femto-st SCIENCES & TECHNOLOGIES

Shelf schedules for independent moldable tasks to minimize the energy consumption

Anne Benoit
Louis-Claude Canon
Redouane Elghazi
Pierre-Cyrille Héam

ENS de Lyon, FEMTO-ST, Univ. Franche-Comté, CNRS

SCALE – December 3, 2021

Scheduling moldable tasks

In a computing center, tasks or jobs must be scheduled, hence a classical scheduling problem for moldable tasks:

- p identical processors;
- n independent *moldable* tasks with task i executed on j processors having a known execution time of $t_{i,j}$;
- for each *moldable* task, the number of processors j must be chosen once at the beginning of the execution, as opposed to *rigid* tasks, for which the number of processors for each task is given.

Scheduling moldable tasks – Example

Example instance, with 3 tasks and 2 processors:

Task 1:	$t_{1,1} = 6$	or	$t_{1,2} = 5$
Task 2:	$t_{2,1} = 5$	or	$t_{2,2} = 3$
Task 3:	$t_{3,1} = 8$	or	$t_{3,2} = 4$

Scheduling moldable tasks – Example

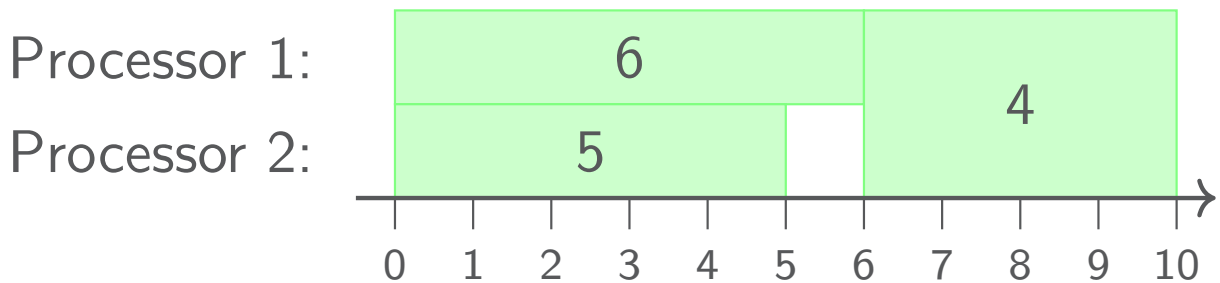
Example instance, with 3 tasks and 2 processors:

Task 1: $t_{1,1} = 6$ or $t_{1,2} = 5$

Task 2: $t_{2,1} = 5$ or $t_{2,2} = 3$

Task 3: $t_{3,1} = 8$ or $t_{3,2} = 4$

Example solution:



Scheduling moldable tasks

This problem has been studied for the minimization of the makespan, which is NP-hard.

- The simpler problem with the additional constraint that all tasks must begin simultaneously is also studied.
- There exist approximation algorithms for the minimization of the makespan.

Scheduling moldable tasks

This problem has been studied for the minimization of the makespan, which is NP-hard.

- The simpler problem with the additional constraint that all tasks must begin simultaneously is also studied.
- There exist approximation algorithms for the minimization of the makespan.

What can we do to minimize the energy consumption of such a schedule?

Introducing the speed

The energy consumption of a processor is not a linear function of the processor speed: changing the speed will change the energy consumption of the schedule.
Changing the speed can be achieved using DVFS (Dynamic Voltage and Frequency Scaling).

Introducing the speed

The energy consumption of a processor is not a linear function of the processor speed: changing the speed will change the energy consumption of the schedule.
Changing the speed can be achieved using DVFS (Dynamic Voltage and Frequency Scaling).

Now, we can formulate the problem of minimizing the energy of a schedule for moldable tasks.

Outline

Problem formulation

We consider the problem MinE-Mold, with as input:

- p identical processors with a static power P_{stat} and a set $S = \{s_1, s_2, \ldots, s_k\}$ of possible speeds;
- n moldable tasks $\{T_1, T_2, \ldots, T_n\}$ with execution profiles $(w_{i,j})_{i \in [\![1,n]\!], j \in [\![1,p]\!]}$;
- the execution time of a task i executed on j processors with speed s is $t_{i,j,s} = \frac{w_{i,j}}{s}$.

