IMMEDIATE SCHEDULE ADJUSTMENT AND SEMIDEFINITE RELAXATION

JINLING ZHAO AND WEI CHEN

School of Mathematics and Physics, University of Science and Technology Beijing
30 Xueyuan Road, Haidian District, Beijing 100083, China

SU ZHANG∗
Business School, Nankai University
94 Weijin Road, Nankai District, Tianjin 300071, China

(Communicated by Jie Sun)

Abstract. This paper considers the problem of temporary shortage of some resources within a project execution period. Mathematical models for two different cases of this problem are established. Semidefinite relaxation technique is applied to get immediate solvent of these models. Relationship between the models and their semidefinite relaxations is studied, and some numerical experiments are implemented, which show that these mathematical models are reasonable and feasible for practice, and semidefinite relaxation can efficiently solve the problem.

1. Introduction. It is well known that scheduling problems are notably hard combinatorial optimization problems. For example, the deterministic resource-constrained project scheduling problem (the RCPSP) is known to be NP-complete ([1]). Blazewicz et al. have shown that the RCPSP belongs to the class of the strongly NP-hard problems ([3]). However, the scheduling problems have a wide range of applications in practice such as construction, manufacturing, military operations, etc. Thus they have attracted attention. In decades of time, not only the classic scheduling problems such as job-shop scheduling or travelling salesman problems, but also stochastic resource-constrained project scheduling problems and other scheduling problems have been extensively studied (See for example, [6, 8, 10], and the references there in), since a mathematical model for the RCPSP was developed by Pritsker et al. ([13]). At the same time, semidefinite relaxation and randomization techniques have played important roles in successfully solving some combinatorial optimization problems and polynomial programming problems ([4, 7, 14], etc.).

In this paper, the problem of temporary shortage of some resources within a project execution period is considered. Given a schedule of a certain project, we

2010 Mathematics Subject Classification. Primary: 90C90; Secondary: 90B99.
Key words and phrases. Schedule adjustment, immediate, resource-constrained, semidefinite relaxation.

The first author is supported by National Natural Science Foundation of China No. 11101028, 11271206, and the Fundamental Research Funds for the Central Universities. The third author is supported by National Natural Science Foundation of China No. 11401322.

∗ Corresponding author: Su Zhang.
are to obtain an adjustment of the schedule in face of some unexpected temporary resource shortage. Scheduling problems and resource-constrained project scheduling problems have been well investigated. However, to the extent of our knowledge, this kind of schedule adjustment problem has been rarely considered in the literature. To solve this problem, of course we can implement some existing methodology to solve the resource-constrained scheduling problem and get a new schedule. But this may cost much more time. Therefore, we focus on getting an immediate solution.

The remainder of this paper is organized as follows: first, a detailed description of the problem is given in Section 2. Then in Section 3, mathematical models of two different cases are established under some certain assumptions. Section 4 presents the semidefinite relaxations of these models. After that in Section 5, some numerical experiments are implemented. Finally in Section 6, a conclusion is stated.

2. Problem description. An activity-on-node project network can be described by a directed graph $G(V,E)$, where $V = \{0, 1, ..., n, n+1\}$ denotes the set of activities of the project. Here, Activities 0 and $n+1$ stand for the start and the end of the project, respectively, which are dummy activities. The symbol $E$ represents the set of precedence relationships among the activities in $V$. And the set $K = \{1, 2, ..., m\}$ denotes the renewable resources that are required for the project to execute. Some other notations to be used are listed in Table 1.

| Symbol | Definition                                  |
|--------|---------------------------------------------|
| $\text{Pred}(i)$ | set of direct predecessors of Activity $i$ |
| $\text{Succ}(i)$ | set of direct successors of Activity $i$ |
| $d_i$ | processing time (or duration) of Activity $i$ |
| $s_i$ | start time of Activity $i$ according to the existing schedule |
| $f_i$ | completion time of Activity $i$ according to the existing schedule |
| $R_k$ | amount of originally available units of renewable resource $k$ in unit time |
| $r_{ik}$ | usage of Activity $i$ of renewable resource $k$ in unit time |
| $t_0$ | start time of the temporary shortage of resources |
| $T$ | lasting time of the resources shortage |
| $\Delta R_k$ | amount of decrement of resource $k$ in unit time |

We assume $s_0 = 0$, $d_0 = d_{n+1} = 0$.

In what follows, we assume that the execution schedule of a certain project $G(V,E)$ have been obtained. While the project is being executed, at time $t_0 > 0$, a temporary shortage of some resources occurs, and it will last for $T$ units of time. We also assume that the resources will be sufficiently replenished at the time $t_0 + T$. It will cost too much time to resolve this scheduling problem by some traditional methods unless $t_0$ is near to $f_{n+1}$ in the existing schedule. In this paper, we are interested in the case that $t_0 > 0$ is still far away from $f_{n+1}$, and we are aiming at a fast adjustment of the original schedule.

3. Mathematical modelling. In this part, we are to establish the mathematical model for the scheduling adjustment problem mentioned in the last section. At the beginning, we define some helpful index sets for convenience.

Let $\bar{I}$ denote the set of indices of activities that have been completed before $t_0$, i.e.,

$$\bar{I} := \{i \mid f_i \leq t_0, \ i = 1, \cdots, n+1\}.$$
Of course these activities can be ignored, since they will not affect the adjustment. Define $I_0$ to be the set of indices of ongoing activities at the time $t_0$, i.e.,

$$I_0 := \{ i \mid t_0 - d_i < s_i \leq t_0, \ i = 1, \ldots, n \},$$

and $I_1$ to be the set of indices of oncoming activities, i.e.,

$$I_1 := \{ i \mid t_0 < s_i < t_0 + T, \ i = 1, \ldots, n \},$$

which will be started between $t_0$ and $t_0 + T$ according to the existing schedule.

