Updated complexity results in single-machine primary-secondary scheduling for minimizing two regular criteria

Jinjiang Yuan (J.J. Yuan)
Zhengzhou University, P. R. China

June 9, 2021
Schedulingseminar.com
Abstract

In the primary-secondary scheduling problem, we have a primary scheduling criterion and a secondary scheduling criterion. The goal of the problem is to find a schedule which minimizes the secondary criterion, subject to the restriction that the primary criterion is minimized. In 1993, Lee and Vairaktarakis [LV1993] presented a comprehensive review for the computational complexity of the single-machine primary-secondary scheduling problems, where all the jobs are released at time zero. When both of the two criteria are regular, more than twenty problems were posed as open in [LV1993]. This talk will report the research progress of these open problems.

1 Schedules and criteria

We have \( n \) jobs \( \mathcal{J} = \{ J_1, J_2, \ldots, J_n \} \) to be scheduled on a single machine. Each job \( J_j \in \mathcal{J} \) has a processing time \( p_j \), a due date \( d_j \), and a weight \( w_j \). All the parameters \( p_j, d_j, w_j \) are nonnegative integers.

Since we only consider the classical scheduling problems, each scheduling criterion \( f \) is a function of the from

\[
    f = f(C_1, C_2, \ldots, C_n),
\]

where \( C_j \) is the completion time of job \( J_j \) for \( j = 1, 2, \ldots, n \).

A scheduling criterion \( f \) is called regular if \( f \) is nondecreasing in the completion times of the jobs.

In this report, we only consider the following regular criteria

\[
    f_{\text{max}}, L_{\text{max}}, \sum C_j, \sum U_j, \sum T_j, \sum w_j C_j, \sum w_j U_j, \sum w_j T_j.
\]
We assume that all the jobs are released at time 0. Then we only consider the schedules in which the jobs are consecutively scheduled without idle times.

As a result, a schedule $\sigma$ of $\mathcal{J}$ is denoted by

$$\sigma = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)}),$$

where, for each index $i \in \{1, 2, \ldots, n\}$, $J_{\sigma(i)}$ is the $i$-th job in $\sigma$.

For two distinct jobs $J_i$ and $J_j$, we use the notation $J_i \prec_\sigma J_j$ to indicate that $J_i$ is scheduled before $J_j$ in schedule $\sigma$. 
2 The primary-secondary scheduling problems

Let $f$ and $g$ be two regular scheduling criteria. The single-machine primary-secondary scheduling problem with $f$ being the primary criterion and $g$ being the secondary criterion is denoted by

$$1||\text{Lex}(f, g)$$

which aims to find a schedule $\sigma$ such that the secondary criterion $g(\sigma)$ is minimized under the constraint that the primary criterion $f(\sigma)$ is minimized.

Let $\Pi(f)$ be the set of the optimal schedules for the single-criterion problem $1||f$. Then a schedule $\sigma$ of $J$ is optimal for problem $1||\text{Lex}(f, g)$ if and only if

$$\begin{cases} 
\sigma \in \Pi(f), \\
g(\sigma) = \min\{g(\pi) : \pi \in \Pi(f)\}.
\end{cases}$$

For each optimal schedule $\sigma$ for problem $1||\text{Lex}(f, g)$, we call $(f(\sigma), g(\sigma))$ the optimal vector of the problem.
3 Computational complexity

When we consider the computational complexity, a scheduling problem is either polynomially solvable, or ordinary NP-hard, or unary NP-hard. Here, a problem is ordinary NP-hard if it is binary NP-hard and is solvable in pseudo-polynomial time.

A problem is called open if up to now we do not know any information about its complexity classification.

A problem is called E-open if up to now we only know partial information about its complexity classification, and so, the exact complexity is still open.

In particular, if an E-open problem is binary NP-hard (thus, the pseudo-polynomial solvability or unary NP-hardness is still unknown), we call the problem OU-open. For an OU-open problem, the remaining issue is to determine that it is ordinary NP-hard or unary NP-hard.
4 Open problems in [LV1993]

In 1993, Lee and Vairaktarakis [LV1993] presented a comprehensive review for the computational complexity of the single-machine primary-secondary scheduling problems $1||\text{Lex}(f, g)$ with

$$f, g \in \{f_{\text{max}}, L_{\text{max}}, \sum C_j, \sum w_j C_j, \sum U_j, \sum w_j U_j, \sum T_j, \sum w_j T_j\}.$$

According to different choices of the two criteria $f$ and $g$, the complexity status of all the problems at that time (before 1993) were reported in Lee and Vairaktarakis [LV1993], where more than twenty problems are still open or E-open at that time.

