Modeling parallel real-time tasks by di-graph

RTNS'2016

H. ZAHAF[1,2], AH. BENYAMINA[1], R.OLEJNIK[2], G. LIPARI[2], P. BOULET[2]

21 octobre 2016

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Université de Lille
cnrs
centralelille

Plan

Plan

Context & Motivations

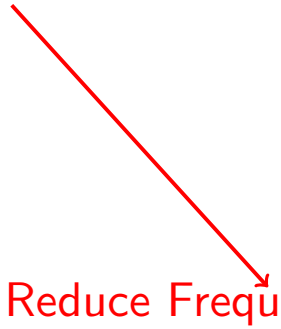
Modeling tasks

Context & Motivations

Modeling real-time tasks

Context & Motivations

Modeling parallel real-time tasks



Reduce Frequency and Energy consumption

Context & Motivations

Parallel task models are not expressive enough for dynamic tasks

Modeling parallel real-time tasks by digraphs

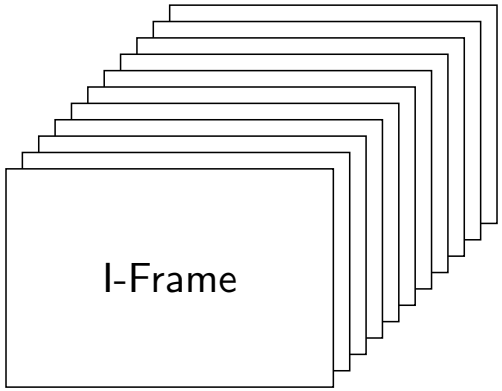
Reduce Frequency and Energy consumption

MPEG Encoding

- Compress the video to reduce its size
- Describing the differences between frames

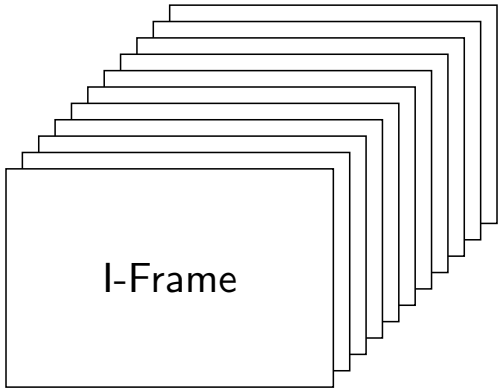
MPEG Encoding

- Compress the video to reduce its size
- Describing the differences between frames



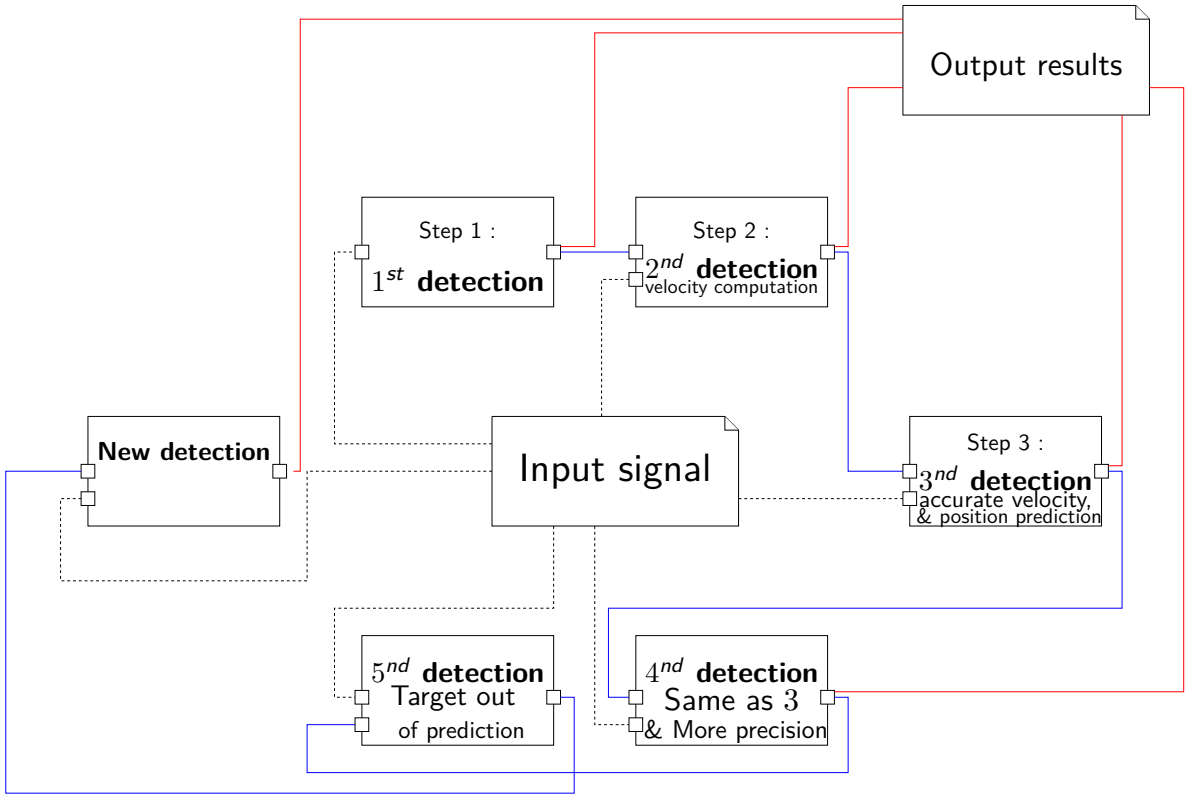
MPEG Encoding

- Compress the video to reduce its size
- Describing the differences between frames



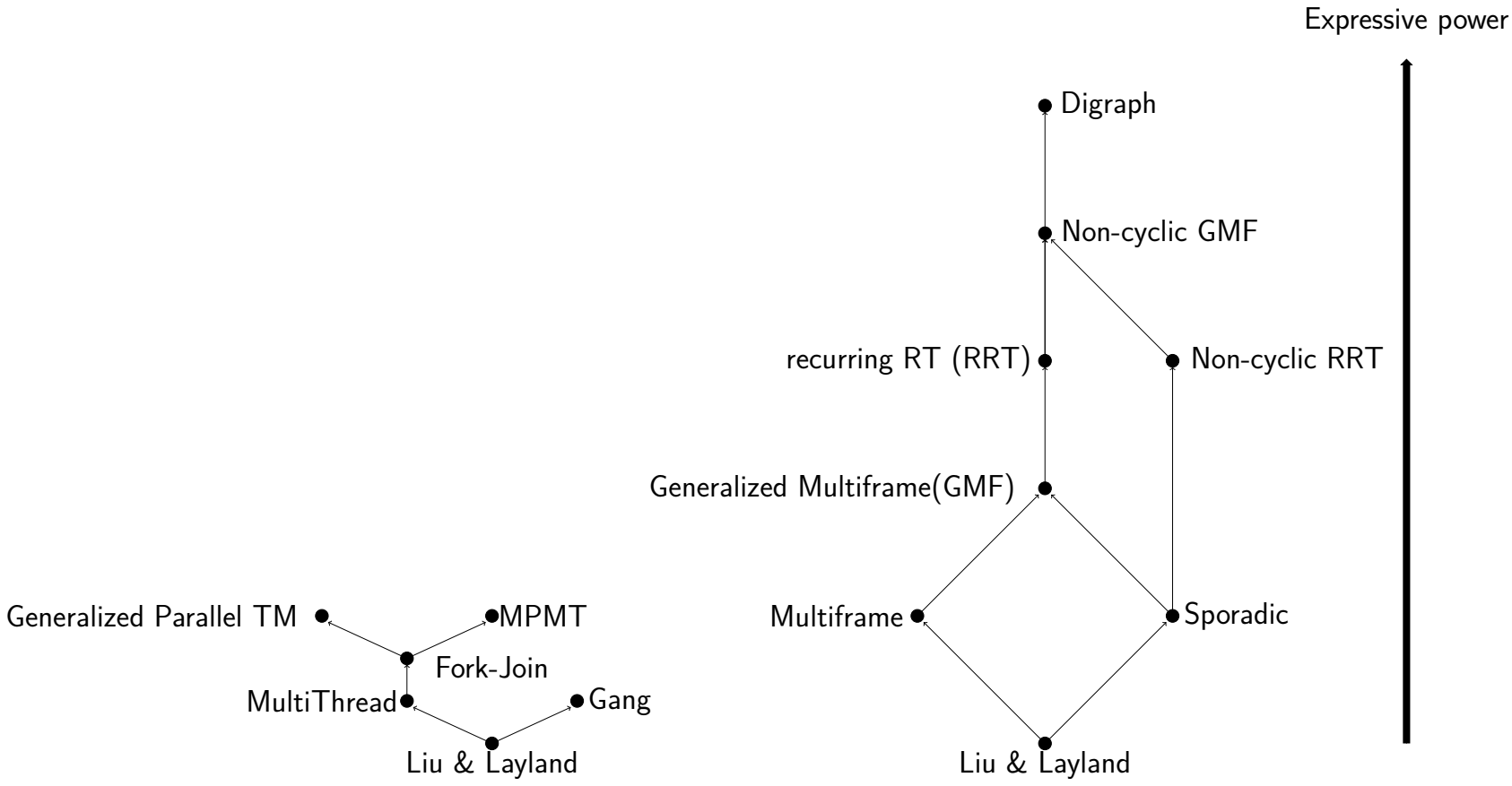
- The typical sequence of Frames is : **IBBPBBPBBPBBI**
- But this sequence may change dynamically
- The processing of each frame type is parallelizible.

Radar MTI

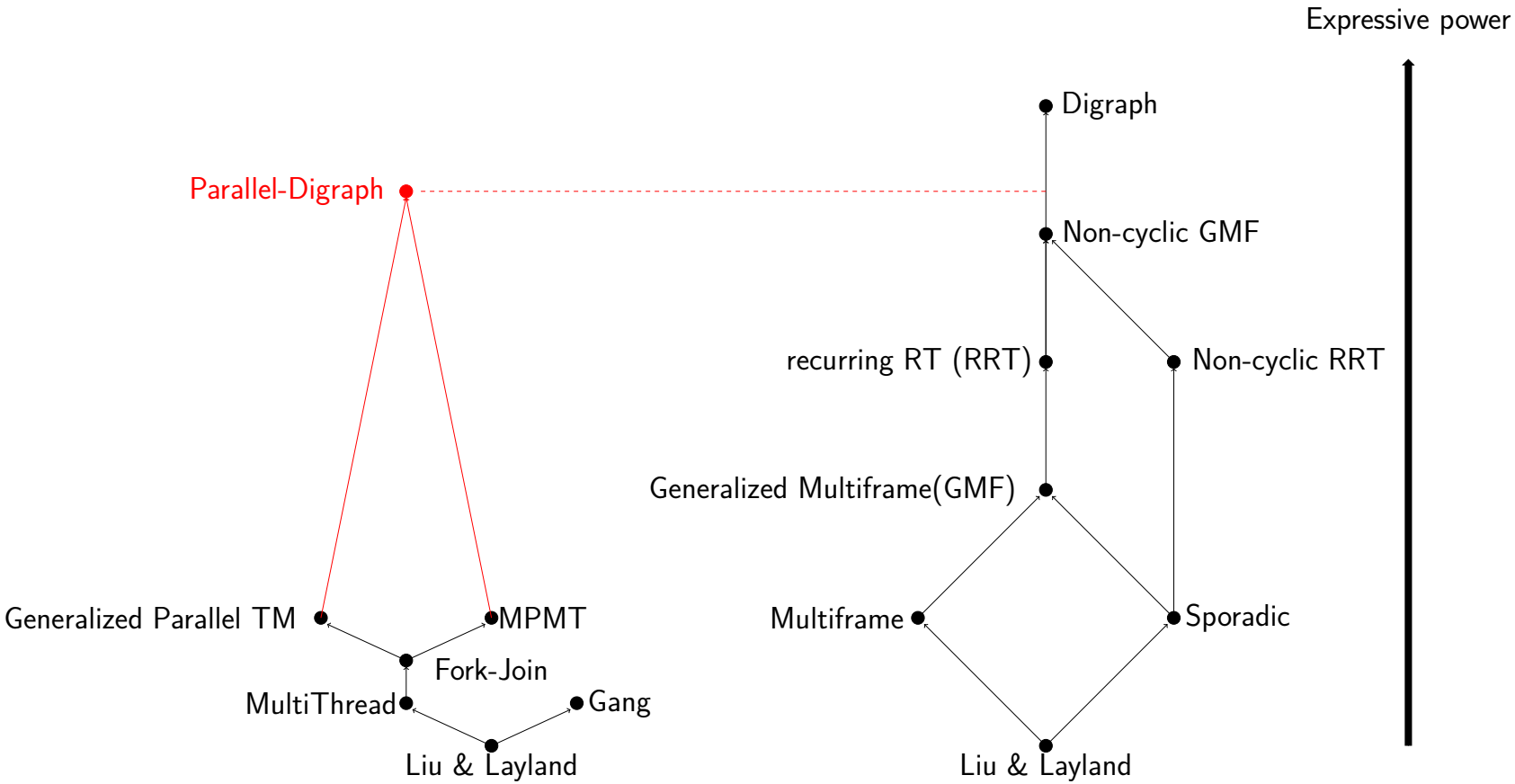


- The processing depends on the value of data
- The processing is parallelizible

Task Models


Expressive power
Digraph
Non-cyclic GMF
recurring RT (RRT)
Non-cyclic RRT
Generalized Multiframe(GMF)
Multiframe
Sporadic
Liu & Layland
Generalized Parallel TM
MPMT
Fork-Join
MultiThread
Gang
Liu & Layland

