

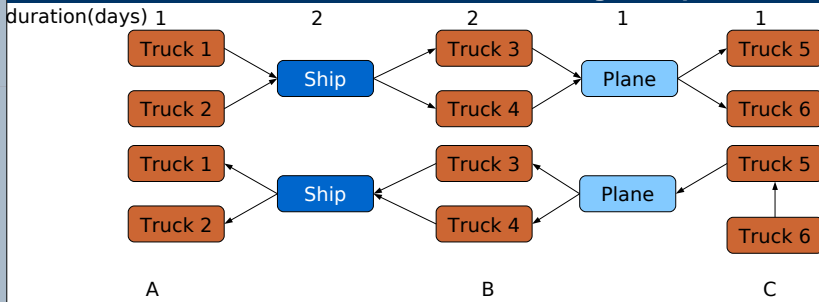
Autonomous Scheduling

Chetan Yadati¹, Cees Witteveen¹, Yingqian Zhang¹, Mengxiao Wu², Han la Poutre²

¹ Delft University of Technology, Delft

² Centrum voor Wiskunde en Informatica Amsterdam

Motivating example: Multi-Modal transportation



Objective:

"to transport goods from A to C Via B and from C to A via B with **minimal make-span**"

Setup

- Three agents are involved - the truck agent with 6 trucks, the ship agent with a single ship and the plane agent with a single plane.
- Each of the agents can perform a different **fixed number of tasks** simultaneously.
- Agents **unable/unwilling** to share exact schedules

Note:

The best case make-span achievable is 8 days whereas the worst case make-span possible is 16 days

Problem definition and framework

Autonomous scheduling problem

Given a number of actors who each have to schedule a set of tasks that might be **interdependent**.

Find a **minimal** set of constraints such that

- each actor is able to schedule its own activities **independently** of the others
- the merge of their schedules is always a **make-span efficient** schedule.

Framework

- Set $T = \{t_1, \dots, t_m\}$ of tasks with durations $d: T \rightarrow \mathbb{Z}^+$, precedence constraints $t_i < t_j$ between tasks constitute partial order $(T, <)$.
- Task allocation function $\varphi: T \rightarrow A$ with $A = \{a_1, \dots, a_n\}$ set of agents.
- Concurrency bound $c: A \rightarrow \mathbb{Z}^+$. it specifies the number of tasks an agent can perform simultaneously.

Agents with Unbounded concurrency

Intuition

For **unbounded** concurrency, **Critical Path Method (CPM)** can be used to determine **earliest** starting time and **latest** starting time interval for tasks. Overlap between intervals for tasks dependent upon each other and allocated to different agents is removed.

The Complete Algorithm (ISA)

- Find the depth() and height() of every task.
- Compute $depth(T)$ as the maximum depth of the task graph.
- Set the earliest starting time for a task $t = lb(t) = depth(t)$.
- Set the latest starting time for a task $t = ub(t) = depth(T) - height(t)$.
- If intervals overlap,

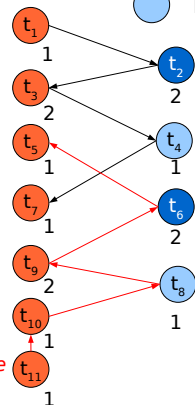
$$ub(t) = lb(t) + \left\lfloor \frac{ub(t') - lb(t) - d(t)}{2} \right\rfloor$$

$$lb(t') = ub(t) + d(t)$$

- Truck tasks
- Ship tasks
- Plane tasks

Example: Multi-modal transportation

$t_1 = [0,0]; t_2 = [1,1]$
 $t_3 = [3,3]; t_4 = [5,5]$
 $t_5 = [7,7]; t_6 = [5,5]$
 $t_7 = [6,7]; t_8 = [2,2]$
 $t_9 = [3,3]; t_{10} = [1,1]$
 $t_{11} = [0,0]$
 Makespan = 8 days



Results

- ISA ensures **minimum make-span**
- No algorithm can guarantee more flexibility for agents.**

Conclusions

We have shown that one can design additional precedence constraints on a given scheduling problem to enable autonomous scheduling by agents.

- The ISA is a surprisingly simple but yet very efficient algorithm which also ensures maximum flexibility in case of unbounded agents.
- In the case of sequential agents it is NP-hard to find a make-span optimal schedule, but ISA can be adapted to obtain a 2-approximate algorithm

Future work

- To study structural properties of the task graph as it can have a significant influence on both the running time of the algorithms and the approximation ratio.
- Handling non-preemptive tasks
- Handling dynamic task networks

Agents with Bounded concurrency

Intuition

We have shown that if tasks are unit duration, then maximum matching can be used to find a 2 approximate solution for autonomous scheduling. Thus, we allow for pre-emption and split tasks into unit duration tasks and then apply matching to derive the intervals. Conflicts are resolved by arbitrarily choosing to precede one conflicting task over the other.

The Complete Algorithm (ISA_SEQ)

Step 1 Split each task into a sequence of unit duration tasks.

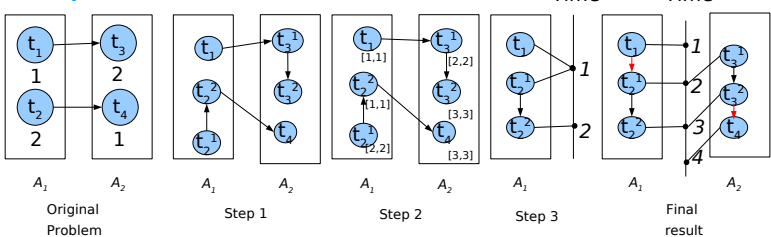
Step 2 Compute intervals using ISA

Step 3 If a maximal matching is found for each agent, then quit.

Step 4 Else, until a maximal matching is found, **iteratively**

- Impose an **additional** precedence constraint between **conflicting** tasks,
- Compute the new time intervals using ISA,
- Extend the set of time points

Example:



Results

ISA_SEQ is a 2-approximate algorithm.