Without going into the details of these results, we only report the new achievements obtained these years for the open or E-open problems posed in [LV1993].

First, let us list these open or E-open problems.
(1) $1\|\text{Lex}(\sum w_j C_j, \sum T_j), \text{OU-open.}$
(2) $1\|\text{Lex}(\sum w_j C_j, \sum \hat{w}_j U_j), \text{OU-open.}$
(3) $1\|\text{Lex}(\sum w_j C_j, \sum w_j T_j), \text{open.}$
(4) $1\|\text{Lex}(\sum T_j, \sum w_j C_j), \text{OU-open.}$
(5) $1\|\text{Lex}(f_{\text{max}}, \sum U_j), \text{open.}$
(6) $1\|\text{Lex}(\sum U_j, f_{\text{max}}), \text{open.}$
(7) $1\|\text{Lex}(f_{\text{max}}, \sum w_j U_j), \text{OU-open.}$
(8) $1\|\text{Lex}(\sum w_j U_j, f_{\text{max}}), \text{OU-open.}$
(9) $1\|\text{Lex}(L_{\text{max}}, \sum T_j), \text{OU-open.}$
(10) $1\|\text{Lex}(\sum T_j, L_{\text{max}}), \text{OU-open.}$
(11) $1\|\text{Lex}(f_{\text{max}}, \sum T_j), \text{OU-open.}$
(12) $1\|\text{Lex}(\sum w_j U_j, \sum \hat{w}_j U_j), \text{OU-open.}$
(13) $1\|\text{Lex}(\sum U_j, \sum C_j), \text{open.}$
(14) $1\|\text{Lex}(\sum U_j, \sum T_j), \text{open.}$
(15) $1\|\text{Lex}(\sum T_j, \sum U_j), \text{OU-open.}$
(16) $1\|\text{Lex} (\sum T_j, f_{\text{max}})$, OU-open.

(17) $1\|\text{Lex} (\sum T_j, \sum C_j)$, OU-open.

(18) $1\|\text{Lex} (\sum T_j, \sum w_j U_j)$, OU-open.

(19) $1\|\text{Lex} (\sum T_j, \sum w_j T_j)$, OU-open.

(20) $1\|\text{Lex} (L_{\text{max}}, \sum U_j)$, open.

(21) $1\|\text{Lex} (\sum U_j, L_{\text{max}})$, open.

(22) $1\|\text{Lex} (L_{\text{max}}, \sum w_j U_j)$, OU-open.

(23) $1\|\text{Lex} (\sum w_j U_j, L_{\text{max}})$, OU-open.

(24) $1\|\text{Lex} (\sum w_j U_j, \sum C_j)$, OU-open.

(25) $1\|\text{Lex} (\sum w_j U_j, \sum T_j)$, OU-open.

Up to now, the complexity status of problems (1)-(15) have been addressed or partially addressed.

We will report on these results.
5 Problems (1)-(3)

(1) 1||$\text{Lex}(\sum w_j C_j, \sum T_j)$, OU-open.
(2) 1||$\text{Lex}(\sum w_j C_j, \sum \hat{w}_j U_j)$, OU-open.
(3) 1||$\text{Lex}(\sum w_j C_j, \sum w_j T_j)$, open.

Exact complexities of problems (1)-(3) are in fact implied in the early literature.

From Smith (1956), the unique strategy for solving problem 1||$\sum w_j C_j$ is to sequence the jobs in the WSPT (weighted shortest processing time) order, i.e., the nondecreasing order of the ratios $p_j/w_j$.

Thus, for every $f$, problem 1||$\text{Lex}(\sum w_j C_j, f)$ can be solved in the following way:

- First sequence the jobs by the WSPT order, which minimizes the primary criterion $\sum w_j C_j$.
- Then for each block of jobs with the same ratio $p_j/w_j$, reschedule the jobs by an optimal schedule for problem 1||$f$ to minimize the secondary criterion $f$. 
Lawler (1977) showed that problem $1|| \sum T_j$ is pseudo-polynomially solvable. Thus, problem (1), i.e., $1|| \text{Lex} (\sum w_j C_j, \sum T_j)$, is pseudo-polynomially solvable, and so, ordinary NP-hard.

