<table>
<thead>
<tr>
<th>Title</th>
<th>Optimality Conditions and Algorithms for Semi-Infinite Programs with an Infinite Number of Second-Order Cone Constraints (The evolution of optimization models and algorithms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Okuno, Takayuki; Hayashi, Shunsuke; Fukushima, Masao</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (2011), 1726: 161-171</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2011-02</td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

Kyoto University
Optimality Conditions and Algorithms for Semi-Infinite Programs with an Infinite Number of Second-Order Cone Constraints

Takayuki Okuno, Shunsuke Hayashi, Masao Fukushima
Department of Applied Mathematics and Physics
Graduate School of Informatics, Kyoto University

Abstract

The semi-infinite program (SIP) is normally represented with infinitely many inequality constraints, and has been much studied so far. However, there have been very few studies on the SIP involving second-order cone (SOC) constraints, even though it has important applications such as Chebychev-like approximation and filter design.

In this paper, we focus on the SIP with a convex objective function and infinitely many SOC constraints, called the SISOCP for short. We show that, under a generalized Slater constraint qualification, an optimum of the SISOCP satisfies the KKT conditions that can be represented only with a finite subset of the SOC constraints. Next we introduce the regularization and the explicit exchange methods for solving the SISOCP. We first provide an explicit exchange method without a regularization technique, and show that it has global convergence under the strict convexity assumption on the objective function. Then we propose an algorithm combining a regularization method with the explicit exchange method. For the SISOCP, we establish global convergence of the hybrid algorithm without the strict convexity assumption.

1 Introduction

In this paper, we focus on the following semi-infinite program with an infinite number of second-order cone constraints:

\[
\begin{align*}
\text{Minimize} & \quad f(x) \\
\text{subject to} & \quad A(t)^{\top}x - b(t) \in \mathcal{K} \quad \text{for all } t \in T
\end{align*}
\]
where \( f : \mathbb{R}^n \to \mathbb{R} \) is a continuously differentiable convex function, \( A : \mathbb{R}^t \to \mathbb{R}^{n \times m} \) and \( b : \mathbb{R}^t \to \mathbb{R}^m \) are continuous functions, \( T \subseteq \mathbb{R}^t \) is a given compact set, and \( \mathcal{K}^m_i \subseteq \mathbb{R}^{m_i} \) \( (i = 1, 2, \ldots, s) \) is the second-order cone (SOC), that is, \( \mathcal{K} := \mathcal{K}^m_1 \times \mathcal{K}^m_2 \times \cdots \times \mathcal{K}^m_s \) with \( m = m_1 + m_2 + \cdots + m_s \) and

\[
\mathcal{K}^m_i := \begin{cases} 
\{ (x_1, \bar{x}^T) \in \mathbb{R} \times \mathbb{R}^{m_i-1} \mid x_1 \geq \| \bar{x} \| \} & (m_i > 1) \\
\mathbb{R}_+ = \{ x \in \mathbb{R} \mid x \geq 0 \} & (m_i = 1).
\end{cases}
\]

Throughout this paper, \( \| \cdot \| \) denotes the Euclidean norm defined by \( \| x \| := \sqrt{x^T x} \), and \( \tilde{v} \) denotes \( (v_2, v_3, \ldots, v_{m_i-1})^T \in \mathbb{R}^{m_i-1} \) for \( v = (v_1, v_2, \ldots, v_{m_i})^T \in \mathbb{R}^{m_i} \). For simplicity, we will often write \( (x_1, \bar{x})^T \) for \( (x_1, \bar{x}^T)^T \).

