§ 17 Bounding the distribution function of the makespan

Approximate methods and bounds

Simulation
- requires (complete) information about distributions
- difficult to model stochastic dependencies

Bounding the distribution function
- possible with incomplete information
- permits stochastic dependencies
- combinatorial algorithms

↳ here, makes heavy use of arc diagrams

Chain minor

Let N, N' be project networks.

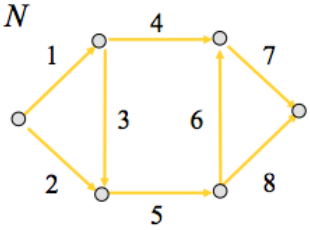
N is a chain minor of N' if

- every job of N is represented by (one or several) copies in N'
- every chain of N is contained in a chain of N'

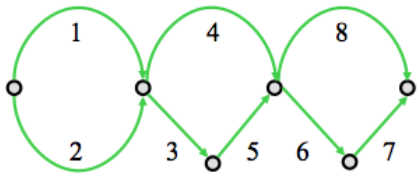
↳ by taking an appropriate copy

Chain minors: an example

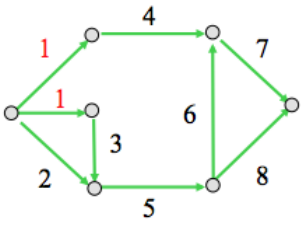
N



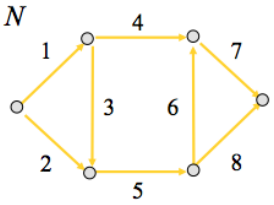
N'

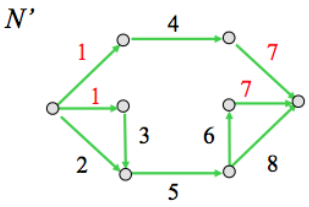


no copies of jobs but proper containment of chains



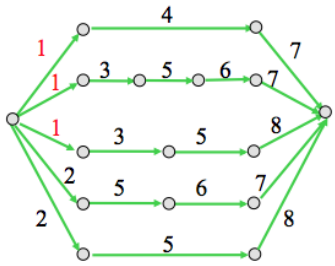
job 1 represented by 2 copies
equality of chains





extreme case:
parallel composition
of all chains

maximal



Bounds based on chain minors

M. & Müller `99

- Let N be a chain minor of N'
  - Give copies of a job from N in N' the same processing time distribution as their original
  - Take all processing time distributions in N' as stochastically independent
- Then the makespan of N is stochastically smaller than the makespan of N'

$Q_{C_{max}(N)} \leq_{st} Q_{C_{max}(N')}$

17.1 Theorem

↳ need definition

Let X_1, X_2 be real-valued random variables with distributions P_1, P_2 and distribution functions F_1, F_2

Then X_1 is stochastically smaller than X_2

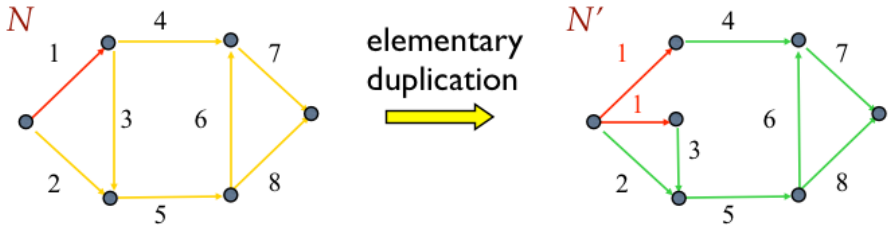
P_1 is -"- than P_2

if $F_1 \geq F_2$ pointwise, i.e. $\text{Prob}\{X_1 \leq t\} \geq \text{Prob}\{X_2 \leq t\}$ $\forall\, t$

$[\Leftrightarrow \int f\,dP_1 \leq \int f\,dP_2 \quad \forall\, f: \mathbb{R}^1 \to \mathbb{R}^1$ non decreasing $]$

Intuition: X_1 is smaller than X_2 with higher probability

Intuition for chain minor bounding principle



copies have same processing time as original ⇒ same makespan

independent variation increases makespan stochastically, i.e.