For problem $1|| \sum w_j U_j$, Lawler and Moore (1969) presented an $O(nP)$-time algorithm and Sahni (1976) presented an $O(nW)$-time algorithm, where $P = \sum_{j=1}^{n} p_j$ and $W = \sum_{j=1}^{n} w_j$. Thus, problem (2), i.e., $1|| \text{Lex} (\sum w_j C_j, \sum \hat{w}_j U_j)$, is pseudo-polynomially solvable, and so, ordinary NP-hard.

Arkin and Roundy (1991) showed that the problem $1|w_j = \lambda p_j| \sum w_j T_j$ is binary NP-hard and solvable in pseudo-polynomial time. Thus, problem (3), i.e., $1|| \text{Lex} (\sum w_j C_j, \sum w_j T_j)$, is ordinary NP-hard.


6 Problem (4)

(4) $1||\text{Lex}(\sum T_j, \sum w_jC_j)$, OU-open.

Exact complexity of this problem is also implied in the early literature.

Lenstra et al. (1977) showed that the problem $1|\bar{d}_j|\sum w_jC_j$ is unary NP-hard, where $\bar{d}_j$ is the deadline of job $J_j$ which requires that $C_j \leq \bar{d}_j$ for every job $J_j$.

Let us consider a feasible instance $\mathcal{J}$ of problem $1|\bar{d}_j|\sum w_jC_j$.

By setting $d_j = \bar{d}_j$, it is clear that a schedule $\sigma$ of $\mathcal{J}$ is feasible (subject to the deadlines) if and only if $\sum T_j(\sigma) = 0$, i.e., $\sigma$ is optimal for problem $1||\sum T_j$.

Thus, problem $1|\bar{d}_j|\sum w_jC_j$ on feasible instances polynomially reduces to problem $1||\text{Lex}(\sum T_j, \sum w_jC_j)$.

This implies that problem (4), i.e., $1||\text{Lex}(\sum T_j, \sum w_jC_j)$, is unary NP-hard.
7 Problems (5)-(8)

(5) 1||Lex($f_{\text{max}}, \sum U_j$), open.
(6) 1||Lex($\sum U_j, f_{\text{max}}$), open.
(7) 1||Lex($f_{\text{max}}, \sum w_j U_j$), OU-open.
(8) 1||Lex($\sum w_j U_j, f_{\text{max}}$), OU-open.

The work of Yuan [Y2017] implies that all the problems (5)-(8) are unary NP-hard.

Next we only consider (5) and (6) since problems (7) and (8) are more general.

Reference:
Yuan [Y2017] showed that problem $1|\bar{d}_j| \sum U_j$ is unary NP-hard.

Let us consider a feasible instance $\mathcal{J}$ of problem $1|\bar{d}_j| \sum U_j$. By setting, for each time $t \geq 0$ and each index $j \in \{1, 2, \ldots, n\}$,

$$f_j(t) = \begin{cases} 0, & \text{if } t \leq \bar{d}_j, \\ +\infty, & \text{if } t > \bar{d}_j. \end{cases}$$

it is clear that a schedule $\sigma$ of $\mathcal{J}$ is feasible (subject to the deadlines) if and only if $f_{\text{max}}(\sigma) = 0$, i.e., $\sigma$ is optimal for problem $1||f_{\text{max}}$. Then the following statement can be observed.

- A schedule of $\mathcal{J}$ is optimal for problem $1|\bar{d}_j| \sum U_j$ if and only if it is optimal for problem $1||\text{Lex}(f_{\text{max}}, \sum U_j)$.

This statement means that problem $1|\bar{d}_j| \sum U_j$ polynomially reduces to problem $1||\text{Lex}(f_{\text{max}}, \sum U_j)$. Thus, problem (5), i.e., $1||\text{Lex}(f_{\text{max}}, \sum U_j)$, is also unary NP-hard.
Let us further consider a feasible instance $\mathcal{J}$ of problem $1|\bar{d}_j|\sum U_j$.

Let $U^*$ be the optimal value of the problem $1||\sum U_j$ on instance $\mathcal{J}$, without considering the deadline restriction.

Yuan [Y2017] also showed that the following decision problem is unary NP-complete.

**DECISION[1]:** Is there a feasible schedule $\sigma$ of instance $\mathcal{J}$ (subject to the deadlines) such that $\sum U_j(\sigma) = U^*$?

With $f_j(t) = \begin{cases} 0, & \text{if } t \leq \bar{d}_j, \\ +\infty, & \text{if } t > \bar{d}_j, \end{cases}$ we have the following statement.

