Scheduling
Lecture 1: Introduction

Loris Marchal

CNRS
INRIA GRAAL project-team
Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, France

January 28, 2009

Welcome

- Who am I ?: CNRS researcher at LIP, used to be a student of ENS Lyon, and a PhD student at LIP.
- Any information about the class: `loris.marchal@ens-lyon.fr` + website (google me)
- Mosty on the board, but slides will be available on the website
- Outline of the class:
  - today: introduction to scheduling
  - after: study of particular scheduling problems
  - focus on scheduling for large-scale platforms
  - at the end: more on-going research stuff
  - evaluations: research papers
- Slides and documentation source:
  - myself (a little bit)
  - Frédéric Vivien `http://graal.ens-lyon.fr/~fvivien/`
  - EPIT school `http://graal.ens-lyon.fr/~fvivien/EPIT2007.html`, forthcoming book

Outline

- Vocabulary
  - Basic complex scheduling problem
  - Processor scheduling
- Graham classification
- Types of results: easy and hard problems
- Some scheduling problems
  - $1||\sum w_i C_i$, polynomial (Smith-ratio)
  - $P|prec|C_{\max}$, NP-hard, Graham 2-approx
  - $1||\sum U_i$, Moore-Hodgson algorithm
- Other types of scheduling problems ?

Outline

Vocabulary
- Basic complex scheduling problem
- Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems
- $1||\sum w_i C_i$, polynomial (Smith-ratio)
- $P|prec|C_{\max}$, NP-hard, Graham 2-approx
- $1||\sum U_i$, Moore-Hodgson algorithm

Other types of scheduling problems ?

What is scheduling ?

- allocation of limited resources to activities over time
- activities: tasks in computer environment, steps of a construction project, operations in a production process, lectures at the University, etc.
- resources: processors, workers, machines, lecturers, rooms, etc.

Many variations on the model, on the resource/activity interaction and on the objective.

Outline

Vocabulary

- Basic complex scheduling problem
- Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems

- $1||\sum w_i C_i$, polynomial (Smith-ratio)
- $P|prec|C_{\max}$, NP-hard, Graham 2-approx
- $1||\sum U_i$, Moore-Hodgson algorithm

Other types of scheduling problems ?

Basic complex scheduling problem

Resource-constrained project scheduling problem:
Schedule activities over time on scarce resources, such that some constraints are satisfied and some objective function is optimized

- n activities (jobs) $j = 1, \ldots n$,
- r renewable resources $i = 1, \ldots, r$
- R_k: amounts of resource k available at any time
- activity j processed for p_j time units,
  using an amount $r_{j,k}$ of resource k
- Integer numbers
- $R_k = 1 \Leftrightarrow$ resource disjunctive (0 $\Leftrightarrow$ resource cumulative)
- Precedence constraints: $i \rightarrow j$
  - j cannot start before i is completed
- Precedence constraint graph: DAG (directed acyclic graph)

Basic complex scheduling problem

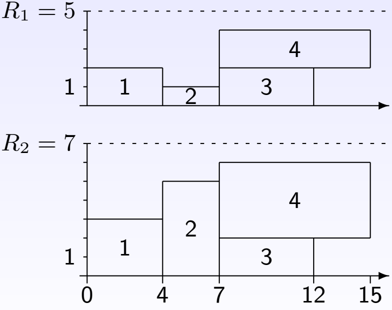
Objective: find starting time S_j for each activity, such that

- At each time, the total resource demand is less than (or equal to) the resource availability for each resource
- Precedence constraints are satisfied: $S_i + p_i \leq S_j$ if $i \rightarrow j$
- conceptMakespan $C_{\max} = \max C_j$ is minimized, with $C_j = S_j + p_j$

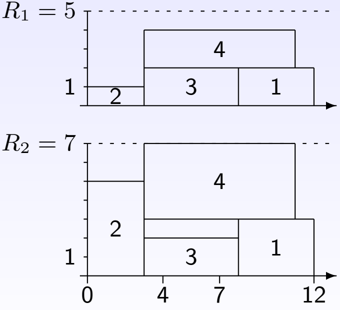
- $C_j = S_j + p_j$ implies no preemption (activity splitting).
- Dummy starting activity 0 + dummy termination activity $n+1$, with $S_0 = 0$ and $C_{\max} = C_{n+1}$
- Without preemption, vector S defines a schedule
- S is called feasible if all resource and precedence constraints are fulfilled