Problem formulation

We consider the problem MinE-Mold, with as input:

- p identical processors with a static power P_{stat} and a set $S = \{s_1, s_2, \ldots, s_k\}$ of possible speeds;
- n moldable tasks $\{T_1, T_2, \ldots, T_n\}$ with execution profiles $(w_{i,j})_{i \in [\![1,n]\!], j \in [\![1,p]\!]}$;
- the execution time of a task i executed on j processors with speed s is $t_{i,j,s} = \frac{w_{i,j}}{s}$.

The objective function we want to minimize is the energy consumption, which is the sum of two parts:

- the static energy consumed by a processor u is $E_{u,stat} = T_{u,stat} \times P_{stat}$, where $T_{u,stat}$ is the total powered up duration of the processor;
- the dynamic energy consumed by a task i executed on j processors with speed s is $E_{i,dyn} = j \times t_{i,j,s} \times s^{\alpha}$, where α is a constant between 2 and 3.

MinE-Mold-Indep is tractable

In a first case, we consider the problem MinE-Mold-Indep: the total powered up duration of a processor is its activity time (i.e., it is independent of the other processors).

MinE-Mold-Indep is tractable

In a first case, we consider the problem MinE-Mold-Indep: the total powered up duration of a processor is its activity time (i.e., it is independent of the other processors).

Theorem

MinE-Mold-Indep *can be solved in polynomial time.*

This is done by independently minimizing the energy of each task as the total energy can be expressed as the sum of the energy of each task. Tasks can then be executed sequentially (makespan minimization is not the objective).

MinE-Mold-Sim is NP-complete

In a second case, we consider the problem MinE-Mold-Sim: the processors need to be powered up simultaneously (i.e., $T_{u,stat} = T_{stat}$).

MinE-Mold-Sim is NP-complete

In a second case, we consider the problem MinE-Mold-Sim: the processors need to be powered up simultaneously (i.e., $T_{u,stat} = T_{stat}$).

Theorem

The decision problem associated to MinE-Mold-Sim *is NP-complete.*

To prove that, we do a reduction from 3-Partition, with each processor corresponding to a different subset.

MinE-Mold-Sim with one shelf is tractable

In a third case, we consider the problem MinE-Mold-Sim as previously defined, but with one single shelf (i.e., under the constraint that all tasks need to start at the same time).

MinE-Mold-Sim with one shelf is tractable

In a third case, we consider the problem MinE-Mold-Sim as previously defined, but with one single shelf (i.e., under the constraint that all tasks need to start at the same time).

Theorem

MinE-Mold-Sim *with one shelf can be solved in polynomial time.*

This is done through dynamic programming by solving all sub-instances $\mathcal{I}_{k,p',i,j,s}$ defined as followed:

- we only consider the tasks $1, 2, \ldots, k$;
- we only use up to p' processors;
- we let all p processors powered up for a duration $t_{i,j,s}$.

Outline

Two categories

As the problem is NP-complete, we studied approximation algorithms for MinE-Mold-Sim.

We first provide two approximation algorithms for the rigid case MinE-Rig-Sim, where tasks have a predefined number of processors:

- ListBased, a list-based algorithm that lists the tasks in some order, and then assigns them greedily;
- ShelfBased, a shelf-based algorithm that creates batches of tasks to be executed one after the other, with each task of a batch starting at the same time.

We then provide a way to transform approximation algorithms for the rigid case into approximation algorithms for the moldable case.

Approximation ratios

In the following table, we present the approximation ratios for our two algorithms for two problems:

Algorithm	MinE-Mold	MinE-Mold with the same speed for all tasks
ListBased	3-approximation	2-approximation
ShelfBased	3-approximation	3-approximation

The proofs for these results are based on:

- existing ratios for the makespan;
- the fact that among all the schedules these algorithms will try, one will be well balanced between static and dynamic energy.

Approximation ratios

In the following table, we present the approximation ratios for our two algorithms for two problems:

Algorithm	MinE-Mold	MinE-Mold with the same speed for all tasks
ListBased	3-approximation	2-approximation
ShelfBased	3-approximation	3-approximation

The proofs for these results are based on:

- existing ratios for the makespan;
- the fact that among all the schedules these algorithms will try, one will be well balanced between static and dynamic energy.

