An Efficient Algorithm for Partial Order Production


Jean Cardinal
ULB/CS


Samuel Fiorini
ULB/Math


Gwenaël Joret
ULB/CS


Raphaël Jungers
UCL/INMA


Ian Munro
Waterloo/CS

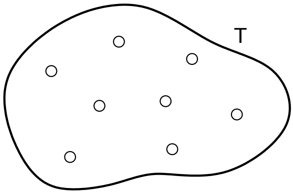
Sorting by Comparisons

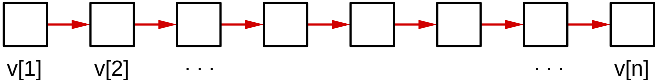
Input: a set T of size n, totally ordered by $\leqslant$

Goal: place the elements of T in a vector v in such a way that

$$v[1] \leqslant_T v[2] \leqslant_T \cdots \leqslant_T v[n]$$

after asking a min number of questions of the form "is $t \leqslant_T t'$?"





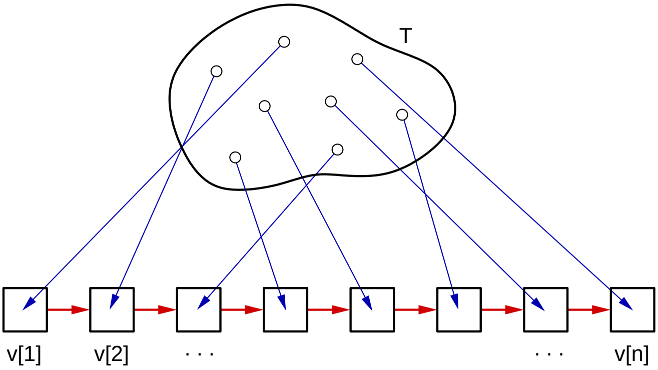
Sorting by Comparisons

Input: a set T of size n, totally ordered by $\leqslant$

Goal: place the elements of T in a vector v in such a way that

$$v[1] \leqslant_T v[2] \leqslant_T \cdots \leqslant_T v[n]$$

after asking a min number of questions of the form "is $t \leqslant_T t'$?"



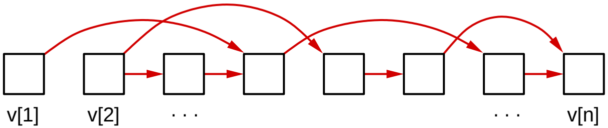
Partial Order Production ("Partial Sorting")

Input: a set T of size n, totally ordered by $\leqslant_T$
a partial order $\leqslant_P$ on the set of positions $[n]$

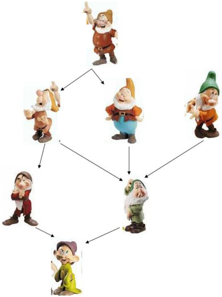
Goal: place the elements of T in a vector v in such a way that

$$v[i] \leqslant_T v[j] \quad \text{whenever } i \leqslant_P j$$

after asking a min number of questions of the form "is $t \leqslant_T t'$?"



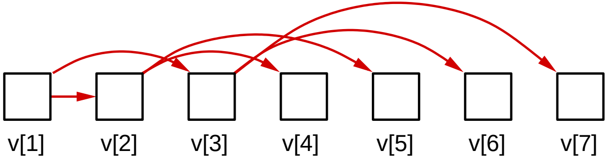

HARDWOOD
LUMP
CHARCOAL



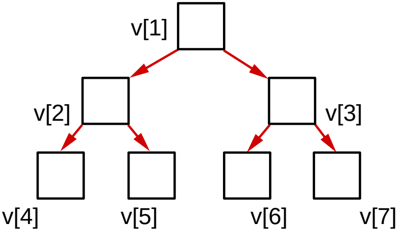

HARDWOOD
LUMP
CHARCOAL

Particular Cases (1/2)

Heap Construction

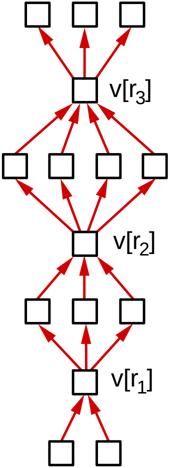


or



Particular Cases (2/2)

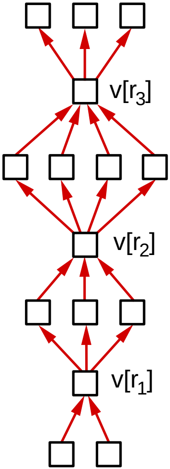
Multiple Selection



Find the elements of rank $r_1, r_2, \ldots, r_k$

Particular Cases (2/2)

Multiple Selection

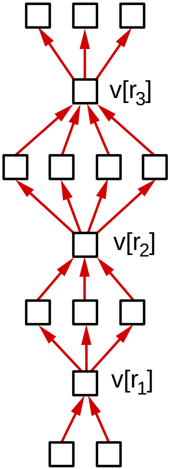


Find the elements of rank $r_1, r_2, \ldots, r_k$

Target poset $P := ([n], \leqslant_P)$ is a weak order

Particular Cases (2/2)