Basic complex scheduling problem – Example

j	1	2	3	4
p_j	4	3	5	8
$r_{j,1}$	2	1	2	2
$r_{j,2}$	3	5	2	4



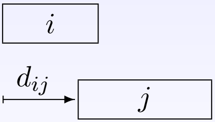
(a) A feasible schedule



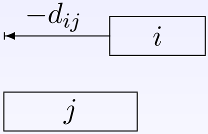
(b) An optimal schedule

Generalization of precedence relations

- Generalize precedence relation: $S_i + d_{i,j} \leq S_j$
  - different cases + models all relations between start/finish times
- Release times r_j and deadlines d_j: $S_0 + r_j \leq S_j$, $S_j - (d_j - p_j) \leq S_0$
- Communication delays $c_{i,j}$



(a) positive time-lag



(b) negative time-lag

Other objectives

- Total flow time: $\sum_{j=1}^{n} C_j$
- Weighted (total) flow time: $\sum_{j=1}^{n} w_j C_j$
- With due dates d_j:
  - lateness: $L_j = C_j - d_j$
  - tardiness: $T_j = \max\{0, C_j - d_j\}$
  - unit penalty: $U_j = 0$ if $C_j \leq d_j$, 1 otherwise
- maximum lateness: $L_{mathrmmax} = \max L_j$
- total tardiness $\sum T_j$
- total weighted tardiness $\sum w_j T_j$
- number of late activities $\sum U_j$
- weighted number of late activities $\sum w_j U_j$

Outline

Vocabulary
- Basic complex scheduling problem
- Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems
- $1||\sum w_i C_i$, polynomial (Smith-ratio)
- $P|prec|C_{\max}$, NP-hard, Graham 2-approx
- $1||\sum U_i$, Moore-Hodgson algorithm

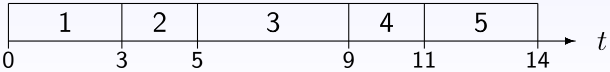
Other types of scheduling problems ?

Single processor scheduling

- n jobs $J_1, \ldots, J_n$ with processing times p_j
- 1 processor
- precedence constraint

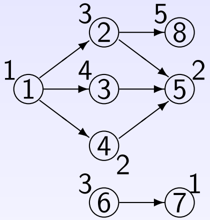
Example:

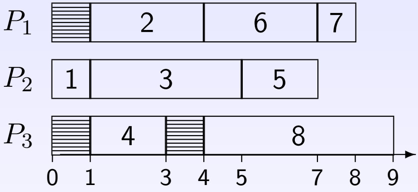
- 5 jobs, with processing times $3, 2, 4, 2, 5$
- precedence constraints: $1 \to 3, 2 \to 4, 4 \to 5$



Parellel processor scheduling

- m identical processors $P_1, \ldots, P_m$
- all tasks have the same processing time P_j on all processors
- $\Leftrightarrow$ RCPSP with one machine, $R_1 = m$





Variants:

- unrelated processors: $p_{j,k}$ depends on P_k and J_j
- uniform processors: $p_{j,k} = P_j / s_k$, s_k is the speed of processor P_k

Multi-processor task scheduling

- jobs $J_1, \dots J_n$
- processors $P_1, \dots, P_m$
- each job J_j has processing time p_j, and makes use of a subset of processor $\mu_j \subseteq \{P_1, \dots, P_m\}$
- $+$ precedence constraints

another variant: identical processors, and each job J_j makes us of any subset of $size_j$ processors

Shop scheduling

Jobs consist in several operations, to be processed on different resources.

General shop scheduling problem:

- jobs $J_1, \dots J_n$
- processors $P_1, \dots, P_m$
- J_j consists in n_j operation $O_{1,j}, \dots, O_{n_j,j}$
- two operations of the same job cannot be processed at the same time
- a processor can process one operation at a time
- operations $O_{i,j}$ has processing time $p_{i,j}$ and makes use of processor $\mu_{i,j}$
- arbitrary precedence pattern

Shop scheduling

job-shop scheduling problem:

- chain of precedence constraints:

$$O_{1,j} \to O_{2,j} \to \cdots \to O_{n_j,j}$$

flow-shop scheduling problem:

- special job-shop scheduling problem
- $n_j = m$ for all j, and $\mu_{i,j} = P_i$ for all i, j:
  operation $O_{i,j}$ must be processed by P_i

open-shop scheduling problem

- like a flow-shop, but no precedence constraints

Outline

Graham notation

Classes of scheduling problems can be specified in terms of the three-field classification $\alpha|\beta|\gamma$ where

- α specifies the machine environment,
- β specifies the job characteristics,
- γ and describes the objective function(s).

