dkgreenrgb0,0.6,0 grayrgb0.5,0.5,0.5 mauvergb0.58,0,0.82

CS3052 Poly Reduction Report

170011474

April 2020

1 Overview

This practical requires students to implement 3 reductions: from SAT to 3-SAT, from 3-SAT to K-colour problem, from K-colour problem to SAT. Additional extension is finished: from 3-SAT to vertex cover problem.

2 Input and Output Format

The input format for SAT is `Dimacs` CNF format, and the input format for graph-related problems is `Dimacs` graph format from http://prolland.free.fr/works/research/dsat/dimacs.html with colours specified.

```
c FILE: myciel3.col
c SOURCE: Michael Trick (trick@cmu.edu)
p edge 11 20
colours 3
e 1 2
e 1 4
e 1 7
```

3 SAT $\leq_p$ 3-SAT

3.1 Algorithm

The algorithm is implemented based on Lecture 13. It iterates over each clause in the formula until it meets a clause containing more than 3 literals. Splitting the clause into 2 shorter clause with a dummy literal added.

3.2 Polynomial Time Proof

The program is guaranteed to terminate because each split causes 2 shorter clauses. The reduction is polynomial because the number of splits needed is dependent on the length of the formula. So it is in $O(n)$. Besides, each split needs $O(n)$ to go over the length of the current clause. So altogether $O(n^2)$ which is in polynomial time.

3.3 Correctness Proof

Let us assume there is a formula $C = x_1 \vee x_2 \vee x_3 \vee x_4$ and converts it into $C_1 = (x_1 \vee x_2 \vee d) \wedge (\neg d \vee x_3 \vee x_4)$. Now we need to prove that the satisfiability is consistent. This means each split preserves the original satisfiability.

Firstly we show that a satisfying assignment for C leads to a satisfying assignment for C_1. If C is satisfiable, then at least one literal l is true. The literal l is either in the first clause in C_1 or in the second clause. If it is in the first clause, we can set $d = 0$ and thus $\neg d = 1$. So the second clause is also satisfiable as well the first clause. Therefore, C_1 is also satisfiable.

Secondly we show that a satisfying assignment for C_1 leads to a satisfying assignment for C. If C_1 is satisfiable, there are 2 possible cases: $d = 0$ or $d = 1$ in the first clause. If $d = 0$, then there is at least one literal l in the first clause is true. If $d = 1$, then there is at least one literal l in the second clause is true. So there is always a true literal except the dummy literal d, always resulting a satisfiable clause C.

3.4 Testing

We can manually construct an unsatisfied CNF formula. If we have n variables, then the unsatisfied formula for it has 2^n clauses. Those clauses are all possible combinations of variables. For example, if $n = 2$, then those 4 clauses are $(1 \vee 2) \wedge (1 \vee -2) \wedge (-1 \vee 2) \wedge (-1 \vee -2)$ and this formula must be unsatisfied. In this way, from any k-SAT formula to 3-SAT formula, we check if the 3-SAT instance as result of conversion is also unsatisfied. The SAT solver used here is at https://www.comp.nus.edu.sg/ gregory/sat/. After pressing the button, red colour represents unsatisfied and green colour means satisfied. To generate negative instances automatically, a `NegativeSATGenerator` is implemented in the source code.

In the implementation of the reduction, all negative instances is converted into negative instances in 3-SAT problems. In other words, if the instance before conversion is shown as red in the online solver, then the result of conversion is also shown as red in it.

All positive instances downloaded from https://www.cs.ubc.ca/ hoos/SATLIB/benchm.html are turned into positive ones in 3-SAT. In other words, instances before conversion and results are all shown in green in the online solver.

4 3-SAT $\leq_p$ K-colour

4.1 Algorithm

The algorithm consists of 2 steps (Sharma and Chaudhari, 2012): the first one is to reduce 3-SAT to 3-colour, and the second one is to reduce 3-colour to K-colour Problem.

