Dynamic Power Management Algorithms in Maximizing Net Profit

Zhi Zhang    Fei Li

Department of Computer Science
George Mason University
{zzhang8, lifei}@cs.gmu.edu

March 18, 2010
Motivation

- Real-Time system: Jobs associated with time constraints (deadlines)
  - Air Traffic Control – hard real-time system
  - Networked Multimedia System – soft real-time system (in this paper)

**Throughput** matters: key metric in judging the system performance

- Energy issue arises in many applications:
  - Mobile Devices – limit power
  - Data Centers – large electric bill

**Energy** matters: a critical factor degrades user experience or system performance
Previous Work

- **Model 1:** Minimize total energy cost on completing all the jobs. [Irani et al. SIGACT 05] [Baptiste et al. ESA 07]

- **Model 2:** Given limit energy budget, find an optimal subset of jobs to maximize the total throughput. [Rusu et al. TECS 03] [Devadas et al. ECRTS 09]

**Consider:** energy $E$ for $n$ jobs vs. energy $E/4$ for $3n/4$ jobs

**To Be Answered Question**

How do we evaluate efficiency of expending energy?

- **Model 3:** Maximize net profit by finishing a subset of jobs.
**Business.** Orders arrived at a manufacturer yielding products

**Computer Science.** Jobs arrived at a machine outputting solutions

**Constraints.** Time (deadlines)

**Goal.** Maximize net profit – the total revenue gained from products/jobs finished by their deadlines less total resources/energy costs paid
1 Introduction
   - Problem Setting
   - Our Contributions

2 General Net Profit Model
   - Challenges
   - Hardness Analysis

3 Variants in Underloaded System
   - Jobs Sharing a Common Deadline
   - Jobs Sharing a Common Release Time
   - Jobs with Agreeable Deadlines

4 Conclusions
A machine

- process one job at a time
- two states:
  - active $\rightarrow$ processing jobs, $\mu > 0$
  - sleep $\rightarrow$ cannot run any job, $\lambda = 0$
- transition costs $C = C_1 + C_2$:
  - power on (sleep to active): $C_1$
  - power down (active to sleep): $C_2$
- no time delays in state transitions

Figure: machine states and transitions between states
Model

- **Set of jobs** $J$, for each job $j$
  - release time $r_j \in \mathbb{Z}^+$
  - processing time $p_j \in \mathbb{R}^+$
  - value (reward) $w_j > 0$
  - deadline $d_j \in \mathbb{Z}^+$

**Figure:** jobs’ properties

- Total avenue gained by completing $S \subseteq J$
  $$V = \sum_{j \in S} w_j$$
Model

- Energy Cost:
  - \( E \): total energy spent in finishing job set \( S \subseteq J \)
  - \( T(a) \): total time spent in active state
  - \( T(s) \): total time spent in sleep state
  - \( m \in \mathbb{Z}^+ \): number of times that the machine is powered on

\[
E = \mu \cdot T(a) + \lambda \cdot T(s) + (C_1 + C_2) \cdot m \\
= \mu \cdot T(a) + C \cdot m
\]

Figure: total energy cost
Objective: Net Profit ($r$: value paid per unit of energy)

$$P = V - r \cdot E$$

$$V = w_1 + w_2 + w_3 + w_4 + w_5 + w_6$$

$$E = \mu \cdot (T_1\alpha + T_2\alpha) + 2 \cdot (C_1 + C_2)$$

$$P = V - r \cdot E$$

Figure: net profit
Definitions

**Definition**

**Underloaded System.** There exist a schedule to complete all jobs in a given input.

**Definition**

**Overloaded System.** There does not exist a schedule to complete all jobs in a given input.

**Definition**

**Non-preemption Setting.** Any job that is being processed cannot be preempted.

**Definition**

**Preemption-resume Setting.** The algorithm is allowed to preempt a job during its execution, and the preempted job can be resumed later.
## Table: summary of hardness of the net profit model.

