On the Parameterized Tractability of Machine Scheduling Problems

Dvir Shabtay
Department of Industrial Engineering and Management
Ben Gurion University of the Negev, Israel

schedulingseminar. com

Parameterized Complexity

- The theory of parameterized complexity is a branch of the theory of computational complexity developed by the computer science community at the end of the 90's.
- It deals with the tractability of NP-hard problems with respect to their natural parameters.
  - i.e., it deals with the question whether an NP-hard problem becomes tractable when a subset of its parameters is of a limited size.

Motivation in Scheduling

- Consider for example the classical $1| \quad |\sum T_j$ problem.
  - Instance:
    - n - # of jobs to be scheduled;
    - p_j – the processing time of job J_j (j=1,…,n);
    - d_j– the due date of job J_j (j=1,…,n).
  - Objective:
    - Determine a schedule (job processing permutation) that minimizes $\sum T_j$, where T_j=max{0,C_j-d_j}, and C_j is the completion time of job J_j.

Motivation in Scheduling

- This problem is NP-hard in general (Du and Leung (1990)).

However

- In many real-life instances, the value of at least one of the following parameters is bounded:
  - The number of different processing times, v_p.
  - The number of different due-dates, v_d.

Motivation in Scheduling

- The value of the first parameter, ν_p, is bounded when only a limited number of different products is produced in the shop.

- The value of the second parameter, ν_d , is bounded when shipment cost is high, and therefore only few different due dates are assigned to the jobs.

Motivation in Scheduling

- Although the problem is NP-hard in general, it is well-known to be solvable in polynomial time when:
  - All processing times are equal (ν_p=1);
  - All due dates are equal (ν_d=1).

- A natural question:
  - Is the $1| \quad |\sum T_j$ problem solvable in polynomial time when the value of ν_p (or ν_d) is upper bounded by a constant?

Motivation in Scheduling

- The answer is…

YES

- We can design a quite simple O(n^k) time algorithm ($k \in \{v_p, v_d\}$) to solve the $1|\;|\sum T_j$ problem (using DP).

- The main question in parameterized complexity:

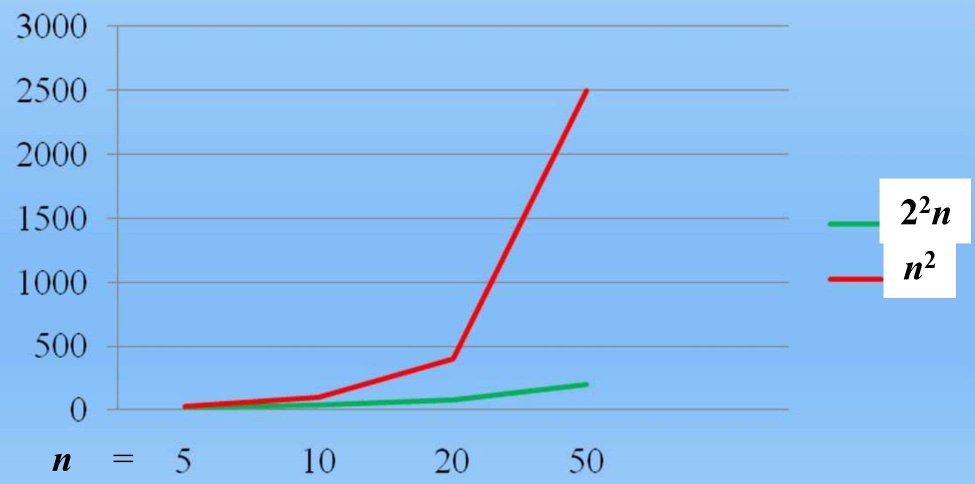
$\boxed{\text{Can we make the exponent of } n \text{ independent of } k\text{ ?}}$

  – e.g., $2^{O(k)}\, n^3$, or more generally $f(k)\, n^{O(1)}$?

Fixed Parameterized Complexity

A new battle between "good" and "bad" algorithms.

	"Good"	"Bad"
"Classical"	$n^{O(1)}$	$2^{n^{O(1)}}$
"Parameterized"	$f(k)\, n^{O(1)}$	$n^{f(k)}$



– The difference between "good" $2^k\, n$ and "bad" n^k:

– graph plot for k =2.

