A Framework for Automated Competitive Analysis of On-line Scheduling of Firm-Deadline Tasks

Krishnendu Chatterjee [1], **Andreas Pavlogiannis** [1],
Alexander Kößler [2], Ulrich Schmid [2]

[1]IST Austria, [2]TU Wien

July 22, 2015

IST AUSTRIA
Institute of Science and Technology

TU WIEN

Motivation Overview

Setting

- Scheduling firm deadline tasks on a single processor
- Jobs arrive in an online fashion and ask for the processor for some time
- Jobs have relative deadlines, and contribute some utility upon completion

Design task: Implement a scheduling policy to maximize utility

- Various online algorithms: FIFO, EDF, DSTAR ...
- Performance assessment of algorithm A through competitive factor
  - "In the worst case, how much less is the utility of $\mathcal{A}$ than the utility of a clairvoyant"
  - Algorithms $\mathcal{A}$ and $\mathcal{B}$ compared by comparing their competitive factors

Motivation Overview

Setting

- Scheduling firm deadline tasks on a single processor
- Jobs arrive in an online fashion and ask for the processor for some time
- Jobs have relative deadlines, and contribute some utility upon completion

Design task: Implement a scheduling policy to maximize utility

- Various online algorithms: FIFO, EDF, DSTAR ...
- Performance assessment of algorithm A through competitive factor
  - "In the worst case, how much less is the utility of $\mathcal{A}$ than the utility of a clairvoyant"
  - Algorithms $\mathcal{A}$ and $\mathcal{B}$ compared by comparing their competitive factors

Our Contribution

- Competitive factor might be too general ("worst case").
- This work: quantify competitiveness given some constraints on the environment that the algorithm operates

Given:

1. A fixed taskset from which jobs are spawned
2. A set of constraints on how jobs arrive

quantify the competitiveness of an online scheduling algorithm

Our Contribution

- Competitive factor might be too general ("worst case").
- This work: quantify competitiveness given some constraints on the environment that the algorithm operates

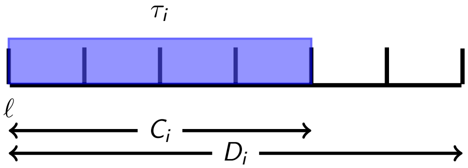
Given:

1. A fixed taskset from which jobs are spawned
2. A set of constraints on how jobs arrive

quantify the competitiveness of an online scheduling algorithm

Scheduling Setting

- A single processor
- Discrete notion of time in *slots*
- A set of tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$, each task is $\tau_i = (C_i, D_i, V_i)$
  - $\mathbf{C_i}$ is the execution time
  - $\mathbf{D_i}$ is the relative deadline
  - $\mathbf{V_i}$ is the utility value
- In every slot ℓ, a set Σ of task instances is released
- Each instance of task τ_i requires the processor for C_i slots in the interval $[\ell, \ell + D_i]$. On completion the system receives utility V_i
  - Preemption is allowed
  - Non-completed jobs contribute no utility



Labeled Transition Systems

Having fixed a taskset, we model scheduling algorithms as labeled transition systems
$L = (S, s_1, \Sigma, \Pi, \Delta)$ where

1. S is a finite set of states
2. $s_1 \in S$ is the initial state
3. Σ is a finite set of input actions
4. Π is a finite set of output actions
5. and $\Delta \subseteq S \times \Sigma \times S \times \Pi$ is the transition relation.

Σ is a set of each possible subset of jobs to be released at each slot

Π is a set of single-slot scheduling decisions

A job sequence $\sigma \in \Sigma^\infty$ generates a run ρ_L^σ and a schedule $\pi_L^\sigma \in \Pi^\infty$

Utility of π_L^σ in the first k slots $V(\pi_L^\sigma, k)$
Interested in $k \to \infty$

Labeled Transition Systems

Having fixed a taskset, we model scheduling algorithms as labeled transition systems

$L = (S, s_1, \Sigma, \Pi, \Delta)$ where

1. S is a finite set of states
2. $s_1 \in S$ is the initial state
3. Σ is a finite set of input actions
4. Π is a finite set of output actions
5. and $\Delta \subseteq S \times \Sigma \times S \times \Pi$ is the transition relation.

