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
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 $A$ than the utility of a clairvoyant”
- Algorithms $A$ and $B$ 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 $A$ than the utility of a clairvoyant”
  - Algorithms $A$ and $B$ 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 $\tau_i = (C_i, D_i, V_i)$
  - $C_i$ is the execution time
  - $D_i$ is the relative deadline
  - $V_i$ is the utility value
- In every slot $\ell$, a set $\Sigma$ of task instances is released
- Each instance of task $\tau_i$ 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

![Diagram of task $\tau_i$ with execution time $C_i$ and relative deadline $D_i$ in slot $\ell$]
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. \( \Sigma \) is a finite set of input actions
4. \( \Pi \) is a finite set of output actions
5. and \( \Delta \subseteq S \times \Sigma \times S \times \Pi \) is the transition relation.

\( \Sigma \) is a set of each possible subset of jobs to be released at each slot
\( \Pi \) is a set of single-slot scheduling decisions

A job sequence \( \sigma \in \Sigma^\infty \) generates a run \( \rho^\sigma_L \) and a schedule \( \pi^\sigma_L \in \Pi^\infty \)

Utility of \( \pi^\sigma_L \) in the first \( k \) slots \( V(\pi^\sigma_L, k) \)
Interested in \( k \to \infty \)
Labeled Transition Systems

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

\[ L = (S, s_1, \Sigma, \Pi, \Delta) \]

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

\( \Sigma \) is a set of each possible subset of jobs to be released at each slot
\( \Pi \) is a set of single-slot scheduling decisions

A job sequence \( \sigma \in \Sigma^\infty \) generates a run \( \rho^\sigma_L \) and a schedule \( \pi^\sigma_L \in \Pi^\infty \)
Utility of \( \pi^\sigma_L \) in the first \( k \) slots \( V(\pi^\sigma_L, k) \)
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
Safety automaton $L_S = (S_S, s_S, \Sigma, \emptyset, \Delta_S)$ with a distinguished reject state $s_r \in S_S$

Job sequence $\sigma \in \Sigma^\infty$ admissible to $L_S$ if $s_r$ is never visited in $\rho^S$

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)
Safety automaton \( L_S = (S_S, s_S, \Sigma, \emptyset, \Delta_S) \) with a distinguished reject state \( s_r \in S_S \)

Job sequence \( \sigma \in \Sigma^\infty \) \textit{admissible} to \( L_S \) if \( s_r \) is never visited in \( \rho_S^\sigma \)

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** $L_L = (S_L, s_L, \Sigma, \emptyset, \Delta_L)$, with a distinguished accept state $s_a \in S_L$.

Job sequence $\sigma \in \Sigma^\infty$ admissible to $L_L$ if $s_a$ is visited infinitely often in $\rho^\sigma_L$.

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
Liveness automaton $L_L = (S_L, s_L, \Sigma, \emptyset, \Delta_L)$, with a distinguished accept state $s_a \in S_L$

Job sequence $\sigma \in \Sigma^\infty$ admissible to $L_L$ if $s_a$ is visited infinitely often in $\rho_\sigma^L$

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

\[ \mathcal{T} = \{\tau_1, \tau_2\} \] with \( C_1 = C_2 = 1 \)

“\( \tau_2 \) released infinitely often”

\[ \{\}, \{\tau_1\} \quad \{\tau_2\}, \{\tau_1, \tau_2\} \]

\[ \{\tau_2\}, \{\tau_1, \tau_2\} \quad \{\}, \{\tau_1\} \]

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

Given some $\vec{\lambda} \in \mathbb{Q}^d$, job sequence $\sigma \in \Sigma^\infty$ admissible to $L_{\mathcal{W}}$ if

$$\liminf_{k \to \infty} \frac{1}{k} \cdot w(\rho_{\mathcal{W}}^\sigma, k) \leq \vec{\lambda}$$

Models

- Something good happens on average
- Limit-average workload restrictions (i.e., the long run average released workload does not exceed a threshold)
Limit-average automaton \( L_W = (S_W, s_W, \Sigma, \emptyset, \Delta_W) \) with a weight function \( w : \Delta_W \to \mathbb{Z}^d \)

Given some \( \vec{\lambda} \in \mathbb{Q}^d \), job sequence \( \sigma \in \Sigma^\infty \) admissible to \( L_W \) if
\[
\liminf_{k \to \infty} \frac{1}{k} \cdot w(\rho^\sigma_{W}, k) \leq \vec{\lambda}
\]

Models

- Something good happens on average
- Limit-average workload restrictions (i.e., the long run average released workload does not exceed a threshold)
\[ \mathcal{T} = \{\tau_1, \tau_2\} \text{ with } C_1 = C_2 = 1 \]
Given

1. A fixed taskset $\mathcal{T}$
2. Constraint automata $L_S, L_L, L_W$ whose language intersection defines a set of admissible job sequences $\mathcal{J}$
3. Online algorithm as a deterministic LTS $L_A$
4. Clairvoyant algorithm as a non-deterministic LTS $L_C$

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

$$CR_{\mathcal{J}}(A) = \inf_{\sigma \in \mathcal{J}} \liminf_{k \to \infty} \frac{1 + V(\pi^\sigma_A, k)}{1 + V(\pi^\sigma_C, k)}$$
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.
Average workload constraints change the optimal scheduling algorithms in a fixed taskset.