Multiple Selection



Find the elements of rank $r_1, r_2, \ldots, r_k$

Target poset $P := ([n], \leqslant_P)$ is a weak order

$\exists$ near-optimal algorithm (Kaligosi, Mehlhorn, Munro and Sanders, 05)

Worst Case Lower Bounds

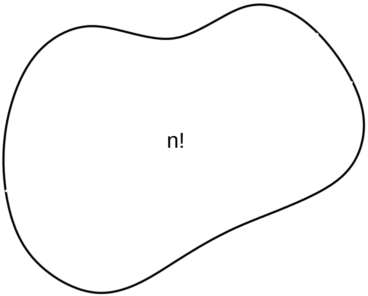
Well known fact. *For Sorting by Comparisons:*

$$\text{worst case } \#\text{comparisons} \geq \lg n!$$

Fact. (Schönage 76, Aigner 81) *For Partial Order Production:*

$$\text{worst case \#comparisons} \geq \underbrace{\lg n! - \lg e(P)}_{=:\, LB}$$

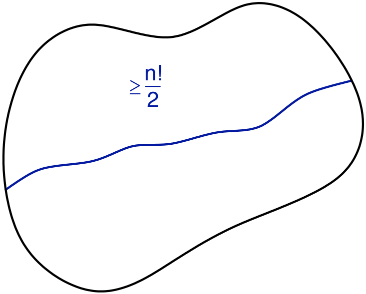
where $e(P) := \#$ linear extensions of P



Fact. (Schönage 76, Aigner 81) *For Partial Order Production:*

$$\text{worst case \#comparisons} \geq \underbrace{\lg n! - \lg e(P)}_{=:\, LB}$$

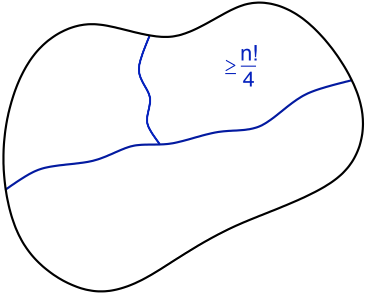
where $e(P) := \#$ linear extensions of P



Fact. (Schönage 76, Aigner 81) *For Partial Order Production:*

$$\text{worst case \#comparisons} \geq \underbrace{\lg n! - \lg e(P)}_{=:\,LB}$$

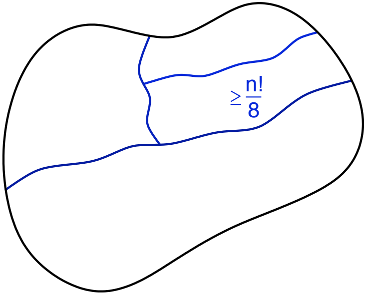
where $e(P) := \#$ linear extensions of P



Fact. (Schönage 76, Aigner 81) *For Partial Order Production:*

$$\text{worst case } \#\text{comparisons} \geq \underbrace{\lg n! - \lg e(P)}_{=:\, LB}$$

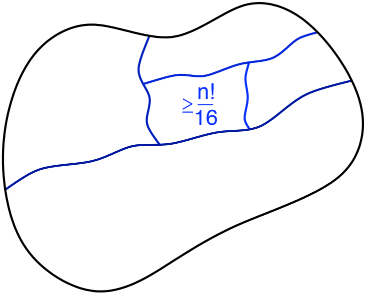
where $e(P) := \#$ linear extensions of P



Fact. (Schönage 76, Aigner 81) *For Partial Order Production:*

$$\text{worst case \#comparisons} \geq \underbrace{\lg n! - \lg e(P)}_{=:\, LB}$$

where $e(P) := \#$ linear extensions of P



$$|\text{leaf set}| \leq e(P) \Longrightarrow \#\text{comparisons} \geq \lg \frac{n!}{e(P)} = LB$$

Problem History

1976 Schönage defined POP problem

Problem History

1976 Schönage defined POP problem

1981 Aigner studied POP problem

Problem History

1976 Schönage defined POP problem

1981 Aigner studied POP problem

1985 Two surveys: Bollobás & Hell, and Saks

Saks conjectured that $\exists$ algorithm for POP problem

s.t. worst case #comparisons $= O(LB) + O(n)$

Problem History

1976 Schönage defined POP problem

1981 Aigner studied POP problem

1985 Two surveys: Bollobás & Hell, and Saks

Saks conjectured that $\exists$ algorithm for POP problem

s.t. worst case #comparisons $= O(LB) + O(n)$

1989 Yao solved Saks' conjecture, stated open problems

Our Result

There exists an algorithm for the POP problem with

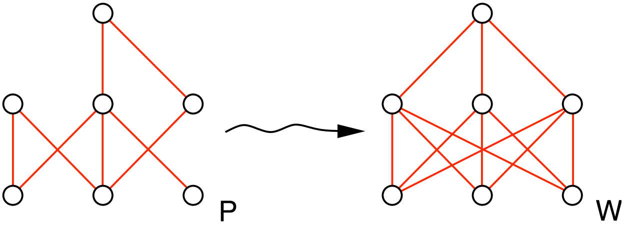
- preprocessing time $O(n^3)$
- worst case $\#$comparisons $= LB + o(LB) + O(n)$

Improvements over Yao's algorithm:

- overall complexity is polynomial
- smaller number of comparisons

A Simple Plan

1. Extend the target poset P to a weak order W
2. Solve the problem for W using Multiple Selection algorithm



Key Tool: the Entropy of a Graph

The entropy of $G = (V, E)$ equals:

$$H(G) := \min_{x \in STAB(G)} -\frac{1}{n} \sum_{v \in V} \lg x_v$$

where $STAB(G) :=$ stable set polytope of G

Key Tool: the Entropy of a Graph

The entropy of $G = (V, E)$ equals:

$$H(G) := \min_{x \in STAB(G)} -\frac{1}{n} \sum_{v \in V} \lg x_v$$

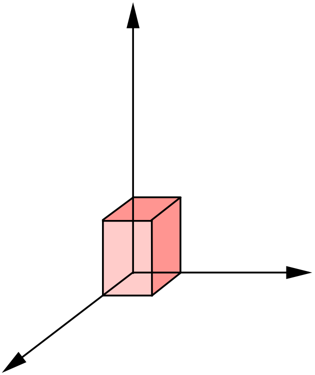
where $STAB(G) :=$ stable set polytope of G

- Introduced in information theory by J. Körner (73)
- Graph invariant with lots of applications (mostly in TCS)
  - lower bounds for perfect hashing
  - lower bounds for monotone Boolean functions
  - sorting under partial information (Kahn and Kim 95)
  - ...

Exercise 1. Prove the following:

Lemma. (Kahn and Kim 95)

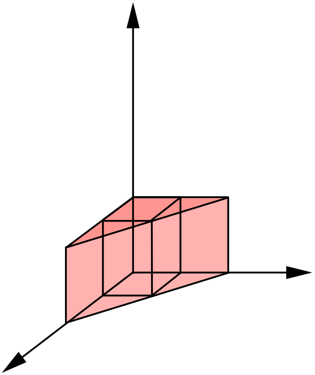
$$\underbrace{-n\,H(G)}_{=\lg Vol(Box)} \leq \lg \mathrm{Vol}(STAB(G)) \leq \underbrace{n \lg n - \lg n! - n\,H(G)}_{=\lg Vol(Simplex)}$$



Exercise 1. Prove the following:

Lemma. (Kahn and Kim 95)

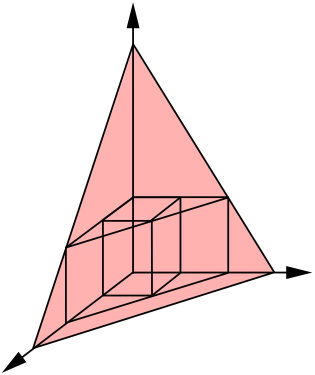
$$\underbrace{-n\,H(G)}_{=\lg Vol(Box)} \leq \lg \mathrm{Vol}(STAB(G)) \leq \underbrace{n \lg n - \lg n! - n\,H(G)}_{=\lg Vol(Simplex)}$$



Exercise 1. Prove the following:

Lemma. (Kahn and Kim 95)

$$\underbrace{-n\,H(G)}_{=\lg Vol(Box)} \leq \lg \mathrm{Vol}(STAB(G)) \leq \underbrace{n \lg n - \lg n! - n\,H(G)}_{=\lg Vol(Simplex)}$$



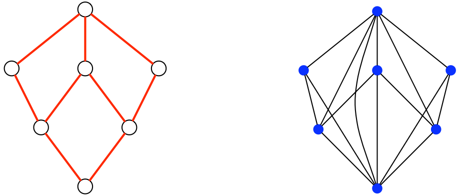
Exercise 2. Prove the min-max relation, for G perfect:

$$H(G) + H(\overline{G}) = \lg n$$

Comparability Graphs and Entropy

$G(P) :=$ comparability graph of target poset P

$H(P) := H(G(P))$



P $\qquad$ $G(P)$

Lemma. (Stanley 86) $\quad \mathrm{Vol}\Big(STAB\big(G(P)\big)\Big) = \dfrac{e(P)}{n!}$

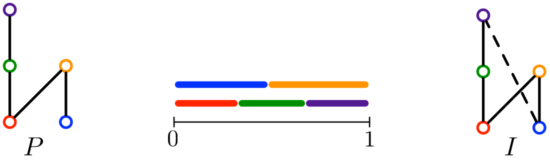
Corollary. $\quad n\,H(P) - n \lg e \le LB \le n\,H(P)$

Entropy of a Poset

A more intuitive definition!? (The magic of extended formulations.)

$\{(y_{v^-}, y_{v^+})\}_{v \in V}$ is *consistent* with P if

- $\forall v \in V$: (y_{v^-}, y_{v^+}) open interval $\subseteq (0, 1)$
- $v \leqslant_P w \implies y_{v^+} \leq y_{w^-}$



$$\boxed{H(P) := \min \left\{ -\frac{1}{n} \sum_{v \in V} \lg x_v \;\middle|\; \exists \{(y_{v^-}, y_{v^+})\}_{v \in V} \text{ consistent with } P \text{ s.t. } x_v = y_{v^+} - y_{v^-} \quad \forall v \in V \right\}}$$

$$H(P) = \tfrac{1}{n} \times \text{“information” in } P$$

Working out the Extended Formulation

Exercise 3. Check that

$$\begin{cases} x_v = y_{v^+} - y_{v^-} & \forall v \in V \\ y_{v^+} \leq y_{w^-} & \forall v \leqslant_P w \\ y_{v^\pm} \geq 0 & \forall v \in V \\ y_{v^\pm} \leq 1 & \forall v \in V \end{cases}$$

is an extended formulation of $STAB(G(P))$, that is, we get $STAB(G(P))$ when we eliminate the $y_{v^\pm}$ variables.

