On modeling connectedness in reductions from graph problems to extended satisfiability

Ricardo Tavares de Oliveira, Fabiano Silva, Bruno Ribas, and Marcos Castilho

Departamento de Informática, Universidade Federal do Paraná
POBox 19081, 81531-980, Curitiba, Brazil
{rtoliveira,fabiano,ribas,marcos}@inf.ufpr.br
http://www.inf.ufpr.br

Abstract. In this paper we present an efficient way to encode connectedness in reductions from graph problems to SAT and MaxSAT. We show and prove linear reductions from Minimum Path and Clique and a quadratic reduction from Steiner Tree, although others NP-complete and NP-hard problems can be reduced with this complexity as well. These reductions use a new class of operators that extends the traditional set of connectors of propositional logic.

Keywords: Polynomial reductions, satisfiability, maximum satisfiability, Steiner Tree, Clique

1 Introduction

Due to the big effort put on SAT and MaxSAT solvers in the last years, the interest on reductions from many NP problems to (Max)SAT has grown, mainly hard problems from the Graph Theory.

However, it's not trivial to reduce problems in graphs that requires the solutions to be *connected* in an efficient way. There are formulas easily built that states the requirement of *complete* connected subgraphs (cliques), as shown in [3]. These reductions are linear in the size of the complement of the given graph, which can be large for sparse graphs. Also, creating a boolean formula that states that a solution must be a connected graph, but not necessarily a complete one, is not trivial.

One can build a formula that states that a certain graph must be a (connected) tree by modeling a depth-first transversal [1]. It's not trivial to built such a formula efficiently. Also, approximated reductions encode connectedness by enumerating some (not all) paths between pair of vertices [2]. These reductions require a large number of encoded paths to guarantee a solution.

We suggest a simpler way to built such reductions by allowing the resulting formula to have operators other than the classic ones ($\neg, \wedge, \vee$). It's possible to convert these new operators to formulae with the classic ones in linear time. Also, these operators can be implemented on non clausal SAT solvers. With them, we reduce Steiner Tree and Hamiltonian Cycle, for instance, in quadratic time in the size of the given graph. Also, we reduce the decision version of Clique in linear time in the size of the given graph.

Section 2 presents the new defined operators. Section 3 shows reductions from graph problems to SAT or MaxSAT. Section 4 shows experimental results, and section 5 concludes and describes some future works.

2 Definition of the new used operators

Let $\mathbb{B} = \{0, 1\}$ be the set of truth values, where 0 is equivalent to $false$ and 1 is equivalent to $true$. An operator of arity k is a function $o : \mathbb{B}^k \to \mathbb{B}$ that associates a truth value for k truth values.

Let $H \subseteq \{0, ..., k\}$ be a finite set of non-negative integer numbers. Let's define the **Choose-H** (C_H) operator in the following way:

$$C_H(f_1, f_2, ..., f_k) = \begin{cases} 1, \text{if} \sum_{i=1}^{k} f_i \in H \\ 0, \text{otherwise} \end{cases}$$

This operator assumes the true value if and only if the number of true arguments is in H. The operator $C_{\{0,2\}}(f_1, ..., f_k)$, for instance, assumes 1 if there is exactly 0 or 2 true values among $f_1, ..., f_k$, and 0 otherwise. When H is an unit set $H = \{x\}$, we write C_x instead of $C_{\{x\}}$ for simplicity.

The operator can be clearly implemented as a disjunction of $|H|$ counters and comparators. These circuits can be converted to a CNF formula in linear time [4]. Also, if H has constant size, then the entire operator can be converted also in linear time by renaming the resulting formulae.

It's worth mentioning that the operators $\wedge$, $\vee$ and $\neg$ can be directly translated to some *Choose* operator with the following properties: $(f_1 \wedge ... \wedge f_k) = C_k(f_1, ..., f_k)$; $(f_1 \vee ... \vee f_k) = C_{\{1,...,k\}}(f_1, ..., f_k)$; $\neg f_1 = C_0(f_1)$. Also, the property $\neg C_H(f_1, ..., f_k) = C_{\{0,...,k\} \setminus H}(f_1, ..., f_k)$ holds.

