An EPTAS for Scheduling Fork-Join Graphs with Communication Delay

Klaus Jansen[1], **Oliver Sinnen[2]**, Huijun (Tony) Wang[2],

[1]: Department of Computer Science
University of Kiel, Germany
[2]: Parallel and Reconfigurable Computing Lab
Department of Electrical, Computer and Software Engineering
University of Auckland, New Zealand

Overview

Problem domain

Scheduling task graphs (DAGs) with communication delays on homogeneous processors – $P|prec, c_{ij}|C_{max}$

- No known constant approximation algorithm
- No PTAS

Overview

Problem domain

Scheduling task graphs (DAGs) with communication delays on homogeneous processors – $P|prec, c_{ij}|C_{max}$

- No known constant approximation algorithm
- No PTAS

Here today

EPTAS (Efficient Polynomial Time Approximation Scheme) for **fork-join** graphs – $P|fork - join, c_{ij}|C_{max}$

- Polynomial 2-approximation algorithm [Aussois 2018]
- Very important graph structure, realistic for many computations
- Very hard to schedule optimally (maximum degree of freedom, but order and allocation matters)

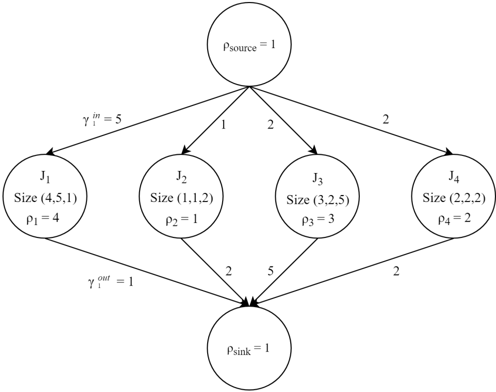
Content

Content

Scheduling problem

Fork-join graph

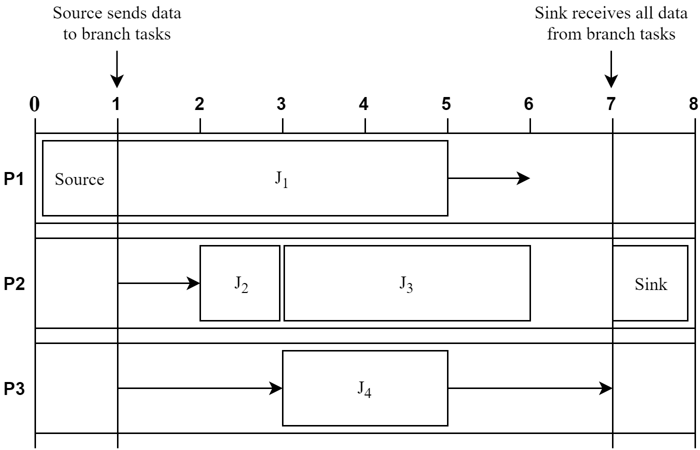
- j_{source}, j_{sink}: source task, sink task
- $j \in J$: branch task
- $(\rho_j, \gamma_j^{in}, \gamma_j^{out}) \in P \times \Gamma \times \Gamma$
  - task size
  - (comp., in-comm., out-comm.)
  - only remote comm. costs!
  - all $\in \mathbb{N}_0$



Scheduling problem

- M identical processors (machines)
- minimizing makespan – OPT

Schedule



Source sends data
to branch tasks
Sink receives all data
from branch tasks
0
1
2
3
4
5
6
7
8
P1
Source
J_1
P2
J_2
J_3
Sink
P3
J_4

Approach overview

Idea

- Formulating problem as ILP for given T, makespan $< T$
- Binary search over makespan T
- Problem: Solving ILP is NP-complete, exponential runtime in input size

EPTAS Approach

- Accuracy parameter $\epsilon > 0$
- EPTAS gives solution $(1 + \epsilon)OPT$
  - Efficient complexity $\mathcal{O}(f(1/\epsilon) \times \text{poly}(n))$, n not in exponent
- Transforming/simplifying problem instance
  - eventual instance size is $f(1/\epsilon)$
- Use configuration ILP

Approach structure

Original Instance I

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T
$\hookrightarrow$ Solve configuration ILP

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T $\Rightarrow$ Opt. solution for I_5
$\hookrightarrow$ Solve configuration ILP

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3 ... Fill placeholders with small tasks
$\Downarrow$ ... $\Uparrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T $\hookrightarrow$ Solve configuration ILP $\Rightarrow$ Opt. solution for I_5

