# A New Reduction from 3SAT to $n$-Partite Graphs 

Daniel J Hulme<br>Dept. of Computer Science<br>University College London<br>Gower St, London<br>WC1E 6BT<br>T: +44 (0)20 76086840<br>F: +44 (0)20 76084064<br>d.hulme@cs.ucl.ac.uk

Robin Hirsch<br>Dept. of Computer Science<br>University College London<br>Gower St, London<br>WC1E 6BT<br>T: +44 (0)20 76791379<br>F: +44 (0)20 73871397<br>r.hirsch@cs.ucl.ac.uk

R.Beau Lotto<br>Institute of Ophthalmology<br>University College London<br>11-43 Bath St, London EC1V 9EL<br>T: +44 (0)20 76084052<br>F: +44 (0)20 76084064<br>lotto@ucl.ac.uk

Dept. of Computer Science
University College London
Gower St, London
WC1E 6BT
T: +44 (0)20 76797294
b.


#### Abstract

The Constraint Satisfaction Problem (CSP) is one of the most prominent problems in artificial intelligence, logic, theoretical computer science, engineering and many other areas in science and industry. One instance of a CSP, the satisfiability problem in propositional logic (SAT), has become increasingly popular and has illuminated important insights into our understanding of the fundamentals of computation.

Though the concept of representing propositional formulae as $n$-partite graphs is certainly not novel, in this paper we introduce a new polynomial reduction from 3SAT to $G_{7}^{n}$ graphs and demonstrate that this framework has advantages over the standard representation. More specifically, after presenting the reduction we show that many hard 3SAT instances represented in this framework can be solved using a basic path-consistency algorithm, and finally we discuss the potential advantages and implications of using such a representation.


## I. Introduction

The Constraint Satisfaction Problem (CSP) describes a general framework for problems in which values must be assigned to a set of variables subject to specific constraints [1], [2], and is one of the most prominent problems in artificial intelligence (AI), logic, theoretical computer science, engineering and many other areas in science and industry.

One instance of a CSP, the satisfiability problem in propositional logic (SAT), has become increasingly popular and has lead to important insights into the nature of satisfiability and dramatic improvements in CSP algorithms, which can solve hard instances with thousands of variables [3] as well as many restricted instances in polynomial-time [4].

Fundamental to the study of CSP, the $P$ versus $N P$ problem, formulated independently by Stephen Cook [5] and Leonid Levin [6], has been one of the most important scientific questions posed to date. Indeed, over the past several decades researchers have been trying to determine whether or not there is a polynomial solution to any of the problems that have been shown (using polynomial reduction) to be $N P$-complete [7], many of which are described in Computers and Intractability: A Guide to the Theory of NP-completeness [8].

In this paper we introduce a new framework to represent SAT problems and demonstrate that this framework has advantages over the standard representation. More specifically, after introducing some elementary concepts in complexity
and graph theory we present a new polynomial reduction from 3SAT to $G_{7}$. We demonstrate that a basic polynomialtime algorithm can solve a number of hard SAT benchmark instances represented using this framework, yet fails when applied to the standard $G_{3}^{n}$ graph representation. Finally we briefly discuss the potential advantages, practical benefits and possible implications of using such a framework.

## II. BACKGROUND

## A. Satisfiability and the Conjunctive Normal Form

The satisfiability problem in conjunctive normal form (CNF) consists of the conjunction ( $\wedge$ representing the Boolean and connective) of a number of clauses, where a clause is a disjunction ( $\vee$ representing the Boolean or connective) of a number of propositions or their negations.

If $x_{i}$ represent propositions that can assume only the values True or False, then an example formula in CNF would be

$$
\begin{equation*}
\left(x_{0} \vee x_{2} \vee \bar{x}_{3}\right) \wedge\left(x_{3}\right) \wedge\left(x_{1} \vee \bar{x}_{2}\right) \tag{1}
\end{equation*}
$$

where $\bar{x}_{i}$ is the negation of $x_{i}$.
Given a set of clauses $C_{0}, C_{1}, \ldots, C_{n-1}$ on the propositions $x_{0}, x_{1}, \ldots, x_{m-1}$, the satisfiability problem is to determine if the formula $F=\bigwedge_{j<n} C_{j}$ has an assignment of values to the propositions such that it evaluates to True.

