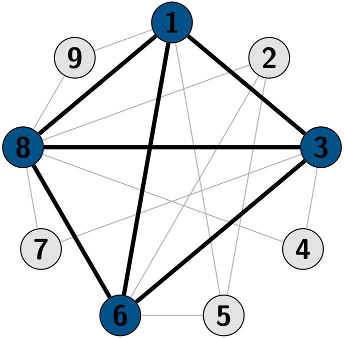
University of Glasgow

Russian Doll Search

Ciaran McCreesh

An Old Maximum Clique Algorithm



An Old Maximum Clique Algorithm

ELSEVIER

Discrete Applied Mathematics 120 (2002) 197–207

DISCRETE APPLIED MATHEMATICS

A fast algorithm for the maximum clique problem ☆

Patric R. J. Östergård *

Department of Computer Science and Engineering, Helsinki University of Technology, P.O. Box 5400, 02015 HUT, Finland

Received 12 October 1999; received in revised form 29 May 2000; accepted 19 June 2001

An Old Maximum Clique Algorithm

- Let $G[V']$ be the subgraph of G induced by V'
  - The subgraph with some of the vertices, and all of the edges between those vertices.
- Let $V' + w$ mean $V' \cup \{w\}$, with the assertion that $w \notin V'$.
- Let $\omega(G)$ be the size of a maximum clique in G.
- Now $\omega(G[V' + w])$ is either $\omega(G[V'])$ or $\omega(G[V']) + 1$.
  - And if the latter, it must contain w.

An Old Maximum Clique Algorithm

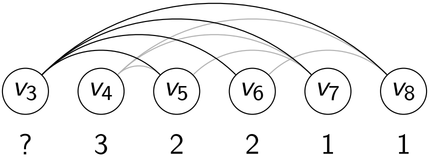
- Write out the vertices of a graph in some static order, $v_1, v_2, \ldots, v_n$.
- Let V_i be the vertices $\{v_i, v_{i+1}, \ldots, v_n\}$.
- $\omega(G[V_n])$ is 1.
- $\omega(G[V_i])$ is either $\omega(G[V_{i+1}])$ or $\omega(G[V_{i+1}]) + 1$.
  - And if the latter, it must contain v_i.
- $\omega(G)$ is $\omega(G[V_1])$.
- So we find $\omega(G[V_n])$, then $\omega(G[V_{n-1}])$, and so on, down to $\omega(G[V_1]) = \omega(G)$.

An Old Maximum Clique Algorithm

- We grow cliques recursively: C is our growing clique, and P is the undecided vertices which can be added to C.
- We pick a vertex v, add it to C, and remove from P any vertex not adjacent to v. Then we recurse until P is empty, and then backtrack and pick a new v.

An Old Maximum Clique Algorithm

- Now the clever bit: if j is the lowest-numbered vertex left in P, then $\omega(G[V_j])$ gives us a bound on how much further we can extend the growing clique.
  - And $j > i$, thanks to the static variable ordering: P initially contains only vertices to the right of our top-level i.
- This is a better bound than $|P|$, because it knows about relationships between undecided vertices.



Valued Constraint Satisfaction Problems

- A Valued Constraint Satisfaction Problem has a set of variables, a set of constraints, and a well-behaved way of assigning a cost to violating a subset of the constraints.
  - Flexible enough to handle Partial CSPs, Additive CSPs with hard constraints, Probabilistic CSPs, some Objective CSPs, ...
- We seek an assignment of values to variables which minimises this cost.

Valued Constraint Satisfaction Problems

- One possible model for maximum clique:
  - A boolean variable V_i for each vertex v_i, determining whether it is in the clique.
  - For each pair of nonadjacent vertices v_i and v_j, a constraint "not (V_i and V_j)", with cost infinity.
  - For each vertex, a constraint "V_i is true", with cost 1.
  - The total cost is the sum of the cost of violated constraints, and tells us the number of vertices *not* in the clique.

Valued Constraint Satisfaction Problems

- A simple lower bound looks at violations in already-assigned variables (backward checking).
- Add up the costs of all of the constraints we've already violated.
- If the cost of the constraints we've violated so far is greater than or equal to the cost of the best solution we've found so far, we can backtrack immediately.
- For clique: how many vertices have we rejected so far?

Valued Constraint Satisfaction Problems

- We can also look at constraints which have only one unassigned variable (forward checking).
- For each constraint with one uninstantiated variable, pick the value for that variable that has the cheapest cost. Now add these costs together.
- For clique: how many vertices can no longer be included?

Russian Doll Search

Russian Doll Search for Solving Constraint Optimization Problems

Gérard Verfaillie and **Michel Lemaître**
CERT/ONERA
2 av. Edouard Belin, BP 4025
31055 Toulouse Cedex, France
{verfaillie, lemaitre}@cert.fr

Thomas Schiex
INRA
Chemin de Borde Rouge, Auzeville, BP 27
31326 Castanet Tolosan Cedex, France
tschiex@toulouse.inra.fr

Russian Doll Search

Russian Doll Search

- Write out the variables in some static order, $V_1, V_2, \ldots, V_n$.
- Solve the problem containing just V_n, and remember the cost.
- Solve the problem containing just V_{n-1} and V_n, using the static variable ordering, and remember the cost. If at any point the cost of a partial assignment plus the best cost of assigning V_n (which we know) is more than the best so far, backtrack immediately.
- Now add V_{n-2} and solve.
- And so on, until we solve for all variables.