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T
$\hookrightarrow$ Solve configuration ILP $\Rightarrow$ Opt. solution for I_5
$\Uparrow$
Fill placeholders with small tasks
$\Uparrow$
Recover orig. processing times

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T
$\hookrightarrow$ Solve configuration ILP $\Rightarrow$ Opt. solution for I_5
$\Uparrow$
Fill placeholders with small tasks
$\Uparrow$
Recover orig. processing times
$\Uparrow$
Recover orig. comm. times

Approach structure

Original Instance I				Solution for I, $(1+\epsilon')OPT$
$\Downarrow$				$\Uparrow$
Communication truncation, I_1				Recover orig. comm. times
$\Downarrow$				$\Uparrow$
Big task truncation, I_2				Recover orig. processing times
$\Downarrow$				$\Uparrow$
Small task placeholders, I_3				Fill placeholders with small tasks
$\Downarrow$				
Placeholder arrangement, I_4				$\Uparrow$
$\Downarrow$				
Gaps resolution, I_5	$\Rightarrow$	Binary search over T $\hookrightarrow$ Solve configuration ILP	$\Rightarrow$	Opt. solution for I_5

Content

Simplification 1: Communication truncation

Normalise communication times: $\forall\gamma \in \Gamma$: truncate value to nearest multiple of $\epsilon T \Rightarrow$ new instance I_1

Simplification 1: Communication truncation

Normalise communication times: $\forall \gamma \in \Gamma$: truncate value to nearest multiple of ϵT $\Rightarrow$ new instance I_1

Lemma (Communication truncation)

Let I_1 be the instance obtained after this communication times truncation step, and T_1 be the minimum makespan for I_1. Then

$$T_1 \leq OPT \leq T_1 + 2\epsilon T$$

Simplification 2: Big tasks

Distinguish tasks

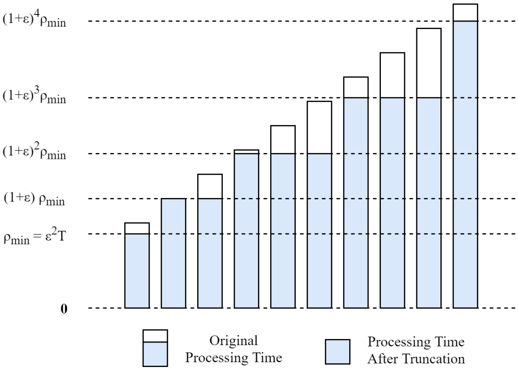
- Big tasks: $\rho_i \geq \epsilon^2 T$
- Small tasks: $\rho_i < \epsilon^2 T$

Big tasks

Normalise/truncate computation times to

$$\{(1+\epsilon)^n \epsilon^2 T \mid n \in \mathbb{N}_0\}$$

$\Rightarrow$ new instance I_2



Simplification 2: Big tasks

Distinguish tasks

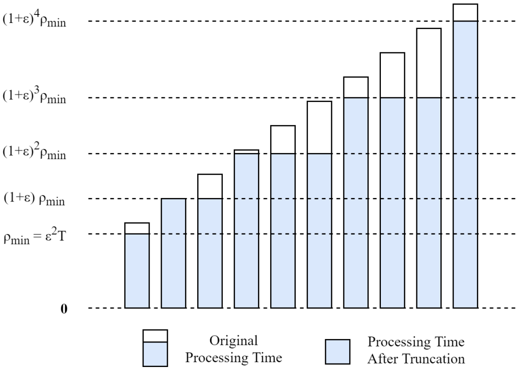
- Big tasks: $\rho_i \geq \epsilon^2 T$
- Small tasks: $\rho_i < \epsilon^2 T$

Big tasks

Normalise/truncate computation times to

$$\{(1+\epsilon)^n \epsilon^2 T \mid n \in \mathbb{N}_0\}$$

$\Rightarrow$ new instance I_2



Lemma (Big task truncation)

Let I_2 be the instance obtained by applying this task time truncation step to I_1, and T_2 be the minimum schedule makespan for I_2.

$$T_2 \leq T_1 \leq T_2 + \epsilon T$$

Simplification 3: Small task placeholders

- Remove all small tasks $J_{small}^{\gamma^{in},\gamma^{out}}$
- Replace by placeholders tasks with uniform $\rho = \epsilon^3 T$
- Number of placeholders

$$\left\lfloor \frac{\sum_{j \in J_{small}^{\gamma^{in},\gamma^{out}}} \rho_j}{\epsilon^3 T} \right\rfloor$$