We call the problem (1.1) a semi-infinite second-order cone problem (SISOCP). One of typical applications for SISOCP (1.1) is a Chebychev-like approximation with vector-valued functions. Let \( Y \subseteq \mathbb{R}^n \) be a given compact set, and \( \Phi : Y \to \mathbb{R}^m \) and \( F : \mathbb{R}^t \times Y \to \mathbb{R}^m \) be given functions. Then, how can we determine a parameter \( u \in \mathbb{R}^t \) such that \( \Phi(y) \approx F(u, y) \) for all \( y \in Y \)? One relevant approach is to solve the following problem:

\[
\text{Minimize} \quad \max_{y \in Y} \| \Phi(y) - F(u, y) \|
\]

which can be rewritten as

\[
\text{Minimize} \quad r \\
\text{subject to} \quad \left( \Phi(y) - F(u, y) \right)^T \in \mathcal{K}^{m+1} \text{ for all } y \in Y
\]

by introducing the auxiliary variable \( r \in \mathbb{R} \). If \( F \) is affine with respect to \( u \), then the above problem reduces to SISOCP (1.1) with \( \mathcal{K} = \mathcal{K}^{m+1} \).

When \( m = 1 \) and \( \mathcal{K} = \mathbb{R}_+ \), SISOCP (1.1) is the classical semi-infinite program (SIP) \([3, 5, 8, 13, 15, 16]\), which has a wide application in engineering (e.g., the air pollution control, the robot trajectory planning, the stress of materials, etc.)\([8, 13]\). So far, many algorithms have been proposed for solving SIPs, such as the discretization method \([3]\), the local reduction based method \([4, 11, 18]\) and the exchange method \([5, 6, 16]\). The discretization method solves the relaxed SIP with \( T \) replaced by a finite set \( T^k \subseteq T \), and the sequence of index sets \( \{T^k\} \) is generated so that the distance\(^1\) from \( T^k \) to \( T \) converges to 0 as \( k \) goes to infinity. While this method is comprehensible and easy to implement, the computational cost tends to be huge since the cardinality of \( T^k \) grows unboundedly. In the local reduction based method, the infinite number of constraints in the SIP is rewritten as a finite number of constraints with implicit functions. Although the SIP can be reformulated as a finitely constrained optimization problem by this method, it is not possible in general to evaluate the implicit functions exactly in a direct manner. The exchange method solves a relaxed subproblem with \( T \) replaced by a finite subset \( T^k \subseteq T \). In

\(^1\)For two sets \( X \subseteq Y \), the distance from \( X \) to \( Y \) is defined as \( \text{dist}(X, Y) := \sup_{y \in Y} \inf_{x \in X} \| x - y \| \).
this method, $T^k$ is updated so that $T^{k+1} \subseteq T^k \cup \{t_1, t_2, \cdots, t_r\}$ with $\{t_1, t_2, \cdots, t_r\} \subseteq T \setminus T^k$.

Studies on the second-order cones (SOCs) have been advanced significantly in the last decade. One of the most popular problems associated with SOCs is the linear second-order cone program (LSOCP). The primal-dual interior-point method [1, 12] is well known as an effective algorithm for solving LSOCP, and some software packages implementing them [17, 19] have been produced. The nonlinear second-order cone program (NLSOCP) [9, 10, 20] is more complicated and not studied so much as LSOCP. The second-order cone complementarity problem (SOCCP) is another important problem involving SOCs. The Karush-Kuhn-Tucker conditions for LSOCP and NLSOCP are particularly represented as SOCCPs. The smoothing method [2, 7] is one of useful algorithms for solving SOCCP.

The main purpose of the paper is threefold. First, we provide the optimality conditions for SISOPC (1.1). The KKT conditions for SISOPC (1.1) could naturally be described by means of integration and Borel measure since $T$ is infinite. However, we show that, under Slater’s constraint qualification, the KKT conditions at the optimum can be represented by using a finite number of elements in $T$. Second, we propose an explicit exchange method for solving the SISOPC (1.1) and show its global convergence under the strict convexity of the objective function. Third, we propose an algorithm that can solve SISOPC (1.1) without the strict convexity. This algorithm is a hybrid of the explicit exchange method and the regularization method, which is known to be effective in handling ill-posed problems. With the help of regularization, a global convergence of the algorithm can be established for SISOPC (1.1) without the strict convexity. As the notation used in this paper, $\mathcal{K} \ni x \perp y \in \mathcal{K}$ denotes the SOC complementarity condition, that is, $x^\top y = 0$, $x \in \mathcal{K}$ and $y \in \mathcal{K}$.