Σ is a set of each possible subset of jobs to be released at each slot

Π is a set of single-slot scheduling decisions

A job sequence $\sigma \in \Sigma^\infty$ generates a run ρ_L^σ and a schedule $\pi_L^\sigma \in \Pi^\infty$

Utility of π_L^σ in the first k slots $V(\pi_L^\sigma, k)$

Interested in $k \to \infty$

Three Types of Constraints on the Environment

Job sequences $\sigma \in \Sigma^\infty$ subject to:

1. Safety constraints
2. Liveness constraints
3. Limit-average constraints

Environment Constraints: Safety

Safety automaton $L_{\mathcal{S}} = (S_{\mathcal{S}}, s_{\mathcal{S}}, \Sigma, \emptyset, \Delta_{\mathcal{S}})$ with a distinguished reject state $s_r \in S_{\mathcal{S}}$

Job sequence $\sigma \in \Sigma^{\infty}$ *admissible* to $L_{\mathcal{S}}$ if s_r is never visited in $\rho_{\mathcal{S}}^{\sigma}$

Models

- "Nothing bad ever happens"
- Absolute workload restrictions (i.e., the released workload does not exceed a threshold in any fixed interval)
- Sporadicity (i.e., certain tasks are not released too often)
- Periodicity (i.e., certain tasks are released periodically)

Environment Constraints: Safety

Safety automaton $L_{\mathcal{S}} = (S_{\mathcal{S}}, s_{\mathcal{S}}, \Sigma, \emptyset, \Delta_{\mathcal{S}})$ with a distinguished reject state $s_r \in S_{\mathcal{S}}$

Job sequence $\sigma \in \Sigma^\infty$ *admissible* to $L_{\mathcal{S}}$ if s_r is never visited in $\rho_{\mathcal{S}}^{\sigma}$

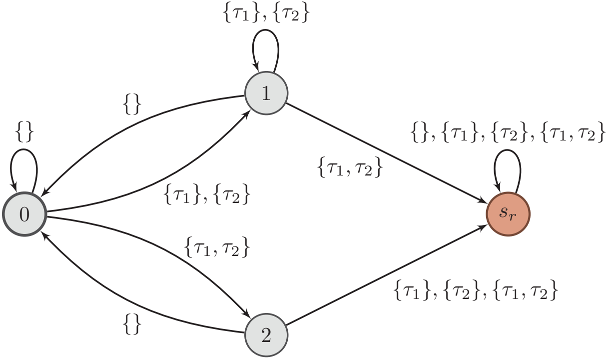
Models

- "Nothing bad ever happens"
- Absolute workload restrictions (i.e., the released workload does not exceed a threshold in any fixed interval)
- Sporadicity (i.e., certain tasks are not released too often)
- Periodicity (i.e., certain tasks are released periodically)

Environment Constraints: Safety

$\mathcal{T} = \{\tau_1, \tau_2\}$ with $C_1 = C_2 = 1$

"At most 2 units of workload in the last 2 rounds"



Environment Constraints: Liveness

Liveness automaton $L_\mathcal{L} = (S_\mathcal{L}, s_\mathcal{L}, \Sigma, \emptyset, \Delta_\mathcal{L})$, with a distinguished *accept* state $s_a \in S_\mathcal{L}$

Job sequence $\sigma \in \Sigma^\infty$ *admissible* to $L_\mathcal{L}$ if s_a is visited infinitely often in $\rho^\sigma_\mathcal{L}$

Models

- "Something good happens infinitely often"
- Finite intervals of (over)load (i.e., infinitely often there is no (over)load in the system)
- Some tasks are released infinitely often

Environment Constraints: Liveness

Liveness automaton $L_{\mathcal{L}} = (S_{\mathcal{L}}, s_{\mathcal{L}}, \Sigma, \emptyset, \Delta_{\mathcal{L}})$, with a distinguished *accept* state $s_a \in S_{\mathcal{L}}$

Job sequence $\sigma \in \Sigma^{\infty}$ *admissible* to $L_{\mathcal{L}}$ if s_a is visited infinitely often in $\rho_{\mathcal{L}}^{\sigma}$

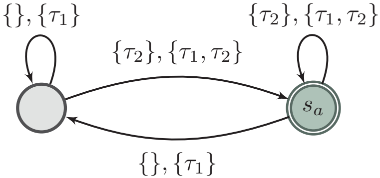
Models

- "Something good happens infinitely often"
- Finite intervals of (over)load (i.e., infinitely often there is no (over)load in the system)
- Some tasks are released infinitely often