Parameterized Complexity

- Definition 1: Problem π belongs to the fixed-parameter tractable (FPT) set, *wrt*. parameter k, if there exists an algorithm that solves any instance of π in $f(k)n^{O(1)}$ time, for some computable f function that solely depends on k.

- Definition 2: Problem π belongs to the *XP* set, *wrt*. parameter k, if there exists an algorithm that solves any instance of π in $n^{f(k)}$ time.

Parameterized Complexity

- FPT$\subseteq$XP.

Hardness Proofs

Given problem π and a parameter k:

- If π is NP-hard for a constant value of k, then (unless P=NP) it cannot be solved in XP time wrt. k.

Parameterized Complexity

- **Definition 3**: A decision problem π is W[i]-hard *wrt*. parameter k if π being FPT with respect to k leads to that all problems in W[i] are FPT as well (which is believed to be very unlikely).

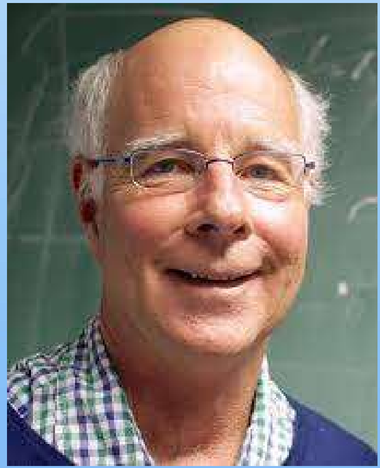
Parameterized Complexity

- To prove that a problem is W[i]-hard we can provide a parametrized reduction from a known W[i]-hard problem.
- An example for a problem that is known to be W[1]-hard:

<u>**k-sum problem**</u>: Given a set **A**=$\{a_1,\ldots,a_n\}$ of integers. Is there a subset of <u>exactly k</u> elements of **A** that adds up to a specific target.

History
Parametrized Complexity

- **Late 80's**: The development of FPT theory by **Rodney Downey** and **Michael Fellows** .



Downey, R., and Fellows, M., 1999, *Parameterized Complexity*. Springer, Berlin.

History

Parametrized Complexity

- **Ever Since**: It is a well-established area with hundreds of articles published every year in the most prestigious TCS journals and conferences.

- The area of scheduling was almost neglected up to 2015.

History

Parametrized Complexity and Scheduling

Papers I found (up to 2015):

- Bodlaender, HL., & Fellows, MR., 1995, W[2]-hardness of precedence constrained *k*-processor scheduling. *Operations Research Letters*, **18**(2),93–97.

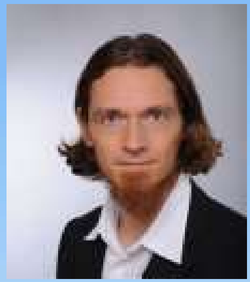
- Fellows MR, & McCartin C., 2003, On the parametric complexity of schedules to minimize tardy tasks. Theoretical Computer Science, **298**(2), 317–324 .

History
Parametrized Complexity and Scheduling

Since 2015 many papers with 2 main groups:

Group 1: Matthias Mnich, René van Bevern, Rolf Niedermeier, Mathias Weller, Andreas Wiese and Ondra Suchý



History

Group 1 – Selected Papers

- Mnich, M., & Wiese, A., 2015, Scheduling meets fixed-parameter tractability. ***Mathematical Programming***, **154**(1), 533–562.
- van Bevern, R., Mnich, M., Niedermeier, R., & Weller, M., 2015, Interval scheduling and colorful independent sets. ***Journal of Scheduling***, **18**(5), 449–469.

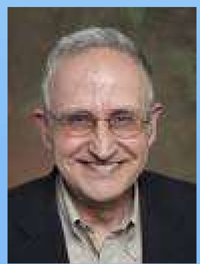
History

Group 1 – Selected Papers

- van Bevern, R., Niedermeier, R., & Suchý, O., 2017, A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: few machines, small looseness, and small slack. ***Journal of Scheduling***, **20**(3), 255–265.
- Mnich, M., & van Bevern, R., 2018, Parameterized complexity of machine scheduling: 15 open problems. ***Computers and Operations Research***, ***100***, 254-261.