- A schedule $\sigma$ of $\mathcal{J}$ is an YES-solution of DECISION[1] if and only if $\sigma$ is an optimal schedule for problem $1||\text{Lex}(\sum U_j, f_{\text{max}})$ with objective vector $(U^*, 0)$.

This statement means that DECISION[1] polynomially reduces to problem $1||\text{Lex}(\sum U_j, f_{\text{max}})$.

Thus, problem (6), i.e., $1||\text{Lex}(\sum U_j, f_{\text{max}})$, is also unary NP-hard.
8 Problems (9) and (10)

(9) $1||\text{Lex}(L_{\text{max}}, \sum T_j)$, OU-open.
(10) $1||\text{Lex}(\sum T_j, L_{\text{max}})$, OU-open.

The work of Koulamas and Kyparisis [KK2001] implies that problems (9) and (10) are ordinary NP-hard.

Reference:

From Lee and Vairaktarakis [LV1993], both (9) and (10) are binary NP-hard. We next show that both (9) and (10) are pseudo-polynomially solvable.
Problem 1\(|(d_j, \bar{d}_j)| \sum T_j\) was studied in Koulamas and Kyparisis [KK2001], where “\((d_j, \bar{d}_j)\)” in the \(\beta\)-field means that the jobs have agreeable due dates and deadlines, or equivalently, the jobs of \(J\) can be renumbered such that

\[
d_1 \leq d_2 \leq \cdots \leq d_n \text{ and } \bar{d}_1 \leq \bar{d}_2 \leq \cdots \leq \bar{d}_n.
\]

By establishing a Separation Theorem similar to that in Lawler (1977), the authors showed that problem 1\(|(d_j, \bar{d}_j)| \sum T_j\) is solvable in \(O(n^5 p_{\max})\) time which is pseudo-polynomial.

We use ALGORITHM[1] to denote the algorithm in Koulamas and Kyparisis [KK2001] for solving problem 1\(|(d_j, \bar{d}_j)| \sum T_j\).
To solve problem (9), i.e., $1||\text{Lex}(L_{\text{max}}, \sum T_j)$, we use the following procedure.

**PROCEDURE[1]:** For solving problem $1||\text{Lex}(L_{\text{max}}, \sum T_j)$ on instance $\mathcal{J}$.

- Solve the problem $1||L_{\text{max}}$ on instance $\mathcal{J}$ and let $L^*$ be its optimal value.
- Set $\bar{d}_j = d_j + L^*$ for $j = 1, 2, \ldots, n$. Let $\mathcal{J}'$ be the new instance with such deadlines.

Observe that the jobs have agreeable due dates and deadlines in $\mathcal{J}'$.
- Run ALGORITHM[1] to solve the problem $1|(d_j, \bar{d}_j)| \sum T_j$ on instance $\mathcal{J}'$ and let $\sigma$ be its optimal schedule.

It is easy to see that the schedule $\sigma$ returned by PROCEDURE[1] is also optimal for problem $1||\text{Lex}(L_{\text{max}}, \sum T_j)$ on instance $\mathcal{J}$.

Thus, problem (9), i.e., $1||\text{Lex}(L_{\text{max}}, \sum T_j)$, is solvable in pseudo-polynomial $O(n^5 p_{\text{max}})$ time.
Now we consider problem (10), i.e., $1\|\text{Lex}(\sum T_j, L_{\max})$.

Again, let $L^*$ be the optimal value of problem $1\|L_{\max}$ on instance $\mathcal{J}$.

Let $(T', L')$ be the optimal vector of problem $1\|\text{Lex}(\sum T_j, L_{\max})$ on instance $\mathcal{J}$.

$T'$ can be obtained by solving problem $1\|\sum T_j$ on instance $\mathcal{J}$.

The remaining issue is to determine the value $L'$.

- It is obvious that
  \[ L' \in \{L^*, L^* + 1, \ldots, L^* + P\}, \]
  where $P = p(\mathcal{J})$ is the total processing time of the jobs of $\mathcal{J}$.

Thus, for each $\tau \in \{0, 1, \ldots, P\}$, we define

\[
\begin{align*}
L^{(\tau)} &= L^* + \tau, \\
\bar{d}_{j}^{(\tau)} &= d_j + L^{(\tau)}, \text{ for } j = 1, 2, \ldots, n,
\end{align*}
\]

and use $\mathcal{J}^{(\tau)}$ to denote the instance (induced from $\mathcal{J}$) with deadlines $\bar{d}_{j}^{(\tau)}$.