Environment Constraints: Liveness

$\mathcal{T} = \{\tau_1, \tau_2\}$ with $C_1 = C_2 = 1$

"τ_2 released infinitely often"



Environment Constraints: Limit-average

Limit-average automaton $L_{\mathcal{W}} = (S_{\mathcal{W}}, s_{\mathcal{W}}, \Sigma, \emptyset, \Delta_{\mathcal{W}})$ with a weight function $w : \Delta_{\mathcal{W}} \to \mathbb{Z}^d$

Given some $\vec{\lambda} \in \mathbb{Q}^d$, job sequence $\sigma \in \Sigma^\infty$ *admissible* to $L_{\mathcal{W}}$ if $\liminf_{k\to\infty} \frac{1}{k} \cdot w(\rho^{\sigma}_{\mathcal{W}}, k) \leqslant \vec{\lambda}$

Models

- Something good happens on average
- Limit-average workload restrictions (i.e., the long run average released workload does not exceed a threshold)

Environment Constraints: Limit-average

Limit-average automaton $L_{\mathcal{W}} = (S_{\mathcal{W}}, s_{\mathcal{W}}, \Sigma, \emptyset, \Delta_{\mathcal{W}})$ with a weight function $w : \Delta_{\mathcal{W}} \to \mathbb{Z}^d$

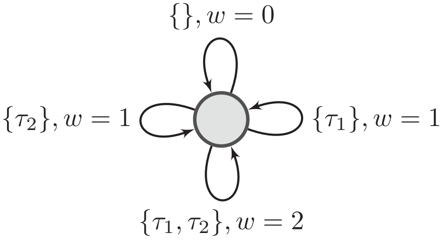
Given some $\vec{\lambda} \in \mathbb{Q}^d$, job sequence $\sigma \in \Sigma^\infty$ *admissible* to $L_{\mathcal{W}}$ if $\liminf_{k\to\infty} \frac{1}{k} \cdot w(\rho^\sigma_{\mathcal{W}}, k) \leqslant \vec{\lambda}$

Models

- Something good happens on average
- Limit-average workload restrictions (i.e., the long run average released workload does not exceed a threshold)

Environment Constraints: Limit-average

$\mathcal{T} = \{\tau_1, \tau_2\}$ with $C_1 = C_2 = 1$



Competitive Ratio

Given

1. A fixed taskset $\mathcal{T}$
2. Constraint automata $L_{\mathcal{S}}$, $L_{\mathcal{L}}$, $L_{\mathcal{W}}$ whose language intersection defines a set of admissible job sequences $\mathcal{J}$
3. Online algorithm as a *deterministic* LTS $L_{\mathcal{A}}$
4. Clairvoyant algorithm as a *non-deterministic* LTS $L_{\mathcal{C}}$

the competitive ratio of $L_{\mathcal{A}}$ w.r.t $\mathcal{J}$ is

$$\mathcal{CR}_{\mathcal{J}}(\mathcal{A}) = \inf_{\sigma \in \mathcal{J}} \liminf_{k \to \infty} \frac{1 + V(\pi^{\sigma}_{\mathcal{A}}, k)}{1 + V(\pi^{\sigma}_{\mathcal{C}}, k)}$$

Experiments

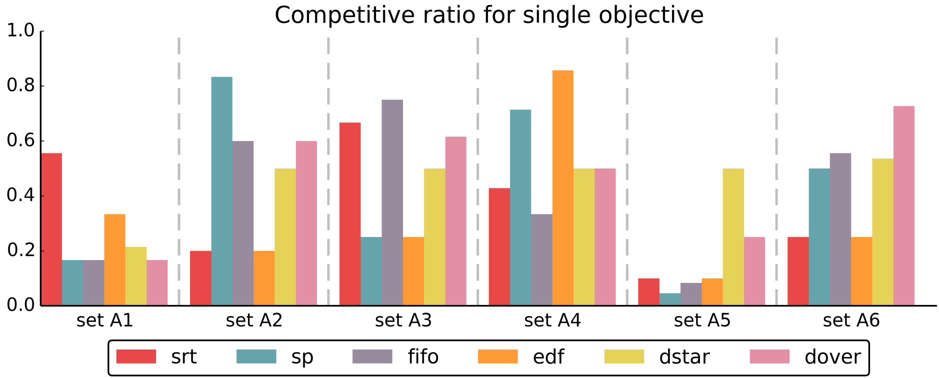
Implemented and analyzed 6 online scheduling algorithms in this framework:

1. SRT (Shortest Remaining Time)
2. SP (Static Priorities)
3. FIFO (First-in First-out)
4. EDF (Earliest Deadline First)
5. DSTAR
6. DOVER - proved to have optimal competitive factor

Prototype implementation in
`http://pub.ist.ac.at/~pavlogiannis/rtss14/`

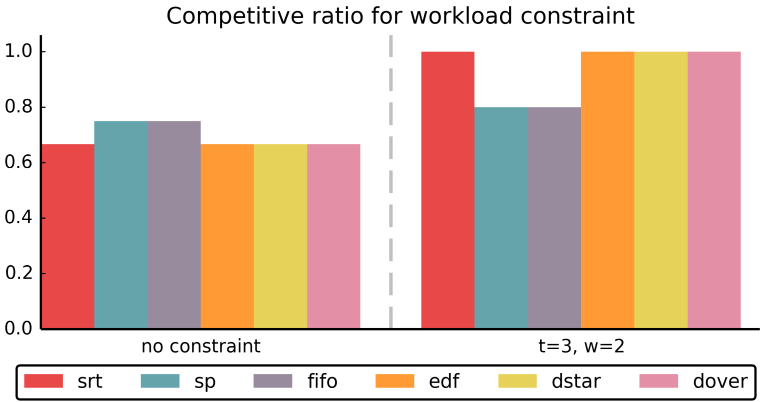
Results 1: No Constraints

For every examined scheduling algorithm, there is a taskset for which it is optimal among the others



Results 2: Safety constraints

Absolute workload constraints change the optimal scheduling algorithms in a fixed taskset



Results 3: Limit-average constraints

Average workload constraints change the optimal scheduling algorithms in a fixed taskset

✓indicates optimal for the given threshold

	1.5	1	0.8	0.6	0.4	0.3	0.1	0.078	0.05
fifo	✓	✓	✓	✓	✓				✓
sp	✓						✓		✓
srt	✓				✓	✓	✓	✓	✓

Message

1. Define competitiveness in constrained environments
   - Competitive ratio w.r.t. constraint automata
2. It makes sense to do so
   - Different constraints completely change the competitive algorithms
3. Automated way to determine the competitive ratio
   - Multi-graph objectives

Message

1. Define competitiveness in constrained environments
   - Competitive ratio w.r.t. constraint automata
2. It makes sense to do so
   - Different constraints completely change the competitive algorithms
3. Automated way to determine the competitive ratio
   - Multi-graph objectives

Message

1. Define competitiveness in constrained environments
   - Competitive ratio w.r.t. constraint automata
2. It makes sense to do so
   - Different constraints completely change the competitive algorithms
3. Automated way to determine the competitive ratio
   - Multi-graph objectives

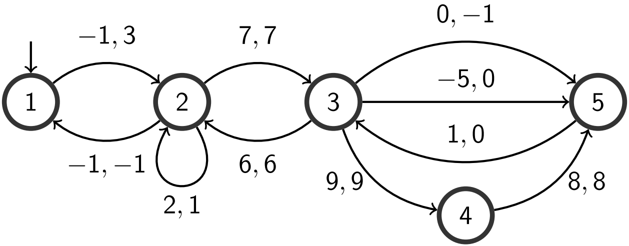
Message

1. Define competitiveness in constrained environments
   - Competitive ratio w.r.t. constraint automata
2. It makes sense to do so
   - Different constraints completely change the competitive algorithms
3. Automated way to determine the competitive ratio
   - Multi-graph objectives

Multi-Graphs

Consider multi-graph $G = (V, E)$

- Weight function $w : E \to \mathbb{Z}^d$ in d dimensions.
  - $d > 1$ in the presence of limit-average constraints
- An infinite *path* $\rho = (e^i)_{i \geqslant 1}$ is an infinite sequence of edges $e^i \in E$



Objectives

- An objective Φ is a set of paths
- G satisfies Φ if Φ is non-empty

$$\text{Competitive ratio} \quad \longrightarrow \quad \Phi = \mathsf{Safe}(X) \cap \mathsf{Live}(Y) \cap \mathsf{MP}(w, \vec{\nu})$$

Objectives - Complexity

Theorem

Let $\Phi = \mathrm{Safe}(X) \cap \mathrm{Live}(Y) \cap \mathrm{MP}(w, \vec{\nu})$. The decision problem of whether G satisfies the objective Φ requires

1. $O(|V| \cdot |E|)$ *time, if* $d = 1$.
2. *Polynomial time, if* $d > 1$.