These operators allow us to count the number of true values among a certain set of variables. In our reductions, we use them to force the degree of certain vertices, as shown in section 3.

3 Reductions from problems in graphs

We present in this section some reductions from problems in graphs to SAT and MaxSAT using the operators presented in section 2. We start by showing reductions from Path Checking and Minimum Path. Although these problems are in P, these reductions are used as subroutines while reducing NP problems.

3.1 Path Checking

Let $G = (V, E)$ be a graph and let $v_a, v_b \in V$ be a pair of distinct vertices in G. Let's recall that a (single) path P between v_a and v_b with length k in G is a sequence P of edges in E, $P = (e_1, e_2, ..., e_k)$, with the following properties:

- $v_a \in e_1$ and there is no edge in $P \setminus \{e_1\}$ containing v_a;
- $v_b \in e_k$ and there is no edge in $P \setminus \{e_k\}$ containing v_b;
- $|e_i \cap e_{i+1}| = 1$ and there is no edge in $P \setminus \{e_i, e_{i+1}\}$ that contains the vertex in $e_i \cap e_{i+1}$, for all $1 \leq i < k$.

The Path Checking problem consists of, given G, v_a and v_b, checking whether there is a path between v_a and v_b in G.

Let $f_p(G, v_a, v_b)$ be a boolean formula that is satisfiable if and only if there is at least one path between v_a and v_b in the graph G. Also, any model A for this formula describes a subgraph $G_p(A)$ of G that contains at least one path between the given vertices. This formula can be built in the following way:

- Let's associate, for each $e_i \in E$, a boolean variable ϱ_i, and let $\mathscr{E} = \{\varrho_i : e_i \in E\}$. In a given assignment $A : \mathscr{E} \to \mathbb{B}$ for $f_p(G, v_a, v_b)$, $A(\varrho_i) = 1$ if and only if e_i is in the subgraph described by A. The described graph $G_p(A)$ is defined then as the subgraph of G induced by the edge set $\{e_i \in E : A(\varrho_i) = 1\}$;
- Let $\mathscr{N}(v_i) = \{\varrho_j \in \mathscr{E} : e_j \in N(v_i)\}$ be the set of variables associated with the *edges* in $N(v_i)$, the neighbourhood of v_i. Let's force the degree $d()$ of both vertices v_a and v_b to be 1, i.e., $d(v_a) = d(v_b) = 1$ in the described subgraph. This can be done with the constraints $C_1(\mathscr{N}(v_a))$ and $C_1(\mathscr{N}(v_b))$;
- For each other vertex $v_i \in V \setminus \{v_a, v_b\}$, let's force its degree to be 0 or 2 in the required subgraph. This can be done with the constraint $C_{\{0,2\}}(\mathscr{N}(v_i))$.

The formula $f_p(G, v_a, v_b)$ is built then as a conjunction of these constraints:

$$f_p(G, v_a, v_b) = C_1(\mathscr{N}(v_a)) \wedge C_1(\mathscr{N}(v_b)) \wedge \\ C_{\{0,2\}}(\mathscr{N}(v_1)) \wedge C_{\{0,2\}}(\mathscr{N}(v_2)) \wedge ... \wedge C_{\{0,2\}}(\mathscr{N}(v_m)) \quad (1)$$

where $\{v_1, v_2, ..., v_m\} = V \setminus \{v_a, v_b\}$.

To prove that there is one path between v_a and v_b in G if and only if $f_p(G, v_a, v_b)$ is satisfiable, it's enough to show the following two theorems:

Theorem 1: If there is a path $P = (e_1, ..., e_k)$ between v_a and v_b in G, then there is a model $A : \mathscr{E} \to \mathbb{B}$ for $f_p(G, v_a, v_b)$ for which $G_p(A)$ contains P.

Proof: Let A be the assignment that assigns 1 to and only to variables associated to edges in P, i.e., $A(\varrho_i) = 1$ if $e_i \in P$, and $A(\varrho_i) = 0$ otherwise. Clearly $G_p(A)$ contains only the edges in P. Let's show that A satisfies $f_p(G, v_a, v_b)$.