<table>
<thead>
<tr>
<th>Settings</th>
<th>Overloaded</th>
<th>Underloaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-preemption</td>
<td>Strongly NP-complete</td>
<td>Strongly NP-complete</td>
</tr>
<tr>
<td>Preemption-resume</td>
<td>NP-complete</td>
<td>Polynomial time algorithms for some variants</td>
</tr>
</tbody>
</table>
1 Introduction
- Problem Setting
- Our Contributions

2 General Net Profit Model
- Challenges
- Hardness Analysis

3 Variants in Underloaded System
- Jobs Sharing a Common Deadline
- Jobs Sharing a Common Release Time
- Jobs with Agreeable Deadlines

4 Conclusions
An Example

<table>
<thead>
<tr>
<th>jobs</th>
<th>release time</th>
<th>deadlines</th>
<th>reward/reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j_1$</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$j_2$</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$j_3$</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>$j_4$</td>
<td>7</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>$j_5$</td>
<td>8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>$j_6$</td>
<td>9</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure: schedules minimizing total energy cost versus maximizing net profit

Challenge

Minimizing Energy $\neq$ Maximizing Net Profit
Theorem

Consider a non-preemptive underloaded system, the general net profit problem is a strongly NP-complete problem.
Consider a non-preemptive underloaded system, the general net profit problem is a strongly NP-complete problem.

Proof.

We can reduce the strongly NP-complete 3-Partition problem to this variant: Given an instance that has a finite set \( A \) with \( 3m \) sizes \( s(1), \ldots, s(3m) \) and a target sum \( B \), where \( \sum_{a \in A} s(a) = m \cdot B \) the objective of 3-Partition problem is to find out if \( A \) could be partitioned into \( m \) disjoint sets \( S_1, \ldots, S_m \) such that

\[
\sum_{a \in S_i} s(a) = B, \quad 1 \leq i \leq m.
\]

Given any instance \( A \) of the 3-Partition problem, we

1. Create \( 3m \) jobs: each job \( i \) has processing time \( p_i = s(i) \). All these \( 3m \) jobs share the same release time \( R = 0 \), a uniform value 1 and a common deadline \( D = \sum_{i=1}^{3m} s(i) + \delta \cdot (m - 1) \);

2. Create another \( m \) jobs: each job \( j \) has a release time \( j \cdot B + (j - 1)\delta \), a processing time \( \delta \), a value 1 and a deadline \( j \cdot B + j \cdot \delta \);

3. Let all energy consumption be zero.
Theorem

Consider an overloaded system, even if all jobs share the same release time and same deadline, maximizing net profit is a \textit{NP-complete} problem.
NP-Completeness of Some Variants

Theorem

Consider an overloaded system, even if all jobs share the same release time and same deadline, maximizing net profit is a NP-complete problem.

Proof.

We can reduce the NP-complete \textit{SUBSET-SUM} problem to this variant: Given an instance of a set \( \mathcal{I} \) with \( n \) integers \( m_1, \ldots, m_n \) and a target integer sum \( M \), the objective of \textit{SUBSET-SUM problem} is to find out if there exists a subset \( \mathcal{I}' \subseteq \mathcal{I} \) such that

\[
\sum_{i \in \mathcal{I}'} s_i = M.
\]

Given any instance \( \mathcal{I} \) of the \textit{SUBSET-SUM} problem, we

1. Create \( n \) jobs, such that a job \( i \) has processing time \( p_i = m_i \) and value \( w_i = m_i \);
2. All jobs share the same release time \( R = 0 \) and a common deadline \( D = M \);
3. Let all energy consumption be zero.
1 Introduction
   - Problem Setting
   - Our Contributions

2 General Net Profit Model
   - Challenges
   - Hardness Analysis

3 Variants in Underloaded System
   - Jobs Sharing a Common Deadline
   - Jobs Sharing a Common Release Time
   - Jobs with Agreeable Deadlines

4 Conclusions
**Property:** Jobs sharing a same deadline could be finished in a back-to-back manner. Therefore, maximizing net profit is the same as maximizing the total reward.

**Figure:** jobs with common deadlines aligned in order of release time
**Greedy-Based Algorithm**

1. Remove the jobs with $w_j \leq r \cdot \mu \cdot p_j$, since they contribute non-positive reward to our objective.
2. The remaining jobs consist a set with maximal reward, and they can be aligned in order of release time.
3. If (the total reward is larger than the total energy cost)
   - run all jobs in this subset.
4. Else
5. run nothing.

**Theorem**

*Given a set of jobs sharing a common deadline, Greedy-Based algorithm has a running complexity of $O(n \log n)$ in maximizing net profit, where $n$ is the number of jobs.*
Jobs Sharing a Common Release Time

**Property:** Jobs released at the same time could be finished in a back-to-back manner. Therefore, maximizing net profit is the same as maximizing the total reward.