Task Models


Expressive power
Digraph
Parallel-Digraph
Non-cyclic GMF
recurring RT (RRT)
Non-cyclic RRT
Generalized Multiframe(GMF)
Generalized Parallel TM
MPMT
Multiframe
Sporadic
Fork-Join
MultiThread
Gang
Liu & Layland
Liu & Layland

Plan

A parallel task in di-graph

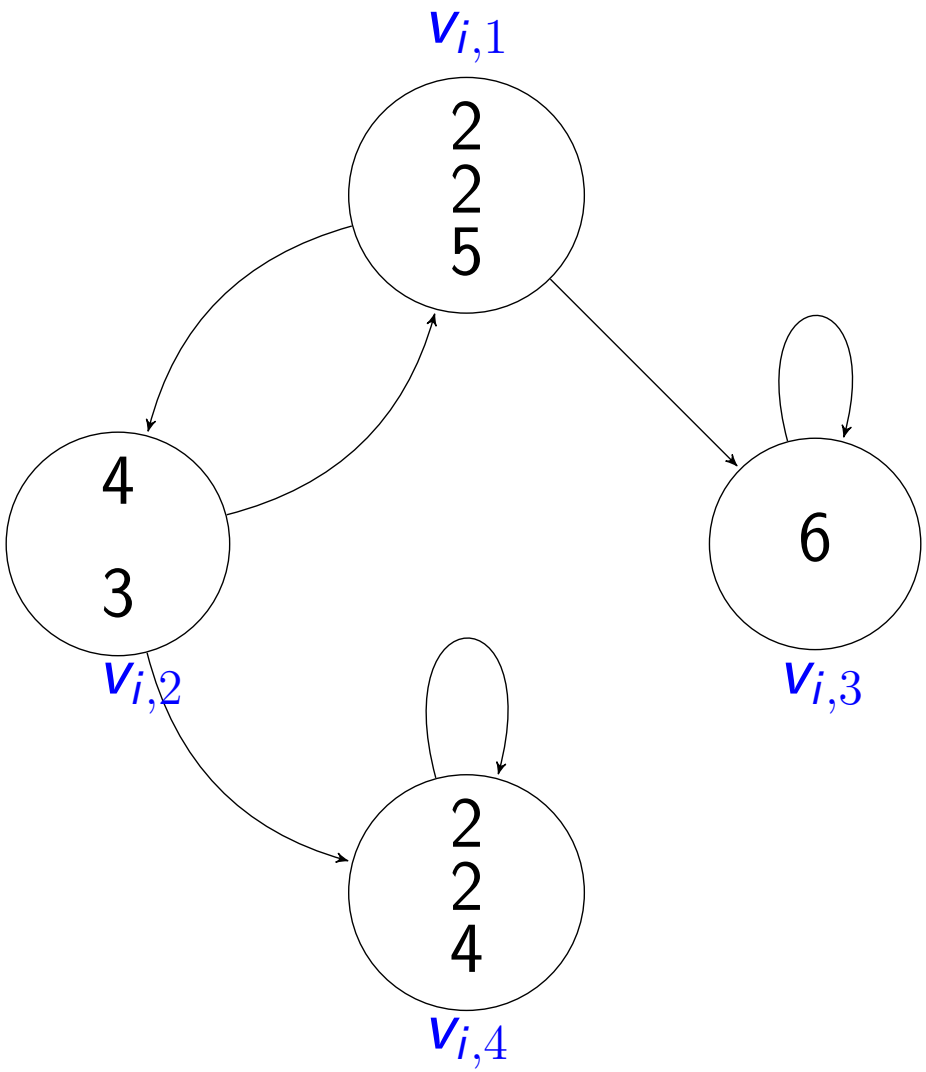


FIGURE : τ_i digraph, $\mathsf{D}_i = 10, \mathsf{T}_i = 12$

Tasks are :

- Sporadic (constrained deadline)
- Modeled by an *automaton*

Each vertex

- is a job
- consists of one or more threads

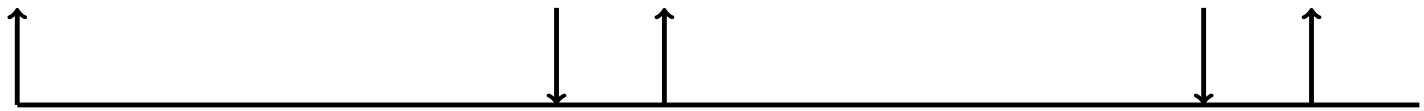
Transition between vertices

- is non-deterministic
- represents the precedence order between two jobs

Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

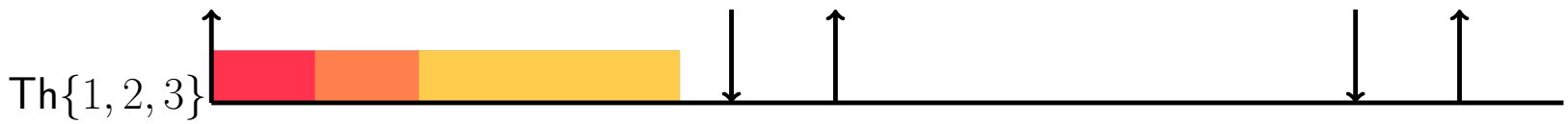
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 1$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

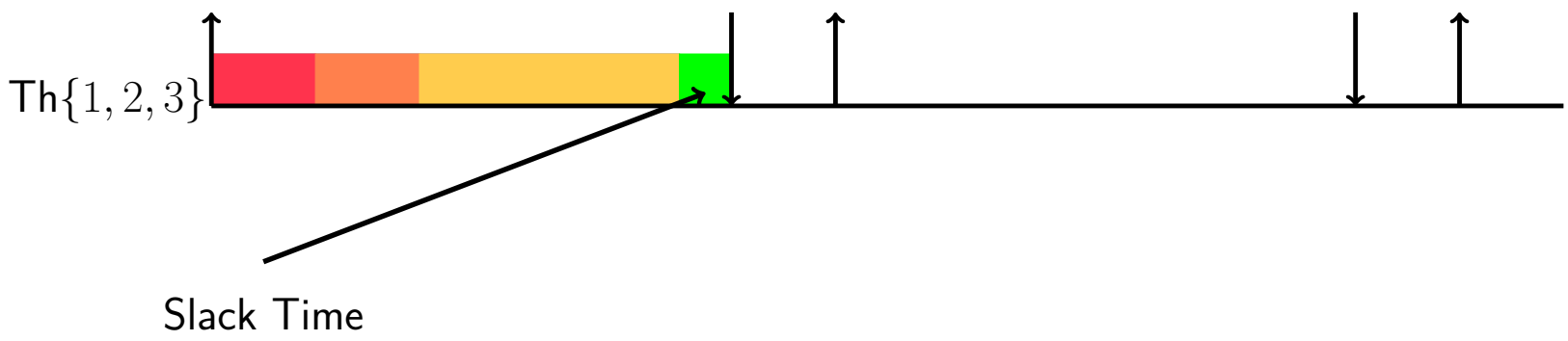
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 1$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

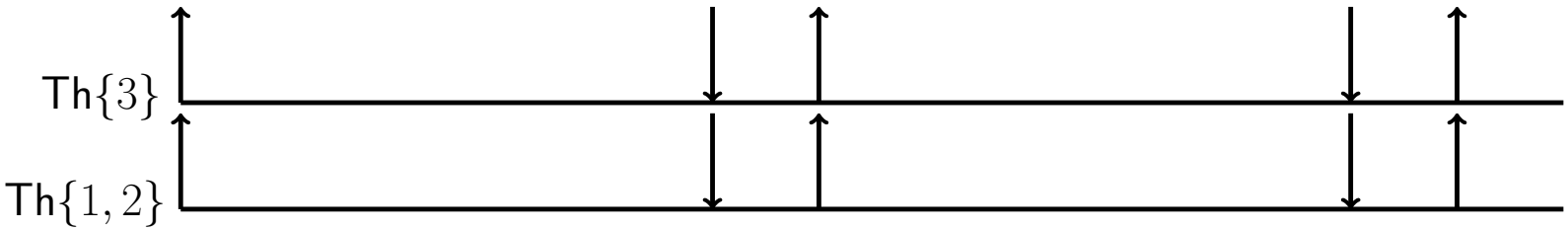
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 1$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

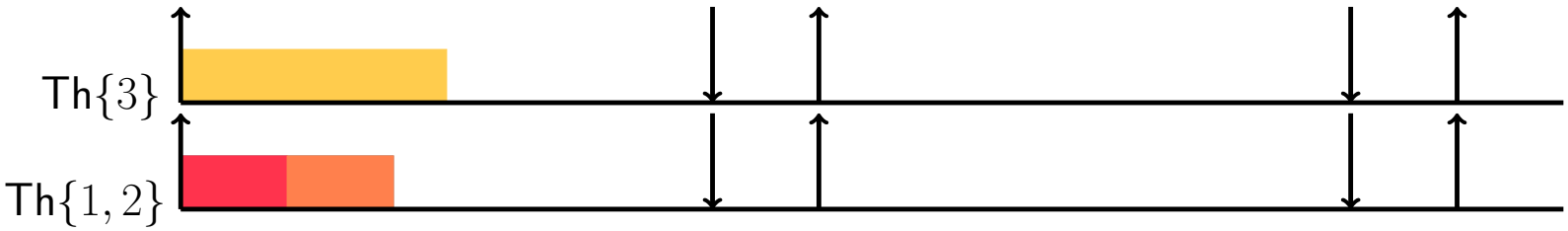
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 1$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

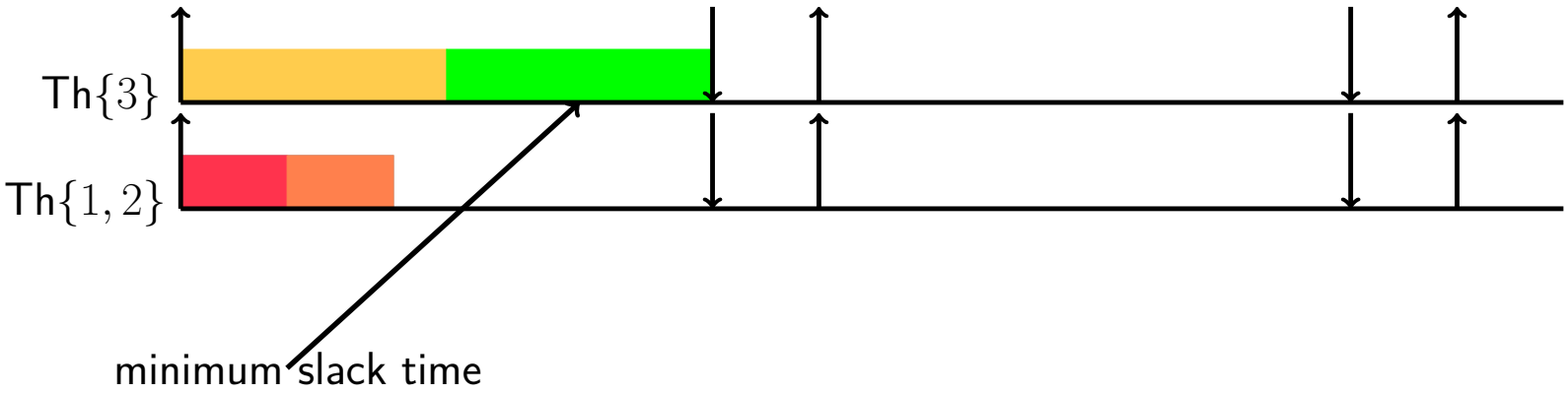
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 1$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

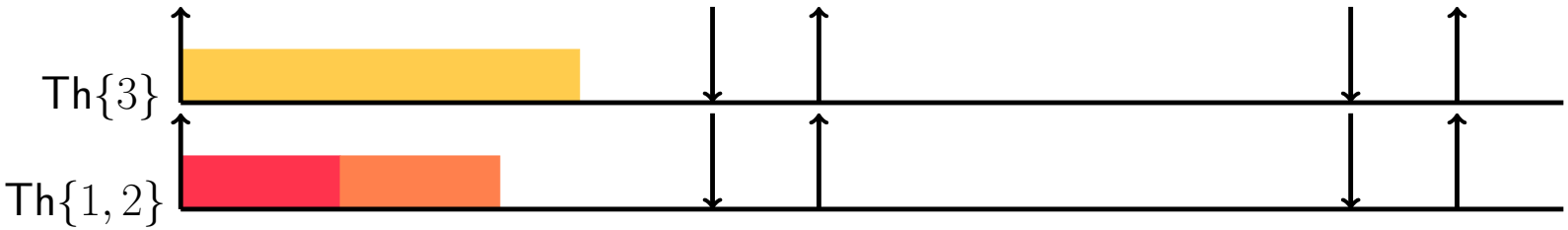
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 1$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

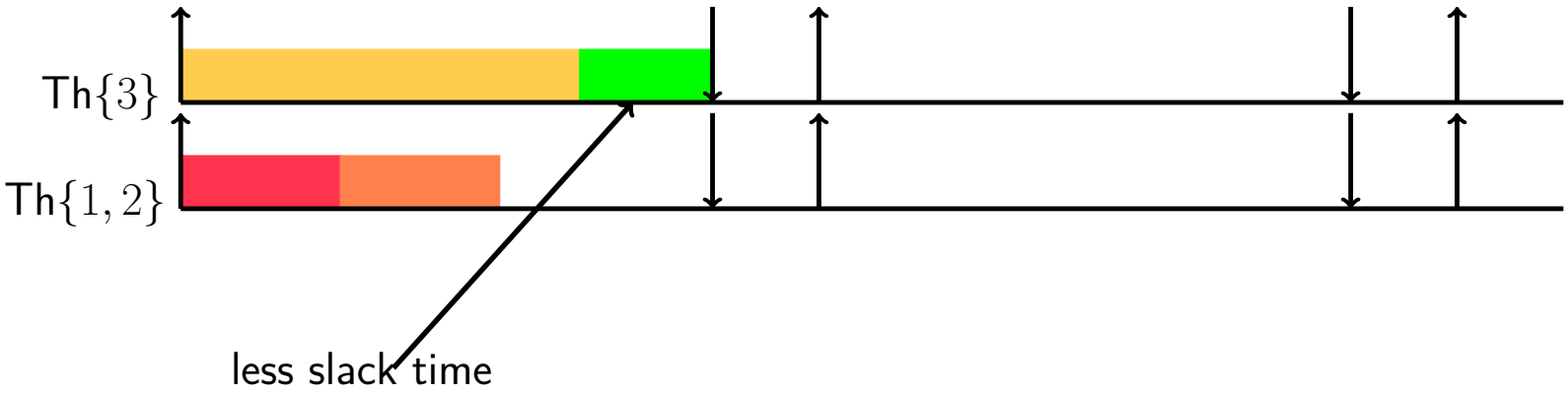
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 0.75$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

