The Pennsylvania State University

Computer Science Department
University Park, PA 16802

Technical Report No. 207
September 1976

HAVING A GRUNDY-NUMBERING IS NP-COMPLETE

Jan van Leeuwen

Abstract

An assignment of integers to the vertices of a (directed) graph is a Grundy-numbering if and only if at each node x the number assigned to x is the smallest nonnegative integer not assigned to any of its neighbors. We show that the computational problem of testing a graph for having a Grundy-numbering or not is NP-complete. As a bonus we obtain a proof of Chvátal's result that testing a graph for having a kernel is NP-complete.

1. INTRODUCTION

Let $G = \langle V,\Gamma\rangle$ be an arbitrary (directed) graph, and let for all $x \in V$: $\Gamma x = \{y \in V | x \to y \in \Gamma\}$ be the set of neighbors of x (as in Berge [2]).

A function $g: V \to \mathbb{N}$ assigning numbers to vertices is a Grundy-numbering if and only if for all x in the graph $g(x)$ is the smallest nonnegative integer not assigned to any element of $\Gamma(x)$ (cf. Berge [2]). The concept of such a numbering originated in the theory of games (Nim), but was introduced for arbitrary graphs by Berge and Schützenberger.

For undirected ("symmetric") graphs is known that having a Grundy-numbering with $\max_{x \in V}\{g(x)\} \leq k$ is equivalent to the graph being k-colorable, at the cost of a polynomial time bounded conversion (Berge [3]). As the latter property is known to be NP-complete (cf. Karp [6]), it immediately follows that finding a Grundy-numbering in undirected graphs is NP-complete.

For directed graphs it can happen that there is no Grundy-numbering at all (see Fig. 1), and we run into a different question. Whereas the problem of finding a Grundy-numbering when there is one cannot possibly be simpler than in the undirected case, there might be a more easily testable global criterion for determining whether or not a graph has a Grundy-numbering at all. We show that in a well-understood sense the existence of such an efficient criterion is unlikely: we prove that testing for a Grundy-numbering still is an NP-complete task.

A set of vertices $S \subseteq V$ is a kernel of G (french: noyau, cf. Berge [3]) if and only if for all x: $x \in S \to \Gamma x \cap S = \phi$ and $x \notin S \to \Gamma x \cap S \neq \phi$.

Some graphs do and other graphs do not have a kernel. Berge [3] showed that if a graph has a Grundy-numbering then it has a kernel (namely:

$S = \{v \varepsilon V | g(v) = 0\}$, but the converse does not always hold.

As a bonus from the construction for proving that Grundy-numbering is NP-complete, we obtain a simple argument that the computational task of testing a graph for having a kernel is NP-complete. (Again, we do not require that an actual kernel be produced once the existence is ascertained). The same result was reportedly also proved by V. Chvátal some time ago ([5]).

For an introduction to NP-complete problems and the relevance for the theory of computing one is referred to Aho, Hopcroft, and Ullman ([1], Ch 10).

2. TESTING FOR A GRUNDY-NUMBERING

Let GRUNDY be the collection of graphs (in a suitably encoded form) which admit a Grundy-numbering.

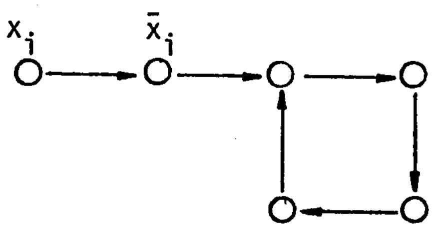
A problem L is called p-reducible to problem M if and only if there is a polynomial time computable transformation f such that for all tested instances α of problem L: $\alpha \varepsilon L \leftrightarrow f(\alpha) \varepsilon M$.

Cook [4] showed that the problem (named: SAT_3) of testing arbitrary propositional formulae in conjunctive normal form with 3 literals per clause for satisfiability is NP-complete. Following a strategy of Karp [6], one may prove NP-completeness of GRUNDY by showing that (i) GRUNDY ε NP and (ii) SAT_3 is p-reducible to GRUNDY.