2 Karush-Kuhn-Tucker Conditions

In this section, we show that the KKT conditions for SISOPC (1.1) can be represented with finitely many second-order cone constraints. We first introduce the Slater constraint qualification (SCQ).

**Definition 2.1 (SCQ).** We say that the Slater Constraint Qualification (SCQ) holds for SISOPC (1.1) if there exists a point $x_0 \in \mathbb{R}^n$ such that $A(t)^\top x_0 - b(t) \in \text{int} \mathcal{K}$ for all $t \in T$.

Under the SCQ, the following theorem holds.

**Theorem 2.2** (Theorem 2.12 [14]). Let $x^* \in \mathbb{R}^n$ be an optimum of SISOPC (1.1) and suppose that the SCQ holds for SISOPC (1.1). Then, there exist $p$ indices $t_1, t_2, \ldots, t_p \in T$ and Lagrangian multipliers $y^1, y^2, \ldots, y^p \in \mathbb{R}^m$ such that $p \leq n + 1$,

$$
\nabla f(x^*) - \sum_{i=1}^p A(t_i) y^i = 0,
$$

$$
\mathcal{K} \ni y^i \perp A(t_i)^\top x^* - b(t_i) \in \mathcal{K} \text{ for } i = 1, 2, \ldots, p.
$$
3 Explicit Exchange Method

In this section, we propose an explicit exchange method for solving SISOPC (1.1). Moreover, we show that the algorithm has a global convergence property under mild assumptions. The algorithm proposed in this section requires solving second-order cone programs (SOCP) with finitely many constraints as subproblems. Let SOCP $\text{SOCP}(T')$ denote the relaxed problem of SISOPC (1.1) with $T$ replaced by a finite subset $T' := \{t_1, t_2, \ldots, t_p\} \subseteq T$. Then, the SOCP$(T')$ can be formulated as follows:

\[
\begin{align*}
\text{Minimize} & \quad f(x) \\
\text{subject to} & \quad A(t_j)\top x - b(t_j) \in \mathcal{K} \quad (j = 1, 2, \ldots, p).
\end{align*}
\]

We suppose that the subproblem SOCP $\text{SOCP}(T')$ can be solved by any suitable existing algorithm. Let $\bar{x}$ be an optimal solution of SOCP$(T')$. Then, $\bar{x}$ satisfies the following KKT conditions under some constraint qualification [1, 12]:

\[
\begin{align*}
\nabla f(\bar{x}) - \sum_{t_j \in T'} A(t_j) y_{t_j} &= 0, \\
\mathcal{K} \ni y_{t_j} \perp A(t_j)\top \bar{x} - b(t_j) \in \mathcal{K} \quad (j = 1, 2, \ldots, p),
\end{align*}
\]

(3.1)

where $y_{t_j}$ is a Lagrange multiplier vector corresponding to the constraint $A(t_j)\top \bar{x} - b(t_j) \in \mathcal{K}$ for each $j$.

Now, we propose the following algorithm.

Algorithm 1 (Explicit exchange method)

Step 0. Choose a positive sequence $\{\gamma_k\} \subseteq \mathbb{R}_{++}$ such that $\lim_{k \to \infty} \gamma_k = 0$. Choose a finite subset $E^0 := \{t_1^0, \ldots, t_p^0\} \subseteq T$ and solve SOCP$(E^0)$ to obtain an optimal solution $x^0$. Set $k := 0$.