✓ indicates optimal for the given threshold

<table>
<thead>
<tr>
<th></th>
<th>1.5</th>
<th>1</th>
<th>0.8</th>
<th>0.6</th>
<th>0.4</th>
<th>0.3</th>
<th>0.1</th>
<th>0.078</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>fifo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>sp</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>srt</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
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
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
Define competitiveness in constrained environments
- Competitive ratio w.r.t. constraint automata

It makes sense to do so
- Different constraints completely change the competitive algorithms

Automated way to determine the competitive ratio
- Multi-graph objectives
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
Consider multi-graph $G = (V, E)$

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

- An objective $\Phi$ is a set of paths
- $G$ satisfies $\Phi$ if $\Phi$ is non-empty

$$\text{Competitive ratio} \quad \rightarrow \quad \Phi = \text{Safe}(X) \cap \text{Live}(Y) \cap \text{MP}(w, \vec{v})$$
Theorem

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

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

\( d = 1 \): Find the minimum-mean cycle of \( G \)

\( d > 1 \): Solve a linear program in \( G \)

If the objective is satisfied, a witness path is reported.
Theorem

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

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

\( d = 1 \): Find the minimum-mean cycle of \( G \)

\( d > 1 \): 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 $\Phi$ is a set of paths of $G$

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

**Liveness** Given $Y \subseteq V$, the objective
$\text{Live}(Y) = \{\rho \in \Omega : \forall j \exists i > j \text{ s.t. } \rho^i \in Y\}$ is the set of all paths that visit $Y$ infinitely often.
Mean-payoff: Given a weight function $w : E \to \mathbb{Z}^d$ and threshold vector $\vec{\nu}$, the objective

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

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 \( w_1, w_2 : E \rightarrow \mathbb{N}^d \) and a threshold vector \( \vec{\nu} \), the objective

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

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 $G_{SCC} = (V_{SCC}, E_{SCC})$

$$x_e \geq 0$$

$$\sum_{e \in \text{IN}(u)} x_e = \sum_{e \in \text{OUT}(u)} x_e$$

$$\sum_{e \in E_{SCC}} x_e \cdot w(e) \leq \bar{v}$$

$$\sum_{e \in E_{SCC}} x_e \geq 1$$
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 $C_2 = ((3, 5), (5, 3))$
<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>$D_{\text{max}}$</th>
<th>Size (nodes)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Clairv.</td>
<td>Product</td>
</tr>
<tr>
<td>set B01</td>
<td>2</td>
<td>7</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>set B02</td>
<td>2</td>
<td>8</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>set B03</td>
<td>2</td>
<td>9</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>set B04</td>
<td>3</td>
<td>7</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>set B05</td>
<td>3</td>
<td>8</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>set B06</td>
<td>3</td>
<td>9</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>set B07</td>
<td>4</td>
<td>7</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>set B08</td>
<td>4</td>
<td>8</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>set B09</td>
<td>4</td>
<td>9</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>set B10</td>
<td>5</td>
<td>7</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>set B11</td>
<td>5</td>
<td>8</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>set B12</td>
<td>5</td>
<td>9</td>
<td>44</td>
<td></td>
</tr>
</tbody>
</table>
(i) $C_0 = 1$  
(ii) $C_{i+1} = \eta \cdot C_i - \sum_{j=0}^{i} C_j$

<table>
<thead>
<tr>
<th>Name</th>
<th>$\eta$</th>
<th>Taskset</th>
<th>Comp. Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>set C1</td>
<td>2</td>
<td>{1, 1}</td>
<td>1</td>
</tr>
<tr>
<td>set C2</td>
<td>3</td>
<td>{1, 2, 3}</td>
<td>1/2</td>
</tr>
<tr>
<td>set C3</td>
<td>3.1</td>
<td>{1, 3, 7, 13, 19}</td>
<td>7/25</td>
</tr>
<tr>
<td>set C4</td>
<td>3.2</td>
<td>{1, 3, 7, 13, 20, 23}</td>
<td>1/4</td>
</tr>
<tr>
<td>set C5</td>
<td>3.3</td>
<td>{1, 3, 7, 14, 24, 33}</td>
<td>1/4</td>
</tr>
<tr>
<td>set C6</td>
<td>3.4</td>
<td>{1, 3, 7, 14, 24, 34}</td>
<td>1/4</td>
</tr>
</tbody>
</table>
K. Chatterjee, A. Pavlogiannis, A. Kößler, U. Schmid
Automated Competitive Analysis of On-line Scheduling of Firm-Deadline Tasks
8