning operations. Finally, we propose and describe the algorithm.

2.4.1. Two-level optimization

Let us show that the feasible set $\mathbb{Y}'$ may be divided into a finite number of particular subsets. Suppose that $t : \mathbb{Y}' \to \{0,1\}^{2s}$ is a 0-1 vector-valued function such that:

$$t_i(y) = \begin{cases} 1, & \text{if } y_{mn+i} > 0, \\ 0, & \text{if } y_{mn+i} = 0, \end{cases}$$

$1 \le i \le 2s$. Let $t(\mathbb{Y}') = \{t(y) : y \in \mathbb{Y}'\}$ denote the range (image) of the function t. Every complementarity constraint $y_{(mn+i)}y_{(mn+1+i)} = 0$ implies that $0 \le t_i(y) + t_{i+1}(y) \le 1$, $i = 1, 3, 5, \ldots, 2s-1$. Thus, the range of t may be defined as follows:

$$t(\mathbb{Y}') = \{z \in \{0,1\}^{2s} : z_i + z_{i+1} \le 1,\ i = 1, 3, 5, \ldots, 2s-1\}.$$

It is clear that the set $t(\mathbb{Y}')$ is a finite set. Let us remember the symbols $\mathbb{E}$ and $\mathbb{I}$, defined in (2.9), and let $\mathbb{I}_{eq}(z) \subset \mathbb{I}$ be a set such that

$$\mathbb{I}_{eq}(z) = \{i \in \mathbb{I} : z_{i-(m+s)} = 0\}, \tag{2.15}$$

where $z \in t(\mathbb{Y}')$. Then $\mathbb{Y}' = \bigcup_{z \in t(\mathbb{Y}')} \mathbb{Y}'(z)$, where

$$\begin{aligned} \mathbb{Y}'(z) = \ & \{y \in \mathbb{Y}' : c_i^T y = 0,\ i \in \mathbb{I}_{eq}(z)\} = \\ & \{y \in \mathbb{R}^{mn^2} : c^T_{\mathbb{E} \cup \mathbb{I}_{eq}(z)} y = 0,\ c^T_{\mathbb{I} \setminus \mathbb{I}_{eq}(z)} y \ge 0\}. \end{aligned}$$

Therefore, the feasible set of problem (2.9) may be divided into a finite number of subsets. In addition, it is sufficient to consider those $z \in t(\mathbb{Y}')$ which satisfy equalities $z_i + z_{i+1} = 1$, $i = 1, 3, 5, \ldots, 2s-1$. Indeed, suppose that

$$\begin{aligned} z^1 &= (z^1_1, \ldots, z^1_{(i-1)}, 1, 0, z^1_{(i+2)}, \ldots, z^1_{(2s)})^T, \\ z^2 &= (z^2_1, \ldots, z^2_{(i-1)}, 0, 0, z^2_{(i+2)}, \ldots, z^2_{(2s)})^T \text{ and} \\ z^3 &= (z^3_1, \ldots, z^3_{(i-1)}, 0, 1, z^3_{(i+2)}, \ldots, z^3_{(2s)})^T \in t(\mathbb{Y}'), \end{aligned}$$

where $z^1_j = z^2_j = z^3_j$, $j \in \{1, \ldots, 2s\} \setminus \{i, i+1\}$. From the definition of the set $\mathbb{Y}'(z)$, it follows that $\mathbb{Y}'(z^2) \subset \mathbb{Y}'(z^1)$ and $\mathbb{Y}'(z^2) \subset \mathbb{Y}'(z^3)$. Thus, the feasible set $\mathbb{Y}'$ may be defined as follows: $\mathbb{Y}' = \bigcup_{z \in \mathbb{Z}} \mathbb{Y}'(z)$, where

$$\mathbb{Z} = \{z \in \{0,1\}^{2s} : z_i + z_{i+1} = 1,\ i = 1, 3, 5, \ldots, 2s-1\}.$$

Also note that if $N = s/m = n(n-1)/2$ and $y = (x^T, d^{\pm T})^T \in \mathbb{Y}'(z)$, $z \in \mathbb{Z}$, then:

$$\begin{aligned} \textstyle\sum_{i=1}^n x_{ki} = 0, \quad & x_{ki} - x_{kj} = d^+_{kij} - d^-_{kij}, \\ (1 - z_{2k-1}) d^+_{k12} = 0, \quad & (1 - z_{2k}) d^-_{k12} = 0, \\ (1 - z_{2(m+k)-1}) d^+_{k13} = 0, \quad & (1 - z_{2(m+k)}) d^-_{k13} = 0, \\ & \ldots \\ (1 - z_{2Nk-1}) d^+_{k(n-1)n} = 0, \quad & (1 - z_{2Nk}) d^-_{k(n-1)n} = 0, \\ d^+_{kij} \ge 0, \quad & d^-_{kij} \ge 0, \end{aligned} \tag{2.16}$$

for all $1 \leq i < j \leq n$, $1 \leq k \leq m$.

Therefore, problem (2.9) may be expressed as the following two-level optimization problem:

$$\underset{z \in \mathbb{Z},\, y^* \in \mathbb{Y}'^*_g(z)}{\text{minimize}} \quad g(y^*), \tag{2.17}$$

where $\mathbb{Y}'^*_g(z) \subset \mathbb{Y}'(z)$ denotes a set of all global minimizers of the problem

$$\underset{y \in \mathbb{Y}'(z)}{\text{minimize}} \quad g(y). \tag{2.18}$$

Upper-level problem (2.17) is a combinatorial optimization problem, and lower-level problem (2.18) is a convex (non-strictly) QP problem. Note that if $z_i,\ z_{i+1} \in \{0, 1\}$, then $z_i + z_{i+1} = 1$, if and only if $(z_i, z_{i+1}) \in \{(0, 1),\ (1, 0)\}$. Therefore, the number of the lower-level problems is equal to $|\mathbb{Z}| = 2^s$.

2.4.2. Operations

We applied the branch-and-bound method for upper-level combinatorial optimization problem (2.17). Let us show how we implemented the following operations: 1) the division of the upper-level feasible set $\mathbb{Z}$ (branching), 2) the evaluation of the minimum value of the upper-level objective function g on every feasible subset (bounding), and 3) the elimination of feasible subsets (pruning).

2.4.2.1. Branching. The upper-level feasible set $\mathbb{Z}$ is divided as follows. Let $\mathbb{V} \subset \{0, 1\}^{2s}$ be a set such that $\mathbb{V} = \mathbb{V}(0) \cup \mathbb{V}(1) \cup \mathbb{V}(2) \cup \ldots \cup \mathbb{V}(s)$, and

$$\begin{aligned} \mathbb{V}(k) = \{v \in \{0, 1\}^{2s} :\ & v_{2i-1} + v_{2i} = 1,\ 1 \leq i \leq k, \\ & v_{2i-1} + v_{2i} = 2,\ k+1 \leq i \leq s\}, \end{aligned}$$

for all $0 \leq k \leq s$. Then, the upper-level feasible set is divided into the following subsets:

$$\begin{aligned} \mathbb{Z}(v) = \{z \in \{0, 1\}^{2s} :\ & z_i = v_i,\ 1 \leq i \leq 2k, \\ & z_{2i-1} + z_{2i} = 1,\ k+1 \leq i \leq s\}, \end{aligned} \tag{2.19}$$

where $v \in \mathbb{V}(k)$, $k \in \{0, 1, \ldots, s\}$. It is not hard to check that:

(1) $|\mathbb{V}(k)| = 2^k$.

(2) $|\mathbb{V}| = \sum_{k=0}^{s} |\mathbb{V}(k)| = \sum_{k=0}^{s} 2^k = 2^{s+1} - 1$.
(3) $|\mathbb{Z}(v)| = 2^{s-k}$, $v \in \mathbb{V}(k)$.
(4) $\mathbb{V}(s) = \mathbb{Z}(v) = \mathbb{Z}$, $v \in \mathbb{V}(0)$.
(5) $\bigcup_{i=1}^{2^k} \mathbb{Z}(v_i) = \mathbb{Z}$, $v_i \in \mathbb{V}(k)$, $1 \le i \le 2^k$.
(6) $\bigcap_{i=1}^{2^k} \mathbb{Z}(v_i) = \emptyset$, $v_i \in \mathbb{V}(k)$, $1 \le i \le 2^k$.

The set $\mathbb{V}$ may be depicted as a tree. Every node of the tree corresponds to a certain element of $\mathbb{V}$ and, in turn, to a certain subset of $\mathbb{Z}$. The tree is often called the search tree. Figure 2.5 contains a search tree for problem (2.17). Here, every node

$$\begin{array}{l} \overbrace{v_1 \dots v_{2k-1} 1 \dots 11}^{s} \\ \underbrace{v_2 \dots v_{2k}}_{k} \quad 1 \dots 11 \end{array}$$

corresponds to a particular vector $v \in \mathbb{V}(k)$, $k \in \{0, 1, \dots, s\}$, which, in turn, corresponds to a certain subset $\mathbb{Z}(v) \subseteq \mathbb{Z}$.

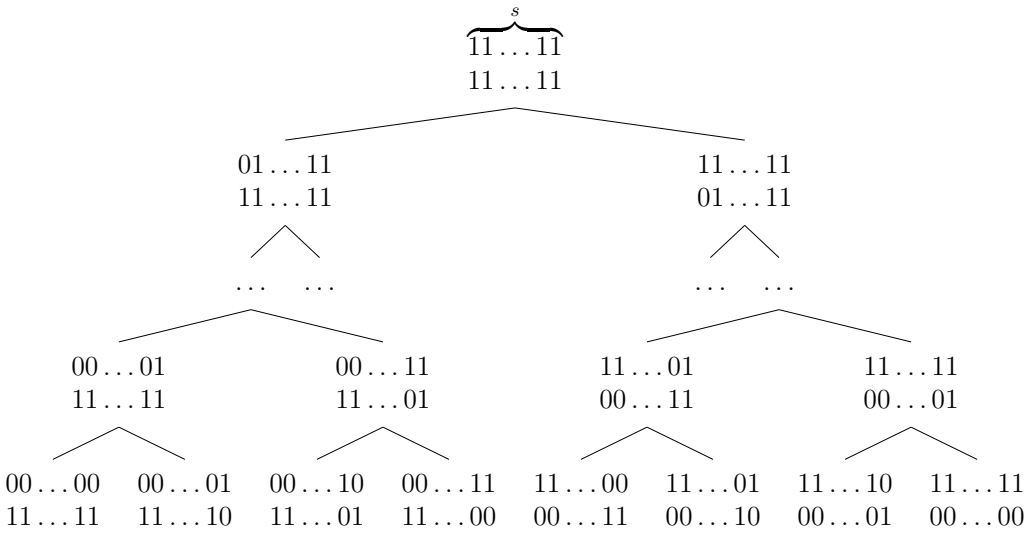


FIGURE 2.5. The search tree for problem (2.17).

2.4.2.2. Bounding. The lower bound for the minimum value of the objective function g on the subset $\mathbb{Z}(v)$, $v \in \mathbb{V}$, is calculated as follows. The minimum value of g on $\mathbb{Z}(v)$ is equal to $g(y^*)$, where

$$y^* \in \operatorname{argmin}\{g(y) : y \in \bigcup_{z \in \mathbb{Z}(v)} \mathbb{Y}'(z)\}.$$

Note that $s = |\mathbb{I}_{eq}(z)| \ge |\mathbb{I}_{eq}(v)| = k$ whenever $z \in \mathbb{Z}(v)$ and $v \in \mathbb{V}(k)$, $k \in \{0, 1, \dots, s\}$ (the set $\mathbb{I}_{eq}$ is defined by (2.15)). Thus, the set $\mathbb{Y}'(v)$ is not

smaller than the set $\bigcup_{z\in\mathbb{Z}(v)}\mathbb{Y}'(z)$, i.e., $\bigcup_{z\in\mathbb{Z}(v)}\mathbb{Y}'(z)\subseteq\mathbb{Y}'(v)$ for all $v\in\mathbb{V}$. Therefore, $g(y^{**})\leq g(y^{*})$, where

$$y^{**}\in\operatorname{argmin}\{g(y):y\in\mathbb{Y}'(v)\}.$$

The objective value at the point y^{**} – i.e., the value of $g(y^{**})$ – is used here as the lower bound for the minimum value of the function g on the subset $\mathbb{Z}(v)$, $v\in\mathbb{V}$.

2.4.2.3. Pruning. We do not have to solve lower-level problems (2.18) for all $z\in\mathbb{Z}$. We may eliminate some of them. Let us define three rules of the elimination of z from $\mathbb{Z}$.

Rule I. This rule is based on the fact that the function f_1 is invariant under mirroring when the mirrored points are obtained by exchanging directions of coordinate axes in the m-dimensional Cartesian coordinate system (see Section 2.1.3). Let us show that it is sufficient to consider those $z\in\mathbb{Z}$ which belong to the subset $\mathbb{Z}(v)$ with a certain $v\in\mathbb{V}(m)$.