## B. $3 S A T$

One of the original problems shown by Cook [5] to be $N P$ complete, 3SAT is considered the 'mother' of all SAT problems. Instances of 3SAT are restricted to Boolean formulae in CNF with three literals per clause. For example, the formula

$$
\begin{equation*}
\left(x_{0} \vee x_{1} \vee x_{2}\right) \wedge\left(\bar{x}_{0} \vee x_{1} \vee \bar{x}_{2}\right) \wedge\left(x_{0} \vee \bar{x}_{1} \vee x_{3}\right) \wedge\left(\bar{x}_{0} \vee \bar{x}_{2} \vee \bar{x}_{3}\right) \tag{2}
\end{equation*}
$$

is a 3 CNF formula with four clauses and is a $Y E S$ instance to 3SAT since the truth assignment $\theta$ satisfies the formula, where one of the nine satisfying assignments is $\theta\left(x_{0}\right)=\theta\left(x_{1}\right)=$ True and $\theta\left(x_{2}\right)=\theta\left(x_{3}\right)=$ False.


Fig. 1. An example of a 4-partite graph.

## C. Satisfiability as a Constraint Satisfaction Problem

Representing a SAT instance as a CSP is specified by giving a formula in propositional logic (such as CNF) and asking whether there is an assignment to the set of propositions which makes the formula True [9].

Example 1: For instance, finding a satisfying truth assignment for Formula 1 can be formulated as a CSP instance. Perhaps the most straightforward way is to construct the instance with a set of:

- Variables $\mathrm{V}=\left\{x_{0}, x_{1}, x_{2}, x_{3}\right\}$.
- Values $\mathrm{D}=\{0,1\}$, corresponding to False and True.
- Constraints $\mathrm{C}=\left\{C_{0}, C_{1}, C_{2}\right\}$, where
$-C_{0}=\left\langle\left\langle x_{0}, x_{2}, x_{3}\right\rangle, D^{3} \backslash\langle 0,0,1\rangle\right\rangle$
- $C_{1}=\left\langle\left\langle x_{3}\right\rangle, D^{1} \backslash\langle 0\rangle\right\rangle$
$-C_{2}=\left\langle\left\langle x_{1}, x_{2}\right\rangle, D^{2} \backslash\langle 0,1\rangle\right\rangle$


## D. Partite Graphs

Definition 1: A graph is $n$-partite iff the vertices can be partitioned into $n$ independent subsets - i.e., no two vertices within the same set are adjacent (connect by an edge).

Fig. 1 is an example of a 4-partite graph, which in this case has four independent sets of vertices: $\{A, B, C\},\{D, E, F\}$, $\{G\},\{H\}$. It should be noted that all graphs in this paper are undirected and irreflexive.

## E. Cliques

Definition 2: In a graph $G$, a clique $C$ is a subset of vertices of $G$ such that every pair of distinct vertices in $C$ are adjacent.

For instance, in Fig. 1 there are several cliques of size three (3-clique), including $\{F, G, H\},\{A, F, G\}$ and $\{B, E, H\}$, with only one 4 -clique $\{A, F, G, H\}$.

## F. The $G_{m}$ Graph Problem

Definition 3: A $G_{m}^{n}$ graph has a partition into $n$ independent sets, each of which contains exactly $m$ vertices. An instance of $G_{m}$ is a $G_{m}^{n}$ graph (for some $n$ ), and is a YES instance if it contains an $n$-clique and a $N O$ instance otherwise.


Fig. 2. Formula 2 reduced to a $G_{3}^{4}$ graph.