$\Rightarrow$ new instance I_3

Simplification 3: Small task placeholders

- Remove all small tasks $J_{small}^{\gamma^{in},\gamma^{out}}$
- Replace by placeholders tasks with uniform $\rho = \epsilon^3 T$
- Number of placeholders

$$\left\lfloor \frac{\sum_{j \in J_{small}^{\gamma^{in},\gamma^{out}}} \rho_j}{\epsilon^3 T} \right\rfloor$$

$\Rightarrow$ new instance I_3

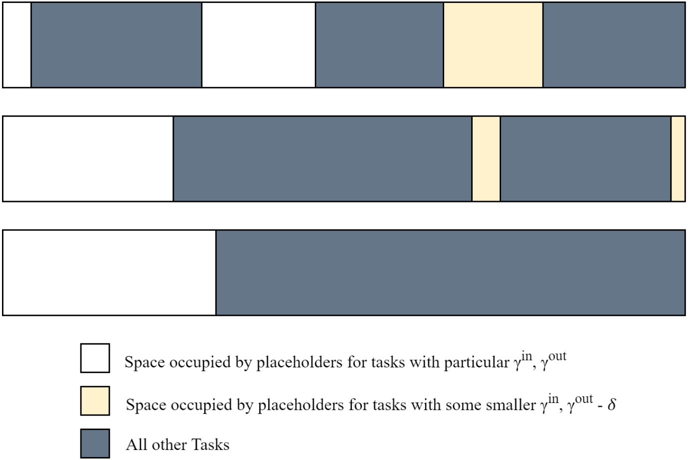
Lemma (Small tasks placeholders)

Let I_3 be the instance obtained after applying this small task replacement step to I_2, and T_3 be the minimum schedule makespan for I_3.

$$T_3 - \epsilon T \leq T_2 \leq T_3 + 2\epsilon T$$

Simplification 3: Small task placeholders – reverse step

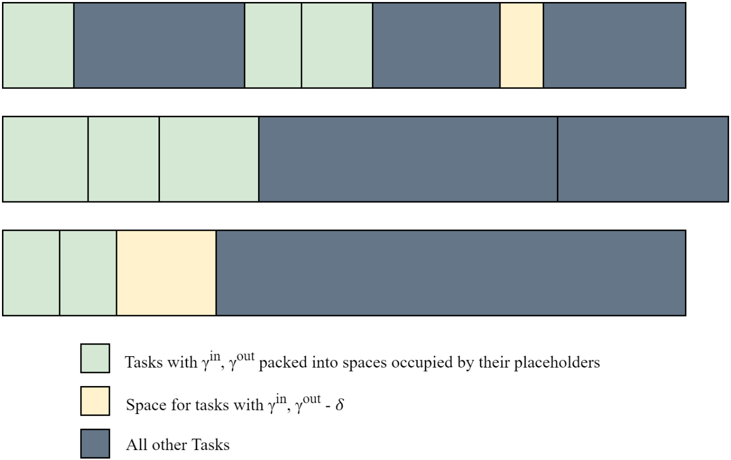
Schedule with placeholders



Simplification 3: Small task placeholders – reverse step

Schedule with small tasks recovered

Using a NextFit algorithm per communication size $(\gamma^{in}, \gamma^{out})$

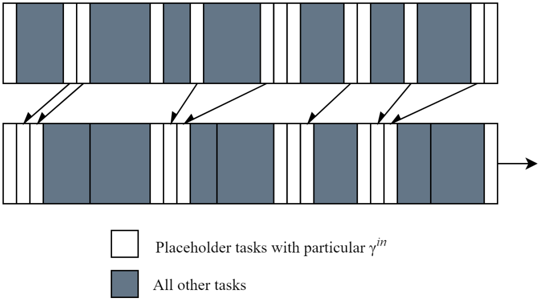


Simplification 4: Placeholder Arrangement

Placeholders created in previous step forced into groups

- placeholders $\forall \gamma_i$, $J^{\gamma^{in}}_{small}$, forced into groups of $\frac{1}{\epsilon}$
- their total processing time $\epsilon^2 T$
- remainder in *stub* group of $\leq \frac{1}{\epsilon}$ placeholders