History
Group 2

Group 2: Danny Hermilen, Dvir Shabtay, Mike Pinedo, Gerhard J. Woeginger, Nimrod Talmon, Liron Yedidsion, Shlomo Karhi, George Manoussakis.


History

Group 2 – Selected Papers

- Hermelin, D., Kubitza, J., Shabtay, D., Talmon, N., & Woeginger, G., 2019, Scheduling two agents on a single machine: A parameterized analysis of NP-hard problems, ***Omega***, **83**, 275-286.
- Hermelin, D., Pinedo, M., Shabtay, D., Talmon, N., & Woeginger, G., 2019, On the parameterized tractability of single machine scheduling with rejection, ***European Journal of Operational Research***, **273**(1), 67-73.
- Hermelin, D., Karhi, S., Pinedo, M., & Shabtay, D., 2021, New algorithms for minimizing the total weighted number of tardy jobs on a single machine, ***Annals of Operations Research***, **298** (1), 271-287.

History
Group 2 – Selected Papers

- Hermelin, D., Shabtay, D., and Talmon, N., 2019, On the parameterized tractability of the just-in-time scheduling problem, ***Journal of Scheduling***, **22**(6), 663-676.

- Hermelin, D., George Manoussakis, Pinedo, M., Shabtay, D., & Yedidsion, L., 2020, Parameterized multi-scenario single-machine scheduling problems, ***Algoritmica, 82*** (9), 2644-2667.

History
Among the Other Papers

- Knop, D., & Koutecký, M., 2018, Scheduling meets n-fold integer programming, **Journal of Scheduling**, **21**(5), 493-503.

- Bessy , S., & Giroudeau, R., 2019, Parameterized complexity of a coupled-task scheduling problem, **Journal of Scheduling**, **22**, 305–313.

- Bodlaender, H.L., and van der Wegen, M., 2020, Parameterized complexity of scheduling chains of jobs with delays, arXiv preprint arXiv:2007.09023.

Problem 1*

- Consider the classical $1|\ |\sum w_j U_j$ problem.
  - Instance:
    - n - # of jobs to be scheduled;
    - p_j – the processing time of job J_j (j=1,…,n);
    - d_j– the due date of job J_j (j=1,…,n).
    - w_j– the weight of job J_j (j=1,…,n) - (a penalty for the job being tardy).

* ***Annals of Operations Research***, **298** (1), 271-287.

Problem Definition

- A solution (schedule) is simply a job processing permutation, π, on the single machine.

- The objective is to determine a solution that minimizes the weighted number of tardy jobs, $\sum w_j U_j$, where U_j=1 if job J_j is completed after its due date, and U_j=0, otherwise.

An importance problem?

- The $1|\ |\sum w_j U_j$ problem is a fundamental problem in the field of combinatorial optimization in general, and particularly in scheduling theory.
- It is one out of the problems that appears in the seminal work by Karp [1972] about reducibility between combinatorial problems.

An importance problem?

- It is one out of a set of three problems in which the concept of FPTAS has been originally presented (Sahni [1976]).
- The problem is an extension of the well known 0-1 knapsack problem.

Known Results

- The $1|\quad|\sum w_j U_j$ problem is
  - NP-hard even if all due dates are equal (Karp (1972));
  - Solvable in pseudo-polynomial time (Lawler and Moore (1969) and Sahni (1976));
  - Solvable in O($n$log$n$) time when all weights are equal (Moore (1968));
  - Solvable in O($n$log$n$) time when all processing times are equal (Peha (1995)).

Research Goals

- We analyze the tractability of the $1|\ |\sum w_j U_j$ problem with respect to the following three parameters:
  - v_d - the number of different due dates.
  - v_p - the number of different processing times.
  - v_w - the number of different weights.

Are those "natural" parameters?

- In many practical instances at least one of those parameters is indeed of a limited size.
  - v_d when delivery costs are high and thus products are batched to only few shipments;
  - v_p when the number of job types that the manufacturer produces is limited; and
  - v_w when customers are batched into few subsets according to their importance.

