Improved tardiness bounds for Global EDF

Jeremy Erickson Sanjoy Baruah UmaMaheswari Devi

University of North Carolina at Chapel Hill

IBM Research Lab, Bangalore, India

January 27, 2010

Devi/Anderson Bounds - Basic Idea

- Bound tardiness of each task at $x + C_i$ for some x.

Devi/Anderson Bounds - Basic Idea

- Bound tardiness of each task at $x + C_i$ for some x.
- Nontrivial part is finding x.

Devi/Anderson Bounds - Basic Idea

- Bound tardiness of each task at $x + C_i$ for some x.
- Nontrivial part is finding x.
- Bound does vary per task, but x does not.

Devi/Anderson Bounds - Specifics

- Devi & Anderson 2005 and later papers report several bounds on the tardiness of EDF.

Devi/Anderson Bounds - Specifics

- Devi & Anderson 2005 and later papers report several bounds on the tardiness of EDF.
- Derived in 2005 conference paper:

$$C_i + \frac{C_{\text{sum}} - C_{\text{min}}}{m - U_{\text{sum}}} \tag{1}$$

Devi/Anderson Bounds - Specifics

- Devi & Anderson 2005 and later papers report several bounds on the tardiness of EDF.
- Derived in 2005 conference paper:

$$C_i + \frac{C_{\mathrm{sum}} - C_{\mathrm{min}}}{m - U_{\mathrm{sum}}} \quad (1)$$

- Improved EDF-BASIC: Use only $m - 2$ execution values.

Devi/Anderson Bounds - Specifics

- Devi & Anderson 2005 and later papers report several bounds on the tardiness of EDF.
- Derived in 2005 conference paper:

$$C_i + \frac{C_{\text{sum}} - C_{\text{min}}}{m - U_{\text{sum}}} \tag{1}$$

- Improved EDF-BASIC: Use only $m - 2$ execution values.
- Further improved EDF-ITER: Like EDF-BASIC, but only use values from selected $m - 1$ tasks.

Our Improvement

- Use different x_i value for each task.

Our Improvement

- Use different x_i value for each task.
- In worst case, becomes same results as Devi/Anderson.

Our Improvement

- Use different x_i value for each task.
- In worst case, becomes same results as Devi/Anderson.
- Use concept of a compliant vector.

$\mathbf{L}(\vec{x})$

- Vector $\vec{x}$ with x_i for each task

$\mathbf{L}(\vec{x})$

- Vector $\vec{x}$ with x_i for each task
- $$\mathbf{L}(\vec{x}) = \sum_{(m-1) \text{ largest}} \left(x_i U_i + C_i \right) \tag{2}$$

$\mathbf{L}(\vec{x})$

- Vector $\vec{x}$ with x_i for each task
- $$\mathbf{L}(\vec{x}) = \sum_{(m-1) \text{ largest}} \left(x_i U_i + C_i\right) \tag{2}$$
- Can use improved definition $\mathbf{L}(\vec{x})$: the largest sum obtained by summing $(m-2)$ of the $(x_i U_i + C_i)$'s plus an additional C_i.

Compliant Vector

- Using $\mathbf{L}(\vec{x})$ as defined, a vector is *compliant* iff $\forall i$,

$$\frac{\mathbf{L}(\vec{x}) - C_i}{m} \leq x_i \tag{3}$$

Compliant Vector

- Using $\mathbf{L}(\vec{x})$ as defined, a vector is *compliant* iff $\forall i$,

$$\frac{\mathbf{L}(\vec{x}) - C_i}{m} \leq x_i \qquad (3)$$

- A compliant vector is *minimal* if reducing any one component would produce a non-compliant vector.

Theorem 1

Theorem

Let $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ *denote any compliant vector. For each* $\tau_i \in \tau$*, each job generated by task* τ_i *completes no later than* $(C_i + x_i)$ *time units after its deadline.*

- Rather than using **LAG** (as in previous papers), use $W(t)$

Theorem 1

Theorem

Let $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ denote any compliant vector. For each $\tau_i \in \tau$, each job generated by task τ_i completes no later than $(C_i + x_i)$ time units after its deadline.

- Rather than using **LAG** (as in previous papers), use $W(t)$
- I = set of jobs with deadlines no later than t.