Let

$$d^r_i := \begin{cases} 
0, & \text{if } i \in \bar{I} \\
\max(s_i - t_0, t_i - d_i), & \text{if } i \in I_0 \\
d_i, & \text{otherwise}
\end{cases}$$

be the remaining duration of Activity $i$ from $t_0$.

### 3.1. One-batch-activity case

First we assume $T \leq \min\{\min_{i \in I_0} d^r_i, \min_{i \in I_1} (s_i - t_0 + d^r_i)\}$. Define $x_i$, $i \in I_0 \cup I_1$ as the decision variable, where $x_i = 1$ means that Activity $i$ will be delayed until $t_0 + T$, and $x_i = 0$ means that Activity $i$ will not be delayed. Moreover, for $i \in I_0 \cup I_1$, denote $p^r_i$ to be the maximal remaining time of the paths from Activity $i$ to $n + 1$, namely, for $i \in I_0$, $p^r_i$ is the maximal remaining time of the paths from Activity $i$ to $n + 1$ from the time $t_0$; and for $i \in I_1$, $p^r_i$ is the maximal remaining time of the paths from Activity $i$ to $n + 1$ from the start time $s_i$. Here, $p^r_i$ includes the remaining duration of Activity $i$, namely, $d^r_i$.

We want to minimize the remaining processing time of the project that may contain some delay. The mathematical model is as follows:

$$\begin{align*}
\min & \quad \max \left\{ \max_{i \in I_0} \{x_i T + p^r_i\}, \max_{i \in I_1} \{s_i - t_0 + x_i(t_0 + T - s_i) + p^r_i\} \right\} \\
\text{s.t.} & \quad \begin{cases} 
x_i = 0 \text{ or } 1, & i \in I_0 \cup I_1 \\
\sum_{i \in I_0} (1 - x_i)r_{ik} + \sum_{i \in I_1} (1 - x_i)r_{ik} \leq R_k - \Delta R_k, & k = 1, \ldots, m
\end{cases}
\end{align*}$$

which is a min-max 0-1 integer programming problem. It is equivalent to

$$\begin{align*}
\min & \quad t \\
\text{s.t.} & \quad \begin{cases} 
x_i T + p^r_i \leq t, & i \in I_0 \\
\sum_{i \in I_0} (1 - x_i)r_{ik} + \sum_{i \in I_1} (1 - x_i)r_{ik} \leq R_k - \Delta R_k, & k = 1, \ldots, m
\end{cases}
\end{align*}$$

Here, for $i \in I_0$, if Activity $i$ will be delayed until $t_0 + T$, then $x_i = 1$ and the corresponding maximal remaining time of the paths from Activity $i$ to $n + 1$ from the time $t_0$ is $T + p^r_i$; while if Activity $i$ will not be delayed, then $x_i = 0$ and the remaining time is still $p^r_i$. For $i \in I_1$, if Activity $i$ will be delayed and start at $t_0 + T$, then the maximal remaining time of the paths from Activity $i$ to $n + 1$ from $s_i$ is $t_0 + T - s_i + p^r_i$, since it will be delayed $t_0 + T - s_i$ units of time, and thus the remaining time from $t_0$ is $T + p^r_i$; while if Activity $i$ will not be delayed, then the remaining time from $t_0$ is $s_i - t_0 + p^r_i$.

In fact, noting that the start time of the activities in $I_0 \cup I_1$ need to be rescheduled to fit in the shortage of some resources, if we let $s'_i = \max\{t_0, s_i\}$, then we can
deal with them together. In this case, the modelling problem can be rewritten as
\[
\begin{align*}
\min \quad & t \\
\text{subject to} \quad & x_i T + (1 - x_i) (s_i - t_0) + p_i^r \leq t, \quad i \in I_0 \cup I_1 \\
& x_i (x_i - 1) = 0, \quad i \in I_0 \cup I_1 \\
& \sum_{i \in I_0 \cup I_1} (1 - x_i) r_{ik} \leq R_k - \Delta R_k, \quad k = 1, \ldots, m.
\end{align*}
\]

(3)

3.2. Two-batch-activity case. Now let us consider the case that the resource-shortage period will last longer. Let
\[
I_0 := \{i \mid t_0 - d_i < s_i \leq t_0, \quad i = 1, \ldots, n\},
\]
and \(I_1 := \{i \mid t_0 < s_i < t_0 + T, \quad i = 1, \ldots, n\}\).

We know that for \(i \in I_0\), its successors may be contained in \(I_1\), and for some \(i \in I_1\), its predecessors may be contained in \(I_0\). For convenience, we divide the indices set \(I_1\) into two parts:
\[
\begin{align*}
I_1^{(1)} := & \{i \mid I_1 \text{ | } f_j \leq t_0, \text{ for } \forall j \in \text{Pred}(i)\}, \\
I_1^{(2)} := & \{i \mid I_1 \text{ | } \exists j \in \text{Pred}(i), \text{ such that } f_j > t_0\}.
\end{align*}
\]

We can see that activities in \(I_1^{(1)}\) do not have any direct predecessor in \(I_0\).