The edge e_1 is the only edge in P that contains v_a, since P is a simple path as defined above. Thus, $d(v_a) = 1$ in $G_p(A)$, satisfying $C_1(\mathscr{N}(v_a))$. For the same reason, e_k, the last edge in P, is the only edge in it that contains v_b, and thus $C_1(\mathscr{N}(v_b))$ is satisfied as well.

Since P is a simple path with length k, e_i and e_{i+1} share exactly one vertex, say v_i, for each $1 \leq i < k$. The edges e_i and e_{i+1} are the only edges in P that contains v_i. Thus, $d(v_i) = 2$ in $G_p(A)$, which satisfies the constraint $C_{\{0,2\}}(\mathscr{N}(v_i))$. For all the vertices v_j that are not in any edge of P, there is no edge in $N(v_j)$ present in $G_p(A)$. This satisfies the constraint $C_{\{0,2\}}(\mathscr{N}(v_j))$ as well, since $d(v_j) = 0$ in the described graph.

Hence, A satisfies all the given constraints, satisfying $f_p(G, v_a, v_b)$. □

Theorem 2: Let A be a complete assignment of $\mathscr{E}$. If A satisfies $f_p(G, v_a, v_b)$, then there is a path between v_a and v_b in the graph $G_p(A)$.

Proof: Let A be a complete assignment of $\mathscr{E}$ that satisfies $f_p(G, v_a, v_b)$. Assume that $G_p(A)$ doesn't contain a path between v_a and v_b. v_a and v_b belong to different connected components of $G_p(A)$. Let $G'_p(A, v_a)$ be the subgraph of $G_p(A)$ consisting of the connected component v_a belongs to.

Since A satisfies $f_p(G, v_a, v_b)$, $d(v_a) = 1$ in $G_p(A)$ and, since $G'_p(A, v_a)$ is a connected subgraph of G containing v_a, $d(v_a) = 1$ in $G'_p(A, v_a)$ as well.

Since A satisfies $f_p(G, v_a, v_b)$, all vertices in $V \setminus \{v_a, v_b\}$ have even degree (0 or 2) in $G_p(A)$. Also, except for v_a, the only vertex that has odd degree (1) in $G_p(A)$ is v_b, which is not in $G'_p(A, v_a)$, as stated. Due to this, v_a is the only vertex in $G'_p(A, v_a)$ that has odd degree. So there's an odd number of vertices (only one, v_a) with odd degree in $G'_p(A, v_a)$.

It's well know that there must be an even number of vertices with odd degree in any graph. Since $G'_p(A, v_a)$ is a graph, the conclusion above is an absurd. Hence, a connected component of $G_p(A)$ with v_a and without v_b cannot exist, and thus there is a path between v_a and v_b in the graph $G_p(A)$. □

There are $|E|$ variables in the formula. Also, there are $|V|$ constraints in it, one for each vertex in G. The arity of each one is equal to the degree of the corresponding vertex in G. Hence, the number of elements in the entire formula is bounded to $|V|$ plus the sum of the degrees of the vertices in G, $2|E|$. Hence, the size of the resulting formula is linear in the size of the given graph.

3.2 Minimum Path

Once reduced, Path Checking can be used to reduce Minimum Path to MaxSAT.

Let's describe the (Partial Weighted) MaxSAT problem as, given a hard formula f_h, a set of soft formulae $f_s = \{f_{s_1}, f_{s_2}, ..., f_{s_{|f_s|}}\}$ and a function $w_s : f_s \to \mathbb{N}$ that associates a weight to each soft formula, finding a model for f_h that maximizes the sum of the weights of the satisfied soft formulae.

Let $w : E \to \mathbb{N}$ be a function that associates, for each edge in a given graph G, a natural weight. The weight of a path P is the sum of the weights of the edges in P. The minimum path problem consists of, given G, w and a pair $v_a, v_b \in V$, finding a path between v_a and v_b whose weight is minimum.

Minimum Path can be reduced to MaxSAT by defining a hard formula that states that there must be a path between the given pair of vertices, and a set of soft formulae that states that the path must be minimum.

The hard formula can be $f_p(G, v_a, v_b)$ as described above. Each soft formula can be defined simply as the unit clause $(\neg\varrho_i)$ with weight $w(e_i)$, for each $e_i \in E$. These formulae indicate the willing of maximizing the sum of the weights of the edges not present in the desired path, minimizing then the weight of such a path.