Step 1. Set $r := 0$, $T_0 := E^k$ and $v^0 := x^k$. Do the following (a)–(c):

(a) Find a $t^r_{\text{new}} \in T$ such that

\[
A(t^r_{\text{new}})\top v^r - b(t^r_{\text{new}}) \notin -\gamma_k e + \mathcal{K}.
\]

(3.2)

If such a $t^r_{\text{new}}$ does not exist, i.e.,

\[
A(t)\top v^r - b(t) \in -\gamma_k e + \mathcal{K}
\]

for any $t \in T$, then set $x^{k+1} := v^r$, $E^{k+1} := T_r$, and go to Step 2. Otherwise, let

\[
\overline{T}_{r+1} := T_r \cup \{t^r_{\text{new}}\},
\]

and go to (b).

(b) Solve SOCP$(\overline{T}_{r+1})$ to obtain an optimum $v_{r+1}$ and Lagrange multipliers $y^r_{t} \neq 0$, for $t \in \overline{T}_{r+1}$.

(c) Let $T_{r+1} := \{t \in \overline{T}_{r+1} \mid y^r_{t} \neq 0\}$. Set $r := r + 1$ and return to (a).
Step 2. If $\gamma_k$ is sufficiently small, terminate. Otherwise, set $k := k + 1$ and return to Step 1.

In Step 1-(a), $e \in \mathbb{R}^m$ is defined as $e := (e^1, e^2, \ldots, e^s)^T$ and $e^i := (1, 0, \ldots, 0)^T \in \mathcal{K}^m$. To verify (3.3), we have to solve a certain optimization problem on $T$ and check the nonnegativity of the optimal value. For a detail, see [14]. Since this problem is not necessarily convex, it is not easy to solve it. But, in this paper, we suppose that we can obtain a global optimum of this problem every Step 1. In Step 1-(b), SOCP($\overline{T}_{r+1}$) can be solved by applying an existing method such as the primal-dual interior point method, the regularized smoothing method, and so on [1, 2, 7, 10, 12]. In Step 1-(c), SOCP($T_{r+1}$) is obtained from SOCP($\overline{T}_{r+1}$) by removing only the constraints with zero Lagrange multipliers, then the optimal values of those two problems are equal. In addition, the feasible region of SOCP($\overline{T}_{r+1}$) is contained in that of SOCP($T_r$). Therefore, we have

$$V_P(T_0) \leq V_P(\overline{T}_1) = V_P(T_1) \leq \cdots \leq V_P(T_r) \leq V_P(\overline{T}_{r+1}) = V_P(T_{r+1}) \leq \cdots \leq V_P(T) < +\infty,$$

(3.4)

where $V_P(T')$ denotes the optimal value of SOCP($T'$).

For the proposed method to be well-defined, Step 1 have to terminate in finitely many iterations. To ensure this, we suppose the assumptions as follows:

**Assumption A.** i) Function $f$ is strictly convex over the feasible region of SISOPC (1.1). ii) In Step 1-(b) of Algorithm 1, SOCP($\overline{T}_{r+1}$) is solvable for each $r$. iii) A sequence generated $\{v^r\}$ in every Step 1 of Algorithm 1 is bounded.

Under the Assumption A, the following theorem holds.

**Theorem 3.1.** [14, Theorem 4.1] Let that Assumption A hold. Then, the inner iterations in every Step 1 of Algorithm 1 terminate finitely.

Moreover, we have the following theorems for the globally convergent property.

**Theorem 3.2.** [14, Theorem 4.2] Let Assumption A hold. Let $x^*$ be the optimum of SISOCP (1.1), and $\{x^k\}$ be the sequence generated by Algorithm 1. Then, it follows that

$$\lim_{k \rightarrow \infty} x^k = x^*.$$

## 4 Regularized Explicit Exchange Method

In the previous section, we proposed the explicit exchange method for SISOCP (1.1) and analyzed the convergence property. However, to ensure the global convergence, the strict convexity of the objective function was required (Assumption A). In this section, we propose a regularized explicit exchange method, and establish global convergence of the method without assuming the strict convexity.