$\mathbf{d = 1}$: Find the minimum-mean cycle of G

$\mathbf{d > 1}$: Solve a linear program in G

If the objective is satisfied, a witness path is reported

Objectives - Complexity

Theorem

Let $\Phi = \mathsf{Safe}(X) \cap \mathsf{Live}(Y) \cap \mathsf{MP}(w, \vec{\nu})$*. The decision problem of whether* G *satisfies the objective* Φ *requires*

1. $O(|V| \cdot |E|)$ *time, if* $d = 1$*.*
2. *Polynomial time, if* $d > 1$*.*

$\mathbf{d = 1}$: Find the minimum-mean cycle of G

$\mathbf{d > 1}$: Solve a linear program in G

If the objective is satisfied, a witness path is reported

Thank you!

Questions?

Qualitative objectives: Safety, Liveness

An *objective* Φ is a set of paths of G

Safety Given $X \subseteq V$, the objective $\mathrm{Safe}(X) = \{\rho \in \Omega : \ \forall i \geqslant 1, \rho^i \notin X\}$ is the set of all paths that never visit X.

Liveness Given $Y \subseteq V$, the objective $\mathrm{Live}(Y) = \{\rho \in \Omega : \ \forall j \exists i > j \text{ s.t. } \rho^i \in Y\}$ is the set of all paths that visit Y infinitely often.

Quantitative objectives: Mean-payoff

Mean-payoff Given a weight function $w : E \to \mathbb{Z}^d$ and threshold vector $\vec{\nu}$, the objective

$$\mathrm{MP}(w, \vec{\nu}) = \left\{ \rho \in \Omega : \ \liminf_{k \to \infty} \frac{1}{k} \cdot w(\rho, k) \leqslant \vec{\nu} \right\}$$

is the set of all paths such that the long-run average of their weights is at most $\vec{\nu}$

Quantitative objectives: Ratio

Ratio Given weight functions $w_1,\ w_2 : E \to \mathbb{N}^d$ and a threshold vector $\vec{\nu}$, the objective

$$\text{Ratio}(w_1, w_2, \vec{\nu}) = \left\{ \rho \in \Omega : \ \liminf_{k\to\infty} \frac{\vec{\mathbf{1}} + w_1(\rho, k)}{\vec{\mathbf{1}} + w_2(\rho, k)} \leqslant \vec{\nu} \right\}$$

is the set of all paths such that the ratio of cumulative rewards w.r.t w_1 and w_2 is at most $\vec{\nu}$

Multi-dimensional mean-payoff

For a strongly connected component $G_{\mathsf{SCC}} = (V_{\mathsf{SCC}}, E_{\mathsf{SCC}})$

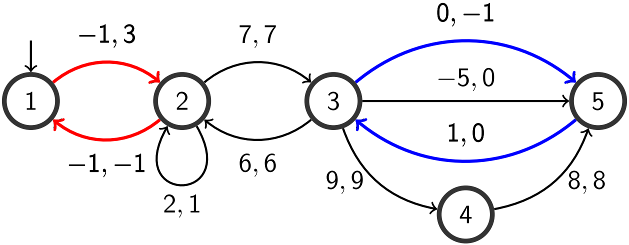
$$
\begin{aligned}
& x_e \geqslant 0 && e \in E_{\mathsf{SCC}} \\
& \sum_{e \in \mathrm{IN}(u)} x_e = \sum_{e \in \mathrm{OUT}(u)} x_e && u \in V_{\mathsf{SCC}} \\
& \sum_{e \in E_{\mathsf{SCC}}} x_e \cdot w(e) \leqslant \vec{\nu} && \\
& \sum_{e \in E_{\mathsf{SCC}}} x_e \geqslant 1 &&
\end{aligned}
$$

Objectives - witness

When $d > 1$, witness is a multi-cycle $\mathcal{MC} = \{(C_1, m_1), \ldots, (C_k, m_k)\}$

- C_i is a simple cycle
- m_i is its multiplicity

Out of the $\mathcal{MC}$ we construct a (generally) non-periodic path



Here, $\mathcal{MC} = \{(C_1, 1), (C_2, 2)\}$, with $C_1 = ((1, 2), (2, 1))$ and $C_2 = ((3, 5), (5, 3))$