Weak Order Extensions $\to$ Colorings

Observation.

Every weak order extension W of P gives a coloring of $G(P)$

$$\Downarrow$$

Want: "good" coloring of $G(P)$

$$\boxed{W \text{ extends } P \Longrightarrow H(W) \geq H(P)}$$

Weak Order Extensions $\rightarrow$ Colorings

Observation.

Every weak order extension W of P gives a coloring of $G(P)$

$$\Downarrow$$

Want: "good" coloring of $G(P)$

$$\boxed{W \text{ extends } P \Longrightarrow H(W) \geq H(P)}$$

Intuition.

$H(W)$ should be as small as possible

$$\Downarrow$$

Entropy of distribution of class sizes should be as small as possible

Weak Order Extensions → Colorings

Observation.

Every weak order extension W of P gives a coloring of $G(P)$

$$\Downarrow$$

Want: "good" coloring of $G(P)$

$$\boxed{W \text{ extends } P \Longrightarrow H(W) \geq H(P)}$$

Intuition.

$H(W)$ should be as small as possible

$$\Downarrow$$

Entropy of distribution of class sizes should be as small as possible

Exercise 4. For W a weak order with classes of sizes $s_1, \ldots, s_k$:

$$H(W) = H\left(\frac{s_1}{n}, \ldots, \frac{s_k}{n}\right) = -\sum_{i=1}^{k} \frac{s_i}{n} \lg \frac{s_i}{n}$$

Greedy Colorings and Greedy Points

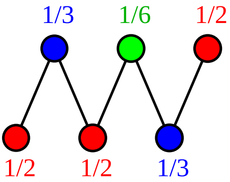
For G = perfect graph

Iterativelу remove a maximum stable set from G

$\rightsquigarrow$ sequence $S_1, S_2, \ldots, S_k$ of stable sets

- Gives greedy coloring (k colors, ith color class $= S_i$)
- Also gives greedy point:

$$\tilde{x} := \sum_{i=1}^{k} \frac{|S_i|}{n} \cdot \chi^{S_i} \quad \in \quad STAB(G)$$



Theorem. *Let G be a perfect graph on n vertices and denote by $\tilde{g}$ the entropy of an arbitrary greedy point $\tilde{x} \in STAB(G)$. Then*

$$\tilde{g} \leq \frac{1}{1-\delta}\left(H(G) + \lg \frac{1}{\delta}\right)$$

for all $\delta > 0$, and in particular (optimizing on δ)

$$\tilde{g} \leq H(G) + \lg H(G) + O(1)$$

Proof idea. Define

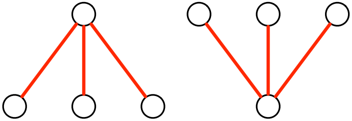
$$z_v := \frac{\delta}{n^{\delta}} \left(\frac{1}{\tilde{x}_v}\right)^{1-\delta}$$

check $z \in STAB(\overline{G})$. Now

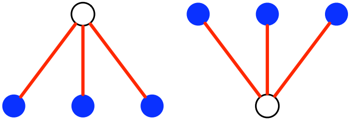
$$H(G) = \lg n - H(\overline{G}) \geq \lg n - \left(-\frac{1}{n}\sum_{v \in V} \lg z_v\right)$$

$\square$

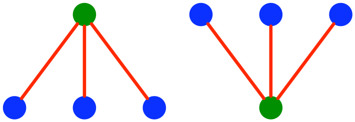
Colorings $\not\to$ Weak Order Extensions



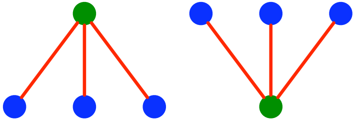
Colorings $\not\to$ Weak Order Extensions



Colorings $\not\to$ Weak Order Extensions



Colorings $\not\to$ Weak Order Extensions

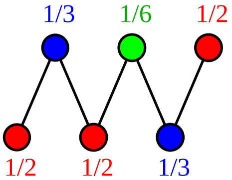


Weak order extensions of $P \to$ colorings of $G(P)$

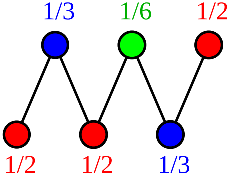
$\not\leftarrow$

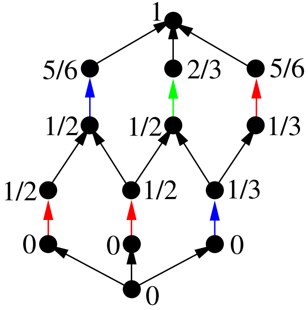
$\Longrightarrow$ need to "uncross" our greedy colorings