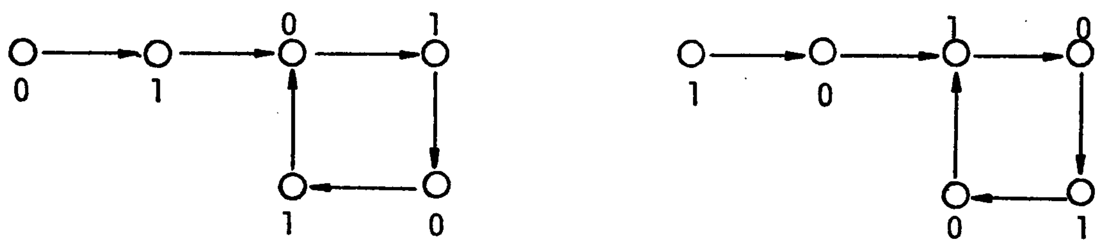
Suppose one must test an arbitrary instance α of SAT_3 for satisfiability, with α containing m clauses and n distinct variables. We shall give a simple, uniform procedure for transforming α into a directed graph G_α with $8n + 3m$ vertices such that α is satisfiable if and only if G_α has a Grundy-numbering.

Given α, our first concern is a compact simulation of all possible truth-value assignments for its variables $x_1,\dots,x_n$.

Consider the graphs



The only possible Grundy-numberings for these graphs are

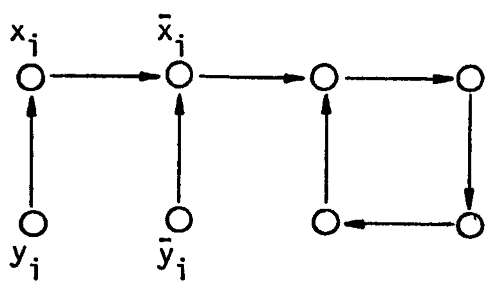


,

and it immediately follows that we can realize (or: obtain) an arbitrary assignment by identifying $val(x_i)$ and $val(\bar{x}_i)$ with $g(x_i)$ and $g(\bar{x}_i)$ respectively (interpreting 0 as "false" and 1 as "true").

It will turn out to be helpful in the construction below that we have access to the complement of a truth-value assignment for the x_i's, and we shall be using graphs

H_i:



,

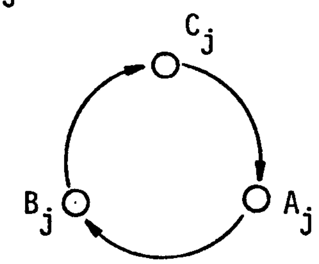
thus forcing that $g(x_i) = 0 \to g(y_i) = 1$ and so on.

<u>THEOREM 2.1</u>: GRUNDY is NP-complete.

<u>Proof</u>: GRUNDY is clearly in NP, because one can simply guess an assignment of non-negative integers (which need not be larger than the number of nodes in the graph, cf. Berge [3]), and verify in polynomial time that at each vertex the condition for a Grundy-numbering holds.

To show that SAT_3 is p-reducible to GRUNDY, lay out the (sub)graphs $H_1,\ldots,H_n$ and continue the construction of G_α as follows.

For each clause C_j of α $(1 \leq j \leq m)$, lay out a subgraph



(identifying C_j with the node shown).

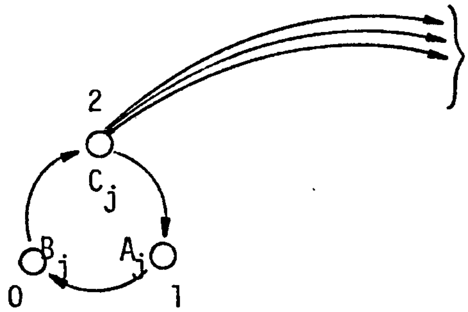
Complete the construction of G_α by drawing an arc from C_j to each y_i or $\bar{y}_i$ such that x_i or $\bar{x}_i$ (respectively) occurs in clause C_j, for each j. An example of the complete construction appears in the Appendix.

We claim that G_α has a Grundy-numbering if and only if α is satisfiable.

Let α be satisfiable, and let the Grundy-numbering for $H_1,\ldots,H_n$ be choosen so as to represent the assignment satisfying α. (Note that the possible numbering of any H_i is not at all affected by the added in-coming edges, i.e., this part of the numbering is not depending on how the remainder of the graph is numbered).