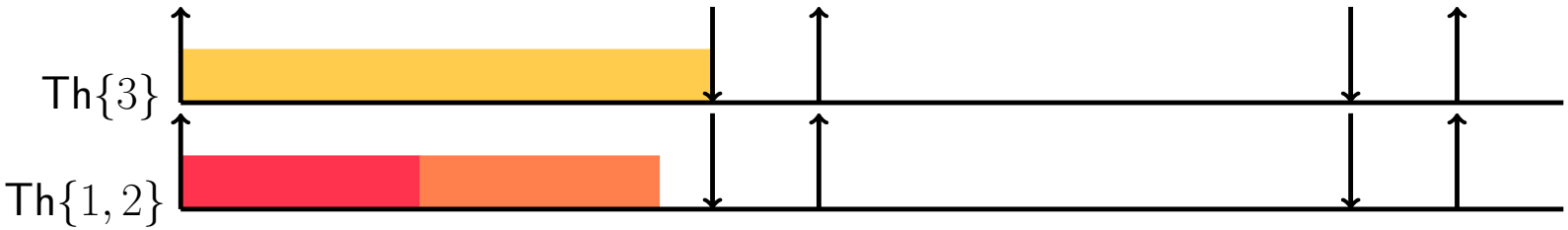
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 0.75$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

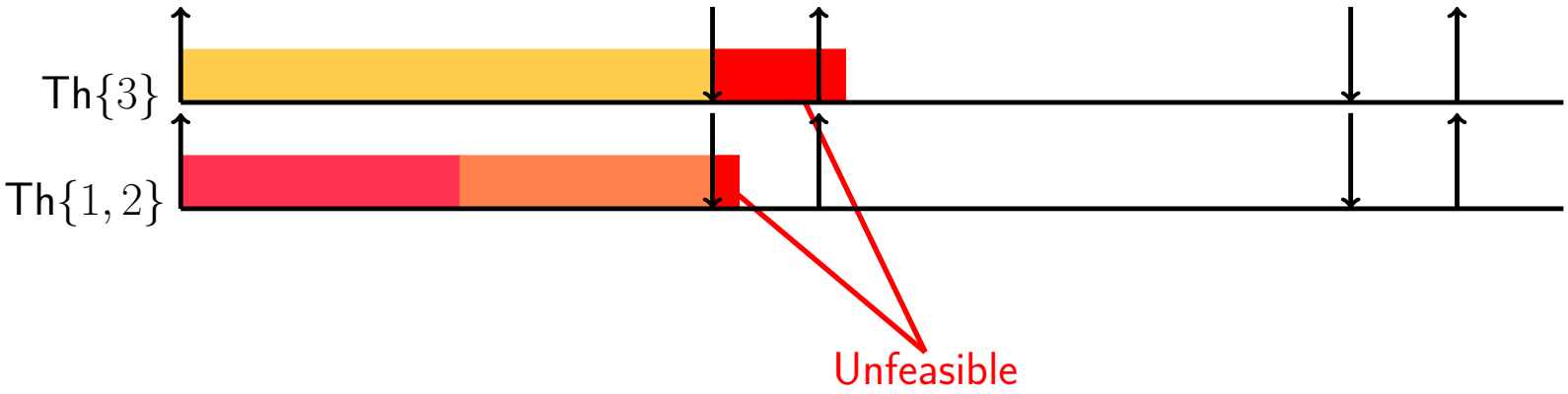
- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 0.50$



Parallelization and Frequency selection

Let consider vertex $v_{i,1}$:

- The vertex has three threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$,with an execution time of 2,2,5 on a core operating at speed s $= 0.35$



Problem formulation

We have :

Problem formulation

We have :

- A set of parallel digraph tasks

Problem formulation

We have :

- A set of parallel digraph tasks
- A set of identical cores with the same variable frequency

Problem formulation

We have :

- A set of parallel digraph tasks
- A set of identical cores with the same variable frequency

Objective

- Allocate the digraph tasks to the set of cores
- and select the minimal frequency

Constraints

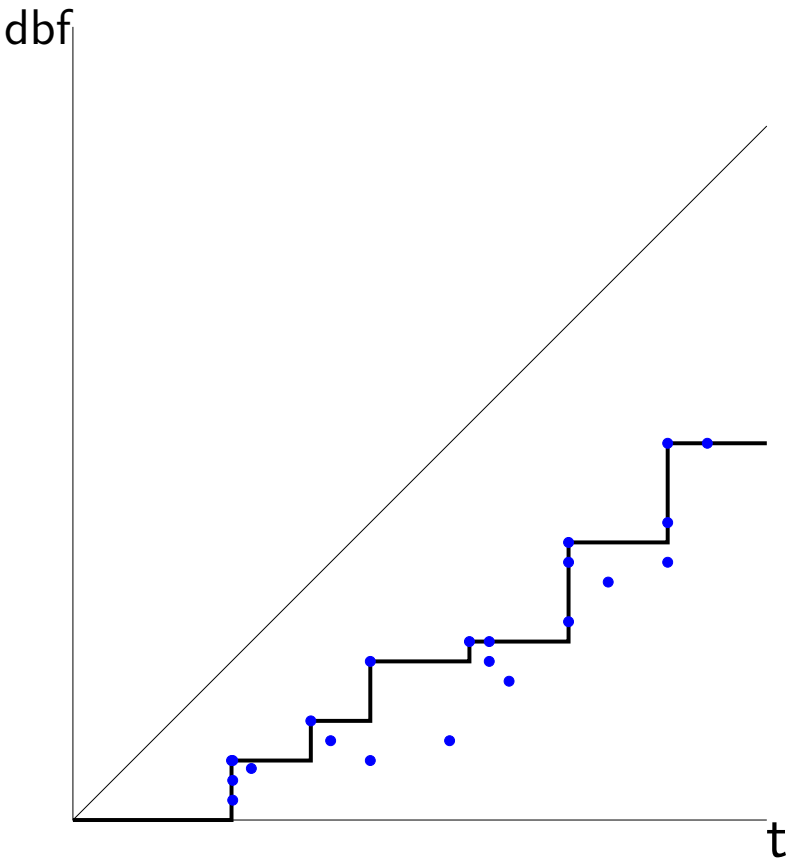
- All deadlines are met

Plan

Uniprocessor

Uniprocessor

Let $\mathcal{T}$ be a task set, and t an non-negative integer. $\mathcal{T}$ is feasible if and only if (Stigge et al.) : $\forall t, \text{dbf}(\mathcal{T}, t) \leq t$



To compute $\text{dbf}(\mathcal{T}, t)$ we need :

1. Compute *df* for all paths of every task τ_i
2. For each path and for each task, take the maximum *df* at every t
3. Sum the results of 2.

Multiprocessor

Multiprocessor

- We consider partitioned-EDF

Multiprocessor

- We consider partitioned-EDF

1. We use one processor at time
2. We try to allocate a digraph task into a core

Multiprocessor

- We consider partitioned-EDF

1. We use one processor at time
2. We try to allocate a digraph task into a core
3. If 2 is not possible, we try to split the task into two parts :

Multiprocessor

- We consider partitioned-EDF

1. We use one processor at time
2. We try to allocate a digraph task into a core
3. If 2 is not possible, we try to split the task into two parts :
   - The first part is allocate on the current core

Multiprocessor

- We consider partitioned-EDF

1. We use one processor at time
2. We try to allocate a digraph task into a core
3. If 2 is not possible, we try to split the task into two parts :
   - The first part is allocate on the current core
   - The second part is put-back in the task list

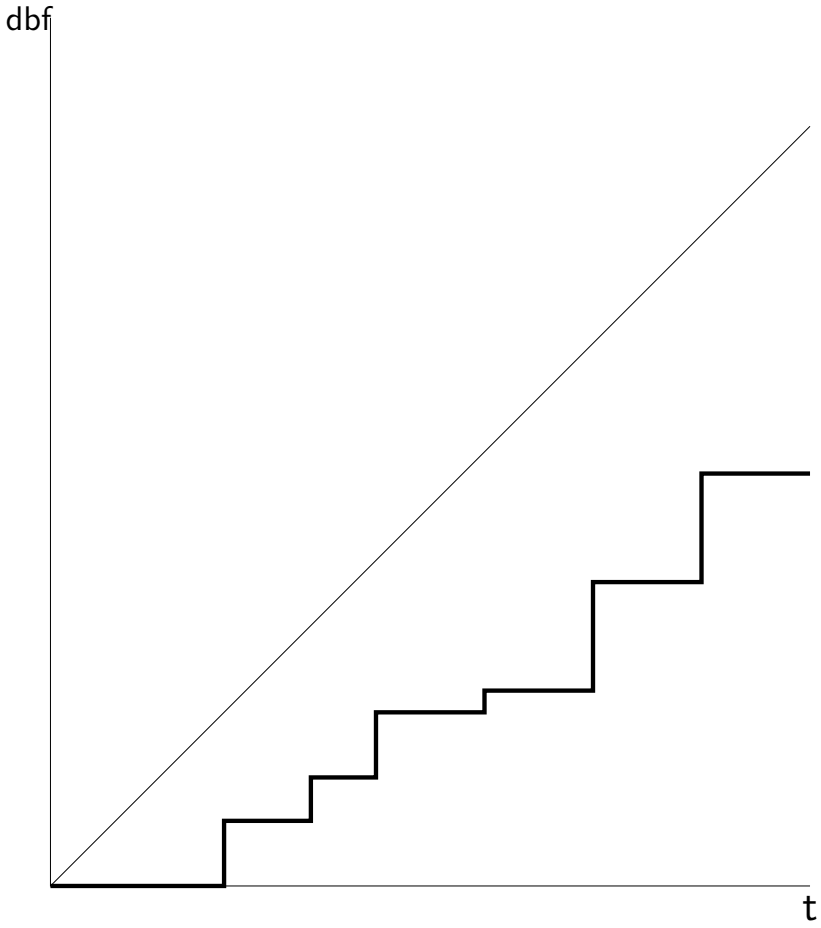
Multiprocessor

- We consider partitioned-EDF

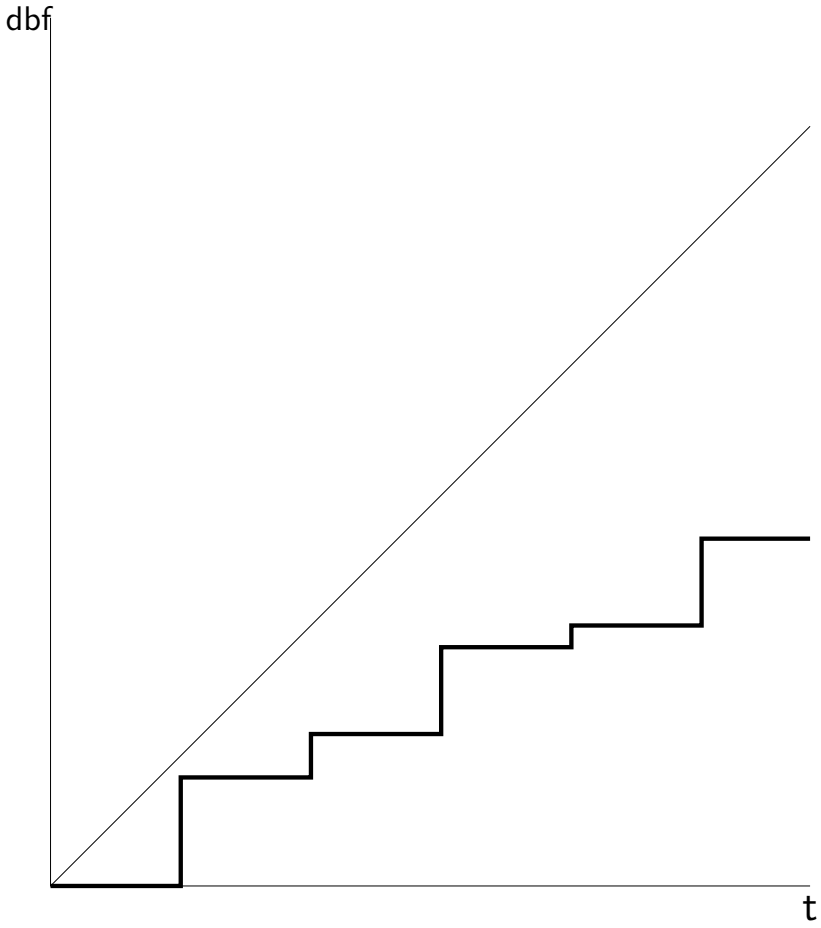
1. We use one processor at time
2. We try to allocate a digraph task into a core
3. If 2 is not possible, we try to split the task into two parts :
   - The first part is allocate on the current core
   - The second part is put-back in the task list

We need to check the combination of all tasks, all vertices, all threads, all cores, and all frequencies to obtain the exact solution

