

On Symbolic Scheduling Independent Tasks with Restricted Execution Times

Daniel Sawitzki

University of Dortmund
Germany

WEA 2005
May 10–13, 2005

Overview

- 1** Leung's Scheduling Algorithm
- 2** Ordered Binary Decision Diagrams
- 3** A Symbolic Scheduling Algorithm
- 4** Experimental Results
- 5** Conclusions

Problem Definition

General problem:

- Schedule N independent tasks onto m parallel identical machines
- Job i has execution time $t_i \in \mathbb{N}$
- Decision variant: Is there a schedule $\mathcal{S}: [N] \rightarrow [m]$ with makespan at most D ?
- NP-hard even for $m = 2$!

Restricted problem:

- Only k different execution times: $|\{t_1, \dots, t_N\}| = k$
- Dynamic Programming: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$ (Leung 1982)

Problem Definition

General problem:

- Schedule N independent tasks onto m parallel identical machines
- Job i has execution time $t_i \in \mathbb{N}$
- Decision variant: Is there a schedule $\mathcal{S}: [N] \rightarrow [m]$ with makespan at most D ?
- NP-hard even for $m = 2$!

Restricted problem:

- Only k different execution times: $|\{t_1, \dots, t_N\}| = k$
- Dynamic Programming: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$ (Leung 1982)

Problem Definition

General problem:

- Schedule N independent tasks onto m parallel identical machines
- Job i has execution time $t_i \in \mathbb{N}$
- Decision variant: Is there a schedule $\mathcal{S}: [N] \rightarrow [m]$ with makespan at most D ?
- NP-hard even for $m = 2$!

Restricted problem:

- Only k different execution times: $|\{t_1, \dots, t_N\}| = k$
- Dynamic Programming: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$ (Leung 1982)

Problem Definition

General problem:

- Schedule N independent tasks onto m parallel identical machines
- Job i has execution time $t_i \in \mathbb{N}$
- Decision variant: Is there a schedule $\mathcal{S}: [N] \rightarrow [m]$ with makespan at most D ?
- NP-hard even for $m = 2$!

Restricted problem:

- Only k different execution times: $|\{t_1, \dots, t_N\}| = k$
- Dynamic Programming: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$ (Leung 1982)

Problem Definition

General problem:

- Schedule N independent tasks onto m parallel identical machines
- Job i has execution time $t_i \in \mathbb{N}$
- Decision variant: Is there a schedule $\mathcal{S}: [N] \rightarrow [m]$ with makespan at most D ?
- NP-hard even for $m = 2$!

Restricted problem:

- Only k different execution times: $|\{t_1, \dots, t_N\}| = k$
- Dynamic Programming: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$ (Leung 1982)

Problem Definition

General problem:

- Schedule N independent tasks onto m parallel identical machines
- Job i has execution time $t_i \in \mathbb{N}$
- Decision variant: Is there a schedule $\mathcal{S}: [N] \rightarrow [m]$ with makespan at most D ?
- NP-hard even for $m = 2$!

Restricted problem:

- Only k different execution times: $|\{t_1, \dots, t_N\}| = k$
- Dynamic Programming: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$ (Leung 1982)

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t.:
- $0 \leq \ell \leq \log_2 m$
- $0 \leq p_i \leq N_i$
- E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
- E is maximum
- T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}\left(N^{k-1} \cdot \log m\right)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t. :
 - $0 \leq \ell \leq \log_2 m$
 - $0 \leq p_i \leq N_i$
 - E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
 - E is maximum
 - T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}\left(N^{k-1} \cdot \log m\right)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t.:
- $0 \leq \ell \leq \log_2 m$
- $0 \leq p_i \leq N_i$
- E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
- E is maximum
- T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}\left(N^{k-1} \cdot \log m\right)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t.:
- $0 \leq \ell \leq \log_2 m$
- $0 \leq p_i \leq N_i$
- E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
- E is maximum
- T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}\left(N^{k-1} \cdot \log m\right)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t.:
- $0 \leq \ell \leq \log_2 m$
- $0 \leq p_i \leq N_i$
- E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
- E is maximum
- T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}\left(N^{k-1} \cdot \log m\right)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t.:
- $0 \leq \ell \leq \log_2 m$
- $0 \leq p_i \leq N_i$
- E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
- E is maximum
- T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}(N^{k-1} \cdot \log m)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t.:
- $0 \leq \ell \leq \log_2 m$
- $0 \leq p_i \leq N_i$
- E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
- E is maximum
- T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}(N^{k-1} \cdot \log m)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (1/2)