In what follows, we assume \(t_0 + T \leq \min\{\min_{i \in I_1^{(1)}} f_i, \min_{j \in \text{Succ}(I_0)} f_j\}\) for simplicity. Here, \(\text{Succ}(I_0) := \{j \mid j \in \text{Succ}(i) \text{ for } i \in I_0\}\). In this case, we need to consider the activities in \(I_0\) and their direct successors starting at \(t \in (t_0, t_0 + T)\), and the activities in \(I_1^{(1)}\). Actually, under this assumption, \(I_1^{(2)} = \text{Succ}(I_0) \cap I_1\), containing all direct successors starting at \(t \in (t_0, t_0 + T)\) of activities in \(I_0\). Thus there are two batches of activities during \((t_0, t_0 + T)\) to be considered.

To establish the mathematical model for this case, we set \(x_i, \quad i \in I_0 \cup I_1\) where \(x_i = 1\) means Activity \(i\) will be delayed until \(t_0 + T\) (for \(i \in I_1^{(2)}\), it might be delayed after \(t_0 + T\) if one of its predecessors is delayed), and \(x_i = 0\) means Activity \(i\) will not be delayed.

Note that for an activity in \(I_1^{(2)}\), if a predecessor of it is decided to be delayed until \(t_0 + T\), then it will be delayed too; while if all predecessors of it are not delayed, then it may or may not be delayed. In other words, for an activity in \(I_0\), if it is decided to be delayed until \(t_0 + T\), then all its successors will be delayed; while if it is decided not to be delayed, then its successors in \(I_1^{(2)}\) may or may not be delayed. The above analysis leads to
\[
x_i \left( \sum_{j \in \text{Succ}(i) \cap I_1} (1 - x_j) \right) = 0, \quad i \in I_0.
\]

In fact, only activities in \(I_0\) that have a direct successor in \(I_1\) are required to satisfy this constraint.

Moreover, for \(i \in I_0 \cup I_1\), denote \(p_i^r\) to be the maximal remaining time of the paths from Activity \(i\) to \(n + 1\), namely, for \(i \in I_0\), \(p_i^r\) is the maximal remaining time of the paths from Activity \(i\) to \(n + 1\) from the time \(t_0\), and for \(i \in I_1\), \(p_i^r\) is the maximal remaining time of the paths from Activity \(i\) to \(n + 1\) from the start time \(s_i\), just as before.
At the same time, we also need to sort activities in $I_1$ in order of their start time. Let $h$ be the number of activities in the set $I_1$, and $i_j \in I_1, j = 1, \ldots, h$ satisfy $s_{i_j} \leq s_{i_{j+1}}$.

Then we can establish a model of scheduling adjusting for this two-batch activities case:

$$\begin{align*}
\min & \quad t \\
\text{s.t.} & \quad x_i T + p_i^r \leq t, \quad i \in I_0 \\
& \quad s_i - t_0 + x_i (t_0 + T - s_i) + p_i^r \leq t, \quad i \in I_1 \\
& \quad x_i (x_i - 1) = 0, \quad i \in I_0 \bigcup I_1 \\
& \quad x_i \left( \sum_{j \in \text{succ}(i) \cap I_1} (1 - x_j) \right) = 0, \quad i \in I_0 \\
& \quad \sum_{i \in I_0} (1 - x_i) r_{ij} \leq R_k - \Delta R_k, \quad k = 1, \ldots, m \\
& \quad \sum_{i \in I_0} (1 - x_i) r_{ik} + \sum_{l \leq j} [(1 - x_i) r_{ik} - \sum_{p \in I_p^j(l)} (1 - x_p) r_{pk}] \leq R_k - \Delta R_k, \\
& \quad i_j \in I_1, \quad j = 1, \ldots, h, \quad k = 1, \ldots, m
\end{align*}$$

(4)

Here, we set $I_p^j(l) := \{ p \in I_0 | s_{i_{l-1}} < f_p \leq s_{i_l} \}$ for $i_l \in I_1, l = 1, \ldots, h$ and $s_{i_0} = 0$. For illustration, the constraint $x_i \left( \sum_{j \in \text{succ}(i) \cap I_1} (1 - x_j) \right) = 0, \quad i \in I_0$ assures that if Activity $i \in I_0$ is delayed, then all its successors in $I_1$ will be delayed; the constraint $\sum_{i \in I_0} (1 - x_i) r_{ik} \leq R_k - \Delta R_k$ is to make sure that the resource constraints are satisfied at the time $t_0$; and the constraint $\sum_{i \in I_0} (1 - x_i) r_{ik} + \sum_{l \leq j} [(1 - x_i) r_{ik} - \sum_{p \in I_p^j(l)} (1 - x_p) r_{pk}] \leq R_k - \Delta R_k$ makes it sure that the resource constraints keep satisfied every time an activity in $I_1$ is added into execution.

The following conclusion holds for this model.

**Theorem 3.1.** For an activity $i \in I_0$ that has successors in $I_1$, if $x_i = 1$, or, if $x_i = 0$ and $x_j = 0$ for $\forall j \in \text{succ}(i) \cap I_1$, then

$$x_i T + p_i^r \geq s_j - t_0 + x_j (t_0 + T - s_j) + p_j^r, \quad \forall j \in \text{succ}(i) \cap I_1.$$  

**Proof.** By the definition of $p_i^r$, for $i \in I_0$, we have

$$p_i^r = \max_{j \in \text{succ}(i)} \{ s_j - t_0 + p_j^r \}.$$  

This leads to

$$p_i^r \geq \max_{j \in \text{succ}(i) \cap I_1} \{ s_j - t_0 + p_j^r \}.$$  

If $x_i = 1$, then $x_j = 1 \forall j \in \text{succ}(i) \cap I_1$ because of $x_i \left( \sum_{j \in \text{succ}(i) \cap I_1} (1 - x_j) \right) = 0$. Thus,

$$x_i T + p_i^r = T + p_i^r \geq T + \max_{j \in \text{succ}(i) \cap I_1} \{ s_j - t_0 + p_j^r \} \geq T + s_j - t_0 + \max_{j \in \text{succ}(i) \cap I_1} \{ s_j - t_0 + p_j^r \} \geq T + s_j - t_0 + x_j (t_0 + T - s_j) + p_j^r$$

holds for all $j \in \text{succ}(i) \cap I_1$, noting that $s_j > t_0$ for $j \in I_1$.