There is a minimum path P between v_a and v_b in the given graph if and only if there is an optimal solution A for the given instance for which $G_p(A)$ contains only P. This is shown by the following two theorems:

Theorem 3: If $A : \mathscr{E} \to \mathbb{B}$ is a solution for the given MaxSAT instance, then $G_p(A)$ contains exactly a minimum path between v_a and v_b.

Proof: Let $A : \mathscr{E} \to \mathbb{B}$ be a model for $f_p(G, v_a, v_b)$ and let m be the sum of the satisfied soft formulae by A.

Since each soft formula consists only of the clause $(\neg\varrho_i)$ with weight $w(e_i)$ for each $e_i \in E$, m is equal to the sum of the weights of the edges not present in $G_p(A)$. Let $M = \sum_{e_i \in E} w(e_i)$ be the sum of the weights of all edges in G.

Since MaxSAT consists of maximizing m, the sum of the weights of the edges in $G_p(A)$, $M - m$, is minimized. Thus, if A is optimal, then the sum of the weights of all edges in $G_p(A)$ is minimum. Also, according to the theorem 2, there is a path between v_a and v_b in $G_p(A)$. Then, $G_p(A)$ is a subgraph of G that contains a path between the given vertices and whose sum of its edges is minimum. Clearly $G_p(A)$ is a subgraph consisting of a required minimum path.

Hence, if A is an optimal solution, then $G_p(A)$ contains only a minimum path between the given pair of vertices. □

Theorem 4: Let P be a minimum path between v_a and v_b in a given graph G. There is an optimal assignment A that is a solution for the built MaxSAT instance for which $G_p(A)$ contains exactly the edges in P.

Proof: Let P be a minimum path between v_a and v_b in the given graph.

Consider the assignment A shown in theorem 1. A satisfies $f_p(G, v_a, v_b)$, and $G_p(A)$ contains only the edges in P. Let $w(P)$ be the weight of the path P. Since $A(\varrho_i) = 1$ for each $e_i \in P$, all and only the soft formulae corresponding to an edge in P is unsatisfied by A. Thus, the sum of the weights of all satisfied soft formulae (the objective function of MaxSAT) is equal to $M - w(P)$. Since $w(P)$ is minimum, $M - w(P)$ is maximum.

Hence, A is an optimal solution for the given MaxSAT instance. □

Since $|f_p(G, v_a, v_b)|$ is linear in the size of the given graph G as stated in section 3.1, and since there is only an unit clause in f_s for each edge in G, this reduction is still linear in the size of G.

3.3 Connectedness Checking

The reduction from Minimum Path can be used as subroutine to reduce others NP problems, such as Steiner Tree. To reduce it, however, it's necessary to reduce Connectedness Checking firstly.

Let $G = (V, E)$ be a graph and $S \subseteq V$ be a subset of the vertices in G. The Connectedness Checking problem consists of checking whether all vertices in S are connected, i.e., there is a path in G between s_i and s_j, for all $s_i, s_j \in S$.

Let's built a formula $f_c(G, S)$ that is satisfiable if and only if all vertices in S belongs to the same connected component in G. Analogously to $f_h(G, v_a, v_b)$, any model A for $f_c(G, S)$ describes a subgraph $G_c(A)$ of G in which all vertices in S are connected. This formula can be built in the following manner:

- Let $G_1, G_2, ..., G_{|S|-1}$ be $|S|-1$ graphs, all of them identical to G. Also, let $e_{i,j}$ be the edge i in the graph G_j. Let's associate, for each $e_{i,j} \in \cup_{j=1}^{|S|-1} E(G_j)$, a boolean variable $\varrho_{i,j}$. Let $\mathscr{E}_j = \{\varrho_{i,j} : e_i \in E(G_j)\}$ and $\mathscr{E} = \cup_{j=1}^{|S|-1} \mathscr{E}_j$. Given a model $A : \mathscr{E} \to \mathbb{B}$ for $f_c(G, S)$, $G_c(A)$ is then defined as the union of all graphs $G_{j_p}(A_j)$, where A_j is the assignment A restricted to the variables in $\mathscr{E}_j$, i.e., $A_j = \{(\varrho_{i,j}, A(\varrho_{i,j})) : \varrho_{i,j} \in \mathscr{E}_j\}$;
- Let $S' = (s_1, ..., s_{|S|})$ be a permutation of the set S. Let's state that there must be a path between each pair of adjacent vertices in S', in each graph. This can be done with the constraints $f_p(G_j, s_j, s_{j+1})$ for all $1 \leq j < |S|$.