Partitioning scheme

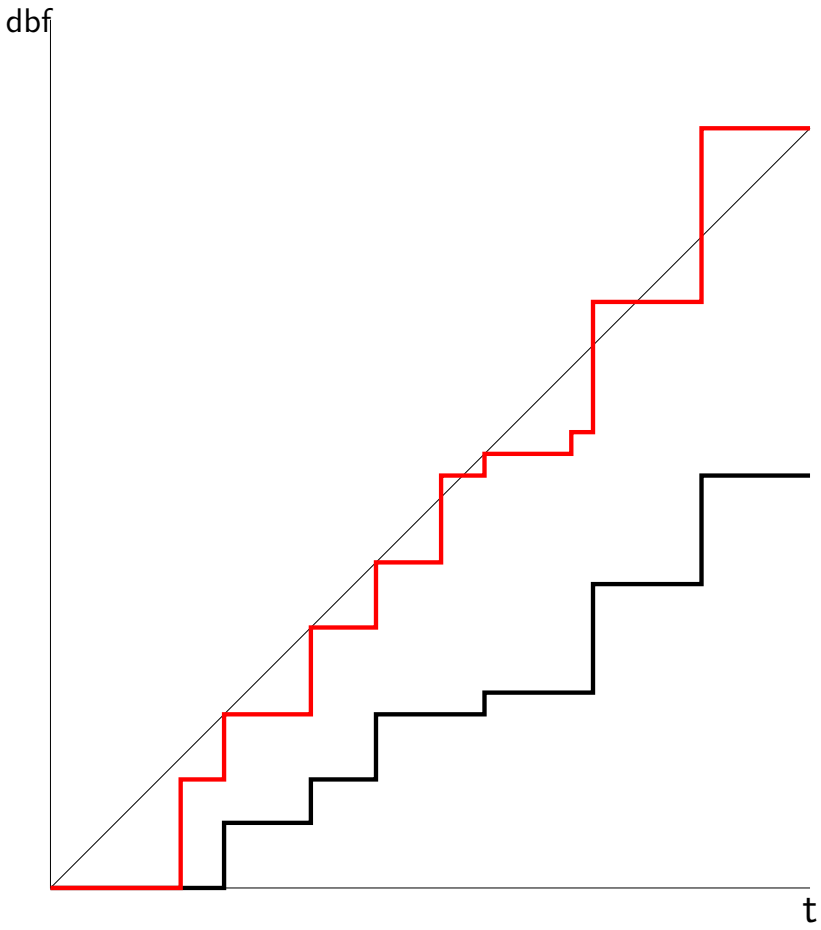


Task set $\mathcal{T}$

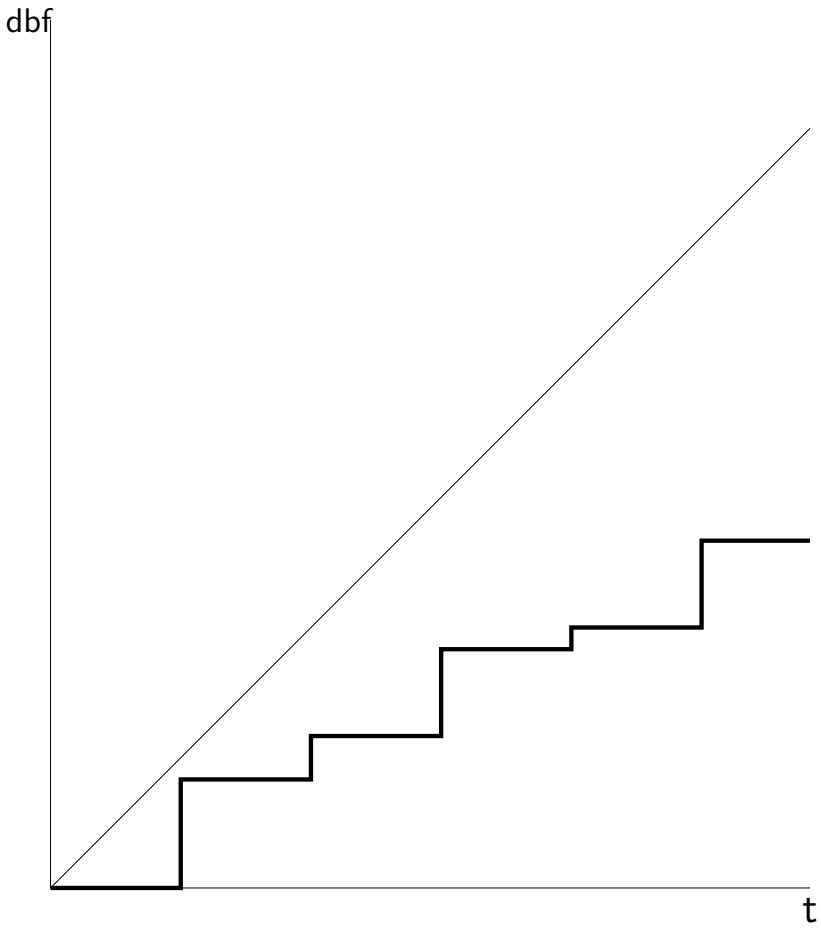


Task τ

Partitioning scheme

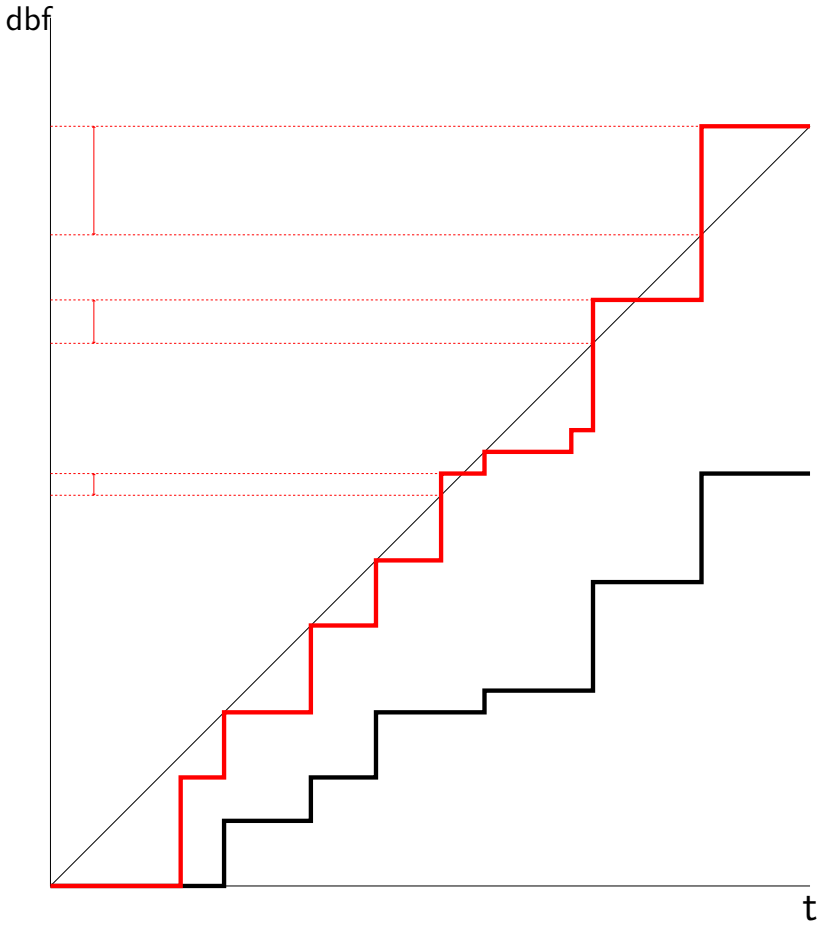


Task set $\mathcal{T}$

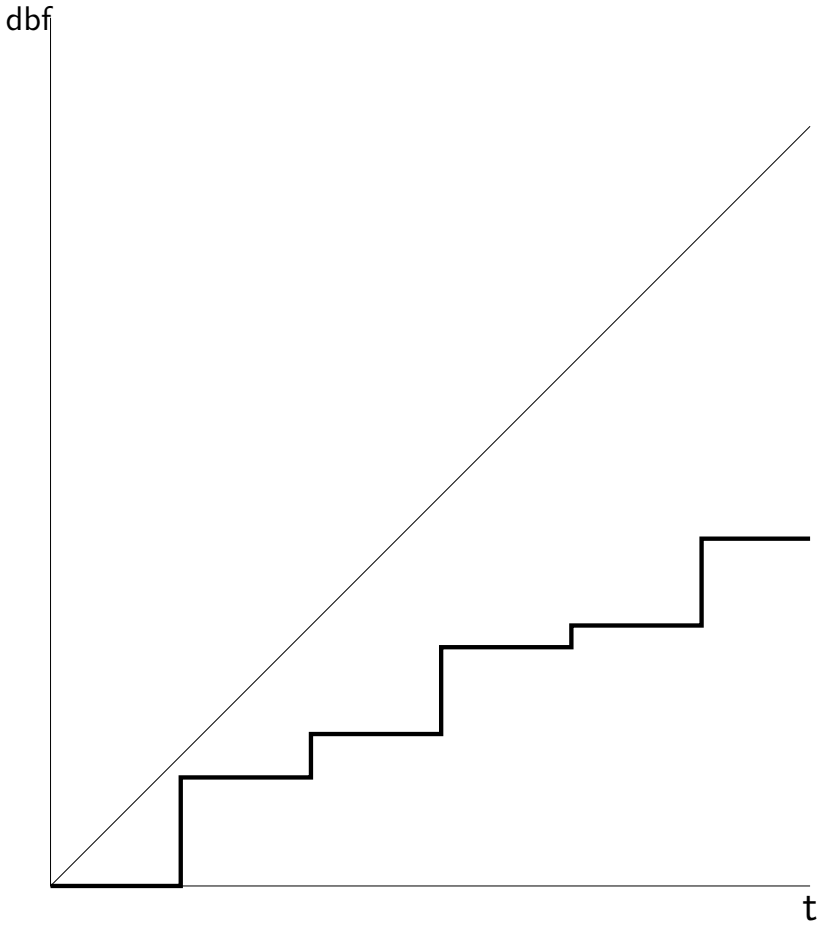


Task τ

Partitioning scheme

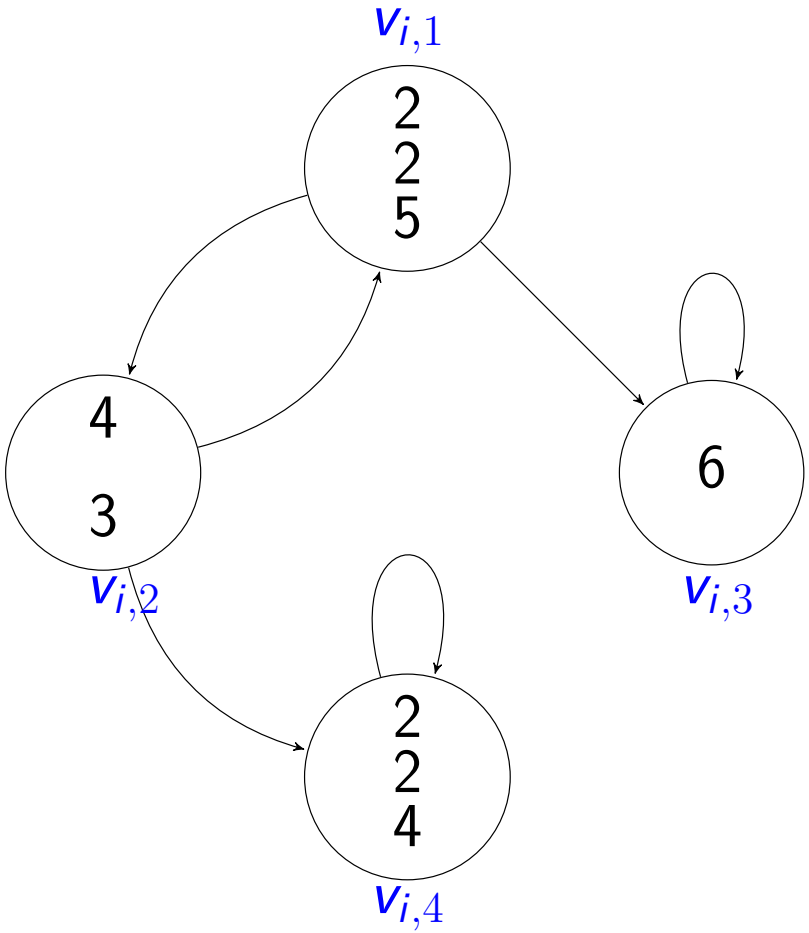


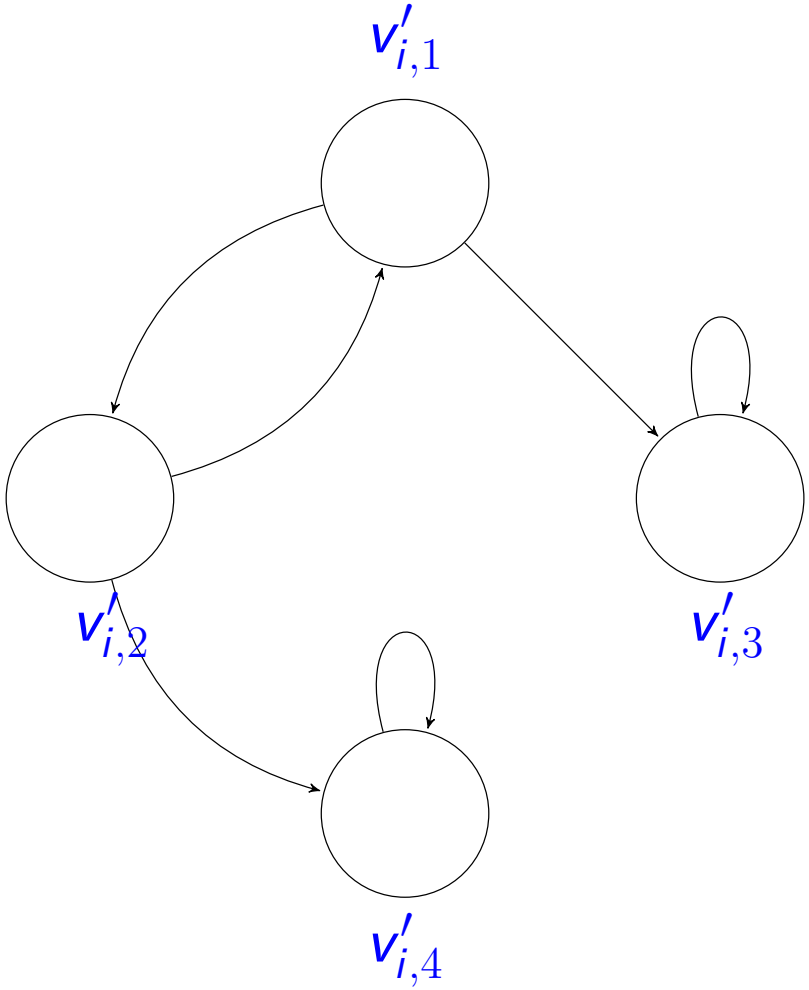
Task set $\mathcal{T}$



