Department of Mathematics and Computer Science
University of Southern Denmark, Odense

January 3, 2011
Marco Chiarandini

DM811 - Heuristics for Combinatorial Optimization

Exam Project, Fall 2010

Note 1 The project is carried out individually and it is not allowed to collaborate. It consists of: algorithm design, implementation, experimentation and written report.

The evaluation of the project is based on the report. However, a program that implements the best algorithm described in the report must also be submitted. The program will serve to verify the correctness of the results presented. The report may be written in English or in Danish.

Note 2 Corrections or updates to the project description will be published on the course web page and will be announced by email to the addresses available in the Blackboard system. In any case, it remains students' responsibility to check for updates on the web page.

Note 3 *Submission.* An archive containing the electronic version of the written report and the source code of the program must be handed in through the Blackboard system **before 12:00 of Monday, 25 January 2011**. This is the procedure:

- choose the course DM811 in Blackboard,
- choose "Exam Project Hand In" in the menu on the left,
- fill the form and conclude with submit,
- print the receipt (there will be a receipt also per email).

See Appendix C for details on how to organize the electronic archive. Reports and codes handed in after the deadline will generally not be accepted. System failures, illness, etc. will not automatically give extra time.

---

1 Introduction

Several hard graph problems become easy when restricted to trees. Based on this fact one would like to find out how "tree-like" a given graph is. This question is the source of the concept of tree decomposition introduced by Neil Robertson and Paul Seymour [RS84]. Related to the *tree decomposition* is the *treewidth* measure, the smaller the treewidth number the more tree-like the graph is. Recently, treewidth received interest in fixed-parameter tractability as the parameter in which many problems are recognized as tractable.

Instances of constraint satisfaction problem can be solved efficiently if the treewidth of the corresponding constrained graph is small [RN03]. The solution method first generates a tree decomposition with small treewidth, solves the problem by dynamic programming at each vertex of the decomposition and computes the overall final solution from the knowledge of the solution in the separated problems.

Other uses of tree decomposition appear in expert system applications [SL90]. Decision support systems have probabilistic networks as underlying knowledge representation. In this networks dependencies between variables are modelled using directed graphs: to each vertex corresponds a vertex of the graph and there is an arc between two vertices if the corresponding variables are in a dependency relation. An important problem on this network is probabilistic inference: determining the probability distribution for a variable given a value assignment for the other variables. An efficient algorithm to carry out inference is based on a tree decomposition of the graph.

Finally, in computational biology, tree decomposition has been used for protein structure prediction.

2 Basic Definitions

All graphs we consider are undirected and simple, i.e., without loops or parallel edges. A graph G consists of a set of vertices $V(G)$ and edges $E(G)$. Every edge is incident with two vertices. Vertices linked by an edge are said to be *adjacent* and the set of vertices adjacent to v in graph $G = (V, E)$ is denoted by $A_G(v) = \{w \in V \mid vw \in E\}$. The size of $A_G(v)$ is the degree of a vertex and is denoted by $d_G(v)$.

For a vertex set $W \subseteq V$ the subgraph of $G = (V, E)$ *induced* by W is $G[W] = (W, \{uv \in E \mid u, v \in W\})$. A set of vertices Q is called a clique if there is an edge between each pair of distinct vertices of Q. The maximum cardinality of a clique in G is denoted by $\omega(G)$. A *cycle* of length j is a sequence of distinct vertices $[v = v_1, v_2, \dots, v_j = v]$ such that for all $1 \le i < j$ we have $v_i v_{i+1} \in E$. A *chord* is an edge between two non-consecutive vertices in a cycle. A graph is *connected* if for each $u, v \in V$, there is a path between u and v. A *tree* $T = (V, E)$ is a connected graph with no cycle.