Russian Doll Search

- Typically, the Russian dolls pass is used only once, at the top of search. After that, regular branch and bound is used.
- We could do Russian dolls at every level. But this is very expensive, and might not improve the bound by much.

Where is it Used?

- Seems to be good if we have a fairly well-behaved objective function, but a very weak bound.
  - Earth Observation Satellite Scheduling
  - Radio Link Frequency Assignment
  - DNA Analysis
  - Maximum Density Still-Life (until Chu and Stuckey solved it)

Where is it Used?

RUSSIAN DOLL SEARCH ALGORITHMS FOR
DISCRETE OPTIMIZATION PROBLEMS

Vesa Vaskelainen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Electronics, Telecommunications and Automation for public examination and debate in Auditorium S4 at Aalto University School of Science and Technology (Espoo, Finland) on the 19th of November, 2010, at 12 noon.

- Covering problems in directed graphs and hypergraphs:
  - Steiner triple
  - Maximum transitive subtournament
  - Best barbequeue

Pseudo-Tree Decompositions

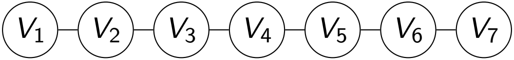
Pseudo-Tree Search with Soft Constraints

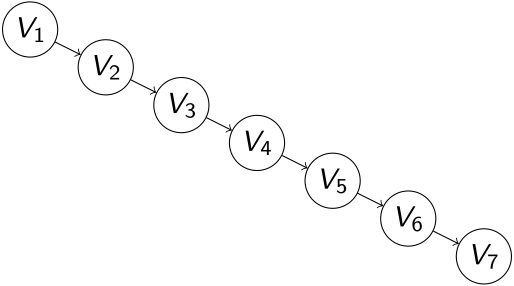
Javier Larrosa[1] and **Pedro Meseguer**[2] and **Martí Sánchez**[3]

Russian Doll Search with Tree Decomposition

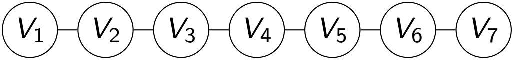
M. Sanchez, D. Allouche, S. de Givry, T. Schiex
UBIA, UR 875, INRA, F-31320 Castanet Tolosan, France
{msanchez,allouche,degivry,tschiex}@toulouse.inra.fr

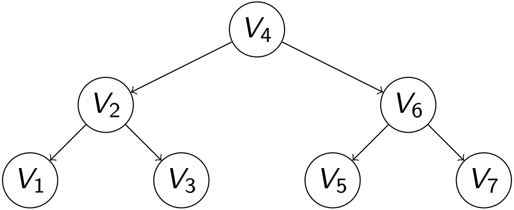
Pseudo-Tree Decompositions



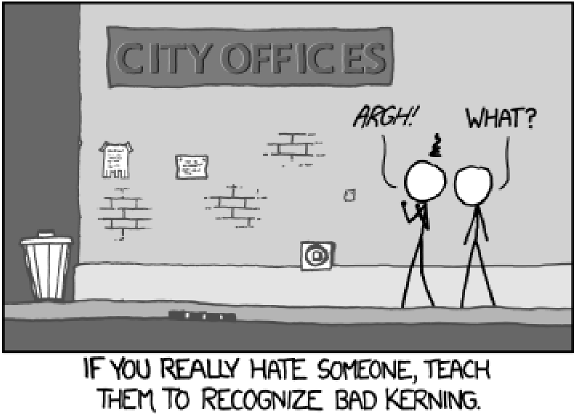


Pseudo-Tree Decompositions





Pseudo-Tree Decompositions



http://xkcd.com/1015/ (Randall Munroe), CC-BY-NC 2.5

Specialisation

Opportunistic Specialization in Russian Doll Search*

Pedro Meseguer[1], Martí Sánchez[1], and Gérard Verfaillie[2]

[1] IIIA-CSIC
Campus UAB, 08193 Bellaterra, Spain
{pedro|marti}@iiia.csic.es
[2] ONERA
2 av. Edouard Belin, BP 4025, 31055 Toulouse Cedex, France
verfaillie@cert.fr

- Instead of looking at variables, look at variable-value pairs.
- This is even more expensive, but sometimes it pays off if we do it selectively.

Elimination Rules

A Bit-Parallel Russian Dolls Search for a Maximum Cardinality Clique in a Graph☆

Ricardo C. Corrêa[a], Philippe Michelon[b], Bertrand Le Cun[c], Thierry Mautor[c], Diego Delle Donne[d]

[a]Universidade Federal do Ceará, Departamento de Computação, Campus do Pici, Bloco 910, 60440-554 Fortaleza - CE, Brazil
[b]Université d'Avignon et des Pays du Vaucluse, Laboratoire d'Informatique d'Avignon, F-84911 Avignon, Cedex 9, France
[c]Université de Versailles Saint Quentin, 45 Avenue des Etats Unis, 78035, Versailles, France
[d]Sciences Institute, National University of General Sarmiento, J. M. Gutiérrez 1150, Malvinas Argentinas, (1613) Buenos Aires, Argentina

- We can reduce the number of dolls using a greedy heuristic: we can move variables that may be added without increasing the cost to the left of our current variable, and skip the following iterations.

Parallelism?

```
17:31 < sdstrowes> you know the problem with russian dolls?
17:31 < sdstrowes> they're just so full of themselves.
```

http://dcs.gla.ac.uk/~ciaran
c.mccreesh.1@research.gla.ac.uk