An Adaptive Partial Linearization Method for Optimization Problems on Product Sets

Igor Konnov

Received: 27 May 2016 / Accepted: 30 September 2017 / Published online: 10 October 2017
© Springer Science+Business Media, LLC 2017

Abstract We consider a general class of composite optimization problems where the goal function is the sum of a smooth function and a non-necessary smooth convex separable function associated with some space partition, whereas the feasible set is a Cartesian product concordant to this partition. We suggest an adaptive version of the partial linearization method, which makes selective component-wise steps satisfying some descent condition and utilizes a sequence of control parameters. This technique is destined to reduce the computational expenses per iteration and maintain the basic convergence properties. We also establish its convergence rates and describe some examples of applications. Preliminary results of computations illustrate usefulness of the new method.

Keywords Composite optimization · Decomposable problems · Partial linearization method · Conditional gradient method · Tolerance control

Mathematics Subject Classification 90C30 · 90C06 · 65K05

1 Introduction

The conditional gradient method is one of the oldest smooth constrained optimization methods. It was first suggested in [1] for the case when the goal function is quadratic and the feasible set is polyhedral and further was developed by many authors, see, e.g., [2–6]. We recall that the main idea of this method consists in linearization of the goal...
function, so that solution of the linearized problem over the initial feasible set serves for finding the descent direction. During rather long time, the conditional gradient method was not considered as very efficient due to its relatively slow convergence in comparison with Newton- and projection-type methods. However, it has gained a great amount of attention very recently due to several features significant for many applications, where huge dimensionality and inexact data create certain drawbacks for more rapid methods. Moreover, in the case of a polyhedral feasible set its direction finding problem appears simpler than those in the other methods, and its solution yields usually so-called sparse approximations, see, e.g., [7–9] and the references therein.

Furthermore, this method can be extended to the case where the goal function is the sum of a smooth function and a non-smooth convex function. The appearance of the non-smooth term is usually caused by regularization or exact penalty techniques, see, e.g., [6,10]. In this case, one can apply the partial linearization (PL for short) method from [11] (see [12,13] for further development) where only the smooth function is linearized in the direction finding problem. This approach proved to be useful if the initial problem is (partially) decomposable, which is typical for very large dimensional problems. Then, its direction finding problem is decomposed into fully independent component subproblems.

The main goal of this paper is to suggest a modification of PL methods for this class of composite optimization problems, which maintains the basic convergence properties, but enables one to reduce the computational expenses per iteration. We follow in general the approach suggested in [14] for regularized splitting methods. Namely, the method contains an explicit sequence of tolerance parameters. At each iteration, we make only one component-wise step, whose choice is adjusted to the current tolerance value and provides some sufficient descent property. This rule gives significant freedom in selection of the component index. However, the direction finding problem in [14] has always a unique solution and yields the continuity of the descent mapping. Unlike [14], our current direction finding problem is based on utilization of the PL technique without any regularization, which seems simpler essentially, but implies the set-valuedness and dis-continuity of the direction mapping. Therefore, the method proposed will require new substantiation schemes. We take the inexact Armijo-type line-search rule, which makes our method different from those in [11,13] even in the non-decomposable case, although other stepsize choice rules can also be applied.

The remainder of the paper is organized as follows. Section 2 contains the formulation of the basic optimization problem and its necessary properties. Some examples of applications are given in Sect. 3. Section 4 is devoted to the description and substantiation of the new method. In Sect. 5, we discuss some modifications of the line-search procedure in this method. Its convergence rates are given in Sect. 6. Section 7 contains results of preliminary series of computational experiments.

2 Problem Formulation and Preliminary Properties

We first fix our notation. In what follows, we denote by $\mathbb{R}^s$ the real $s$-dimensional Euclidean space, all elements of such spaces being column vectors represented by
a lower case Roman alphabet in boldface, e.g., $x$. We use superscripts to denote different vectors and subscripts to denote different scalars or components of vectors. For any vectors $x$ and $y$ of $\mathbb{R}^s$, we denote by $\langle x, y \rangle$ their scalar product, i.e.,

$$\langle x, y \rangle = x^\top y = \sum_{i=1}^s x_i y_i,$$

and by $\|x\|$ the Euclidean norm of $x$, i.e., $\|x\| = \sqrt{\langle x, x \rangle}$. We denote by $\mathbb{R}^s_+$ the nonnegative orthant in $\mathbb{R}^s$, i.e., $\mathbb{R}^s_+ = \{ u \in \mathbb{R}^s : u_i \geq 0, \ i = 1, \ldots, s \}$. We also set $\mathcal{R} = \mathbb{R} \cup \{-\infty, +\infty\}$. Given a function $\varphi : \mathbb{R}^s \to \mathcal{R}$, we can define its domain

$$\text{dom}\varphi = \{ x \in \mathbb{R}^s : \varphi(x) > -\infty \}.$$

We shall consider our basic optimization problem in the space $\mathbb{R}^N$. Let $\mathcal{N} = \{1, \ldots, N\}$, define some partition

$$\mathcal{N} = \bigcup_{i=1}^n \mathcal{N}_i$$

with $|\mathcal{N}_i| = N_i$, $N = \sum_{i=1}^n N_i$, and $\mathcal{N}_i \cap \mathcal{N}_j = \emptyset$ if $i \neq j$. Then, any point $x = (x_1, \ldots, x_N)^\top \in \mathbb{R}^N$ is represented by $x = (x_1, \ldots, x_n)^\top$ where $x_i = (x_j)_{j \in \mathcal{N}_i} \in \mathbb{R}^{N_i}$ for $i = 1, \ldots, n$.

We consider the partitionable optimization problem of the form

$$\min_{x \in X_1 \times \cdots \times X_n} \mu(x) := \left\{ f(x) + \sum_{i=1}^n h_i(x_i) \right\}, \quad (1)$$

where $X_i$ is a non-empty, convex, and compact set in $\mathbb{R}^{N_i}$, $h_i : \mathbb{R}^{N_i} \to \mathcal{R}$ is a convex, proper, lower semi-continuous function with $\text{dom} h_i \supseteq X_i$ for $i = 1, \ldots, n$, $f : \mathbb{R}^N \to \mathcal{R}$ is a smooth, but not necessary convex function. That is,

$$X := X_1 \times \cdots \times X_n = \prod_{i=1}^n X_i \subset \mathbb{R}^N.$$

The solution set of (1) will be denoted by $X^*$ and the optimal value of the function by $\mu^*$, i.e.,

$$\mu^* = \inf_{x \in X} \mu(x).$$
For brevity, set \( g(x) = f'(x) \), then
\[
g(x) = (g_1(x), \ldots, g_n(x))^\top, \text{ where } g_i(x) = \left( \frac{\partial f(x)}{\partial x_j} \right)_{j \in N_i} \in \mathbb{R}^{N_i}, \ i = 1, \ldots, n;
\]
and let
\[
h(x) = \sum_{i=1}^n h_i(x_i).
\]

From the above assumptions, it follows that the function \( \mu \) is directionally differentiable at each point \( x \in X \), that is, its directional derivative with respect to any vector \( d \) is defined by the formula:
\[
\mu'(x; d) = \langle g(x), d \rangle + h'(x; d), \text{ with } h'(x; d) = \sum_{i=1}^n \max_{b_i \in \partial h_i(x_i)} \langle b_i, d_i \rangle,
\]
see, e.g., [15].