Uncrossing a Greedy Coloring

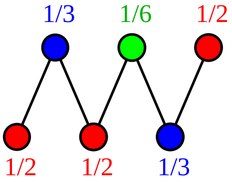


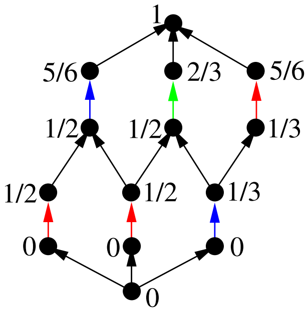
Uncrossing a Greedy Coloring

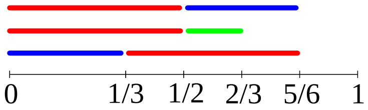




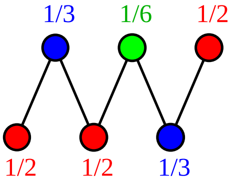
Uncrossing a Greedy Coloring

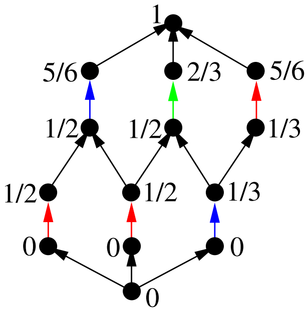


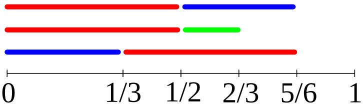


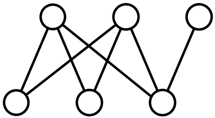


Uncrossing a Greedy Coloring









Main Steps of our Algorithm

1. $P \overset{greedy+DP}{\hookrightarrow} I$

2. $I \overset{greedy}{\hookrightarrow} W$

3. Use Multiple Selection algorithm of Kaligosi *et al.* on W

Theorem. *The algorithm above solves the POP problem, in $O(n^3)$ time, after performing at most*

$$LB + o(LB) + O(n)$$

comparisons

Sorting under partial information (without the ellipsoid algorithm)

Jean Cardinal
ULB

Samuel Fiorini
ULB

Gwenaël Joret
ULB

Raphaël Jungers
UCL/MIT

Ian Munro
Waterloo

Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: #queries

Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: $\#$queries

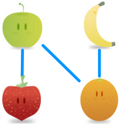
Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: $\#$queries



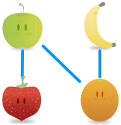
Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: #queries



 OR  ?

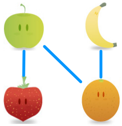
Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: #queries



 OR  ?

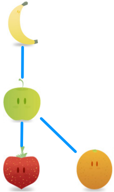
Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: $\#$queries



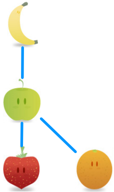
Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: #queries



 OR  ?

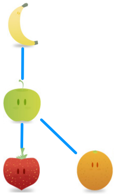
Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

Objective function: #queries



OR

?

Sorting by comparisons under partial information

Input:

- a set $V = \{v_1, \ldots, v_n\}$, totally ordered by an unknown linear order $\leqslant$
- a poset $P = (V, \leqslant_P)$ compatible with $(V, \leqslant)$

Goal: Discover $\leqslant$ by making queries "is $v_i \leqslant v_j$?"

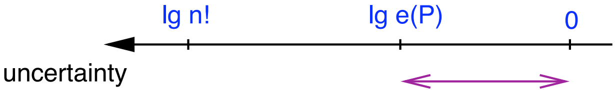
Objective function: #queries

Lower bound on #queries

$e(P) :=$ #linear extensions of P

Every algorithm can be forced to a #queries that is *at least*

$$\lg e(P)$$



$\Rightarrow$ Interested in algorithms that perform close to $\lg e(P)$

Results 

- **Known results**

	#queries	complexity
Fredman 1976	$\lg e(P) + 2n$	super-polynomial
Kahn & Saks 1984	$O(\lg e(P))$	super-polynomial
Kahn & Kim 1992	$O(\lg e(P))$	polynomial (ellipsoid alg.)

Results 

- **Known results**

	#queries	complexity
Fredman 1976	$\lg e(P) + 2n$	super-polynomial
Kahn & Saks 1984	$O(\lg e(P))$	super-polynomial
Kahn & Kim 1992	$O(\lg e(P))$	polynomial (ellipsoid alg.)

- **Our contribution**: two ellipsoid-free algorithms

	#queries	complexity
Algorithm 1	$(1+\varepsilon)\lg e(P) + O_\varepsilon(n) \quad \forall \varepsilon > 0$	$O(n^{2.5})$
Algorithm 2	$O(\lg e(P))$	$O(n^{2.5})$

Bad news and a cure

Computing $e(P)$ is #P-complete — Brightwell & Winkler 1991

Bad news and a cure

Computing $e(P)$ is #P-complete — Brightwell & Winkler 1991

As Kahn & Kim 1992, use the *entropy* $H(\bar{P})$

Why?