- Let there be N_i jobs with time t_i , $i = 1, \dots, k$
- Compute table T with entries $T(\ell, p_1, \dots, p_{k-1}) = E$ s. t.:
- $0 \leq \ell \leq \log_2 m$
- $0 \leq p_i \leq N_i$
- E type- k -jobs can be scheduled onto 2^ℓ machines together with p_i type- i -jobs ($i = 1, \dots, k - 1$),
- E is maximum
- T has size

$$(\log_2 m + 1) \cdot (N_1 + 1) \cdots (N_{k-1} + 1) = \mathcal{O}\left(N^{k-1} \cdot \log m\right)$$

- Valid schedule exists $\Leftrightarrow T(\log_2 m, N_1, \dots, N_{k-1}) \geq N_k$

Leung's Scheduling Algorithm (2/2)

- Initial values for $1 = 2^0$ machines:

$$T(0, p_1, \dots, p_{k-1}) := \left\lfloor \left(D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) / t_k \right\rfloor$$

- Recursion:

$$\begin{aligned} T(\ell + 1, p_1, \dots, p_{k-1}) \\ := \max \{ & T(\ell, a_1, \dots, a_{k-1}) + T(\ell, b_1, \dots, b_{k-1}) \\ & \mid \forall i: p_i = a_i + b_i \} \end{aligned}$$

- Time per entry: $\Theta((N_1 + 1) \cdots (N_{k-1} + 1)) = \mathcal{O}(N^{k-1})$
- Altogether: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$

Leung's Scheduling Algorithm (2/2)

- Initial values for $1 = 2^0$ machines:

$$T(0, p_1, \dots, p_{k-1}) := \left\lfloor \left(D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) / t_k \right\rfloor$$

- Recursion:

$$\begin{aligned} T(\ell + 1, p_1, \dots, p_{k-1}) \\ := \max \{ & T(\ell, a_1, \dots, a_{k-1}) + T(\ell, b_1, \dots, b_{k-1}) \\ & \mid \forall i: p_i = a_i + b_i \} \end{aligned}$$

- Time per entry: $\Theta((N_1 + 1) \cdots (N_{k-1} + 1)) = \mathcal{O}(N^{k-1})$
- Altogether: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$

Leung's Scheduling Algorithm (2/2)

- Initial values for $1 = 2^0$ machines:

$$T(0, p_1, \dots, p_{k-1}) := \left\lfloor \left(D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) / t_k \right\rfloor$$

- Recursion:

$$\begin{aligned} T(\ell + 1, p_1, \dots, p_{k-1}) \\ := \max \{ & T(\ell, a_1, \dots, a_{k-1}) + T(\ell, b_1, \dots, b_{k-1}) \\ & \mid \forall i: p_i = a_i + b_i \} \end{aligned}$$

- Time per entry: $\Theta((N_1 + 1) \cdots (N_{k-1} + 1)) = \mathcal{O}(N^{k-1})$
- Altogether: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$

Leung's Scheduling Algorithm (2/2)

- Initial values for $1 = 2^0$ machines:

$$T(0, p_1, \dots, p_{k-1}) := \left\lfloor \left(D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) / t_k \right\rfloor$$

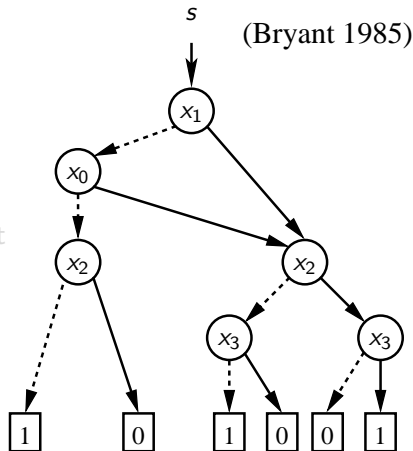
- Recursion:

$$\begin{aligned}
 &T(\ell + 1, p_1, \dots, p_{k-1}) \\
 &:= \max \{ T(\ell, a_1, \dots, a_{k-1}) + T(\ell, b_1, \dots, b_{k-1}) \\
 &\quad \mid \forall i: p_i = a_i + b_i \}
 \end{aligned}$$

- Time per entry: $\Theta((N_1 + 1) \cdots (N_{k-1} + 1)) = \mathcal{O}(N^{k-1})$
- Altogether: $\mathcal{O}(N^{2(k-1)} \cdot \log m)$