---

2From the strictly convexity of $f$, SISOCP (1.1) has a unique solution.
Let $\epsilon$ be a positive number, and $T' := \{t_1, t_2, \ldots, t_p\}$ be a finite subset of $T$. The regularized explicit exchange method solves the following SOCP, denoted $\text{SOCP}(\epsilon, T')$, in each iteration.

\[
\begin{align*}
\text{SOCP}(\epsilon, T') & : \quad \text{Minimize} \quad \frac{1}{2}\epsilon\|x\|^2 + f(x) \\
& \quad \text{subject to} \quad A(t_j)^T x - b(t_j) \in \mathcal{K} \quad (j = 1, 2, \ldots, p).
\end{align*}
\]

When the function $f$ is convex, $\frac{1}{2}\epsilon\|x\|^2 + f(x)$ is strongly convex. Then, if we solve $\text{SOCP}(\epsilon_k, \overline{T}_{r+1})$ with $\epsilon_k > 0$ instead of $\text{SOCP}(\overline{T}_{r+1})$ in Step 1-(b) of Algorithm 1, it is ensured by Theorem 3.1 that the inner iterations terminate\(^3\) finitely. Moreover, by choosing positive sequences $\{\epsilon_k\}$ and $\{\gamma_k\}$ both converging to 0, the generated sequence is expected to converge to a solution of SISOCP (1.1). Now we propose the following algorithm for SISOCP (1.1).

**Algorithm 2 (Regularized Explicit Exchange Method)**

**Step 0.** Choose positive sequences $\{\gamma_k\} \subseteq \mathbb{R}_{++}$ and $\{\epsilon_k\} \subseteq \mathbb{R}_{++}$ such that $\lim_{k \to \infty} \gamma_k = 0$, $\lim_{k \to \infty} \epsilon_k = 0$ and $\gamma_k = O(\epsilon_k)$. Choose a finite subset $E^0 := \{t_1^0, \ldots, t_l^0\} \subseteq T$. Set $k := 0$.

**Step 1.** Set $r := 0$ and $T_0 := E^k$. Solve $\text{SOCP}(\epsilon_k, T_0)$ and let $v^0$ be an optimum. Do the following (a)–(c):

(a) Find $t^r_{\text{new}} \in T$ such that

\[
A(t^r_{\text{new}})^T v^r - b(t^r_{\text{new}}) \notin -\gamma_k e + \mathcal{K}. \tag{4.2}
\]

If such a $t^r_{\text{new}}$ does not exist, i.e.,

\[
A(t)^T v^r - b(t) \in -\gamma_k e + \mathcal{K} \tag{4.3}
\]

for any $t \in T$, then set $x^{k+1} := v^r$ and $E^{k+1} := T_r$, and go to Step 2. Otherwise, let

$\overline{T}_{r+1} := T_r \cup \{t^r_{\text{new}}\}$,

and go to (b).

(b) Solve $\text{SOCP}(\epsilon_k, \overline{T}_{r+1})$ to obtain an optimum $v^{r+1}$ and Lagrange multipliers $y_t^{r+1}$, for $t \in \overline{T}_{r+1}$.

(c) Let $T_{r+1} := \{t \in \overline{T}_{r+1} | y_t^{r+1} \neq 0\}$. Set $r := r + 1$ and return to (a).

**Step 2.** Both $\epsilon_k$ and $\gamma_k$ are sufficiently small, then terminate. Otherwise, set $k := k + 1$ and return to Step 1.

---

\(^3\)We can verify Assumption A as follows. Assumption A i) is obvious from the definition of strict and strong convexity. Assumption A ii) holds, since $\text{argmin}\{g(x) \mid x \in X\}$ is nonempty if $g$ is strongly convex and $X$ is closed. Assumption A iii) also holds, since the sequence $\{v^r\}$ generated in Step 1 is contained in the set $L := \{x \mid f(x) \leq V_P(T)\}$ from (3.4), and $L$ is bounded due to the strong convexity of $f$. 