Suppose that $v^1\in\mathbb{V}(m)$, $z^1\in\mathbb{Z}(v^1)$ and $y^1=({x^1}^T,{d^{\pm}}^{1^T})^T\in\mathbb{Y}'(z^1)$. Let $y^2=({x^2}^T,{d^{\pm}}^{2^T})^T$ be an element of the set $\mathbb{R}^{mn^2}$ such that

$$\begin{array}{l}x^2_{k1}=-x^1_{k1},\ x^2_{k2}=-x^1_{k2},\ x^2_{ki}=x^1_{kj}\text{ and}\\ d^{+\,2}_{k12}=d^{-\,1}_{k12},\ d^{-\,2}_{k12}=d^{+\,1}_{k12},\ d^{+2}_{kij}=d^{+1}_{kij},\ d^{-2}_{kij}=d^{-1}_{kij}\end{array}$$

for all $1\leq i<j\leq n$ $(j\neq 2)$, $1\leq k\leq m$. Note that $|x^1_{k1}-x^1_{k2}|=|(-x^1_{k1})-(-x^1_{k2})|=|x^2_{k1}-x^2_{k2}|$ for all $1\leq k\leq m$. Thus, $f_1(x^1)=f_1(x^2)$ and, consequently, $g(y^1)=g(y^2)$. Let z^2 be an element of the set $\mathbb{Z}$ such that

$$\begin{array}{l}z^2_{2i-1}=z^1_{2i},\ z^2_{2i}=z^1_{2i-1},\ 1\leq i\leq m,\\ z^2_i=z^1_i,\ 2m+1\leq i\leq 2s.\end{array}$$

It is not hard to check that $y^2\in\mathbb{Y}'(z^2)$ and $z^2\in\mathbb{Z}(v^2)$ with a certain $v^2\in\mathbb{V}(m)$ $(v^1\neq v^2)$. Therefore, for every $v^1\in\mathbb{V}(m)$, $z^1\in\mathbb{Z}(v^1)$ and $y^1\in\mathbb{Y}'(z^1)$ there exist particular $v^2\in\mathbb{V}(m)$, $z^2\in\mathbb{Z}(v^2)$ and $y^2\in\mathbb{Y}'(z^2)$ such that $g(y^1)=g(y^2)$. Thus, it is sufficient to consider those $z\in\mathbb{Z}$ which belong to the subset $\mathbb{Z}(v)$, $v\in\mathbb{V}(m)$. In addition, the set $\mathbb{Z}(v)$, $v\in\mathbb{V}(m)$, contains 2^{s-m} elements. Thus, in applying this rule we may reduce the search space 2^m times.

Rule II. Suppose that $\mu:\mathbb{N}^3\to-\mathbb{N}\cup\mathbb{N}$ is a function defined as follows:

$$\mu(k,i,j)=2(k+m(j-i+(2n-i)(i-1)/2-1))-1. \tag{2.20}$$

Note that

$$\begin{array}{llll}
\mu(1,1,2)=1, & \ldots, & \mu(m,1,2)=2m-1,\\
\mu(1,1,3)=2(m+1)-1, & \ldots, & \mu(m,1,3)=2(2m)-1,\\
 & \ldots & \\
\mu(1,n-1,n)=2(s-(m-1))-1, & \ldots, & \mu(m,n-1,n)=2s-1.
\end{array}$$

Let us show that it is sufficient to consider those $z \in \mathbb{Z}$ which satisfy the following condition:

$$\begin{array}{ll}
(z_{\mu(k,i,l)}, z_{\mu(k,l,j)}, z_{\mu(k,i,j)}) \in & \{(0,0,0),(0,1,0),(0,1,1),\\
 & (1,0,0),(1,0,1),(1,1,1)\}
\end{array} \tag{2.21}$$

for all $k \in \{1,\ldots,m\}$, i, l, $j \in \{1,\ldots,n\}$ $(i<l<j)$.

Let $z^1 \in \mathbb{Z}$ and

$$(z^1_{\mu(k,i,l)}, z^1_{\mu(k,l,j)}, z^1_{\mu(k,i,j)}) \in \{(0,0,1),(1,1,0)\} \tag{2.22}$$

with certain $k \in \{1,\ldots,m\}$ and i, l, $j \in \{1,\ldots,n\}$ $(i<l<j)$. Suppose that $y^1 = ({x^1}^T, {d^{\pm 1}}^T)^T \in \mathbb{Y}'(z^1)$. Then

$$\begin{array}{ll}
d^{+1}_{kil} = d^{+1}_{klj} = d^{-1}_{kij} = 0, & \text{if } (z^1_{\mu(k,i,l)}, z^1_{\mu(k,l,j)}, z^1_{\mu(k,i,j)}) = (0,0,1) \text{ and}\\
d^{-1}_{kil} = d^{-1}_{klj} = d^{+1}_{kij} = 0, & \text{if } (z^1_{\mu(k,i,l)}, z^1_{\mu(k,l,j)}, z^1_{\mu(k,i,j)}) = (1,1,0).
\end{array}$$

Let us consider the following system of linear equations:

$$\begin{array}{l}
x^1_{ki} - x^1_{kl} = d^{+1}_{kil} - d^{-1}_{kil}\\
x^1_{kl} - x^1_{kj} = d^{+1}_{klj} - d^{-1}_{klj}\\
x^1_{ki} - x^1_{kj} = d^{+1}_{kij} - d^{-1}_{kij}.
\end{array} \tag{2.23}$$

It is not hard to check that

$$0 = d^{+1}_{kil} - d^{-1}_{kil} + d^{+1}_{klj} - d^{-1}_{klj} - d^{+1}_{kij} + d^{-1}_{kij}$$

is a linear combination of equations of system (2.23). Thus, if $d^{+1}_{kil} = d^{+1}_{klj} = d^{-1}_{kij} = 0$, then $d^{-1}_{kil} = d^{-1}_{klj} = d^{+1}_{kij} = 0$, and vice versa. Let z^2 be an element of the set $\mathbb{Z}$ such that it satisfies condition (2.21) and $z^2_q = z^1_q$, where

$$\begin{array}{ll}
q \in \{1,\ldots,2s\} \setminus & \{\mu(k,i,l), \mu(k,l,j), \mu(k,i,j),\\
 & \mu(k,i,l)+1, \mu(k,l,j)+1, \mu(k,i,j)+1\}.
\end{array}$$

According to the definition of the set $\mathbb{Y}'(z)$, $z \in \mathbb{Z}$, $y^1 \in \mathbb{Y}'(z^2)$ and, consequently, $\mathbb{Y}'(z^1) \subset \mathbb{Y}'(z^2)$. Thus, it is sufficient to consider those $z \in \mathbb{Z}$ which satisfy condition (2.21) or, vice versa, we may eliminate all $z \in \mathbb{Z}$ which satisfy condition (2.22).

Rule III. This rule is based on the fact that the function f_1 is invariant under mirroring when the mirrored points are obtained by exchanging coordinate axes in the m-dimensional Cartesian coordinate system (see Section 2.1.2). Let us remember that $N = n(n-1)/2$. For every $z \in \mathbb{Z}$, let us define a vector $w(z) = (w_1(z), \ldots, w_m(z))^T \in \{0, 1, \ldots, 2^N - 1\}^m$ such that

$$w_k(z) = \sum_{i=1}^{N} 2^{N-i} z_{2(k+(i-1)m)-1},$$

$1 \leq k \leq m$. Vector $w(z)$ is unique for every $z \in \mathbb{Z}$, i.e., if z^1, $z^2 \in \mathbb{Z}$ $(z^1 \neq z^2)$, then $w(z^1) \neq w(z^2)$. Let us show that it is sufficient to consider those $z \in \mathbb{Z}$ which satisfy the following inequalities:

$$w_1(z) \geq \ldots \geq w_m(z). \tag{2.24}$$

First of all, let us explain the meaning of numbers $w_k(z)$, $1 \leq k \leq m$. Suppose that elements of vector $z = (z_1, z_2, \ldots, z_{2s})^T \in \mathbb{Z}$ are written in a particular order, presented in Table 2.1. Because of $z_i \in \{0, 1\}$, $1 \leq i \leq 2s$, the first row (the second row, too) of the table may be perceived as m binary numbers with N-digits (N-bits). Thus, the number $w_k(z)$ denotes a decimal number, composed of the k-th binary number, written in the first row of Table 2.1.

Next, let $w_{k_1}(z), \ldots, w_{k_m}(z)$ denote components of the vector $w(z)$, written in decreasing order, i.e., $w_{k_1}(z) \geq \ldots \geq w_{k_m}(z)$. Let $w_{\downarrow}(z) = (w_{k_1}(z), \ldots, w_{k_m}(z))^T$. Suppose that z^1, $z^2 \in \mathbb{Z}$ $(z^1 \neq z^2)$ and $w_{\downarrow}(z^1) = w_{\downarrow}(z^2)$. According to the 1) meaning of the numbers $w_k(z)$, $1 \leq k \leq m$, 2) relationship between $z \in \mathbb{Z}$ and $y \in \mathbb{Y}'(z)$ (see (2.16)), and 3) equalities (2.2), (2.7), for every $y^1 \in \mathbb{Y}'(z^1)$ there exists a certain $y^2 \in \mathbb{Y}'(z^2)$ such that $g(y^1) = g(y^2)$ and vice versa. Therefore, it is sufficient to consider those $z \in \mathbb{Z}$ which satisfy inequalities (2.24). In addition, if $m = 1$, then there are $2^N = 2^s$ different vectors $w(z)$, which satisfy inequalities (2.24). If $m = 2$, the number of such vectors is equal to $\sum_{i=1}^{2^N} i = 2^N(1 + 2^N)/2$. If $m = 3$, we have $\sum_{i=1}^{2^N} i(1+i)/2 = 2^N(2^N + 1)(2^N + 2)/6$ such different vectors.

TABLE 2.1. Elements of vector $z = (z_1, \ldots, z_{2s})^T \in \mathbb{Z}$, written in the following order: $z_{2k-1}, \ldots, z_{2(k+(N-1)m)-1}$ (the first row) and $z_{2k}, \ldots, z_{2(k+(N-1)m)}$ (the second row) for all $1 \le k \le m$.

$k=1$				$2 \le k \le m-1$			$k=m$		
z_1	z_{2m+1}	$\cdots$	$z_{2m(N-1)+1}$	z_3	$\cdots$	z_{2s-3}	z_{2m-1}	$\cdots$	z_{2s-1}
z_2	z_{2m+2}	$\cdots$	$z_{2m(N-1)+2}$	z_4	$\cdots$	z_{2s-2}	z_{2m}	$\cdots$	z_{2s}

2.4.3. Algorithm

Here, we propose and describe an algorithm that returns a global minimizer of g on $\mathbb{Y}' = \bigcup_{z \in \mathbb{Z}} \mathbb{Y}'(z)$. The algorithm is based on the branch-and-bound method. Thus, we called it "Branch-and-Bound" (BB). A pseudocode of the algorithm is presented in Algorithm 2.2. Let us describe the main steps of the algorithm BB.

Lines 1–3 of Algorithm 2.2 define the part of initialization. Based on the first pruning rule (see Section 2.4.2.3, Rule I), it is sufficient to look for a global minimizer of the function g on the subset $\bigcup_{z \in \mathbb{Z}(v)} \mathbb{Y}'(z) \subset \mathbb{Y}'$, where $v \in \mathbb{V}(m)$, instead of on the whole set $\mathbb{Y}'$. Thus, we select any $v \in \mathbb{V}(m)$ (line 1 of the algorithm). In order to find a minimizer of g on $\bigcup_{z \in \mathbb{Z}(v)} \mathbb{Y}'(z)$, we 1) divide the set $\bigcup_{z \in \mathbb{Z}(v)} \mathbb{Y}'(z)$ into subsets, and 2) evaluate the minimum value of the function g on every subset. More precisely, we divide the set

Algorithm 2.2 Branch-and-Bound (BB)

Input: $m \in \{1, 2, 3\}$, $n \in \mathbb{N}$ $(n > 1)$, $\delta_{ij} \in \mathbb{R}$ $(\delta_{ij} = \delta_{ji} > 0,\ \delta_{ii} = 0)$, $1 \le i, j \le n$
Output: y^* – a global minimizer of g on $\mathbb{Y}'$

1: $v \in \mathbb{V}(m)$
2: $V \leftarrow \{\mathbb{Z}(v)\}$
3: $g^* \leftarrow +\infty$
4: **while** $V \neq \emptyset$ **do**
5: $\quad \mathbb{Z}(\nu) \in V$; $V \leftarrow V \setminus \{\mathbb{Z}(\nu)\}$
6: $\quad i^* \leftarrow \sum_{i=1}^{s} (\nu_{2i-1} + \nu_{2i})$
7: $\quad$ **for** $i = 1, 2$ **do**
8: $\quad\quad \nu^i \leftarrow \nu$; $\nu^i_{(2i^*+i)} \leftarrow 0$
9: $\quad\quad$ Remove all unnecessary elements from the subset $\mathbb{Z}(\nu^i)$
10: $\quad\quad$ **if** $\mathbb{Z}(\nu^i) \neq \emptyset$ **then**
11: $\quad\quad\quad y^{**} \in \operatorname{argmin}\{g(y) : y \in \mathbb{Y}'(\nu^i)\}$
12: $\quad\quad\quad$ **if** $g(y^{**}) < g^*$ **then**
13: $\quad\quad\quad\quad$ **if** $y^{**} \in \mathbb{Y}'$ **then**
14: $\quad\quad\quad\quad\quad y^* \leftarrow y^{**}$; $g^* \leftarrow g(y^{**})$
15: $\quad\quad\quad\quad$ **else**
16: $\quad\quad\quad\quad\quad V \leftarrow V \cup \{\mathbb{Z}(\nu^i)\}$

$\mathbb{Z}(v)$. The subsets of $\mathbb{Z}(v)$ are stored in the set V (line 2). The initial objective minimum value g^* is set to the positive infinity (line 3).