$f_c(G, S)$ is defined then as a conjunction of these constraints, i.e.,

$$f_c(G, S) = f_p(G_1, s_1, s_2) \wedge f_p(G_2, s_2, s_3) \wedge ... \wedge f_p(G_{|S|-1}, s_{|S|-1}, s_{|S|}) \quad (2)$$

Let's show that $f_c(G, S)$ is satisfiable if and only if all vertices in S are connected in G. To do so, let's prove the following theorems:

Theorem 5: If all vertices in S are connected in G, then there is a model $A : \mathscr{E} \to \mathbb{B}$ for $f_c(G, S)$ for which all vertices in S are connected in $G_c(A)$.

Proof: Since all vertices in S belongs to the same connected component of G, there is a path in G between each pair of adjacent vertices in S'. Let P_j be the path between the vertices s_j and s_{j+1}. As shown in theorem 1, there is a model $A_j : \mathscr{E}_j \to \mathbb{B}$ for the formula $f_p(G_j, s_j, s_{j+1})$, for all $1 \leq j < |S|$. Also, $G_{j_p}(A_j)$ contains P_j.

Let $A : \mathscr{E} \to \mathbb{B}$ be the union of all $A_j : \mathscr{E}_j \to \mathbb{B}$ described above. Since A_j satisfies $f_p(G_j, s_j, s_{j+1})$, A satisfies all the constraints in $f_c(G, S)$, and thus is a model for it. Also, since $G_c(A)$ is defined as the union of all $G_{j_p}(A_j)$ that contains P_j, $G_c(A)$ contains a path between each adjacent pair of vertices in S'. Due to transitivity, $G_c(A)$ contains a path between all pairs of elements in S and, thus, contains a connected subgraph of G that contains all vertices in S. □

Theorem 6: Let A be a complete assignment of $\mathscr{E}$. If A is a model for $f_c(G, S)$, then all vertices in S are connected in $G_c(A)$.

Proof: Let $A : \mathscr{E} \to \mathbb{B}$ be a model for $f_c(G, S)$. Since A satisfies the formula $f_p(G_j, s_j, s_{j+1})$ for all $1 \leq j < |S|$, there is a path P_j between the vertices s_j and s_{j+1} in the graph $G_{j_p}(A_j)$ as shown in theorem 2, where A_j is the assignment A restricted to the variables in $\mathscr{E}_j$.

Since $G_{j_p}(A_j)$ is a subgraph of $G_c(A)$, there is a path in $G_c(A)$ for all $1 \leq j < |S|$ as well. Due to transitivity, there is a path between each pair of vertices in S in $G_c(A)$, thus all vertices in S are connected in this subgraph of G. □

There are $|S|-1$ constraints in $f_c(G, S)$. Each of them has $|V|+2|E|$ elements, as shown in subsection 3.1. Hence, the size of this formula, which has $|E|(|S|-1)$ variables, is equal to $(|V|+2|E|)(|S|-1)$. It's quadratic in the size of the graph in the worst case. However, it's worth mentioning that $|S| \leq |V|$, and, if $|V| \leq |E|$, the size of the formula is smaller than $(|V| + |E|)^2$.

3.4 Steiner Tree

The Steiner Tree problem consists of, given a graph $G = (V, E)$, a set of its vertices $S \subseteq V$ and a function $w : E \to \mathbb{N}$ that associates a weight to each edge in G, finding a subgraph of G that is connected, contains all the vertices in S and whose weight is minimum. The weight of a subgraph is equal to the sum of the weights of its edges. Clearly an optimal subgraph is always a tree. The problem is known to be NP-hard [5].

Steiner Tree can be reduced to MaxSAT if the hard formula states that all vertices in S are connected, and the soft formulae state that the required graph must have the minimum sum of weights of its edges.