**Figure:** jobs with common release times aligned in order of deadline
EDF-Based Algorithm

1. Remove the jobs with \( w_j \leq r \cdot \mu \cdot p_j \), since they contribute non-positive reward to our objective.
2. The remaining jobs consist a set with maximal reward, and they can be aligned in order of deadlines.
3. If (the total reward is larger than the total energy cost)
   4. run all jobs in this subset.
4. Else
   5. run nothing.

Theorem

Given a set of jobs released at the same time, EDF-Based algorithm has a running complexity of \( O(n \log n) \) in maximizing net profit, where \( n \) is the number of jobs.
Jobs with Agreeable Deadlines

**Definition**

**Block.** A set of jobs consecutively executing in a back-to-back manner without any gap in-between.

**Agreeable deadline.** For any two jobs $i, j$, $r_i < r_j$ implies $d_i \leq d_j$. 

**Figure:** jobs organized as blocks
Jobs with Agreeable Deadlines

**Definition**

**Block with fixed position.** A block cannot move to the left or to the right further without generating a gap.

---

![Diagram of block with fixed position](image)
DP-Based Algorithm

**Figure:** main idea of DP-based algorithm
DP-Based Algorithm

1. Remove the jobs with $w_j \leq r \cdot \mu \cdot p_j$.
2. Sort all jobs in $J$ in a decreasing order of deadlines as $S$.
3. Let the last released job in $S$ be $l$.
4. Use $B(i)$ to denote the block containing a job $i$.
5. Construct the last block including $l$ with the earliest execution time for the whole block as $B(l)$.
6. The optimal net profit $P(S)$ is calculated recursively as

   $$P(S) = \max \left\{ P(S \setminus B(l)) + \sum_{i \in B(l)} w_i - r(C + \mu |B(l)|), \quad P(S \setminus \{l\}) \right\}$$

7. Apply above formula for each subset of jobs after we identify whether or not we keep the block of jobs $B(l)$ in each iteration.

Theorem

*Given a set of jobs with agreeable deadline and $C = \mu$, DP-Based algorithm has a running complexity of $O(n^2 \log n)$ in maximizing net profit, where $n$ is the number of jobs.*
DP-Based Algorithm

<table>
<thead>
<tr>
<th>jobs</th>
<th>release time</th>
<th>deadlines</th>
<th>reward/reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>( j_1 )</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( j_2 )</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>( j_3 )</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>( j_4 )</td>
<td>7</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>( j_5 )</td>
<td>8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>( j_6 )</td>
<td>9</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table:** \( C = \mu = 1, r = 1 \)

<table>
<thead>
<tr>
<th>( P(1) )</th>
<th>( P(2) )</th>
<th>( P(3) )</th>
<th>( P(4) )</th>
<th>( P(5) )</th>
<th>( P(6) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( B(1) )</th>
<th>( B(2) )</th>
<th>( B(3) )</th>
<th>( B(4) )</th>
<th>( B(5) )</th>
<th>( B(6) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( j_1 )</td>
<td>( j_1, j_2 )</td>
<td>( j_1, j_2, j_3 )</td>
<td>( j_2, j_3, j_4 )</td>
<td>( j_2, j_3, j_4, j_5 )</td>
<td>( j_2, j_3, j_4, j_5, j_6 )</td>
</tr>
</tbody>
</table>

**Table:** The table of \( P(i) \) and \( B(l) \)
1. **Introduction**
   - Problem Setting
   - Our Contributions

2. **General Net Profit Model**
   - Challenges
   - Hardness Analysis

3. **Variants in Underloaded System**
   - Jobs Sharing a Common Deadline
   - Jobs Sharing a Common Release Time
   - Jobs with Agreeable Deadlines

4. **Conclusions**
Open Problems

<table>
<thead>
<tr>
<th>Settings</th>
<th>Overloaded</th>
<th>Underloaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-preemption</td>
<td>Strongly NP-complete</td>
<td>Strongly NP-complete</td>
</tr>
<tr>
<td>Preemption-resume</td>
<td>NP-complete</td>
<td>Greedy-based</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EDF-based</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DP-based</td>
</tr>
</tbody>
</table>

1. Finding efficient algorithm for more general models of net profit problems
2. Online algorithms
3. Are there any other interesting and practical variants?
Questions?