Lines 4–16 of the algorithm define the part of calculations. Here, a subset $\mathbb{Z}(\nu)$ of the set $\mathbb{Z}(v)$ is selected for further processing and it is removed from the set V (line 5). Note that $\nu \in \mathbb{V}(k)$, where $k \in \{m, \ldots, s-1\}$ and $\nu_i = v_i$, $1 \le i \le 2m$. There are a few strategies to select the subset [**16**]. In numerical experiments, we used a strategy called depth-first-search: the last added subset is selected. Next, the selected subset $\mathbb{Z}(\nu)$ is divided into two disjoint parts $\mathbb{Z}(\nu^1)$ and $\mathbb{Z}(\nu^2)$ (lines 6–8). Note that ν^1, $\nu^2 \in \mathbb{V}(i^*+1)$, where $i^* = \sum_{i=1}^{s} (\nu_{2i-1} + \nu_{2i})$. Based on the second and the third pruning rules (see Section 2.4.2.3, Rule II, Rule III), we remove unnecessary elements from the subset $\mathbb{Z}(\nu^i)$, $i \in \{1, 2\}$ (line 9). Actually, either all or no z from $\mathbb{Z}(\nu^i)$ are removed. Thus, if the subset $\mathbb{Z}(\nu^i)$ is not empty, the lower bound for the minimum value of g on $\bigcup_{z \in \mathbb{Z}(\nu^i)} \mathbb{Y}'(z)$ is found by solving the following convex QP problem (line 11):

$$\begin{array}{ll} \text{minimize} \\ y \in \mathbb{Y}'(\nu^i) \end{array} \quad g(y). \tag{2.25}$$

In numerical experiments, the active-set method was used to solve problem (2.25). Let y^{**} be a minimizer of g on $\mathbb{Y}'(\nu^i)$. Suppose that $g(y^{**})$ is smaller than the current objective minimum value g^* (line 12). If y^{**} is a feasible point (line 13), then y^{**} is accepted as a current minimizer and the current objective minimum value is updated (line 14). If $y^{**} \notin \mathbb{Y}'$ (line 15), then there is at least one index, say $j^* \in \{i^*+1, \ldots, s\}$, such that $y^{**}_{mn+2j^*-1} > 0$ and $y^{**}_{mn+2j^*} > 0$. In this case, the subset $\mathbb{Z}(\nu^i)$ has to be divided further (line 16). If $g(y^{**}) \ge g^*$, the division of the subset $\mathbb{Z}(\nu^i)$ is meaningless, because we will never find a better minimizer of g on $\bigcup_{z \in \mathbb{Z}(\nu^i)} \mathbb{Y}'(z)$ than we have at the moment.

2.5. Algorithm, based on a parallel branch-and-bound method

In this section, the third algorithm for problem (2.9) is presented and described. The algorithm is a parallel version of the Algorithm 2.2 and it has the parallelism of type 1 (see Section 1.1.2.4). It returns a global minimizer of g on $\mathbb{Y}'$. Suppose that a certain computer may perform $P = 2^l + 1$ ($l = 0, 1, 2, \ldots$) processes simultaneously, i.e., it has P cores (processing units). In performing the proposed algorithm on such a computer, the set $\mathbb{Z}(v)$, $v \in \mathbb{V}(m)$, is divided into $P-1$ disjoint subsets $\mathbb{Z}(v^i)$, where $v^i \in \mathbb{V}(m+l)$,

$1 \leq i \leq P-1$, and $v_j^1 = \ldots = v_j^{P-1}$, $1 \leq j \leq 2m$. Then the i-th process finds a minimizer of g on $\bigcup_{z \in \mathbb{Z}(v^i)} \mathbb{Y}'(z)$ by using Algorithm 2.2 and returns it to a particular process, called the master process, $1 \leq i \leq P-1$. Finally, the master process compares $P-1$ minimizers and returns the best of them. Thus, such a number of processes helps us to ensure the equal distribution of the set $\mathbb{Z}(v)$, $v \in \mathbb{V}(m)$, – i.e., $|\mathbb{Z}(v^i)| = 2^{s-m-l}$ for all $1 \leq i \leq P-1$ – and to implement the master-slave model of communication between processes [**77**].

A pseudocode of the proposed algorithm is shown in Algorithm 2.3. The algorithm is based on a parallel branch-and-bound method. Thus, we called it PBB. Let us describe the basic steps of the algorithm PBB.

The total number of available processes P and the number of process R ($R \in \{0, 1, \ldots, P-1\}$) are received (line 1 of Algorithm 2.3). Suppose that $R > 0$ (lines 16–41). Every process $R > 0$ defines a unique vector $v \in \mathbb{V}(m+l)$ (lines 16–22). Then a minimizer of g on $\bigcup_{z \in \mathbb{Z}(v)} \mathbb{Y}'(z)$ is found by using the steps of Algorithm 2.2 and messages from the master process 0 (lines 23–41). If a minimizer Y^* is found, it is sent to the process 0 (lines 37, 38, 41). Together with Y^*, a message – defined in variable N – is sent, too. The message indicates whether the minimizer Y^* is still a current ($N = 1$, line 38) or the final ($N = 0$, line 41) minimizer.

Suppose that $R = 0$ (lines 4–14). The master process 0 receives current and final minimizers from other processes (lines 6, 7), finds currently the best minimizer (lines 10, 11), and sends it to other processes (lines 12–14). When process $R > 0$ receives currently the best minimizer (line 32), it tries to update its own current minimizer (lines 33, 34). Variable $W \in \{0, 1\}^{P-1}$ (line 4) helps to control the process of sending and receiving messages.

Algorithm 2.3 Parallel Branch-and-Bound (PBB)

Input: $m \in \{1, 2, 3\}$, $n \in \mathbb{N}$ $(n > 1)$, $\delta_{ij} \in \mathbb{R}$ $(\delta_{ij} = \delta_{ji} > 0,\ \delta_{ii} = 0)$, $1 \leq i, j \leq n$
Output: y^* – a global minimizer of g on $\mathbb{Y}'$

RECEIVE P, R
$g^* \leftarrow +\infty$
if $R = 0$ **then**
    $W \leftarrow 1 \in \{0, 1\}^{P-1}$
    while $\sum_{i=1}^{P-1} W_i > 0$ **do**
        if THERE IS A MESSAGE FROM PROCESS i **then**
            RECEIVE $[Y^*, N]$
            if $N = 0$ **then**
                $W_i \leftarrow 0$
            if $g(Y^*) < g^*$ **then**
                $y^* \leftarrow Y^*$; $g^* \leftarrow g(Y^*)$
                for $j = 1, P - 1$ **do**
                    if $W_j > 0$ **then**
                        SEND $[y^*]$ TO PROCESS j
else
    $v \in \mathbb{V}(m)$; $l \leftarrow \log(P - 1)$; $j \leftarrow R - 1$
    for $i = m + 1, m + l$ **do**
        if $\mathrm{mod}(j, 2) = 1$ **then**
            $v_{2i-1} \leftarrow 0$
        else
            $v_{2i} \leftarrow 0$
        $j \leftarrow \mathrm{div}(i, 2)$
    $V \leftarrow \{\mathbb{Z}(v)\}$
    while $V \neq \emptyset$ **do**
        $\mathbb{Z}(\nu) \in V$; $V \leftarrow V \setminus \{\mathbb{Z}(\nu)\}$; $i^* \leftarrow \sum_{i=1}^{s} (\nu_{2i-1} + \nu_{2i})$
        for $i = 1, 2$ **do**
            $\nu^i \leftarrow \nu$; $\nu^i_{(2i^*+i)} \leftarrow 0$
            Remove all unnecessary elements from the subset $\mathbb{Z}(\nu^i)$
            if $\mathbb{Z}(\nu^i) \neq \emptyset$ **then**
                $y^{**} \in \mathrm{argmin}\{g(y) : y \in \mathbb{Y}'(\nu^i)\}$
                if THERE IS A MESSAGE FROM PROCESS 0 **then**
                    RECEIVE $[y^*]$
                    if $g(y^*) < g(y^{**})$ **then**
                        $Y^* \leftarrow y^*$; $g^* \leftarrow g(y^*)$
                if $g(y^{**}) < g^*$ **then**
                    if $y^{**} \in \mathbb{Y}'$ **then**
                        $Y^* \leftarrow y^{**}$; $g^* \leftarrow g(y^{**})$
                        $N \leftarrow 1$; SEND $[Y^*, N]$ TO PROCESS 0
                    else
                        $V \leftarrow V \cup \{\mathbb{Z}(\nu^i)\}$
    $N \leftarrow 0$; SEND $[Y^*, N]$ TO PROCESS 0

3. Numerical investigation of the algorithms

In order to reveal the advantages and disadvantages of the proposed algorithms, we implemented them and performed a set of numerical experiments. All algorithms were implemented in Fortran 95. The numerical experiments were performed using dissimilarity matrices, described in Appendix A. In this chapter, we give a few details about the implementation of the proposed algorithms. Also, here we present a description and the results of the numerical experiments. Certain parts of this chapter may be found in [**FGŽ14, GŽ14, GŽ13**].

3.1. The algorithm MAS

In this section, we give a few notes about the implementation of Algorithm 2.1. A description and the results of a certain numerical investigation of the algorithm MAS are presented here, too. At present, the algorithm SMOOTH (see Section 1.2.2.1) is one of the best algorithms for finding a local minimizer of the function f_1 on the set $\mathbb{R}^{mn}$. Thus, results of the numerical investigation of the algorithm MAS were compared with corresponding results obtained by applying the algorithm SMOOTH.

3.1.1. Implementation of MAS

Here, we discuss the following questions: how to select an initial feasible point, i.e., point $y^0 \in \mathbb{Y}'$ (line 1 of Algorithm 2.1), and how to solve KKT system (2.14) (line 12 of Algorithm 2.1).

One of the techniques to select an initial point $y^0 \in \mathbb{Y}'$ is shown in Algorithm 3.1. In this algorithm, a mn-vector of uniformly distributed random numbers between u and v ($u, v \in \mathbb{R}$, $u < v$) is picked (lines 1–3 of Algorithm 3.1). Then, based on this mn-vector, a point $y \in \mathbb{R}^{mn^2}$ is constructed such that:

$$\sum_{i=1}^{n} y_{(k+(i-1)m)} = 0, \ 1 \leq k \leq m, \tag{3.1}$$

and

$$\begin{aligned} y_{(k+(i-1)m)} - y_{(k+(j-1)m)} &= y_{\kappa(k,i,j)} - y_{\kappa(k,i,j)+1}, \\ y_{\kappa(k,i,j)} y_{\kappa(k,i,j)+1} &= 0, \\ y_{\kappa(k,i,j)}, y_{\kappa(k,i,j)+1} &\geq 0, \end{aligned} \tag{3.2}$$

where $\kappa(k,i,j) = mn + \mu(k,i,j)$ and $1 \leq i < j \leq n$, $1 \leq k \leq m$. The function μ is defined in (2.20). The steps of Algorithm 3.1, defined in lines 4–7, ensure that the point y satisfies equalities (3.1). The steps, defined in lines 8–17, ensure that the point y satisfies equalities and inequalities (3.2). Thus, in this way the constructed point y belongs to the feasible set $\mathbb{Y}'$.

