RWTH Aachen University **WS 2010/11**
Lehrgebiet Theoretische Informatik Tutorial 3
Kneis—Langer—Sikdar 02.11.2010

Tutorial Exact Algorithms

Exercise T5

The problem Independent Dominating Set is defined as follows:

Input: A graph $G = (V, E)$ and an integer k.
Question: Is there a set $U \subseteq V$, such that $|U| = k$ and U is both, independent *and* dominating?

Which of the following reduction rules or solution strategies known for Independent Set and/or Dominating Set can be applied or slightly adjusted for this problem?

- Deletion of degree zero vertices
- Deletion of degree one vertices
- Folding
- Domination
- Reduction to Set Cover

Design an algorithm for this problem that for every k is exponentially faster than simple enumeration.

Solution

- Isolated nodes must be in any solution in order to be *dominated*.
- We do not have something like "at least k nodes", but "exactly k". In particular, the property of being an independent dominating set is not a monotone property in the sense that if $I \subseteq V$ is an independent dominating set, then either any subset or superset of I is an indepenendet dominating set. We therefore cannot derive any easy, suitable exchange arguments for degree one vertices similar to the case of Independent Set. This one is a "no" so far.
- If v is a node of degree two and u_1, u_2 are the neighbors of v, and, say, u_2 must be contained in any solution, then we cannot derive any rule whether or not the new "folded" node will be contained in the solution or not. In particular, the cycles of length five, four and three give a series of counter examples, where both cases occur. Hence, "no".
- Degree one is a special case of domination, so "no" is easy here.

- A simple reduction to Set Cover as in the case of Dominating Set seems unlikely, since for Set Cover there is no notion of "independence".

  Note that any NP-complete problem can be reduced in polynomial time to Set Cover, i.e., in particular Independent Dominating Set. However, to capture the "indepenence" aspect, we need to use some kind of *gadget* to encode "independence" of vertices into the Set Cover instance. This gadget will most likely have a very negative impact to the running time of the algorithm.

The algorithm now first applies the first reduction rule, and then branches on whether a node v is contained in the solution or not. If v is in the solution, no neighbor of v can be contained in the solution. We therefore get a branching vector of at least $(1, 2)$, and the running time of the algorithm is at most $O^*(\tau(1, 2)^n) = O^*(1.619^n)$, which is significantly faster than the $O^*(2^n)$ brute-force enumeration.

Exercise T6

The problem MaxCut is defined as follows:

Input:	A graph $G = (V, E)$.
Feasible solutions:	Any bipartition $V = V_1 \cup V_2$.
Goal:	Maximize the number of edges that are *cut* by the bipartition.

Design an algorithm or this problem with a running time of $O^*(\tau(3, 3)^m)$, where as usual $m = |E|$.

Solution

Next week!

Homework Assignment H5 (10 Points)

Let F be a formula in 3-CNF. A clause $\{l_1, l_2, l_3\}$ is *not-all-equal satisfied* if there are literals l_i, l_j such that l_i is true and l_j is false. A formula is *not-all-equal satisfied* by an assignment if each clause is not-all-equal satisfied.

Find an algorithm that decides whether a formula F in 3-CNF can be not-all-equal satisfied that is exponentially faster than simple enumeration, i.e., with a runnign time of $O^*(c^n)$, where $c < 2$ and n is the number of variables in F.