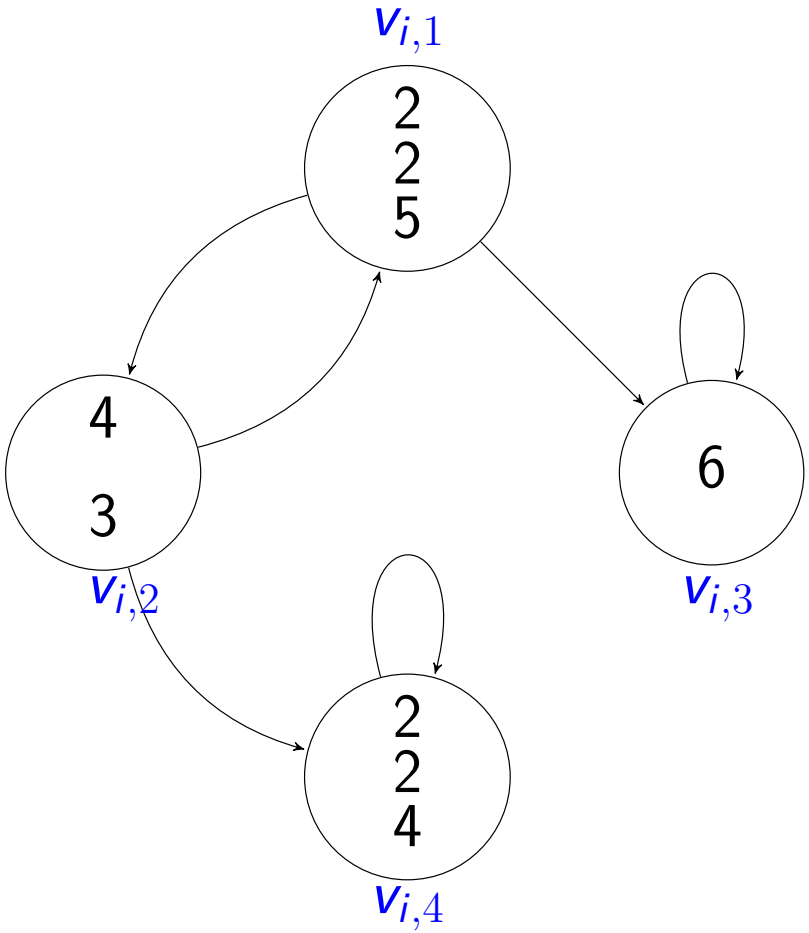
Task τ

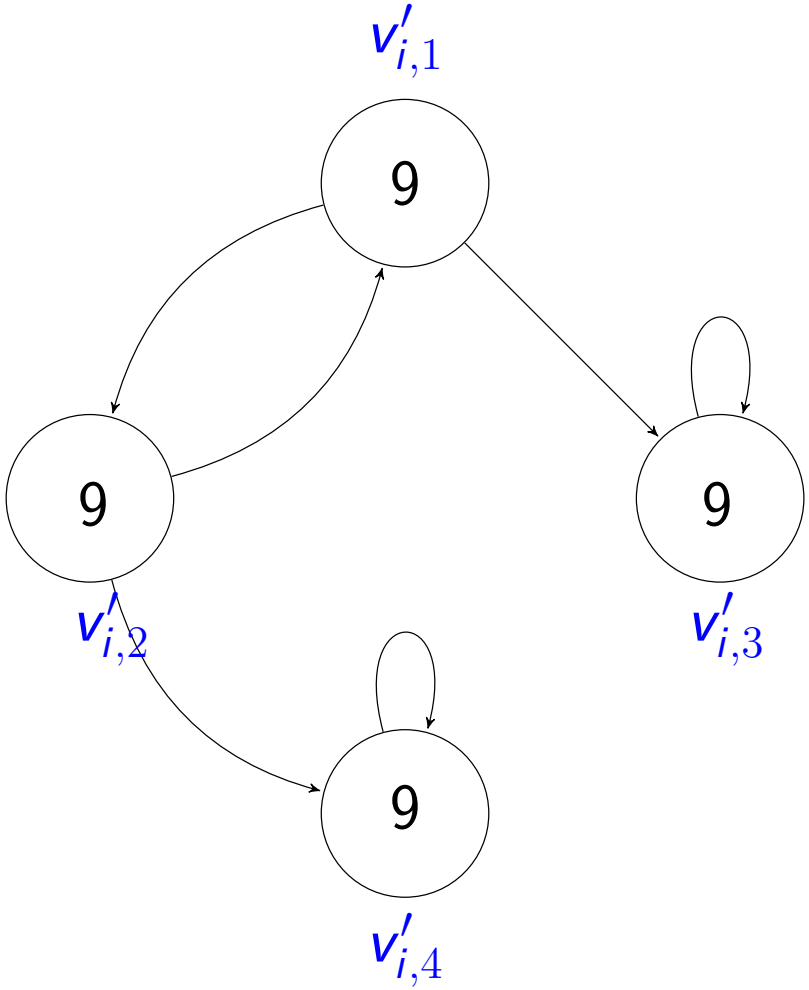
Equivalent single-thread task



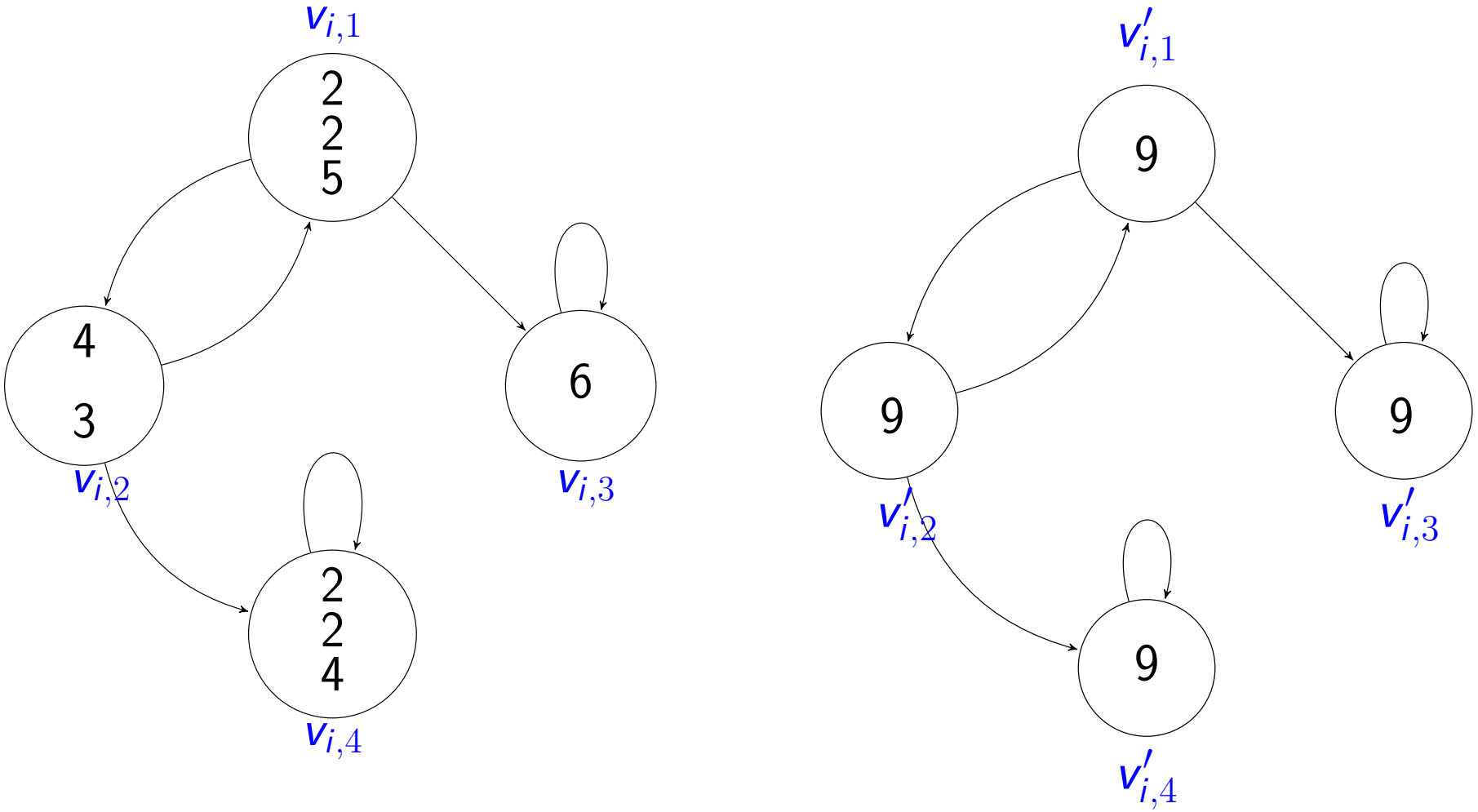


Equivalent single-thread task



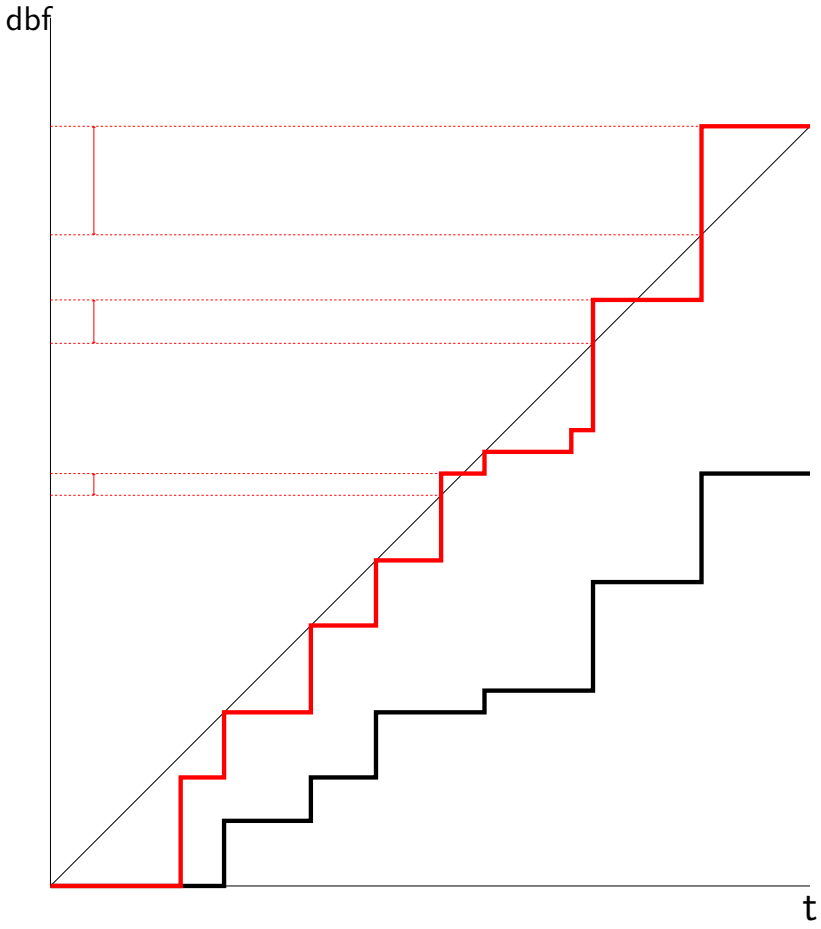


Equivalent single-thread task

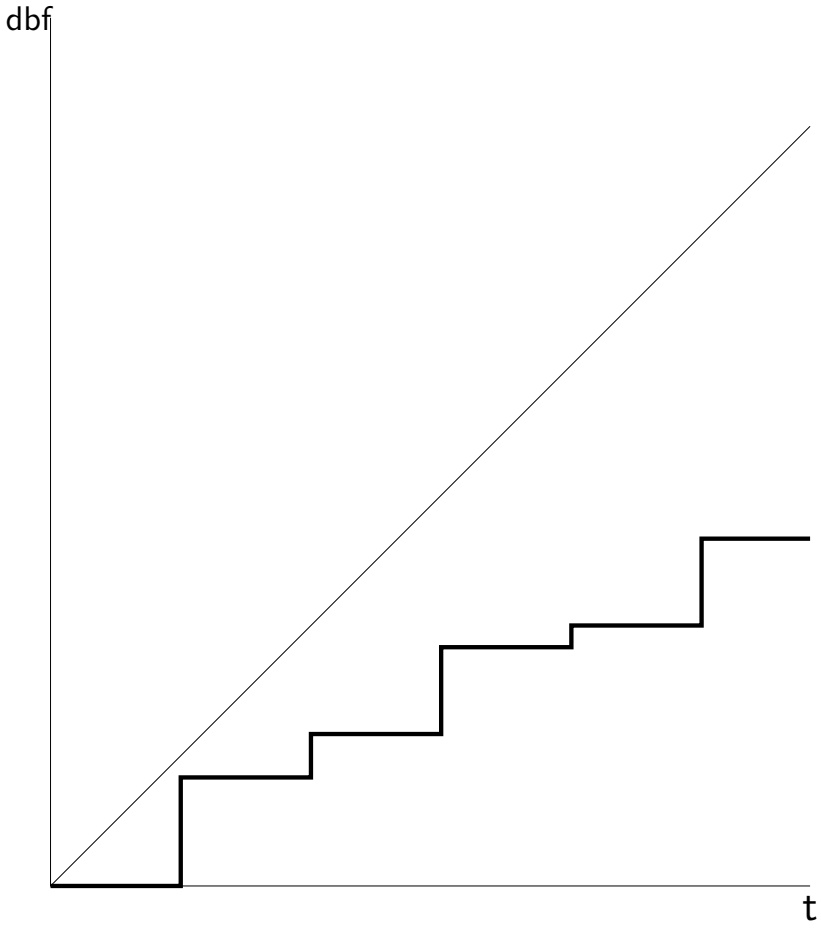


- This is a conversion from our model to the Liu and Layland model.