If $x_i = 0$ and $x_j = 0$ for $\forall j \in \text{succ}(i) \cap I_1$, then

$$x_i T + p_i^r = p_i^r \geq \max_{j \in \text{succ}(i) \cap I_1} \{ s_j - t_0 + p_j^r \} \geq s_j - t_0 + p_j^r = s_j - t_0 + x_j (t_0 + T - s_j) + p_j^r.$$  

This completes the proof. \hfill $\Box$
From this theorem, we know that for an activity \( i \in I_0 \) that has successors in \( I_1 \), if it is delayed, or, if it and all its successors in \( I_1 \) are not delayed, then it holds

\[
x_i T + p_i^f \geq s_j - t_0 + x_j(t_0 + T - s_j) + p_j^f, \forall j \in \text{Succ}(i) \cap I_1.
\]

However, if it is not delayed but some of its successors in \( I_1 \) are delayed, then it is possible that

\[
x_i T + p_i^f \leq s_j - t_0 + x_j(t_0 + T - s_j) + p_j^f, \text{ for some } j \in \text{Succ}(i) \cap I_1.
\]

Therefore, both constraints

\[
x_i T + p_i^f \leq t, \quad i \in I_0
\]

and

\[
s_i - t_0 + x_i(t_0 + T - s_i) + p_i^f \leq t, \quad i \in I_1
\]

are needed in the mathematical model.

Furthermore, we can also minimize the number of activities that will be delayed. Then the models (2) and (4) can be reformed as

\[
\begin{align*}
\min & \quad \alpha t + (1 - \alpha) \sum_{i \in I_0 \cup I_1} x_i \\
\text{s.t.} & \quad x_i T + p_i^f \leq t, \quad i \in I_0 \\
& \quad s_i - t_0 + x_i(t_0 + T - s_i) + p_i^f \leq t, \quad i \in I_1 \\
& \quad x_i(x_i - 1) = 0, \quad i \in I_0 \cup I_1 \\
& \quad \sum_{i \in I_0} (1 - x_i)r_{ik} + \sum_{i \in I_1} (1 - x_i)r_{ik} \leq R_k - \Delta R_k, \quad k = 1, \ldots, m
\end{align*}
\]

(5)

\[
\begin{align*}
\min & \quad \alpha t + (1 - \alpha) \sum_{i \in I_0 \cup I_1} x_i \\
\text{s.t.} & \quad x_i T + p_i^f \leq t, \quad i \in I_0 \\
& \quad s_i - t_0 + x_i(t_0 + T - s_i) + p_i^f \leq t, \quad i \in I_1 \\
& \quad x_i(x_i - 1) = 0, \quad i \in I_0 \cup I_1 \\
& \quad x_i(\sum_{j \in \text{Succ}(i) \cap I_1} (1 - x_j)) = 0, \quad i \in I_0 \\
& \quad \sum_{i \in I_0} (1 - x_i)r_{ik} \leq R_k - \Delta R_k, \quad k = 1, \ldots, m \\
& \quad \sum_{i \in I_0} (1 - x_i)r_{ik} + \sum_{i \in I_1} [(1 - x_i)r_{ik} - \sum_{p \in I'_0(l)} (1 - x_p)r_{pk}] \leq R_k - \Delta R_k, \\
& \quad i_j \in I_1, \quad j = 1, \ldots, h, \quad k = 1, \ldots, m
\end{align*}
\]

(6)

where \( \alpha \in (0, 1) \) is a weight related to the importance of time and the number of activities that will be delayed.

4. Semidefinite relaxation. The mathematical models in the previous section are mixed 0-1 integer programming problems, which are nonlinear and nonconvex. As is well-known, nonconvexity always brings much difficulty in solving the problems, especially when it is a large-scale problem. So we aim at obtaining a convex relaxation for the models in this part. The semidefinite relaxation (SDP relaxation, for short) technique is very useful in combinatorial optimization, nonconvex polynomial optimization, and many other fields ([7, 19, 20], etc.). Therefore, we are to get the SDP relaxations for these models.

Let \( x \) be a column vector with entries \( x_i, i \in I_0 \cup I_1 \). Set \( X = xx^T \), then \( X \) is a symmetric matrix, with \( \text{rank}(X) = 1 \). Then \( x_i(x_i - 1) = 0 \) results in \( X_{ii} = x_i \), and \( x_i(\sum_{j \in \text{Succ}(i) \cap I_1} (1 - x_j)) = 0 \) in (4) and (6) implies \( \sum_{j \in \text{Succ}(i) \cap I_1} (x_i - X_{ij}) = 0 \), in
which we are able to write the nonlinear constraints as linear ones. By the lifting procedure as in [7], we relax $X = xx^T$ to $X - xx^T \succeq 0$, which is in turn equivalent to $\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0$ by Schur Complement Lemma ([5, 21]) since $X - xx^T$ is Schur complement of 1 for this matrix. Here, $M \succeq 0$ means the matrix $M$ is positive semidefinite.