Together with problem (1), we will consider the following mixed variational inequality (MVI for short): Find a point \( x^* \in X = X_1 \times \cdots \times X_n \) such that
\[
\sum_{i=1}^n \left[ \langle g_i(x^*), y_i - x_i^* \rangle + h_i(y_i) - h_i(x_i^*) \right] \geq 0
\]
\[
\forall y_i \in X_i, \text{ for } i = 1, \ldots, n.
\]
This problem can be used as an optimality condition for problem (1).

**Proposition 2.1** [14, Proposition 2.1]

(a) Each solution of problem (1) solves MVI (3).
(b) If \( f \) is convex, then each solution of MVI (3) solves problem (1).

In what follows, we denote by \( X^0 \) the solution set of MVI (3) and call it the set of *stationary points* of problem (1).

For each point \( x \in X \) and each \( i = 1, \ldots, n \), we define by \( Y_i(x) \) the solution set of the optimization problem:
\[
\min_{y_i \in X_i} \rightarrow \{ \langle g_i(x), y_i \rangle + h_i(y_i) \}.
\]

Under the above assumptions, \( Y_i(x) \) is non-empty, convex, and compact. We define the set
\[
Y(x) = Y_1(x) \times \cdots \times Y_n(x),
\]
thus defining the set-valued marginal mapping \( x \mapsto Y(x) \). Also, we denote by \( y_i(x) \) a point of \( Y_i(x) \) and set \( y(x) = (y_1(x), \ldots, y_n(x))^\top \in X \). Moreover, we set
\[\sigma_i(x, y_i) := (g_i(x), x_i - y_i) + h_i(x_i) - h_i(y_i)\]

and define the function
\[\varphi(x) := \sum_{i=1}^{n} \varphi_i(x), \quad \varphi_i(x) := \sigma_i(x, y_i(x)) = \max_{y_i \in X_i} \sigma_i(x, y_i) \quad \text{for } i = 1, \ldots, n.\]

We will make use of this function for the substantiation of our method. We now establish weakened continuity-type properties of this function and the marginal mapping. We recall that given a set \(V \subseteq \mathbb{R}^s\), a set-valued mapping \(v \mapsto Q(v)\) is said to be closed on a set \(W \subseteq V\), if for each pair of sequences \({u^k}\) → \(u\), \({q^k}\) → \(q\) such that \(u^k \in W\) and \(q^k \in Q(u^k)\), we have \(q \in Q(u)\).

**Lemma 2.1**

(a) The function \(\varphi : \mathbb{R}^N \rightarrow \mathcal{R}\) is semi-continuous on \(X\);

(b) The mapping \(x \mapsto Y(x)\) is closed on \(X\).

**Proof**

Assertion (a) has been proved in [13, Lemma 4]. To obtain (b), take sequences \({x^k}\) → \(\tilde{x}\), \({y^k}\) → \(\tilde{y}\) with \(y^k \in Y(x^k)\). Then, from (4) we have

\[\langle g_i(x^k), u_i - y_i^k \rangle + h_i(u_i) - h_i(y_i^k) \geq 0\]
\[\forall u_i \in X_i, \quad \text{for } i = 1, \ldots, n.\]

Since \(g_i\) is continuous and \(h_i\) is lower semi-continuous, taking the limit \(k \rightarrow \infty\) gives

\[\langle g_i(\tilde{x}), u_i - \tilde{y}_i \rangle + h_i(u_i) - h_i(\tilde{y}_i) \geq 0\]
\[\forall u_i \in X_i, \quad \text{for } i = 1, \ldots, n;\]

hence, \(\tilde{y} \in Y(\tilde{x})\) and \(x \mapsto Y(x)\) is closed. \(\square\)

We now show that the value \(\varphi(x)\) gives an accuracy measure at a point \(x\) for problem (3), i.e., \(\varphi\) can serve as a gap function for this problem, see, e.g., [16] for more detail on gap functions.

**Proposition 2.2**

(a) For any point \(x \in X\), it holds that \(\varphi(x) \geq 0\), or, equivalently, \(\varphi_i(x) \geq 0\) for \(i = 1, \ldots, n;\)

(b) \(x \in X^0 \iff x \in Y(x) \iff \varphi(x) = 0 \iff \varphi_i(x) = 0, i = 1, \ldots, n.\)

**Proof**

Since \(\sigma_i(x, x_i) = 0\), assertion (a) is true. It follows that

\[\varphi(x) = 0 \iff \varphi_i(x) = 0, i = 1, \ldots, n.\]

Besides, \(\varphi(x) = 0\) clearly implies \(x \in Y(x)\). Next, if \(x = y(x) \in Y(x)\), then (4) implies \(x \in X^0\) and \(\varphi(x) \leq 0\); hence, by (a), \(\varphi(x) = 0\). Conversely, let \(x\) solve MVI (3), but \(x \not\in Y(x)\). Then, there exist an index \(l\) and a point \(x'_l \in X_l\) such that \(\sigma_l(x, x'_l) > 0\). It follows that

\(\square\) Springer
\[- \sum_{i \neq l} \sigma_i (x, x_i) - \sigma_l (x, x'_l) < 0,\]
i.e., \(x \notin X^0\), which is a contradiction. This means that assertion (b) is true. \(\square\)

We establish now a useful descent property of component-wise steps. Define for brevity \(I = \{1, \ldots, n\}\).

**Lemma 2.2** Take any points \(x \in X, y(x) \in Y(x)\) and an index \(s \in I\). If

\[
d_i = \begin{cases} y_s(x) - x_s & \text{if } i = s, \\ 0 & \text{if } i \neq s, \end{cases}
\]

then

\[
\mu'(x; d) \leq -\varphi_s(x).
\] (5)

**Proof** Due to the definition of \(d\) and (2), we have

\[
\mu'(x; d) = \langle g(x), d \rangle + h'(x; d) = \langle g_s(x), d_s \rangle + \max_{b_s \in \partial h_s(x_s)} \langle b_s, d_s \rangle.
\]

By convexity, we have

\[
\langle b_s, d_s \rangle \leq h_s(y_s(x)) - h_s(x_s)
\]

for any \(b_s \in \partial h_s(x_s)\). It follows that

\[
\mu'(x; d) \leq \langle g_s(x), y_s(x) - x_s \rangle + h_s(y_s(x)) - h_s(x_s) = -\varphi_s(x),
\]

and hence, (5) holds true. \(\square\)

3 Some Examples of Applications

We intend now to give some examples of applied problems which reduce to decomposable composite optimization problems of form (1), where utilization of the proposed adaptive PL method may give certain advantages. Besides, we recall for instance that various engineering problems based on the so-called group LASSO regression method have this format (see [17,18]), as well as many problems of network resource allocation in wireless multi-user interfering systems (see [19]).