Sophisticated proofs, check the paper for details!

Outline

Empirical setup

- We tested our algorithms through simulations, monitoring both the execution time of the algorithms and the quality of their output.
- The algorithms have been implemented in C++.
- The simulations were set up and exploited in Python 3.8.5.
- We compared both the versions using multiple speeds for different tasks (MS), or a single speed for all the tasks (SS).

- All the code is available on Figshare[①].

[①] https://doi.org/10.6084/m9.figshare.14854395

Compared algorithms

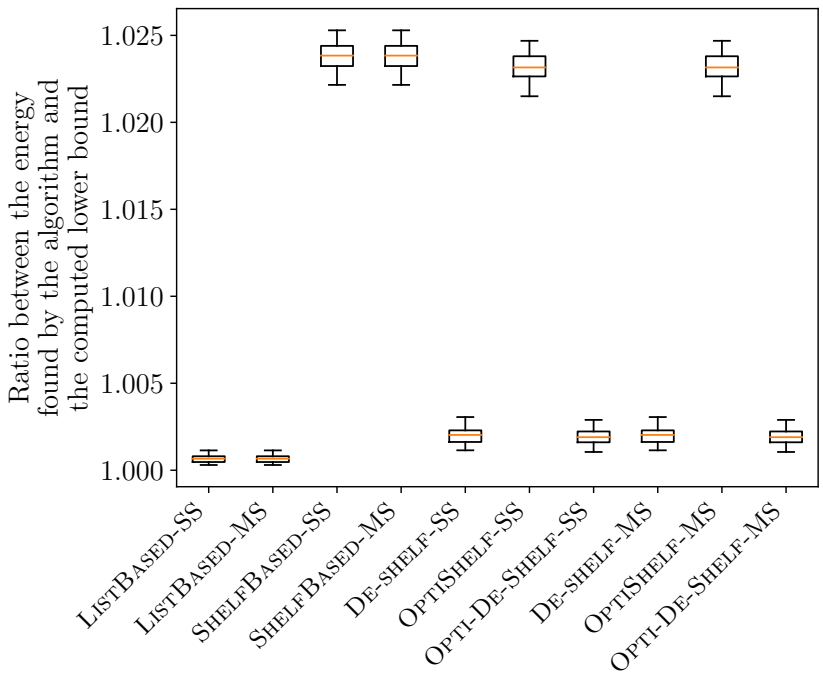
We compare various algorithms:

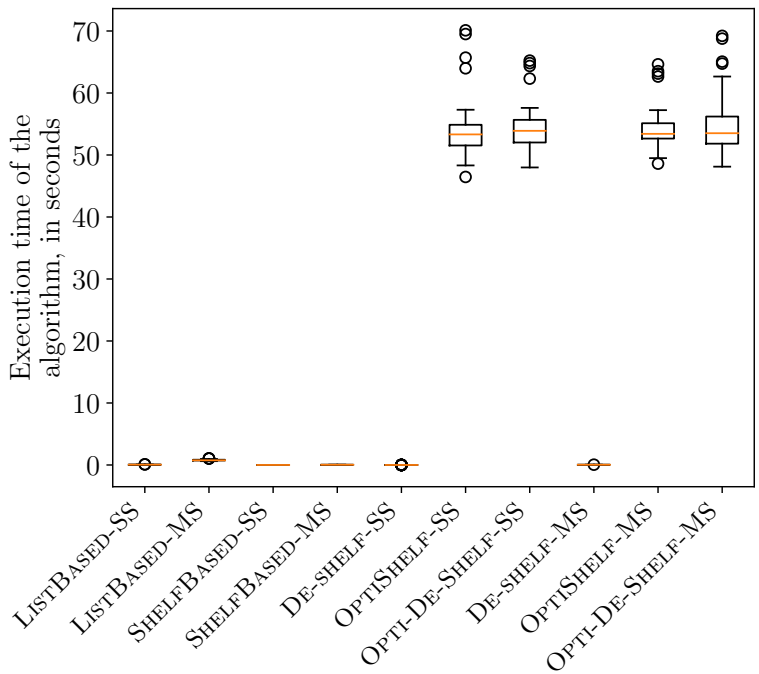
- ListBased that sorts the tasks and allocates them greedily as soon as resources become available;
- ShelfBased that creates shelves, or batches of tasks, to be executed one after the other;
- OptiShelf that is based on ShelfBased but that optimizes each batch using the exact algorithm that we have for a single shelf;
- De-shelf that is based on ShelfBased but that then removes the constraint of shelves;
- Opti-De-Shelf that combines OptiShelf and De-shelf.