Every KKT system (2.14) was solved by using the null-space method, described in Section 1.1.2.2. Let us remember that in applying this method, two particular matrices have to be selected. One of the matrices is a basis matrix of the null-space of $c_{\mathbb{W}^k}^T$. In order to select these matrices, we used the QR decomposition of $c_{\mathbb{W}^k}$. Suppose that $c_{\mathbb{W}^k} = QR$, where $Q \in \mathbb{R}^{mn^2 \times mn^2}$, $R \in \mathbb{R}^{mn^2 \times |\mathbb{W}^k|}$. It is not hard to check that, if $Q = (Q_1 \; Q_2)$, where $Q_1 \in \mathbb{R}^{mn^2 \times |\mathbb{W}^k|}$, $Q_2 \in \mathbb{R}^{mn^2 \times (mn^2 - |\mathbb{W}^k|)}$, then $Q_2^T c_{\mathbb{W}^k} = 0$, i.e., matrix Q_2 is a basis matrix of the null-space of $c_{\mathbb{W}^k}^T$. When the basis matrix is selected, the reduced KKT system (to find a step-direction p^k) and another system of linear equations (to find a vector of Lagrange multipliers λ^k) are constructed. The reduced KKT system was solved by using a LAPACK

Algorithm 3.1 A point selection on the set $\mathbb{Y}'$

Input: $m \in \{1,2,3\}$, $n \in \mathbb{N}$ $(n > 1)$
Output: $y \in \mathbb{Y}'$

1: $r \leftarrow 0 \in \mathbb{R}^{mn}$
2: **for** $i = 1, mn$ **do**
3: $\quad r_i \leftarrow$ a random number, uniformly distributed over interval $[u, v] \subset \mathbb{R}$
4: **for** $k = 1, m$ **do**
5: $\quad q \leftarrow \frac{1}{n} \sum_{j=1}^{n} r_{(k+(j-1)m)}$
6: $\quad$ **for** $i = 1, n$ **do**
7: $\quad\quad y_{(k+(i-1)m)} \leftarrow r_{(k+(i-1)m)} - q$
8: $l \leftarrow mn + 1$
9: **for** $i = 1, (n-1)$ **do**
10: $\quad$ **for** $j = (i+1), n$ **do**
11: $\quad\quad$ **for** $k = 1, m$ **do**
12: $\quad\quad\quad q \leftarrow y_{(k+(i-1)m)} - y_{(k+(j-1)m)}$
13: $\quad\quad\quad$ **if** $q < 0$ **then**
14: $\quad\quad\quad\quad y_l \leftarrow 0$; $y_{(l+1)} \leftarrow -q$
15: $\quad\quad\quad$ **else**
16: $\quad\quad\quad\quad y_l \leftarrow q$; $y_{(l+1)} \leftarrow 0$
17: $\quad\quad\quad l \leftarrow l + 2$

(version 3.5.0) subroutine for linear least squares problems. Another system was solved by using a LAPACK subroutine for regular systems of linear equations.

Suppose that $|\mathbb{W}^{k+1}| = |\mathbb{W}^k| + 1$ or $|\mathbb{W}^{k+1}| = |\mathbb{W}^k| - 1$, i.e., the set $\mathbb{W}^k$ was updated by adding or removing an index (lines 17, 32 of Algorithm 2.1). In this case, the matrices $c_{\mathbb{W}^{k+1}}$ and $c_{\mathbb{W}^k}$ differ only in one column, i.e., the matrix $c_{\mathbb{W}^{k+1}}$ has one column more or less than the matrix $c_{\mathbb{W}^k}$. Thus, it is not necessary to define the QR decomposition of the matrix $c_{\mathbb{W}^{k+1}}$ from scratch. We may simply update the factors Q and R, where $QR = c_{\mathbb{W}^k}$ [**44**].

3.1.2. Investigation of MAS

Suppose that $m \in \{1, 2, 3\}$, $n \in \mathbb{N}$ $(n > 1)$ and $\Delta^n = (\delta_{ij}) \in \mathbb{R}^{n \times n}$ $(\delta_{ij} = \delta_{ji} > 0,\ \delta_{ii} = 0,\ 1 \leq i, j \leq n)$ are given quantities. For the sake of simplicity, algorithms SMOOTH and MAS let us denote by the symbols $\mathcal{S}$ and $\mathcal{M}$, respectively. Let $E^*_{\mathcal{A}}$ be the value of the function Stress-1, defined in (1.21), at a local minimizer x^* of f_1 on $\mathbb{R}^{mn}$ obtained by applying the algorithm $\mathcal{A} \in \{\mathcal{S}, \mathcal{M}\}$, i.e,

$$E^*_{\mathcal{A}} = f_1^{S1}(x^*) = \sqrt{\frac{f_1(x^*)}{\sum_{i<j}^{n} \delta_{ij}^2}}. \tag{3.3}$$

The value of $E^*_{\mathcal{A}}$ is sometimes referred to as a relative error (or a relative visualization error) of an image, defined by x^* [**26**]. The closer $E^*_{\mathcal{A}}$ is to zero, the better image is obtained. In applying the algorithms $\mathcal{S}$ and $\mathcal{M}$, an initial feasible point has to be selected (line 1 of Algorithm 1.2 and Algorithm 2.1). Consequently, the algorithms often return different local minimizers with the same dissimilarity matrix and the same value of m. Let

$$E_{\mathcal{A}} = \{{E^*_{\mathcal{A}}}^1, \ldots, {E^*_{\mathcal{A}}}^{30}\}$$

be a multiset (a set in which multiplicity is significant) of relative errors of 30 images obtained by applying the algorithm $\mathcal{A} \in \{\mathcal{S}, \mathcal{M}\}$. Such a number of relative errors, i.e., 30 relative errors, is usually used in literature and it is enough to reveal the main advantages and disadvantages of the such types of algorithms. According to the suggestion of the authors of the algorithm $\mathcal{S}$, suppose that ${E^*_{\mathcal{S}}}^i$ is the smallest relative error obtained by executing the

algorithm $\mathcal{S}$ ten times, i.e.,

$$E_{\mathcal{S}}^{*\,i} = \min\{E_{\mathcal{S}}^{*\,1i}, \ldots, E_{\mathcal{S}}^{*\,10i}\},$$

$1 \leq i \leq 30$.

Let $t_{\mathcal{S}}^i$ be the time, in seconds, required to find the i-th element of the multiset $E_{\mathcal{S}}$. Suppose that $E_{\mathcal{M}}^{*\,i}$ is the smallest relative error obtained by executing the algorithm $\mathcal{M}$ at least $t_{\mathcal{S}}^i$ seconds, $1 \leq i \leq 30$. Let $t_{\mathcal{M}}^i$ be the time in seconds and $K_{\mathcal{M}}^i$, the number of local minimizers required to find the i-th element of the multiset $E_{\mathcal{M}}$. Thus

$$E_{\mathcal{M}}^{*\,i} = \min\{E_{\mathcal{M}}^{*\,1i}, \ldots, E_{\mathcal{M}}^{*\,K_{\mathcal{M}}^i i}\} \text{ and } t_{\mathcal{M}}^i = t_{\mathcal{S}}^i + \epsilon,$$

where $\epsilon \geq 0$, $1 \leq i \leq 30$. Note that $K_{\mathcal{S}}^i = 10$ and the value of ϵ may be perceived as the approximate time required to find one local minimizer f_1 on $\mathbb{R}^{mn}$ by applying the algorithm $\mathcal{M}$, $1 \leq i \leq 30$. The main purpose of this experiment was to compare a few measures of the multisets $E_{\mathcal{S}}$ and $E_{\mathcal{M}}$ by using different dissimilarity matrices and different values of m. Namely, we were interested in the following measures: the smallest value ($\min E_{\mathcal{A}}$), the biggest value ($\max E_{\mathcal{A}}$), the arithmetic mean ($\operatorname{mean} E_{\mathcal{A}}$), and the standard deviation ($\operatorname{std} E_{\mathcal{A}}$), where $\mathcal{A} \in \{\mathcal{S}, \mathcal{M}\}$.

The numerical experiment was conducted on a computer with an Intel(R) Core(TM)2 Duo processor, running at 2.40 GHz, and with the Xubuntu 14.04 (64-bit, kernel version 3.13.0-24-generic) operating system (OS). Every element of the multiset $E_{\mathcal{S}}$ was obtained by using the program "SMOOTH" [**40**] (version 0.4, July, 2003, written by Patrick Groenen), that implements Algorithm 1.2. Unfortunately, the program was precompiled on a computer with DOS/Windows OS. Thus, in order to use the program on a computer with Linux OS properly, we used a Windows emulator called Wine (version 1.6). Values of parameters γ and $i_{\max}$ of the algorithm $\mathcal{S}$ were selected as follows: $\gamma = 10^{-8}$, $i_{\max} = 1000$. Such values were recommended by the author of the program "SMOOTH".

Results of the experiment are shown in Table 3.1. Here, $t_{\mathcal{A}}$ denotes the average time, in seconds, and $K_{\mathcal{A}}$, the average number of local minimizers required to find one element of the multiset $E_{\mathcal{A}}$, i.e., $t_{\mathcal{A}} = \sum_{i=1}^{30} t_{\mathcal{A}}^i/30$ and $K_{\mathcal{A}} = \sum_{i=1}^{30} K_{\mathcal{A}}^i/30$, $\mathcal{A} \in \{\mathcal{S}, \mathcal{M}\}$. Let us pay the attention to the values of standard deviations of the multisets $E_{\mathcal{S}}$ and $E_{\mathcal{M}}$. These values are almost equal when dissimilarity matrices Δ_{cube}, Δ_{regs} and Δ_{simp} are used. Note that

TABLE 3.1. Results of the numerical investigation of the algorithm $\mathcal{M}$ and corresponding results obtained by using the algorithm $\mathcal{S}$.

Δ^n	m	$\min E_{\mathcal{A}}$	$\max E_{\mathcal{A}}$	$\text{mean}\, E_{\mathcal{A}}$	$\text{std}\, E_{\mathcal{A}}$	$t_{\mathcal{A}}$	$K_{\mathcal{A}}$	$\mathcal{A}$
Δ^4_{cube}	3	0.0001	0.0012	0.0009	0.0002	0.519	10	$\mathcal{S}$
		0.0000	0.0000	0.0000	0.0000	0.519	975	$\mathcal{M}$
Δ^8_{cube}	3	0.0012	0.0013	0.0013	0.0000	1.449	10	$\mathcal{S}$
		0.0000	0.0000	0.0000	0.0000	1.463	62	$\mathcal{M}$
Δ^7_{regs}	3	0.0945	0.0945	0.0945	0.0000	1.566	10	$\mathcal{S}$
		0.0945	0.0945	0.0945	0.0000	1.571	134	$\mathcal{M}$
Δ^9_{regs}	2	0.2991	0.2991	0.2991	0.0000	1.382	10	$\mathcal{S}$
		0.2991	0.3031	0.2996	0.0011	1.387	184	$\mathcal{M}$
$\Delta^{13}_{\text{regs}}$	1	0.5311	0.5311	0.5311	0.0000	0.857	10	$\mathcal{S}$
		0.5311	0.5311	0.5311	0.0000	0.858	379	$\mathcal{M}$
Δ^7_{simp}	3	0.0015	0.0016	0.0016	0.0000	1.673	10	$\mathcal{S}$
		0.0000	0.0000	0.0000	0.0000	1.679	150	$\mathcal{M}$
Δ^9_{simp}	2	0.2759	0.2759	0.2759	0.0000	0.972	10	$\mathcal{S}$
		0.2759	0.2808	0.2760	0.0009	0.977	88	$\mathcal{M}$
$\Delta^{13}_{\text{simp}}$	1	0.5279	0.5281	0.5279	0.0000	1.203	10	$\mathcal{S}$
		0.5279	0.5279	0.5279	0.0000	1.205	314	$\mathcal{M}$
Δ^9_{hwa}	1	0.0109	0.0109	0.0109	0.0000	0.074	10	$\mathcal{S}$
		0.0107	0.0107	0.0107	0.0000	0.075	55	$\mathcal{M}$
	2	0.0108	0.0110	0.0110	0.0001	0.655	10	$\mathcal{S}$
		0.0000	0.0027	0.0001	0.0005	0.713	40	$\mathcal{M}$
Δ^{12}_{hwa}	1	0.1790	0.1790	0.1790	0.0000	0.211	10	$\mathcal{S}$
		0.1790	0.1871	0.1821	0.0023	0.214	48	$\mathcal{M}$
Δ^8_{ruusk}	1	0.2975	0.2975	0.2975	0.0000	0.201	10	$\mathcal{S}$
		0.2975	0.3292	0.3112	0.0087	0.201	480	$\mathcal{M}$
	2	0.1096	0.1096	0.1096	0.0000	1.130	10	$\mathcal{S}$
		0.1097	0.1306	0.1198	0.0055	1.133	172	$\mathcal{M}$
	3	0.0189	0.0254	0.0214	0.0018	2.386	10	$\mathcal{S}$
		0.0188	0.0411	0.0289	0.0066	2.402	86	$\mathcal{M}$
$\Delta^{20}_{\text{ruusk}}$	2	0.0524	0.0555	0.0546	0.0010	4.074	10	$\mathcal{S}$
		0.0523	0.1322	0.0850	0.0232	4.962	3	$\mathcal{M}$
$\Delta^{12}_{\text{uhlen}}$	1	0.2112	0.2112	0.2112	0.0000	0.199	10	$\mathcal{S}$
		0.2112	0.2251	0.2151	0.0036	0.201	52	$\mathcal{M}$
	2	0.0825	0.0909	0.0874	0.0023	2.407	10	$\mathcal{S}$
		0.0840	0.1248	0.1033	0.0105	2.440	37	$\mathcal{M}$
$\Delta^{10}_{\text{cola}}$	1	0.3645	0.3645	0.3645	0.0000	0.318	10	$\mathcal{S}$
		0.3656	0.3959	0.3762	0.0075	0.319	414	$\mathcal{M}$
	2	0.1679	0.1694	0.1694	0.0003	1.692	10	$\mathcal{S}$
		0.1729	0.2089	0.1837	0.0078	1.702	100	$\mathcal{M}$

such dissimilarity matrices define relationships between multidimensional elements in geometry. However, the values of $\operatorname{std} E_{\mathcal{S}}$ and $\operatorname{std} E_{\mathcal{M}}$ often differ when other dissimilarity matrices are selected. These observations show that the algorithm $\mathcal{M}$ is more sensitive to the dissimilarity matrices than the algorithm $\mathcal{S}$.