- $\boxed{\lg e(P) = \Theta(nH(\bar{P}))}$ — Kahn & Kim 1992
- $H(\bar{P})$ can be "computed" in poly-time using the ellipsoid algorithm

Bounds

"additive" and "multiplicative"

$$nH(\bar{P}) \leq \lg e(P) + n \lg e$$

K&K 1992

Bounds

"additive" and "multiplicative"

$$nH(\bar{P}) \leq \lg e(P) + n \lg e$$ K&K 1992

$$\lg e(P) \leq nH(\bar{P}) \leq c \cdot \lg e(P)$$ K&K 1992

where $c = 1 + 7 \lg e \simeq 11.1$

Bounds

"additive" and "multiplicative"

$$nH(\bar{P}) \leq \lg e(P) + n \lg e$$ K&K 1992

$$\lg e(P) \leq nH(\bar{P}) \leq c \cdot \lg e(P)$$ K&K 1992

where $c = 1 + 7 \lg e \simeq 11.1$

Rmk: Lower bound tight, but not upper bound

- K&K conjectured

$$nH(\bar{P}) \leq (1 + \lg e) \cdot \lg e(P)$$ $(1 + \lg e \simeq 2.44)$

Bounds

"additive" and "multiplicative"

$$nH(\bar{P}) \leq \lg e(P) + n \lg e$$ K&K 1992

$$\lg e(P) \leq nH(\bar{P}) \leq c \cdot \lg e(P)$$ K&K 1992

where $c = 1 + 7 \lg e \simeq 11.1$

Rmk: Lower bound tight, but not upper bound

- K&K conjectured

$$nH(\bar{P}) \leq (1 + \lg e) \cdot \lg e(P)$$ $(1 + \lg e \simeq 2.44)$

- We prove

$$nH(\bar{P}) \leq 2 \cdot \lg e(P)$$

(tight)

Exercise 1. Prove that

$$nH(\bar{P}) \leq 4 \cdot \lg e(P)$$

Hint: induct on n, then on $e(P)$. For $n = 2$, the result is trivial. Assume $n \geq 3$. For $e(P) = 1$, the result is trivial (P is a chain). Now consider an optimal set of intervals for P, and let $a \in V$ be such that the length of the interval for a (i.e., x_a) is maximum. If a is comparable to every $b \in V - a$ then $\lg e(P) = \lg e(P - a)$ and $nH(\bar{P}) = (n-1)H(\overline{P - a})$. Otherwise let $b \in V - a$ be such that the interval for b contains the midpoint of the interval for a. Then find a new poset P' on V with $\lg e(P) - \lg e(P') \geq 1$ and $nH(\bar{P}) - nH(\overline{P'}) \leq 4$, by carefully modifying the intervals for a and b.

K&K's algorithm

Key lemma:

$\exists$ incomparable pair a, b s.t.

$$\max\left\{nH(\bar{P}(a<b)), nH(\bar{P}(a>b))\right\} \leq nH(\bar{P}) - c,$$

where $c \simeq 0.2$

Algorithm:

1. Repeat:
   1.1 Compute $H(P)$ and optimal solution x^*
   1.2 Find good incomparable pair a, b using x^*
   1.3 Compare a and b
   1.4 Update P

$\#\text{steps} = O(nH(\bar{P})) = O(\lg e(P))$

Algorithms

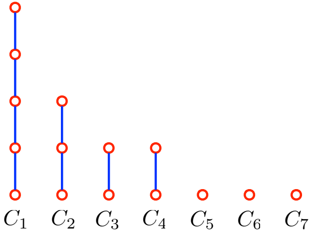
	#queries		complexity
Algorithm 1	$(1+\varepsilon)\lg e(P) + O_\varepsilon(n)$	$\forall \varepsilon > 0$	$O(n^{2.5})$
Algorithm 2	$O(\lg e(P))$		$O(n^{2.5})$

Algorithm 1: greedy + merge sort

Algorithm 2: greedy + "cautious" merge sort

Greedy

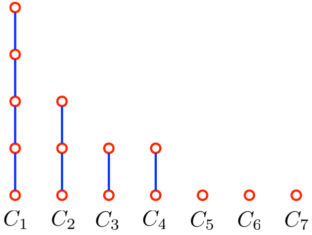
Greedy chain decomposition of $P \quad \rightarrow \quad U := C_1 \cup \cdots \cup C_k$



$$H(\bar{U}) = \lg n - H(U) = \sum_{i=1}^{k} -\frac{|C_i|}{n} \lg \frac{|C_i|}{n}$$

Greedy

Greedy chain decomposition of P $\quad \rightarrow \quad$ $U := C_1 \cup \cdots \cup C_k$



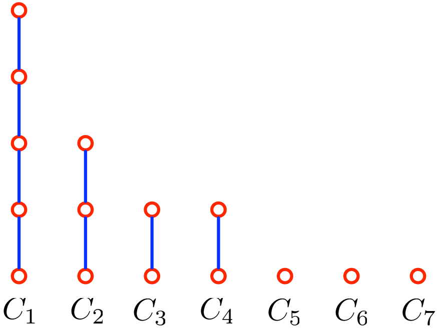
$$H(\bar{U}) = \lg n - H(U) = \sum_{i=1}^{k} -\frac{|C_i|}{n} \lg \frac{|C_i|}{n}$$

From perfectness of incomparability graph of P:

$$\boxed{H(\bar{U}) \leq (1+\varepsilon)H(\bar{P}) + O_\varepsilon(1) \qquad \forall \varepsilon > 0}$$