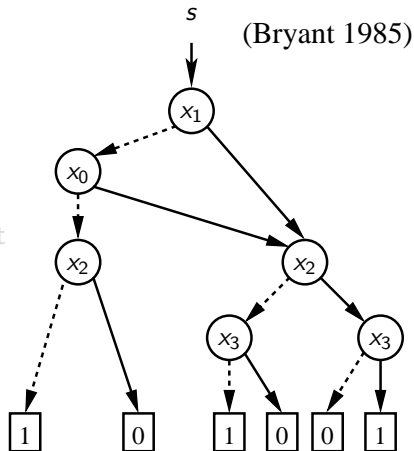
Ordered Binary Decision Diagrams (OBDDs)

- Represents $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} is acyclic digraph with **inner nodes** and **sinks**
- Inner nodes: Labeled with variable, left by 0- and 1-edge
- Sinks correspond to $f(x_0, \dots, x_{n-1})$
- Pointer marks **source node** s
- Variables are read w. r. t. $\pi \in \Sigma_n$



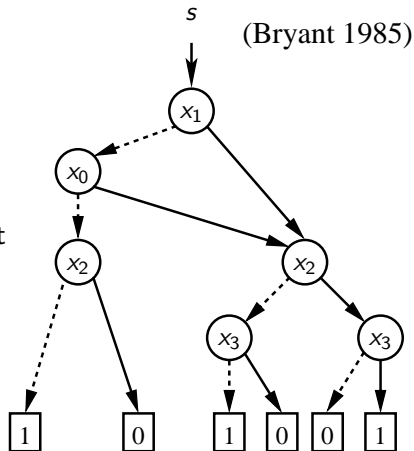
Ordered Binary Decision Diagrams (OBDDs)

- Represents $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} is acyclic digraph with **inner nodes** and **sinks**
- Inner nodes: Labeled with variable, left by 0- and 1-edge
- Sinks correspond to $f(x_0, \dots, x_{n-1})$
- Pointer marks **source node** s
- Variables are read w. r. t. $\pi \in \Sigma_n$



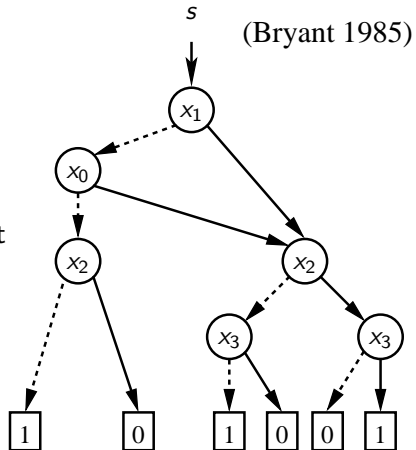
Ordered Binary Decision Diagrams (OBDDs)

- Represents $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} is acyclic digraph with **inner nodes** and **sinks**
- Inner nodes: Labeled with variable, left by 0- and 1-edge
- Sinks correspond to $f(x_0, \dots, x_{n-1})$
- Pointer marks **source node** s
- Variables are read w. r. t. $\pi \in \Sigma_n$



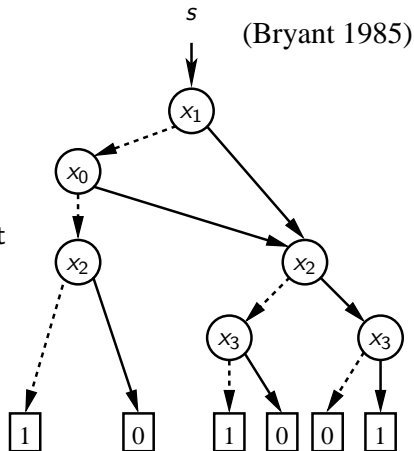
Ordered Binary Decision Diagrams (OBDDs)

- Represents $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} is acyclic digraph with **inner nodes** and **sinks**
- Inner nodes: Labeled with variable, left by 0- and 1-edge
- Sinks correspond to $f(x_0, \dots, x_{n-1})$
- Pointer marks **source node** s
- Variables are read w. r. t. $\pi \in \Sigma_n$