Graham notation – machines

To describe the machine environment the following symbols are used:

- 1 single machine
- P parallel identical
- Q uniform machines
- R unrelated machines
- MPM multi-purpose machines
- J job-shop
- F flow-shop
- O open-shop

The above symbols are used if the number of machines is part of the input. If the number of machines is fixed to m we write Pm, Qm, Rm, MPMm, Jm, Fm, Om.

Graham notation – Job characteristics

- pmtn preemption
- r_j release times
- d_j deadlines
- $p_j = 1$ or $p_j = p$ or $p_j \in 1, 2$: restricted processing times
- prec : arbitrary precedence constraints
- intree: (outtree) intree (or outtree) precedences
- chains: chain precedences
- series-parallel: a series- parallel precedence graph

Graham notation – Objectives

- makespan $C_{\max}$
- maximum lateness: $L_{\max}$
- mean flow-time $\sum C_i$
- mean weighted flow-time $\sum w_i C_i$

- sum of tardiness $\sum T_j$
- sum of weighted tardiness $\sum w_j T_j$

- number of late jobs $\sum U_j$
- weighted number of late activities $\sum w_j U_j$

(lateness: $L_j = C_j - d_j$, tardiness: $T_j = \max\{0, C_j - d_j\}$, unit penalty: $U_j = 0$ if $C_j \leq d_j$, 1 otherwise)

Graham notation – Examples

$$1|r_j; pmtn|L_{\max}$$

$$P2|p_j = p; r_j; tree|C_{\max}$$

$$Jm|p_{i,j} = 1|\sum w_j U_j$$

Outline

Vocabulary

Basic complex scheduling problem

Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems

$1||\sum w_i C_i$, polynomial (Smith-ratio)

$P|prec|C_{\max}$, NP-hard, Graham 2-approx

$1||\sum U_i$, Moore-Hodgson algorithm

Other types of scheduling problems ?

Polynomial problems

- a solution to a scheduling problem is a function h:
  - x is the input (parameters)
  - $h(x)$ is the solution (starting times, etc.)
- $|x|$ defined as the length of some encoding of x
  - usually, binary encoding: integer a encoded in $\log_2 a$ bits
- Complexity of an algorithm computing $h(x)$ for all x: running time
- An algorithm is called polynomial, if it computes $h(x)$ for all x it at most $O(p(|x|)$ steps, where P is a polynomial
- A problem is called polynomial if it can be solved by a polynomial algorithm

Pseudo-polynomial problems

If we replace the binary encoding by an unary encoding (integer a encoded with size a): we can solve more difficult problems in time polynomial with $|x|$.

- An algorithm is pseudo-polynomial if it solves the problem for all x with a number of steps at most $O(p(|x|)$ steps, where P is a polynomial and $|x|$ the size of of an unary encoding of x.

Example:
An algorithm for a scheduling problem, whose running time is $O(p_j)$ is pseudo-polynomial.

P and NP

- Decision problems
- To each optimization problem, we can define a decision problem
- P: class of polynomially solvable decision problems
- NP: class of polynomially *checkable* decision problems
  for each "yes"-answer, a certificate exists which can be used to check the answer in polynomial time
- Decisions problems of scheduling problems belongs to NP
- $P \subseteq NP$. $P \stackrel{?}{=} NP$ still open

NP-complete problems

- a decision problem Q is NP-complete if all problems in NP can be polynomially reduced to Q
- if any single NP-complete decision problem Q could be solved in polynomial time then we would have P = NP.
- To prove that a problem is NP-complete: reduction to a well-known NP-complete problem

- Weakly NP-complete (or binary NP-complete): strongly depends on the binary coding of the input. If unary coding is used, the problem might become polynomial (pseudo-polynomial).
  - 2-Partition vs 3-Partition

How to solve NP-complete problems ?