Definition 1. Tree Decomposition [RS84]
Let $G = (V, E)$ be a graph. A tree decomposition of G is a pair (T, X), where T is a tree and $X = \{X_i \mid i \in V(T)\}$ is such that:

(i) $\bigcup_{i \in V(T)} X_i = V(G)$,

(ii) for all $vw \in E$, there is an $i \in V(T)$ with $v, w \in X_i$,

(iii) for all $i, j, k \in V(T)$: if j lies on the path from i to k in T then $X_i \cap X_k \subseteq X_j$.

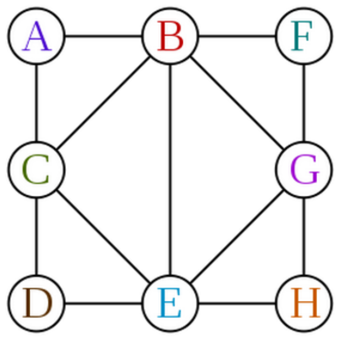
Each X_i is also called a bag. The width of (T, X) is $\max\{|X_i| - 1 \mid i \in V(T)\}$.

Definition 2. Treewidth
The treewidth of G denoted by $tw(G)$ is the minimum k such that G has a tree decomposition of width k.

An example of tree decomposition is given in Figure 1. A clique of n vertices has treewidth $n - 1$. The corresponding tree decomposition trivially consists of one bag containing all graph vertices. In fact, no tree decomposition with smaller width is attainable. More generally, it is known that every complete subgraph of a graph G is completely "contained" in a bag of G's tree decomposition. In contrast, a tree has treewidth 1 and the bags of the corresponding tree decomposition are simply the two-element vertex sets formed by the edges of the tree.[1]

Computing the treewidth is NP-hard [ACP87].

[1]An intuitively appealing characterization of tree decomposition is in terms of the cops-and-robbers game. The robber stands on a vertex of the graph and can at any time run at great speed to any other vertex along a path of the graph. However, he is not permitted to run through a cop. There are k cops, each of whom at any time either stands on a vertex or is in a helicopter (that is, he is temporarily removed from the game). The objective of the player controlling the movement of the cops is to land a cop via helicopter on the vertex occupied by the robber, and the robber's objective is to elude capture. (The point of the helicopters is that cops are not constrained to move along paths of the graph — they move from vertex to vertex arbitrarily.) The robber can see the helicopter approaching its landing spot and may run to a new vertex before the helicopter actually lands. Thus, for the cops to capture the robber they will need to first occupy all vertices adjacent to the vertex where the capture is to take place, because otherwise the robber will be able to run to a different vertex and not be captured. The cops can see the robber at all times — the difficulty is just to corner him somewhere.
The minimum number of cops needed to catch a robber is the treewidth of the graph plus one.



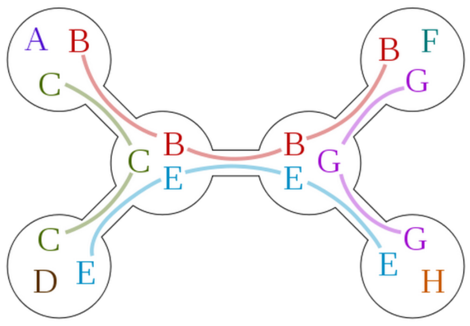


Figure 1: A graph with eight vertices and a tree decomposition of it onto a tree with six nodes. Each graph edge connects two vertices that are listed together at some tree node, and each graph vertex is listed at the nodes of a contiguous subtree of the tree. Each tree node lists at most three vertices, so the width of this decomposition is two.

A graph is called *triangulated* if every cycle of length at least four contains a chord. Triangulated graphs are also called *chordal* due to the presence of a chord in every cycle. Computing the tree width for triangulated graphs can be done in linear time.