Each algorithm also has a suffix that determines whether it allows different speeds for different tasks (MS for Multiple Speeds) or not (SS for Single Speed).

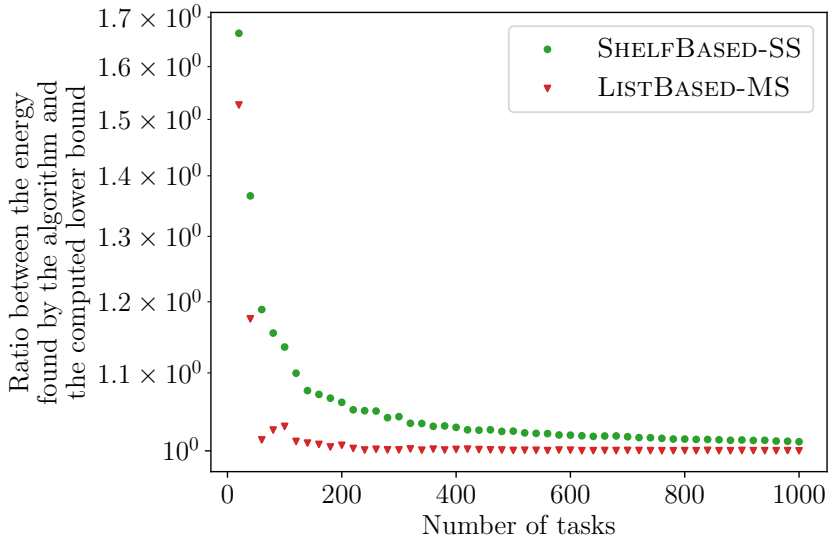
Full comparison

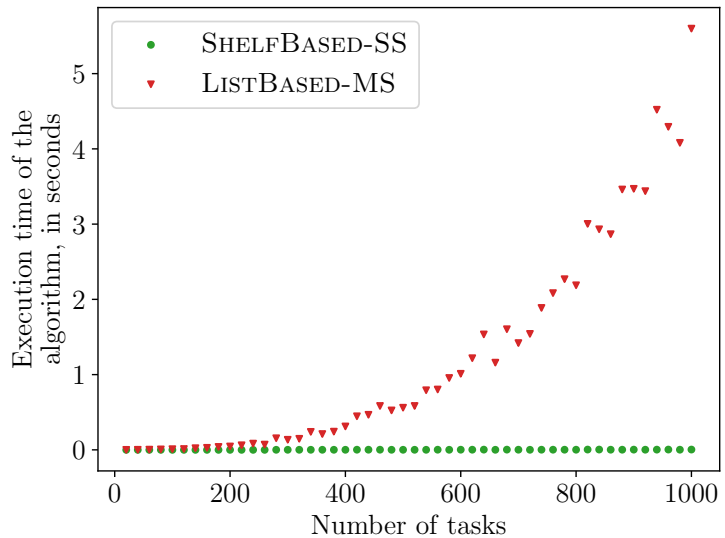
Results for all algorithms and their variants:



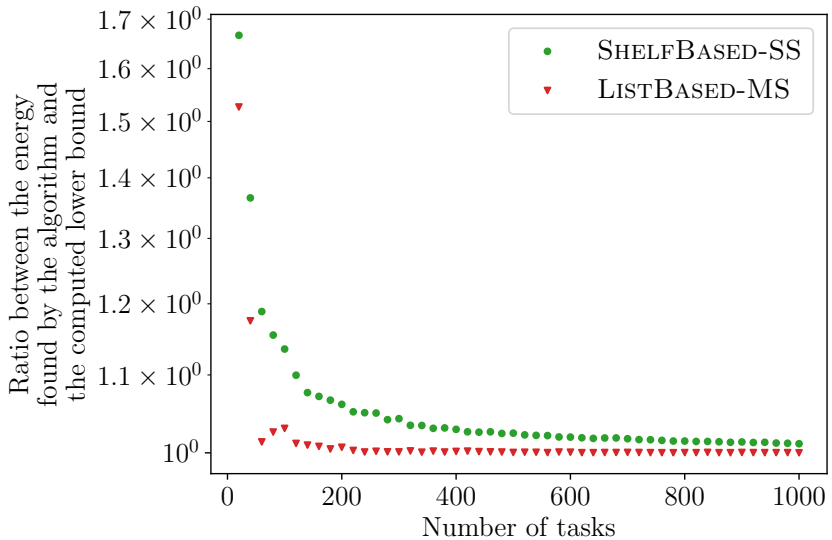


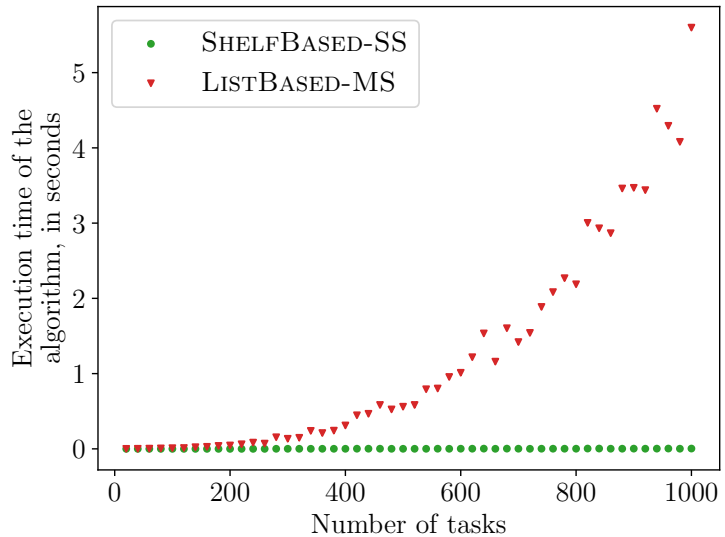
ListBased-MS against ShelfBased-SS





ListBased-MS against ShelfBased-SS





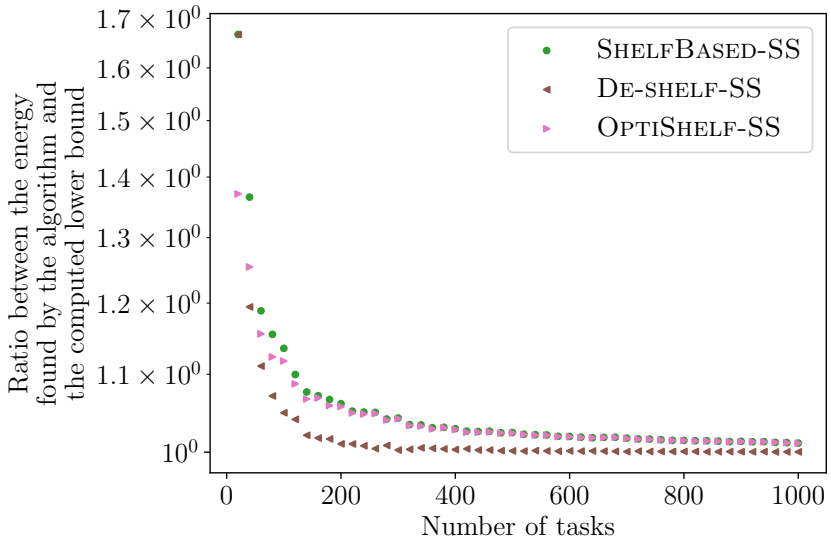
ListBased-MS gives better results than ShelfBased-SS at the cost of a much larger time complexity: a trade-off?