Our Results for the the $1|\quad|\sum w_j U_j$ problem

Parameter	v_d	v_w	v_p	(v_d,v_p)	(v_d,v_w)	(v_p,v_w)
Result	Hard	XP	XP	FPT	FPT	FPT

- The hardness results is straightforward from Karp's NP-hardness proof for the common due date case.
- The XP algorithms are based on extensions of the well-known Moore's algorithm that solves the unit weight case.
- The FPT algorithms are based on MILP formulation with O(k) integer variables.

Remains Open:

- Is the problem FPT w.r.t v_w and v_p?

An FPT with respect to (υ_p, υ_w)

- Sketch of how we obtain the result:
  - We formulate the $1|\quad|\sum w_j U_j$ problem as an ILP with (too many…) $O(k+n)$ integer variables ($k = \upsilon_p \upsilon_w$). Let F be the corresponding formulation.
  - We relax F to a MILP formulation, F', that has only k integer variables; and then
  - Use Lenstra's algorithm from 1983 to solve F' in FPT time.

An FPT with respect to (υ_p, υ_w)

- Continue: Sketch of how we obtain the result:
  - If the optimal solution for *F*' (obtained by solving the MILP) is an integer solution, it is also optimal to *F* and we are done.
  - Otherwise, we provide a polynomial time rounding procedure to obtain an alternative **optimal integer solution** for *F*', which is also optimal for *F*.

An FPT with respect to (υ_p, υ_w)

- We begin by partitioning the set of jobs into k classes, $S_1,\dots,S_k$, such that all jobs in S_i have the same processing time p_i, and weight w_i (i=1,…,k).

- Let $n_i = |S_i|$ denote the number of jobs in each S_i.

An FPT with respect to $k = (\upsilon_p, \upsilon_w)$

- The following lemma is used to formulate F:

Lemma 1: There exists an optimal solution for the $1|\ |\sum w_j U_j$ problem, where the non-tardy jobs are scheduled first in an EDD order, followed by the tardy jobs in an arbitrary order.

An FPT with respect to (υ_p, υ_w)

- Let $d_1, \ldots, d_{\upsilon\_d}$ be the set of different due dates in our input job set J, and assume without loss of generality that $d_1 < d_2 < \ldots < d_{\upsilon\_d}$.

- Moreover, let δ_{ij} be the number of jobs in S_i having a due date of d_j, for $i = 1, \ldots, k$ and $j = 1, \ldots, \upsilon_d$.

The Formulation of F

- Decision variables:
  - y_i be an **integer variable** representing the number of tardy jobs in job set S_i, for each i=1,…,k.
  - x_{ij} be an **integer variable** representing the number of early jobs in S_i that have a due date of d_j.

The Formulation of F

Min $Z=\sum_{i=1}^{k} w_i y_i$

$s.t$

$$n_i - \sum_{j=1}^{v_d} x_{i,j} = y_i \quad \text{for all} \quad i \in \{1, \dots, k\}.$$

$$\sum_{i=1}^{k} \sum_{j=1}^{l} p_i x_{ij} \leq d_l \quad \text{for all} \quad l \in \{1, \dots, v_d\}$$

$$x_{ij} \leq \delta_{ij} \quad \text{for all} \quad i \in \{1, \dots, k\} \text{ and } j \in \{1, \dots, v_d\}.$$

$$x_{ij},\ y_i = int$$

- F has O(n+k) integer variables (too many).

The construction of *F'*

MILP relaxation:

- We construct formulation *F'* out of *F* by relaxing the x_{ij} variables, such that we only require that they have to be non-negative.

- *F'* is an MILP formulation with only k integer variables. Therefore, according to Lenstra [1983], it is solvable in FPT time.

Using the solution of *F*' to solve *F*

- Let S^*=(x^*, y^*), where $x^* = (x^*_{ij} | i$=1,…,k and j=1,…,v_d) and $y^* = (y^*_i | i$=1,…,k) be the solution obtained by solving for *F*' and let $x^*_i = \sum_{j=1}^{v_d} x^*_{ij}$.