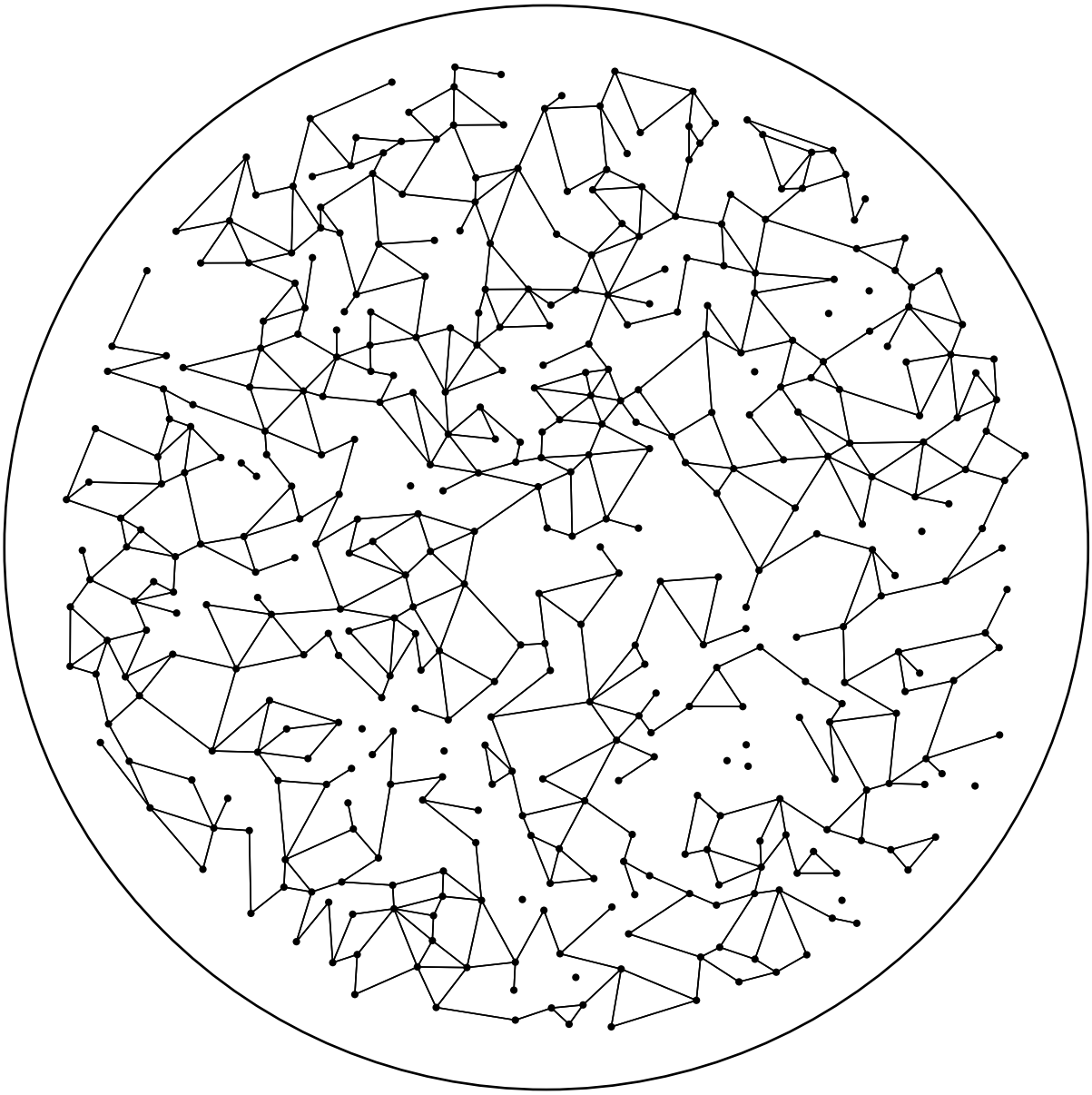
Solution

Each clause must be satisfied. Among the $2^3 = 8$ ways to assign values to the literals, two are forbidden: All literals to zero (does not satisfy), all literals to one (all equal).

The algorithm therefore picks a clause of length three and branches on all the six possible assignments to its variables. Since each clause can be assumed to contain at least three literals — if there are only clauses of length two left, the problem can be solved in polynomial time — we gain three variables in each branch and hence a branching vector of $(3, 3, 3, 3, 3, 3)$ with $\tau(3, 3, 3, 3, 3, 3) = 1.817121$.

Homework Assignment H6 (10 Points)

Find an optimal *independent set* in the following graph (and mark the vertices in the solution). How big is your solution?



Lösungsvorschlag The following picture shows an optimal *vertex cover* C for this graph $G = (V, E)$ in red, which contains 217 vertices. Since $I = V \setminus C$ is an optimal independent set, the black vertices in this figure form an independent set of size ≈ 213 vertices.