Algorithms for Compact Letter Displays: Comparison and Evaluation

Jens Gramm[1] Jiong Guo[1] Falk Hüffner[1]
Rolf Niedermeier[1] Hans-Peter Piepho[2] Ramona Schmid[3]

[1]Friedrich-Schiller-Universität Jena
Institut für Informatik

[2]Universität Hohenheim
Institut für Pflanzenbau und Grünland

[3]Universität Bielefeld
AG Praktische Informatik

Statistik unter einem Dach
30 March 2007

Outline

All-pairwise comparisons

- Multiple pairwise comparisons among all pairs in a set of n treatments: common task in routine analyses based on analysis of variance (ANOVA) techniques
- Need a way to visualize the $\sim n^2$ pairwise comparison results (significantly different or not significantly different)

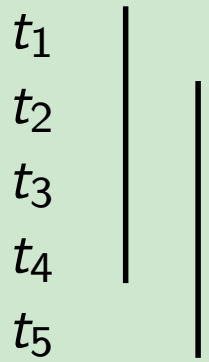
Line displays

Line display

Exactly those pairwise comparisons among treatments are non-significant that are connected by a common line.

Example

Given treatments $t_1, \ldots, t_5$, let the comparison of t_1 and t_5 is significant and all other comparisons non-significant.



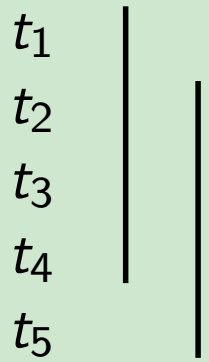
Line displays

Line display

Exactly those pairwise comparisons among treatments are non-significant that are connected by a common line.

Example

Given treatments $t_1, \ldots, t_5$, let the comparison of t_1 and t_5 is significant and all other comparisons non-significant.



Disadvantage: not always possible to find a line display

[Piepho, Biometrical J. 2000]

Letter displays

Letter display

Exactly those pairwise comparisons among treatments are non-significant that have a common letter.

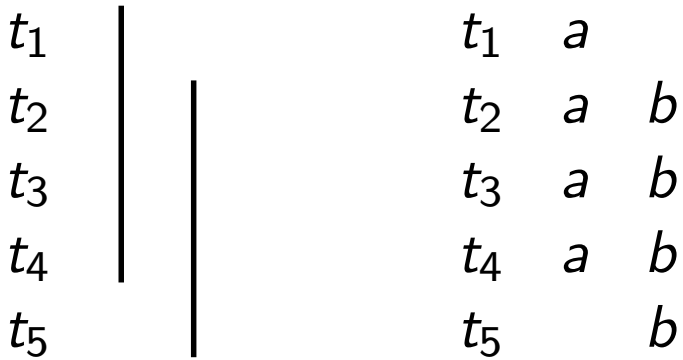
Example

Given treatments $t_1, \ldots, t_5$, let the significant comparisons be $\{\{t_1, t_5\}, \{t_1, t_3\}, \{t_2, t_4\}\}$.

t_1	a	b		
t_2		b		d
t_3			c	d
t_4	a		c	
t_5			c	d

Line displays vs. letter displays

Letter displays generalize line displays



Letter display

Always possible to find?

Letter display

Always possible to find?

Yes: Create a new column with two letters for each pair of not significantly different treatments.

Example

Given treatments $t_1, \ldots, t_5$, let the significant comparisons be $\{\{t_1, t_5\}, \{t_1, t_3\}, \{t_2, t_4\}\}$.

t_1	a	b					
t_2	a		c	d			
t_3			c		e	f	
t_4		b			e		g
t_5				d		f	g

Letter display

Always possible to find?

Yes: Create a new column with two letters for each pair of not significantly different treatments.

Example

Given treatments $t_1, \ldots, t_5$, let the significant comparisons be $\{\{t_1, t_5\}, \{t_1, t_3\}, \{t_2, t_4\}\}$.

t_1	a	b					
t_2	a		c	d			
t_3			c		e	f	
t_4		b			e		g
t_5				d		f	g

$\sim n^2$ columns: too large.