## G. Polynomial Reduction

The method of showing that a problem is $N P$-complete by polynomial reduction is one of the most elegant and productive in computational complexity [10]. It is a means of providing compelling evidence that a problem in $N P$ is not in $P$. Cook [11] defines the following:

Definition 4: Suppose that $L_{i}$ is a language over $\Sigma_{i}, i=1,2$. Then $L_{1} \leq_{p} L_{2}\left(L_{1}\right.$ is polynomially reducible to $\left.L_{2}\right)$ iff there is a polynomial-time computable function $f: \Sigma_{1} \rightarrow \Sigma_{2}$ such that $x \in L_{1} \Leftrightarrow f(x) \in L_{2}$, for all $x \in \Sigma_{1}$.

Definition 5: A language $L$ is $N P$-complete iff $L$ is in $N P$, and $L^{\prime} \leq_{p} L$ for every language $L^{\prime}$ in $N P$.

## Proposition 1: Given any two languages, $L_{1}$ and $L_{2}$ :

1) If $L_{1} \leq_{p} L_{2}$ and $L_{2} \in P$ then $L_{1} \in P$.
2) If $L_{1}$ is $N P$-complete, $L_{2} \in N P$, and $L_{1} \leq_{p} L_{2}$ then $L_{2}$ is $N P$-complete.
3) If $L \in P$ and $L$ is $N P$-complete, then $P=N P$.

## H. $3 S A T \leq{ }_{p} G_{3}$

The standard reduction [5, Theorem 2] of a 3CNF formula $F$ (that has a set of $n$ clauses $C$ ) to a $G_{3}^{n}$ graph $G=(V, E)$ such that $G$ has an $n$-clique iff $F$ is satisfiable is as follows:

1) For each clause $C_{k}$ in $F(k<n)$, put a triple of vertices in $V$ respectively labelled by the three literals in $C_{k}$.
2) For each pair of vertices $i, j \in V$ add an edge $(i, j)$ to $E$ iff the vertices are in different triples, and their corresponding literals are not contradictory.

Fig. 2 is the graph of Formula 2 resulting from this reduction. The number of $n$-cliques contained in a $G_{3}^{n}$ graph can be greater than the number of possible satisfying assignments, since it also represents assignments to subsets of literals that also make the formula True. For instance, in Fig. 2 there is a 4-clique between $\left\{x_{0}, x_{0}, \bar{x}_{2}, \bar{x}_{2}\right\}$ and also between $\left\{x_{0}, x_{0}, x_{1}, \bar{x}_{3}\right\}$.

## III. A New Reduction: 3 SAT $\leq{ }_{p} G_{7}$

In a similar way to the standard $G_{3}$ reduction we can reduce 3 SAT to $G_{7}$. In this case, rather than constructing a graph using the literals as vertices, we use the seven possible satisfying assignments to each clause. Two vertices are adjacent iff there are no contradictory assignments to the literals represented by each vertex.

Theorem 1: There is a quadratic-time reduction from 3SAT to $G_{7}$.

Proof: Let $F=\bigwedge_{i<n} C_{i}$ (for some $n$ ) be an instance of 3SAT, where each $C_{i}$ is a disjunction of exactly three literals. We will define an instance of $G_{7}$ from $F$, in polynomialtime. We can assume that no clause contains a literal and its negation (else we could exclude that clause and the result would be logically equivalent). A partial valuation to a clause is a valuation defined on the propositional variables occurring in that clause only. Given a clause $C=\left(l_{0} \vee l_{1} \vee l_{2}\right)$ there are at most seven partial valuations $v$ to $\left\{l_{0}, l_{1}, l_{2}\right\}$ such that $v(C)=\top$ (if the propositional variables in the literals are all distinct then there is one valuation making all three literals false and seven other valuations making at least one literal true; if the propositional variables are not distinct then there will be less than seven partial valuations).