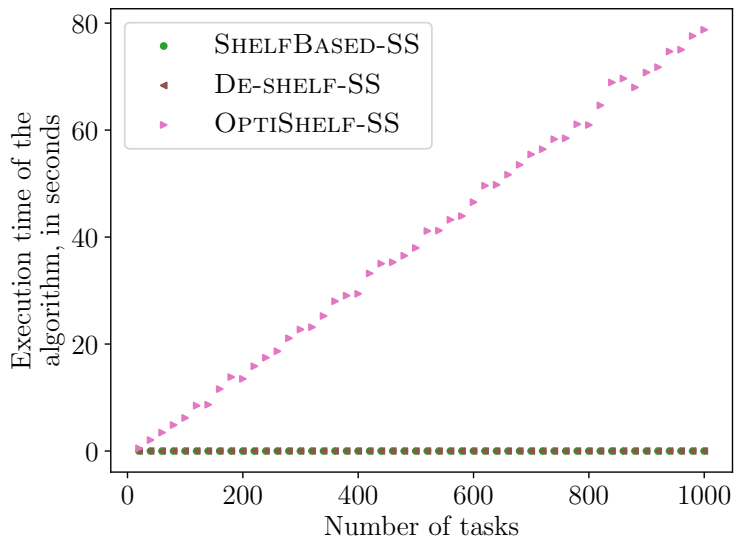
Optimizing ShelfBased-SS

The solution space explored by ShelfBased is very structured, leaving the possibility to optimize even further the result.

Optimizing ShelfBased-SS

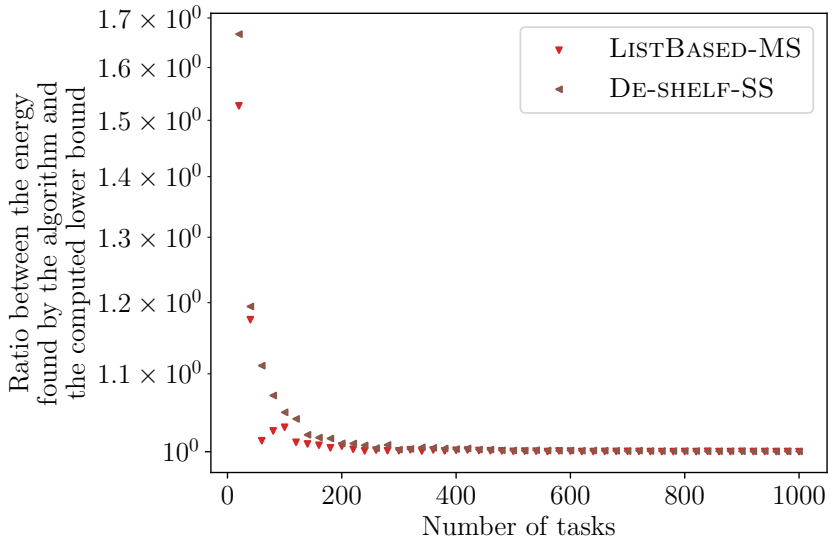
The solution space explored by ShelfBased is very structured, leaving the possibility to optimize even further the result.