- Note that x^*_i is an integer value for i =1,…,k due to the constraint that $n_i - \sum_{j=1}^{v_d} x_{i,j} = y_i \quad \text{for all} \quad i \in \{1, \ldots, k\}$ and the fact that both n_i and y_i are integer values.

Using the solution of *F'* to solve *F*

- If S^* is a feasible solution for F (i.e., all x^*_{ij} values are integer), then S^* is feasible (and therefore also optimal) solution for F.

- Otherwise, we use a simple *rounding procedure* to obtain an alternative optimal solution for F'.

Rounding Procedure

- The rounding procedure is based on exploiting the following lemma:
- **Lemma 2:** If x_i is the optimal number of early jobs in S_i then there exists an optimal solution in which the x_i jobs with the latest due date in S_i are early.

Rounding Procedure

For each i=1,…,k, let r_i be the integer satisfying

$$\sum_{j=r_i+1}^{v_d} \delta_{ij} \le x_i^* \le \sum_{j=r_i}^{v_d} \delta_{ij}$$

and define

$$\tilde{x}_{ij} = \left\{ \begin{array}{ccc} 0 & \text{for} & j = 1, \dots, r_i - 1 \\ x_i^* - \sum_{j=r_i+1}^{v_d} \delta_{ij} & \text{for} & j = r_i \\ \delta_{ij} & \text{for} & j = r_i + 1, \dots, v_d \end{array} \right\}$$

Result

<u>Theorem:</u> $(\tilde{x}, y^*)$ is an **optimal integer solution** for *F*'. Therefore, it is an optimal solution for *F*.

An XP algorithm with respect v_w

- Following Lemma 1, we renumber the jobs according to the EDD rule, such that $d_1 \leq d_2 \leq \ldots \leq d_n$.
- We say that a job is of type i if its weight is w_i ($i=1,..,v_w$).
- Let S_1 and S_2 be two partial schedules on job set $\{J_1,\ldots,J_j\}$, both with e_i early jobs of type i ($i=1,..,v_w$).

An XP algorithm with respect υ_w

- Moreover, let $P(S_i)$ be the total processing time of the $\sum_{i=1}^{\upsilon_w} e_i$ early jobs in partial schedule S_i for i=1,2.
- **Lemma 3**: If $P(S_1) \leq P(S_2)$ then S_2 is dominated by S_1.

An XP algorithm with respect v_w

- Based on Lemma 3, we developed a DP algorithm that construct the set of non dominated partial schedules. To do so, define:
  - $P_j(e_1, \ldots, e_{v_w})$ as the minimum total processing time of the early jobs among all partial schedules on job set $\{J_1,\ldots,J_j\}$ with e_i early jobs of type i $(i=1,..,v_w)$.
  - $E_{ij}(e_1, \ldots, e_{v_w})$ be the corresponding early sets.

An XP algorithm with respect v_w

- Each of the early sets, $E_{ij}(e_1, \dots, e_{v_w})$, is maintained during the DP as a list ordered according to the LPT rule.
- Note that the job in $E_{ij}(e_1, \dots, e_{v_w})$ with the largest processing time in the set is at the head of the list.

An XP algorithm with respect v_w

- Consider now the case where job J_j is of type i. We can reach state $(e_1, \ldots, e_{v_w})$ at stage j from either one of the following states in stage j-1:
  - State $(e_1, \ldots, e_{v_w})$ by setting $E_{ij}(e_1, \ldots, e_{v_w})$=$E_{i,j-1}(e_1, \ldots, e_{w_\#}) \cup \{J_j\}$ and then excluding the job at the head of $E_{ij}(e_1, \ldots, e_{v_w})$ from the list.
  - State $(e_1, \ldots, e_{i-1}, e_i - 1, e_{i+1}, \ldots, e_{v_w})$ by setting $E_{ij}(e_1, \ldots, e_{v_w})$=$E_{i,j-1}(e_1, \ldots, e_i - 1, e_{i+1}, \ldots, e_{v_w}) \cup \{J_j\}$. This is feasible only if $P_j(e_1, \ldots, e_i - 1, e_{i+1}, \ldots, e_{v_w}) + p_j \leq d_j$.