$\Pr(C_{\max}(N) \le t) \ge \Pr(C_{\max}(N') \le t)$ for all t

Proof: Proper containment makes chains in N' stochastically larger

⇒ is suffices to consider only duplications

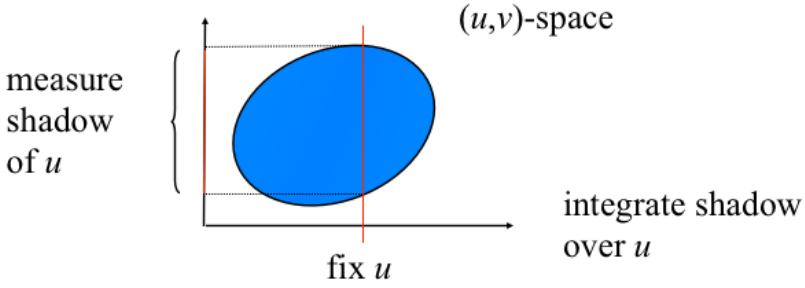
The proof below uses just one duplication in the above example.

It is generic and extends to the general case of more jobs (and one duplication).

The argument can be repeated if there are more than one duplication.

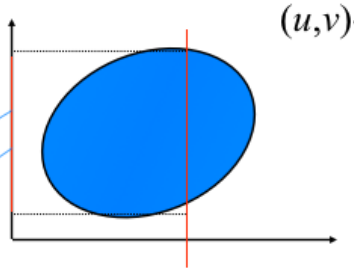
$\Pr\{C_{\max}(N) \le t\}$ $\quad \ge \quad$ $\Pr\{C_{\max}(N') \le t\}$

measure of a set in $\mathbb{R}^8$ $\qquad$ measure of a set in $\mathbb{R}^9$



$$\Pr\{C_{\max}(N) \le t\} \quad \ge \quad \Pr\{C_{\max}(N') \le t\}$$

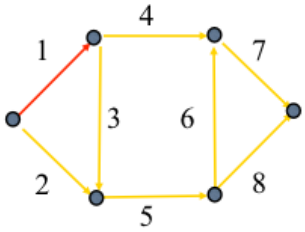
(u,v)-space

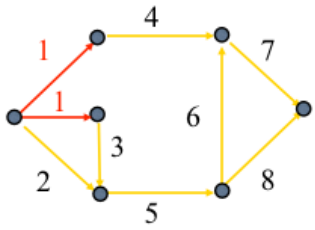


$u = (x_2, \ldots, x_8)$

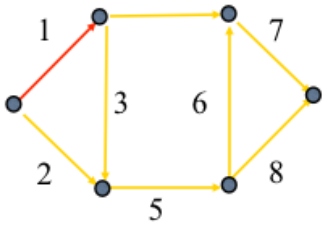
$v = x_1$ $\qquad$ $v = (x_1^1, x_1^2)$

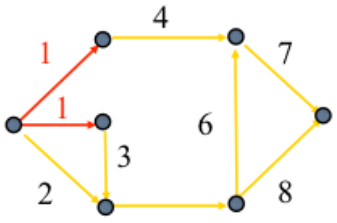
$= \{x_1 \mid C_{\max}(x_1, u) \le t\}$ $\qquad$ $= \{(x_1^1, x_1^2) \mid C'_{\max}(x_1^1, x_1^2, u) \le t\}$





$Q_1(\ |\ ) \quad \ge \quad Q_1 \otimes Q_1(\ \blacksquare\ )$ use independence $\Rightarrow$ product measure





$|\ = [0, a]$ $\qquad$ $\blacksquare = [0, b] \times [0, c]$

Assume $b \ge c$. Then $a \ge c$. $\leftarrow$ see argument below

$$\begin{aligned} Q_1(\ |\ ) &= Q_1([0, a]) \\ &\ge Q_1([0, c]) \\ &\ge Q_1([0, b])\, Q_1([0, c]) \\ &= Q_1 \otimes Q_1([0, b] \times [0, c]) \end{aligned} \quad = Q_1 \otimes Q_1(\ \blacksquare\ )$$

Arguments for the pictures:

Let $(p_1, p_2, \dots, p_8)$ be a vector of processing times for N

Let $(p_1^1, p_1^2, p_2, \dots p_8)$ be the vector after duplication in N'

Set $u := (p_2, \dots, p_8)$, the "joint processing times"

Let $A := \{p_1 \mid C_{max}(p_1, u) \leq t\}$ and $B := \{(p_1^1, p_1^2) \mid C'_{max}(p_1^1, p_1^2, u) \leq t\}$

$\Rightarrow$ A is a 1-dimensional interval, say $A = [0, a]$

B = is a 2-dimensional interval, say $B = [0,b] \times [0,c]$, since the two copies of job 1 can independently achieve their maximum b, c.

