Algorithms for Power Savings
S. Irani, S. Shukla, R. Gupta, TALG ’07

Giorgos Zois

Department of Informatics,
Athens University of Economics and Business
georzois@aueb.gr

November 2010
Two main mechanisms for minimizing power usage

- **Sleep State.** If a system is idle, it can be put into a low-power sleep state. A fixed amount of energy is required to transition the system back.

- **Dynamic Speed Scaling.** Tasks can be executed at different speeds. The power usage is a convex function $P(s)$ of the system’s speed $s$. The goal is to complete all jobs between their release time and their deadline in a way that minimizes the total energy consumption.
Two main mechanisms for minimizing power usage

- **Sleep State.** If a system is idle, it can be put into a low-power sleep state. A fixed amount of energy is required to transition the system back.

- **Dynamic Speed Scaling.** Tasks can be executed at different speeds. The power usage is a convex function $P(s)$ of the system’s speed $s$. The goal is to complete all jobs between their release time and their deadline in a way that minimizes the total energy consumption.

**Online and Offline** algorithms for both mechanisms:

- Ski Rental Problem [Irani, Karlin ’97], multiple sleep states [Augustine et al. 2008]

- YDS, AVR and Optimal available [Yao et. al ’95], BKP [Bensal et al. ’07]
Motivation

Battery-Operated Embedded Systems

Portable devices, such as mobile phones, personal digital assistants, communicators, palmtops etc.

1 Dynamic Power Management
2 Dynamic Voltage Scaling
Motivation

Battery-Operated Embedded Systems

Portable devices, such as mobile phones, personal digital assistants, communicators, palmtops etc.

1. Dynamic Power Management
2. Dynamic Voltage Scaling

Embedded Systems that **combine** 1 & 2: Rockwell WINS node, Smart Bridge
DSS-S: Dynamic Speed Scaling with a Sleep State

- Jobs can be executed at different speeds
- Preemption is allowed
- Each job \( j \) has a release date \( r_j \), deadline \( d_j \) and workload \( R_j \)
- A schedule is a triplet of functions \( S = (s, \text{job}, \phi) \) over \([t_0, t_1]\)
- \( \phi \) is the system’s state: on or sleep
- The cost to transition back from a sleep state is 1
- Power consumption rate is a convex function
  \[
  P(s, \phi) = \begin{cases} 
  P(s), & \phi = \text{on}, \\
  0, & \phi = \text{sleep}.
  \end{cases}
  \]
- When the system is idle and on we have \( s = 0 \) and \( P(0) > 0 \)
DSS-S: Dynamic Speed Scaling with a Sleep State

The total energy consumed by a schedule $S$ is

$$\text{cost}(S) = k + \int_{t_0}^{t_1} P(s(t), \phi(t)) \, dt,$$

$k$ is the number of transitions from sleep state to on state.

**Goal:** Find a feasible schedule $S$ with the minimum $\text{cost}(S)$. 
The Problem DSS-NS

What happens without Sleep States: DSS-NS problem

- \( P(s, \phi) = P(s) \), since \( \phi \) is always on (even when the system is idle)
- \( \text{cost}(S) = \int_{t_0}^{t_1} P(s(t)) \, dt \)

Lemma

Consider an instance \( J \) of DSS-NS (or DSS-S) with power function \( P \). There is an optimal schedule in which for every job \( j \) in \( J \), there is a constant speed \( s_j \) such that \( s(t) = s_j \), at every time \( t \) that the job is executed.
An optimal Algorithm for Problem DSS-NS

Intensity of an Interval \( I = [z, z'] \)

\[
g(I) = \frac{\sum_{j: [r_j, d_j] \subseteq I} R_j}{\text{length}(I)}
\]
An optimal Algorithm for Problem DSS-NS

Intensity of an Interval $I = [z, z']$

$$g(I) = \frac{\sum_{j: [r_j, d_j] \subseteq I} R_j}{\text{length}(I)}$$

The YDS algorithm [Yao et al. ’95]:

1. Iteratively select an interval $I$ of maximum intensity
2. Schedule all jobs contained in $I$ at a speed of $g(I)$ using Earliest Deadline First (EDF) rule and then black out the interval $I$
3. Remove the scheduled jobs from the set of remaining jobs
An optimal Algorithm for Problem DSS-NS

Intensity of an Interval $I = [z, z']$

$$g(I) = \frac{\sum_{j: [r_j, d_j] \subseteq I} R_j}{\text{length}(I)}$$

The YDS algorithm [Yao et al. '95]:

1. Iteratively select an interval $I$ of maximum intensity
2. Schedule all jobs contained in $I$ at a speed of $g(I)$ using Earliest Deadline First (EDF) rule and then black out the interval $I$
3. Remove the scheduled jobs from the set of remaining jobs

Note: Let $I_1, I_2, ..., I_r$ be the sequence of intervals after $r$ iterations. Then for any $k = 0, 1, ..., r$, $g(I_{k-1}) \geq g(I_k)$
An optimal Algorithm for Problem DSS-NS

Interval $I_x$

- $d_i$
- $r_l$
- $r_j$
- $d_k$

$t$

$r_i$

$r_k$

$d_l$

$d_j$
An optimal Algorithm for Problem DSS-NS

Interval $I_x$

... $d_i$ $r_l$ $r_j$ $d_k$ ...

$t$

... $r_i$ $r_k$ $d_l$ $d_j$ $t$

... $r_i$ $d_i$ $r_j$ $d_j$ $t$

Interval $I_x$
An Algorithm for DSS-S

A critical speed $s_{\text{crit}}$

Consider a job $j$ with workload $R_j$ that is executed at a constant speed $s$. It requires power expenditure of $P(s)$ for a period of time of $\frac{R_j}{s}$. The optimal speed for the job would be the $s$ that minimizes

$$\frac{R_j}{s} P(s)$$
An Algorithm for DSS-S

We can follow the optimal YDS algorithm for DSS-NS problem until a job is scheduled at a speed less than $s_{crit}$.

Lemma

Suppose that we have a partial schedule $P$ for the DSS-S problem. Let $I$ be the interval of maximum intensity. If the intensity of $I$ is at least $s_{crit}$, then there is an optimal extension of $P$ in which only jobs contained in $I$ are scheduled during interval $I$. Furthermore, the optimal schedule schedules all jobs contained in $I$ using earliest-deadline-first with no sleep periods.
An Algorithm for DSS-S

Algorithm 1 Algorithm for DSS-S

1. **Scheduling Fast Jobs**: Run YDS until you find an I having \( g(I) \leq s_{\text{crit}} \).

For the remaining jobs

2. **Scheduling Slow Jobs**:  
   - **While system is active** execute the jobs at speed \( s_{\text{crit}} \) until there are no more jobs to run
   - **When it became idle** stay idle as long as possible and wake up in order to complete all jobs by their deadline at speed \( s_{\text{crit}} \).

After all idle periods have been determined

3. Decide whether the system will transition into the sleep state during each such interval (check if interval’s length is at least \( 1/P(0) \))
An Algorithm for DSS-S

\[ g(I_x) \geq s_{crit} \quad g(I_y) \geq s_{crit} \quad g(I_k) \leq s_{crit} \]

\[ I_x \quad I_y \quad I_k \]

\[ t_0 \quad t \]

\[ t_0 \quad t \]
An Algorithm for DSS-S

execute every job at $s_{crit}$

idle interval

$t_w$

$t_0$
When the system is not currently running a job

- We order all jobs according to EDF: \( d_1 \leq \cdots \leq d_k \)
- \( \forall j \in \{1, \ldots, k\} \), we compute

\[
    t_j = d_j - \left( \frac{\sum_{j=1}^{k} R_j}{s_{crit}} \right)
\]

- We set wake up time to be \( t_w = \min_j t_j \)
Energy expenditure for a schedule $S$