Note that the jobs have agreeable due dates and deadlines in instance $\mathcal{J}^{(\tau)}$. 
Suppose that $L' = L^{(\tau')} = L^* + \tau'$ for some $\tau' \in \{0, 1, \ldots, P\}$.

For each $\tau \in \{0, 1, \ldots, P\}$, we use $T^{(\tau)}$ to denote the optimal value of the problem $1|(d_j, \bar{d}_j)| \sum T_j$ on instance $\mathcal{J}^{(\tau)}$.

It is clear that $T^{(\tau)} \geq T'$ for all $\tau \in \{0, 1, \ldots, P\}$.

We have the following statement for $\tau'$.

- $\tau'$ is the minimum value of $\tau \in \{0, 1, \ldots, P\}$ such that $T^{(\tau)} = T'$, i.e., the optimal value of the problem $1|(d_j, \bar{d}_j)| \sum T_j$ on instance $\mathcal{J}^{(\tau)}$ is $T'$.

For an integer $\tau \in \{0, 1, \ldots, P\}$,

- if $T^{(\tau)} = T'$, we know that $\tau' \leq \tau$;
- if $T^{(\tau)} > T'$, we know that $\tau' > \tau$.

Thus, $\tau'$ can be determined by binary search on $\tau \in \{0, 1, \ldots, P\}$ with the decision “$T^{(\tau)} = T'$ or not” being answered by applying ALGORITHM[1] for solving problem $1|(d_j, \bar{d}_j)| \sum T_j$ on instance $\mathcal{J}^{(\tau)}$. 
We finally observe that

- A schedule of $\mathcal{J}$ is optimal for problem $1||\text{Lex}(\sum T_j, L_{\text{max}})$ if and only if it is optimal for the problem $1||(d_j, \bar{d}_j)|\sum T_j$ on instance $\mathcal{J}^{(\tau')}$. 

As a result, problem $1||\text{Lex}(\sum T_j, L_{\text{max}})$ can be solved by the following procedure.

**PROCEDURE[2]:** For solving problem $1||\text{Lex}(\sum T_j, L_{\text{max}})$ on instance $\mathcal{J}$.

- Determine the value $T'$ by solving problem $1||\sum T_j$ on instance $\mathcal{J}$.
- Apply binary search for $\tau \in \{0, 1, \ldots, P\}$ to determine the value $\tau'$, where we need to solve $O(\log P)$ problems $1||(d_j, \bar{d}_j)|\sum T_j$ on instance $\mathcal{J}^{(\tau)}$ for the picked values $\tau$, each problem is solved by using ALGORITHM[1] in $O(n^5 p_{\text{max}})$ time.
- Set $L' = L^* + \tau'$. Output the optimal vector $(T', L')$.

Thus, problem (10), i.e., $1||\text{Lex}(\sum T_j, L_{\text{max}})$, is solvable in $O(n^5 p_{\text{max}} \log P)$ time.
9 Problem (11)

(11) 1||\text{Lex}(f_{\text{max}}, \sum T_j), \text{OU-open.}

The work of Chen and Yuan [CY2019] implies that this problem is unary NP-hard.

Reference:

Chen and Yuan [CY2019] showed that problem 1|\bar{d}_j|\sum T_j is unary NP-hard.

Again, by setting, for each time $t$ and each index $j$, $f_j(t) = \begin{cases} 0, & \text{if } t \leq \bar{d}_j, \\ +\infty, & \text{if } t > \bar{d}_j, \end{cases}$

we see that the problem 1|\bar{d}_j|\sum T_j on feasible instances polynomially reduces to the problem 1||\text{Lex}(f_{\text{max}}, \sum T_j).

Thus, problem (11), i.e., 1||\text{Lex}(f_{\text{max}}, \sum T_j), is also unary NP-hard.
10 Problem (12)

(12) 1||Lex(\(\sum w_j U_j, \sum \hat{w}_j U_j\)), OU-open.
A work of Agnetis et al. [ABGPS2014] implies that this problem is ordinary NP-hard.

Reference:

In Agnetis et al. [ABGPS2014], the authors showed that the constraint problem 1|| \(\sum \hat{w}_j U_j : \sum w_j U_j \leq Q\) is solvable in pseudo-polynomial time.

By setting \(Q\) to be the optimal value of problem 1|| \(\sum w_j U_j\), which can be obtained in \(O(nP)\) time, we see that problem (12), i.e., 1||Lex(\(\sum w_j U_j, \sum \hat{w}_j U_j\)), is pseudo-polynomially solvable, and so, ordinary NP-hard.
11 Problems (13) and (14)

(13) $1\|\text{Lex}(\sum U_j, \sum C_j)$, open.