In linear algebra and matrix theory, the Schur complement is defined as follows. Suppose $A, B, C, D$ are respectively $p \times p, p \times q, q \times p, q \times q$ matrices, and $D$ is invertible. Let $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, then the Schur complement of the block $D$ of the matrix $M$ is $M/D := A - BD^{-1}C$, and the Schur complement of the block $A$ of the matrix $M$ is $M/A := D - CA^{-1}B$, if $A$ is invertible.

Lemma 4.1. (Schur Complement Lemma, [5, 21]) For any symmetric matrix $M$ of the form $M = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix}$, if $D \succ 0$, then $M \succeq 0$ if and only if $A - BD^{-1}B^T \succeq 0$.

Combining the above analysis with the fact that all the other constraints are linear, we get the following SDP relaxation for (2):

$$\begin{align*}
\min_t & \quad t \\
\text{s.t.} & \quad x_i^T + p_i^r \leq t, \quad i \in I_0 \\
& \quad s_i - t_0 + x_i(t_0 + T - s_i) + p_i^r \leq t, \quad i \in I_1 \\
& \quad \sum_{i \in I_0 \cup I_1} (1 - x_i)r_{ik} \leq R_k - \Delta R_k, \quad k = 1, \ldots, m \\
& \quad X_{ii} = x_i, \quad i \in I_0 \cup I_1
\end{align*}$$

(7)

And the SDP relaxation for (4) can be written as

$$\begin{align*}
\min_t & \quad t \\
\text{s.t.} & \quad x_i^T + p_i^r \leq t, \quad i \in I_0 \\
& \quad s_i - t_0 + x_i(t_0 + T - s_i) + p_i^r \leq t, \quad i \in I_1 \\
& \quad \sum_{i \in I_0} (1 - x_i)r_{ik} \leq R_k - \Delta R_k, \quad k = 1, \ldots, m \\
& \quad \sum_{i \in I_0} (1 - x_i)r_{ik} + \sum_{l \leq j} [(1 - x_i)r_{ik} - \sum_{p \in I_0(t)} (1 - x_p)r_{pk}] \leq R_k - \Delta R_k, \\
& \quad \sum_{j \in Succ(i) \cap I_1} (x_i - X_{ij}) = 0, \quad i \in I_0 \\
& \quad X_{ij} = x_i, \quad i \in I_0 \cup I_1 \\
& \quad \sum_{j \in Succ(i) \cap I_1} (x_i - X_{ij}) = 0, \quad i \in I_0
\end{align*}$$

(8)
Similarly, if we further want to minimize the number of activities that will be delayed, then the problems in (5) and (6) can be relaxed as

$$
\begin{align*}
\min \quad & \alpha t + (1 - \alpha) \sum_{i \in I_0 \cup I_1} x_i \\
\text{s.t.} \quad & \begin{cases}
x_i T + p^r_i \leq t, & i \in I_0 \\
s_i - t_0 + x_i(t_0 + T - s_i) + p^r_i \leq t, & i \in I_1 \\
\sum_{i \in I_0} (1 - x_i) r_{ik} \leq R_k - \Delta R_k, & k = 1, \ldots, m \\
\sum_{i \in I_0} (1 - x_i) r_{ik} + \sum_{j \leq i} [(1 - x_i) r_{ij,k} - \sum_{p \in I_p(t)} (1 - x_p) r_{pj,k}] \leq R_k - \Delta R_k, & i_j \in I_1, j = 1, \ldots, h, k = 1, \ldots, m \\
x_i, x_i, & i \in I_0 \cup I_1 \\
X_{ii} = x_i, & i \in I_0 \cup I_1 \\
\sum_{j \in \text{Succ}(i) \cap I_1} (x_i - X_{ij}) = 0, & i \in I_0
\end{cases}
\end{align*}
$$

and

$$
\begin{align*}
\min \quad & \alpha t + (1 - \alpha) \sum_{i \in I_0 \cup I_1} x_i \\
\text{s.t.} \quad & \begin{cases}
x_i T + p^r_i \leq t, & i \in I_0 \\
s_i - t_0 + x_i(t_0 + T - s_i) + p^r_i \leq t, & i \in I_1 \\
\sum_{i \in I_0} (1 - x_i) r_{ik} \leq R_k - \Delta R_k, & k = 1, \ldots, m \\
\sum_{i \in I_0} (1 - x_i) r_{ik} + \sum_{j \leq i} [(1 - x_i) r_{ij,k} - \sum_{p \in I_p(t)} (1 - x_p) r_{pj,k}] \leq R_k - \Delta R_k, & i_j \in I_1, j = 1, \ldots, h, k = 1, \ldots, m \\
x_i, x_i, & i \in I_0 \cup I_1 \\
X_{ii} = x_i, & i \in I_0 \cup I_1 \\
\sum_{j \in \text{Succ}(i) \cap I_1} (x_i - X_{ij}) = 0, & i \in I_0
\end{cases}
\end{align*}
$$

respectively.

Theorem 4.2. Let $v^*_1$, $v^*_2$, $v^*_3$ and $v^*_4$ be the optimal values of (2), (4), (5) and (6), respectively; and let $v^*_r$, $v^*_2$, $v^*_3$ and $v^*_4$ be the optimal values of (7), (8), (9) and (10), respectively. Then we have $v^*_r \leq v^*_i$, $i = 1, 2, 3, 4$.