3.1 Selective Classification Problems

One of the most popular approaches to data classification is support vector machine techniques, see, e.g., [20,21]. The simplest linear support vector machine problem for data classification consists in creating an optimal hyperplane separating two convex hulls of a collection of known points \(x'_i \in \mathbb{R}^m, i = 1, \ldots, l\), attributed to previous data.
observations with different labels $y_i \in \{-1, +1\}, i = 1, \ldots, l$, where $m$ is the number of features. That is, the distance between the hyperplane and each convex hull should be as long as possible. This separation of the feature space enables us to classify new data points. However, this requirement appears too strong for real problems where the so-called soft margin approach, which minimizes the penalties for mis-classification, is utilized. This problem can be formulated as the optimization problem

$$\min_{w \in \mathbb{R}^n} \rightarrow (1/p) \|w\|_p^p + C \sum_{i=1}^l L((w, x^i); y_i),$$

where $L$ is a loss function and $C > 0$ is a penalty parameter. The usual choice is $L(z; y) = \max\{0; 1 - yz\}$ and $p$ is either 1 or 2. Taking $p = 2$, we can rewrite this problem as

$$\min_{w, u} \rightarrow 0.5\|w\|^2 + C \sum_{i=1}^l u_i,$$

subject to

$$1 - y_i \langle w, x^i \rangle \leq u_i, \quad u_i \geq 0, \quad i = 1, \ldots, l.$$ 

In this formulation, each observation $i$ is attributed to some data point $x^i$; however, it seems worthwhile to use sets here since any object may be often represented by some set of features, this is also the case for noisy observations, see [22]. So, let object $i$ be represented by a set $X_i \in \mathbb{R}^m$. Suppose it is the convex hull of the points $x^{ik}$, $k = 1, \ldots, t$, which thus have the same label $y_i \in \{-1, +1\}$. Then, we write the soft margin classification problem as follows:

$$\min_{w, u} \rightarrow 0.5\|w\|^2 + C \sum_{i=1}^l u_i,$$

subject to

$$1 - y_i \langle w, x^{ik} \rangle \leq u_i, \quad k = 1, \ldots, t; \quad i = 1, \ldots, l; \quad u_i \geq 0, \quad i = 1, \ldots, l;$$

which somewhat differs from those in [22]. Here, $u_i$ is the set slack variable, and we impose the penalty for the sum of set slacks. By using the convex optimization theory, we can write the dual of the above classification problem that has the quadratic programming format:

$$\max_{\alpha} \rightarrow \sum_{i=1}^l \sum_{k=1}^t \alpha_{ik} - 0.5 \| \sum_{i=1}^l \sum_{k=1}^t (\alpha_{ik} y_i) x^{ik} \|^2$$

subject to

\[ \sum_{i=1}^l \alpha_{ik} = 1, \quad k = 1, \ldots, t. \]
\[
\sum_{k=1}^{t} \alpha_{ik} \leq C, \ i = 1, \ldots, l;
\]
\[
\alpha_{ik} \geq 0, \ k = 1, \ldots, t; \ i = 1, \ldots, l.
\]

Then, the solution of the primal problem is given by the formula:
\[
w = \sum_{i=1}^{l} \sum_{k=1}^{t} (\alpha_{ik} y_i) x^{ik}.
\]

We observe that the dual problem falls into format (1). Hence, our method can be suitable in the high-dimensional case, where the number of sets is also very large.

### 3.2 Network Equilibrium Problems

Various network equilibrium problems represent one of the main tools for evaluation of flows distribution in traffic and communication networks. We now describe the path flow formulation of the network equilibrium problem with elastic demands, see, e.g., [23] and references therein.

The model is determined on an oriented graph, each of its arc being associated with some flow and some cost (for instance, time of delay), which depends on the values of arc flows. Usually, the number of nodes and arcs is very large for applied problems.

Let us be given a graph with a finite set of nodes \( V \) and a set of oriented arcs \( A \) which join the nodes so that any arc \( a = (i, j) \) has the origin \( i \) and the destination \( j \). Next, among all the pairs of nodes of the graph we extract a subset of origin–destination (O/D) pairs \( M \) of the form \( m = (i \rightarrow j) \). Besides, each pair \( m \in M \) is associated with a variable flow demand \( v_m \) and with the set of paths \( P_m \) which connect the origin and destination for this pair. We suppose that each \( v_m \) is nonnegative with some upper bound \( \gamma_m \), possibly infinite, for \( m \in M \). Denote by \( \tau_m \) the desired minimal dis-utility for pair \( m \) and suppose that it depends on the flow demand, i.e., \( \tau_m = \tau_m(v_m) \). Also, denote by \( u_p \) the path flow for the path \( p \). Then, the feasible set of flows/demands \( W \) can be defined as follows:
\[
W = \prod_{m \in M} W_m,
\]
where
\[
W_m = \left\{ w_m = (u_m, v_m) : \sum_{p \in P_m} u_p = v_m, u_p \geq 0, \ p \in P_m, \ 0 \leq v_m \leq \gamma_m \right\}, \ \forall m \in M,
\]
where \( u_m = (u_p)_{p \in P_m} \). Given a flow vector \( u = (u_m)_{m \in M} \), one can determine the arc flow
\[
f_a = \sum_{m \in M} \sum_{p \in P_m} \alpha_{pa} u_p
\]
for each arc $a \in A$, where

$$\alpha_{pa} = \begin{cases} 1 & \text{if arc } a \text{ belongs to path } p, \\ 0 & \text{otherwise}. \end{cases}$$

If the vector $f = (f_a)_{a \in A}$ of arc flows is known, one can determine the arc cost $c_a(f_a)$. We suppose for simplicity that it depends on the arc flow of just this arc. Usually, arc costs are monotone increasing functions of arc flows. Then, one can compute costs for each path $p$:

$$g_p(u) = \sum_{a \in A} \alpha_{pa} c_a(f_a).$$

We say that a feasible flow/demand pair $(u^*, v^*) \in W$ is an equilibrium point if it satisfies the following conditions:

$$\forall m \in M, \exists \lambda_m \text{ such that } g_p(u^*) \begin{cases} \geq \lambda_m & \text{if } u^*_p = 0, \\ = \lambda_m & \text{if } u^*_p > 0, \end{cases} \forall p \in P_m \quad (6)$$

and

$$\tau_m(v^*_m) \begin{cases} \leq \lambda_m & \text{if } v^*_m = 0, \\ = \lambda_m & \text{if } v^*_m \in (0, \gamma_m), \quad \forall p \in P_m. \\ \geq \lambda_m & \text{if } v^*_m = \gamma_m; \end{cases} \quad (7)$$

However, conditions (6)–(7) determine equivalently the following VI: Find a pair $(u^*, v^*) \in W$ such that

$$\sum_{m \in M} \sum_{p \in P_m} g_p(u^*) (u_p - u^*_p) - \sum_{m \in M} \tau_m(v^*_m) (v_m - v^*_m) \geq 0 \quad \forall (u, v) \in W. \quad (8)$$

Furthermore, due to the separability of the functions $c_a$ and $\tau_m$, their continuity implies integrability, i.e., then there exist functions

$$\eta_a(f_a) = \int_0^{f_a} c_a(t) dt \quad \forall a \in A, \quad \sigma_m(v_m) = \int_0^{v_m} \tau_m(t) dt \quad \forall m \in M.$$ 

It follows that VI (8) also gives an optimality condition of the following optimization problem:

$$\min_{(u, v) \in W} \left\{ \sum_{a \in A} \eta_a(f_a) - \sum_{m \in M} \sigma_m(v_m) \right\}. \quad (9)$$

Hence, each solution of (9) is a solution to VI (8), the inverse assertion is true if the functions $\eta_a$ and $-\sigma_m$ are convex, and this seems rather natural. However, this problem falls into the basic format (1).
3.3 Penalty Method for Decomposable Optimization Problems