For each clause $C_{i} \quad(i<n)$ and each partial valuation $v$ making $C_{i}$ true, create a node $(i, v)$ of a new graph $G$. $G$ has at most $7 \times n$ nodes - seven for each clause. To complete the definition of our reduction we must define the edges of $G$. Let $((i, v),(j, w))$ be an edge of $G$ if $i \neq j$ and $v$ and $w$ do not contradict each other - i.e., we allow this edge so long as there is no propositional variable $p$ such that $v$ and $w$ are both defined on $p$ but one makes $p$ true and the other makes it false. It is easy to see that this graph $G$ is a $G_{7}^{n}$ graph, hence an instance of $G_{7}$.

Now we must check that the reduction is correct. If $F$ is a $Y E S$ instance of 3SAT, let $v$ be a valuation such that $v(F)=\top$. We must show that there is a $n$-clique of the graph $G$. For each clause $C_{i}$, let $v_{i}$ be the restriction of $v$ to the propositional variables in $C_{i}$. Since $v(F)=\top$ we must have $v_{i}\left(C_{i}\right)=$ $v\left(C_{i}\right)=\top$, so $\left(i, v_{i}\right)$ is a node of $G$. Let $S=\left\{\left(i, v_{i}\right)\right.$ : $i<n\}$. Since all the $v_{i}$ 's are restrictions of the same global valuation $v$, none of them can contradict each other. Hence, for any $i, j<n, \quad\left(\left(i, v_{i}\right),\left(j, v_{j}\right)\right)$ is an edge of $G$. Therefore $S$ is a $n$-clique, so $G$ is a $Y E S$ instance of $G_{7}$. Conversely, suppose $G$ is a $Y E S$ instance of $G_{7}$, so let $S$ be a $n$-clique of $G$. For each $i<k$ there must be one vertex $(i, v) \in S$. We have $\left(v\left(C_{i}\right)=\top\right.$. Let $w$ be the valuation, defined on all propositions $p$ in $F$ by $w(p)=v(p)$ if there is $i<n$ such that $v$ is defined on $p$ and $(i, v) \in S$. Since $S$ is a clique, no two partial valuations $(i, v),\left(j, v^{\prime}\right)$ contradict each other, so this is well-defined. Now, for each $(i, v) \in S, v$ is a restriction of $w$, hence $w\left(C_{i}\right)=v\left(C_{i}\right)=\top$ and therefore $w(F)=\top$, as required.

Creating upto $7 n$ nodes takes $O(n)$ time. Checking if $v$ contradicts $v^{\prime}$ and adding an edge from $(i, v)$ to $\left(j, v^{\prime}\right)$ takes constant time. Adding all the edges takes $O\left(n^{2}\right)$ time.


Fig. 3. Formula 2 reduced to a $G_{7}^{4}$ graph.

To illustrate this using an example, Fig. 3 represents the graph generated from the 3 CNF Formula 2:
$\left(x_{0} \vee x_{1} \vee x_{2}\right) \wedge\left(\bar{x}_{0} \vee x_{1} \vee \bar{x}_{2}\right) \wedge\left(x_{0} \vee \bar{x}_{1} \vee x_{3}\right) \wedge\left(\bar{x}_{0} \vee \bar{x}_{2} \vee \bar{x}_{3}\right)$
In the following example, the subscript to $x_{i}$, indicates the index of the literal, with the superscript to it $x^{a}$, as its Boolean assignment - i.e., $x_{2}^{0}$ states that the literal $x_{2}$ is assigned the Boolean value ' 0 ', representing False.