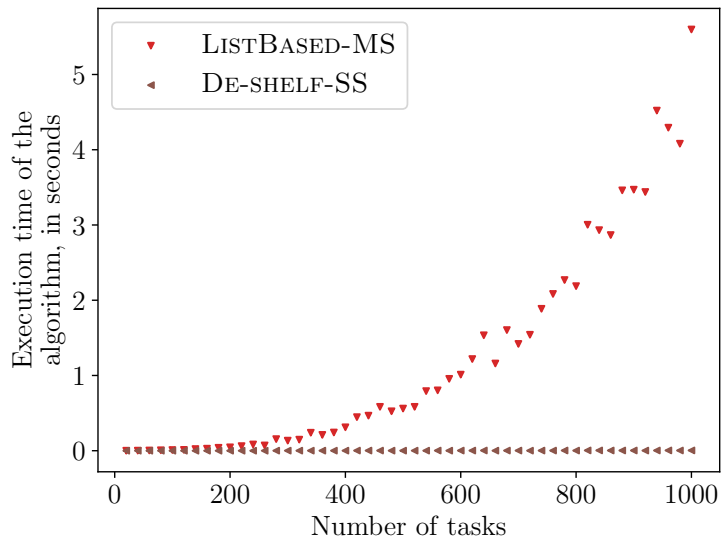




ListBased-MS against optimized ShelfBased-SS

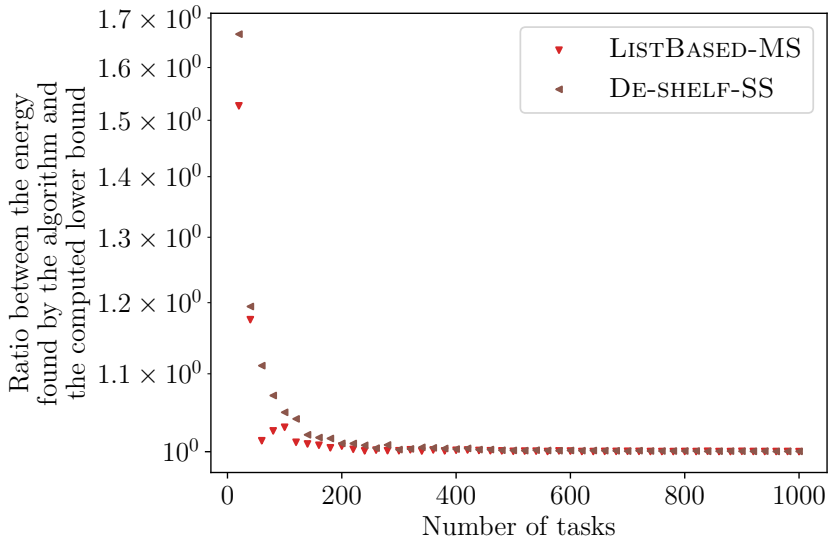
We now compare the best version of ShelfBased to ListBased-MS:

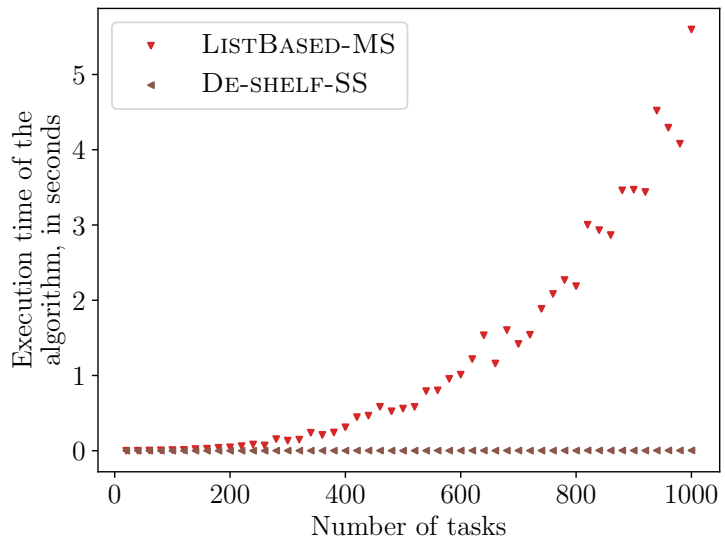




ListBased-MS against optimized ShelfBased-SS

We now compare the best version of ShelfBased to ListBased-MS:





The optimized version of ShelfBased competes with ListBased for most inputs, while having a much lower time complexity.

Outline

Conclusion

What has been done:

- formulating several problems of energy minimization for scheduling independent moldable tasks;
- proving that MinE-Mold-Indep is tractable, while MinE-Mold-Sim is NP-complete;
- providing algorithms for MinE-Mold-Sim and proving multiple approximation ratios for those algorithms;
- providing an algorithm minimizing the energy consumption under the constraint that all tasks start at the same time (i.e., with only one shelf);
- providing an empirical study comparing various algorithms, allowing us to identify the best optimization for ShelfBased, De-shelf-SS.

Future work

Future work:

- compare an optimized version of ListBased to De-shelf;
- conduct experiments on real HPC systems;
- consider other energy models (e.g., change the energy formula, introduce a cost of time and energy for any speed change, ...).

Thank you for your attention

redouane.elghazi@femto-st.fr

femto-st
SCIENCES & TECHNOLOGIES