A great number of optimization problems related to large-scale systems are written as follows:

\[
\max \to \sum_{i=1}^{n} \langle c_i, x_i \rangle \quad (10)
\]

subject to

\[
\sum_{i=1}^{n} A_i x_i = b_0, \quad (11)
\]

\[
x_i \in X_i = \{ y \in \mathbb{R}_{+}^{l_i} : B_i y \leq b_i \}, \quad i = 1, \ldots, n; \quad (12)
\]

for instance, it can be attributed to the total income maximization in a system containing \( n \) subsystems (producers), who utilize common and particular factors. That is, producer \( i \) chooses an output vector \( x_i \in \mathbb{R}^{l_i} \); his/her consumption rates are described by an \( m_0 \times l_i \) matrix \( A_i \) of common factors and by an \( m_i \times l_i \) matrix \( B_i \) of particular factors; whereas the vector \( c_i \) denotes prices of his/her outputs, the vector \( b_i \in \mathbb{R}^{m_i} \) (respectively, \( b_0 \in \mathbb{R}^{m_0} \)) denotes inventories of particular (respectively, common) factors, see, e.g., [6,24]. Due to its very large dimensionality, a suitable decomposition approach can be utilized to reduce the computer memory and calculation expenses. For instance, the price (Dantzig–Wolfe) decomposition principle replaces problem (10)–(11) with its dual defined with the help of the Lagrangian including only the term associated with the common constraints in (12). However, we can also utilize the penalty approach and replace problem (10)–(11) with the sequence of auxiliary problems of the form

\[
\min \to 0.5 \tau \left\| \sum_{i=1}^{n} A_i x_i - b_0 \right\|^2 - \sum_{i=1}^{n} \langle c_i, x_i \rangle \quad (13)
\]

subject to

\[
x_i \in X_i, \quad i = 1, \ldots, n, \quad (14)
\]

where \( \tau > 0 \) is a penalty parameter. Clearly, problems (13)–(14) also fall into the basic format (1). Hence, application of component-wise steps with the conditional gradient type direction finding problem leads to a decomposition method for the initial problem. In fact, in order to find \( y_s^k = y_s(x^k) \in Y_s(x^k) \) (cf. (4)), we now have to solve the separate problem

\[
\min_{y_s \in X_s} \to \langle g_s(x^k), y_s \rangle,
\]

whereas the partial gradient in \( x_i \) of the cost function at \( x \) is written as follows
\[ g_i(x) = \tau A_i^\top \left[ \sum_{j=1}^n A_j x_j - b_0 \right] - c_i; \]

hence,

\[ g_i(x^k + \theta d^k) = g_i(x^k) + \theta \tau A_i^\top A_s d_s^k. \]

This means that we can make shifts only in the selected component \( d_s^k \) at each iteration.

### 4 The Descent Method with Inexact Line-Search

Denote by \( \mathbb{Z}_+ \) the set of nonnegative integers. The adaptive PL method with decomposition for MVI (3) is described as follows.

**Method (APL)**

**Initialization:** Choose a point \( z^0 \in X \), numbers \( \beta \in ]0, 1[ \), \( \theta \in ]0, 1[ \), and a sequence \( \{\delta_l\} \searrow 0 \). Set \( l = 1 \).

**Step 0:** Set \( k = 0, x^0 = z^{l-1} \).

**Step 1:** Choose an index \( s \in I \) such that \( \phi_s(x^k) \geq \delta_l \), set \( s_k = s \),

\[ d_i^k = \begin{cases} y_s - x_s^k & \text{if } i = s_k, \\ 0 & \text{if } i \neq s_k, \end{cases} \]

where \( y_s = y_s(x^k) \in Y_s(x^k) \) and go to Step 3. Otherwise (i.e., when \( \phi_i(x^k) < \delta_l \) for all \( i \in I \)) go to Step 2.

**Step 2:** Set \( z^l = x^k, l = l + 1 \) and go to Step 0. (Restart)

**Step 3:** Determine \( m \) as the smallest number in \( \mathbb{Z}_+ \) such that

\[ \mu(x^k + \theta^m d^k) \leq \mu(x^k) - \beta \theta^m \phi_s(x^k), \]

set \( \lambda_k = \theta^m, x^{k+1} = x^k + \lambda_k d^k, k = k + 1 \), and go to Step 1.

Thus, the method has two levels. Each its outer iteration (stage) \( l \) contains some number of inner iterations in \( k \) with the fixed tolerance \( \delta_l \). Completing each stage, which is marked as restart, leads to decreasing the tolerance value. Observe that the iteration point then remains the same, hence all the calculated subproblem solutions can be used for choosing the index \( s \) at \( x^0 \) without new calculations. It should be also noticed that calculation of \( \phi_s(x^k) \) in Step 1 requires a solution of the separate component problem (4), which gives a significant part of computational efforts per iteration. All the component problems must be solved at each iteration of the usual PL method. Here, the flexible index selection inequality is destined just for reduction of total computational efforts.

**Lemma 4.1** The line-search procedure in Step 3 is always finite.
Proof If we suppose that the line-search procedure is infinite, then
\[
θ^m(μ(x^k + θ^m d^k) - μ(x^k)) = -βφ_s(x^k),
\]
for \( m \to \infty \), hence, by taking the limit we have \( \mu'(x^k; d^k) \geq -βφ_s(x^k) \), but Lemma 2.2 gives \( \mu'(x^k; d^k) \leq -φ_s(x^k) \) and hence \( (1 - β)φ_s(x^k) \leq 0 \), a contradiction. \( \Box \)

We recall that a single-valued mapping \( p : \mathbb{R}^s \to \mathbb{R}^s \) is said to be uniformly continuous on a set \( V \subseteq \mathbb{R}^s \), if for any number \( ε > 0 \) there exists a number \( τ > 0 \) such that \( \|p(x) - p(y)\| < ε \) for each pair of points \( x, y \in V \) with \( \|x - y\| < τ \). Our convergence analysis will be based on the following property.

**Proposition 4.1** Suppose that the gradient map \( g : \mathbb{R}^N \to \mathbb{R}^N \) is uniformly continuous on \( X \). Then, the number of inner iterations at each outer iteration \( l \) is finite.

**Proof** Fix any \( l \). By construction, we have \(-∞ < μ^* ≤ μ(x^k) \) and \( μ(x^{k+1}) \leq μ(x^k) - βδ_l λ_k \) and hence
\[
lim_{k \to ∞} λ_k = 0. \tag{16}
\]
Besides, the sequence \( \{x^k\} \) is bounded and must have limit points. Suppose that the sequence \( \{x^k\} \) is infinite. Since the set \( I \) is finite, there is an index \( s_k = s \), which is repeated infinitely. Take the corresponding subsequence \( \{k_i\} \). We intend to evaluate the difference \( μ(x^{k_i} + λ_{k_i} d^{k_i}) - μ(x^{k_i}) \), but we temporarily remove these indices for more convenience. Then, using the mean value theorem and convexity of \( g \), we have
\[
\begin{align*}
μ(x + λ d) - μ(x) &= f(x + λ d) - f(x) + h_s(x_s + λ d_s) - h_s(x_s) \\
&≤ λ \langle g_s(x), y_s - x_s \rangle + h_s(y_s) - h_s(x_s) + λ \langle g_s(x + ξ λ d) - g_s(x), y_s - x_s \rangle \\
&≤ -λφ_s(x) + λ \|g_s(x + ξ λ d) - g_s(x)\| \|d_s\|,
\end{align*}
\]
where \( ξ = ξ_{k_i} \in (0, 1) \). Since \( X_s \) is bounded, \( \|d_s\| ≤ C_s < ∞ \). Due to the uniform continuity of \( g \), there exists a number \( λ' > 0 \) such that
\[
\|g_s(x + ξ λ d) - g_s(x)\| ≤ (1 - β)δ_l / C_s
\]
if \( λ ≤ λ' \), besides, \( φ_s(x) ≥ δ_l \). It follows that
\[
μ(x + λ d) - μ(x) ≤ -λφ_s(x) + λ(1 - β)δ_l ≤ -βλφ_s(x)
\]
if \( λ ≤ λ' \), hence \( λ_{k_i} ≥ \tilde{λ} > 0 \) by the stepsize rule, which contradicts (16). \( \Box \)