Partitioning scheme

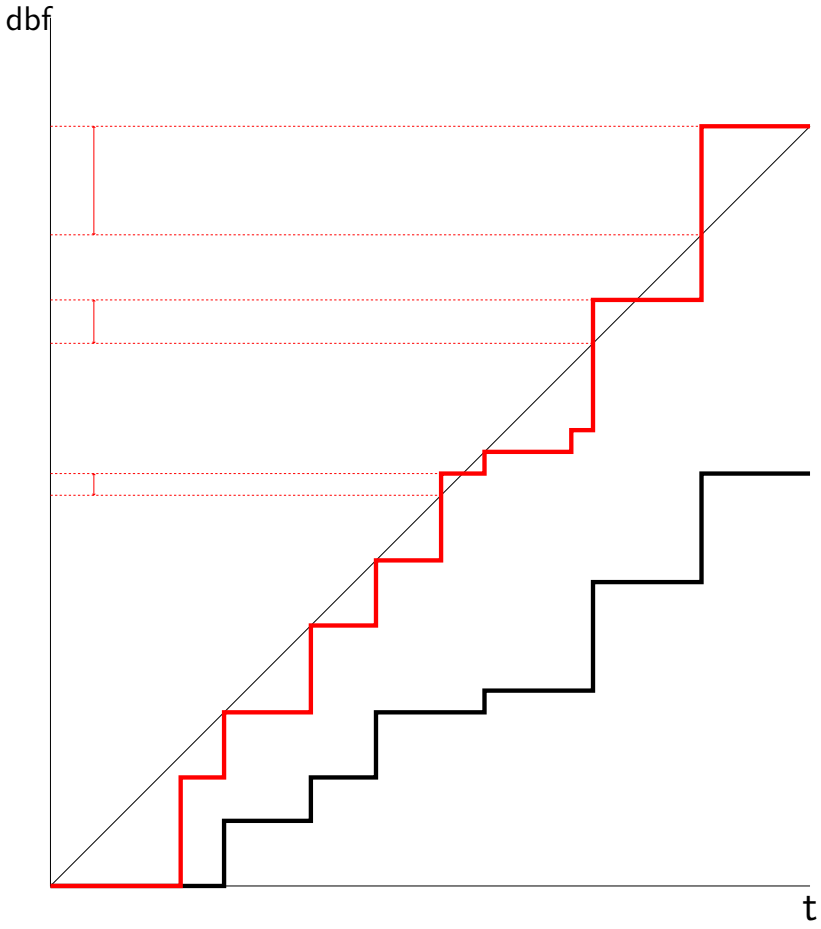


Task set $\mathcal{T}$

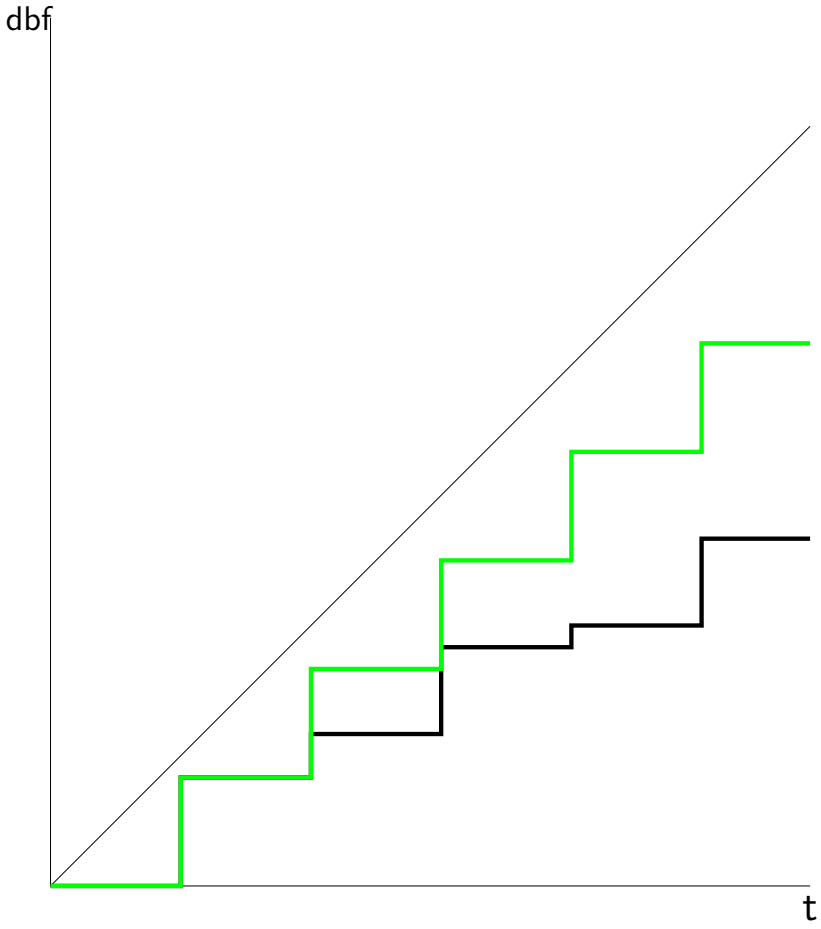


Task τ

Partitioning scheme

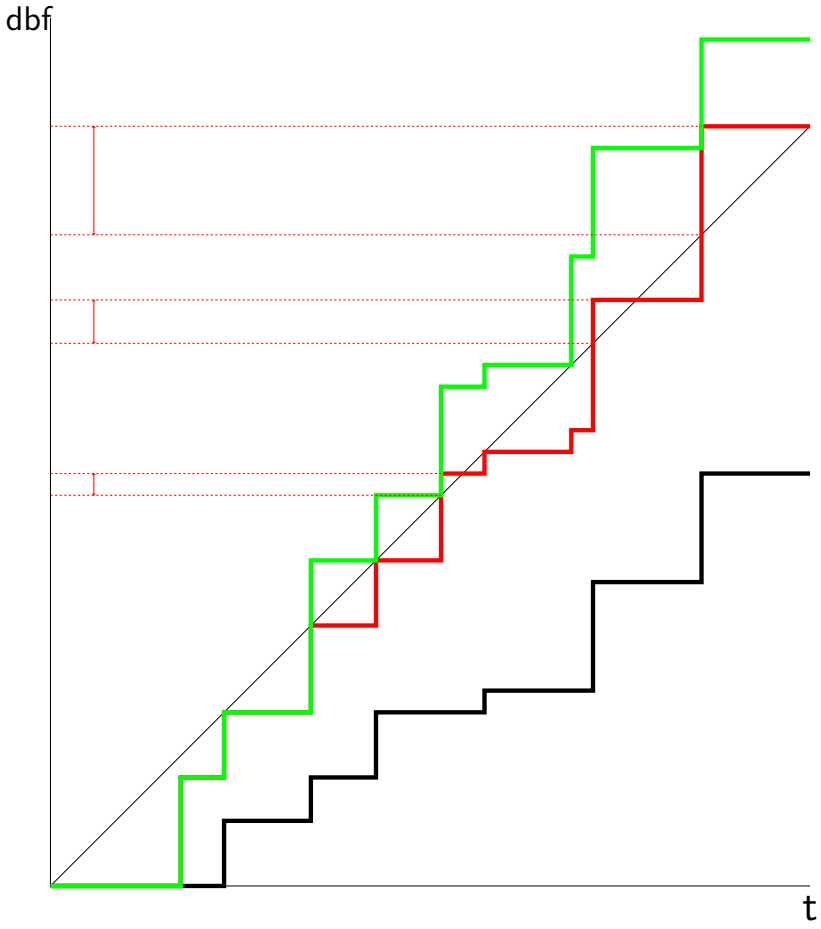


Task set $\mathcal{T}$

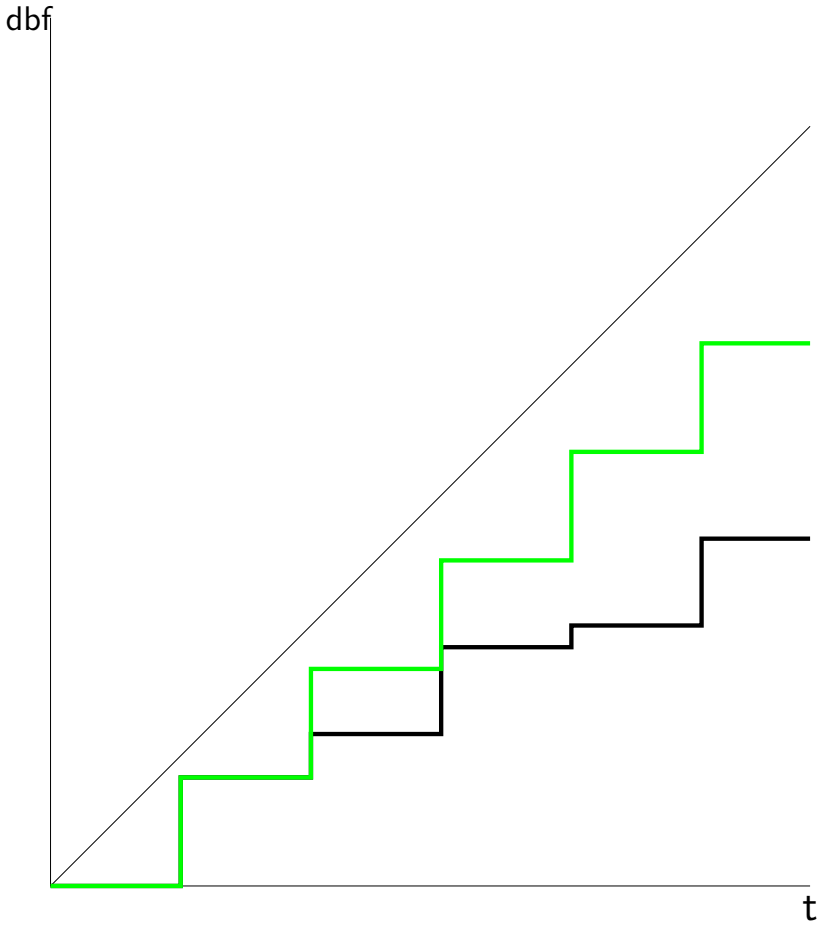


Task τ

Partitioning scheme

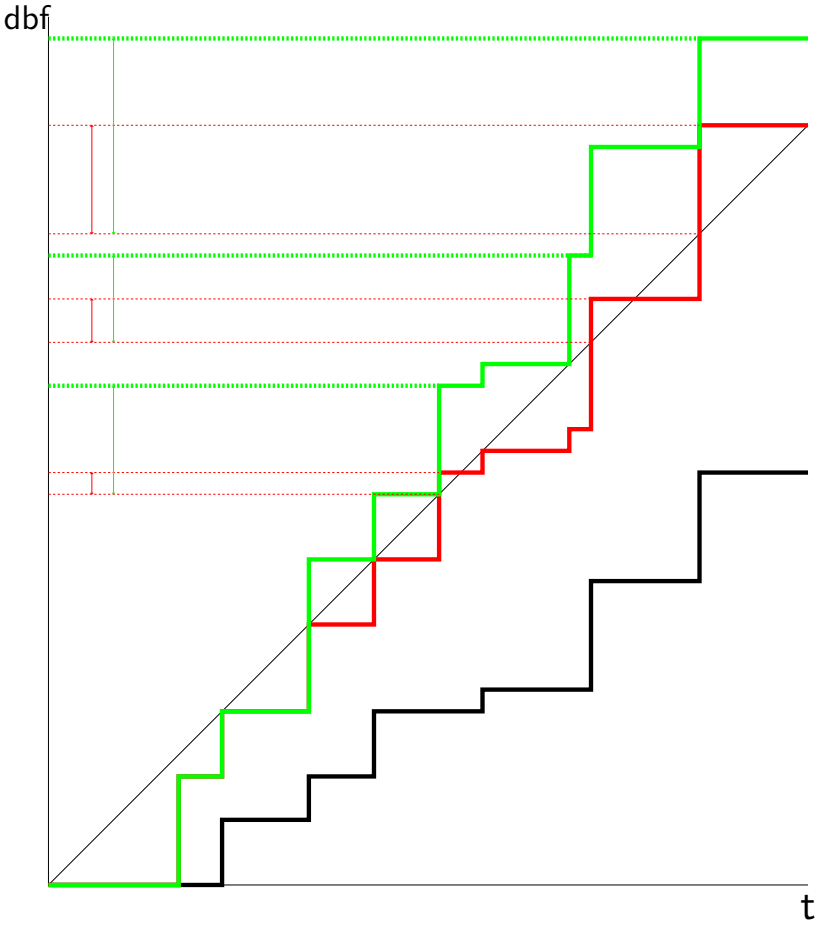


Task set $\mathcal{T}$

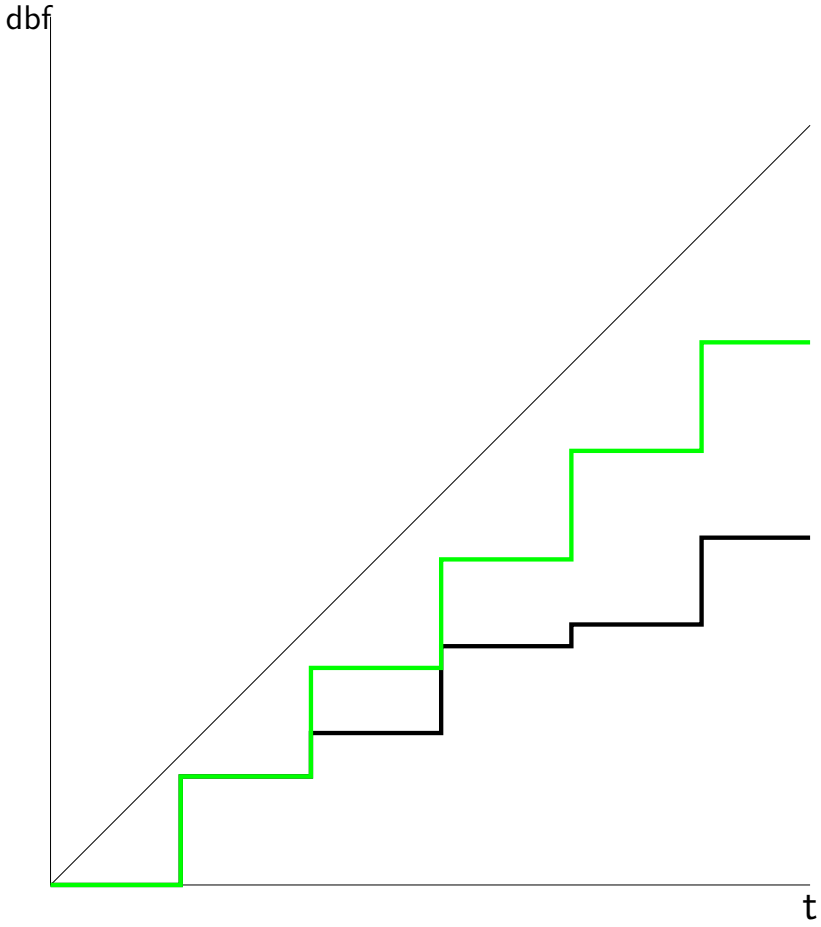


Task τ

Partitioning scheme



Task set $\mathcal{T}$



Task τ

Excess-time **evaluation**

The excess is computed as :

$$\forall v'_{i,j} \in V'_i, \max_{t \in [0,h]} \left\{ t - \right.$$

Excess-time **evaluation**

The excess is computed as :

$$\forall v'_{i,j} \in V'_i, \max_{t \in [0,h]} \left\{ t - (\mathrm{dbf}(\mathcal{T}_k, t) \right.$$

Excess-time **evaluation**

The excess is computed as :

$$\forall v'_{i,j} \in \mathsf{V}'_i, \max_{t \in [0,h]} \left\{ t - (\mathrm{dbf}(\mathcal{T}_k, t) + \left\lfloor \frac{t + \mathsf{T}_i - \mathsf{D}_i}{\mathsf{T}_i} \right\rfloor \cdot \max\{\mathcal{C}^{\mathsf{v}}_{i,j}(\mathsf{f}_{op})\}) \right.$$

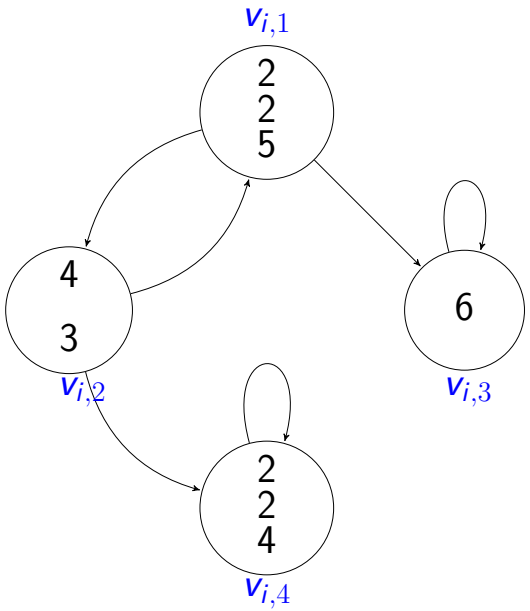
Excess-time **evaluation**

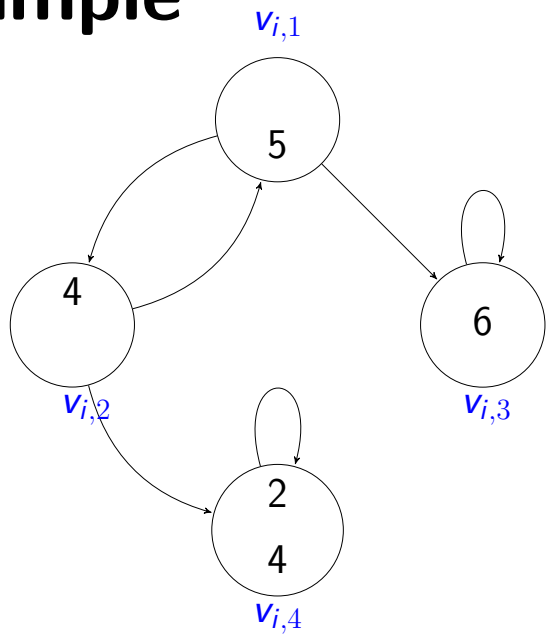
The excess is computed as :

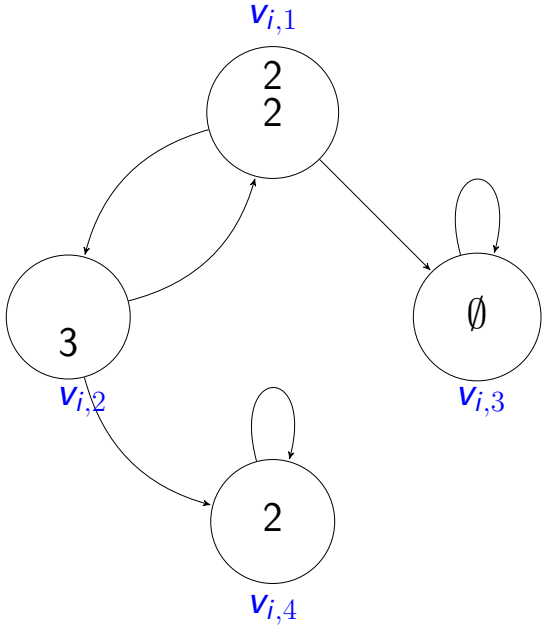
$$\forall v'_{i,j} \in \mathsf{V}'_i,\ \max_{t\in[0,h]} \left\{ \frac{t - (\mathrm{dbf}(\mathcal{T}_k, t) + \left\lfloor \frac{t+\mathsf{T}_i-\mathsf{D}_i}{\mathsf{T}_i} \right\rfloor \cdot \max\{\mathcal{C}^{\mathsf{v}}_{i,j}(\mathsf{f}_{op})\})}{\left\lfloor \frac{t+\mathsf{T}_i-\mathsf{D}_i}{\mathsf{T}_i} \right\rfloor} \right\}$$

Task decomposition example

- Let assume that excess is evaluated to 3.
- and the task to decompose is :







Allocation algorithm

Steps of the allocation algorithm

1. For each task

Allocation algorithm

Steps of the allocation algorithm

1. For each task
2. For each core

Allocation algorithm

Steps of the allocation algorithm

1. For each task