Scalability

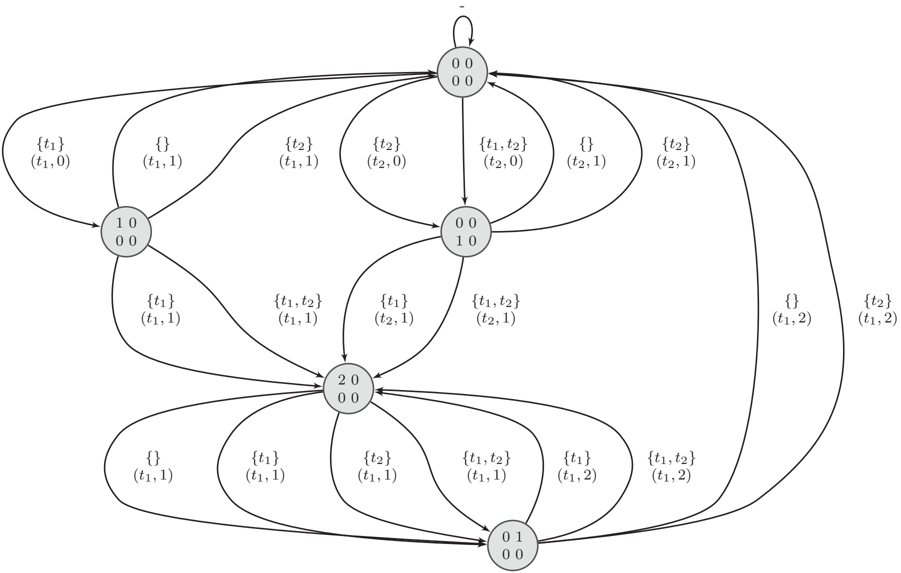
Name	N	$D_{\max}$	Size (nodes)		Time (s)	
			Clairv.	Product	Mean	Max
set B01	2	7	19	823	0.04	0.05
set B02	2	8	26	1997	0.4	0.6
set B03	2	9	34	4918	10	15
set B04	3	7	19	1064	0.2	0.4
set B05	3	8	26	1653	0.6	2
set B06	3	9	34	7705	51	130
set B07	4	7	19	1711	2.1	6.3
set B08	4	8	26	3707	14	34
set B09	4	9	44	10040	130	310
set B10	5	7	19	2195	5.7	16
set B11	5	8	32	9105	140	360
set B12	5	9	44	16817	550	1300

TD1

$$(i)\ C_0 = 1 \qquad (ii)\ C_{i+1} = \eta \cdot C_i - \sum_{j=0}^{i} C_j$$

Name	η	Taskset	Comp. Ratio
set C1	2	$\{1, 1\}$	**1**
set C2	3	$\{1, 2, 3\}$	**1/2**
set C3	3.1	$\{1, 3, 7, 13, 19\}$	**7/25**
set C4	3.2	$\{1, 3, 7, 13, 20, 23\}$	**1/4**
set C5	3.3	$\{1, 3, 7, 14, 24, 33\}$	**1/4**
set C6	3.4	$\{1, 3, 7, 14, 24, 34\}$	**1/4**

EDF LTS


-
0 0
0 0
1 0
0 0
0 0
1 0
2 0
0 0
0 1
0 0
$\{t_1\}$ $(t_1, 0)$
$\{\}$ $(t_1, 1)$
$\{t_2\}$ $(t_1, 1)$
$\{t_2\}$ $(t_2, 0)$
$\{t_1, t_2\}$ $(t_2, 0)$
$\{\}$ $(t_2, 1)$
$\{t_2\}$ $(t_2, 1)$
$\{t_1\}$ $(t_1, 1)$
$\{t_1, t_2\}$ $(t_1, 1)$
$\{t_1\}$ $(t_2, 1)$
$\{t_1, t_2\}$ $(t_2, 1)$
$\{\}$ $(t_1, 2)$
$\{t_2\}$ $(t_1, 2)$
$\{\}$ $(t_1, 1)$
$\{t_1\}$ $(t_1, 1)$
$\{t_2\}$ $(t_1, 1)$
$\{t_1, t_2\}$ $(t_1, 1)$
$\{t_1\}$ $(t_1, 2)$
$\{t_1, t_2\}$ $(t_1, 2)$