+

Tractable Scheduling Problems

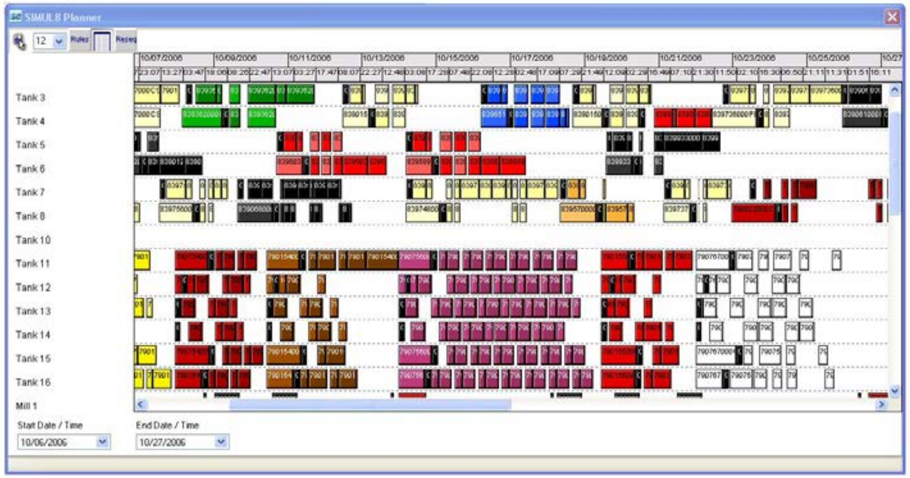
Geoffrey Chu, Serge Gaspers, Nina Narodytska, Andreas Schutt, **Toby Walsh**

To appear in Proc. of IJCAI 2013, Beijing, China

+

Motivation

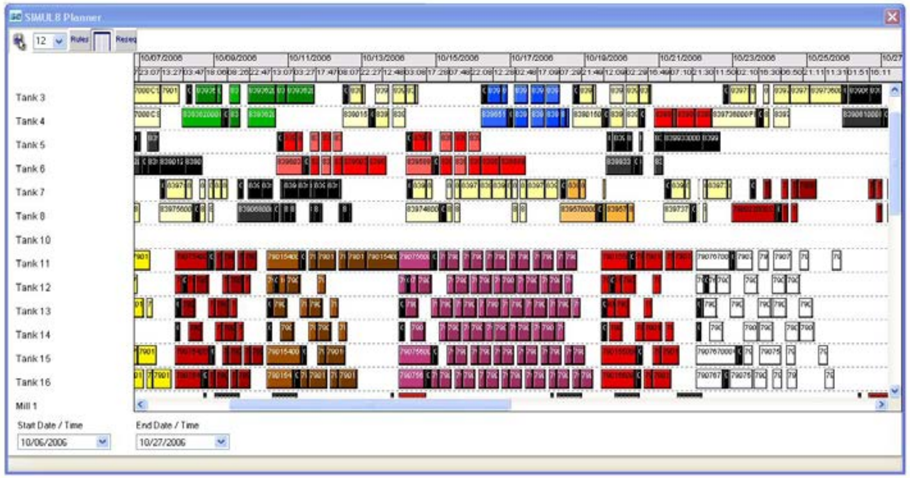
- Scheduling an important but computationally challenging problem
  - Given: number of machines, set of jobs, each with a release time and due date, duration, resources required
  - Question: can we find start times and machines for each job to satisfy release time, due date, and resource usage constraints?