Let w.l.o.g. $b \geq c$

Claim: Then $a \geq c$

If not, then $a < c$

$\Rightarrow$ (c, u) gives $C_{max}(c,u) > t$ in N, but $C'_{max}(c, c, u) \leq t$ in N', a contradiction to $C_{max}(c,u) = C'_{max}(c,c,u)$

So $a \geq c$.

Then $Q_1(A) = Q_1([0,a]) \geq Q_1([0,c])$ (*)

Let $\bar{Q}$ be the joint distribution of the two independent copies (X_1^1, X_1^2)

Independence $\Rightarrow$ $\bar{Q} = Q_1 \otimes Q_1$ (product measure)

$\Rightarrow$ $\bar{Q}(B) = \bar{Q}([0,b] \times [0,c]) = Q_1([0,b]) \cdot Q_1([0,c])$ (**)

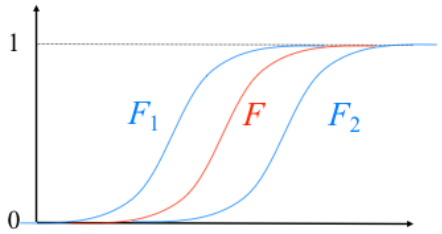
Then $Q_1(A) \overset{(*)}{\geq} Q_1([0,c]) \geq Q_1([0,b]) \cdot Q_1([0,c]) \overset{(**)}{=} \bar{Q}(B)$ □

Using the bounding principle for upper and lower bounds on the distribution function $F_{C_{max}}$ of the makespan C_{max}

Sandwiching a network with minors

Given N, look for special networks N_1, N_2, such that

$$N_1 \subseteq_{minor} N \subseteq_{minor} N_2$$

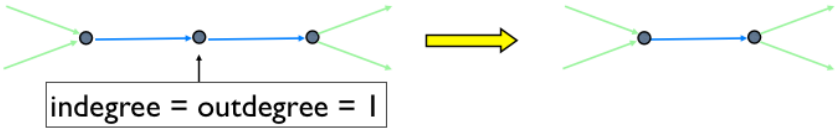


special = easier to calculate the makespan distribution

= series-parallel networks

Series-parallel networks

series reduction



parallel reduction

A network N is series-parallel if it can be reduced by a finite sequence of series and parallel reductions to one job.

Properties of series-parallel networks

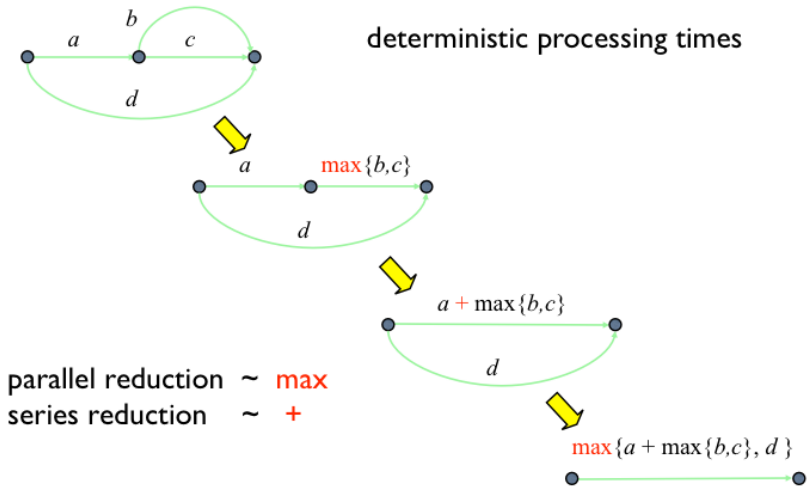
Valdes, Tarjan & Lawler '82

- Series-parallel networks can be recognized in linear time
- A reduction sequence can be constructed in linear time

- Makespan and makespan distribution can be calculated along a reduction sequence

See example below

Makespan computation along a reduction sequence



need only operations + and max in the deterministic case.