In the first step, each variable has a triangle and each clause has a triangle. All triangles of variables have a common vertex of B (base vertex which preempts one colour). Each clause has 3 nodes to represent 3 literals and they are only connected to the literals the clause has. For example, if the formula is $(x \vee y \vee z') \wedge (x \vee y' \vee z')$ then the resulting 3-colour graph is as Figure 1.

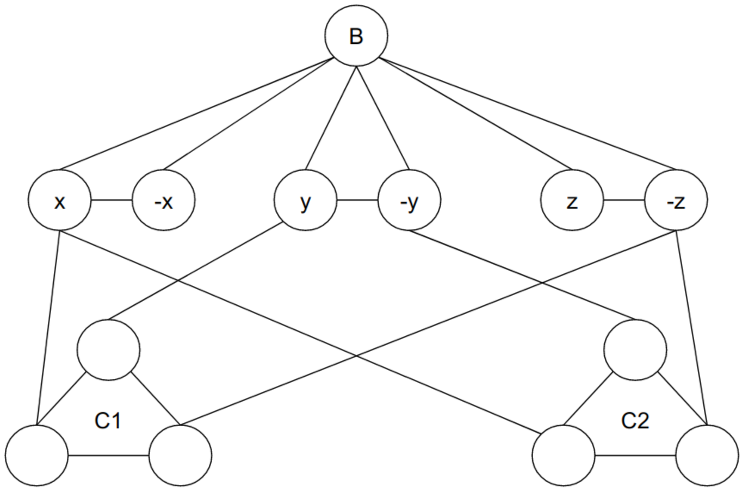


Figure 1: 3-colour

In the second step, we would construct $k-2$ base vertices to preempt $k-2$ colours. Those extra base vertices would connect to every other nodes in the graph. For example, in the same case as above, if we have $k = 4$, then the resulting graph is in Figure 2.

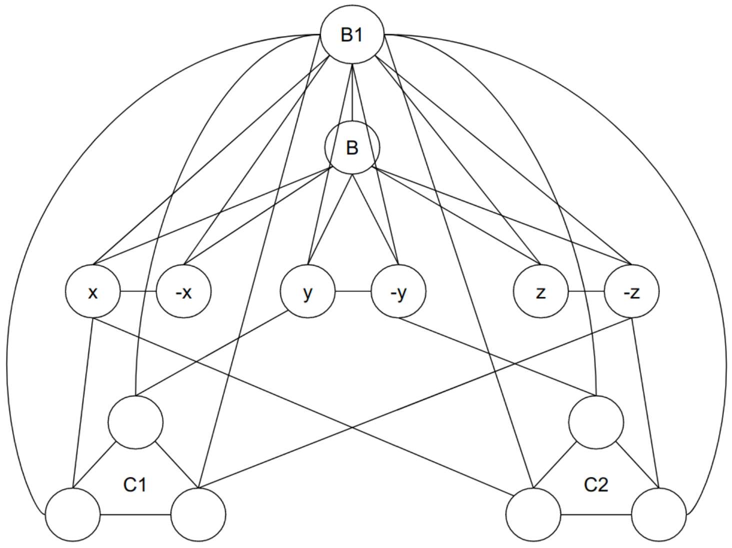


Figure 2: 4-colour

4.2 Polynomial Time Proof

To construct a corresponding graph, the number of vertices $|V| = 2n+3m+(k-2)$, where n is the number of variables, m is the number of clauses, $(k-2)$ is the number of base vertices. For example, if we have a variable x then there is also a negative variable $-x$, so $2n$ variable vertices and they takes $O(n)$. $3m$ is the number of vertices in clauses and it takes $O(m)$. $k-2$ is a constant so $O(1)$. Therefore, constructing vertices is in $O(n)+O(m)$, in polynomial time.