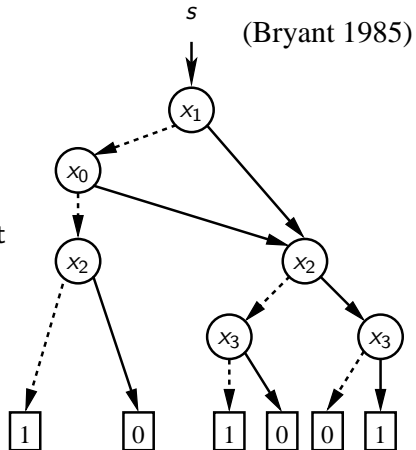
Ordered Binary Decision Diagrams (OBDDs)

- Represents $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} is acyclic digraph with **inner nodes** and **sinks**
- Inner nodes: Labeled with variable, left by 0- and 1-edge
- Sinks correspond to $f(x_0, \dots, x_{n-1})$
- Pointer marks **source node** s
- Variables are read w. r. t. $\pi \in \Sigma_n$



Ordered Binary Decision Diagrams (OBDDs)

- Represents $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$
- OBDD \mathcal{G} is acyclic digraph with **inner nodes** and **sinks**
- Inner nodes: Labeled with variable, left by 0- and 1-edge
- Sinks correspond to $f(x_0, \dots, x_{n-1})$
- Pointer marks **source node** s
- Variables are read w. r. t. $\pi \in \Sigma_n$



OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
- **Binary Synthesis:** $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
- **Quantification:** $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
- **Equivalence Check:** $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
- **Variable Replacement:** $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
- **Binary Synthesis:** $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
- **Quantification:** $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
- **Equivalence Check:** $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
- **Variable Replacement:** $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
- **Binary Synthesis:** $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
- **Quantification:** $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
- **Equivalence Check:** $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
- **Variable Replacement:** $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
 - Binary Synthesis: $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
 - Quantification: $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
 - Equivalence Check: $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
 - Variable Replacement: $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
- **Binary Synthesis:** $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
- **Quantification:** $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
- **Equivalence Check:** $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
- **Variable Replacement:** $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
- **Binary Synthesis:** $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
- **Quantification:** $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
- **Equivalence Check:** $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
- **Variable Replacement:** $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
- **Binary Synthesis:** $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
- **Quantification:** $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
- **Equivalence Check:** $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
- **Variable Replacement:** $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

OBDDs – Properties and Operations

- OBDD are well-established in CAD and Model Checking
- Every function f on n vars. has an OBDD of size $(2 + o(1))2^n/n$
- Hope for structured function: OBDD size $\text{poly}(n)$
- Efficient operations on OBDDs \mathcal{G}_f and \mathcal{G}_h :
- **Binary Synthesis:** $f \otimes h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) \cdot \text{size}(\mathcal{G}_h))$
- **Quantification:** $(\exists/\forall x_i)f$ in time $\mathcal{O}(\text{size}^2(\mathcal{G}_f))$
- **Equivalence Check:** $f = h$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f) + \text{size}(\mathcal{G}_h))$
- **Variable Replacement:** $f_{|x_i=0/1}$ in time $\mathcal{O}(\text{size}(\mathcal{G}_f))$

A Symbolic Scheduling Algorithm

- **Motivation:** Table size $\Omega\left(N_{\min}^{k-1} \cdot \log m\right)$ intractable
- **Hypothesis:** Table T is well-structured \Rightarrow Compact OBDD
- **Heuristic:** Represent T 's ℓ th row by Boolean function τ_ℓ

$$\tau_\ell(p_1, \dots, p_k) = 1 \Leftrightarrow T(\ell, p_1, \dots, p_{k-1}) = p_k$$

- Compute OBDDs τ_ℓ by func. operations and building blocks:

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \geq \Phi \right)$$

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \equiv \Phi \pmod{M} \right)$$

A Symbolic Scheduling Algorithm

- **Motivation:** Table size $\Omega\left(N_{\min}^{k-1} \cdot \log m\right)$ intractable
- **Hypothesis:** Table T is well-structured \Rightarrow Compact OBDD
- **Heuristic:** Represent T 's ℓ th row by Boolean function τ_ℓ

$$\tau_\ell(p_1, \dots, p_k) = 1 \Leftrightarrow T(\ell, p_1, \dots, p_{k-1}) = p_k$$

- Compute OBDDs τ_ℓ by func. operations and building blocks:

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \geq \Phi \right)$$

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \equiv \Phi \pmod{M} \right)$$