An XP algorithm with respect v_w

- Accordingly, the following recursive relation holds:
  - If $P_j(e_1, \dots, e_i - 1, e_{i+1}, \dots, e_{v_w}) + p_j > d_j$

    $$P_j(e_1, \dots, e_{v_w}) = P_{j-1}(e_1, \dots, e_{v_w}) + \min\{0, p_j - p^h_{i,j-1}(e_1, \dots, e_{v_w})\}$$

  - If $P_j(e_1, \dots, e_i - 1, e_{i+1}, \dots, e_{v_w}) + p_j \leq d_j$

    $$P_j(e_1, \dots, e_{v_w}) = \min\begin{cases} P_{j-1}(e_1, \dots, e_i - 1, e_{i+1}, \dots, e_{v_w}) + p_j \\ P_{j-1}(e_1, \dots, e_{v_w}) + \min\{0, p_j - p^h_{i,j-1}(e_1, \dots, e_{v_w})\} \end{cases}$$

An XP algorithm with respect v_w

- Initial Condition:

$$P_0\left(e_1, \dots, e_{v_w}\right) = \begin{cases} 0 & \text{if} \quad e_1 = e_2 = \cdots = e_{v_w} = 0 \\ \infty & \text{otherwise} \end{cases}$$

The optimal solution is given by

$$F^* = min\left\{\sum_{i=1}^{w_\#} w_i(n_i - e_i) \,\middle|\, P_n\left(e_1, \dots, e_{v_w}\right) < \infty\right\}$$

<u>**Theorem**</u>: The $1| \quad |\sum w_j U_j$ problem is solvable in $O(n^{v_w+1} logn)$ time. Thus, it belongs to the XP set w.r.t. parameter v_w.

Problem 2**

- We study the $Fm| \quad |\sum w_j R_j$ problem.
  - In a flow shop systems, all jobs follow the same route trough the machines.
  - **Instance**: the number of machines (m); the number of jobs (n); and for each job J_j, we are also given:
    - Its processing time on each one of the machines, p_{ij};
    - Its due date, d_j;
    - Its weight, w_j (a gain for being completed in a JIT mode).
  - **Problem**: Find a schedule that **maximizes** $\sum w_j R_j$, where R_j =1 if job J_j is completed ***exactly*** at its due date, and R_j=0, otherwise.

** ***Journal of Scheduling***, **22** **(6)**, 663-676.

Known Results

- **Known Results**: The $Fm| \quad |\sum w_j R_j$ problem is
  - Strongly NP-hard when m=3 (Choi and Yoon [2007].
  - Ordinary NP-hard when m=2 (Choi and Yoon [2007] and Shabtay and Bensoussan [2012]);
  - Solvable in $O(n^3)$ time when m=2 and all weights are equal (Shabtay [2012]).

- **Objective:** To analyze the parameterized tractability of the $Fm| \quad |\sum w_j R_j$ problem with respect to ν_d, which is the number of different due dates.

Problem 2 - Table of Results

	$F_2\|\quad\|\sum w_j R_j$	$F_3\|\quad\|\sum w_j R_j$
ν_d	W[1]-hard*, XP	W[1]-hard*, XP
(ν_d, ν_w)	FPT	W[1]-hard*
$(\nu_d, p^1_\#)$	FPT	W[1]-hard*

* even if all processing time on the second machine are of unit length.

Methods:

- The W[1]-hardness results have been obtained by a *parametrized reduction* from the *k*-sum problem.
- The XP and FPT algorithms are specially designed algorithms.

Problem 3***

- There are two agents each of which has its own set of jobs.
- All jobs are available at time zero and are to be processed on a single machine.
- Let $J^{(1)} = \{J_1^{(1)}, J_2^{(1)}, \ldots, J_n^{(1)}\}$ and $J^{(2)} = \{J_1^{(2)}, J_2^{(2)}, \ldots, J_k^{(2)}\}$ be the two set of jobs.

*** ***Omega***, **83**, 275-286.

Problem Definition

- Input:
  - $p_j^{(i)}$ – the processing time of job $J_j^{(i)}$.
  - A_i – a given bound on the objective value of agent i.