The big gap between the values of $\min E_{\mathcal{A}}$ and $\max E_{\mathcal{A}}$ indicates the big sensitivity of the algorithm $\mathcal{A}$ to the choice of an initial feasible point. Based on the information shown in Table 3.1, we may state that the algorithm $\mathcal{M}$ is more sensitive to the selection of an initial feasible point than $\mathcal{S}$. For example, in the case of $\Delta^{12}_{\text{uhlen}}$ and $m = 2$ (in the worst case), the gap between $\min E_{\mathcal{S}}$ and $\max E_{\mathcal{S}}$ is equal to 0.0084. The gap between $\min E_{\mathcal{M}}$ and $\max E_{\mathcal{M}}$ is equal to 0.0408. Hence, the gap generated by $\mathcal{M}$, is about 5 times bigger than the gap generated by $\mathcal{S}$.

Finally, note that $K_{\mathcal{M}} > K_{\mathcal{S}}$ in all cases, except the case of $\Delta^{20}_{\text{ruusk}}$. Thus, the algorithm $\mathcal{M}$ generated more local minimizers than the algorithm $\mathcal{S}$ during approximately the same time. Let us remember that every KKT system (2.14) was solved by applying the null-space method. In applying the null-space method, p^k and λ^k are found by constructing and solving two particular systems of linear equations (see (1.17a) and (1.17b)). In the case of $\Delta^{20}_{\text{ruusk}}$, the system used to find a vector of Lagrange multipliers λ^k has at least 762 linear equations (but no more than 800). Unfortunately, the sequential version of a LAPACK subroutine was used to solve such systems. Thus, in the case of $\Delta^{20}_{\text{ruusk}}$, a local minimizer of f_1 on $\mathbb{R}^{mn}$ was found by applying the algorithm $\mathcal{M}$ approximately 4 times slower than the algorithm $\mathcal{S}$.

3.2. The algorithm BB

Here, we present a few notes about the implementation of Algorithm 2.2. We performed a certain numerical investigation of the algorithm BB. A description and the results of the investigation are given here, too. At present, the algorithm BB2009 (see Section 1.2.2.2) is the only algorithm for problem (1.23), based on the branch-and-bound method. Thus, results of the investigation of the algorithm BB were compared with corresponding results obtained by using the algorithm BB2009.

3.2.1. Implementation of BB

Here, we show how we selected an initial vector $v \in \mathbb{V}(m)$ (line 1 of Algorithm 2.2) and implemented the set V, which contains the subsets of $\mathbb{Z}(v)$.

We selected $v \in \mathbb{V}(m)$ as follows:

$$\begin{aligned} v_{2i-1} &= 0, \quad 1 \le i \le m, \\ v_{2i-1} &= 1, \quad m+1 \le i \le s, \\ v_{2i} &= 1, \quad 1 \le i \le s. \end{aligned}$$

Every subset of the set $\mathbb{Z}(v)$ may be defined uniquely by a vector $\nu \in \mathbb{V}(k) \subset \mathbb{V} \subset \{0,1\}^{2s}$, where $k \in \{m, \ldots, s\}$ and $\nu_i = v_i$, $1 \le i \le 2m$ (see (2.19)). Any vector of the set $\mathbb{V}$ may be easily implemented as a 1-dimensional boolean array with $2s$ elements. Also, let us remember that a subset of $\mathbb{Z}(v)$ has to be selected in the set V (line 5 of the algorithm BB). In the implementation of the algorithm, the last added subset is selected. After the selection, the selected subset is removed. Therefore, the set V does not contain more than $(s - m + 1)$ elements. Thus, we implemented the set V as a boolean 2-dimensional array with $(s - m + 1)$ rows and $2s$ columns. Such an implementation of the set V allows us to implement the selection (line 5), addition (line 16), and subtraction (line 5) operations very easily. Indeed, suppose that $|V| > 0$ and $\mathcal{V}$ denotes the array that implements the set V. If we have to select an element (a subset) of V, we simply select the $|V|$-th row of the array $\mathcal{V}$. If we have to add an element, we add the corresponding 1-dimensional boolean array to the $(|V| + 1)$-th row of $\mathcal{V}$. Finally, if we have to remove an element, we do nothing with the array $\mathcal{V}$: we simply work with the first $|V| - 1$ rows of it.

3.2.2. Investigation of BB

For the sake of simplicity, let us denote the algorithms BB2009 and BB by the symbols $\mathcal{B}_{09}$ and $\mathcal{B}_{14}$, respectively. One of the most expensive steps of the algorithm $\mathcal{B}_{14}$ is the solution of convex QP problem (2.25) (line 11 of Algorithm 2.2). A similar step is performed in the algorithm $\mathcal{B}_{09}$, too (see Section 1.2.2.2, [**94**]). Let $K_{\mathcal{A}}$ denote the number of times the algorithm $\mathcal{A}$ performs this step during its execution, $\mathcal{A} \in \{\mathcal{B}_{09}, \mathcal{B}_{14}\}$. The main purpose of the numerical experiment was to compare the numbers $K_{\mathcal{B}_{09}}$ and $K_{\mathcal{B}_{14}}$ by using various dissimilarity matrices and values of m.

The numerical experiment was conducted on a computer with an Intel(R) Pentium(R) 4 CPU 3.00GHz processor, running at 2.80 GHz, and with the CentOS 6.5 (64-bit) operating system. Results of the experiment are shown in Table 3.2. Here, the symbol E^* denotes the value of the function Stress-1 at a global minimizer x^* of f_1 on $\mathbb{R}^{mn}$ (see (3.3)). Let us pay attention to the fact that in cases of $m = 1$, the numbers $K_{\mathcal{B}_{14}}$ are generally greater than the numbers $K_{\mathcal{B}_{09}}$ (except in cases of Δ^8_{cube}, Δ^9_{hwa}, and Δ^{12}_{hwa}). In cases of $m = 2$ and $m = 3$, almost all the numbers $K_{\mathcal{B}_{14}}$ are lower than the numbers $K_{\mathcal{B}_{14}}$ (except cases of Δ^7_{simp}, $m = 2$, and Δ^8_{ruusk}, $m = 2$). Let us remember that in the algorithm $\mathcal{B}_{14}$, the upper-level feasible set $\mathbb{Z}$ (more precisely, a subset $\mathbb{Z}(v) \subset \mathbb{Z}$, where $v \in \mathbb{V}(m)$) is divided into two disjoint parts (see Section 2.4.3). In the algorithm $\mathcal{B}_{09}$, the corresponding upper-level feasible set (we denoted it by the symbol $\mathbb{T}$, see Section 1.2.2.2) is divided into more than two subsets [**94**]. This is one of the main reasons why the numbers $K_{\mathcal{B}_{14}}$ and $K_{\mathcal{B}_{09}}$ are so different.

3.3. The algorithm PBB

The parallel algorithm PBB (see Algorithm 2.3) was implemented for distributed-memory parallel computers by using Message Passing Interface (MPI) [**43, 86**]. There are various metrics used to measure the performance of a parallel algorithm [**34, 76**]. Two of them, the speed-up and the efficiency of a parallel algorithm, are used very often. Let us describe these metrics.

Suppose that there are two algorithms for a certain problem: sequential and parallel. Let t denote the run-time required to solve the problem on a computer with one core (processing unit) by using the sequential algorithm. Similarly, let t_P denote the run-time required to solve the same problem on a computer with P ($P \in \mathbb{N}$) cores by using the parallel algorithm. The ratio of time t to time t_P is called the speed-up of the parallel algorithm and it is usually denoted by

$$\mathcal{S}_P = \frac{t}{t_P}.$$

The closer $\mathcal{S}_P$ is to P, the better (more accelerated) the parallel algorithm is constructed. The ratio of speed-up $\mathcal{S}_P$ to the number of cores P is called the efficiency of the parallel algorithm and it is usually denoted by

$$\mathcal{E}_P = \frac{\mathcal{S}_P}{P}.$$

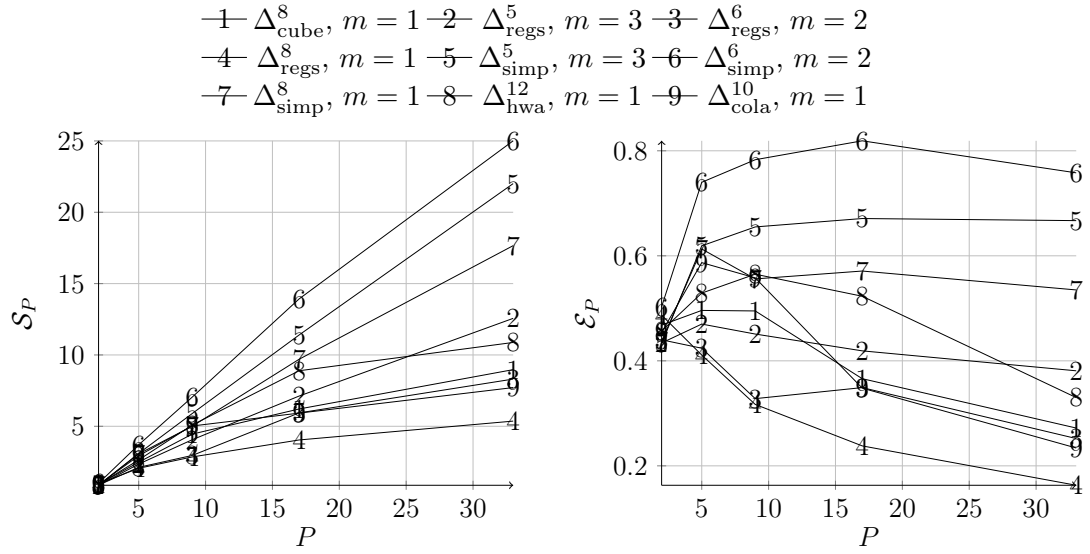
TABLE 3.2. Results of the numerical investigation of the algorithm $\mathcal{B}_{14}$ and corresponding results obtained by using the algorithm $\mathcal{B}_{09}$.

Δ^n	m	E^*	$\mathcal{B}_{09}$	$\mathcal{B}_{14}$
Δ^4_{cube}	1	0.4082	14	18
	2	0.0000	73	12
	3	0.0000	353	6
Δ^8_{cube}	1	0.4787	11,260	10,948
	2	0.2245	2,157,090	205,032
	3	0.0000	35,216,122	355,611
Δ^4_{regs}	1	0.4082	14	22
	2	0.0000	63	32
	3	0.0000	133	38
Δ^5_{regs}	1	0.4472	73	142
	2	0.1907	1,322	800
	3	0.0000	23,017	256
Δ^6_{regs}	1	0.4714	432	868
	2	0.2309	27,255	21,393
	3	0.0000	335,771	55,606
Δ^7_{regs}	1	0.4880	2,951	6,478
	2	0.2621	1,655,631	1,020,040
	3	0.0945	92,710,201	20,115,704
Δ^4_{simp}	1	0.3651	14	21
	2	0.0000	73	13
	3	0.0000	313	12
Δ^5_{simp}	1	0.4140	73	112
	2	0.0000	662	66
	3	0.0000	9,837	39
Δ^6_{simp}	1	0.4554	432	702
	2	0.1869	16,076	15,632
	3	0.0000	578,691	1,185
Δ^7_{simp}	1	0.4745	2,951	4,890
	2	0.2247	422,940	585,436
	3	0.0000	20,674,563	168,547
Δ^9_{hwa}	1	0.0107	2,217	1,591
	2	0.0000	2,344,833	151,835
Δ^{12}_{hwa}	1	0.1790	71,748	32,629
Δ^8_{ruusk}	1	0.2975	665	1,590
	2	0.1096	82,617	374,190
$\Delta^{12}_{\text{uhlen}}$	1	0.2112	36,559	236,168
$\Delta^{10}_{\text{cola}}$	1	0.3642	60,077	63,744

The efficiency evaluates the useful work of the cores of the computer. The closer $\mathcal{E}_P$ is to number 1, the more effectively the cores are used. The main purpose of the numerical investigation of the algorithm PBB was to evaluate

its speed-up $\mathcal{S}_P$ and efficiency $\mathcal{E}_P$ by using different dissimilarity matrices and different values for m and P.