Proof. Noting that $x$ is the column vector of entries $x_i$, $i \in I_0 \cup I_1$, let $X = xx^T$, then $X - xx^T \preceq 0$ holds. Thus by Lemma 4.1, we have $\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0$. Moreover, $x_i(x_i - 1) = 0$ means $X_{ii} = x_i^2 = x_i$, i.e., $\text{diag}(X) = x$, and $x_i(\sum_{j \in \text{Succ}(i) \cap I_1} (1 - x_j)) = 0$ implies $\sum_{j \in \text{Succ}(i) \cap I_1} (x_i - X_{ij}) = 0$. Thus, the feasible set of (2) is included in that of (7), the feasible set of (4) is included in that of (8), the feasible set of (5) is included in that of (9), and the feasible set of (6) is included in that of (10). So we have $v^*_r \leq v^*_i$, $i = 1, 2, 3, 4$, since these are minimizing problems.

For more theoretical analysis on semidefinite relaxation models, please refer to [2, 11, 15, 17, 18, 22].

5. Numerical experiments. In this section, we are to present some preliminary numerical results for the mathematical models and their SDP relaxations. All codes are written in Matlab R2014, and run on laptop computer (Intel(R) Core(TM) i5-3337u CPU(1.80GHz)). The software of SeDuMi ([12, 16]) is applied.
**Example 1.** Let $A_i, i = 1, \ldots, 12$ be activities in a project, and $A_0, A_{13}$ be dummy activities, as shown in Figure 1. $s_i$ is the starting time of activities according to the existing schedule of the project, $d_i$ is the duration of Activity $i$, and $r_{i1}, r_{i2}$ are the usage of Activity $i$ of resources 1 and 2 in unit time, respectively. Suppose this project faces insufficient supply of resource 1 from the time $t_0 = 4$, and this situation will last for 2 units of time, i.e., $T = 2$. Assume $R_1 = 8$ and $\Delta R_1 = 3$.

It is the one-batch-activity case since $T = 2$. It is easily seen that activities $A_3, A_4, A_5, A_6$ must be considered, and $d_3 = 3$, $d_4 = 5$, $d_5 = 3$, $d_6 = 2$ at the time $t_0 = 4$. We can solve Model (2) by applying the function ‘intlinprog’ in Optimization Toolbox of Matlab, since it is a mixed-integer linear programming problem. The solution is $x_3^* = 1, x_4^* = x_5^* = x_6^* = 0$, and $t^* = 11$. Here $t^*$ stands for the minimum processing time of the remain activities of the project from the time $t_0$, so the whole executing time of the project is still $t_0 + t^* = 15 = f_{13}$. Therefore, we can delay Activity 3, and the project will be completed on time.

If we solve its semidefinite relaxation (7) by SeDuMi, we get $x_3^* = x_5^* = x_6^* = 1, x_4^* = 0$, and $t^* = 11$. We see that Activities 3, 5 and 6 should be delayed, however, the whole project will still not be delayed.

Furthermore, if we also want to minimize the number of activities that will be delayed, then we can solve the relaxation problem (9). Set $\alpha = 0.5$, and we obtain $x_3^* = 1, x_4^* = x_5^* = x_6^* = 0$, and $t^* = 11$.

Let the period of resource shortage last longer, namely, $T = 4$, then it becomes the two-batch-activity case. Then activities $A_3, A_4, A_5, A_6, A_7, A_{10}$ must be considered. In this case, Models (4) and (6) can be applied. These two problems are still mixed integer programming problems, but not linear any more, thus we cannot use the function ‘intlinprog’ in Matlab to solve them. Solve their relaxations (8) and (10) by SeDuMi, and then get integers from the solution.

For (8) or equivalently (10) in the case of $\alpha = 1$, we get $x_3 = x_5 = x_6 = x_7 = x_{10} = 1, x_4 = 0$, and $t^* = 13$, which means all activities in the shortage period except Activity 4 will be delayed by 4 units of time, and the completion time of the whole project will be delayed by 2 units of time, since $t_0 + t^* = 17 > f_{13} = 15$.

To minimize the number of delayed activities in addition, we solve (10) with the parameter $\alpha = 0.5$. In this case, we can obtain $x_3 = x_7 = 1, x_4 = x_5 = x_6 = x_{10} = 0$, and $t^* = 13$. So only Activity 3 and its direct successor 7 will be delayed in the considered period. Of course, all successors of Activity 3 will be delayed by 4 units of time, and thereby the whole project will be delayed by 2 units of time.
Sometimes, we may not obtain integer solution by solving the relaxation problems. For the above example, here we simply get a integer solution from the solution of the SDP relaxation model (10) by the rounding method. However, in many cases this might not be a good choice. So we try some other method in the following example.

Example 2. Let $A_i, i = 1, \cdots, 25$ be activities in a project, and $A_0, A_{25}$ be dummy activities, as shown in Figure 2. $s_i$ is the starting time of activities according to the existing schedule of the project, $d_i$ is the duration of Activity $i$, and $r_{i1}, r_{i2}$ are the usage of Activity $i$ of resources 1 and 2 in unit time, respectively.

Suppose this project faces insufficient supply of resource 1 from the time $t_0 = 4$, and this situation will last for 5 units of time, i.e., $T = 5$. Assume $R_1 = 15$ and $\Delta R_1 = 8$.

It is easily seen that activities $A_4, A_7, A_8, A_{11}, A_9, A_{10}, A_{12}, A_{13}, A_{14}, A_{15}, A_{17}$ must be considered, and it is the two-batch case. In this case, Models (4) and (6) can be applied. These two problems are not linear mixed integer programming, thus we cannot apply the function ‘intlinprog’ in Matlab to solve them. So we solve their relaxations (8) and (10) by SeDuMi, and then get integers from the solution.