+

Motivation

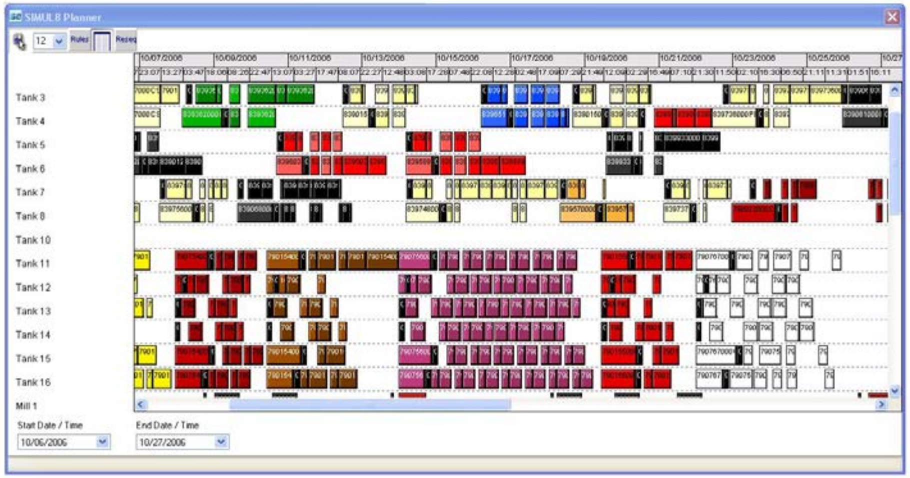
- Scheduling an important but computationally challenging problem
  - NP-hard in general
  - Can we identify structural restrictions under which it becomes fixed parameter tractable?



+ Motivation

- Scheduling an important but computationally challenging problem
  - NP-hard in general
  - Can we identify structural restrictions under which it becomes fixed parameter tractable?
    - Number of processors
    - Number of start times
    - Structure of release and due dates
    - …

  [Fellows, Gaspers & Rosamond, Parameterizing by the number of numbers]



Parameterizing by the Number of Numbers [Fellows, Gaspers, Rosamond 2010]

- Number problems
  - Subset sum
  - Partition
  - 3-Partition
  - 3-D Matching
  - …
- Input
  - Bag of numbers
  - Interesting parameter: number of numbers

Parameterizing by the Number of Numbers [Fellows, Gaspers, Rosamond 2010]

- Number problems
  - Subset sum
  - Partition
  - 3-Partition
  - 3-D Matching
  - …
- Input
  - Bag of numbers
  - Interesting parameter: number of numbers
- Often FPT
  - Often use an ILP encoding with polynomial number of vars
  - E.g. Subset sum is FPT in number of numbers being partitioned

Parameterizing by the Number of Numbers [Fellows, Gaspers, Rosamond 2010]

- Number problems
  - Subset sum
  - Partition
  - 3-Partition
  - 3-D Matching
  - …
- Input
  - Bag of numbers
  - Interesting parameter: number of numbers
- Often FPT
  - Often use an ILP encoding with polynomial number of vars
  - E.g. Subset sum is FPT in number of numbers being partitioned
- Inspired perhaps a little by [Bessiere et al AAAI 2008]?
  - Propagating global constraint like NVALUE is NP-hard but FPT in number of numbers in domains

Outline of this talk

- 4 case studies
  - 3 positive (FPT algorithms)
  - 1 negative (NP-hard even under strong structural restrictions)
- Previous work limited
  - [Marx 2011] observed limited work on FPT algorithms for scheduling
  - One exception where parameter is tree width of precedence graph and (#late tasks or #tasks on time)

Global constraints in scheduling

- CUMMULATIVE
- INTER DISTANCE

+ Global constraints in scheduling

- CUMMULATIVE
  - Introduced in CHIP
  - Each task has release time, length, due date and resource usage
  - Can we schedule tasks so tasks execute or or after release time, finish before due dates without exceeding resource capacity?
    - NP-hard to enforce domain consistency
- INTER DISTANCE

Global constraints in scheduling

- CUMMULATIVE
  - Introduced in CHIP [1993]
  - Each task has release time, length, due date and resource usage
  - Can we schedule tasks so tasks execute or or after release time, finish before due dates without exceeding resource capacity?
    - NP-hard to enforce domain consistency