The numerical investigation was conducted on a cluster of computers with Intel(R) Core(TM) i5-760 processors and with the CentOS 6.3 (32-bit) operating system. Each processor consists of four cores running at 2.8 GHz. Algorithm 2.3 has been evaluated on 2, 5, 9, 17, and 33 cores. The speed-up and the efficiency of the algorithm PBB are shown in Figure 3.1. The best values of the speed-up $\mathcal{S}_P$ are achieved in cases of Δ_{simp}, the worst – in the case of Δ^8_{regs}, $m = 1$. However, in all cases, the speed-up is not linear. The best values of the efficiency $\mathcal{E}_P$ are achieved in cases of Δ_{simp} and the worst – in the case of Δ^8_{regs}, $m = 1$, too. Unfortunately, in all cases, the efficiency is decreasing. These results show that the speed-up and the efficiency of the parallel algorithm PBB depend on the input data. In addition, the static distribution of works for processes is one of the main reasons for the non-linear speed-up and decreasing efficiency. Indeed, a process stops working if the assigned job is done.



(A) Speed-up. (B) Efficiency.

FIGURE 3.1. Speed-up and efficiency of the parallel algorithm PBB.

4. Conclusions

(1) The objective function of the optimization problem arising in multidimensional scaling (MDS) with city-block distances contains a set of absolute value (modulus) terms. Such an objective function may even be non-differentiable at a minimizer. Meanwhile, the objective function of the problem CMDS, which is equivalent to the optimization problem arising in MDS with city-block distances, is a convex quadratic function which does not have modulus terms.

(2) The objective function of the optimization problem arising in MDS with city-block distances is defined on the mn-dimensional real vector space. The objective function of the problem CMDS is defined on the mn^2-dimensional real vector space. Thus, the number of components of a variable of the problem CMDS is n times larger than the number of components of a variable of the problem arising in MDS with city-block distances.

(3) The feasible set of the problem CMDS is defined by both linear and complementarity constraints. These complementarity constraints are non-convex quadratic constraints. The objective function of the problem CMDS is a convex quadratic function. Hence, the problem CMDS is a non-convex quadratically constrained quadratic programming problem which, in turn, is an NP-hard problem.

(4) The problem CMDS may be formulated as a two-level optimization problem. The first $m+s$ linear constraints of the lower-level feasible set do not depend on the upper-level variable. The lower-level objective function does not depend on the upper-level variable, either. Thus, in solving lower-level problems, a portion of the calculations may be performed in advance.

(5) The optimization problem arising in MDS with city-block distances may be formulated as a two-level optimization problem. However, the two-level formulations of the problem CMDS and the original problem – i.e., the optimization problem arising in MDS with city-block distances – are different. Namely, the number of lower-level problems at

the two-level formulation of the problem CMDS is equal to $2^{mn(n-1)/2}$. Meanwhile, the number of lower-level problems at the two-level formulation of the original problem is equal to $(n!)^m$.

(6) The proposed algorithm MAS, based on the active-set method, is more sensitive to the selection of an initial feasible point than the algorithm SMOOTH, which is currently the best known local search algorithm for MDS with city block distances. However, the algorithm MAS is considerably faster: during approximately the same time, MAS generates up to 97 times more local minimizers than the algorithm SMOOTH.

(7) The proposed algorithm BB, based on the branch-and-bound method, considerably (up to 9×10^{18} times) decreases the number of lower-level problems which have to be solved. Moreover, the number of lower-level problems solved by using the algorithm BB is up to 488 times smaller than by using the corresponding algorithm, described in literature.

(8) The proposed parallel algorithm PBB, based on a parallel branch-and-bound method, is up to 25 times faster than the sequential version on 33 processes (processing units). On average, the efficiency of the parallelization is around 0.4.

A. Appendix

Here, we present and describe seven dissimilarity matrices, used in the numerical experiments of the proposed algorithms. Every matrix was obtained by evaluating relationships between certain multidimensional elements in geometry ((1)–(3) matrices), in pharmacology ((4)–(6) matrices), and in a well-known experiment with soft drinks. Thus, we selected the following matrices:

(1) Matrix $\Delta^{2^k}_{\text{cube}}$. Every element $\delta_{ij} \in \{0, \ldots, k\}$ $(1 \leq i, j \leq 2^k)$ of $\Delta^{2^k}_{\text{cube}}$ defines city-block distance between vertices i and j of a unit k-dimensional hypercube [**97**].

$$\Delta^{2^k}_{\text{cube}} = \begin{pmatrix} 0 & 1 & 1 & 2 & 1 & 2 & 2 & 3 & \ldots & k{-}1 & k \\ 1 & 0 & 2 & 1 & 2 & 1 & 3 & 2 & \ldots & k & k{-}1 \\ 1 & 2 & 0 & 1 & 2 & 3 & 1 & 2 & \ldots & k{-}2 & k{-}1 \\ 2 & 1 & 1 & 0 & 3 & 2 & 2 & 1 & \ldots & k{-}1 & k{-}2 \\ 1 & 2 & 2 & 3 & 0 & 1 & 1 & 2 & \ldots & k{-}2 & k{-}1 \\ 2 & 1 & 3 & 2 & 1 & 0 & 2 & 1 & \ldots & k{-}1 & k{-}2 \\ 2 & 3 & 1 & 2 & 1 & 2 & 0 & 1 & \ldots & k{-}3 & k{-}2 \\ 3 & 2 & 2 & 1 & 2 & 1 & 1 & 0 & \ldots & k{-}2 & k{-}3 \\ \vdots & & & & & & & & \ddots & & \vdots \\ k{-}1 & k & k{-}2 & k{-}1 & k{-}2 & k{-}1 & k{-}3 & k{-}2 & \ldots & 0 & 1 \\ k & k{-}1 & k{-}1 & k{-}2 & k{-}1 & k{-}2 & k{-}2 & k{-}3 & \ldots & 1 & 0 \end{pmatrix}.$$

(2) Matrix $\Delta^{k+1}_{\text{regs}}$. Every element $\delta_{ij} \in \{0, 1\}$ $(1 \leq i, j \leq k+1)$ of $\Delta^{k+1}_{\text{regs}}$ defines city-block distance between vertices i and j of a regular k-simplex. Note that distances between all vertices of a regular k-simplex are equal by using any Minkowski distance [**97**].

$$\Delta^{k+1}_{\text{regs}} = \begin{pmatrix} 0 & 1 & 1 & 1 & \ldots & 1 \\ 1 & 0 & 1 & 1 & \ldots & 1 \\ 1 & 1 & 0 & 1 & \ldots & 1 \\ 1 & 1 & 1 & 0 & \ldots & 1 \\ \vdots & & & & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \ldots & 0 \end{pmatrix}.$$

(3) Matrix $\Delta^{k+1}_{\text{simp}}$. Every element $\delta_{ij} \in \{0, 1, 2\}$ $(1 \leq i, j \leq k+1)$ of $\Delta^{k+1}_{\text{simp}}$ defines city-block distance between vertices i and j of a k-simplex, in which one vertex is located at the origin of the coordinate system and other vertices are located at the coordinate axes with equal distances from the origin. In this case, only distances between the origin and

the other vertices are equal [**97**].

$$\Delta_{\text{simp}}^{k+1} = \begin{pmatrix} 0 & 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & 2 & 2 & \dots & 2 \\ 1 & 2 & 0 & 2 & \dots & 2 \\ 1 & 2 & 2 & 0 & \dots & 2 \\ \vdots & & & & \ddots & \vdots \\ 1 & 2 & 2 & 2 & \dots & 0 \end{pmatrix}.$$

(4) Matrices Δ_{hwa}^{9} and Δ_{hwa}^{12}. Matrix Δ_{hwa}^{9} defines dissimilarities between 9 ligands that bind to certain wild-type and mutant α_1 adrenergic receptors (α_1-adrenoceptors). Every element of the matrix Δ_{hwa}^{12} defines dissimilarity between two particular α_1-adrenoceptors that are found in nature (wild-type) and mutated by a human [**96, 51**].

$$\Delta_{\text{hwa}}^{9} = \begin{pmatrix} 0 & 0.2619 & 2.3276 & 0.5914 & 0.6112 & 1.0379 & 0.6123 & 0.4949 & 1.1386 \\ 0.2619 & 0 & 2.0935 & 0.3573 & 0.3771 & 0.8037 & 0.8465 & 0.7291 & 1.3727 \\ 2.3276 & 2.0935 & 0 & 1.7362 & 1.7164 & 1.2898 & 2.9399 & 2.8225 & 3.4662 \\ 0.5914 & 0.3573 & 1.7362 & 0 & 0.0536 & 0.4464 & 1.2038 & 1.0863 & 1.7300 \\ 0.6112 & 0.3771 & 1.7164 & 0.0536 & 0 & 0.4266 & 1.2236 & 1.1061 & 1.7498 \\ 1.0379 & 0.8037 & 1.2898 & 0.4464 & 0.4266 & 0 & 1.6502 & 1.5328 & 2.1765 \\ 0.6123 & 0.8465 & 2.9399 & 1.2038 & 1.2236 & 1.6502 & 0 & 0.2144 & 0.5263 \\ 0.4949 & 0.7291 & 2.8225 & 1.0863 & 1.1061 & 1.5328 & 0.2144 & 0 & 0.6437 \\ 1.1386 & 1.3727 & 3.4662 & 1.7300 & 1.7498 & 2.1765 & 0.5263 & 0.6437 & 0 \end{pmatrix},$$

$$\Delta_{\text{hwa}}^{12} = \begin{pmatrix} 0 & 0.0691 & 0.0625 & 0.0454 & 0.3403 & 0.0630 & 0.1209 & 0.1795 & 0.3243 & 0.1542 & 0.1552 & 0.4581 \\ 0.0691 & 0 & 0.0969 & 0.0732 & 0.3569 & 0.1106 & 0.1376 & 0.1961 & 0.3525 & 0.1506 & 0.1603 & 0.4748 \\ 0.0625 & 0.0969 & 0 & 0.0683 & 0.3075 & 0.0701 & 0.0979 & 0.1467 & 0.3305 & 0.1561 & 0.1994 & 0.4254 \\ 0.0454 & 0.0732 & 0.0683 & 0 & 0.3108 & 0.0826 & 0.0914 & 0.1516 & 0.3257 & 0.1136 & 0.1423 & 0.4286 \\ 0.3403 & 0.3569 & 0.3075 & 0.3108 & 0 & 0.3575 & 0.2194 & 0.1749 & 0.2278 & 0.2064 & 0.2544 & 0.1178 \\ 0.0630 & 0.1106 & 0.0701 & 0.0826 & 0.3575 & 0 & 0.1382 & 0.1967 & 0.3266 & 0.1900 & 0.1759 & 0.4754 \\ 0.1209 & 0.1376 & 0.0979 & 0.0914 & 0.2194 & 0.1382 & 0 & 0.1027 & 0.2726 & 0.1015 & 0.1584 & 0.3372 \\ 0.1795 & 0.1961 & 0.1467 & 0.1516 & 0.1749 & 0.1967 & 0.1027 & 0 & 0.1981 & 0.0597 & 0.0879 & 0.2786 \\ 0.3243 & 0.3525 & 0.3305 & 0.3257 & 0.2278 & 0.3266 & 0.2726 & 0.1981 & 0 & 0.2294 & 0.2110 & 0.2109 \\ 0.1542 & 0.1506 & 0.1561 & 0.1136 & 0.2064 & 0.1900 & 0.1015 & 0.0597 & 0.2294 & 0 & 0.0638 & 0.3242 \\ 0.1552 & 0.1603 & 0.1994 & 0.1423 & 0.2544 & 0.1759 & 0.1584 & 0.0879 & 0.2110 & 0.0638 & 0 & 0.3665 \\ 0.4581 & 0.4748 & 0.4254 & 0.4286 & 0.1178 & 0.4754 & 0.3372 & 0.2786 & 0.2109 & 0.3242 & 0.3665 & 0 \end{pmatrix}.$$

(5) Matrices $\Delta_{\text{ruusk}}^{8}$ and $\Delta_{\text{ruusk}}^{20}$. The matrix $\Delta_{\text{ruusk}}^{8}$ defines dissimilarities between certain α_2-adrenoceptors that are found in the body of a human and a zebrafish. The matrix $\Delta_{\text{ruusk}}^{20}$ defines dissimilarities between 20 ligands that bind to the α_2-adrenoceptors [**96, 75**].