$\Rightarrow$ new instance I_4

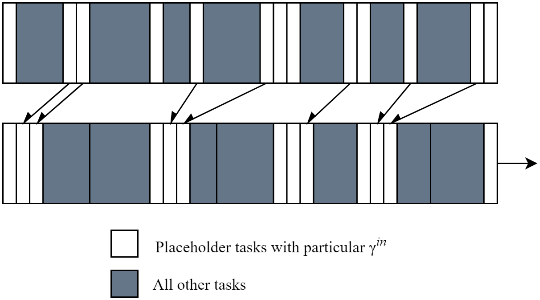


Simplification 4: Placeholder Arrangement

Placeholders created in previous step forced into groups

- placeholders $\forall \gamma_i$, $J_{small}^{\gamma^{in}}$, forced into groups of $\frac{1}{\epsilon}$
- their total processing time $\epsilon^2 T$
- remainder in *stub* group of $\leq \frac{1}{\epsilon}$ placeholders

$\Rightarrow$ new instance I_4



Lemma (Placeholder grouping)

Let I_4 be the instance of the problem with this restriction to the solution, and T_4 be the minimum schedule makespan for I_4.

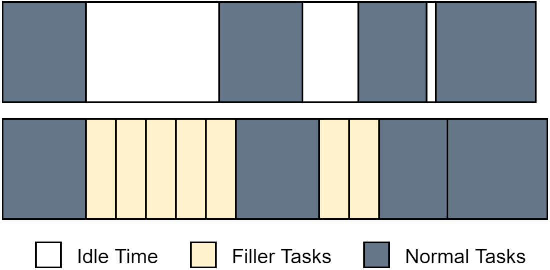
$$T_4 - \epsilon T \leq T_3 \leq T_4$$

Simplification 5: Gap sizes

Idle time gaps scaled to multiples of $\epsilon^2 T$

- same size as small task placeholders
- use filler tasks of size to $\epsilon^2 T$ to fill idle gaps

$\Rightarrow$ new instance I_5

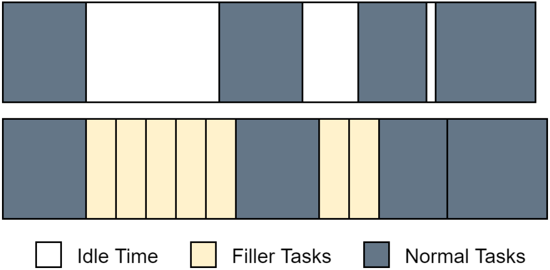


Simplification 5: Gap sizes

Idle time gaps scaled to multiples of $\epsilon^2 T$

- same size as small task placeholders
- use filler tasks of size to $\epsilon^2 T$ to fill idle gaps

$\Rightarrow$ new instance I_5



Lemma (Idle gap scaling)

Let I_5 be the instance where task start times are restricted in this way, and let T_5 be the minimum schedule makespan for I_5.

$$T_5 - \epsilon^2 T \leq T_4 \leq T_5$$

Content

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5

Approach structure

Original Instance I

$\Downarrow$

Communication truncation, I_1

$\Downarrow$

Big task truncation, I_2

$\Downarrow$

Small task placeholders, I_3

$\Downarrow$

Placeholder arrangement, I_4

$\Downarrow$

Gaps resolution, I_5 $\Rightarrow$ Binary search over T

$\hookrightarrow$ Solve configuration ILP

Problem input

Instance I_5 is described by (for given T):

$N_{\rho,\gamma^{in},\gamma^{out}} \quad \forall(\rho, \gamma^{in}, \gamma^{out})$: number (multiplicity) of tasks of size $(\rho, \gamma^{in}, \gamma^{out})$

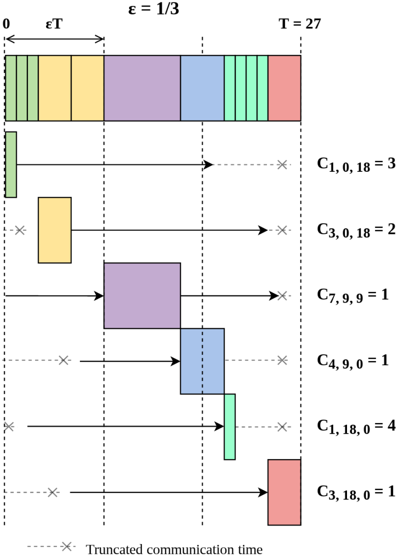
- total number of different sizes is $\mathcal{O}(\text{poly}(1/\epsilon))$
- multiplicity is also in $\mathcal{O}(\text{poly}(1/\epsilon))$