2. For each core
3. Evaluate excess

Allocation algorithm

Steps of the allocation algorithm

1. For each task
2. For each core
3. Evaluate excess
4. If excess == 0 allocate the selected task to the selected core

Allocation algorithm

Steps of the allocation algorithm

1. For each task
2. For each core
3. Evaluate excess
4. If excess $==$ 0 allocate the selected task to the selected core
5. If excess $==$ $\max_j \mathcal{C}^{\mathrm{v}}_{i,j}(\mathrm{f}_{op})$, seek to allocate the task on the next core

Allocation algorithm

Steps of the allocation algorithm

1. For each task
2. For each core
3. Evaluate excess
4. If excess $==$ 0 allocate the selected task to the selected core
5. If excess $==$ $\max_j \mathcal{C}^{\mathrm{v}}_{i,j}(\mathrm{f}_{op})$, seek to allocate the task on the next core
6. If excess is between both, split the task

Allocation algorithm

Steps of the allocation algorithm

1. For each task
2. For each core
3. Evaluate excess
4. If excess $==$ 0 allocate the selected task to the selected core
5. If excess $==$ $\max_j \mathcal{C}^{\mathrm{v}}_{i,j}(\mathrm{f}_{op})$, seek to allocate the task on the next core
6. If excess is between both, split the task
7. If all cores are investigated and no allocation found, Abort

Allocation algorithm

Steps of the allocation algorithm

1. For each task
2. For each core
3. Evaluate excess
4. If excess $==$ 0 allocate the selected task to the selected core
5. If excess $==$ $\max_j \mathcal{C}^{\mathrm{v}}_{i,j}(\mathrm{f}_{op})$, seek to allocate the task on the next core
6. If excess is between both, split the task
7. If all cores are investigated and no allocation found, Abort
8. If the task list is empty, succeed

Allocation algorithm

Frequency Selection

- Our frequency selection algorithm is greedy

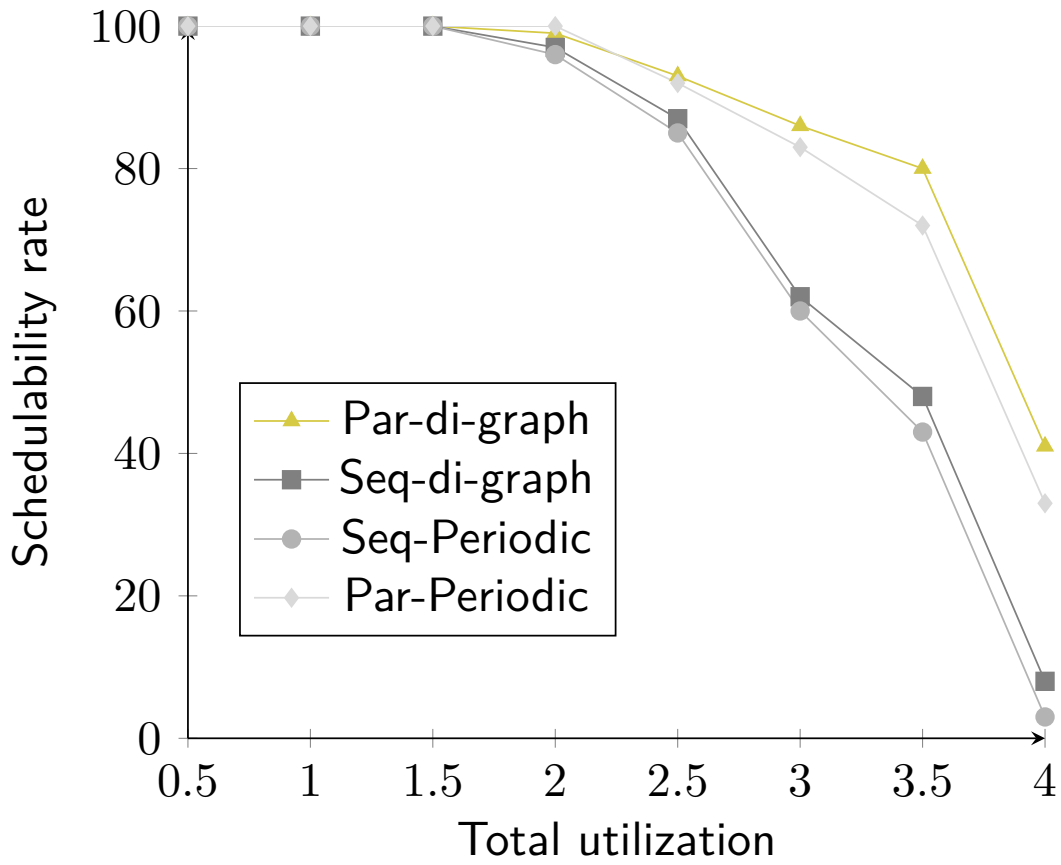
1. Select a frequency
2. Test the schedulability
3. If the test fails, increase the frequency, else return succeed