Example 2: To construct the $G_{7}^{n}$ graph of this Formula:

- Convert each partial evaluation of the clause to an 'assignment vertex' (except the unsatisfiable assignment i.e., $\left\{x_{0}^{0}, x_{1}^{0}, x_{2}^{0}\right\}$ in the first clause).
- If necessary, also remove any 'internally' inconsistent vertcies, e.g., $\left\{x_{0}^{0}, x_{1}^{0}, x_{0}^{1}\right\}$ could not exist since the assignment $x_{0}^{0}$ contradicts $x_{0}^{1}$.
- Add edges between vertices that are not contradictory, for instance:
- Create an edge between $\left\{x_{0}^{0}, x_{1}^{0}, x_{2}^{1}\right\}$ and $\left\{x_{0}^{0}, \bar{x}_{1}^{1}, x_{3}^{1}\right\}$.
- Do not create an edge between $\left\{x_{0}^{0}, x_{1}^{0}, x_{2}^{1}\right\}$ and $\left\{x_{0}^{0}, \bar{x}_{1}^{0}, x_{3}^{0}\right\}$ (because the assignment $x_{1}^{0}$ contradicts $\left.\bar{x}_{1}^{0}\right)$.

Using this framework the resulting $G_{7}^{n}$ graph only represents the $n$-cliques that correspond to each possible satisfying assignment - i.e., the number of $n$-cliques exactly equals the number of assignments that make a formula True. This means that since a $G_{3}^{n}$ graph can represent solutions using subests of literals, it can contain more $n$-cliques than the total number of $n$-cliques represented by the corresponding $G_{7}^{n}$ graph.

## IV. Discussion

The primary focus of this paper is to introduce a new reduction from 3SAT to $G_{7}$, however, to illustrate one benefit of using such a representation, we ran a very basic complete pathconsistency algorithm [2] on a number of 3CNF unsatisfiable SATLIB benchmarks reduced to $G_{7}$ and $G_{3}$ graph problems.

SATLIB [3] is an online resource for SAT-related research with its core component, a freely distributed benchmark suite of SAT instances and a collection of SAT solvers, aimed to facilitate empirical research on SAT by providing a uniform test-bed for SAT solvers.

Generally, it tends to be 'hard' for polynomial algorithms to correctly solve unsatisfiable instances, so these algorithms are usually used preliminarily to reduce the search-space for backtracking algorithms [12]. To clarify what we mean by 'solve', the path-consistency algorithm simply prunes edges that cannot be part of a tri-clique and it runs in polynomialtime, $O\left(n^{2}\right)$. Since both the $G_{7}^{n}$ and standard $G_{3}^{n}$ representations contain an $n$-clique iff there is a satisfiable solution, an empty graph (containing no edges) means that there are no cliques and hence no satisfiable solutions. Therefore, initially for our purpose it is interesting to focus primarily on unsatisfiable benchmarks.

Table I lists some unsatisfiable benchmark instances that this algorithm successfully solves when represented as a $G_{7}^{n}$ graph (including the time taken). Moreover, if we reduce these instances to the standard $G_{3}^{n}$ representation, the simple pathconsistency algorithm fails to solve a single case correctly.

Uniform Random-3SAT (UUF) is a family of SAT problems obtained by generating 3CNF formulae, randomly drawing from the $2 n$ possible literals with uniform probability. The AIM and DUBOIS instances are constructed with Random3SAT instance generators and run in a randomized fashion. Although some SAT-solvers find it difficult to solve the AIM instances, it should be noted that these instances can be solved with polynomial preprocessing.

The results are very encouraging, and it is speculated that this new framework is advantageous over the standard representation since the $G_{7}^{n}$ graphs represent less information than the $G_{3}^{n}$ graphs (i.e., the number of $n$-cliques equals the number of satisfying assignments), resulting in graphs which tend to be significantly less dense with proportionally up to 10 times fewer edges.

Indeed, we intend to further this research by attempting to understand why these polynomial-time algorithms fail to solve some unsatisfiable formulae (graphs with no $n$-cliques) as well as applying more robust complete polynomial-time algorithms [4] to many more of the SATLIB benchmarks and other pertinent scientific problems.

Though still unproven, the general consensus is that $P \neq$ $N P$ [13]. From our preliminary findings for 3SAT, it would appear that even the most basic of polynomial-time algorithms may work for some of the inputs likely to be encountered in practice. In this sense, using more robust algorithms thus might yield many of the stunning practical benefits to be expected in a world in which $P=N P$ [11].