Proposition 1. *[Gav72] If G is triangulated, $tw(G) = \omega(G) - 1$ and computing $\omega(G)$ can be done in $O(n + m)$.*

For an arbitrary graph $G = (V, E)$, it is interesting to look for a triangulated graph that contains G. A triangulated graph $H = (V, E^T)$, with $E \subseteq E^T$, is called a *triangulation* for G. A triangulation $H = (V, E^T)$ is a *minimal triangulation* of $G = (V, E)$ if there is no triangulation of G that is a proper subgraph of H, i.e., if there is no set of edges F such that (V, F) is a triangulation of G with $F \subseteq E^T, F \neq E^T$.

Proposition 2. *For every graph G there exists a triangulation H^* such that $tw(H^*) = tw(G)$.*

Corollary 1. *Finding the treewidth of a graph G is equivalent to finding a triangulation H^* of G with minimum size of the maximum clique.*

Since finding the treewidth of a graph is NP-hard, also finding a triangulation with minimum clique number is an NP-hard problem. However each triangulation H of G gives an upper bound of $tw(G)$ and its clique number can be computed in linear time.

In the following we describe how triangulation can be built. For a graph G, a vertex is said to be *simplicial* if the subgraph induced by $A_G(v)$ is a clique. An *(elimination) ordering* $\pi : \{1, 2, \dots, n\} \mapsto V$ is a *perfect elimination ordering* for G if for any $i \in \{1, \dots, n\}$, $\pi(i)$ is a simplicial vertex in $G[\{\pi(i), \dots, \pi(n)\}]$, that is, the subgraph induced by the set of vertices adjacent to v that succeed it in π, $\{w \in A_G(w) \mid \pi^{-1}(v) < \pi^{-1}(w)\}$ form a clique.

Proposition 3. *(Fulkerson and Gross, 1965) G is triangulated if and only if it has a perfect elimination ordering.*

Given a perfect elimination ordering, the maximal cliques of G are of the form $\{v\} \cup M(v)$, where $M(v) = \{w \in A_G(v) : \pi^{-1}(v) < \pi^{-1}(w)\}$ denotes the set of higher ordered adjacent vertices.

Proposition 4. *(Arnborg 1985) For a triangulated graph G the value $\max_{v \in V} |M(v)|$ derived from a perfect elimination scheme equals the tree width of G.*

Another connection of tree decomposition of graphs with algorithmic graph theory are graph separators, that is, vertex sets whose removal from the graph separates the graph into two or more connected components. Actually each bag of a tree decomposition forms a separator of the corresponding graph.

```
function Triangulation;
input a graph $G = (V, E)$, an elimination ordering $\pi$;
output $H = (V, E_\pi)$ a triangulation for $G$ with $tw(H) \geq tw(G)$;
$H = G$;
for $v \in V$ do
    $M(v) := A_G(v)$;
$E_\pi := E, tw := 0$;
for $i = 1$ to $n$ do
    $v := \pi(i)$;
    if $|M(v)| > tw$ then
        $tw := |M(v)|$;                                   // update treewidth
    for $u, w \in M(v)$ and $uw \notin E_\pi$ do
        $E_\pi := E_\pi \cup uw$ ;          // add edge between higher ordered neighbors
        $M(u) := M(u) \cup \{w\}, M(w) := M(w) \cup \{u\}$ ;     // update adjacency list
    for $u \in M(v)$ do
        $M(u) := M(u) \setminus \{v\}$ ;                  // $v$ is lower ordered than $u$
return $(H)$;
```

Figure 2: Graph triangulation according to an elimination ordering π

For an arbitrary (non-triangulated) graph $G = (V, E)$ and an elimination scheme π, a triangulation $H = (V, E_\pi)$ can be constructed by the algorithm in Figure 2 [Par61, Ros70]. Moreover, the algorithm returns the treewidth of H, which is an upper bound for the treewidth of G. The algorithm adds edges to G to make it triangulated. Vertices are eliminated in the elimination ordering π. At each step i of the algorithm, the necessary edges to make $v = \pi(i)$ be a simplicial vertex are added to the current graph. Then the vertex is deleted. Let $m' = |E_\pi|$. This algorithm can be implemented in $O(n + m')$ time [Ros70].