Compact letter displays

Goal

Find a *compact* letter display (that is, with minimum number of columns).

Questions

- How large can the letter display get?
- How easy is it to calculate a letter display?
- What is a good algorithm for calculating letter displays?

Compact letter displays

Goal

Find a *compact* letter display (that is, with minimum number of columns).

Questions

- How large can the letter display get? unknown
- How easy is it to calculate a letter display? unknown
- What is a good algorithm for calculating letter displays? Heuristic [Piepho, J. Comput. Graph. Stat. 2004]

Theoretical computer science

We approach these questions with the tools of theoretical computer science:

- Focus on *provable* worst-case running time and *provable* solution guarantee

Theoretical computer science

We approach these questions with the tools of theoretical computer science:

- Focus on *provable* worst-case running time and *provable* solution guarantee
- Asymptotic algorithm running time analysis

Theoretical computer science

We approach these questions with the tools of theoretical computer science:

- Focus on *provable* worst-case running time and *provable* solution guarantee
- Asymptotic algorithm running time analysis
  - Running time is stated not in absolute terms, but in relation to the input size n

Theoretical computer science

We approach these questions with the tools of theoretical computer science:

- Focus on *provable* worst-case running time and *provable* solution guarantee
- Asymptotic algorithm running time analysis
  - Running time is stated not in absolute terms, but in relation to the input size n
  - Constant factors are ignored

Theoretical computer science

We approach these questions with the tools of theoretical computer science:

- Focus on *provable* worst-case running time and *provable* solution guarantee
- Asymptotic algorithm running time analysis
  - Running time is stated not in absolute terms, but in relation to the input size n
  - Constant factors are ignored
- Classification into computational complexity classes captures "intrinsic difficulty"

Compact letter display: formal definition

Compact Letter Display

Input: Set T of n treatments, and a set H of m unordered pairs from T.

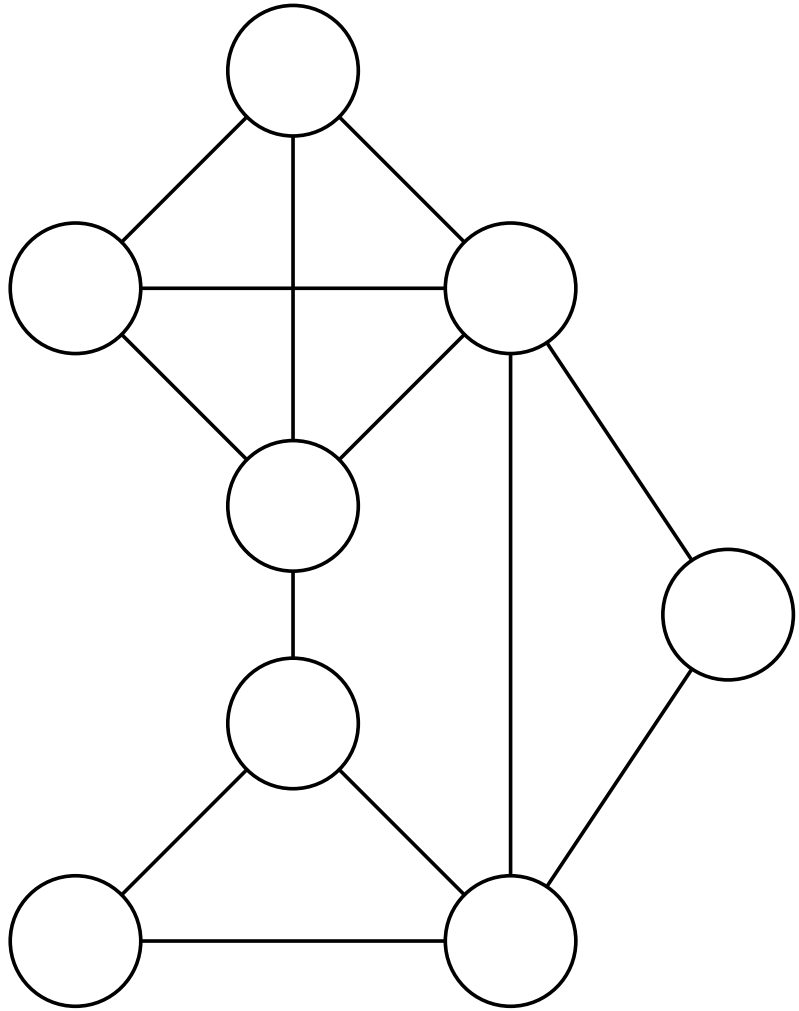
Task: Find a binary $n \times k$ matrix M with minimum k such that