(14) $1\|\text{Lex}(\sum U_j, \sum T_j)$, open.

Complexities of the two problems were updated by Huo et al. [HLZ2007].

Reference:


Huo et al. [HLZ2007] showed that problems (13) and (14) are binary NP-hard.

By our knowledge, the exact complexity (pseudo-polynomially solvable, or unary NP-hard) of any of the two problems is still unaddressed.

Thus, problems (13) and (14) are OU-open now.

Conjecture 1. Problems (13) and (14) are unary NP-hard.
12 Problem (15)

(15) 1||Lex(∑ \( T_j \), ∑ \( U_j \)), OU-open.

Recently, Yuan and Zhao [YZ2021] showed that this problem is pseudo-polynomially solvable, and so, ordinary NP-hard.

Reference:

Recall that Lawler (1977) presented an \( O(n^5p_{\text{max}}) \)-time algorithm for solving problem 1||∑ \( T_j \) based on the following separation theorem.

The Separation Theorem: Suppose that \( d_1 \leq d_2 \leq \cdots \leq d_n \) and let \( J_j \) be a job of the longest processing time. Then there is an index \( k \in \{j, j + 1, \ldots, n\} \) and there is an optimal schedule \( \sigma \) in which the jobs are scheduled in the order

\[
\{J_1, J_2, \ldots, J_k\} \setminus \{J_j\} \prec_\sigma J_j \prec_\sigma \{J_{k+1}, J_{k+2}, \ldots, J_n\}.
\]
Very accidentally, we found the following modified separation theorem for problem 1||Lex(∑T_j, ∑U_j).

**The Modified Separation Theorem:** Suppose that \(d_1 \leq d_2 \leq \cdots \leq d_n\), where “\(d_i = d_j\) and \(p_i < p_j\)” will lead to \(i < j\). Let \(J_j\) be a job such that \(J_j\) has the longest processing time, and subject to this condition, \(d_j\) is as small as possible. Then there is an index \(k \in \{j, j+1, \ldots, n\}\) and there is an optimal schedule \(\sigma\) in which the jobs are scheduled in the order

\[
\{J_1, J_2, \ldots, J_k\} \setminus \{J_j\} \prec_{\sigma} J_j \prec_{\sigma} \{J_{k+1}, J_{k+2}, \ldots, J_n\}.
\]

With this modified separation theorem in hand, by using the similar procedure as that in Lawler (1977), we present an \(O(n^5p_{\text{max}})\)-time pseudo-polynomial algorithm for solving problem 1||Lex(∑T_j, ∑U_j).

Thus, problem (15) is ordinary NP-hard.
13 Problems (16)-(25)

There is no progress on these problems.

(16) $1||\text{Lex}(\sum T_j, f_{\text{max}})$, OU-open.
(17) $1||\text{Lex}(\sum T_j, \sum C_j)$, OU-open.
(18) $1||\text{Lex}(\sum T_j, \sum w_j U_j)$, OU-open.
(19) $1||\text{Lex}(\sum T_j, \sum w_j T_j)$, OU-open.

**Conjecture 2.** Problems (16)-(19) are pseudo-polynomially solvable, possibly based on some new separation theorems together with some new techniques.

- But simple separation theorems as that in Yuan and Zhao [YZ2021] do not exist.

(20) $1||\text{Lex}(L_{\text{max}}, \sum U_j)$, open.
(21) $1||\text{Lex}(\sum U_j, L_{\text{max}})$, open.

**Conjecture 3.** Problems (20) and (21) are polynomially solvable.
(22) \( 1\|\text{Lex}(L_{\text{max}}, \sum w_j U_j), \text{OU-open} \).
(23) \( 1\|\text{Lex}(\sum w_j U_j, L_{\text{max}}), \text{OU-open} \).

**Conjecture 4.** Problems (22) and (23) are pseudo-polynomially solvable.

(24) \( 1\|\text{Lex}(\sum w_j U_j, \sum C_j), \text{OU-open} \).
(25) \( 1\|\text{Lex}(\sum w_j U_j, \sum T_j), \text{OU-open} \).

We have conjectured that problems (13) and (14), i.e., \( 1\|\text{Lex}(\sum U_j, \sum C_j) \) and \( 1\|\text{Lex}(\sum U_j, \sum T_j) \), are unary NP-hard.
Thus, we also have the following conjecture.

**Conjecture 5.** Problems (24) and (25) are unary NP-hard.
Thank You!