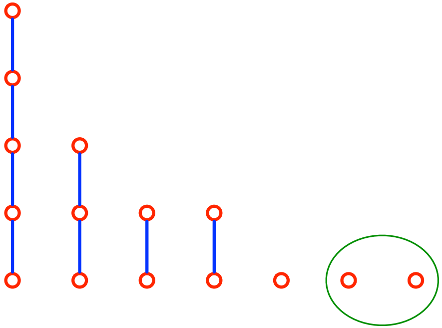
Algorithm 1

1. Compute greedy chain decomposition of P
2. Iteratively merge two smallest chains



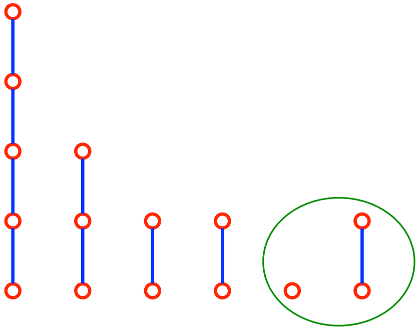
Algorithm 1

1. Compute greedy chain decomposition of P
2. Iteratively merge two smallest chains



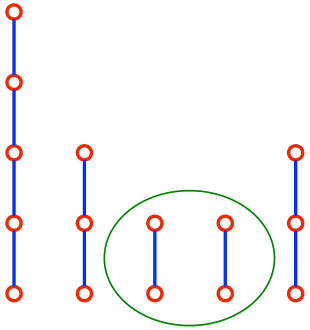
Algorithm 1

1. Compute greedy chain decomposition of P
2. Iteratively merge two smallest chains



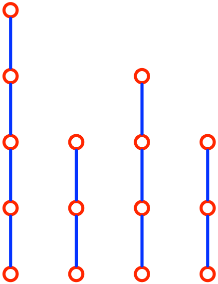
Algorithm 1

1. Compute greedy chain decomposition of P
2. Iteratively merge two smallest chains



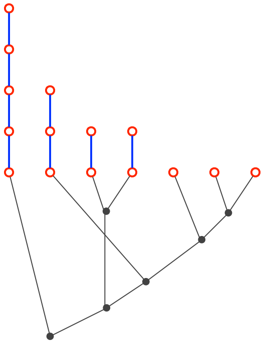
Algorithm 1

1. Compute greedy chain decomposition of P
2. Iteratively merge two smallest chains



ETC.

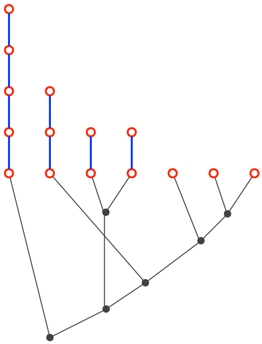
Algorithm 1



Huffman trees: average root-to-leaf distance in tree at most

$$\left(\sum_{i=1}^{k} -\frac{|C_i|}{n} \lg \frac{|C_i|}{n} \right) + 1 = H(\bar{U}) + 1$$

Algorithm 1



Huffman trees: average root-to-leaf distance in tree at most

$$\left(\sum_{i=1}^{k} -\frac{|C_i|}{n} \lg \frac{|C_i|}{n} \right) + 1 = H(\bar{U}) + 1$$

$\Rightarrow \ldots \Rightarrow$ at most $(H(\bar{U}) + 1)n$ comparisons

Algorithm 1

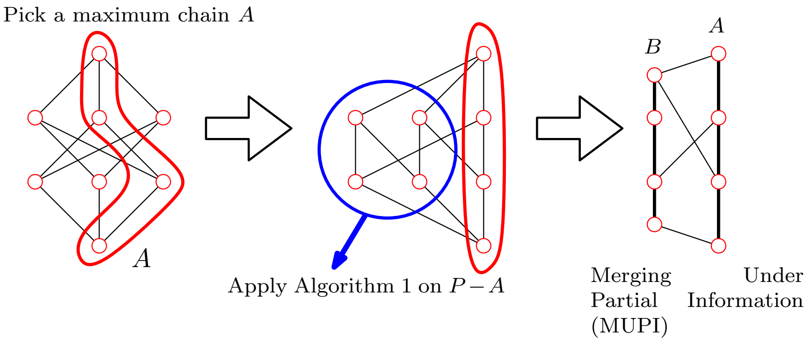
Huffman trees: average root-to-leaf distance in tree at most

$$\left(\sum_{i=1}^{k} -\frac{|C_i|}{n} \lg \frac{|C_i|}{n} \right) + 1 = H(\bar{U}) + 1$$

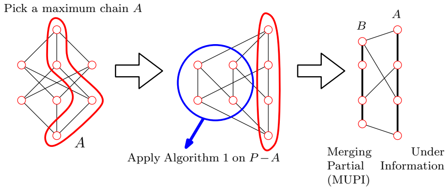
$\Rightarrow \ldots \Rightarrow$ at most $(H(\bar{U}) + 1)n$ comparisons

$$\begin{aligned} (H(\bar{U}) + 1)n &\leq (1 + \varepsilon)nH(\bar{P}) + O_\varepsilon(n) && \text{greedy} \\ &\leq (1 + \varepsilon)\big(\lg e(P) + n \lg e\big) + O_\varepsilon(n) && \text{K\&K's additive bd} \\ &= (1 + \varepsilon)\lg e(P) + O_\varepsilon(n) \end{aligned}$$