**Theorem 4.1** Suppose that the gradient map \( g : \mathbb{R}^N \to \mathbb{R}^N \) is uniformly continuous on \( X \). Then, the sequence \( \{z^l\} \) generated by Method (APL) has limit points; all these limit points are solutions of MVI (3). Besides, if \( f \) is convex, then
\[
\lim_{l \to ∞} μ(z^l) = μ^* \tag{17}
\]
and all the limit points of \( \{z^l\} \) belong to \( X^* \).
Proof Following the proof of Proposition 4.1, we see that \( \mu(z^{l+1}) \leq \mu(z^l) \); hence,
\[
\lim_{l \to \infty} \mu(z^l) = \tilde{\mu}.
\]
Besides, the sequence \( \{z^l\} \) is bounded and must have limit points. Take an arbitrary limit point \( \bar{z} \) of \( \{z^l\} \), then
\[
\lim_{l \to \infty} z^l = \bar{z}.
\]
For \( l > 0 \), we have
\[
\varphi_i(z^l) \leq \delta_i \text{ for all } i \in I,
\]
and hence, \( \varphi(z^l) \leq n\delta_i \). Due to Lemma 2.1, taking the limit \( l = l_t \to \infty \), we obtain \( \varphi(\bar{z}) \leq 0 \) and \( \bar{z} \in X \). Due to Proposition 2.2, this means that \( \varphi(\bar{z}) = 0 \) and that the point \( \bar{z} \) solves MVI (3). Next, if \( f \) is convex, then by Proposition 2.1 (b), each limit point of \( \{z^l\} \) solves problem (1). It follows that \( \tilde{\mu} = \mu^* \) and (17) holds. \( \square \)

The rule \( \{\delta_i\} \searrow 0 \), which provides convergence, gives a significant freedom for the choice of this sequence. Hence, the question of the efficient choice is open in general and can be partially answered from the convergence rate theorems, see Sect. 6. Similarly, the rule of choosing the index \( s \) is not essential for the fact of convergence, but can clearly have a great impact on the convergence rate. For instance, the independent sequential index choice seems rather reasonable and was used in the computational experiments described in Sect. 7.

In case \( h \equiv 0 \), (APL) is a new decomposable version of the conditional gradient method. It should be noted that the decomposable conditional gradient method with independent solution of all the partial direction finding subproblems was considered in [25]. However, even solution of all these subproblems at each iteration may appear too expensive as well as changing all the components of the current iterate. Sequential component-wise versions of this method were proposed in [26, 27]. In [26], this strategy is applied to the multi-commodity formulation of a network equilibrium problem and the maximal violated component corresponding to one origin–destination pair is chosen by solution of suitable dual problems. In [27], three component-wise algorithms for a general smooth convex optimization problem on the Cartesian product set are proposed. The first two are the sequential cyclic (Gauss type) versions of the conditional gradient method, where the second admits series of successful descent steps in the same component. The third utilizes the Gauss-Southwell rule and represents some modification of that from [26], i.e., it also selects the maximal violated component with the help of some gap function. That is, these methods also have in fact to solve all the partial direction finding subproblems. Another way consists in making the random choice of one component. A randomized block-coordinate variant of the conditional gradient method was rather recently proposed in [28]. A general

\[\text{\footnotesize Papers [26, 27] were suggested to the author by an anonymous referee.}\]
scheme of block-descent methods for problems of form (1) was given in [16]. We observe that all the algorithms from [16,25,26,28] and the first algorithm from [27] unlike (APL) do not utilize sequences of tolerance parameters. Next, the second and third algorithms from [27] utilize tolerance parameters for the exit from the series of successful descent component-wise steps, but they do not use parameters for the choice of the component itself as in (APL). Therefore, all these methods relay upon different from (APL) algorithmic frameworks.

Although the dimensions $N_i$ can be arbitrary, we see that (APL) becomes an adaptive coordinate-wise descent method in case $N_i = 1$ for $i = 1, \ldots, n$. We think that (APL) may have advantages, in particular, over the method from [14], in case when $N_i > 1$ and all the sets $X_i$ are polyhedrons. Also, it may have advantages over the usual conditional gradient and partial linearization methods if the number of subsets $n$ is rather large.

Remark 4.1 The initial boundedness requirement for the feasible set $X$ was made in Sect. 2 only for more simplicity of exposition and can be replaced with proper coercivity assumptions. In fact, instead of compactness of each set $X_i$ we can require their closedness and add, e.g., the following conditions.

(C1) For each $i \in I$ and for each sequence $\{u^l_i\}$ such that $u^l_i \in X_i$ and $\{\|u^l_i\|\} \to \infty$ as $l \to \infty$, we have $\{h_i(u^l_i)/\|u^l_i\|\} \to +\infty$.

(C2) For each sequence $\{u^l\}$ such that $u^l \in X$ and $\{\|u^l\|\} \to \infty$ as $l \to \infty$, we have $\{\mu(u^l)\} \to +\infty$.

Then, (C1) provides existence of a solution of auxiliary problem (4); moreover, the sequence $\{d^k\}$ is bounded if so is $\{x^k\}$. From (C2), it follows that $\mu^* > -\infty$, problems (1) and (3) have solutions, and the sequence $\{x^k\}$ is bounded. Therefore, all the assertions of Sect. 4 remain true.

Also, we supposed that $\text{dom} h_i \supseteq X_i$ for $i = 1, \ldots, n$ only for more simplicity of exposition. Set

$$D = \prod_{i=1}^n (\text{dom} h_i \cap X_i).$$

It suffices to assume $D \neq \emptyset$. Then, we should only take the initial point $z^0 \in D$.

5 Modifications of the Line-Search Procedure

Together with the current Armijo rule in (15), we can utilize some other line-search procedures in (APL). For instance, we can take the one-dimensional minimization rule. The modified method can be substantiated under the same assumptions as in Theorem 4.1; however, the one-dimensional minimization is not finite as the current Armijo rule, see Lemma 4.1. We are interested in developing line-search procedures that are concordant to the partition of the space given in Sect. 2 and do not require calculation of all the components of the gradient and new point at each iteration. Note
that rule (15) needs the value of the cost function \( \mu \) (namely \( f \) and \( h_x \)) at each trial point.

Let us first consider the \textit{convex case} where the function \( f \) is convex. Then, we can replace (15) with the following:

\[
(g_i(x^k + \theta^m d^k), d^k_i) + \theta^{-m} \left\{ h_x(x^k + \theta^m d^k) - h_x(x^k) \right\} \leq -\beta \varphi_i(x^k). \tag{18}
\]

Since the trial point \( x^k + \theta^m d^k \) has the shift from \( x^k \) only in \( d^k_i \), it can be implemented independently of other variables. From (18), it now follows that

\[
\mu(x^k + \theta^m d^k) - \mu(x^k) = f(x^k + \theta^m d^k) - f(x^k) + h_x(x^k + \theta^m d^k) - h_x(x^k)
\]

\[
\leq \theta^m \langle g_i(x^k + \theta^m d^k), d^k_i \rangle + h_x(x^k + \theta^m d^k) - h_x(x^k) \leq -\beta \theta^m \varphi_i(x^k),
\]

and (15) holds true. It easy to see that all the assertions of Sect. 4 remain true for this version.