$$\{t_1, t_2\} \in H \iff \exists j : M_{t_1,j} = M_{t_2,j} = 1.$$

Clique Cover

Clique Cover

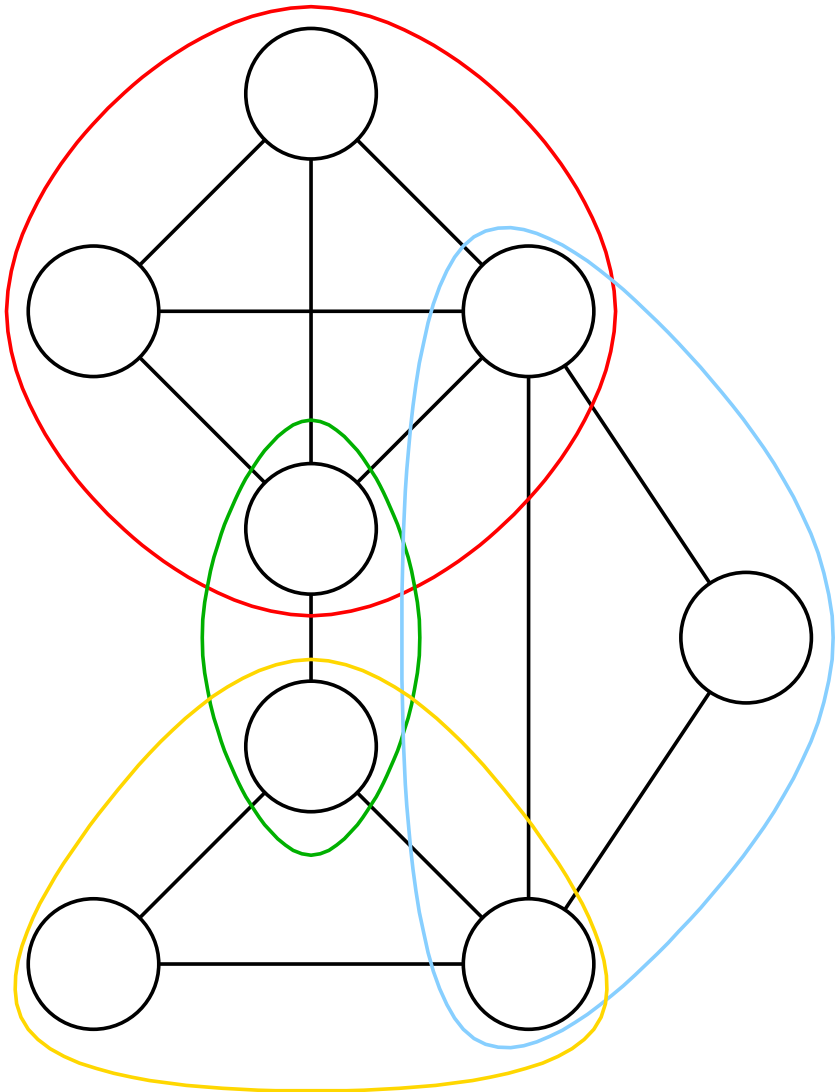
Input: An undirected graph $G = (V, E)$.
Task: Find a minimum number k of cliques (subgraphs with all edges present) such that each edge is contained in at least one clique.