The application of the algorithm Triangulation on a given graph G is illustrated in Figure 3. Suppose that we are given the following elimination ordering: $\pi(1) = 10, \pi(2) = 9, \pi(3) = 8, \ldots$. The vertex 10 is first eliminated from G. When this vertex is eliminated no new edges are added in the graph G and H (graph H is not shown in the figure), as all neighbors of node 10 are connected. Further from the remaining graph G the vertex 9 is eliminated. To connect all neighbors of vertex 9, two new edges are added in G and H (edges $(5, 8)$ and $(6, 7)$). The process of elimination continues until the triangulation H is obtained.

Proposition 5. *For a triangulated graph G there is a tree decomposition (T, X) such that for each $i \in V(T)$, X_i is a clique in G.*

Thus during the vertex elimination process of the algorithm of Figure 2 a tree decomposition can be obtained. First the nodes of tree decomposition are created. This is illustrated in Figure 3. When vertex 10 is eliminated a new tree decomposition node is created. This node contains the vertex 10 and all other vertices which are connected with this vertex in current graph G. Further the next tree node with vertices $\{5, 6, 7, 8, 9\}$ is created when the vertex 9 is eliminated. To the end of elimination process all tree decomposition nodes will be created. The created tree nodes should be connected, such that the connectedness condition for the vertices is fulfilled. This is the third condition in the tree decomposition definition of Definition 1. To fulfil this condition the tree decomposition nodes are connected as following. The tree decomposition node with vertices $\{10, 9, 8\}$ that is created when vertex 10 is eliminated, is connected with the tree decomposition node which will be created when the next vertex in the ordering which appear in $\{10, 9, 8\}$ is eliminated. In this case the node $\{10, 9, 8\}$