  When relevant also:
  - $d_j^{(i)}$– the due date of job $J_j^{(i)}$.
  - $w_j^{(i)}$ – the weight of job $J_j^{(i)}$.

Problem Definition

- Given a schedule of the $n+k$ jobs on the single machine, let $C_j^{(i)}$ be the completion time of job $J_j^{(i)}$.
- We measure the quality of a solution by two criteria, one for each agent.
- We focus on the following criteria:

Problem Definition

- *The weighted sum of completion times*, denoted by $\sum w_j^{(i)} C_j^{(i)}$.
- *The weighted number of tardy jobs*, denoted by $\sum w_j^{(i)} U_j^{(i)}$,

  where $U_j^{(i)} = 1$ if $C_j^{(i)} > d_j^{(i)}$ and $U_j^{(i)} = 0$, otherwise.
- The weighted number of JIT jobs, denoted by $\sum w_j^{(i)} R_j^{(i)}$, where

  $R_j^{(i)} = 1$ if $C_j^{(i)} = d_j^{(i)}$ and $R_j^{(i)} = 0$, otherwise.

Problem Definition

- For each possible combination of the three criteria to the two agents, we consider the following problem:
- Given two bounds A_1 and A_2, one for each agent, find if there exists a job schedule that meets both bounds.
- We refer to the problem by $1|\mathcal{C}_1, \mathcal{C}_2| -$, where

$$\mathcal{C}_i \in \left\{\sum w_j^{(i)} C_j^{(i)} \leq A_i, \sum w_j^{(i)} U_j^{(i)} \leq A_i, \sum w_j^{(i)} R_j^{(i)} \geq A_i\right\}$$

for *i*=1,2.

Problem Definition

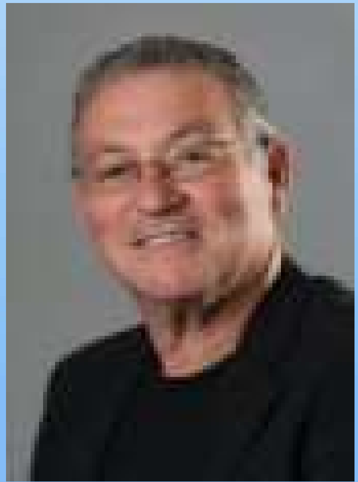
- The set of problems we define is well-studied in the literature and all relevant problems are NP-hard.

- We study the parametrized tractability of these set of problems w.r.t k (the number of jobs belong to the second agent).

Summary of Results – Problem 3

	$\sum \boldsymbol{w_j^{(2)} C_j^{(2)}} \le \boldsymbol{A_2}$	$\sum \boldsymbol{w_j^{(2)} U_j^{(2)}} \le \boldsymbol{A_2}$	$\sum \boldsymbol{w_j^{(2)} E_j^{(2)}} \ge \boldsymbol{A_2}$
$\sum \boldsymbol{w_j^{(1)} C_j^{(1)}} \le \boldsymbol{A_1}$	Hard for $w_j^{(2)} = 1$ (Th. 1), FPT for $w_j^{(1)} = 1$ (Th. 2), FPT for $p_j^{(1)} = 1$ (Th. 3).	Hard for $w_j^{(2)} = 1$ (Th. 1), FPT for $w_j^{(1)} = 1$ (Th. 3).	Hard even when $w_j^{(i)} = 1$ (Cor. 1).
$\sum \boldsymbol{w_j^{(1)} U_j^{(1)}} \le \boldsymbol{A_1}$	Hard in general (Cor. 2), Open for $w_j^{(1)} = 1$.	Hard in general (Cor. 2), FPT for $w_j^{(1)} = 1$ (Th. 6), FPT for $p_j^{(i)} = 1$ (Th. 7).	Hard even when $w_j^{(i)} = 1$ and $d_j^{(1)} = d$ (Cor. 4).
$\sum \boldsymbol{w_j^{(1)} E_j^{(1)}} \ge \boldsymbol{A_1}$	Open in general, FPT when $w_j^{(1)} = 1$ (Th. 8).	FPT (Th. 9)	FPT (Th. 11)

Special Thanks to My Academic Mentors/Advisors



Any questions?