The number of edges $|E| = n + \binom{k-2}{2} + 2n(k-2) + 3m(k-3) + 6m$, where n is the number of edges between positive variables and negative variables, $\binom{k-2}{2}$ for connecting each base vertex with every other vertex, $2n(k-2)$ for connecting base vertices with variables and negative variables, $3m(k-3)$ for connecting extra base vertices with all clause vertices, $6m$ for connecting clause vertices with each other and connecting them to corresponding variable vertices.

n for $O(n)$. For $\binom{k-2}{2}$, when you want $\binom{n}{k}$ from Pascal's triangle as a matrix, the complexity of building that matrix up to size $n*k$, since you can use $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ to simplify it, then it would be $O(nk)$. In this case, $O((k-2)*2)$ so still $O(1)$ since k is constant. $2n(k-3)$ is $O(n)$, $3m(k-3)$ is $O(m)$, and $6m$ is $O(m)$. Therefore, $O(n)+O(1)+O(n)+O(m)+O(m)$ so $O(n)+O(m)$, which is in polynomial time.

Additionally, since NP is equivalent to PolyCheck, we can simply walk the graph and make certain that all adjacent vertices have a different colour, and make certain that only k colours are used. It is bound by $O(|V|+|E|)$.

4.3 Correctness Proof

The property of the newly constructed graph is: a clause triangle can be coloured if and only if not all three vertices are given same colour. In other words, the 3 literal vertices the clause are connected are not the same truth values.

4.3.1 Prove 3-SAT $\leq_p$ 3-colour

Firstly we prove that if 3-SAT instance ϕ is satisfiable then the graph G is 3-colourable. If ϕ is satisfiable and let$(x_1, x_2, ..., x_n)$ be the satisfying assignment. There should not be a clause (x_i, x_j, x_k) where all of them are false (the same colour). In this case, all clause triangles are 3-colourable. Therefore, the Graph G is 3-colourable.

Secondly if G is 3-colourable, then there should not be any clause with all 3 literals are false (the same colour). Therefore, the 3-SAT instance ϕ must be satisfiable.

4.3.2 Prove 3-colour $\leq_p$ K-colour

By induction, we can show that G_k is k-colourable if and only if G_{k-1} is (k-1)-colourable.

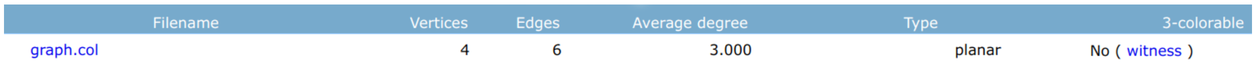
Assume G_{k-1} is (k-1)-colourable. Therefore, G_k is k-colourable because the added base vertex B_{k-3}, which is connected to every other vertices, can preempt k_{th} colour.

Assume G_k is (k)-colourable. Because the base vertex B_{k-3} is connected to all other vertices, B_{k-3} must be the only vertex with a certain colour. Therefore, other nodes are coloured with 1 of $(k-1)$ colours. Therefore, G_{k-1} is (k-1)-colourable.

By chaining this rule, we can conclude that 3-colour $\leq_p$ K-colour.

4.4 Testing

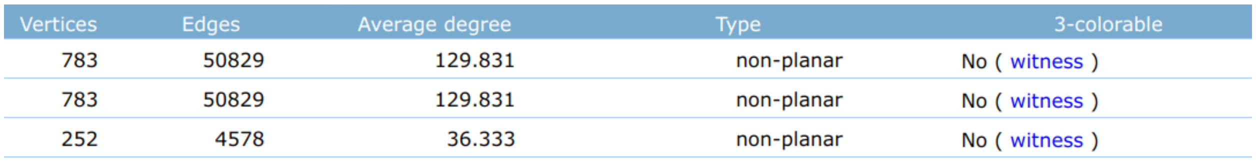
Since $3-colour \leq_p K-colour$ and also $K-colour \leq_p 3-colour$ from Karp's theorem, we can verify an instance's K-colourability by verifying its' 3-colourability. From an online solver for 3-colour problem https://graph-colouring.appspot.com/, after uploading a `.col` file containing a graph, the result would show "no" if the graph cannot be coloured in Figure 5:



Filename	Vertices	Edges	Average degree	Type	3-colorable
graph.col	4	6	3.000	planar	No (witness)

Figure 3: Online Solver for Graph colour

To test correctness, we can use unsatisfied SAT instances generated by `NegativeSATGenerator` and convert them into graphs. Theoretically, those graphs cannot be coloured and we can verify them by the online solver.