Finding valid schedule for I_5 for given T:

Use configuration ILP

- where configuration is order of task sizes and their multiplicity

Configurations



Configuration C:

- set of tasks (multiset of task sizes $(\rho, \gamma^{in}, \gamma^{out})$)
- with an assumed order, with arrival time of last edge $< T$
- to be scheduled onto a single processor

$\boldsymbol{C}$: set of all needed (possible) configurations

Constraints

Decision variables for ILP:

$x_C \; \forall C \in \boldsymbol{C}$ select the number of each configuration used

Constraints

Decision variables for ILP:

x_C $\forall C \in \boldsymbol{C}$ select the number of each configuration used

ILP Constraints:

Configurations = #processors:

$$\sum_{C \in \boldsymbol{C}} x_C = M$$

Constraints

Decision variables for ILP:
$x_C\ \forall C \in \boldsymbol{C}$ select the number of each configuration used

ILP Constraints:
Configurations = #processors:

$$\sum_{C \in \boldsymbol{C}} x_C = M$$

All tasks scheduled:

$$\sum_{c \in \boldsymbol{C}} x_C C_{\rho,\gamma^{in},\gamma^{out}} \geq N_{\rho,\gamma^{in},\gamma^{out}} \quad \forall(\rho, \gamma^{in}, \gamma^{out})$$

Constraints

Decision variables for ILP:

x_C $\forall C \in \boldsymbol{C}$ select the number of each configuration used

ILP Constraints:

Configurations = #processors:

$$\sum_{C \in \boldsymbol{C}} x_C = M$$

All tasks scheduled:

$$\sum_{C \in \boldsymbol{C}} x_C C_{\rho,\gamma^{in},\gamma^{out}} + S_+ - S \geq N_{\rho,\gamma^{in},\gamma^{out}} \quad \forall(\rho, \gamma^{in}, \gamma^{out})$$

Constraints

Decision variables for ILP:

x_C $\forall C \in \boldsymbol{C}$ select the number of each configuration used

ILP Constraints:

Configurations = #processors:

$$\sum_{C \in \boldsymbol{C}} x_C = M$$

All tasks scheduled:

$$\sum_{C \in \boldsymbol{C}} x_C \, C_{\rho,\gamma^{in},\gamma^{out}} + S^{>in}_{\rho,\gamma^{in}+\Delta,\gamma^{out}} + S^{>out}_{\rho,\gamma^{in},\gamma^{out}+\Delta}$$

$$-S^{>in}_{\rho,\gamma^{in},\gamma^{out}} - S^{>out}_{\rho,\gamma^{in},\gamma^{out}} \geq N_{\rho,\gamma}$$

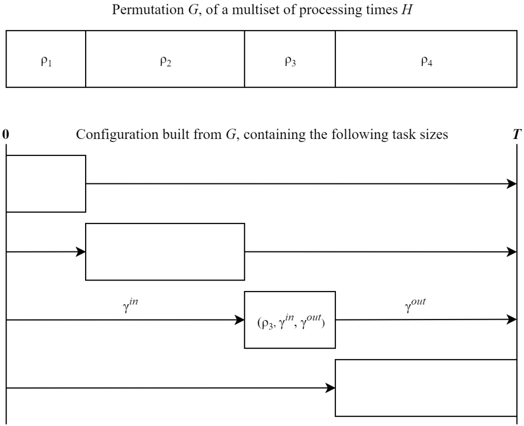
$$\forall(\rho, \gamma^{in}, \gamma^{out})$$

Obtaining a configuration

Straight forward:
One C for every valid permutation of task sizes/multiplicities

Better:
Only considering permutations of **execution cost** ρ & multiplicities

Possible because:
shorter communications can be put in slot for larger communications



Obtaining a configuration

Set of configurations $\boldsymbol{C}$:

- only valid C, i.e. last edge out $< T$
- only maximal C, i.e. no $C \in \boldsymbol{C}$ is subset of other $C' \in \boldsymbol{C}$
- only dominating comm sizes $(\gamma^{in}, \gamma^{out})$

Obtaining a configuration

Set of configurations $\boldsymbol{C}$:

- only valid C, i.e. last edge out $< T$
- only maximal C, i.e. no $C \in \boldsymbol{C}$ is subset of other $C' \in \boldsymbol{C}$
- only dominating comm sizes $(\gamma^{in}, \gamma^{out})$