$$\Delta_{\text{ruusk}}^{8} = \begin{pmatrix} 0 & 10.1831 & 18.1173 & 18.9519 & 17.0474 & 15.7394 & 17.0255 & 16.5943 \\ 10.1831 & 0 & 15.7345 & 14.0502 & 13.0642 & 11.8421 & 11.9610 & 11.6784 \\ 18.1173 & 15.7345 & 0 & 8.6088 & 10.7763 & 7.4855 & 5.5403 & 6.8461 \\ 18.9519 & 14.0502 & 8.6088 & 0 & 14.7345 & 7.9303 & 7.1678 & 8.1943 \\ 17.0474 & 13.0642 & 10.7763 & 14.7345 & 0 & 10.7469 & 9.7950 & 10.1099 \\ 15.7394 & 11.8421 & 7.4855 & 7.9303 & 10.7469 & 0 & 7.9056 & 8.4393 \\ 17.0255 & 11.9610 & 5.5403 & 7.1678 & 9.7950 & 7.9056 & 0 & 2.3735 \\ 16.5943 & 11.6784 & 6.8461 & 8.1943 & 10.1099 & 8.4393 & 2.3735 & 0 \end{pmatrix}, \quad \Delta_{\text{ruusk}}^{20} =$$

$$\begin{pmatrix} 0 & 10.3 & 2.8 & 7.6 & 14.2 & 14.0 & 4.8 & 4.7 & 2.5 & 10.3 & 4.5 & 12.1 & 12.3 & 7.7 & 6.7 & 5.7 & 24.5 & 22.0 & 16.0 & 15.9 \\ 10.3 & 0 & 9.9 & 3.2 & 6.9 & 4.3 & 15.1 & 14.8 & 9.1 & 6.8 & 8.7 & 6.9 & 4.3 & 6.2 & 7.3 & 5.8 & 14.2 & 11.6 & 5.6 & 5.5 \\ 2.8 & 9.9 & 0 & 7.2 & 13.0 & 13.6 & 6.1 & 5.2 & 2.2 & 10.3 & 4.4 & 11.7 & 11.9 & 7.4 & 7.3 & 5.5 & 24.1 & 21.6 & 15.6 & 15.5 \\ 7.6 & 3.2 & 7.2 & 0 & 10.0 & 7.0 & 12.4 & 12.1 & 6.4 & 6.8 & 5.5 & 6.0 & 5.0 & 4.2 & 5.5 & 3.9 & 16.9 & 14.4 & 8.4 & 8.3 \\ 14.2 & 6.9 & 13.0 & 10.0 & 0 & 3.9 & 18.4 & 17.9 & 13.0 & 13.2 & 13.1 & 12.4 & 10.1 & 12.3 & 14.2 & 10.0 & 11.2 & 8.6 & 4.1 & 5.1 \\ 14.0 & 4.3 & 13.6 & 7.0 & 3.9 & 0 & 18.7 & 18.5 & 12.7 & 9.3 & 11.3 & 8.9 & 6.6 & 8.8 & 10.7 & 9.2 & 10.6 & 8.0 & 2.5 & 2.6 \\ 4.8 & 15.1 & 6.1 & 12.4 & 18.4 & 18.7 & 0 & 2.5 & 6.2 & 13.8 & 8.1 & 16.8 & 17.0 & 12.5 & 11.2 & 9.6 & 29.3 & 26.7 & 20.7 & 20.6 \\ 4.7 & 14.8 & 5.2 & 12.1 & 17.9 & 18.5 & 2.5 & 0 & 5.7 & 13.7 & 7.2 & 16.6 & 16.8 & 12.3 & 11.0 & 9.3 & 29.0 & 26.5 & 20.5 & 20.4 \\ 2.5 & 9.1 & 2.2 & 6.4 & 13.0 & 12.7 & 6.2 & 5.7 & 0 & 9.6 & 2.6 & 10.8 & 11.0 & 6.5 & 6.5 & 4.4 & 23.3 & 20.7 & 14.7 & 14.6 \\ 10.3 & 6.8 & 10.3 & 6.8 & 13.2 & 9.3 & 13.8 & 13.7 & 9.6 & 0 & 8.4 & 5.8 & 5.3 & 4.6 & 6.7 & 8.3 & 15.5 & 13.0 & 9.4 & 9.0 \\ 4.5 & 8.7 & 4.4 & 5.5 & 13.1 & 11.3 & 8.1 & 7.2 & 2.6 & 8.4 & 0 & 9.6 & 9.6 & 5.3 & 5.9 & 4.1 & 21.9 & 19.3 & 13.3 & 13.2 \\ 12.1 & 6.9 & 11.7 & 6.0 & 12.4 & 8.9 & 16.8 & 16.6 & 10.8 & 5.8 & 9.6 & 0 & 3.8 & 4.8 & 6.4 & 9.1 & 12.9 & 10.6 & 8.4 & 8.0 \\ 12.3 & 4.3 & 11.9 & 5.0 & 10.1 & 6.6 & 17.0 & 16.8 & 11.0 & 5.3 & 9.6 & 3.8 & 0 & 4.5 & 5.8 & 7.4 & 12.3 & 9.7 & 6.1 & 5.7 \\ 7.7 & 6.2 & 7.4 & 4.2 & 12.3 & 8.8 & 12.5 & 12.3 & 6.5 & 4.6 & 5.3 & 4.8 & 4.5 & 0 & 2.9 & 5.0 & 16.8 & 14.2 & 9.3 & 9.5 \\ 6.7 & 7.3 & 7.3 & 5.5 & 14.2 & 10.7 & 11.2 & 11.0 & 6.5 & 6.7 & 5.9 & 6.4 & 5.8 & 2.9 & 0 & 5.7 & 18.1 & 15.5 & 10.2 & 10.3 \\ 5.7 & 5.8 & 5.5 & 3.9 & 10.0 & 9.2 & 9.6 & 9.3 & 4.4 & 8.3 & 4.1 & 9.1 & 7.4 & 5.0 & 5.7 & 0 & 19.7 & 17.1 & 11.1 & 11.0 \\ 24.5 & 14.2 & 24.1 & 16.9 & 11.2 & 10.6 & 29.3 & 29.0 & 23.3 & 15.5 & 21.9 & 12.9 & 12.3 & 16.8 & 18.1 & 19.7 & 0 & 3.3 & 8.6 & 8.7 \\ 22.0 & 11.6 & 21.6 & 14.4 & 8.6 & 8.0 & 26.7 & 26.5 & 20.7 & 13.0 & 19.3 & 10.6 & 9.7 & 14.2 & 15.5 & 17.1 & 3.3 & 0 & 6.0 & 6.1 \\ 16.0 & 5.6 & 15.6 & 8.4 & 4.1 & 2.5 & 20.7 & 20.5 & 14.7 & 9.4 & 13.3 & 8.4 & 6.1 & 9.3 & 10.2 & 11.1 & 8.6 & 6.0 & 0 & 1.3 \\ 15.9 & 5.5 & 15.5 & 8.3 & 5.1 & 2.6 & 20.6 & 20.4 & 14.6 & 9.0 & 13.2 & 8.0 & 5.7 & 9.5 & 10.3 & 11.0 & 8.7 & 6.1 & 1.3 & 0 \end{pmatrix}.$$

(6) Matrix $\Delta^{12}_{\text{uhlen}}$. This matrix defines dissimilarities between particular α_2-adrenoceptors, found in the body of a human, rat, guinea pig, and pig [**96, 83**].

$$\Delta^{12}_{\text{uhlen}} = \begin{pmatrix} 0 & 0.7976 & 2.4536 & 3.2453 & 1.6306 & 1.7244 & 2.1499 & 5.6167 & 2.4947 & 2.7339 & 2.2523 & 1.8651 \\ 0.7976 & 0 & 2.9771 & 3.9830 & 1.9604 & 2.0471 & 2.6135 & 6.0063 & 2.0847 & 2.4462 & 1.9965 & 2.3740 \\ 2.4536 & 2.9771 & 0 & 1.5652 & 2.3175 & 2.4420 & 2.2802 & 3.4729 & 4.8680 & 4.7743 & 4.3960 & 3.3584 \\ 3.2453 & 3.9830 & 1.5652 & 0 & 3.3233 & 3.3835 & 2.9664 & 5.0381 & 4.7857 & 4.5819 & 4.0194 & 3.2761 \\ 1.6306 & 1.9604 & 2.3175 & 3.3233 & 0 & 0.3428 & 0.9828 & 4.2168 & 3.1995 & 3.0933 & 3.6521 & 2.4572 \\ 1.7244 & 2.0471 & 2.4420 & 3.3835 & 0.3428 & 0 & 1.2612 & 4.1758 & 3.2219 & 3.1343 & 3.6931 & 2.7170 \\ 2.1499 & 2.6135 & 2.2802 & 2.9664 & 0.9828 & 1.2612 & 0 & 4.5881 & 3.6432 & 3.2744 & 3.3278 & 2.5614 \\ 5.6167 & 6.0063 & 3.4729 & 5.0381 & 4.2168 & 4.1758 & 4.5881 & 0 & 6.9689 & 7.3101 & 7.8689 & 6.1184 \\ 2.4947 & 2.0847 & 4.8680 & 4.7857 & 3.1995 & 3.2219 & 3.6432 & 6.9689 & 0 & 0.9274 & 1.5241 & 1.5724 \\ 2.7339 & 2.4462 & 4.7743 & 4.5819 & 3.0933 & 3.1343 & 3.2744 & 7.3101 & 0.9274 & 0 & 1.7415 & 2.0632 \\ 2.2523 & 1.9965 & 4.3960 & 4.0194 & 3.6521 & 3.6931 & 3.3278 & 7.8689 & 1.5241 & 1.7415 & 0 & 1.7506 \\ 1.8651 & 2.3740 & 3.3584 & 3.2761 & 2.4572 & 2.7170 & 2.5614 & 6.1184 & 1.5724 & 2.0632 & 1.7506 & 0 \end{pmatrix}.$$

(7) Matrix $\Delta^{10}_{\text{cola}}$. This matrix was obtained after a certain experiment related to the comparison of soft drinks [**37**]. A group of 38 students tasted the following soft drinks: Pepsi, Coke, Classic Coke, Diet Pepsi, Diet Slice, Diet 7-Up, Dr. Pepper, Slice, 7-Up, and Tab. After the tasting, each student was invited to compare dissimilarities between every pair of the drinks in a 9-point scale. If drinks i and j ($i \neq j$) looked "very similar", then k-th student had to mark $\delta^k_{ij} = 1$. If drinks i and j looked "entirely different", $\delta^k_{ij} = 9$. After the comparisons, the matrix $\Delta^{10}_{\text{cola}}$ was constructed. Every element $\delta_{ij} \in \{0, \ldots, 342\}$ ($1 \leq i, j \leq 10$) of the matrix denotes the accumulated dissimilarities, i.e., $\delta_{ij} = \sum_{k=1}^{38} \delta^k_{ij}$.

$$\Delta^{10}_{\text{cola}} = \begin{pmatrix} 0 & 127 & 169 & 204 & 309 & 320 & 286 & 317 & 321 & 238 \\ 127 & 0 & 143 & 235 & 318 & 322 & 256 & 318 & 318 & 231 \\ 169 & 143 & 0 & 243 & 326 & 327 & 258 & 318 & 318 & 242 \\ 204 & 235 & 243 & 0 & 285 & 288 & 259 & 312 & 317 & 194 \\ 309 & 318 & 326 & 285 & 0 & 155 & 312 & 131 & 170 & 285 \\ 320 & 322 & 327 & 288 & 155 & 0 & 306 & 164 & 136 & 281 \\ 286 & 256 & 258 & 259 & 312 & 306 & 0 & 300 & 295 & 256 \\ 317 & 318 & 318 & 312 & 131 & 164 & 300 & 0 & 132 & 291 \\ 321 & 318 & 318 & 317 & 170 & 136 & 295 & 132 & 0 & 297 \\ 238 & 231 & 242 & 194 & 285 & 281 & 256 & 291 & 297 & 0 \end{pmatrix}.$$

List of publications

[FGŽ14] Roger Fletcher, Nerijus Galiauskas, and Julius Žilinskas. Quadratic programming with complementarity constraints for multidimensional scaling with city-block distances. *Baltic Journal of Modern Computing*, 2(4):248–259, 2014. ISSN 2255-8942.

[GŽ11] Nerijus Galiauskas and Julius Žilinskas. Quadratic programming problems. *Journal of Young Scientists*, 33(4):115–118, 2011. ISSN 1648-8776. [In Lithuanian].

[GŽ13] Nerijus Galiauskas and Julius Žilinskas. Parallel branch and bound for multidimensional scaling with L1 distances formulated as quadratic programming with complementarity constraints. In *3PGCIC 2013: 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiégne, France, October 28–30, 2013*, pages 509–512. IEEE, 2013. ISBN 978-0-7695-5094-7.

[GŽ14] Nerijus Galiauskas and Julius Žilinskas. On multidimensional scaling with city-block distances. In *Lecture Notes in Computer Science*, volume 8426, pages 82–87. Springer, 2014. ISSN 0302-9743.

References

[1] Howard Anton. *Elementary linear algebra*. John Wiley & Sons, 2010.

[2] Phipps Arabie. Was Euclid an unnecessarily sophisticated psychologist? *Psychometrika*, 56(4):567–587, 1991.

[3] Mordecai Avriel. *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.