Exact methods:

- Mixed integer linear programming
- Dynamic programming
- Branch and bound methods

(usually limited to small instances)

Approximate methods:

- Heuristics (no guarantee)
- Approximation algorithms

Approximation algorithms

Consider a minimization problem. On a given instance x,
$f(x)$: value of the objective in the solution given by the algorithm
$f^*(x)$: optimal value of the objective

An algorithm is a ρ-approximation if for any instance x ,
$f(x) \leq \rho \times f^*(x)$

APX class: problems for which there exists a polynomial-time ρ-approximation algorithm, for some $\rho > 0$

An algorithm is a PTAS (Polynomial Time Approximation Scheme) if for any instance x and any $\epsilon > 0$, the algorithm computes a solution $f(x)$ with $f(x) \leq (1 + \epsilon) \times f^*(x)$ in time polynomial in the problem size.

An algorithm is a FPTAS (Fully Polynomial Time Approximation Scheme) if for any instance and any $\epsilon > 0$, it produces a solution $f(x)$ such that $f(x) \leq (1 + \epsilon) \times f^*(x)$, in time polynomial in the problem size and in 1ϵ.

Outline

Vocabulary

- Basic complex scheduling problem
- Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems

- $1||\sum w_i C_i$, polynomial (Smith-ratio)
- $P|prec|C_{\max}$, NP-hard, Graham 2-approx
- $1||\sum U_i$, Moore-Hodgson algorithm

Other types of scheduling problems ?

Outline

Vocabulary
- Basic complex scheduling problem
- Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems
- $1||\sum w_i C_i$, polynomial (Smith-ratio)
- $P|prec|C_{\max}$, NP-hard, Graham 2-approx
- $1||\sum U_i$, Moore-Hodgson algorithm

Other types of scheduling problems ?

$1||\sum w_i C_i$, or the Smith-ratio

- Objective: weighted sum of completion times
- Intuitions:
  - put high weight first
  - put longer tasks last
- $\Rightarrow$ Order task by non-increasing Smith ratio: $w_1/p_1 \geq w_2/p_2 \geq \cdots \geq w_n/p_n$

Proof:

- Consider a different optimal schedule S
- Let i and j be two consecutive tasks in this schedule such that $w_i/p_i < w_j/p_j$
- contribution of these tasks in S: $S_i = (w_i + w_j)(t + p_i) + w_j p_j$
- contribution of these tasks if switched: $S_j = (w_i + w_j)(t + p_j) + w_i p_i$
- we have $\frac{S_i - S_j}{w_i w_j} = \frac{p_i}{w_i} - \frac{p_j}{w_j}$
  Thus we decrease the objective by switching these tasks.

Outline

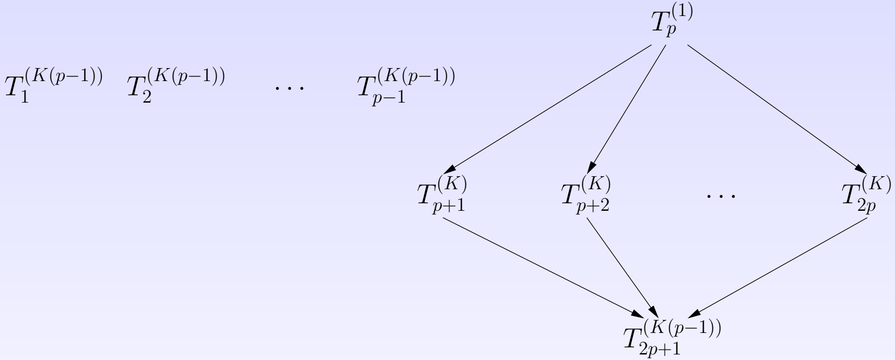
$P|prec|C_{\max}$, Graham 2-approx

- NP-complete
- reduction to 2-partition (or 3-partition, $\longrightarrow$ unary NP-complete)