These operations have stochastic counterparts for independent stochastic processing times X_j:

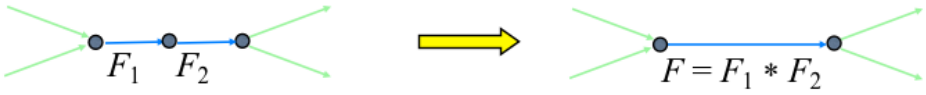
$$F_{\max\{X,Y\}} = F_X \cdot F_Y \qquad \text{product}$$

$$F_{X+Y} = F_X * F_Y \qquad \text{convolution}$$

Makespan distribution of a series-parallel network

parallel reduction ~ product of distribution functions

series reduction ~ convolution of distribution functions



$$F_1 * F_2(t) = \int_0^t F_1(t-x) f_2(x) dx \qquad f_2 = \text{density of } F_2$$

Evaluating the distribution function of a series-parallel network is indeed easier:
(though still NP-hard in the weak sense)

Complexity of evaluating series-parallel networks

M. & Müller `99

- DF is NP-hard (in the weak sense) for series-parallel networks with 2-point processing time distributions

But:

- The reduction sequence algorithm computes the makespan distribution function in a number of steps polynomial in
  $M = \max$ #(values of the makespan)
  along the sequence

17.2 Theorem

Proof of Theorem 17.2 [DF-SP $\hat{=}$ DF for series-parallel networks]

(1) The 2-stage version of DF-SP is NP-hard in the weak sense

Proof by reduction from PARTITION

Let $a_1, \dots, a_n, b \in \mathbb{N}$ be an instance I of partition

Construct an instance I' of DF-SP as follows:

jobs $1, \dots, n$ in a chain = network N

job j has processing times $X_j = \begin{cases} 0 & \text{with prob. } 1/2 \\ a_j & \text{-"- } 1/2 \end{cases}$

Assume there is a polynomial algorithm A that computes $F_N(t)$ for arbitrary t

Then: use A to compute $F_N(t)$ for $t_1 = b$ and $t_2 = b-1$

If I is a yes-instance, then $\exists\ I \subseteq \{1, \dots, n\}$ with $\sum_{i \in I} a_i = b$

$\Leftrightarrow \{C^N_{max} = b\}$ has positive probability

$\Leftrightarrow F_N$ has a jump at $t = b$

$\Leftrightarrow$ the values $F_N(b)$ and $F_N(b-1)$ differ

So can decide in polynomial time if I is a yes-instance of PARTITION by computing $F_N(b)$ and $F_N(b-1)$.

Note: the reduction is not a polynomial reduction but a Turing reduction, in which an algorithm may be used several times on I'

Still it shows that PARTITION is solvable in polynomial time if DF-SP can be solved in polynomial time.

(2) DF-SP can be solved in $O(N^2 \cdot n)$ time for discrete distributions, where

N = max # values of C_{max} along a reduction sequence

Assume that F_j of job j is represented by a list $t_1 < t_2 < \dots < t_\ell$ of its jumps with values $F_j(t_1), \dots, F_j(t_\ell)$

$\Rightarrow$ $F_j(t)$ can be calculated by a scan through the list in $O(l)$ time