Algorithm 2

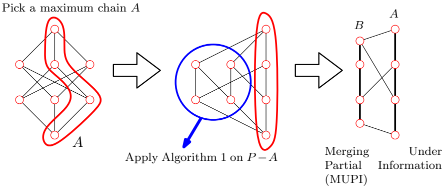

Pick a maximum chain A
A
Apply Algorithm 1 on $P - A$
B
A
Merging Under Partial Information (MUPI)

Algorithm 2



#comparisons in step 2 at most

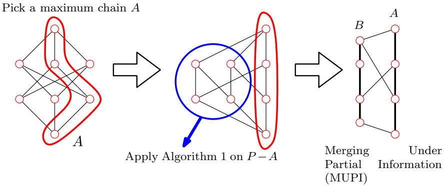
Algorithm 2



#comparisons in step 2 at most

$$(1+\varepsilon)\lg e(P-A) + O_\varepsilon(|P-A|)$$

Algorithm 2



#comparisons in step 2 at most

$$(1+\varepsilon)\lg e(P-A) + O_\varepsilon(|P-A|)$$

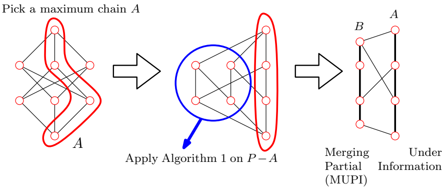
[Interlude] An easy lemma (take all intervals of length $x_v = \frac{1}{|A|}$):

$$H(\bar{P}) \geq -\lg \frac{|A|}{n}$$

$\Rightarrow |A| \geq 2^{-H(\bar{P})} n$

$\Rightarrow |P-A| \leq n\left(1 - 2^{-H(\bar{P})}\right) \leq \ln 2 \cdot nH(\bar{P})$ (using $1-2^{-x} \leq \ln 2 \cdot x$)

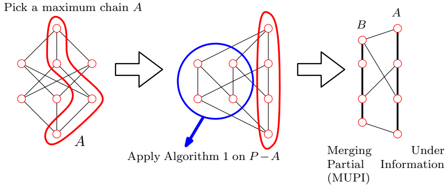
Algorithm 2



#comparisons in step 2 at most

$$
\begin{aligned}
& (1+\varepsilon)\lg e(P-A) + O_\varepsilon(|P-A|) \\
\leq\; & (1+\varepsilon)\lg e(P-A) + O_\varepsilon\left(\ln 2 \cdot nH(\bar{P})\right) \\
\leq\; & (1+\varepsilon)\lg e(P) + O_\varepsilon\left(\lg e(P)\right) \qquad \text{K\&K's multiplicative bd} \\
=\; & O_\varepsilon\left(\lg e(P)\right)
\end{aligned}
$$

Algorithm 2



#comparisons in step 2 at most

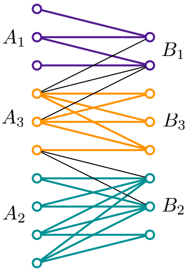
$$
\begin{aligned}
&(1+\varepsilon)\lg e(P-A) + O_\varepsilon(|P-A|) \\
\le\ &(1+\varepsilon)\lg e(P-A) + O_\varepsilon\left(\ln 2 \cdot nH(\bar{P})\right) \\
\le\ &(1+\varepsilon)\lg e(P) + O_\varepsilon\left(\lg e(P)\right) \qquad \text{K\&K's multiplicative bd} \\
=\ &O_\varepsilon\left(\lg e(P)\right)
\end{aligned}
$$

$\Rightarrow$ enough to solve MUPI = Merging under Partial Information!


MUPI

Merging under Partial Information

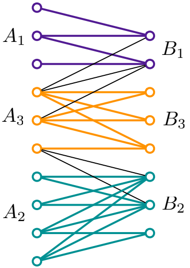
In that special case, the incomparability graph of P is bipartite



Körner and Marton 1988: optimal solution for entropy has "block structure"

Merging under Partial Information

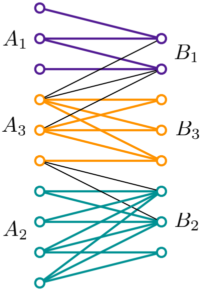
bipartite incomparability graphs $\Longrightarrow$ x^* defining $H(\bar{P})$ has an even nicer structure



- A_i interval of A, B_i interval of B, same ordering
- $x^*_v = (|A_i| + |B_i|)/n|A_i|$ whenever $v \in A_i$

Can compute $H(\bar{P})$ and x^* in time $O(n^2 \log^2 n)$

Solving MUPI - general ideas



Compute entropy and x^*

Apply Hwang-Ling merging algorithm on each component $A_i \cup B_i$ with $|A_i| \geq |B_i|$, in a certain order

Update x^* locally after each merging (details omitted)

Overall #comparisons is $\leq 3nH(\bar{P})$

Thank You!