Graham list scheduling approximation

- Theorem: Any list scheduling heuristic gives a schedule, whose makespan is at most $2 - 1/p$ times the optimal.
- Lemma: there exists a precedence path Ψ such that $\mathrm{Idle} \leq (p-1) \times w(\Psi)$
  - Consider the task with maximum termination time T_1
  - Let t_1 be the last moment (strictly) before $\sigma(T_1)$ when a processor is not active
  - Since a processor is inactive at time t_1, there exists a task T_2, finishing at time t_1, which is an ancestor of T_1 (unless T_1 would be free and scheduled at time t_1 or before)
  - Iterate the process
  - All idle times occur during the processing of these tasks, at most on $p-1$ processors
- Notice that $pC_{\max} = \mathrm{Idle} + \mathrm{Seq}$, with $\mathrm{Seq} = \sum w(T_i)$
- We also have $\mathrm{Seq} \leq pC_{\max}^{\mathrm{opt}}$, thus $C_{\max} \leq \big((p-1) \times w(\Psi)\big) + \big(pC_{\max}^{\mathrm{opt}}\big)$
- We also have $w(\Psi) \leq C_{\max}^{\mathrm{opt}}$, qed.

The approximation bound is tight



$$C_{\max}^{list} = Kp + K(p-1) = K(2p-1)$$
$$C_{\max}^{opt} = 1 + K + K(p-1) = Kp + 1$$

$$\frac{C_{\max}^{list}}{C_{\max}^{opt}} \geqslant \frac{K(2p-1)}{Kp+1} = \frac{2p-1}{p} - \frac{2p-1}{p(Kp+1)} = (2 - \frac{1}{p}) - \epsilon(K),$$

Outline

Vocabulary

- Basic complex scheduling problem
- Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems

- $1||\sum w_i C_i$, polynomial (Smith-ratio)
- $P|prec|C_{\max}$, NP-hard, Graham 2-approx
- $1||\sum U_i$, Moore-Hodgson algorithm

Other types of scheduling problems ?

$1||\sum U_i$, Moore-Hodgson algorithm

One machine, minimize the number of late jobs

Example:

job	1	2	3	4	5
d_j	6	7	8	9	11
p_j	4	3	2	5	6

Tasks are sorted by non-decreasing d_i : $d_1 \leq \cdots \leq d_n$

- $A := \emptyset$
- For $i = 1 \ldots n$
  - $If p(A) + p_i \leq d_i$, then $A := A \cup \{i\}$
  - Otherwise,
    - Let j be the longest task in $A \cup \{i\}$
    - $A := A \cup \{i\} - \{j\}$

Optimal solution : $A = \{2, 3, 5\}$

Feasibility

We first prove that the algorithm produces a feasible schedule:

- By induction: if not task is rejected, ok
- Assume that A is feasible, prove that $A \cup \{i\} - \{j\}$ is feasible too
  - all tasks in A before j: no change
  - all tasks in A after j: shorter completion
  - task i: let k be the last task in A: $p(A) \leq d_k$
    since task j is the longest: $p_i \leq p_j$, thus
    $p \cup \{i\} - \{j\} \leq p(A) \leq d_k \leq d_i$ (because tasks are sorted)

Optimality

Assume that there exist an optimal set O different from the set A_f output by the Moore-Hodgson algorithm

- Let j be the first task rejected by the algorithm
- We prove that there exists an optimal solution without j
- We consider the set $A = \{1, \dots, i-1\}$ at the moment when task j is rejected from A, and i the task being added at this moment
- $A + i$ is not feasible, thus O does not contain $\{1, \dots, i\}$
- Let k be a task of $\{1, \dots, i\}$ which is not in O
- Since the algorithm rejects the longest task, $p(O \cup \{k\} - \{j\}) \leq p(O)$, and by the same arguments than before, $O \cup \{k\} - \{j\}$ is feasible
- We can suppress j from the problem instance, without modifying the behavior of the algorithm or the objective

We can repeat this process, until we get the set of tasks scheduled by the algorithm.

Outline

Vocabulary

- Basic complex scheduling problem
- Processor scheduling

Graham classification

Types of results: easy and hard problems

Some scheduling problems

- $1||\sum w_i C_i$, polynomial (Smith-ratio)
- $P|prec|C_{\max}$, NP-hard, Graham 2-approx
- $1||\sum U_i$, Moore-Hodgson algorithm

Other types of scheduling problems ?

Other types of scheduling problems

- Online problems
  - contrarily to offline, information about future jobs is not known in advance
  - competitive ratio: ratio to the optimal offline algorithm
- Distributed scheduling
  - use only local information
- Multi-criteria scheduling
  - several objectives to optimize simultaneously
  - and/or several users, link with game theory
- Cyclic scheduling
  - infinite but regular pattern of tasks