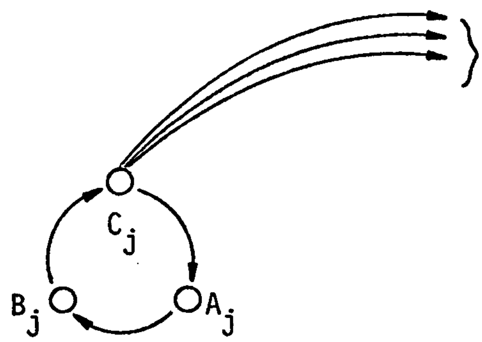
Each clause C_j must contain at least one true literal (some x_i or $\bar{x}_i$), and node C_j must therefore be connected to at least one y_i or $\bar{y}_i$ in G_α which has value 0 assigned.

One can complete the Grundy-numbering of G_α as shown below



Conversely, let us assume that G admits a Grundy-numbering.

Suppose that for some j, C_j is connected to nodes y_i and $\bar{y}_i$ which all have value 1 (recall that they can have either 0 or 1):



If $g(A_j)=0$, then $g(C_j)=2$ and thus $g(B_j)=0$. This contradicts that $g(A_j)$ is the smallest nonnegative integer not assigned to $g(B_j)$.

If $g(A_j)\geq 1$, then $g(C_j)=0$ and thus $g(B_j)=1$. It follows that $g(A_j)$ must be 0, again a contradiction.

We conclude that each C_j must be connected to at least one y_i or $\bar{y}_i$ which has value 0, and that each clause C_j must contain at least one true literal. It follows that α is satisfiable. □

3. TESTING FOR A KERNEL

Let KERNEL be the collection of graphs (in a suitably encoded form) which have a kernel.

The concept of a kernel for directed graphs (cf. Berge [3]) is somewhat similar to the concept of a maximal independent set for undirected graphs. The succint difference is that, whereas an undirected graph always has a maximal independent set (although it may be hard to compute one, see Tarjan & Trojanowski [7]), a directed graph does not always have a kernel.

It is known that the problem of finding a maximal independent set is NP-complete (cf. Karp [6]).

We show here that having a kernel is NP-complete also. It follows that it is unlikely that there is an efficient global criterion for even testing whether a graph has a kernel or not, and there may very well be no more efficient algorithm for it than essentially trying all "reasonable" subsets.

Note that Berge [3] observed: GRUNDY $\subsetneq$ KERNEL, and it follows that testing for a Grundy-numbering is not a conclusive algorithm for the mere existence of a kernel. Nevertheless, in showing that SAT_3 is p-reducible to KERNEL it appears that we can use the very same graphs G_α we had before.

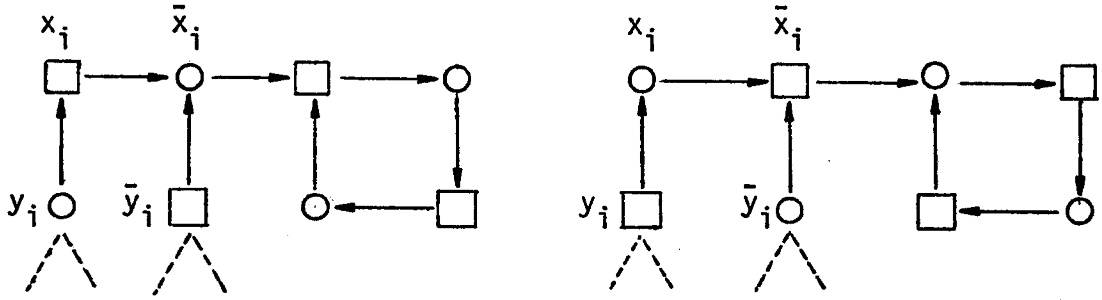
The NP-completeness of KERNEL was reportedly also proved by V. Chvátal some time ago ([5]). The result is included here because it follows in such a natural fashion from our construction for GRUNDY.

<u>THEOREM 3.1</u>: KERNEL is NP-complete.

<u>Proof</u>: KERNEL is clearly in NP, because one can simply guess a subset of V and verify in polynomial time that a node is either in the set while none of its neighbors is or not in the set while at least one of its neighbors is, for all nodes in the graph.

To show that SAT_3 is p-reducible to KERNEL, construct for each instance α of SAT_3 the graph G_α as in 2.1.