Theorem 1

Theorem

Let $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ denote any compliant vector. For each $\tau_i \in \tau$, each job generated by task τ_i completes no later than $(C_i + x_i)$ time units after its deadline.

- Rather than using **LAG** (as in previous papers), use $W(t)$
- I = set of jobs with deadlines no later than t.
- $W(t) = \sum_{\text{jobs in } I}(C_i - \text{work completed before } t)$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$
  - Tardy at end of interval - contribute $U_j(d_k - t_{i+1}) + U_j x_j + C_j$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$
  - Tardy at end of interval - contribute $U_j(d_k - t_{i+1}) + U_j x_j + C_j$
  - Not tardy at end, but running - contribute $U_j(d_k - t_{i+1}) + C_j$

First Lemma

Lemma

For all $t \in [0, d_k)$,

$$W(t) \leq U(\tau) \times (d_k - t) + \mathbf{L}(\vec{x})$$

- We induct over busy and nonbusy intervals
- Busy intervals - trivial
- Nonbusy intervals - several subcases
  - Not running through interval - contribute $U_j(d_k - t_{i+1})$
  - Tardy at end of interval - contribute $U_j(d_k - t_{i+1}) + U_j x_j + C_j$
  - Not tardy at end, but running - contribute $U_j(d_k - t_{i+1}) + C_j$
- Summing contributions reveals claimed upper bound

Second Lemma

Lemma

The job of τ_k with deadline d_k completes by time-instant $d_k + x_k + C_k$.

- Use previous lemma to determine that at most $L(x)$ work is left at d_k

Second Lemma

Lemma

The job of τ_k with deadline d_k completes by time-instant $d_k + x_k + C_k$.

- Use previous lemma to determine that at most $L(x)$ work is left at d_k
- Bound follows from here

Second Lemma

Lemma

The job of τ_k with deadline d_k completes by time-instant $d_k + x_k + C_k$.

- Use previous lemma to determine that at most $L(x)$ work is left at d_k
- Bound follows from here
- After this, we're done

Algorithm for computing minimal compliant vector

FINDCOMPLIANTVECTOR

$\vec{x} \leftarrow \langle 0, 0, \ldots, 0 \rangle$ $\triangleright$ Initialize (to a non-compliant vector)
repeat
    Let τ_i denote any task violating constraint
    Let $\hat{x}_i$ denote smallest value of x_i satisfying constraint
    Replace x_i by $\hat{x}_i$ in $\vec{x}$
until $\vec{x}$ is a compliant vector

Minimality of Computed Vector

Theorem

Procedure FINDCOMPLIANTVECTOR *returns a minimal compliant vector.*

Lemma

For all $j \geq 0$, $\mathbf{L}(\vec{x}_j) \leq \mathbf{L}(\vec{x}_f)$.

- Increasing an x_i value can only increase $\mathbf{L}(\vec{x})$.

Minimality of Computed Vector

Theorem

Procedure FINDCOMPLIANTVECTOR *returns a minimal compliant vector.*

Lemma

For all $j \geq 0$, $\mathbf{L}(\vec{x}_j) \leq \mathbf{L}(\vec{x}_f)$.

- Increasing an x_i value can only increase $\mathbf{L}(\vec{x})$.
- Each bound, when set, was tight, so at end, all bounds tight.

Complexity

- No bound known on runtime - seems very large from experiments

Complexity

- No bound known on runtime - seems very large from experiments
- Can make pseudo-polynomial by setting minimum increase ϵ

Complexity

- No bound known on runtime - seems very large from experiments
- Can make pseudo-polynomial by setting minimum increase ϵ
- Runs tens to thousands of iterations with $\epsilon = .1$ in experiments

Complexity

- No bound known on runtime - seems very large from experiments
- Can make pseudo-polynomial by setting minimum increase ϵ
- Runs tens to thousands of iterations with $\epsilon = .1$ in experiments
- Additive error bounded by $m\epsilon$

Experimental Setup

- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$.