The following theorem shows that a generated sequence globally converges to a solution of SISOCP (1.1).

**Theorem 4.1.** [14, Lemma 5.1, Thereom 5.2] Suppose that the SCQ holds for SISOCP (1.1). Let \( \{x^k\} \) be a sequence generated by Algorithm 2. Then, \( \{x^k\} \) is bounded, and every accumulation point of \( \{x^k\} \) solves SISOCP (1.1).

## 5 Numerical Experiment

In this section, we report some numerical results with Algorithm 2. The program was coded in Matlab 2008a and run on a machine with an Intel®Core2 Duo E6850 3.00GHz CPU and 4GB RAM. In the experiments, we solve the following SISOCP with a linear objective function:

\[
\begin{align*}
\text{Minimize} & \quad c^T x \\
\text{subject to} & \quad A(t)^T x - b(t) \in \mathcal{K} \quad (\forall t \in T)
\end{align*}
\]

(5.1)

with the index set \( T = [-1,1] \), \( c \in \mathbb{R}^{15} \), \( A_{ij}(t) := \alpha_{ij}^0 + \alpha_{ij}^1 t + \alpha_{ij}^2 t^2 + \alpha_{ij}^3 t^3 \) (\( i = 1,2,\ldots,15 \), \( j = 1,2,\ldots,30 \)) and \( b_j(t) := \beta_{j}^0 + \beta_{j}^1 t + \beta_{j}^2 t^2 + \beta_{j}^3 t^3 \) (\( j = 1,2,\ldots,30 \)). The second-order cone \( \mathcal{K} \subset \mathbb{R}^{30} \) is chosen to be one of the following four cases: (i) \( \mathcal{K} = \mathcal{K}^{30} \), (ii) \( \mathcal{K} = \mathcal{K}^{10} \times \mathcal{K}^{20} \), (iii) \( \mathcal{K} = (\mathcal{K}^{10})^3 = \mathcal{K}^{10} \times \mathcal{K}^{10} \times \mathcal{K}^{10} \), (iv) \( \mathcal{K} = (\mathcal{K}^{5})^6 = \mathcal{K}^{5} \times \mathcal{K}^{5} \times \mathcal{K}^{5} \times \mathcal{K}^{5} \times \mathcal{K}^{5} \times \mathcal{K}^{5} \). In (5.1), all components of \( c \in \mathbb{R}^{15} \) are chosen randomly from \([-2,2]\). \( \beta_{j}^0 \) and \( \beta_{j}^1 \) are determined so that \( (\beta_{1}^{0}, \beta_{2}^{0}, \ldots, \beta_{30}^{0})^T = e \in \mathbb{R}^{30} \), with \( e \) is defined as \( e = (e^1, e^2, \ldots, e^s)^T \in \mathcal{K}^{m_1} \times \cdots \times \mathcal{K}^{m_s} \) and \( e^j := (1,0,\ldots,0) \in \mathbb{R}^{m_j} \). In addition, \( \alpha_{ij}^0, \beta_{i}^j \) (\( i = 1,2,\ldots,15 \), \( j = 1,2,\ldots,30 \), \( k = 0,1,2,3 \), \( l = 1,2,3 \)) are chosen randomly from \([-2,2]\) so that the origin is an interior feasible point of (5.1). By using \( A(t) \) and \( b(t) \) generated in this way, (5.1) satisfies the SCQ.