Vertices	Edges	Average degree	Type	3-colorable
783	50829	129.831	non-planar	No (witness)
783	50829	129.831	non-planar	No (witness)
252	4578	36.333	non-planar	No (witness)

Figure 4: Negative Instances from 3-SAT to Graph colour when using n = 4, 5 in NegativeSATGenerator

Notice: you can directly run `CircularTest` and then upload the `output.col` to the online solver.

5 K-colour $\leq_p$ SAT

5.1 Algorithm

According to Dey and Bagchi in 2013,a graph with k colour as a constraint can be converted to 3-SAT problems through 3 types of clauses. Type 1 clauses ensure that 2 adjacent nodes do not share the same colour. Type 2 clauses ensure that each node is assigned at least one colour. Type 3 clauses ensure that each node is assigned at most one colour. Each node would be represented by a number of nodes and the number is the number of colour k.

All 3 types of clauses are some constraints. Type 1 clauses belong to **edge constraint** and type 2 and type 3 clauses belong to **vertex constraints**. In the following notations, v_i means type 1 and type 2 clauses for vertex i.

5.2 Polynomial Time Proof

Assume that the number of nodes is denoted by n, and the number of colours is denoted by k.

From the paper, the number of type 1 clauses is number of nodes in the graph multiplied by the number of colours, which is $O(nk)$. The number of type 2 clauses is number of nodes in the graph, which is $O(n)$. The number of type 3 clauses is number of nodes in the graph multiplied by the number of colour, which is $O(nk)$. The number of new variables needed is number of nodes multiplied by the number of colour, which is $O(nk)$. All together, the time is $O(nk) + O(n) + O(nk) + O(nk) = 3 * O(nk) + O(n)$. Ignoring constants and considering the dominants, then the running time is $O(nk) + O(n)$.

5.3 Correctness Proof

Assume that an undirected graph $G(V, E)$ that is k-colourable and the following is a SAT formula corresponding to the graph G:

$$F = (v_i \wedge e_j)$$

where $v_i = v_1, v_2, ..., v_n$ are n vertices and $e_j = e_1, e_2, ..., e_m$ are m edges. The formula F can be expanded as

$$F = F_v \wedge F_e = ((v_1 \wedge v_2 \wedge ... \wedge v_n) \wedge (e_1 \wedge e_2 \wedge ... \wedge e_m))$$

where F_v is the $(v_1 \wedge v_2 \wedge ... \wedge v_n)$ and F_e is the $(e_1 \wedge e_2 \wedge ... \wedge e_m)$.

Firstly we need to show that if the graph is k-colourable, then the SAT formula is satisfying. This is obvious, because if the graph is k-colourable, it means that 2 endpoints on an edge do not share the same colour, which satisfies each e_j clause. Besides, it also means that every vertex has at most one colour and at least one colour, which gives type 2 clauses and type 3 clauses for every vertex v_i a satisfying assignment. Therefore, every clause in the formula F is satisfying and so is F.

Secondly we need to show that if formula F is satisfiable, then the graph is k-colourable. If F is satisfiable, then F_v is satisfiable and F_e is satisfiable. In this case, every v_i is satisfied so every vertex has exactly one colour. Similarly, every e_j is satisfied so every edge has 2 differently coloured endpoints. Therefore, the graph is k-colourable.

5.4 Testing

Based on successful reductions provided above (SAT $\leq_p$ 3-SAT and 3-SAT $\leq_p$ K-colour), we can use `NegativeSATGenerator` to generate negative instances and reduce them to negative instances of K-colour graphs. Afterwards, those graphs are converted into SAT instances again and we check if they are all still negative.