The hard formula is defined as $f_c(G, S)$ as described above. Each soft formula is defined as $f_{s_i} = (\neg\varrho_{i,1} \wedge ... \wedge \neg\varrho_{i,|S|-1})$ with weight $w(e_i)$, for each $e_i \in E$.

The correctness of this reduction is shown by the following two theorems. Their proof were omitted due to space limitations. However, they can be proved using theorems shown in sections 3.1, 3.2 and 3.3.

Theorem 7: If $A : \mathscr{E} \rightarrow \mathbb{B}$ is a solution for the reduced instance to MaxSAT, then $G_c(A)$ is an optimal tree containing the vertices in S.

Theorem 8: Let T be an optimal tree of G that contains all vertices in S. There is a model A for $f_c(G, S)$ that is a solution for the given MaxSAT instance for which $G_c(A)$ contains only T.

The number of elements in $f_c(G, S)$ is equal to $(|V| + 2|E|)(|S| - 1)$, as shown in subsection 3.3. Also, There are $|E|$ formulae in f_s with size $|S|-1$ each. Hence, the sum of the size of all the formulae is equal to $(|V| + 3|E|)(|S| - 1)$. This reduction is also quadratic in the size of the given graph in the worst case, but smaller than $2(|V| + |E|)^2$ if $|V| < |E|$.

This reduction is partially based on ideas from a previously published approximated reduction from Steiner Tree to MaxSAT [2]. After a pre-processing step, the reduction presented in [2] generates a CNF instance with $|E|+k(|S|-1)$ variables and $O(|E|+k|E|(|S|-1))$ clauses, where k is an approximation factor. k is not polynomial for exact reductions. Our reduction is exact, quadratic and doesn't require a pre-processing step.

Using similar ideas, it's possible to reduce other NP problems than Steiner Tree to SAT and MaxSAT. One can reduce Hamiltonian Cycle, for instance, stating that all vertices in V must be connected and that the degree of each one in the resulting subgraph must be exactly 2. This reduction is also quadratic in the worst case. It's not covered in this paper due to space limitations.

3.5 Clique

As shown in previous sections, it's possible to built a formula stating that some (possible all) vertices of a graph must be connected in a solution in quadratic time, in the worst case. However, reductions can be simplified when the required solution consists of a complete connected subgraph, i.e., a clique.

A clique of size k in a graph G is a complete subgraph of G with exactly k vertices. The decision problem Clique consists of, given G and $k \in \mathbb{N}$, checking whatever there is a clique of size k in G. In this paper we assume $k > 1$, since the problem is trivial for other cases. Clique is a NP-complete problem [5].

Clique can be reduced to SAT by building a formula $f_K(G, k)$ that is satisfiable if and only if there is a clique of size k in G. Also, any model A for $f_K(G, k)$ describes a subgraph $G_K(A)$ of G consisting of a required clique. This formula can be built in the following way:

- Analogously to the reduction from Minimum Path, let's associate a binary variable ϱ_i for each $e_i \in E$, and let $\mathscr{E}$ be the set of all these variables. Given a model $A : \mathscr{E} \rightarrow \mathbb{B}$ for $f_K(G, k)$, $G_K(A)$ is defined then as the subgraph of G induced by the edge set $\{e_i \in E : A(\varrho_i) = 1\}$.
- Let's state that there must be exactly k vertices in $G_K(A)$ with degree $k-1$. This can be done with the constraint, where $\{v_1, v_2, ..., v_{|V|}\} = V$: $C_k(C_{k-1}(\mathscr{N}(v_1)), C_{k-1}(\mathscr{N}(v_2)), ..., C_{k-1}(\mathscr{N}(v_{|V|})))$

– Also, let's state that all others $|V|-k$ vertices must have degree 0 in $G_K(A)$. This can be done with the constraint
$C_{|V|-k}(C_0(\mathscr{N}(v_1)), C_0(\mathscr{N}(v_2)), ..., C_0(\mathscr{N}(v_{|V|})))$.