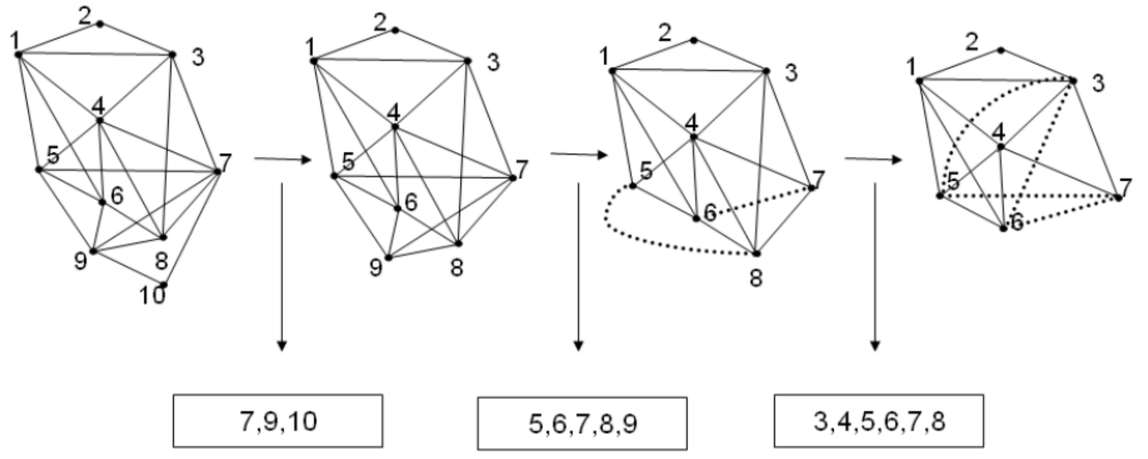


Figure 3: Illustration of the elimination of nodes $10, 9$ and 8 and generation of tree decomposition nodes during the construction of a triangulation

```
function PermutationToTreeDecomposition;
input a graph $G = (V, E)$, an elimination ordering $\pi(1), \dots, \pi(n)$;
output $(T, X)$ a tree decomposition;
$v = \pi(1)$;
if $|V| = 1$ then
    return a tree decomposition with one bag $X_v = \{v\}$
Compute the graph $G' = (V', E')$ by adding to $E$ an edge between each pair of non-adjacent
vertices in the graph induced by $A_G(v)$ and removing $v$ from $V$;
$(T' = (V', F'), \{X_i \mid i \in V'\})$ := PermutationToTreeDecomposition($G', \pi(2), \dots, \pi(n)$);
Let $v_j$ be the lowest numbered neighbor of $v$, i.e., $j = \min\{i \mid \{v, v_i\} \in E\}$;
Construct a bag $X_v = A_G(v) \cup \{v\}$;
return $(T = (V, F), \{X_i \mid i \in V\})$ with $F = F' \cup \{v, v_j\}$
```

Figure 4: A recursion algorithm to derive a tree decomposition from an elimination ordering

should be connected with the node created when vertex 9 is eliminated, because this is the next vertex in the ordering that is contained in $\{10, 9, 8\}$. This rule is further applied for connection of other tree decomposition nodes. This procedure can be formalized in the recursive algorithm of Figure 4.

3 Project Requirements

The aim of the project is to study heuristic algorithms for finding the treewidth of arbitrary graphs and compare the heuristics on the test instances of Table 1. In the table the best known lower bounds (LB) and upper bounds (UB) for those instances are given. The instances can be downloaded from the course web site.

All the following points must be addressed to pass the exam:

1. Implement the procedure Triangulation of Figure 2 and compute the treewidth of random elimination ordering. Call this algorithm Random and use its results as control benchmark for your heuristics. As a minimal requirement they must do better than that.

2. Design and implement one or more construction heuristics that perform better than Random.

name	$\lvert V\rvert$	$\lvert E\rvert$	LB	UB	Time Lim.
myciel3	11	20	4	5	1
queen5_ 5	25	160	12	18	10
queen6_ 6	36	290	15	25	10
queen7_ 7	49	476	20	35	10
queen8_ 8	64	728	23	46	10
queen9_9	81	1056	26	58	10
queen10_10	100	1470	31	72	500
queen11_11	121	1980	34	88	500
queen12_12	144	2596	37	104	500
queen13_13	169	3328	42	122	500
queen14_14	196	4186	45	141	500
queen15_15	225	5180	48	163	500
queen16_16	256	6320	53	186	500
dsjc125.1	125	736	16	65	900
dsjc125.5	125	3891	62	109	900
dsjc125.9	125	6961	108	119	900
dsjc250.1	250	3218	32	173	900
dsjc250.5	250	15668	125	232	900
dsjc250.9	250	27897	218	243	900

Table 1: The set of test instances with vertex set size, number of edges, lower bound, upper bound and time limit in seconds to use. The instance myciel3 is for debugging purposes. The remaining instances are divided into two main classes, queens graphs and uniform random graphs (dsjc).

3. Design and implement one or more local search algorithms.
4. Design and implement an effective algorithm enhancing the heuristics at the previous two points with the use of stochastic local search methods and metaheuristics.
5. For all the methods above carry out an experimental analysis and draw sound conclusions.

In the experimental assessment all algorithms the maximum time in seconds allowed per run on a single instance is given in Table 1.[2]

4 Remarks

Remark 1 For each point above a description must be provided in the report of the work undertaken. In particular for the best algorithms arising from the experimental analysis enough details must be provided in order to guarantee the reproducibility of the algorithm from the report only (i.e., without having to look at the source code).

Remark 2 The results of the experiments must be reported either in graphical form or in form of tables or both. Moreover, for the best solver resulting from the point 4, a table must be provided with the best results for each specific instance of Table 1.

Remark 3 The total length of the report should not be less than 5 pages and not be more than 12 pages, appendix included (lengths apply to font size of 11pt and 3cm margins). Although these bounds are not strict, their violation is highly discouraged. In the description of the algorithms, it is allowed

[2]Times refer to machines in IMADA terminal room.