assumption $\Rightarrow$ $l \leq N$ at every stage along a reduction sequence

$\Rightarrow$ computing $F_i \cdot F_j$ takes $O(N)$ time

computing $F_i * F_j$ takes $O(N^2)$ time

length of reduction sequence $= n-1$ $\Rightarrow$ $O(N^2 n)$ in total $\square$

Specific Bounds

(A) The bound of Dodin '85

Input: Stochastic project network N

Output: An upper bound on the makespan distribution of N

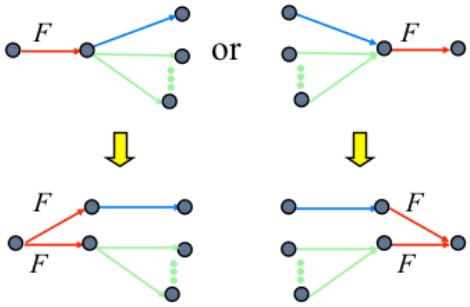
7.3 Algorithm

while N has more than one arc **do**

if a series reduction is possible then apply one

else if a parallel reduction is possible **then** apply one

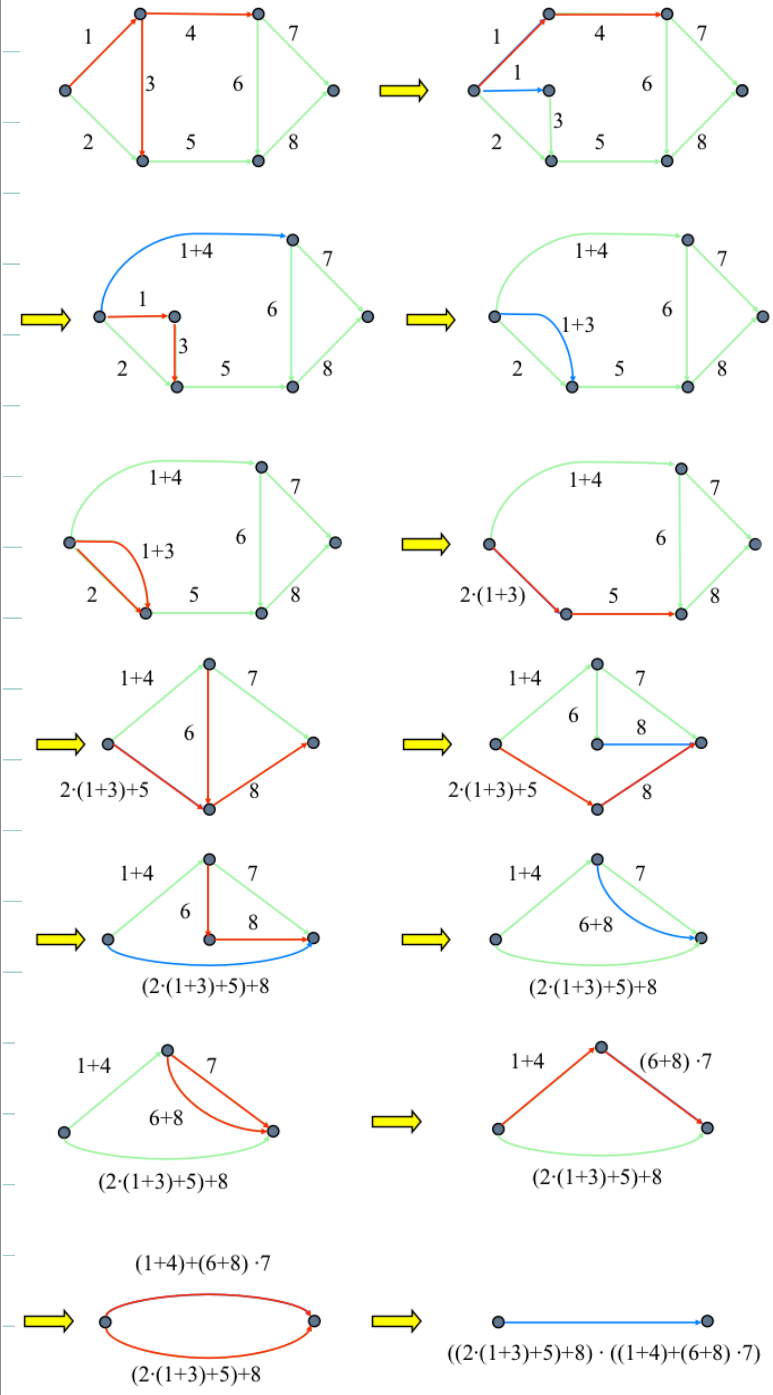
else find



Exercise: show that Dodin's algorithm always terminates with a single job

The distribution of that job is an upper bound for $F^N_{C_{max}}$

An example of Dodin's algorithm



resulting distribution function:

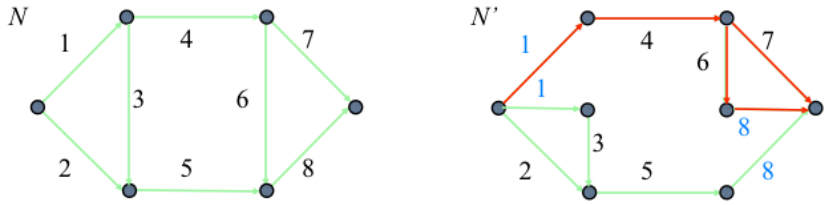
$$\left(\left(F_2 \cdot \left(F_1 * F_3\right) * F_5\right) * F_8\right) \cdot \left(\left(F_1 * F_4\right) * \left(F_6 * F_8\right) \cdot F_7\right)$$