$f_K(G,k)$ can then be built as the conjunction of the given two constraints:

$$f_K(G,k) = C_k(C_{k-1}(\mathscr{N}(v_1)), C_{k-1}(\mathscr{N}(v_2)), ..., C_{k-1}(\mathscr{N}(v_{|V|}))) \wedge \\ C_{|V|-k}(C_0(\mathscr{N}(v_1)), C_0(\mathscr{N}(v_2)), ..., C_0(\mathscr{N}(v_{|V|}))) \quad (3)$$

This reduction to SAT is correct, as shown by the two followings theorems:

Theorem 9: If there is a clique of size k in G, then $f_K(G,k)$ is satisfiable.

Proof: Let G' be a clique of size k in G, and let A be the assignment that associates 1 to the variables corresponding to the edges in G', i.e., $A(\varrho_i) = 1$ if and only if $e_i \in E(G')$.

Since G' is a complete subgraph, all vertices in $V(G')$ have the same and maximum possible degree in it, equal to $|V(G')| - 1$. Since G' has k vertices, $|V(G')| = k$ and thus the degree of all k vertices in it is equal to $k-1$. Hence, there are exactly $k-1$ edges in the neighbourhood of v_i in the clique, for all $v_i \in V(G')$. Since this sentence is valid for exactly k vertices, A satisfies the first constraint of $f_K(G,k)$.

Also, all the others $|V|-k$ vertices in G do not belong to G', and thus none (0) of the edges in their neighbourhood are in the clique. Hence, A satisfies the second constraint of $f_K(G,k)$ as well, and thus satisfies the entire formula. □

Theorem 10: If $f_K(G,k)$ is satisfiable, then G contains a clique of size k.

Proof: Let A be a model for $f_K(G,k)$. Since A satisfies the second constraint of $f_K(G,k)$, there are at least $|V|-k$ vertices for which no edges in $G_K(A)$ connects them, since their degree is zero. Hence, there are at most k vertices in $G_K(A)$.

Also, since A satisfies the first constraint of the formula, there are at least k vertices for which there is at least one edge in $G_c(A)$ connecting it, since we assume $k-1 \geq 1$. $G_K(A)$ has at most and at least k vertices with positive degree. Hence, there are exactly k vertices in $G_K(A)$.

Since A satisfies the first constraint of $f_K(G,k)$, the degree of any vertex in $G_K(A)$ is equal to $k-1$. This degree is maximum, since there are k vertices in $G_K(A)$. Hence, any vertex in the subgraph is directly connected to all others vertices in it, and thus $G_K(A)$ is a clique of size k.

Since $G_K(A)$ is a subgraph of G, G contains a clique of size k. □

The number of elements in both the first and second constraints of $f_K(G,k)$ is proportional to the number of vertices in G plus the sum of the degrees of all vertices in G, $2|E|$. To be exact, there are $4|E|+2|V|+3$ elements in the entire formula, which has $|E|$ variables. Hence, the size of this formula is linear in the size of the given graph.

The decision problem Clique can be used to solve the optimization problem Maximum Clique. The problem can be solved by binary searching k or by incrementing k until the formula is unsatisfiable.

4 Experimental Results

We implemented all described reductions. The resulting formulae were converted to CNF by using recursive counters [4] with variable injections.

We compare the size of the resulting formulae and the time taken by a (Max)SAT solver to solve them against earlier published reductions. We use Lingeling [7] to solve SAT instances, and MiniMaxSAT [3] to solve MaxSAT ones. Both solvers were run on an *Intel(R) Core(TM)2 Duo CPU E8400, 3 GHz, 4 Gb RAM, Ubuntu 10.10 OS.*

The reduction from Steiner Tree was compared against a straightforward, not optimized implementation of a depth-first transversal (DFT) representation [1]. The tested instances were obtained from the SteinLib benchmark [6].

		Presented Reduction			DFT		
Instance	$\|V\|/\|E\|/\|S\|$	No.Vars	No.Claus.	Time	No.Vars.	No.Claus.	Time
b01	50/63/9	2367	7309	2.87s	15939	1032075	TLE
b02	50/63/13	3363	10330	3.56s	15939	1032079	TLE
b03	50/63/25	6711	20705	10.18s	15939	1032091	TLE
b04	50/100/9	4748	14816	TLE	40100	4080209	TLE

Table 1. Experimental results of reductions from Steiner Tree

Table 1 shows the results. The column $|V|/|E|/|S|$ describes the size of the elements of the graph. *No. Vars.* and *No. Claus.* indicate the number of boolean variables and the number of clauses in the resulting formulae, respectively. *Time* describes the time took by the solver to solve the instance. The value *TLE* indicates that the instance was not solved within one hour.