(and encouraged) to use short algorithmic sketches in form of pseudo-code but not to include program codes.

Remark 4 This is a list of factors that will be taken into account in the evaluation:

- quality of the final results;
- level of detail of the study;
- complexity and originality of the approaches chosen;
- organization of experiments which guarantee reproducibility of conclusions;
- clarity of the report;
- effective use of graphics in the presentation of experimental results.

Appendix A Instance Format

All graphs are in DIMACS format. It consists of a file in which each line begins with a letter that defines the content of the line. The legal lines are:

- `c` Comment: remainder of line ignored.
- `p` Problem: is of form:
  - `p edge n m` where n is the number of vertices (to be numbered 1..n) and m the number of edges.
- `e` Edge: is of the form `e n1 n2` where `n1` and `n2` are the endpoints of the edge.

Appendix B Solution Format

In order to check the validity of the results reported, the program submitted must output when finishing the best solution found during its execution in a file with extension `.sln`. The file must be in text format and contain

- In the first line the treewidth
- In the second line the corresponding elimination ordering, that is, an ordered list of vertices

Appendix C Handing in Electronically

The electronic archive to hand in must be organized as follows. It expands in a main directory named with the first 6 digits of your CPR number (e.g., 030907). The directory has the following content:

```
CPRN/README
CPRN/report/
CPRN/src/
CPRN/data/
```

The directory `reprot` contains a pdf or postscript version of your report. *Do not put your name in the author field of the report, instead put your CPR number.* The file README provides instructions for compilation of the program. The directory `src` contains the sources which may be in C, C++, Java or other languages. If needed a Makefile can be included either in the root directory or in `src`. After compilation the executable must be placed in `src`. For java programs, a jar package can also be submitted. The directory `data` contains the instances.

Programs must work on IMADA's computers under Linux environment and with the compilers and other applications present on IMADA's computers. Students are free to develop their program at home, but it is their own responsibility to transfer the program to IMADA's system and make the necessary adjustments such that it works at IMADA.[3] Comet version 1.3 is also installed.

The executable must be called `treewidth`. It must execute from command line by typing in the directory `CPRN/src/`:

```
fctt -i INSTANCE -t TIME -s SEED -o OUTPUT
```

where the flags indicate:

[3]Past issue: the java compiler path is `/usr/local/bin/javac`; in C, any routine that uses subroutines from the math.c library should be compiled with the `-lm` flag – eg, `cc floor.c -lm`.

- `-i INSTANCE` the input instance;
- `-t TIME` the time limit in seconds;
- `-s SEED` the random seed;
- `-o OUTPUT` the file name where the solution is written.

For example:

```
treewidth -i data/queen15_15.col -o queen15_15.sln -t 180 -s 1 > queen15_15.log
```

will run the program on the instance `queen15_15.col` opportunely retrieved from the given path for 180 seconds with random seed 1 and write the solution in the file `queen15_15.sln`.

In its default mode, the program must run the best algorithm developed and must print on the standard output **only one single number** at the end of the run corresponding to the quality of the best solution found during the run.

It is advisable to have a log of algorithm activities during the run. This can be achieved by printing further information on the standard error or in a file. A suggested format is to output a line whenever a new best solution is found containing at least the following pieces of information:

```
best 853 time 10.000000 iter 1000
```

All process times are the sum of user and system CPU time spent during the execution of a program as returned by the linux C library routine `getrusage`. Process times include the time to read the instance.

References

[ACP87] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8:277–284, April 1987.

[Gav72] Fanica Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal of Computing*, 1(2):180–187, 1972.

[Par61] S. Parter. The use of linear graphs in gauss elimination. *SIAM Review*, 3(2):119–130, 1961.

[RN03] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Englewood Cliffs, New Jersey, USA, second edition, 2003.

[Ros70] Donald J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597 – 609, 1970.

[RS84] Neil Robertson and Paul D. Seymour. Graph minors iii: Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36:49–64, 1984.

[SL90] D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.