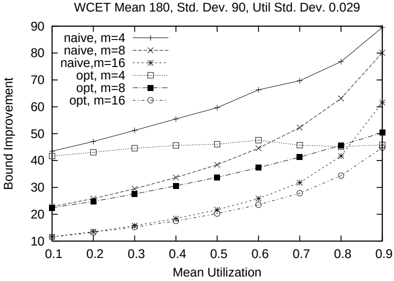
Experimental Setup

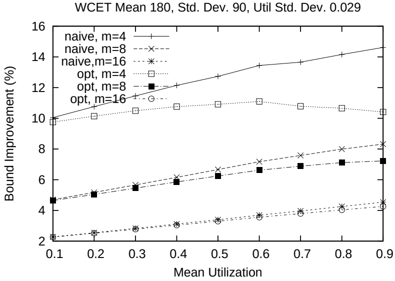
- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$.
- Random task sets - blocks of 1,000 tasks for each parameter tested.

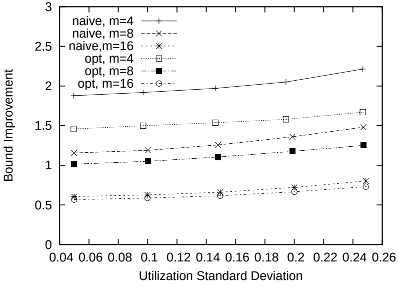
Experimental Setup

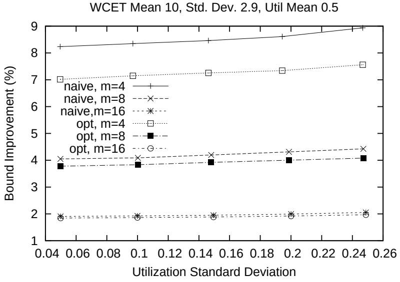
- Used psuedo-polynomial approximation algorithm with $\epsilon = .1$.
- Random task sets - blocks of 1,000 tasks for each parameter tested.
- Always used uniform distribution over $\mathbb{R}$.

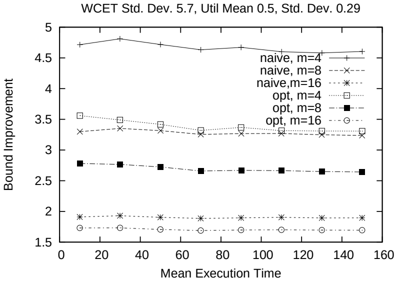
Experimental Results

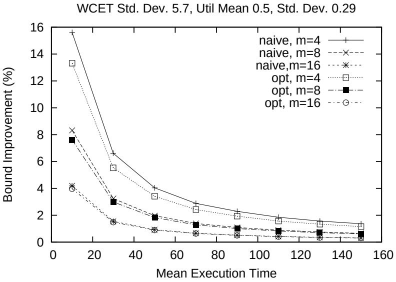




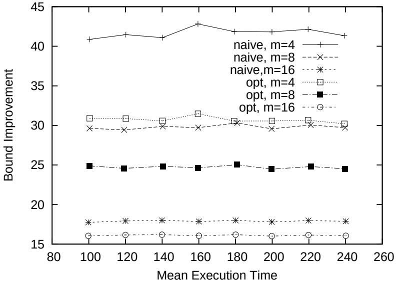

naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
Bound Improvement
Utilization Standard Deviation
0
0.5
1
1.5
2
2.5
3
0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2 0.22 0.24 0.26

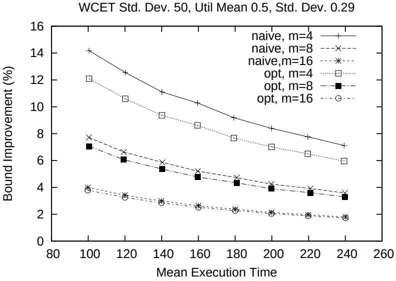

WCET Mean 10, Std. Dev. 2.9, Util Mean 0.5
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
Bound Improvement (%)
Utilization Standard Deviation
1
2
3
4
5
6
7
8
9
0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2 0.22 0.24 0.26

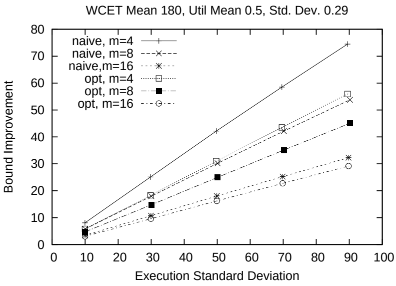

WCET Std. Dev. 5.7, Util Mean 0.5, Std. Dev. 0.29
Bound Improvement
Mean Execution Time
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16