One can observe that our reduction generates formulae which are smaller and can be solved faster than the DFT reduction. It's worth mentioning, however, that DFT's authors themselves point the ineffectiveness of this reduction [1].

It's also possible to notice that the size of a formula is not proportional to the time a SAT solver takes to solve it. The formula generated from the instance $b04$ is smaller, but is not solved as fast as the formula generated from $b03$.

The presented reduction from Clique was compared against the simpler reduction from Maximum Clique to MaxSAT, linear in the complement of the given graph (LCOMP), as shown in [3]. The conversion from our reduced instances to CNF generated formulae larger than the ones generated by LCOMP, mainly due to the large number of encoded half and full adders. Using LCOMP is more efficient than converting our formulae to CNF, even for random sparse graphs (with density 1%).

It's worth mentioning, however, that formulae generated by our reduction from Clique have $4|E| + 2|V| + 3$ elements before the conversion to CNF, while the hard formula generated by LCOMP have $\binom{|V|}{2} - |E|$ clauses with size 2.

Also, before the conversion to CNF, instances generated by our reduction from Steiner Tree have $|E|(|S|-1)$ variables (aprox. 5 times less than the number of variables after the conversion, for the tested instances), and $(|V|+3|E|)(|S|-1)$ elements (aprox. 4 times less than the number of clauses for the same instances).

As discussed in section 5, this motivate the direct implementation of the Choose operators in a non clausal SAT solver. This would discard the need to convert the formulae to CNF.

5 Conclusion and future work

We presented *Choose-H*, a class of operators that simplifies reductions from problems in graphs to SAT and MaxSAT. As shown in section 2, any classical formula can be directly converted to use only these operators. Also, these operators can be converted to classical formulae in linear time, if $|H|$ is constant.

Our experiments showed that, when converted to CNF, the formulae generated by our reduction from Steiner Tree are smaller and can be solved faster than ones generated by the previously presented exact reduction. However, they are not small enough, as our experiments with the reduction from Clique also showed.

A possible way to take advantage of smaller formulae is to adapt some non clausal SAT solver to make it solve MaxSAT instances with *Choose* operators, using Branch and Bound, like MiniMaxSAT [3]. We intend to implement the operators in some free, open source non clausal SAT solver.

It won't be needed to create new variables in the formulae, since, given a (partial) assignment, the solver can count and compare the number of true values among the arguments of an operator in a trivial way. This will keep the number of variables of an instance in its minimum, $|\mathscr{E}|$.

Also, since any classical formula can be converted to use the *Choose* connectors, we are motivated to create a new non clausal SAT solver that handles these operators only. Efficient internal data structures are being studied and implemented to optimize the Boolean Constraint Propagation Procedure on these connectors.

References

1. Kautz, H., Selman, B., Jiang, Y.: A general stochastic approach to solving problems with hard and soft constraints. In: The Satisfiability Problem: Theory and Applications, 573-586. American Mathematical Society. (1996)
2. Menai, M.E.B.: A logic-based approach to solve the steiner tree problem. In: Artificial Intelligence Applications and Innovations III, Proceedings of the 5TH IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI 2009), 296, 73–79. Springer. (2009)
3. Heras, F., Larrosa, J., Oliveras, A.: Minimaxsat: An efficient weighted max-sat solver. In: Journal of Artificial Intelligence Research, 31, 1–32. (2008)
4. Muller, D.E., Preparata, F.P.: Bounds to complexities of networks for sorting and for switching. In: J. ACM, 22, 195–201. (1975)
5. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, 85–103, Plenum Press. (1972)
6. Koch, T., Martin, A., Vos, S.: Steinlib: An updated library on steiner tree problems in graphs. Technical Report ZIB 00-37, Zuse Institute Berlin. (2000)
7. Biere, A.: Lingeling, plingeling, picosat and precosat at SAT race 2010. Technical Report 10/1, FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University. (2010)