In Step 0 of Algorithm 2, we set \( \{\varepsilon_k\} \) and \( \{\gamma_k\} \) such that \( \varepsilon_k = \gamma_k = 2^{-k} \) for each \( k \). Moreover, we choose 10 points \( t_0^1, t_1^1, \ldots, t_0^{10} \in T \) randomly, and set \( E^0 := \{t_0^0, t_1^0, \ldots, t_0^{10}\} \). In Step 1-(a), to find \( t_{\text{new}}^r \) satisfying (4.2), we first check whether or not (4.2) is satisfied at \( t = -1.0, -0.9, -0.8, \ldots, 0.9, 1.0 \). If we fail to find \( t_{\text{new}}^r \) among them, then we solve a certain nonconvex problem (For a detail, see [14]) and check whether or not its optimal value is nonnegative. In Step 1-(b), we solve SOCP(\( e_k, T_{r+1} \)) by the smoothing method [2, 7]. In Step 2, we terminate the algorithm if \( \max(\varepsilon_k, \gamma_k) \leq 10^{-6} \), which means that we always stop the algorithm when \( k = 20 \) since \( \varepsilon_{20} = \gamma_{20} = 2^{-20} < 10^{-6} \).

The obtained results are shown in Table 1, Table 2, Table 3 and Table 4, in which cpu(s), \( t_{\text{add}} \) and \( E_{\text{fin}} \) denote the CPU time in seconds, the cumulative number of times \( t_{\text{new}}^r \) is added to \( T \), in Step 1, and the value of \( E^k \) at the termination of the algorithm, respectively.

From the tables, we can observe that the computational cost tends to be higher as the number of SOCs in \( \mathcal{K} \) gets larger. For example, in the case of \( \mathcal{K} = \mathcal{K}^{30} \) (one SOC in \( \mathcal{K} \)), the cpu time is only 5 seconds or so, whereas it becomes around 20 seconds when \( \mathcal{K} = (\mathcal{K}^{5})^6 \) (six SOCs in \( \mathcal{K} \)). We also note that both \( t = -1 \) and 1 (the extreme points of \( T \)) belong to \( E_{\text{fin}} \) for all the test problems. This implies that
the constraints $A(t)^T x - b(t) \in \mathcal{K}$ with $t = -1$ and 1 are both active at the optimal solutions. For the problems with $E_{\text{fin}} = \{-1, 1\}$, the values of cpu(s) and $t_{\text{add}}$ seem relatively small. In fact, for such problems, the active sets at the optima can often be identified in a small number of iterations. On the other hand, if $E_{\text{fin}}$ has elements other than $-1$ or 1, then the values of cpu(s) and $t_{\text{add}}$ tend to be larger. Especially, problems A4, B3, C2 and D9 yield the largest values among the test problems with $\mathcal{K} = \mathcal{K}^{30}$, $\mathcal{K} = \mathcal{K}^{10} \times \mathcal{K}^{20}$, $\mathcal{K} = (\mathcal{K}^{10})^3$, and $\mathcal{K} = (\mathcal{K}^5)^6$, respectively. Indeed, for those four problems, $E_{\text{fin}}$ has the third element that could not be identified at an early stage of the iterations.

Table 1: Results for $\mathcal{K} = \mathcal{K}^{30}$

<table>
<thead>
<tr>
<th>problem</th>
<th>cpu(s)</th>
<th>$t_{\text{add}}$</th>
<th>$E_{\text{fin}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>4.78</td>
<td>3</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A2</td>
<td>5.27</td>
<td>3</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A3</td>
<td>5.70</td>
<td>3</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A4</td>
<td>8.39</td>
<td>7</td>
<td>{-1, 0.14, 1}</td>
</tr>
<tr>
<td>A5</td>
<td>3.94</td>
<td>3</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A6</td>
<td>4.51</td>
<td>6</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A7</td>
<td>4.99</td>
<td>3</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A8</td>
<td>4.70</td>
<td>3</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A9</td>
<td>4.52</td>
<td>4</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>A10</td>
<td>4.75</td>
<td>3</td>
<td>{-1, 1}</td>
</tr>
</tbody>
</table>