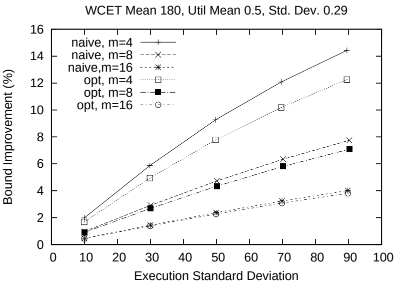

WCET Std. Dev. 5.7, Util Mean 0.5, Std. Dev. 0.29
Bound Improvement (%)
Mean Execution Time
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16

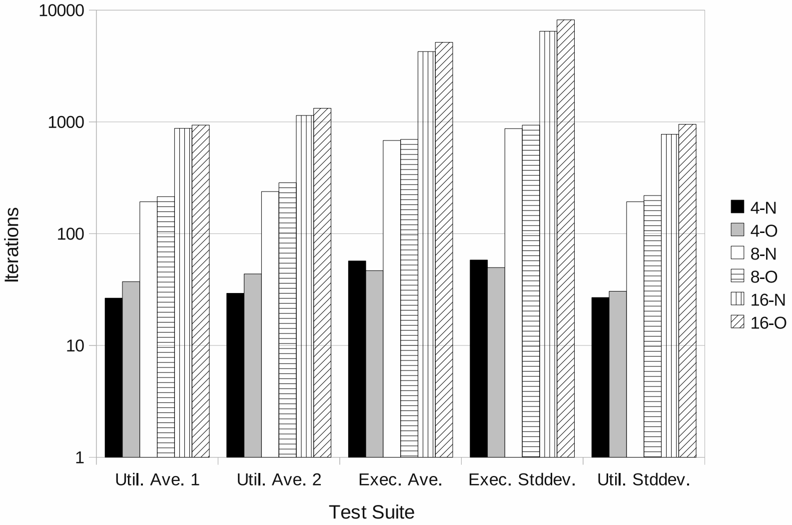
Experimental Results

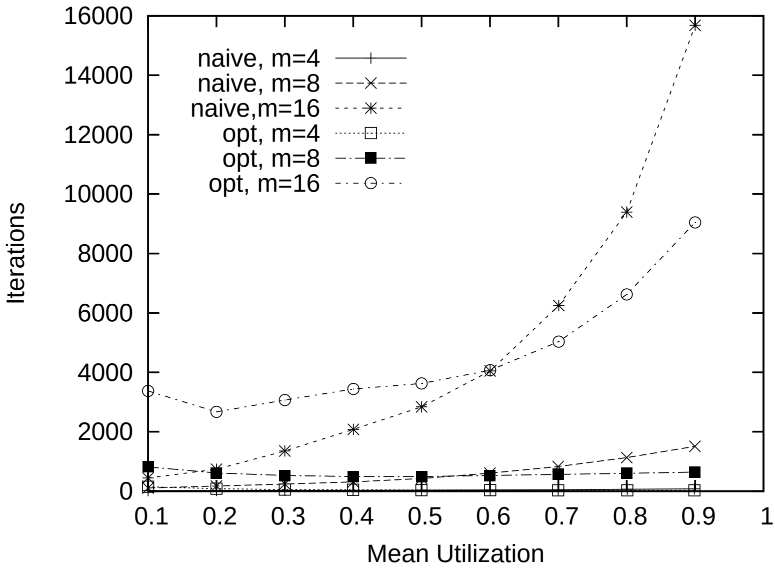


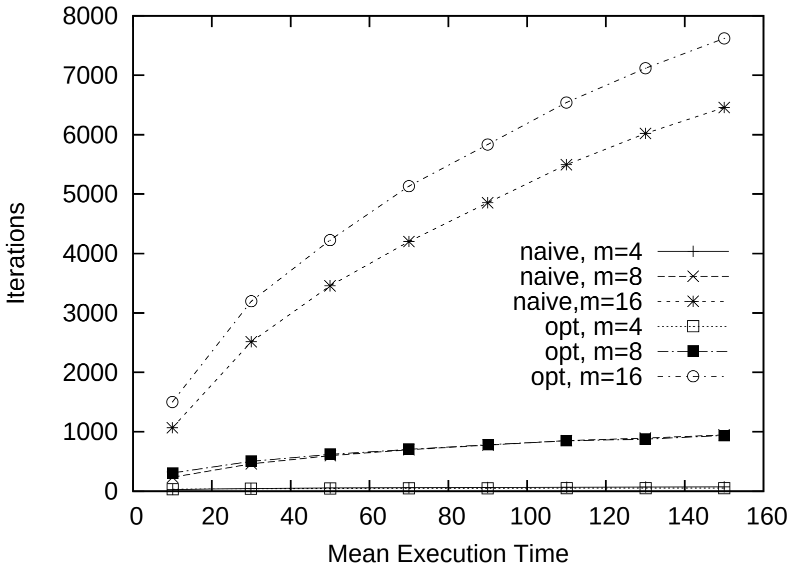


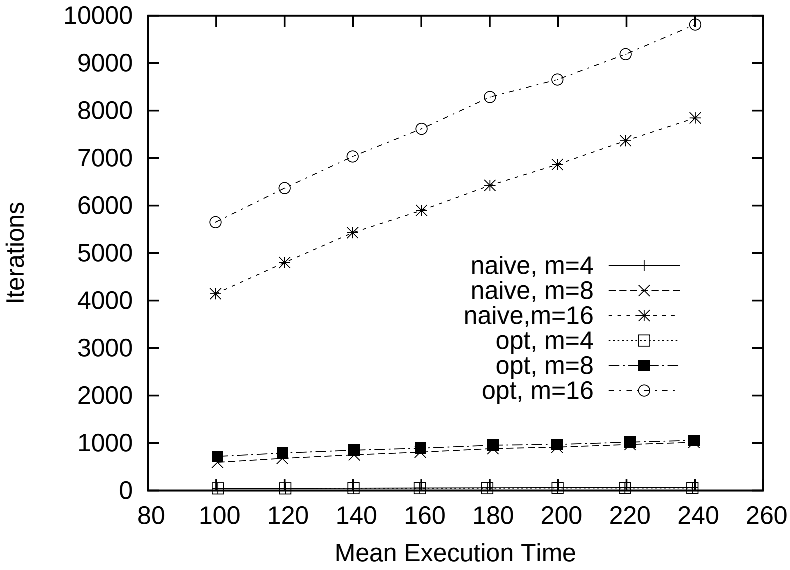