Plan

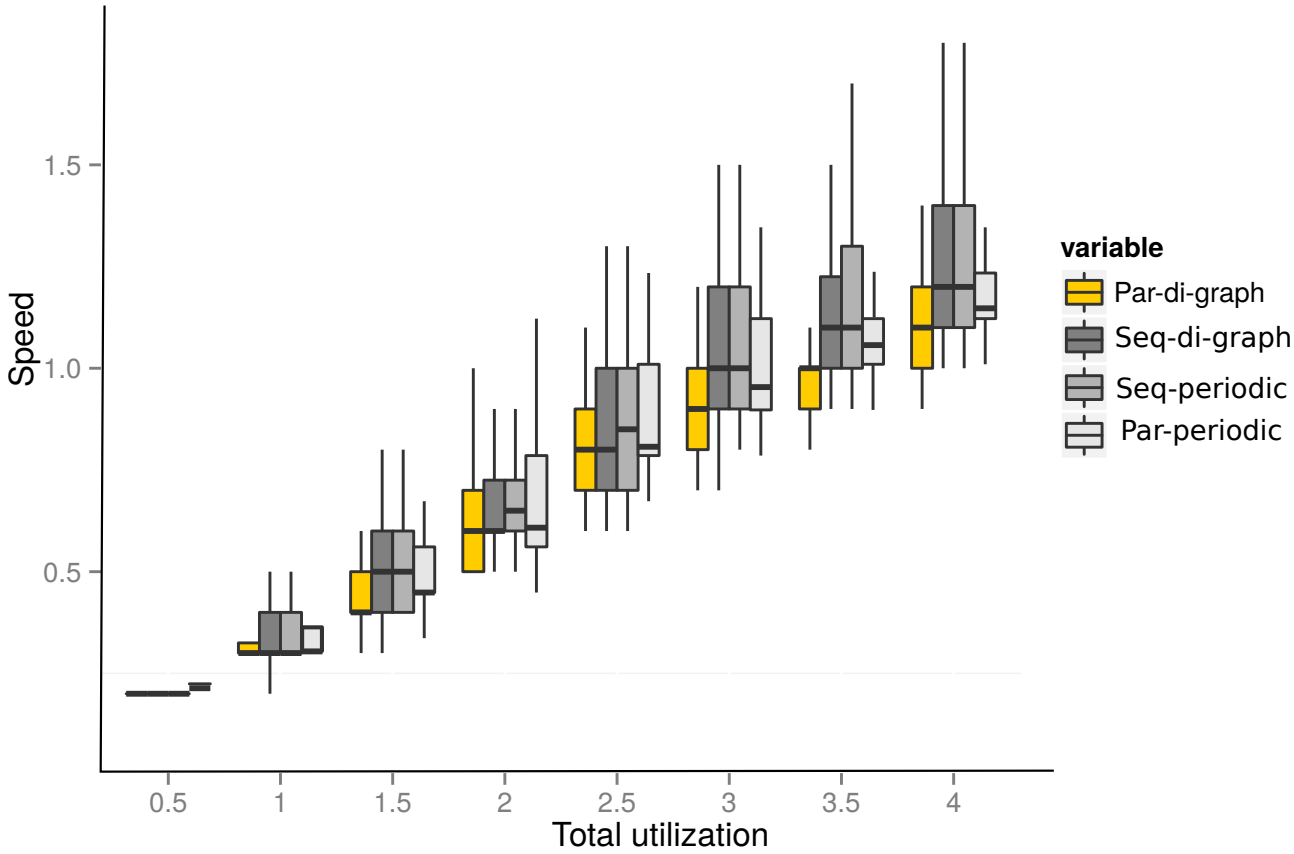
Experimental protocol

- We simulated 4 core platform
- We vary the total utilization from 0.5 to 4 by step 0.5
- We compare our model (Par-di-graph) against :
  - Liu and Layland model (Seq-periodic)
  - a parallel Liu and Layland model (Par-periodic)
  - di-graph model (Seq-di-graph)
- Two scenarios :
  - The variation on utilization between vertices can be in interval ± 0.1 of the original utilization
  - The variation on utilization between vertices can be in interval ± 0.3 of the original utilization

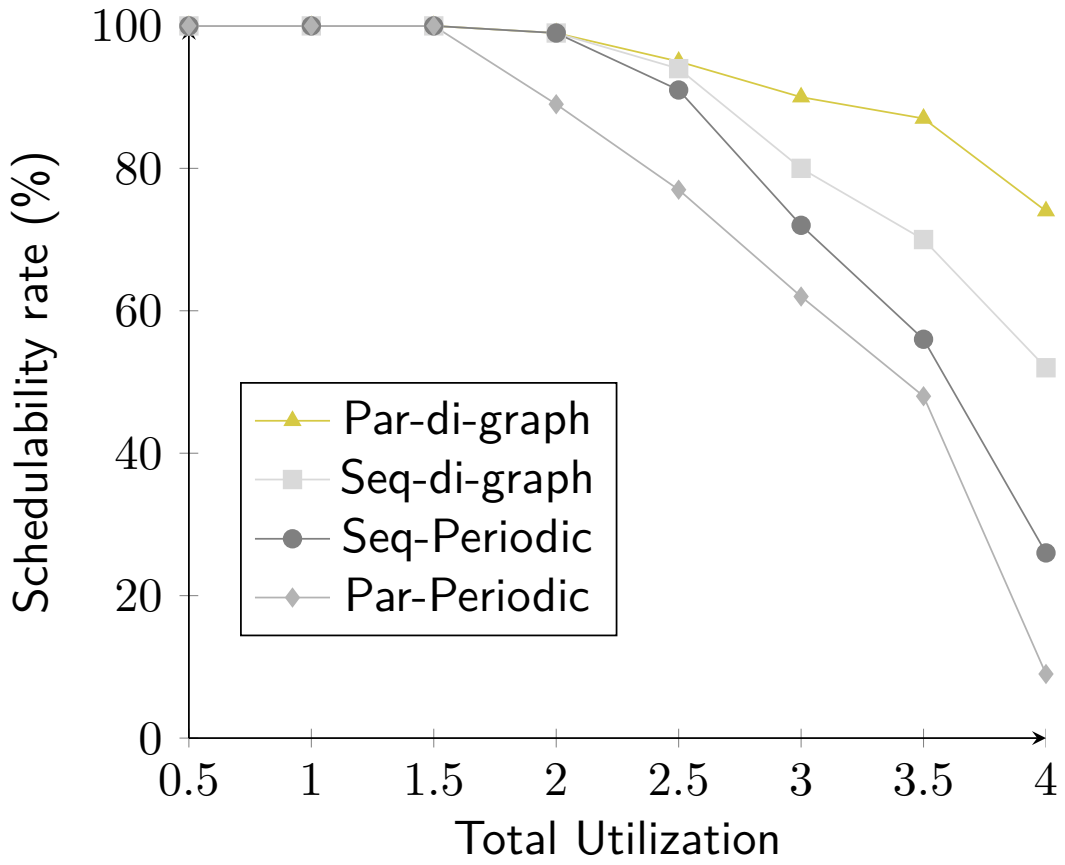
Scenario 1 : Number of schedulable task sets



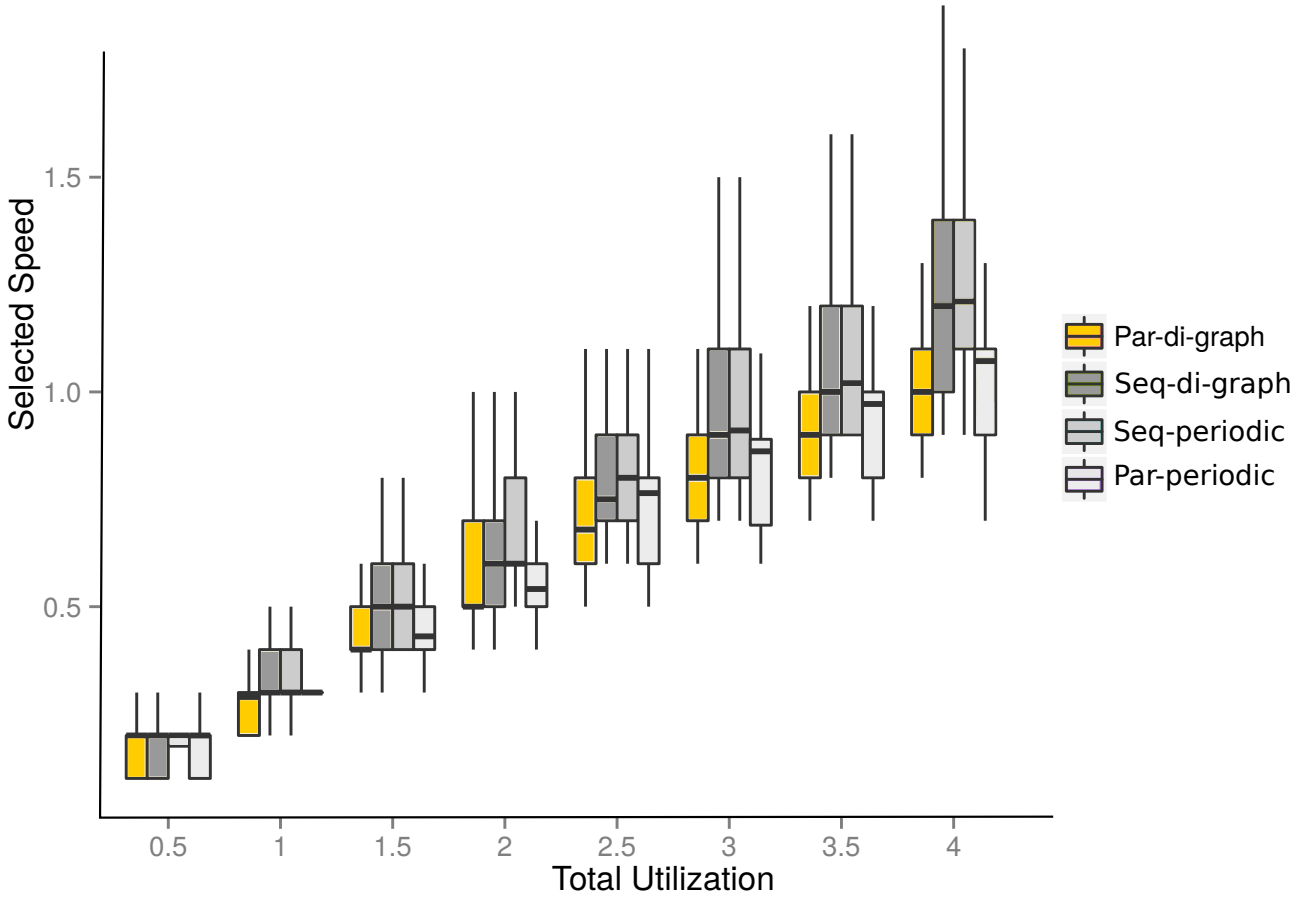
Scenario 1 : Minimization of Speed



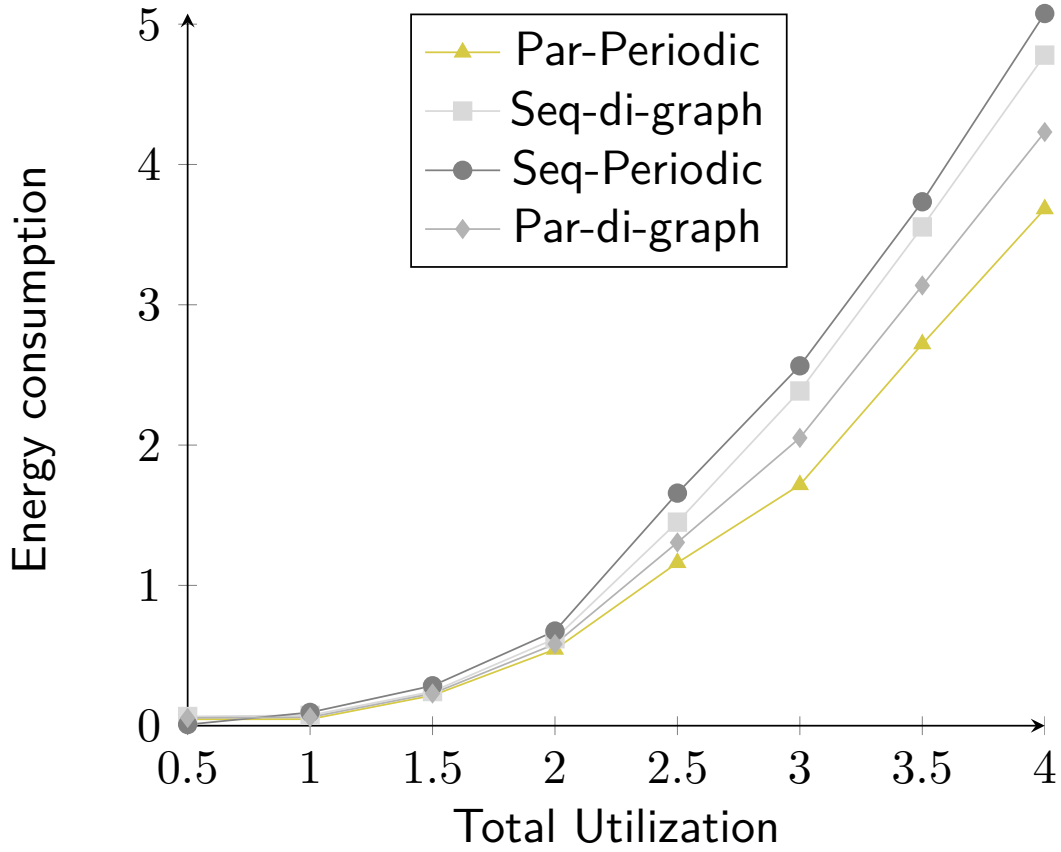
Scenario 2 : Number of schedulable task sets



Scenario 2 : Minimization of Speed



Scenario 2 : Minimization of Energy consumption



Plan

Conclusion & Future Work

- The *parallel di-graph* task model, an extension of the task model proposed by Stigge et al.
- In our model, each vertex is potentially decomposed into a set of parallel threads.
- Sufficient feasibility test for partitioned EDF on a set of identical cores
- Allocate the threads and select the core frequency
- Our model is more effective than other sequential and parallel task models proposed in the literature.

Future Work

- Extend the model : Different arbitrary interarrival time between vetices of the same task (100% of the digraph task model).