[4] David A. Bader, William E. Hart, and Cynthia A. Phillips. Parallel algorithm design for branch and bound. In *Tutorials on Emerging Methodologies and Applications in Operations Research*, pages 5–1. Springer, 2005.

[5] Rajendra Bhatia. *Positive definite matrices*. Princeton University Press, 2009.

[6] Ake Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[7] Ingwer Borg and Patrick J. F. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 2005.

[8] Ingwer Borg, Patrick J. F. Groenen, and Patrick Mair. *Applied multidimensional scaling*. Springer Science & Business Media, 2012.

[9] Jonathan M. Borwein and Adrian S. Lewis. *Convex analysis and nonlinear optimization: theory and examples*, volume 3. Springer Science & Business Media, 2010.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Gilles Brassard and Paul Bratley. *Fundamentals of algorithmics*. Prentice-Hall, Inc., 1996.

[12] Michael J. Brusco. A simulated annealing heuristic for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices. *Journal of Classification*, 18(1):3–33, 2001.

[13] Michael J. Brusco and Stephanie Stahl. *Branch-and-bound applications in combinatorial data analysis*. Springer, 2006.

[14] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[15] Edwin K. P. Chong and Stanislaw H. Zak. *An introduction to optimization*. John Wiley & Sons, 2013.

[16] Jens Clausen. Branch and bound algorithms – principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30, 1999.

[17] Benoít Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[18] Andrew R. Conn, Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. *Mathematical Programming*, 87(2):215–249, 2000.

[19] Michael A. A. Cox and Trevor F. Cox. Multidimensional scaling. In *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics, pages 315–347. Springer Berlin Heidelberg, 2008.

[20] Teodor Gabriel Crainic, Bertrand Le Cun, and Catherine Roucairol. Chapter 1. parallel branch-and-bound algorithms. *Parallel combinatorial optimization*, 1:1–28, 2006.

[21] Etienne De Klerk. *Aspects of semidefinite programming: interior point algorithms and selected applications*. Springer Science & Business Media, 2002.

[22] Caterina De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79(1):71–75, 1990.

[23] G. De Soete, L. Hubert, and P. Arabie. On the use of simulated annealing for combinatorial data analysis. In *Data, expert knowledge and decisions*, pages 329–340. Springer, 1988.

[24] Stephan Dempe, Vyatcheslav V. Kalashnikov, and Nataliya Kalashnykova. Optimality conditions for bilevel programming problems. In *Optimization with Multivalued Mappings*, pages 3–28. Springer, 2006.

[25] Zdenek Dostál. *Optimal quadratic programming algorithms: with applications to variational inequalities*, volume 23. Springer Science & Business Media, 2009.

[26] Gintautas Dzemyda, Olga Kurasova, and Julius Žilinskas. *Multidimensional Data Visualization: Methods and Applications*, volume 75 of *Springer Optimization and Its Applications*. Springer, 2012.

[27] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[28] Roger Fletcher and Tom Johnson. On the stability of null-space methods for KKT systems. *SIAM Journal on Matrix Analysis and Applications*, 18(4):938–958, 1997.

[29] Masao Fukushima. Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems. *Mathematical programming*, 53(1-3):99–110, 1992.

[30] Wendell R Garner. *The processing of information and structure*. Psychology Press, 2014.

[31] Bernard Gendron and Teodor Gabriel Crainic. Parallel branch-and-branch algorithms: Survey and synthesis. *Operations research*, 42(6):1042–1066, 1994.

[32] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical optimization*, volume 5. Academic press London, 1981.

[33] Philip E. Gill and Elizabeth Wong. Methods for convex and general quadratic programming. *Mathematical Programming Computation*, pages 1–42, 2014.

[34] Gene H. Golub and James M. Ortega. *Scientific computing: an introduction with parallel computing*. Elsevier, 2014.

[35] Gene H. Golub and Charles F. Van Loan. *Matrix computations*, volume 3. Johns Hopkins University Press, 2012.

[36] Nicholas I. M. Gould, Mary E. Hribar, and Jorge Nocedal. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4):1376–1395, 2001.

[37] Paul E Green, Frank J Carmone, and Scott M Smith. *Multidimensional scaling: concepts and applications*. Allyn and Bacon, 1989.

[38] Patrick J. F. Groenen and Willem J. Heiser. The tunneling method for global optimization in multidimensional scaling. *Psychometrika*, 61(3):529–550, 1996.

[39] Patrick J. F. Groenen, Willem J. Heiser, and Jacqueline J. Meulman. City-block scaling: smoothing strategies for avoiding local minima. In *Classification, Data Analysis, and Data Highways*, pages 46–53. Springer, 1998.

[40] Patrick J. F. Groenen, Willem J. Heiser, and Jacqueline J. Meulman. Global optimization in least-squares multidimensional scaling by distance smoothing. *Journal of classification*, 16(2):225–254, 1999.

[41] Patrick J. F. Groenen, Rudolf Mathar, and Willem J. Heiser. The majorization approach to multidimensional scaling for Minkowski distances. *Journal of Classification*, 12(1):3–19, 1995.

[42] Patrick John Fitzgerald Groenen. *The majorization approach to multidimensional scaling: some problems and extensions*. DSWO Press, Leiden University, 1993.

[43] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

[44] Sven Hammarling and Craig Lucas. Updating the QR factorization and the least squares problem. 2008. MIMS EPrint: 2008.111.

[45] Willem J. Heiser. The city-block model for three-way multidimensional scaling. In *Multiway data analysis*, pages 395–404. Elsevier Science, 1989.

[46] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, 2012.

[47] Reiner Horst, Panos M. Pardalos, and Nguyen V. Thoai. *Introduction to Global Optimization*, volume 48 of *Nonconvex Optimization and Its Applications*. Kluwer, Dordrecht, 2000.

[48] Peter J. Huber. *Robust statistics*. Wiley, New York, 1981.

[49] Lawrence Hubert, Phipps Arabie, and Matthew Hesson-Mcinnis. Multidimensional scaling in the city-block metric: a combinatorial approach. *Journal of Classification*, 9(2):211–236, 1992.

[50] David R. Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[51] John Hwa, Robert M Graham, and Dianne M Perez. Identification of critical determinants of α_1-adrenergic receptor subtype selective agonist binding. *Journal of Biological Chemistry*, 270(39):23189–23195, 1995.

[52] Toshihide Ibaraki. Representation theorems for equivalent optimization problems. *Information and Control*, 21(5):397–435, 1972.

[53] Lester Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[54] Joseph Jada. *An introduction to parallel algorithms*. Addison Wesley, 1992.

[55] Virendra K. Janakiram, Edward F. Gehringer, Dharma P. Agrawal, and Ravi Mehrotra. A randomized parallel branch-and-bound algorithm. *International Journal of Parallel Programming*, 17(3):277–301, 1988.

[56] J. M. Jansen and F. W. Sijstermans. Parallel branch-and-bound algorithms. *Future Generation Computer Systems*, 4(4):271–279, 1989.

[57] David B. Kirk and W. Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Newnes, 2012.

[58] Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[59] Joseph B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.

[60] Joseph B. Kruskal and Myron Wish. *Multidimensional scaling*. Sage, 1978.

[61] Gautham K. Kudva and Joseph F. Pekny. A distributed exact algorithm for the multiple resource constrained sequencing problem. *Annals of Operations Research*, 42(1):25–54, 1993.

[62] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin/Cummings Publishing Company Redwood City, CA, 1994.

[63] Kenneth Lange, David R. Hunter, and Ilsoon Yang. Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics*, 9(1):1–20, 2000.

[64] Leon S. Lasdon. *Optimization theory for large systems*. Courier Corporation, 2013.

[65] Gue Myung Lee, Nguyen Nang Tam, and Nguyen Dong Yen. *Quadratic programming and affine variational inequalities: a qualitative study*. Springer Science & Business Media, 2006.

[66] Pui Lam Leung and Kin-nam Lau. Estimating the city-block two-dimensional scaling model with simulated annealing. *European Journal of Operational Research*, 158(2):518–524, 2004.

[67] Donald L. Miller and Joseph F. Pekny. The role of performance metrics for parallel mathematical programming algorithms. *ORSA Journal on Computing*, 5(1):26–28, 1993.

[68] Katta G. Murty. *Optimization for Decision Making*. Springer, 2009.

[69] Angelia Nedic, D. P. Bertsekas, and A. E. Ozdaglar. *Convex analysis and optimization*. Athena Scientific, 2003.

[70] Pu-yan Nie. A null space method for solving system of equations. *Applied Mathematics and computation*, 149(1):215–226, 2004.

[71] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 2006.

[72] Pablo Pedregal. *Introduction to optimization*. Springer Science & Business Media, 2003.

[73] Joseph F. Pekny and Donald L. Miller. A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems. *Mathematical programming*, 55(1-3):17–33, 1992.

[74] Vadim Pliner. Metric unidimensional scaling and global optimization. *Journal of classification*, 13(1):3–18, 1996.

[75] Jori O Ruuskanen, Jonne Laurila, Henri Xhaard, Ville-Veikko Rantanen, Karoliina Vuoriluoto, Siegfried Wurster, Anne Marjamäki, Minna Vainio, Mark S Johnson, and Mika Scheinin. Conserved structural, pharmacological and functional properties among the three human and five zebrafish α_2-adrenoceptors. *British journal of pharmacology*, 144(2):165–177, 2005.

[76] L. Ridgway Scott, Terry Clark, and Babak Bagheri. *Scientific parallel computing*, volume 146. Princeton University Press Princeton, 2005.

[77] Gary Shao, Francine Berman, and Rich Wolski. Master/slave computing on the grid. In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, pages 3–16. IEEE, 2000.

[78] Jan Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*. Springer Science & Business Media, 2005.

[79] Hamdy A. Taha. *Operations research: an introduction*. Pearson/Prentice Hall, 2007.

[80] Fabio Tardella. On the equivalence between some discrete and continuous optimization problems. *Annals of Operations Research*, 25(1):291–300, 1990.

[81] Paul R. Thie and Gerard E. Keough. *An introduction to linear programming and game theory*. John Wiley & Sons, 2011.

[82] Warren S Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419, 1952.

[83] Staffan Uhlen, Maija Dambrova, Johnny Näsman, Helgi B Schiöth, Yuchen Gu, Anna Wikberg-Matsson, and JE Wikberg. [^{3}H]RS79948-197 binding to human, rat, guinea pig and pig α_{2A}-, α_{2B}-and α_{2C}-adrenoceptors. Comparison with MK912, RX821002, rauwolscine and yohimbine. *European journal of pharmacology*, 343(1):93–101, 1998.

[84] Luis N. Vicente and Paul H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5(3):291–306, 1994.

[85] Julius Žilinskas. *Black box global optimization: covering methods and their parallelization*. PhD thesis, Kaunas University of Technology, June 2002.

[86] David W. Walker and Jack J. Dongarra. MPI: a standard message passing interface. *Supercomputer*, 12:56–68, 1996.

[87] Wayne L. Winston, Munirpallam Venkataramanan, and Jeffrey B. Goldberg. *Introduction to mathematical programming*. Thomson Brooks/Cole, 2003.

[88] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2000.

[89] Laurence A. Wolsey and George L. Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[90] Rui Xu and Don Wunsch. *Clustering*, volume 10. Wiley-IEEE Press, 2008.

[91] Zhibin Zhu. A sequential equality constrained quadratic programming algorithm for inequality constrained optimization. *Journal of Computational and Applied Mathematics*, 212(1):112–125, 2008.

[92] Antanas Žilinskas and Julius Žilinskas. Parallel hybrid algorithm for global optimization of problems occurring in MDS-based visualization. *Computers & Mathematics with Applications*, 52(1):211–224, 2006.

[93] Antanas Žilinskas and Julius Žilinskas. Two level minimization in multidimensional scaling. *Journal of Global Optimization*, 38(4):581–596, 2007.

[94] Antanas Žilinskas and Julius Žilinskas. Branch and bound algorithm for multidimensional scaling with city-block metric. *Journal of Global Optimization*, 43(2-3):357–372, 2009.

[95] Antanas Žilinskas and Julius Žilinskas. Optimization-based visualization. In *Encyclopedia of Optimization*, pages 2785–2791. Springer, 2009.

[96] J. Žilinskas. Multidimensional scaling in protein and pharmacological sciences. *Computer Aided Methods in Optimal Design and Operations, Series on Computers and Operations Research*, 7:139–148, 2006.

[97] Julius Žilinskas. Reducing of search space of multidimensional scaling problems with data exposing symmetries. *Information Technology and Control*, 36(4):377–382, 2007.

[98] Julius Žilinskas. Parallel branch and bound for multidimensional scaling with city-block distances. *Journal of Global Optimization*, 54(2):261–274, 2012.

Index

Nerijus Galiauskas

OPTIMIZATION ALGORITHMS FOR MULTIDIMENSIONAL SCALING WITH CITY-BLOCK DISTANCES AND THEIR PARALLELIZATION

Doctoral Dissertation
Technological Sciences, Informatics Engineering (07 T)

Editor Alex Sullivan