A Symbolic Scheduling Algorithm

- **Motivation:** Table size $\Omega\left(N_{\min}^{k-1} \cdot \log m\right)$ intractable
- **Hypothesis:** Table T is well-structured \Rightarrow Compact OBDD
- **Heuristic:** Represent T 's ℓ th row by Boolean function τ_ℓ

$$\tau_\ell(p_1, \dots, p_k) = 1 :\Leftrightarrow T(\ell, p_1, \dots, p_{k-1}) = p_k$$

- Compute OBDDs τ_ℓ by func. operations and building blocks:

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \geq \Phi \right)$$

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \equiv \Phi \pmod{M} \right)$$

A Symbolic Scheduling Algorithm

- **Motivation:** Table size $\Omega\left(N_{\min}^{k-1} \cdot \log m\right)$ intractable
- **Hypothesis:** Table T is well-structured \Rightarrow Compact OBDD
- **Heuristic:** Represent T 's ℓ th row by Boolean function τ_ℓ

$$\tau_\ell(p_1, \dots, p_k) = 1 \Leftrightarrow T(\ell, p_1, \dots, p_{k-1}) = p_k$$

- Compute OBDDs τ_ℓ by func. operations and building blocks:

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \geq \Phi \right)$$

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \equiv \Phi \pmod{M} \right)$$

A Symbolic Scheduling Algorithm

- **Motivation:** Table size $\Omega\left(N_{\min}^{k-1} \cdot \log m\right)$ intractable
- **Hypothesis:** Table T is well-structured \Rightarrow Compact OBDD
- **Heuristic:** Represent T 's ℓ th row by Boolean function τ_ℓ

$$\tau_\ell(p_1, \dots, p_k) = 1 :\Leftrightarrow T(\ell, p_1, \dots, p_{k-1}) = p_k$$

- Compute OBDDs τ_ℓ by func. operations and building blocks:

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \geq \Phi \right)$$

$$f(p_1, \dots, p_k) = \left(\sum_{i=1}^k w_i \cdot p_i \equiv \Phi \pmod{M} \right)$$

A Symbolic Scheduling Algorithm: Computing τ_0

- Problem: τ_0 contains division and floor function
- \Rightarrow Rewrite τ_0 s. t. it consists solely of building blocks

$$\begin{aligned}
 \tau_0(p_1, \dots, p_k) = 1 &\Leftrightarrow p_k = \left\lfloor \left(D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) / t_k \right\rfloor \\
 &\Leftrightarrow (\exists y, z) \left[\left(y = D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) \wedge (|z| < t_k) \right. \\
 &\quad \left. \wedge (y - z \equiv 0 \pmod{t_k}) \wedge (y = t_k \cdot p_k + z) \right]
 \end{aligned}$$

A Symbolic Scheduling Algorithm: Computing τ_0

- Problem: τ_0 contains division and floor function
- \Rightarrow Rewrite τ_0 s. t. it consists solely of building blocks

$$\begin{aligned}
 \tau_0(p_1, \dots, p_k) = 1 &\Leftrightarrow p_k = \left\lfloor \left(D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) / t_k \right\rfloor \\
 &\Leftrightarrow (\exists y, z) \left[\left(y = D - \sum_{i=1}^{k-1} t_i \cdot p_i \right) \wedge (|z| < t_k) \right. \\
 &\quad \left. \wedge (y - z \equiv 0 \pmod{t_k}) \wedge (y = t_k \cdot p_k + z) \right]
 \end{aligned}$$

A Symbolic Scheduling Algorithm: Computing $\tau_{\ell+1}$

Recursion:

$$h_{\ell+1}(p_1, \dots, p_k) :=$$

$$(\exists a_1, b_1, \dots, a_k, b_k) \left[\bigwedge_{i=1}^k (p_i = a_i + b_i) \right.$$

$$\left. \wedge \tau_{\ell}(a_1, \dots, a_k) \wedge \tau_{\ell}(b_1, \dots, b_k) \right]$$

Maximality:

$$\tau_{\ell+1}(p_1, \dots, p_k) := h_{\ell+1}(p_1, \dots, p_k)$$

$$\wedge \overline{(\exists y) [(y > p_k) \wedge h_{\ell+1}(p_1, \dots, p_{k-1}, y)]}$$

A Symbolic Scheduling Algorithm: Computing $\tau_{\ell+1}$

Recursion:

$$h_{\ell+1}(p_1, \dots, p_k) :=$$

$$(\exists a_1, b_1, \dots, a_k, b_k) \left[\bigwedge_{i=1}^k (p_i = a_i + b_i) \right.$$

$$\left. \wedge \tau_{\ell}(a_1, \dots, a_k) \wedge \tau_{\ell}(b_1, \dots, b_k) \right]$$