Lemma (C is complete)

There exists $C \in \boldsymbol{C}$ to represent any possible schedule on one processor

Obtaining a configuration

Set of configurations $\boldsymbol{C}$:

- only valid C, i.e. last edge out $< T$
- only maximal C, i.e. no $C \in \boldsymbol{C}$ is subset of other $C' \in \boldsymbol{C}$
- only dominating comm sizes $(\gamma^{in}, \gamma^{out})$

Lemma (C is complete)

There exists $C \in \boldsymbol{C}$ to represent any possible schedule on one processor

Lemma (Number of configurations)

The number of configurations $|\boldsymbol{C}| = 2^{\mathcal{O}(1/\epsilon^2 \log 1/\epsilon)}$

Content

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T $\Rightarrow$ Opt. solution for I_5
$\hookrightarrow$ Solve configuration ILP

Approach structure

Original Instance I

$\Downarrow$

Communication truncation, I_1

$\Downarrow$

Big task truncation, I_2

$\Downarrow$

Small task placeholders, I_3

$\Downarrow$

Placeholder arrangement, I_4

$\Downarrow$

Gaps resolution, I_5 $\Rightarrow$ Binary search over T
$\hookrightarrow$ Solve configuration ILP $\Rightarrow$ Opt. solution for I_5

$\Uparrow$

Fill placeholders with small tasks

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T
$\hookrightarrow$ Solve configuration ILP $\Rightarrow$ Opt. solution for I_5
$\Uparrow$
Fill placeholders with small tasks
$\Uparrow$
Recover orig. processing times

Approach structure

Original Instance I
$\Downarrow$
Communication truncation, I_1
$\Downarrow$
Big task truncation, I_2
$\Downarrow$
Small task placeholders, I_3
$\Downarrow$
Placeholder arrangement, I_4
$\Downarrow$
Gaps resolution, I_5 $\Rightarrow$ Binary search over T
$\hookrightarrow$ Solve configuration ILP $\Rightarrow$ Opt. solution for I_5
$\Uparrow$
Fill placeholders with small tasks
$\Uparrow$
Recover orig. processing times
$\Uparrow$
Recover orig. comm. times

Approach structure

Original Instance I				Solution for I, $(1+\epsilon')OPT$
$\Downarrow$				$\Uparrow$
Communication truncation, I_1				Recover orig. comm. times
$\Downarrow$				$\Uparrow$
Big task truncation, I_2				Recover orig. processing times
$\Downarrow$				$\Uparrow$
Small task placeholders, I_3				Fill placeholders with small tasks
$\Downarrow$				
Placeholder arrangement, I_4				$\Uparrow$
$\Downarrow$				
Gaps resolution, I_5	$\Rightarrow$	Binary search over T $\hookrightarrow$ Solve configuration ILP	$\Rightarrow$	Opt. solution for I_5

EPTAS schedule length

Combining the inequalities from 5 simplifcation lemmas:

$$(1 - 2\epsilon - \epsilon^2)T \leq OPT \leq (1 + 5\epsilon)T$$

Final schedule obtained has makespan:

$$\frac{(1 + 5\epsilon)}{(1 - 2\epsilon - \epsilon^2)}\, OPT$$

To obtain accuracy parameter ϵ' set

$$1 + \epsilon' = \frac{(1 + 5\epsilon)}{(1 - 2\epsilon - \epsilon^2)}$$

EPTAS runtime

Let $N = |J|$ be number of tasks in input of instance I.

- Simplifying instance & obtaining inputs for ILP takes $\mathcal{O}(N)$ time
- Using binary search to find T, EPTAS running is $(\mathrm{ILP} + \mathcal{O}(N)) \cdot \log(N)$

With results from [1] runtime of ILP is:

$$2^{\mathcal{O}(1/\epsilon^3 \log^2 1/\epsilon)} \mathcal{O}(\log N)$$

[1] K. Jansen, L. Rohwedder. "On integer programming and convolution". arXiv:1803.04744 (2018)

Conclusion

Theorem

The EPTAS finds a schedule with makespan no longer than:

$$\frac{(1+5\epsilon)}{(1-2\epsilon-\epsilon^2)}\,OPT$$

in time:

$$2^{\mathcal{O}(1/\epsilon^3 \log^2 1/\epsilon)}\mathcal{O}(\log^2 N) + \mathcal{O}(N \log N)$$

Future work:

- Versions for fork graphs and for join graphs
- Using approach for release time and deadline scheduling