WCET Mean 180, Util Mean 0.5, Std. Dev. 0.29
Bound Improvement
Execution Standard Deviation
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
0
10
20
30
40
50
60
70
80
90
100

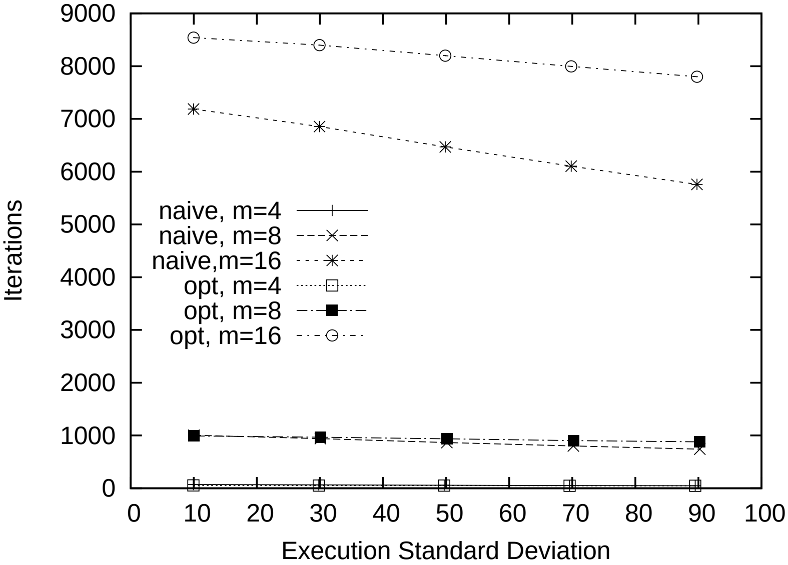

WCET Mean 180, Util Mean 0.5, Std. Dev. 0.29
Bound Improvement (%)
Execution Standard Deviation
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
0
2
4
6
8
10
12
14
16
10
20
30
40
50
60
70
80
90
100