Clique Cover

Clique Cover

Input: An undirected graph $G = (V, E)$.
Task: Find a minimum number k of cliques (subgraphs with all edges present) such that each edge is contained in at least one clique.



Clique Cover

Also known as

- Keyword Conflict [Kellerman, IBM 1973]
- Intersection Graph Basis [Garey&Johnson 1979]

Clique Cover

Also known as

- Keyword Conflict [Kellerman, IBM 1973]
- Intersection Graph Basis [Garey&Johnson 1979]

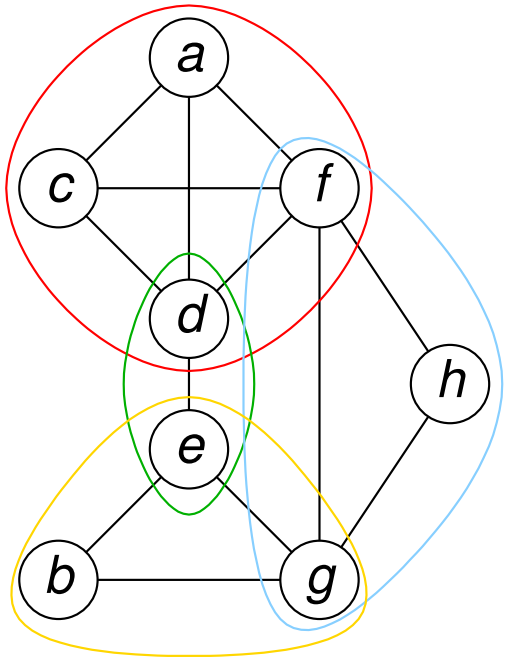
Applications

- compiler optimization,
- computational geometry, ...

Equivalence of Compact Letter Display and Clique Cover

Compact Letter Display	$\hat{=}$	Clique Cover
treatment	$\hat{=}$	vertex
not sign. diff.	$\hat{=}$	edge
column	$\hat{=}$	clique

a	×			
b				×
c	×			
d	×	×		
e		×		×
f	×		×	
g			×	×
h			×	



There is a letter display with k columns
$\iff$ there is a clique cover with k cliques.

Known results on Clique Cover

- Every graph has a clique cover of size at most $n^2/4$ (sharp)
  [Erdős et al., Canad. J. Math. 1966]
- Heuristic [Kellerman, IBM 1973]
- NP-hard [Garey&Johnson 1979]

Immediately transferable to Compact Letter Display!

NP-hardness of Clique Cover

- Theoretical computer science equates "efficiently solvable" with "solvable in polynomial time" (that is, there is some constant c such that solving a problem of size n takes at most n^c time)

NP-hardness of Clique Cover

- Theoretical computer science equates "efficiently solvable" with "solvable in polynomial time" (that is, there is some constant c such that solving a problem of size n takes at most n^c time)
- For none of the several thousand known NP-hard problems has such an efficient algorithm been found

NP-hardness of Clique Cover

- Theoretical computer science equates "efficiently solvable" with "solvable in polynomial time" (that is, there is some constant c such that solving a problem of size n takes at most n^c time)
- For none of the several thousand known NP-hard problems has such an efficient algorithm been found
- If we can solve one of them efficiently, we can solve all of them efficiently

NP-hardness of Clique Cover

- Theoretical computer science equates "efficiently solvable" with "solvable in polynomial time" (that is, there is some constant c such that solving a problem of size n takes at most n^c time)
- For none of the several thousand known NP-hard problems has such an efficient algorithm been found
- If we can solve one of them efficiently, we can solve all of them efficiently
- So we probably cannot solve any of them efficiently...