Maximality:

$$\tau_{\ell+1}(p_1, \dots, p_k) := h_{\ell+1}(p_1, \dots, p_k)$$

$$\overline{\wedge (\exists y) [(y > p_k) \wedge h_{\ell+1}(p_1, \dots, p_{k-1}, y)]}$$

A Symbolic Scheduling Algorithm: Computing $\tau_{\ell+1}$

Recursion:

$$h_{\ell+1}(p_1, \dots, p_k) :=$$

$$(\exists a_1, b_1, \dots, a_k, b_k) \left[\bigwedge_{i=1}^k (p_i = a_i + b_i) \right.$$

$$\left. \wedge \tau_{\ell}(a_1, \dots, a_k) \wedge \tau_{\ell}(b_1, \dots, b_k) \right]$$

Maximality:

$$\tau_{\ell+1}(p_1, \dots, p_k) := h_{\ell+1}(p_1, \dots, p_k)$$

$$\overline{\wedge (\exists y) [(y > p_k) \wedge h_{\ell+1}(p_1, \dots, p_{k-1}, y)]}$$

A Symbolic Scheduling Algorithm: Computing $\tau_{\ell+1}$

Recursion:

$$h_{\ell+1}(p_1, \dots, p_k) :=$$

$$(\exists a_1, b_1, \dots, a_k, b_k) \left[\bigwedge_{i=1}^k (p_i = a_i + b_i) \right.$$

$$\left. \wedge \tau_{\ell}(a_1, \dots, a_k) \wedge \tau_{\ell}(b_1, \dots, b_k) \right]$$

Maximality:

$$\tau_{\ell+1}(p_1, \dots, p_k) := h_{\ell+1}(p_1, \dots, p_k)$$

$$\overline{\wedge (\exists y) [(y > p_k) \wedge h_{\ell+1}(p_1, \dots, p_{k-1}, y)]}$$

A Symbolic Scheduling Algorithm: Computing $\tau_{\ell+1}$

Recursion:

$$h_{\ell+1}(p_1, \dots, p_k) :=$$

$$(\exists a_1, b_1, \dots, a_k, b_k) \left[\bigwedge_{i=1}^k (p_i = a_i + b_i) \right.$$

$$\left. \wedge \tau_{\ell}(a_1, \dots, a_k) \wedge \tau_{\ell}(b_1, \dots, b_k) \right]$$

Maximality:

$$\tau_{\ell+1}(p_1, \dots, p_k) := h_{\ell+1}(p_1, \dots, p_k)$$

$$\wedge \overline{(\exists y) [(y > p_k) \wedge h_{\ell+1}(p_1, \dots, p_{k-1}, y)]}$$

A Symbolic Scheduling Algorithm: Computing $\tau_{\ell+1}$

Recursion:

$$h_{\ell+1}(p_1, \dots, p_k) :=$$

$$(\exists a_1, b_1, \dots, a_k, b_k) \left[\bigwedge_{i=1}^k (p_i = a_i + b_i) \right.$$

$$\left. \wedge \tau_{\ell}(a_1, \dots, a_k) \wedge \tau_{\ell}(b_1, \dots, b_k) \right]$$

Maximality:

$$\tau_{\ell+1}(p_1, \dots, p_k) := h_{\ell+1}(p_1, \dots, p_k)$$

$$\wedge \overline{(\exists y) [(y > p_k) \wedge h_{\ell+1}(p_1, \dots, p_{k-1}, y)]}$$

A Symbolic Scheduling Algorithm: Computing $\tau_{\ell+1}$

Recursion:

$$h_{\ell+1}(p_1, \dots, p_k) :=$$

$$(\exists a_1, b_1, \dots, a_k, b_k) \left[\bigwedge_{i=1}^k (p_i = a_i + b_i) \right.$$

$$\left. \wedge \tau_{\ell}(a_1, \dots, a_k) \wedge \tau_{\ell}(b_1, \dots, b_k) \right]$$

Maximality:

$$\tau_{\ell+1}(p_1, \dots, p_k) := h_{\ell+1}(p_1, \dots, p_k)$$

$$\wedge \overline{(\exists y) [(y > p_k) \wedge h_{\ell+1}(p_1, \dots, p_{k-1}, y)]}$$

A Symbolic Scheduling Algorithm: Time/Space

Theorem

The algorithm executes $\mathcal{O}(k \cdot \log m \cdot \log(mD))$ OBDD operations on OBDDs with $\Theta(k \cdot \log(mD))$ Boolean variables.