TABLE I
SOME SOLVED UNSATISFIABLE BENCHMARKS REPRESENTED AS $G_{7}^{n}$ GRAPHS USING A BASIC PATH-CONSISTENCY ALGORITHM, WHICH FAILS to solve the $G_{3}^{n}$ Graphs correctly. Time is in seconds.

| benchmark set | instances | props | clauses | solves | avg time |
| :--- | :---: | :---: | :---: | :---: | :---: |
| uuf50-218 | 1000 | 50 | 218 | 1000 | 1.16 |
| uuf75-325 | 100 | 75 | 325 | 55 | 25.05 |
| uuf100-430 | 1000 | 100 | 430 | 4 | 120.45 |
| dubois\# | 11 | $60-90$ | $160-240$ | 11 | $0.24-0.75$ |
| dubois100 | 1 | 300 | 800 | 1 | 30.2 |
| aim-50-1_6-no | 4 | 50 | 80 | 4 | 0.032 |
| aim-50-2_0-no | 4 | 50 | 100 | 4 | 0.062 |
| aim-100-1_6-no | 4 | 100 | 160 | 4 | 0.27 |
| aim-100-2_0-no | 4 | 100 | 200 | 4 | 0.40 |
| aim-200-1_6-no | 4 | 200 | 320 | 4 | 1.85 |
| aim-200-2_0-no | 4 | 200 | 400 | 4 | 3.00 |

## Acknowledgment

Daniel J Hulme wishes to acknowledge the support of an EPSRC Engineering Doctorate from the VEIV EngD Programme at UCL.

## REFERENCES

[1] V. Kumar, "Algorithms for constraint-satisfaction problems: A survey," AI Magazine, vol. 13, no. 1, pp. 32-44, 1992.
[2] E. Tsang, Foundations of Constraint Satisfaction. London: Academic Press, 1993.
[3] H. H. Hoos and T. Stützle, "Satlib: An online resource for research on SAT," in SAT'2000. IOS Press, 2000, pp. 283-292.
[4] J. K. Pearson and P. G. Jeavons, "A survey of tractable constraint satisfaction problems," Royal Holloway University of London, Tech. Rep. CSD-TR-97-15, 1997.
[5] S. A. Cook, "The complexity of theorem-proving procedures," in STOC '71: Proceedings of the third annual ACM symposium on Theory of computing. New York, NY, USA: ACM Press, 1971, pp. 151-158.
[6] B. A. Trakhtenbrot, "A survey of Russian approaches to perebor (bruteforce search) algorithms," Annals of the History of Computing, vol. 6, no. 4, pp. 384-400, Oct./Dec. 1984, partial English translation of L. Levin, Universal Search Problems, 9(3), pp. 265-266, (1973).
[7] M. Sipser, "The history and status of the P versus NP question," in In Proceedings of the 24th ACM Symposium on Theory of Computing, 1992, pp. 603-618.
[8] M. R. Garey and D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman \& Co., 1990.
[9] P. Jeavons, D. Cohen, and J. Pearson, "Constraints and universal algebra," Annals of Mathematics and Artificial Intelligence, vol. 24, no. 1-4, pp. 51-67, 1998.
[10] L. Adleman and K. Manders, "Reducibility, randomness, and intractibility (abstract)," in STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing. New York, NY, USA: ACM Press, 1977, pp. 151-163.
[11] S. A. Cook, "The P versus NP problem," 2000, computer Science Department, University of Toronto.
[12] J. Gu, P. Purdom, J. Franco, and B. Wah, "Algorithms for the satisfiability (sat) problem: a survey," 1996.
[13] L. A. Hemaspaandra, "Sigact news complexity theory column 36," SIGACT News, vol. 33, no. 2, pp. 34-47, 2002.