NP-hardness of Clique Cover

- Theoretical computer science equates "efficiently solvable" with "solvable in polynomial time" (that is, there is some constant c such that solving a problem of size n takes at most n^c time)
- For none of the several thousand known NP-hard problems has such an efficient algorithm been found
- If we can solve one of them efficiently, we can solve all of them efficiently
- So we probably cannot solve any of them efficiently...
- ...but there is no proof for this yet!

Insert-Absorb heuristic

Idea

Initially, consider all treatments as not significantly different, and then successively take significantly different pairs into account

1	1		1	0		1	1	0	0
2	1	$\{1,2\}$	0	1	$\{3,4\}$	0	0	1	1
3	1	$\longrightarrow$	1	1	$\longrightarrow$	1	0	1	0
4	1		1	1		0	1	0	1
5	1		1	1		1	1	1	1
6	1		1	1		1	1	1	1

Redundant columns are “absorbed”

Insert-Absorb heuristic

Idea

Initially, consider all treatments as not significantly different, and then successively take significantly different pairs into account

1	1		1	0		1	1	0	0
2	1	{1,2}	0	1	{3,4}	0	0	1	1
3	1	⟶	1	1	⟶	1	0	1	0
4	1		1	1		0	1	0	1
5	1		1	1		1	1	1	1
6	1		1	1		1	1	1	1

Redundant columns are “absorbed”

- Can produce very large letter displays
- Can run very slowly (exponential time)

Clique-Growing heuristic

Idea

Initially, consider all treatments as significantly different, and then successively take *not* significantly different pairs into account

Clique-Growing heuristic

Idea

Initially, consider all treatments as significantly different, and then successively take *not* significantly different pairs into account

- Can produce very large letter displays
- Provable running time bound: n^3

Search-Tree algorithm

Idea

- Data reduction rules: replace the instance by a smaller equivalent one.
- Enumerate all possibility of adapting the letter display to a not significantly different pair, and branch accordingly.

Search-Tree algorithm

Idea

- Data reduction rules: replace the instance by a smaller equivalent one.
- Enumerate all possibility of adapting the letter display to a not significantly different pair, and branch accordingly.

- Produces optimal letter displays
- Can run very slowly (exponential time)

Algorithm analysis: summary

Algorithm	runtime	optimality
Insert-Absorb	exponential	no guarantee
Clique-Growing	polynomial	no guarantee
Search-Tree	exponential	guaranteed