- Time for each operation on some OBDD \mathcal{G} : $\mathcal{O}(\text{size}^2(\mathcal{G}))$
- Trivial bound on OBDD size: $\mathcal{O}((mD)^{3k+2})$
(by bound $(2 + o(1))2^n/n$ for every $f: \{0, 1\}^n \rightarrow \{0, 1\}$)
- Hope: Much more compact OBDDs due to structure of T !
- \Rightarrow Experiments on random instances

A Symbolic Scheduling Algorithm: Time/Space

Theorem

The algorithm executes $\mathcal{O}(k \cdot \log m \cdot \log(mD))$ OBDD operations on OBDDs with $\Theta(k \cdot \log(mD))$ Boolean variables.

- Time for each operation on some OBDD \mathcal{G} : $\mathcal{O}(\text{size}^2(\mathcal{G}))$
- Trivial bound on OBDD size: $\mathcal{O}((mD)^{3k+2})$
(by bound $(2 + o(1))2^n/n$ for every $f: \{0, 1\}^n \rightarrow \{0, 1\}$)
- Hope: Much more compact OBDDs due to structure of T !
- \Rightarrow Experiments on random instances

A Symbolic Scheduling Algorithm: Time/Space

Theorem

The algorithm executes $\mathcal{O}(k \cdot \log m \cdot \log(mD))$ OBDD operations on OBDDs with $\Theta(k \cdot \log(mD))$ Boolean variables.

- Time for each operation on some OBDD \mathcal{G} : $\mathcal{O}(\text{size}^2(\mathcal{G}))$
- Trivial bound on OBDD size: $\mathcal{O}((mD)^{3k+2})$
(by bound $(2 + o(1))2^n/n$ for every $f: \{0, 1\}^n \rightarrow \{0, 1\}$)
- Hope: Much more compact OBDDs due to structure of T !
- \Rightarrow Experiments on random instances

A Symbolic Scheduling Algorithm: Time/Space

Theorem

The algorithm executes $\mathcal{O}(k \cdot \log m \cdot \log(mD))$ OBDD operations on OBDDs with $\Theta(k \cdot \log(mD))$ Boolean variables.

- Time for each operation on some OBDD \mathcal{G} : $\mathcal{O}(\text{size}^2(\mathcal{G}))$
- Trivial bound on OBDD size: $\mathcal{O}((mD)^{3k+2})$
(by bound $(2 + o(1))2^n/n$ for every $f: \{0, 1\}^n \rightarrow \{0, 1\}$)
- Hope: Much more compact OBDDs due to structure of T !
- \Rightarrow Experiments on random instances

Experimental Setting

Properties of random instances:

- Expected load $E[L] := \sum_{i=1}^k t_i \cdot N_i = (mD)/1.2$
- Load L is distributed uniformly onto task types $1, \dots, k$
- Task count N_i is drawn uniformly due to mean parameter $E[N_i]$
- Execution time t_i is drawn due to **uniform**, **exponential**, or **Erlang** distribution

Chosen parameters:

- Three series with $mD = 800, 1600, 3200$ and $k = 3$
- Within each series: $m = 2, 4, 8, 16, 32$
- 10 values for $E[N_i] \in \{(mD)/6, \dots, (mD)/3\}$
- 20 random instances per setting

Experimental Setting

Properties of random instances:

- Expected load $E[L] := \sum_{i=1}^k t_i \cdot N_i = (mD)/1.2$
- Load L is distributed uniformly onto task types $1, \dots, k$
- Task count N_i is drawn uniformly due to mean parameter $E[N_i]$
- Execution time t_i is drawn due to **uniform**, **exponential**, or **Erlang** distribution

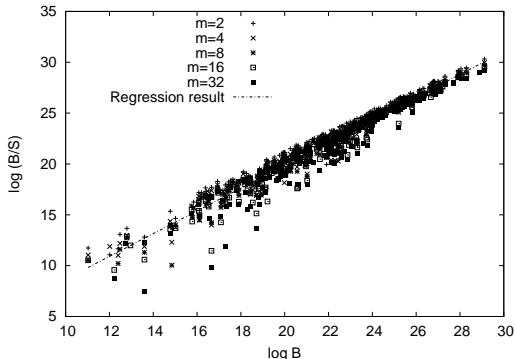
Chosen parameters:

- Three series with $mD = 800, 1600, 3200$ and $k = 3$
- Within each series: $m = 2, 4, 8, 16, 32$
- 10 values for $E[N_i] \in \{(mD)/6, \dots, (mD)/3\}$
- 20 random instances per setting

Experimental Results

Example:

- $mD = 3200$, $k = 3$
- Exponential distr. t_i 's
- Plot: $\log(B/S)$ vs. $\log B$



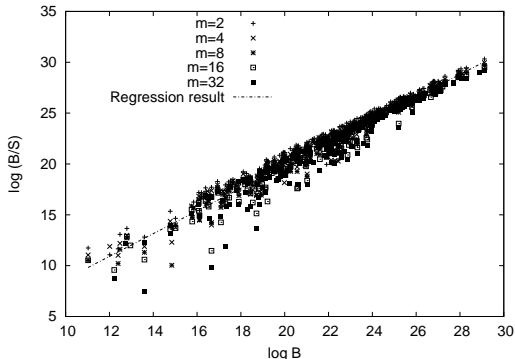
Observations:

- Small N , large t_i 's \Rightarrow small T , large OBDDs
- Large N , small t_i 's \Rightarrow large T , small OBDDs
- Break-even point depends on mD , k , and t_i -distribution

Experimental Results

Example:

- $mD = 3200$, $k = 3$
- Exponential distr. t_i s
- Plot: $\log(B/S)$ vs. $\log B$



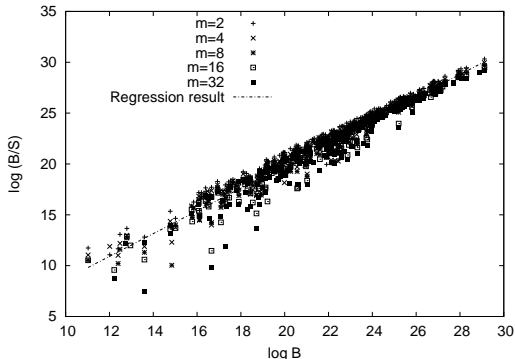
Observations:

- Small N , large t_i s \Rightarrow small T , large OBDDs
- Large N , small t_i s \Rightarrow large T , small OBDDs
- Break-even point depends on mD , k , and t_i -distribution

Experimental Results

Example:

- $mD = 3200$, $k = 3$
- Exponential distr. t_i s
- Plot: $\log(B/S)$ vs. $\log B$



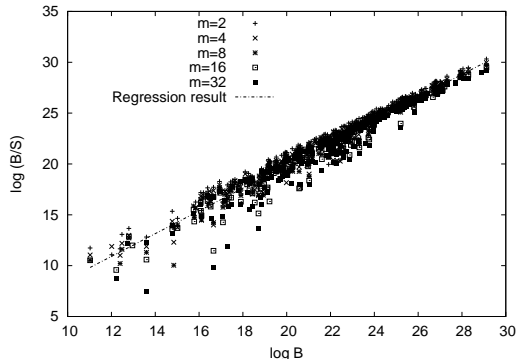
Observations:

- Small N , large t_i s \Rightarrow small T , large OBDDs
- Large N , small t_i s \Rightarrow large T , small OBDDs
- Break-even point depends on mD , k , and t_i -distribution

Experimental Results

Example:

- $mD = 3200$, $k = 3$
- Exponential distr. t_i s
- Plot: $\log(B/S)$ vs. $\log B$



Observations:

- Small N , large t_i s \Rightarrow small T , large OBDDs
- Large N , small t_i s \Rightarrow large T , small OBDDs
- Break-even point depends on mD , k , and t_i -distribution

Conclusions

- For the first time, OBDDs have been applied in Dynamic Programming and scheduling
- Many jobs with small execution times \Rightarrow Structured DP table
- Structured DP table enables OBDD-based algorithm to beat Leung's algorithm

Conclusions

- For the first time, OBDDs have been applied in Dynamic Programming and scheduling
- Many jobs with small execution times \Rightarrow Structured DP table
- Structured DP table enables OBDD-based algorithm to beat Leung's algorithm

Conclusions

- For the first time, OBDDs have been applied in Dynamic Programming and scheduling
- Many jobs with small execution times \Rightarrow Structured DP table
- Structured DP table enables OBDD-based algorithm to beat Leung's algorithm

“That’s all Folks!”