Table 2: Results for $\mathcal{K} = \mathcal{K}^{10} \times \mathcal{K}^{20}$

<table>
<thead>
<tr>
<th>problem</th>
<th>cpu(s)</th>
<th>$t_{\text{add}}$</th>
<th>$E_{\text{fin}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>9.79</td>
<td>7</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>B2</td>
<td>6.31</td>
<td>6</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>B3</td>
<td>10.02</td>
<td>15</td>
<td>{-1, 0.3, 1}</td>
</tr>
<tr>
<td>B4</td>
<td>6.90</td>
<td>3</td>
<td>{-1, -0.24, 1}</td>
</tr>
<tr>
<td>B5</td>
<td>6.63</td>
<td>6</td>
<td>{-1, 0.4, 1}</td>
</tr>
<tr>
<td>B6</td>
<td>8.23</td>
<td>8</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>B7</td>
<td>6.54</td>
<td>4</td>
<td>{-1, -0.14, 1}</td>
</tr>
<tr>
<td>B8</td>
<td>7.32</td>
<td>6</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>B9</td>
<td>6.78</td>
<td>2</td>
<td>{-1, 1}</td>
</tr>
<tr>
<td>B10</td>
<td>6.29</td>
<td>5</td>
<td>{-1, 1}</td>
</tr>
</tbody>
</table>
Table 3: Results for $\mathcal{K} = (\mathcal{K}^{10})^{3}$

<table>
<thead>
<tr>
<th>problem</th>
<th>cpu(s)</th>
<th>$t_{\text{add}}$</th>
<th>$E_{\text{fin}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>7.85</td>
<td>2</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>C2</td>
<td>15.69</td>
<td>7</td>
<td>${-1, 0.16, 1}$</td>
</tr>
<tr>
<td>C3</td>
<td>9.72</td>
<td>7</td>
<td>${-1, -0.38, 1}$</td>
</tr>
<tr>
<td>C4</td>
<td>10.16</td>
<td>3</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>C5</td>
<td>7.31</td>
<td>2</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>C6</td>
<td>7.76</td>
<td>5</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>C7</td>
<td>7.73</td>
<td>5</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>C8</td>
<td>8.97</td>
<td>4</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>C9</td>
<td>10.78</td>
<td>5</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>C10</td>
<td>7.23</td>
<td>2</td>
<td>${-1, 1}$</td>
</tr>
</tbody>
</table>

Table 4: Results for $\mathcal{K} = (\mathcal{K}^{5})^{6}$

<table>
<thead>
<tr>
<th>problem</th>
<th>cpu(s)</th>
<th>$t_{\text{add}}$</th>
<th>$E_{\text{fin}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>27.00</td>
<td>7</td>
<td>${-1, -0.58, 1}$</td>
</tr>
<tr>
<td>D2</td>
<td>20.10</td>
<td>14</td>
<td>${-1, 0.36, 1}$</td>
</tr>
<tr>
<td>D3</td>
<td>17.42</td>
<td>6</td>
<td>${-1, 0.90, 1}$</td>
</tr>
<tr>
<td>D4</td>
<td>26.16</td>
<td>6</td>
<td>${-1, -0.56, 1}$</td>
</tr>
<tr>
<td>D5</td>
<td>20.52</td>
<td>6</td>
<td>${-1, 0.07, 1}$</td>
</tr>
<tr>
<td>D6</td>
<td>27.57</td>
<td>13</td>
<td>${-1, 0.33, 1}$</td>
</tr>
<tr>
<td>D7</td>
<td>25.96</td>
<td>8</td>
<td>${-1, -0.73, 1}$</td>
</tr>
<tr>
<td>D8</td>
<td>22.64</td>
<td>4</td>
<td>${-1, 1}$</td>
</tr>
<tr>
<td>D9</td>
<td>65.15</td>
<td>26</td>
<td>${-1, -0.63, -0.41, -0.23, 0.25, 1}$</td>
</tr>
<tr>
<td>D10</td>
<td>20.12</td>
<td>8</td>
<td>${-1, 1}$</td>
</tr>
</tbody>
</table>

References