To minimize both the time delayed and the number of delayed activities, we first solve (10) with the parameter $\alpha = 0.5$. Let $x$ be the solution of this relaxation model, and $x_i$ be the $i$-th component of $x$. We design the following scheme to get a integer solution from the solution of the relaxation problem: if $x_i \geq 1/3, i \in \{1, \cdots, n\}$, we set $x_i = 1, i \in \{1, \cdots, n\}$; otherwise, we set $x_i = 0, i \in \{1, \cdots, n\}$. That is, we choose 1/3 as a threshold for getting an integer solution. So we call it the 1/3-method.

For this problem, we solve the relaxation model (10) first, and then by using the 1/3-Method, we can obtain $x_4^* = x_7^* = x_{10}^* = x_{11}^* = 0, x_8^* = x_9^* = x_{12}^* = x_{13}^* = x_{14}^* = x_{15}^* = x_{17}^* = 1$, and $t^* = 18$. So Activities 8, 9, 12, 13, 14, 15, 17 will be delayed in the considered period, and the other activities will not be delayed. Of course, all successors of Activity 17 will be delayed by 4 units of time, and thereby the whole project will be delayed by 4 units of time, since $t_0 + t^* = 22 > f_{25} = 18$.

**Figure 2.** Project Network Diagram in Example 2
For Example 2, while computing the relaxation model (10), we also compare the 1/3-method with the rounding method to get integer solution. For different choices of \(\Delta R_1\), we solve the relaxation (10) with \(\alpha = 0.5\), then get integer solution by these two methods separately. It is found that the integer solution obtained by the 1/3-method is better than that obtained by the rounding method, especially for these two methods separately. It is found that the integer solution obtained by the rounding method is better than that obtained by the 1/3-method per-

\[ \text{Procedure: IMMEDIATE SCHEDULE ADJUSTMENT AND SDP RELAXATION} \]

\[ \Delta R_k \quad 1/3-Method \quad \text{Rounding Method} \quad \text{SDR Opt. Val} \quad \text{Opt. Val} \quad \text{Delay Time} \]

| \(\Delta R_k\) | 1/3-Method | Rounding Method | SDR Opt. Val | Opt. Val | Delay Time |
|--------------|-------------|----------------|--------------|----------|------------|
| 2            | 15          | infeasible     | 15.0000      | 15       | 1          |
| 5            | 18          | 15             | 15.0000      | 18       | 1          |
| 8            | 19          | infeasible     | 15.4583      | 18       | 4          |
| 11           | 19          | 15             | 16.7083      | 19       | 5          |

For Example 2, while computing the relaxation model (10), we also compare the 1/3-method with the rounding method to get integer solution. For different choices of \(\Delta R_1\), we solve the relaxation (10) with \(\alpha = 0.5\), then get integer solution by these two methods separately. It is found that the integer solution obtained by the 1/3-method is better than that obtained by the rounding method, especially for these two methods separately. It is found that the integer solution obtained by the rounding method is better than that obtained by the 1/3-method per-

\[ \text{Procedure: IMMEDIATE SCHEDULE ADJUSTMENT AND SDP RELAXATION} \]

\[ \Delta R_k \quad 1/3-Method \quad \text{Rounding Method} \quad \text{SDR Opt. Val} \quad \text{Opt. Val} \quad \text{Delay Time} \]

| \(\Delta R_k\) | 1/3-Method | Rounding Method | SDR Opt. Val | Opt. Val | Delay Time |
|--------------|-------------|----------------|--------------|----------|------------|
| 2            | 15          | infeasible     | 15.0000      | 15       | 1          |
| 5            | 18          | 15             | 15.0000      | 18       | 1          |
| 8            | 19          | infeasible     | 15.4583      | 18       | 4          |
| 11           | 19          | 15             | 16.7083      | 19       | 5          |

Table 2. Comparing the 1/3-Method with the Rounding Method

\[ \begin{align*}
\min & \quad t \\
\text{s.t.} & \quad y_iT + 2p_i^c + T \leq 2t, \quad i \in I_0 \\
& \quad t_0 - s_i + y_i(t_0 + T - s_i) + 2p_i^c \leq 2t, \quad i \in I_1 \\
& \quad \sum_{i \in I_0} (1 - y_i)r_{ik} + \sum_{i \in I_0} [(1 - y_i)r_{ik} - \sum_{p \in I_i^i(t)} (1 - y_p)r_{pk}] \leq 2R_k - 2\Delta R_k, \\
& \quad i_j \in I_1, \quad j = 1, \ldots, h, \quad k = 1, \ldots, m \\
& \quad \left[ \begin{array}{c} Y \\ y \\ \mathbf{1} \end{array} \right] \succeq 0 \\
& \quad Y_{ii} = 1, \quad i \in I_0 \cup I_1 \\
& \quad \sum_{j \in \text{Succ}(i) \cap I_1} (y_i - Y_{ij}) + \sum_{j \in \text{Succ}(i) \cap I_1} (1 - y_j) = 0, \quad i \in I_0 \end{align*} \]

(11)

and the model (10) can be reformulated as

\[ \begin{align*}
\min & \quad dt + 1/2(1 - \alpha) \sum_{i \in I_0 \cup I_1} (y_i + 1) \\
\text{s.t.} & \quad y_iT + 2p_i^c + T \leq 2t, \quad i \in I_0 \\
& \quad t_0 - s_i + y_i(t_0 + T - s_i) + 2p_i^c \leq 2t, \quad i \in I_1 \\
& \quad \sum_{i \in I_0} (1 - y_i)r_{ik} + \sum_{i \in I_0} [(1 - y_i)r_{ik} - \sum_{p \in I_i^i(t)} (1 - y_p)r_{pk}] \leq 2R_k - 2\Delta R_k, \\
& \quad i_j \in I_1, \quad j = 1, \ldots, h, \quad k = 1, \ldots, m \\
& \quad \left[ \begin{array}{c} Y \\ y \\ \mathbf{1} \end{array} \right] \succeq 0 \\
& \quad Y_{ii} = 1, \quad i \in I_0 \cup I_1 \\
& \quad \sum_{j \in \text{Succ}(i) \cap I_1} (y_i - Y_{ij}) + \sum_{j \in \text{Succ}(i) \cap I_1} (1 - y_j) = 0, \quad i \in I_0 \end{align*} \]