Structure of Dodin's bound

17.4 Theorem

Dodin's bound for network N is
- exact if N is series-parallel
- the distribution function of a series-parallel network N' that contains N as a chain-minor

$$((F_2 \cdot (F_1 * F_3) * F_5) * F_8) \cdot ((F_1 * F_4) * (F_6 * F_8) \cdot F_7)$$

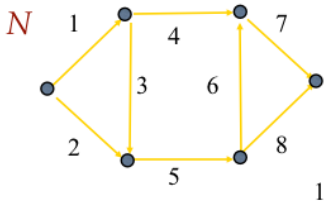


Proof: Clear from the bounding principle □

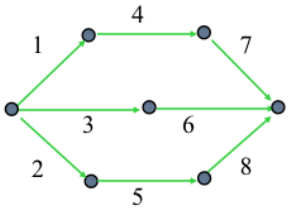
(B)

The bounds of Spelde '76

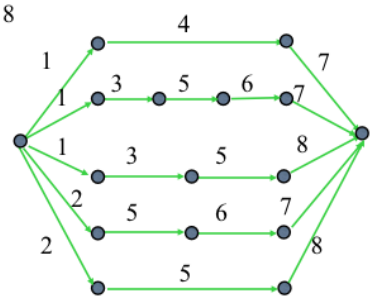
- uses special series-parallel networks (parallel chains)
- yields upper and lower bounds



disjoint chains
⇒ lower bound



all chains
⇒ upper bound



clear from the minor property

Distribution-free evaluation of Spelde's bounds

Large networks ⇒

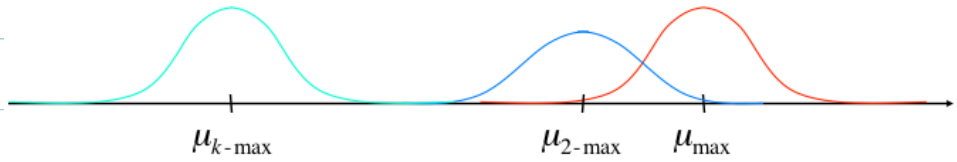
- chain length ≈ normally distributed
  - ⇒ $\mu = \sum \mu_j$ and $\sigma^2 = \sum \sigma_j^2$ along a chain
- ⇒ per job only μ_j and σ_j^2 required

μ_j = mean, σ_j^2 = variance } of job j

Problem: #chains may be exponential

Remedie: Consider only relevant chains

Relevant chains



$\Pr(Y_k \geq Y_1) \leq \varepsilon$ $\quad Y_i$ = length of i-longest chain w.r.t. mean processing times μ_j

Apply k-longest path algorithms to determine k see (*), (**) below

Return the product of the k–1 normal distribution functions F_{Y_i}

Excellent and fast bounds in practice

Special case: PERT, considers only Y_1

(*) Compute 1st, 2nd, ..., k-th longest chain w.r.t. to μ_j until Prob $\{Y_k \geq Y_1\} \leq \varepsilon$ for a specified ε

↑ ↑

length of chain as normal distribution !