Moreover, we can utilize even a pre-defined stepsize in the \textit{Lipschitz gradient case}. Let us suppose that partial gradients of the function \( f \) are Lipschitz continuous, i.e.,

\[
\|g_i(x + d^{(i)}) - g_i(x)\| \leq L_i \|d^{(i)}\| = L_i \|d_i\|
\]

for any vector \( x \), where

\[
d^{(i)}_j = \begin{cases} 
 d_i & \text{if } j = i, \\
 0 & \text{if } j \neq i,
\end{cases}
\]

for \( i \in I \) and any vector \( d = (d_1, \ldots, d_n)^\top \in \mathbb{R}^N \). Clearly, this property holds if the gradient of \( f \) is Lipschitz continuous with some constant \( L > 0 \); since then, \( L_i \leq L \) holds for each \( i \in I \). It is known that any function \( \phi \) having the Lipschitz continuous gradient with the Lipschitz constant \( L_\phi \) satisfies the inequality

\[
\phi(y) \leq \phi(x) + \langle \phi'(x), y - x \rangle + 0.5L_\phi\|y - x\|^2,
\]

see [3, Lemma 1.2]. Similarly, for any vectors \( x \) and \( d \), we have

\[
f(x + d^{(i)}) \leq f(x) + \langle g_i(x), d_i \rangle + 0.5L_i \|d_i\|^2 \quad \forall i \in I.
\]

If \( d_i = y_i(x) - x_i \), then we have

\[
\mu(x + \lambda d^{(i)}) - \mu(x) = f(x + \lambda d^{(i)}) - f(x) + h_i(x_i + \lambda d_i) - h_i(x_i)
\]

\[
\leq \lambda \left\{ \langle g_i(x), d_i \rangle + h_i(y_i(x)) - h_i(x_i) \right\} + 0.5L_i \lambda^2 \|d_i\|^2
\]

\[
\leq -\lambda \varphi_i(x) + 0.5L_i \lambda^2 \|d_i\|^2 \leq -\beta \lambda \varphi_i(x),
\]

if

\[
\lambda \leq \tilde{\lambda}_{(i)}(x) = 2(1 - \beta) \varphi_i(x)/\|d_i\|^2 L_i.
\]
It follows that (15) holds with \( \lambda_k \geq \min\{1, \theta\lambda(x^k)\} > 0 \). Moreover, we can simply set \( \lambda_k = \bar{\lambda}(s)(x^k) > 0 \), and all the assertions of Proposition 4.1 and Theorem 4.1 remain true for this version. This modification reduces the computational expenses essentially since calculations of the goal function values are not necessary and we can calculate values of the partial gradients \( g_i \) and functions \( h_i \) only for necessary separate components. Clearly, the adaptive PL method admits other modifications and extensions, e.g., selection of a group of indices in \( I \) instead of only one component. These opportunities make the method very flexible and suitable for parallel and distributed computations applicable for very high-dimensional optimization problems, see, e.g., [8, 10, 16, 29].

### 6 Convergence Rates

In this section, we give some convergence rates for Method (APL). We suppose that all the basic assumptions of Sect. 2 hold, but will also utilize some additional conditions.

We first establish the finite termination property under the following **sharp solution condition**, which modifies those in [6, Chapter 7, Sect. 1, Sect. 3] and [30, Sect. 2.2].

There exist a number \( \tau > 0 \) and a point \( \bar{x} \in X \) such that

\[
\langle g(\bar{x}), x - \bar{x} \rangle + h(x) - h(\bar{x}) \geq \tau \|x - \bar{x}\| \quad \forall x \in X.
\]

**Theorem 6.1** Let a sequence \( \{z_l\} \) be generated by Method (APL). Suppose that the function \( f \) is convex, its gradient is Lipschitz continuous with constant \( L < \infty \), and the sharp solution condition holds. Then, there exists a stage number \( t \) such that \( X^* = Y(z^t) \).

**Proof** First, we note that the sharp solution condition implies \( \bar{x} \in X^0 \), and, by convexity, \( X^0 = X^* \); see Proposition 2.1. Next, if there exists some other point \( \tilde{x} \in X \), which provides the sharp solution condition, then, again by convexity, we must have

\[
\langle g(\bar{x}), \tilde{x} - \bar{x} \rangle + h(\bar{x}) - h(\tilde{x}) \leq \langle g(\bar{x}), \tilde{x} - \bar{x} \rangle + h(\bar{x}) - h(\tilde{x}) \\
\leq -\tau \|\tilde{x} - \bar{x}\| < 0,
\]

which is a contradiction. Hence, \( X^* = \{\bar{x}\} \). From the sharp solution condition for any point \( x \in X \), we have

\[
\langle g(z'), \tilde{x} - x \rangle + h(\bar{x}) - h(x) \\
= \langle g(\bar{x}), \tilde{x} - x \rangle + h(\tilde{x}) - h(x) + \langle g(z') - g(\bar{x}), \tilde{x} - x \rangle \\
\leq -\tau \|\tilde{x} - x\| + L\|z' - \bar{x}\||\tilde{x} - x\| \\
= -\tau \|\tilde{x} - x\|(1 - L\|z' - \bar{x}\|).
\]

From Theorem 4.1, we now have \( \|z' - \bar{x}\| \to 0 \) as \( l \to +\infty \). Hence,

\[
\langle g(z'), \tilde{x} - x \rangle + h(\tilde{x}) - h(x) < 0 \quad \forall x \in X, x \neq \bar{x},
\]
for \(l\) large enough. It follows that there exists a number \(t\) such that \(Y(z^t) = \{\bar{x}\}\). \(\square\)

In the method, each stage contains a finite number of inner iterations. Therefore, it seems suitable to derive its complexity estimate, which gives the total amount of work of the method. We now suppose in addition that the function \(f\) is convex and its partial gradients satisfy Lipschitz continuity conditions with constants \(L_i\) for each \(i \in I\). Then, it is given in Sect. 5 that we can take the stepsize

\[
\lambda_k = \lambda_i(x^k) = 2(1 - \beta)\varphi_i(x^k)/\left(\|d_s^k\|^2 L_s\right) \geq 2(1 - \beta)\varphi_i(x^k)/\left(\rho^2 L_s\right),
\]

where

\[L = \max_{s \in I} L_s, \quad \rho = \max_{s \in I} \rho_s, \quad \rho_s = \text{diam} X_s,
\]

see (19). We take the value \(\Phi(x) = \mu(x) - \mu^*\) as an accuracy measure for our method. In other words, given a starting point \(z^0\) and a number \(\varepsilon > 0\), we define the complexity of the method, denoted by \(V(\varepsilon)\), as the total number of iterations at \(l(\varepsilon)\) stages such that \(l(\varepsilon)\) is the maximal number \(l\) with \(\Phi(z^l) \geq \varepsilon\); hence,

\[
V(\varepsilon) \leq \sum_{l=1}^{l(\varepsilon)} V_l,
\]

where \(V_l\) denotes the total number of iterations at stage \(l\). We proceed to estimate the right-hand side of (21) and thus evaluate the complexity of the method.

**Theorem 6.2** Let a sequence \(\{z^l\}\) be generated by Method (APL) with the rule:

\[
\delta_l = v^l \delta_0, \quad l = 0, 1, \ldots; \quad v \in (0, 1), \quad \delta_0 > 0.
\]

Suppose that the function \(f\) is convex and its partial gradients satisfy Lipschitz continuity conditions with constants \(L_i\) for each \(i \in I\). Then, the method has the complexity estimate

\[
V(\varepsilon) \leq C_1 v((n\delta_0/\varepsilon) - 1)/(1 - v),
\]

where \(C_1 = \rho^2 Ln/(2\beta(1 - \beta)\delta_0)\).