By the online SAT solver with link above, all negatives instances of graphs are converted to negative SAT instances by showing red colours.

6 Extension: 3-SAT $\leq_p$ Vertex Cover

6.1 Algorithm

It takes 2 steps, 3-SAT $\leq_p$ Clique, and Clique $\leq_p$ Vertex Cover. They are all derived from Introduction to Algorithm. In the first part, if we have a 3-SAT instance $\phi = C_1 \wedge C_2 \wedge ... \wedge C_k$.

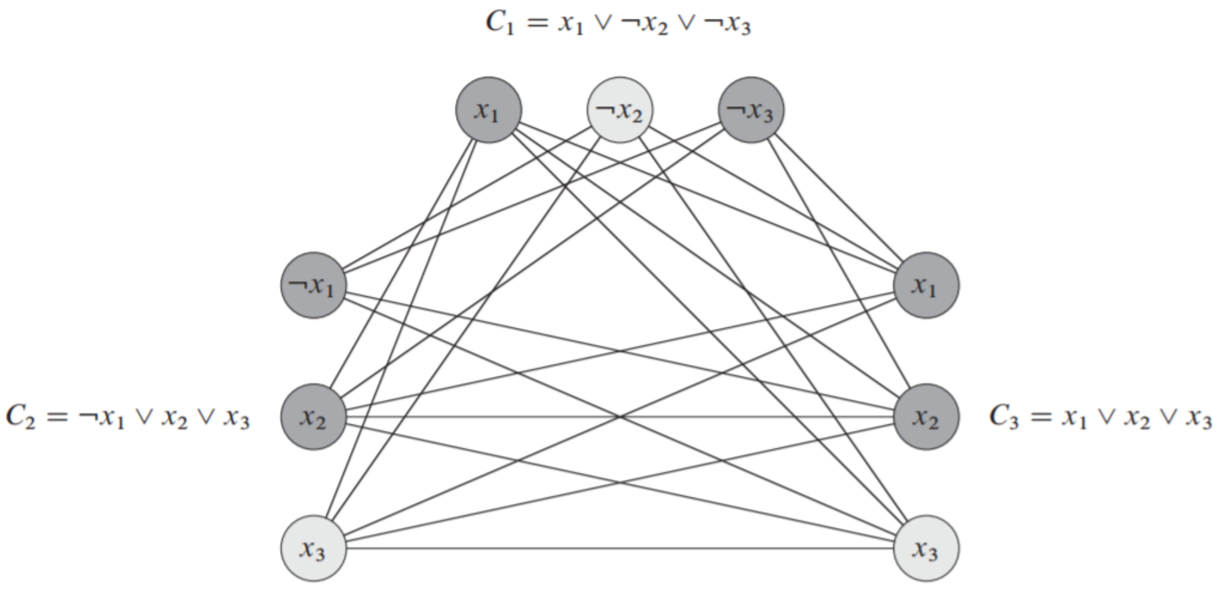

$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$
x_1
$\neg x_2$
$\neg x_3$
$C_2 = \neg x_1 \vee x_2 \vee x_3$
$\neg x_1$
x_2
x_3
$C_3 = x_1 \vee x_2 \vee x_3$
x_1
x_2
x_3

Figure 5:

For each clause C_r, we construct a triple with at most 3 nodes to represent literals in the clause. Edges are placed between 2 nodes if and only if the 2 nodes:

1. They are in different triples (clauses).
2. They are not negation of each other.

In the second part, simply constructs a complement graph G' of the graph G in Clique problem, which is the graph containing exactly those edges that are not in G.

6.2 Polynomial Time Proof

6.2.1 Clique $\in NP$

For a given graph $G = (V, E)$ where V is the collection of all vertices and E is the collection of all edges. Let the set $V' \subseteq V$ in the clique as a proof of the problem. We can check whether V' is a clique in polynomial time by verifying if, for each pair $u, v \in V'$, the edge $(u, v) \in E$. The time required is bounded to the number of vertices except the current triple's vertices. So Clique $\in NP$.