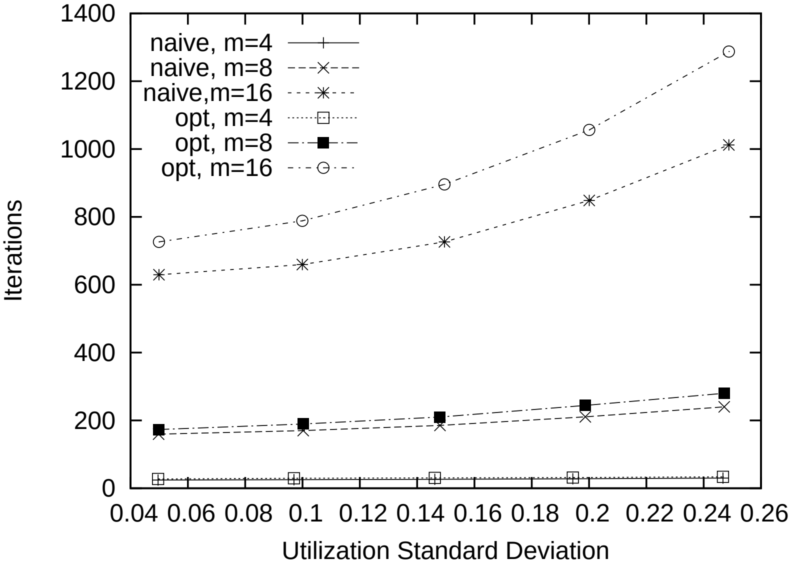

10000
1000
100
10
1
Iterations
Util. Ave. 1
Util. Ave. 2
Exec. Ave.
Exec. Stddev.
Util. Stddev.
Test Suite
4-N
4-O
8-N
8-O
16-N
16-O


16000
14000
12000
10000
8000
6000
4000
2000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Iterations
Mean Utilization
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16


Iterations
Mean Execution Time
0
1000
2000
3000
4000
5000
6000
7000
8000
0
20
40
60
80
100
120
140
160
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16


Iterations
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
80
100
120
140
160
180
200
220
240
260
Mean Execution Time
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Iterations
naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
0
10
20
30
40
50
60
70
80
90
100
Execution Standard Deviation


naive, m=4
naive, m=8
naive,m=16
opt, m=4
opt, m=8
opt, m=16
Iterations
0
200
400
600
800
1000
1200
1400
0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2 0.22 0.24 0.26
Utilization Standard Deviation

Faster Algorithm (Unpublished)

- Can demonstrate that all bounds are tight for a minimal compliant vector.

Faster Algorithm (Unpublished)

- Can demonstrate that all bounds are tight for a minimal compliant vector.
- Thus, only need to determine $\mathbf{L}(\vec{x})$ and verify that it creates a compliant vector.

Faster Algorithm (Unpublished)

- Can demonstrate that all bounds are tight for a minimal compliant vector.
- Thus, only need to determine $\mathbf{L}(\vec{x})$ and verify that it creates a compliant vector.
- This can be done with an efficient binary search, with the number of iterations set to the bits of accuracy for $\frac{\mathbf{L}(\vec{x})}{m}$.

Faster Algorithm (Unpublished)

- Can demonstrate that all bounds are tight for a minimal compliant vector.
- Thus, only need to determine $\mathbf{L}(\vec{x})$ and verify that it creates a compliant vector.
- This can be done with an efficient binary search, with the number of iterations set to the bits of accuracy for $\frac{\mathbf{L}(\vec{x})}{m}$.
- In practice, produces much better bounds in far fewer iterations.

Faster Algorithm (Unpublished)

- Can demonstrate that all bounds are tight for a minimal compliant vector.
- Thus, only need to determine $\mathbf{L}(\vec{x})$ and verify that it creates a compliant vector.
- This can be done with an efficient binary search, with the number of iterations set to the bits of accuracy for $\frac{\mathbf{L}(\vec{x})}{m}$.
- In practice, produces much better bounds in far fewer iterations.
- Still working on theory.

Review

- Devi/Anderson Bounds
- Proof of improved bounds
- Approximation algorithm
- Experimental results