**Proof** By (15), we have

\[
\mu(x^{k+1}) \leq \mu(x^k) - \beta \lambda_k \delta_l,
\]

and hence, in view of (20), we obtain

\[
V_l \leq \rho^2 L \Phi(z^{l-1})/(2\beta(1 - \beta)\delta_l^2).
\]

\(\square\) Springer
Under the above assumptions, for some \( x^* \in X^* \) it holds that

\[
\mu(z^l) - \mu(x^*) = f(z^l) - f(x^*) + h(z^l) - h(x^*) \\
\leq \langle g(z^l), z^l - x^* \rangle + h(z^l) - h(x^*) \\
\leq \max_{y \in X} \left\{ \langle g(z^l), z^l - y \rangle + h(z^l) - h(y) \right\} \\
= \varphi(z^l) \leq n \delta_l.
\]

Using this estimate in (23) gives

\[
V_l \leq \rho^2 L n \delta_{l-1}/(2 \beta (1 - \beta) \delta_l^2).
\]

From (22), it follows that

\[
V_l \leq \rho^2 L n v^{-l}/(2 \beta (1 - \beta) \delta_0 v) = (C_1/v) v^{-l}.
\]

Besides, since \( \varepsilon \leq \Phi(z^l) \leq n \delta_l = n \delta_0 v^l \), we have

\[
v^{-l(\varepsilon)} \leq n \delta_0/\varepsilon.
\]

Combining both the inequalities in (21), we obtain

\[
V(\varepsilon) \leq C_1 \sum_{l=1}^{l(\varepsilon)} v^{-l-1} = C_1 v(v^{-l(\varepsilon)} - 1)/(1 - v) \\
\leq C_1 v((n \delta_0/\varepsilon) - 1)/(1 - v).
\]

\[\square\]

We observe that the order of the estimates is similar to that in the usual conditional gradient methods under the same assumptions, see, e.g., [2,4,6].

7 Computational Experiments

In order to compare the performance of the presented method with the usual non-decomposable version, we carried out preliminary series of computational experiments. For simplicity, we took only the smooth problems, when the function \( h \) was smooth. We took two sets of problems.

7.1 Network Equilibrium Test Problems

First, we considered the performance of the partial linearization (PL) method from [11,12] and our method (APL) on network equilibrium test problems of form (8) or (9) described in Sect. 3.2. It is indicated in Sect. 3.2 that these problems fall into the basic format (1) and the suggested method (APL) can be applied to this problem.
We first discuss some peculiarities of its implementation. The basic auxiliary problem in (4) is now reduced to the problem of finding an element \((\bar{u}_s, \bar{v}_s) = y_s(x^k) \in Y_s(x^k)\) with \(x^k = (u^k, v^k)\), which corresponds to a solution of the problem

\[
\min_{(u_s, v_s) \in W_s} \left\{ \sum_{p \in P_s} g_p(u^k)u_p - \sigma_s(v_s) \right\}
\]

(24)

for some selected (O/D) pair \(s \in \mathcal{M}\). This solution can be found with the simple procedure below, which is based on optimality conditions (6)–(7).

First, we calculate the shortest path \(t \in P_s\) for the pair \(s\) with the minimal cost \(\tilde{\lambda} = g_t(u^k)\).

**Case 1.** If \(\tau_s(0) \leq \tilde{\lambda}\), then set \(\bar{v}_s = 0\) and \(\bar{u}_p = 0\) for all \(p \in P_s\), \(\lambda_s = \tilde{\lambda}\). Otherwise go to Case 2.

**Case 2.** If \(\tau_s(\gamma_s) \geq \tilde{\lambda}\), set \(\lambda_s = \tilde{\lambda}, \bar{v}_s = \gamma_s, \bar{u}_t = \gamma_s\), and \(\bar{u}_p = 0\) for all \(p \in P_s, p \neq t\). Otherwise go to Case 3.

**Case 3.** We have \(\tau_s(\gamma_s) < \tilde{\lambda} < \tau_s(0)\). By continuity of \(\tau_s\), we find the value \(\bar{v}_s \in [0, \gamma_s]\) such that \(\tau_s(\bar{v}_s) = \tilde{\lambda}\), set \(\lambda_s = \tilde{\lambda}, \bar{u}_t = \bar{v}_s\), and \(\bar{u}_p = 0\) for all \(p \in P_s, p \neq t\).

Next, since the number of all the paths may be too large, implementation of the methods is based on the standard reduction technique, where only active paths with positive flows are taken into account for each (O/D) pair, i.e., we deal with some subset of arcs and paths at each iteration, see, e.g., [23]. In fact, extension of this subset is due to appearance of a new shortest path for some (O/D) pair.

**Remark 7.1** We supposed above that each function \(\tau_m\) is continuous, i.e., that each function \(\sigma_m\) is smooth. However, the described procedure for problem (24) is extended easily to the case where \(-\sigma_m\) is convex and continuous, and then, \(\tau_m\) can be set-valued. At the same time, we note that the network equilibrium problem with fixed demands differs only in somewhat simplified formulation of problem (8). Clearly, the described method remains convergent in these cases and seems more flexible in comparison with the usual conditional gradient- and projection-type methods.

For comparison, we took two slightly modified test examples of network equilibrium problems with elastic demands from [31]. We used the arc cost functions \(c_a(f_a) = 1 + 0.5f_a\) for all \(a \in A\) and the minimal path cost (dis-utility) functions \(\tau_m(v_m) = 30 - 0.5v_m\) for all \(m \in M\).

We took \(\Delta_k = \phi(x^k)\) as accuracy measure for the methods. Both the methods were implemented with the Armijo line-search rule where \(\beta = \theta = 0.5\). Due to the above description, we see that we can take the total number of blocks where the line-search procedure was utilized as unified complexity measure for both the methods, which will be called block iterations. We reported this value in the tables for attaining different accuracies together with the processor time expenses. The methods for network equilibrium problems were implemented in C++ with double-precision arithmetic in an Intel(R) Core (TM) i3-4170 CPU, 3.70 GHz and 4 Gb of RAM.
The topology and data of Example 1 are presented in Fig. 1. We chose the rule $\delta_l = 60/l$ for (APL). The performance results are given in Table 1. Example 2 is similar and contains 20 nodes, 104 arcs, and 20 O/D pairs. Here, we only chose the rule $\delta_l = 1000/l$ for (APL). The results are given in Table 2. In the most cases, (APL) showed some preference over (PL) in the number of block iterations.

### 7.2 Smooth Optimization Test Problems

Next, we compared the performance of the usual conditional gradient method (CGM) from [4] and our method (APL), which is treated as its adaptive version, on some smooth optimization test problems where $h \equiv 0$. We took the even partition of $\mathbb{R}^N$. 

![Fig. 1 Example 1, 22 nodes, 12 O/D pairs](image-url)
### Table 3  The numbers of iterations (it) and partial gradients calculations (cl)

| $N$ | $n$ | (CGM) it | cl | (APL) it | cl |
|-----|-----|----------|----|----------|----|
| 10  | 5   | 15       | 75 | 9        | 28 |
| 20  | 5   | 50       | 250| 108      | 189|
| 50  | 5   | 143      | 715| 452      | 676|
| 100 | 5   | 257      | 1285|775       | 1161|
| 50  | 10  | 228      | 2280|632       | 1048|
| 100 | 10  | $\Delta_{500} = 0.11$ 500 | 5000| $\Delta_{1500} = 0.127$ 1500 | 2515|
| 80  | 20  | $\Delta_{500} = 0.3$ 500 | 10,000|766       | 1646|
| 100 | 20  | $\Delta_{500} = 0.367$ 500 | 10,000|1328      | 2820|
| 100 | 25  | $\Delta_{500} = 0.4$ 500 | 12,500|980       | 2346|
| 100 | 50  | $\Delta_{500} = 0.76$ 500 | 25,000|236       | 1036|

i.e., set $N_i = \frac{N}{n}$ for $i = 1, \ldots, n$. Next, each set $X_i$ was chosen to be the standard simplex in $\mathbb{R}^t$, i.e.,

$$X_i = \left\{ \mathbf{u} \in \mathbb{R}^t_+ : \sum_{i=1}^{t} u_i = 1 \right\}.$$

We took $\Delta_k = \varphi(x^k)$ as accuracy measure and chose the accuracy 0.1. We chose the same starting point $(1/t)e$, where $e$ denotes the vector of units in $\mathbb{R}^N$, and the rule $\delta_{l+1} = \nu \delta_l$ with $\nu = 0.5$ for (APL). The methods for these examples were implemented in Delphi with double-precision arithmetic. We report the number of iterations (it) and the total number of calculations (cl) of the partial gradients $g_i$ for attaining the desired accuracy.