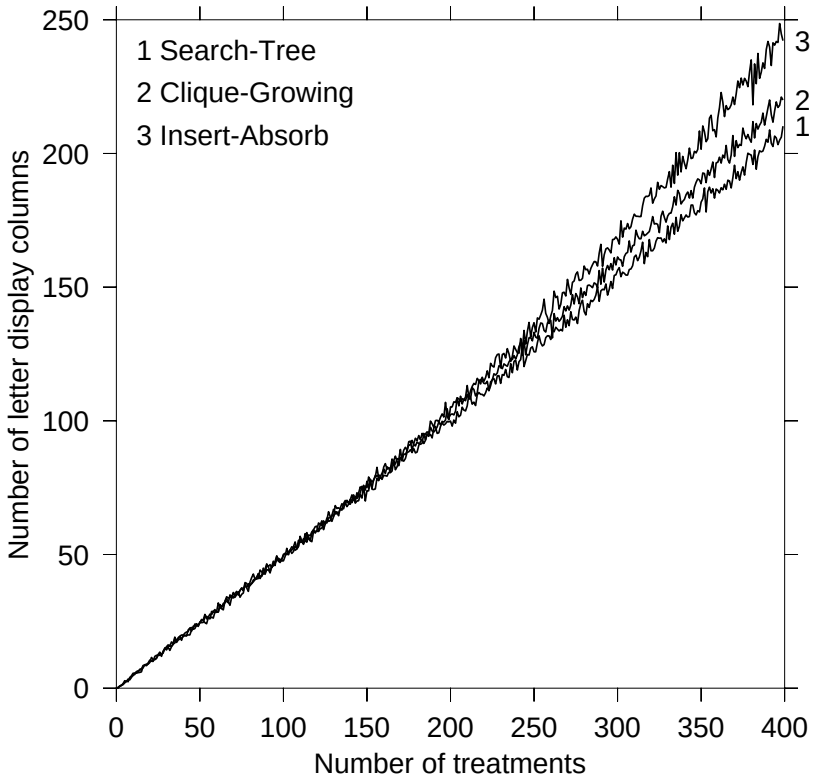
Crop yield trials

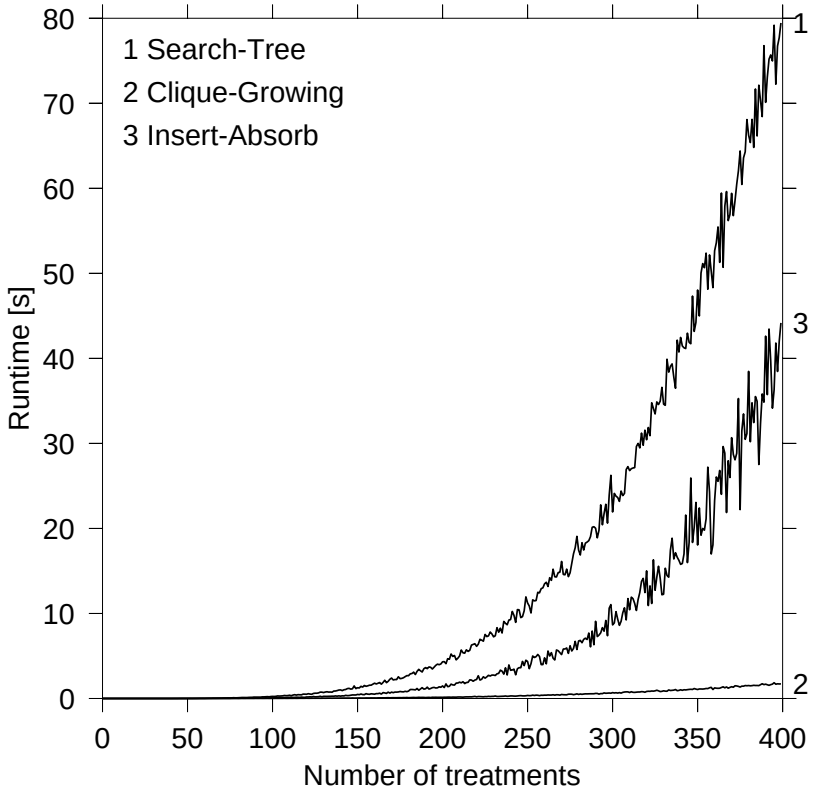
			Insert-Absorb		Clique-Growing		Search-Tree	
Dataset	n	$\lvert H\rvert$	cols	time [s]	cols	time [s]	cols	time [s]
Triticale	17	86	5	0.00	5	0.00	5	0.00
Rapeseed	74	1758	29	0.15	27	0.03	25	0.35
Wheat	124	4847	56	1.93	50	0.20	49	4.00

- Running time tolerable for all algorithms
- Clique-Growing seems to give better results than Insert-Absorb

Simulated Data

Data generated for arbitrary number of trials n by a simulation with parameters chosen to give similar results as the rapeseed data sets





Summary

- Methods of theoretical computer science give insight into the problem of finding compact letter displays
- Finding compact letter displays is hard
- An optimal algorithm (Search-Tree) is fast enough for small to medium size real-world instances
- A heuristic initially developed for the Clique Cover problem (Clique-Growing) has a worst-case time bound and gives good results

Open question

It is also desirable to minimize the number of entries in the letter display. (In the Clique Cover model, this is the sum of the clique sizes.)

Question

Is there a solution that minimizes the number of entries in the letter display, but not the number of columns?