6.2.2 Vertex Cover $\in NP$

Given a graph $G = (V, E)$ and an integer k. Let the set $V' \subseteq V$ be a proof of the problem. After checking $|V'| = k$, it checks for each edge (u, v) in E, if $u \in V'$ or $v \in V'$. This can be verified in polynomial time. Since the maximum number of edges a graph can have is $\binom{|V|}{2}$ and this can be proved in $O(n)$ as above in Section 4.2. Since the operation of searching whether vertex $v \in V'$ is also requires $O(n)$, so $O(n^2)$.

6.3 Correctness Proof

6.3.1 3-SAT $\leq_p$ Clique

Firstly, we can show that if ϕ has a satisfying assignment, then graph G has a clique with size k, where k is the number of clauses. Assume that ϕ has a satisfying assignment. Each clause C_r contains at least one true literal, and each such literal corresponds to a vertex v_i^r. If we pick one such true literal vertex from every clause, then a set V' of k vertices come up. This set V' is a clique with size k, because we already know the rules of constructing graph G is that 2 endpoints of an edge cannot be in the same clause. Since they are all mapped to true, and thus the literals are not complements. Therefore, every edge $(v, u) \in E$ where $v, u \in V'$.

Secondly, we can show that if graph G has a clique with size k, then ϕ has a satisfying assignment. Assume that graph G has a clique V' with size k. Since there is no edge in G connect vertices in the same triple, V' contains exactly 1 vertex in each triple. By assigning true to each literal vertex $v \in V'$ without fear of assigning true to both a literal and its negation, since G has no edges connecting complement literal vertices. In this way, each clause (triple) is satisfied and so ϕ is satisfied.

6.3.2 Clique $\leq_p$ Vertex Cover

We will prove that the graph G has a clique of size k if and only if the graph G' has a vertex cover of size $|V| - k$. Firstly, we can show that if G has a clique $V' \subseteq V$ with $|V'| = k$. We claim that $V - V'$ (the complement of set V' in V) is a vertex cover in G'. Let (u, v) be any edge in $E - E'$. Then $(u, v) \notin E$, which means that at least one of u and v does not in V', since every pair of vertices in V' is connected by an edge of E. Equivalently, at least one of u and v is in $V - V'$. which implies that edge (u, v) is covered by $V - V'$. Since (u, v) was chosen arbitrarily from $E - E'$, every edge of $E - E'$ is covered by a vertex in $V - V'$.

Hence, the set $V - V'$ with size $|V| - k$ forms a vertex cover for G'.

Secondly, we can show that if G' has a vertex cover $V' \in V$ where $|V'| = |V| - k$, for all $u, v \in V$, if $(u, v) \in E - E'$, then $u \in V'$ or $v \in V'$ or both. The contrapositive of this implication is that for all $u, v \in V$, if $u \notin V'$ and $v \notin V'$, then $(u, v) \in E$. In this case, $V - V'$ is a clique with size $|V| - |V'| = k$.

6.4 Testing

References

[1] Dey, M. and Bagchi, A., 2013. *Satisfiability Methods For Colouring Graphs.* [online] https://www.researchgate.net. Available at: https://www.researchgate.net/publication/268460124_Satisfiability_Methods_for_Colouring_Graphs/fulltext/5721424d08ae0926eb45bd3f/Satisfiability-Methods-for-Colouring-Graphs.pdf [Accessed 23 April 2020].

[2] Sharma, P. and Chaudhari, N., 2012. *A New Reduction From 3-SAT To Graph Kcolourability For Frequency Assignment Problem.* [online] Pdfs.semanticscholar.org. Available at: https://pdfs.semanticscholar.org/1b7d/e31398887fe79f5b363a62ab493672027293.pdf [Accessed 3 May 2020].