In the first series, we took the convex quadratic cost function. We chose $\mu(\mathbf{x}) = f_1(\mathbf{x})$ where

$$f_1(\mathbf{x}) = 0.5 \langle P \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{q}, \mathbf{x} \rangle,$$

the elements of the matrix $P$ are defined by

$$p_{ij} = \begin{cases} 
\sin(i) \cos(j) & \text{if } i < j, \\
\sin(j) \cos(i) & \text{if } i > j, \\
\sum_{s \neq i} |p_{is}| + 1 & \text{if } i = j,
\end{cases}$$

and elements of the vector $\mathbf{q}$ are defined by $q_j = \sin(j)/j$ for all $i, j$. The results are given in Table 3. In the second series, we took the composite convex cost function

$$\mu(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}),$$
where $f_1$ was defined as above and

$$f_2(x) = 1/((c, x) + \tau),$$

where $c_i = 2 + \sin(i)$ for $i = 1, \ldots, N$ and $\tau = 5$. The results are given in Table 4. In almost all the cases, (APL) showed some preference over (CGM) in the number of partial gradients calculations. At the same time, tuning parameters of (APL) need further investigations.

### 8 Conclusions

We described a new adaptive component-wise method for decomposable composite optimization problems involving non-smooth functions, where the feasible set is a Cartesian product. The method combines component-wise steps with a special control of tolerance sequences. This rule gives significant freedom in selection of the component index, which provides the suitable descent property. We showed that this keeps the convergence properties of the usual PL one together with reduction of the total computational expenses. We describe several classes of significant applications for the new method. The preliminary results of computational tests showed rather satisfactory convergence.

Investigations of (APL) can be continued in several directions. It seems worthwhile to extend the results to the case of limit problems where only approximation sequences are known instead of exact values of the goal function and feasible set. Also, it would be useful to compare different index selection strategies in order to reveal efficient rules for choosing the parameters. Moreover, special adjustment of (APL) to significant
classes of applications such as network equilibrium problems needs more detailed investigations.

Acknowledgements  The results of this work were obtained within the state assignment of the Ministry of Science and Education of Russia, Project No. 1.460.2016/1.4. This work was also supported by the RFBR grant, Project No. 16-01-00109a. The author is grateful to Olga Pinyagina for her assistance in carrying out computational experiments.

References

1. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Nav. Res. Logist. Quart. 3, 95–110 (1956)
2. Levitin, E.S., Polyak, B.T.: Constrained minimization methods. USSR Comput. Maths. Math. Phys. 6, 1–50 (1966)
3. Dem’yanov, V.F., Rubinov, A.M.: Approximate Methods for Solving Extremum Problems. Leningrad University Press, Leningrad [Engl. transl. in Elsevier Science B.V., Amsterdam (1970)] (1968)
4. Pshenichnyi, B.N., Danilin, YuM: Numerical Methods in Extremal Problems. MIR, Moscow (1978)
5. Dunn, J.C.: Convergence rates for conditional gradient sequences generated by implicit step length rules. SIAM J. Control Optim. 18, 473–487 (1980)
6. Polyak, B.T.: Introduction to Optimization. Nauka, Moscow (1983) [Engl. transl. in Optimization Software, New York (1987)]
7. Clarkson, K.L.: Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. ACM Trans. Algol 6(4), 1–30 (2010)
8. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13), pp. 427–435 (2013)
9. Freund, R.M., Grigas, P.: New analysis and results for the Frank–Wolfe method. Math. Progr. 155, 199–230 (2016)
10. Facchinei, F., Sagratella, S., Scutari, G.: Flexible parallel algorithms for big data optimization. arXiv:1311.2444, pp. 1–8 (November 11, 2013)
11. Mine, H., Fukushima, M.: A minimization method for the sum of a convex function and a continuously differentiable function. J. Optim. Theory Appl. 33, 9–23 (1981)
12. Patriksson, M.: Cost approximation: a unified framework of descent algorithms for nonlinear programs. SIAM J. Optim. 8, 561–582 (1998)
13. Bredies, K., Lorenz, D.A., Maass, P.: A generalized conditional gradient method and its connection to an iterative shrinkage method. Comput. Optim. Appl. 42, 173–193 (2009)
14. Konnov, I.V.: Sequential threshold control in descent splitting methods for decomposable optimization problems. Optim. Meth. Softw. 30, 1238–1254 (2015)
15. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)
16. Patriksson, M.: Nonlinear Programming and Variational Inequality Problems: A Unified Approach. Kluwer Academic Publishers, Dordrecht (1999)
17. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. J. R. Stat. Soc. B 68, 49–67 (2006)
18. Meier, L., van de Geer, S., Bühlmann, P.: The group lasso for logistic regression. J. R. Statist. Soc. B 70, 53–71 (2008)
19. Scutari, G., Facchinei, F., Song, P., Palomar, D.P., Pang, J.-S.: Decomposition by partial linearization: parallel optimization of multi-agent systems. IEEE Trans. Signal Process. 62, 641–656 (2014)
20. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. 2, 121–167 (1998)
21. Agarwal, C.C.: Data Mining. Springer, Heidelberg (2015)
22. Seref, O., Kundakcioglu, O.E., Pardalos, P.M.: Selective linear and nonlinear classification. In: Pardalos, P.M., Hansen, P. (eds.) Data Mining and Mathematical Programming, pp. 211–234. American Mathematical Society, Providence (2008)
23. Magnanti, T.L.: Models and algorithms for predicting urban traffic equilibria. In: Florian, M. (ed.) Transportation Planning Models, pp. 153–185. North-Holland, Amsterdam (1984)
24. Lasdon, L.S.: Optimization Theory for Large Systems. Macmillan, New York (1970)
25. Mikhalevich, V.S., Ermol’ev, YuM, Shkurba, V.V., Shor, N.Z.: Complex systems and the solution of extremal problems. Kibernetika 3(5), 29–39 (1967)
26. Petersen, E.R.: A primal-dual traffic assignment algorithm. Manag. Sci. 22, 87–95 (1975)
27. Migdalas, A.: Cyclic linearization and decomposition of team game models. In: Butenko, S., Murphey, R., Pardalos, P. (eds.) Recent Developments in Cooperative Control and Optimization, pp. 332–348. Kluwer Academic Publishers, Dordrecht (2004)
28. Lacoste-Julien, S., Jaggi, M., Schmidt, M., Pletscher, P.: Block-coordinate Frank-Wolfe optimization for structural SVMs. International Conference on Machine Learning, Atlanta (2013)
29. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice-Hall, Upper Saddle River (1989)
30. Konnov, I.V.: Combined Relaxation Methods for Variational Inequalities. Springer, Berlin (2001)
31. Nagurney, A.: Comparative tests of multi-modal traffic equilibrium problems. Transp. Sci. B 18, 469–485 (1984)