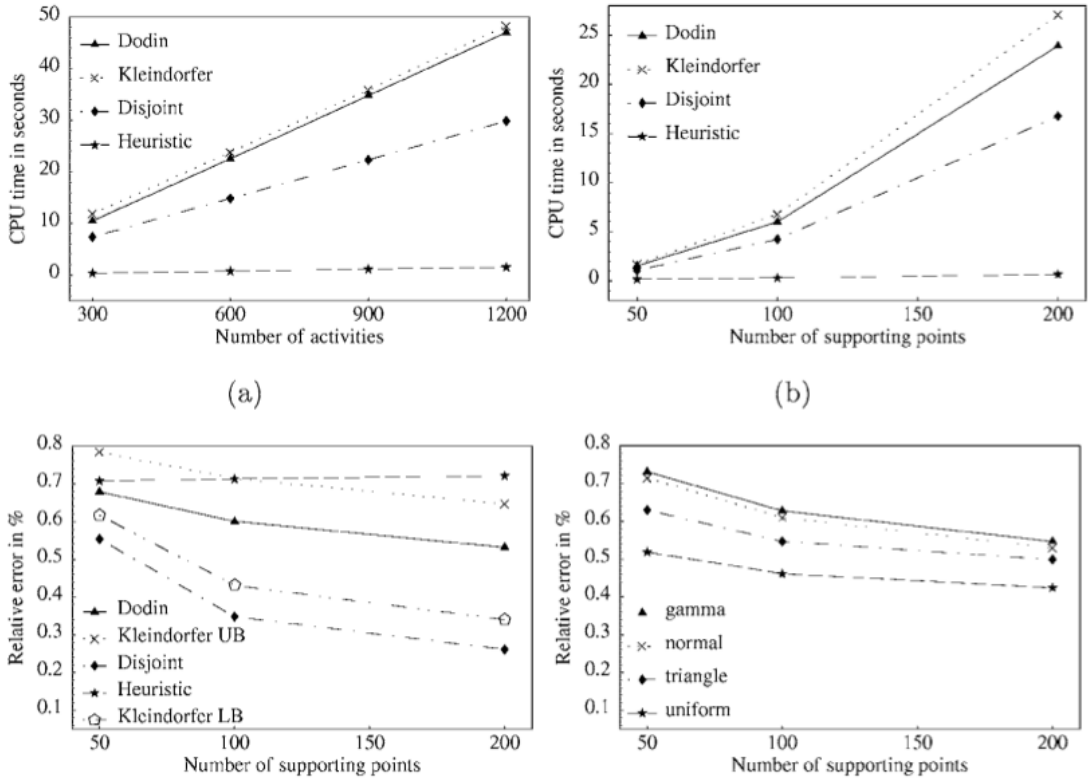
Then $F := F_1 \cdot F_2 \cdot \ldots \cdot F_k$ is an upper bound for F^N_{Cmax}

(**) To compute the k-longest paths one may use a k-shortest path algorithm since N is acyclic

Such algorithms exist and run in $O(kn(m + n\log n))$

see www.ics.uci.edu/~eppstein/bibs/kpath.bib

Computational experience with Dodin, Spelde and other bounds



Heuristic = Spelde distribution free

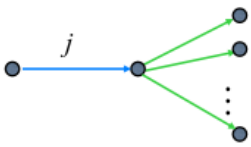
Number of supporting points = representation of discrete distribution functions

Relative error $\hat{=}$ in comparison to a very precise simulation

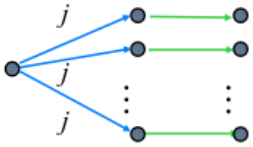
The "distance" of a partial order to a series-parallel order determines the quality of the bounds e.g. in Dodin's algorithm.

Appropriate distance measures have been explored in combinatorics

Measuring the distance to series-parallel networks

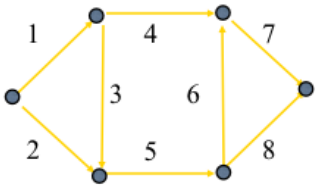


node reduction ⇨

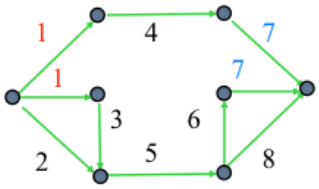


and the "dual" picture as with Dodin

Measure 1: # node reductions



path expression ⇨



$(1+4+7)\cdot((1+3)\cdot 2+5+(6+7)\cdot 8)$

Measure 2: # duplicated subexpressions

Which measure is better?

- [Bein, Kamburowski & Stallman '92]
  - Measure 1 ≥ Measure 2
  - Measure 1 can be determined in polynomial time
- [Naumann '94]
  - Measure 1 = Measure 2

Exercises

17.1 Consider series-parallel partial orders and their comparability graphs

An undirected graph is called a complement-reducible graph or simply cograph if every induced subgraph H of G has the property that H or its complementary graph $\overline{H}$ is disconnected.

Show that G is a cograph iff it is the comparability graph of a series-parallel partial order

17.2* Let G be a finite undirected graph. Show

G is a cograph $\Leftrightarrow$ G does not contain as induced subgraph.

called P_4

17.3 Let G be a finite partial order. Show

G is series-parallel $\Leftrightarrow$ G does not contain  as induced suborder

Hasse diagram, called N

17.4 Show that Dodin's algorithm always terminates