As far as determining a kernel is concerned, the subgraphs H_i are again independent of the other parts of the graph. Consider what nodes of H_i can be (or: have to be) in the kernel if G has one at all. Marking kernel-nodes as squares we get as only possibilities

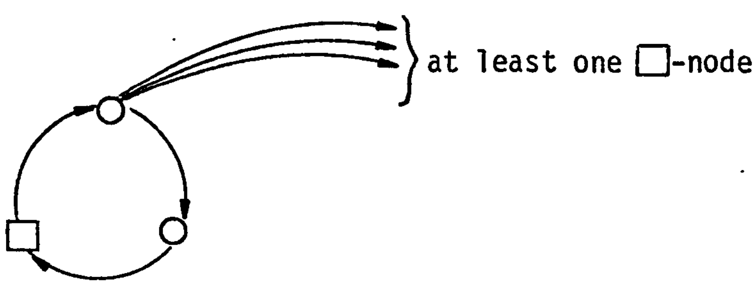


It follows that we can again realize (or: obtain) an arbitrary truth-value assignment by calling a node "false" if it is contained in the kernel, and "true" otherwise.

We claim that G has a kernel if and only if α is satisfiable.

Let α be satisfiable, and let kernel-nodes in $H_1,\ldots,H_n$ be chosen so as to represent the assignment satisfying α.

Each clause C_j must contain at least one true literal, and node C_j is therefore connected to at least one "square" y_i or $\bar{y}_i$. One can complete the kernel by putting



Conversely, let us assume that G has a kernel.

If for some j C_j belongs to the kernel, then neither A_j nor B_j can. This would contradict that a neighbor of A_j is in the kernel.

The only possibility remaining is that B_j belongs to the kernel, which in turn forces that C_j must be connected to at least one node y_i or $\bar{y}_i$ which belongs to the kernel (for each j).

It follows that each clause must contain a true literal, and α is satisfiable. □

REFERENCES

[1] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, The design and analysis of computer algorithms, Addison-Wesley Prubl. Co., Reading, Mass. 1974.

[2] Berge, C., The theory of graphs and its applications, Wiley, New York (1958).

[3] Berge, C., Graphes et hypergraphes, Dunod, Paris (1970).

[4] Cook, S. A., The complexity of theorem-proving procedures, Proc. 3rd Annual ACM Symp. on Theory of Computing (1971) 151-158.

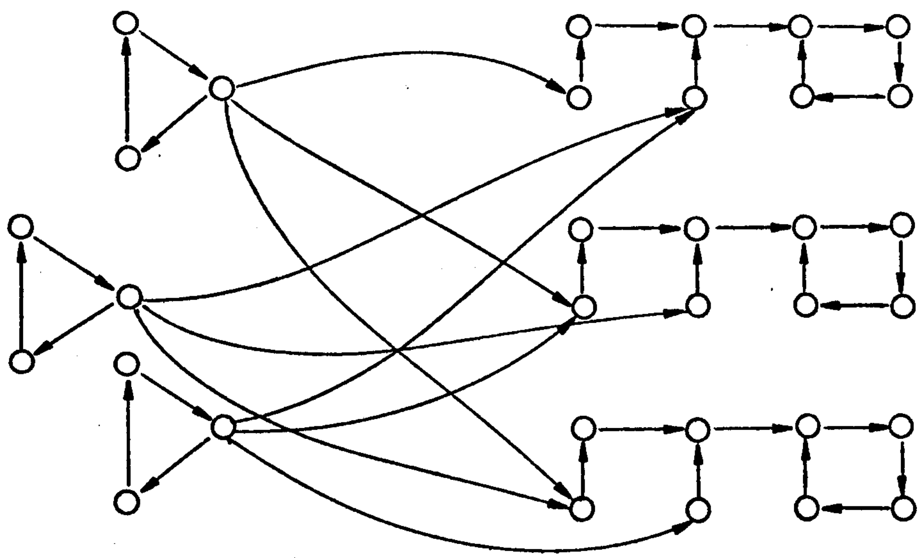
[5] Garey, M. R., private communication (1976).

[6] Karp, R. M. Reducibility among combinatorial problems, in: R. E. Miller and J. W. Thatcher (ed.), Complexity of Computer Computations, Plenum Press, New York (1972) 85-104.

[7] Tarjan, R. E., and A. E. Trojanowski, Finding a maximum independent set, Tech. Rep. STAN-CS-76-550, Stanford University (1976).

APPENDIX

$\alpha = (x_1 \vee x_2 \vee x_3)(\bar{x}_1 \vee \bar{x}_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee \bar{x}_3)$



G_α