- INTER DISTANCE
  - Introduced in [Regin 1997]
  - Scheduling equal length tasks on a single machine
  - Each task has a set of possible start times
  - Generalizes AllDifferent
    - $|S_i-S_j| \geq 1$
    - $|S_i-S_j| \geq m$

Bounded task types, single machine

- Suppose tasks divide into a small number of types
  - Within each type, same release times, due dates, length, and precedences
- THM: Checking consistency of CUMMULATIVE is FPT in number of task types
  - Proof: Divide problem into blocks (periods with no release times or due dates). Within each block, we can put all tasks of the same type together into a "run". Then construct ILP with polynomial number of vars

    Sij = start time in the ith block of the run of task type j

    Xij = number of repititions of task type j in ith block

Bounded task types, multiple machines

- Suppose tasks divide into a small number of types
  - Within each type, same release times, due dates, length, and precedences
  - Each task requires a single machine
- THM: Checking consistency of CUMMULATIVE is FPT in number of task types + number of processors
  - Proof: Similar ILP constructed but now with another index for machine

Nested task types, single machine

- Suppose tasks are nested
  - Think Russian doll
  - $r_1 < r_2 < \ldots < r_m < d_m < \ldots < d_2 < d_1$
  - E.g. take item apart then put it back together
- THM: Checking consistency of CUMMULATIVE is FPT in number of task types
  - Proof: Can push all tasks to left, build ILP with polynomial number of vars encoding such solutions
  - Corrects [Braind et al 2006] who claim this is NP-hard

Pairwise overlapping start intervals

- Start interval of task = $[s_i, d_i - l_i]$
- For any time point, t let S[t] be number of tasks whose start interval contains t
- Let k = max value of S[t] across all time points
  - Related to well known disjunctive ratio [Baptiste & Pappe 2000]

Pairwise overlapping start intervals

- Start interval of task = $[s_i,d_i-l_i]$
- For any time point, t let S[t] be number of tasks whose start interval contains t
- Let k = max value of S[t] across all time points
- THM: Checking consistency of CUMMULATIVE is FPT in k
  - Proof: Dynamic program over all possible subsets of tasks. Only polynomial number need be examined due to problem constraints.

+ InterDistance constraint

- Interesting special case of CUMMULATIVE
  - Single machine
  - All tasks of same length
- Polynomial cases
  - Start times are intervals [Artiouchine & Baptiste 2007]
  - Task length = 1 [Regin 1994]

InterDistance constraint

- Interesting special case of CUMMULATIVE
  - Single machine
  - All tasks of same length
- Polynomial cases
  - Start times are intervals [Artiouchine & Baptiste 2007]
  - Task length = 1 [Regin 1994]
- THM: Checking consistency of InterDistance is NP-hard even if task lengths = 2 and start times contain at most two intervals

InterDistance constraint

- Interesting special case of InterDistance
  - Start times are all of form $\{s_i, s_i+h, s_i+2h, \ldots, s_i+k_ih\}$
  - E.g. aircraft landing times!
- THM: Checking consistency of InterDistance is NP-hard even if k_i in $\{1,k\}$

+ InterDistance constraint

- Interesting special case of InterDistance
  - Start times are all of form $\{s_i, s_i+h, s_i+2h, \ldots, s_i+k_ih\}$
  - E.g. aircraft landing times!
- THM: Checking consistency of InterDistance is NP-hard even if k_i in $\{1,k\}$
  - But linear when $k_i = k$

Conclusions

- We can exploit structural properties of scheduling to identify tractable cases
  - E.g. CUMMULATIVE with bounded task types
  - E.g. CUMMULATIVE with bounded number of nested task types
  - E.g. CUMMULATIVE with bounded number of pairwise overlapping start intervals
- But some scheduling problems resist such analysis
  - InterDistance constraint with start times of form s_i+j.h
- Open question
  - What other common structural parameters give tractability?

Questions?

PS I'm running a summer school on optimsation/constraint solving

http://www.cse.unsw.edu.au/~tw/school/

PPS are you looking for an open access publisher

http://www.AiAccess.org/