(12)
Models (11) and (12) can also be applied to the two-batch-activity case. We can solve their relaxations (11) and (12) by SeDuMi, and then get integer solution from solution of these relaxation models. Comparing the solution from the model (11) with that from (8), we find that the optimal value \( t^* \) and the scheme of delayed activities are the same. This result also holds while comparing the model (12) with (10).

6. Conclusions. In this paper, we consider the problem of schedule adjustment in the case of temporary insufficient supply of some resources within a project execution period. We first establish mathematical models for two different cases, then modify these models to minimize the number of delayed activities furthermore. Among these models, (2) and (5) for the case of one-batch activities are mixed-integer linear programming problems, while (4) and (6) are nonlinear. Then semidefinite relaxation technique is applied to the models, aiming at getting immediate solvent of these models. Relationship between the models and their semidefinite relaxations has been studied. Some preliminary numerical experiments have been implemented in Matlab, which show that these mathematical models are feasible and effective for practical use, and semidefinite relaxation technique helps to solve this kind of problem efficiently. Further study on the semidefinite relaxation of these models, including equivalent conditions for the relaxation and the original model and how to get integer solution after solving the relaxation models, is still needed.

Acknowledgments. The authors thank the editor and the anonymous referees for their careful reading of the paper and valuable comments.

REFERENCES

[1] M. Bartusch, R. H. Mohring and F. J. Randermacher, Scheduling project networks with resource constraints and time windows, *Annals of Operations Research*, **16** (1988), 201–240.
[2] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization*, MOS-SIAM Series on Optimization, 2001.
[3] J. Blazewicz, J. K. Lenstra and A. H. G. Kan Rinnooy, Scheduling subject to resource constraints: Classification and complexity, *Discrete Applied Mathematics*, **5** (1983), 11–24.
[4] S. Boyd and L. Vandenberghe, Semidefinite programming relaxations of non-convex problems in control and combinatorial optimization, *Communications, Computation, Control, and Signal Processing*, Springer, 1997, 279–287.
[5] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, first edition, 2004.
[6] P. Brucker, Scheduling and constraint propagation, *Discrete Applied Mathematics*, **123** (2002), 227–256.
[7] M. Goemans and D. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, *J. Assoc. Comput. Mach.*, **42** (1995), 1115–1145.
[8] S. Hartmann and D. Briskorn, A survey of variants and extensions of the resource-constrained project scheduling problem, *European Journal of Operational Research*, **207** (2010), 1–14.
[9] D. Henrion, J. Lasserre and J. Loeberg, GloptiPoly 3: moments, optimization and semidefinite programming, *Optim. Methods Softw.*, **24** (2009), 761–779, [http://homepages.laas.fr/henrion/software/gloptipoly3](http://homepages.laas.fr/henrion/software/gloptipoly3).
[10] H. Li and N. K. Womer, Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming, *European Journal of Operational Research*, **246** (2015), 20–33.
[11] U. Malik, I. M. Jaimoukha, G. D. Halikias and S. K. Gungah, On the gap between the quadratic integer programming problem and its semidefinite relaxation, *Math. Program.*, **107** (2006), 505–515.
[12] I. Pólik, Addendum to the SeDuMi user guide version 1.1, http://sedumi.ie.lehigh.edu/?page_id=58, 2005.
[13] A. A. B. Pritsker, L. J. Watters and P. M. Wolfe, Multiproject scheduling with limited resources: A zero-one programming approach, Management Science, 16 (1969), 93–108.
[14] F. Rendl, Semidefinite relaxations for integer programming, 50 Years of Integer Programming 1958–2008, (2009), 687–726.
[15] N. Z. Shor, Quadratic optimization problems, Soviet. J. Comput. Systems Sci., 25 (1987), 1–11.
[16] J. F. Sturm, Using SeDuMi 1.02, a matlab toolbox for optimizer over smmetric cones, Optim. Methods Softw., 11/12 (1999), 625–653.
[17] X. Sun and R. Li, New progress in integer programming, Operations Research Transactions, 18 (2014), 39–67.
[18] L. Vandenberghe and S. Boyd, Semidefinite programming, SIAM Review, 38 (1996), 49–95.
[19] H. Waki, S. Kim, M. Kojima and M. Muramatsu, Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity, SIAM Journal on Optimization, 17 (2006), 218–242.
[20] H. Wolkowicz, R. Saigok and L. Vandenberghe, Handbook of Semidefinite Programming, Kluwer’s Publisher, 2000.
[21] F. Zhang, The Schur Complement and Its Applications: Numerical Methods and Algorithms, Springer Science Business Media, 2005.
[22] X. Zheng, X. Sun, D. Li and Y. Xia, Duality gap estimation of linear equality constraintd binary quadratic programming, Mathematics of Operations Research, 35 (2010), 864–880.

Received November 2017; revised January 2018.

E-mail address: j1zhao@ustb.edu.cn
E-mail address: chenwei940718@sina.com
E-mail address: zhangsu@nankai.edu.cn