- The energy expended while the system is active:
  \[
  \text{active}(S) = \int_{t_0}^{t_1} P(s(t))\delta_s(t) \, dt
  \]

- The cost to keep the system on or sleep and wake up during idle periods:
  \[
  \text{idle}(S) = \sum_{I \in ID} \min(P(0), |I|, 1)
  \]

- \(\text{cost}(S_{ALG}) = \text{active}(S_{ALG}) + \text{idle}(S_{ALG})\)
Energy expenditure for a schedule \( S \)

- The cost to keep the system in the on state while the system is on:

\[
on(S) = \int_{t_0}^{t_1} P(0) \delta(\phi(t), \text{on}) \, dt
\]

- The cost to wake up the system at the end of each sleep interval:

\[
sleep(S) = \text{number of intervals in } SL
\]
Algorithm’s Analysis

Lemma

At most two idle intervals from ALG can intersect a single sleep interval from OPT
Lemma

At most two idle intervals from ALG can intersect a single sleep interval from OPT
A 2-approximation

Fact 1. $\text{active}(S_{\text{ALG}}) \leq \text{active}(S_{\text{OPT}})$

Fact 2. $\text{idle}(S_{\text{ALG}}) \leq \text{on}(S_{\text{OPT}}) + 2\text{sleep}(S_{\text{OPT}})$
A 2-approximation

Fact 1. $\text{active}(S_{\text{ALG}}) \leq \text{active}(S_{\text{OPT}})$

Fact 2. $\text{idle}(S_{\text{ALG}}) \leq \text{on}(S_{\text{OPT}}) + 2\text{sleep}(S_{\text{OPT}})$

$$\text{cost}(S_{\text{ALG}}) = \text{active}(S_{\text{ALG}}) + \text{idle}(S_{\text{ALG}}) \leq \text{active}(S_{\text{OPT}}) + \text{on}(S_{\text{OPT}}) + 2\text{sleep}(S_{\text{OPT}}) \leq 2\text{cost}(S_{\text{OPT}})$$
A 2-approximation

Fact 1. $\text{active}(S_{ALG}) \leq \text{active}(S_{OPT})$

Fact 2. $\text{idle}(S_{ALG}) \leq \text{on}(S_{OPT}) + 2\text{sleep}(S_{OPT})$

\[
\text{cost}(S_{ALG}) = \text{active}(S_{ALG}) + \text{idle}(S_{ALG}) \\
\leq \text{active}(S_{OPT}) + \text{on}(S_{OPT}) + 2\text{sleep}(S_{OPT}) \\
\leq 2\text{cost}(S_{OPT})
\]

Theorem

If the power function $P(s)$ is convex, then the Algorithm 1 achieves an approximation ratio of 2.
The online case

An online algorithm for DSS-S

While the system is active it transitions between two modes:

- **Slow mode**: if it is feasible to complete all pending jobs by their deadline at a speed of $s_{\text{crit}}$
- **Fast mode**: if there is excess to the total workload

**Theorem**

Assume that $P(s)$ is a convex function. Let $c_1$ be the competitive ratio of an online algorithm for the DSS-NS problem. Let $f(x) = P(x) - P(0)$. Let $c_2$ be such that $\forall x, y > 0, f(x + y) \leq c_2(f(x) + f(y))$. The competitive ratio of the algorithm is at most $\max\{c_2c_1 + c_2 + 2, 4\}$. 
Some Open Questions

- Proof of NP-hardness or...
Some Open Questions

- Proof of NP-hardness or

- Polynomial time exact algorithm
Some Open Questions

- Proof of NP-hardness or
- Polynomial time exact algorithm
- Energy Reducing Steps: Rerunning YDS algorithm with all the sleep intervals blacked out
Some Open Questions

- Proof of NP-hardness or
- Polynomial time exact algorithm
- **Energy Reducing Steps:** Rerunning YDS algorithm with all the sleep intervals blacked out
- **Improve the competitive ratio**
Thanks for your attention!