A Logic-Based Approach to Solve the Steiner Tree Problem

Mohamed El Bachir Menai

Abstract Boolean satisfiability (SAT) is a well-studied $\mathscr{NP}$-complete problem for formulating and solving other combinatorial problems like planning and scheduling. The Steiner tree problem (STP) asks to find a minimal cost tree in a graph that spans a set of nodes. STP has been shown to be $\mathscr{NP}$-hard. In this paper, we propose to solve the STP by formulating it as a variation of SAT, and to solve it using a heuristic search method guided by the backbone of the problem. The algorithm is tested on a well known set of benchmark instances. Experimental results demonstrate the applicability of the proposed approach, and show that substantial quality improvement can be obtained compared to other heuristic methods.

1 Introduction

The satisfiability problem in propositional logic (SAT) is the task to decide for a given propositional formula in conjunctive normal form (CNF) whether it has a model. More formally, let $C = \{C_1, C_2, \dots, C_m\}$ be a set of m clauses that involve n Boolean variables $x_1, x_2, \dots, x_n$. A literal is either a variable x_i or its negation $\neg x_i$. Each clause C_i is a disjunction of n_i literals, $C_i = \bigvee_{j=1}^{n_i} l_{ij}$. The SAT problem asks to decide whether the propositional formula $\Phi = \bigwedge_{i=1}^{m} C_i$ is satisfiable. SAT is the first problem shown to be $\mathscr{NP}$-complete [2]. The $\mathscr{NP}$-completeness concept deals with the idea of polynomial transformation from a problem P_i to P_j where the essential results are preserved: if P_i returns "yes", then P_j returns "yes" under the same problem input. MAX-SAT (or *unweighted* MAX-SAT) is the optimization variation of SAT. It asks to find a variable assignment that maximizes the number of satisfied clauses. In *weighted* MAX-SAT (or only MAX-SAT), a weight w_i is assigned to each clause C_i (notation: $C_i^{w_i}$), and the objective is to maximize the total weight of

Mohamed El Bachir Menai
Computer Science Department, College of Computer and Information Sciences, King Saud University, PO Box 51178, Riyadh 11543, Kingdom of Saudi Arabia, e-mail: menai@ksu.edu.sa

satisfied clauses $\sum_{i=1}^{m} w_i \cdot \Im(C_i)$, where $\Im(C_i)$ is one if and only if C_i is satisfied and otherwise zero. Partial MAX-SAT (PMSAT) involves two weighted CNF formulas f_A and f_B. The objective is to find a variable assignment that satisfies all clauses of f_A (non-relaxable or hard clauses) together with the maximum clauses in f_B (relaxable or soft clauses). SAT has seen many successful applications in various fields such as planning, scheduling, and Electronic Design Automation. Encoding combinatorial problems as SAT problems has been mostly motivated by the simplicity of SAT formulation, and the recent advances in SAT solvers. Indeed, new solvers are capable of solving very large and very hard real world SAT instances. Optimization problems that involve hard and soft constraints can be cast as a PMSAT, e.g. university course scheduling and FPGA routing. In 1995, Jiang *et al.* [6] proposed the first heuristic local search algorithm to solve this problem as a MAX-SAT. In 1997, Cha *et al.* [1] proposed another local search technique to solve the PMSAT problem. In 2005, Menai *et al.* [9] proposed a coevolutionary heuristic search algorithm to solve the PMSAT. In 2006, Fu and Malik [4] proposed two approaches based on a state-of-the-art SAT solver to solve the PMSAT.

The Steiner tree problem (STP) in graphs is a classic combinatorial problem. It can be defined as follows. Given an arbitrary undirected weighted graph $G = (V, E, w)$, where V is the set of nodes, E denotes the set of edges and $w : E \rightarrow \mathbb{R}^+$ is a non-negative weight function associated with its edges. Any tree T in G spanning a given subset $S \subseteq V$ of terminal nodes is called a *Steiner tree*. Note that T may contain non-terminal nodes referred to as *Steiner nodes*. The cost of a tree is defined to be the sum of its edge weights. The STP asks for a minimum-cost Steiner tree. The decision version of STP has been shown to be $\mathscr{N}P$-complete by Karp [8]. STP has found uses across a wide array of applications including network routing [10] and VLSI design [7]. Several implementations of metaheuristics have been proposed for the approximate solution of STP or its variations, such as Simulated Annealing [11], Tabu Search [5], and Genetic Algorithms [3]. In this paper we are interested in solving the STP as a PMSAT problem. We show how to encode the STP into PMSAT and propose a practical approach to solve it based on one of the best known SAT solver WalkSAT [12] with certain extensions. Indeed, the success of WalkSAT and its variations has led to the paradigm of SAT encoding and solving difficult problems from other problem domains. Our approach is based on exploiting problem structural information, backbone in particular, to guide the search algorithm towards promising regions of the search space. We show empirically that this method is effective by comparing our results to those obtained with specialized Steiner heuristic algorithms. The rest of the paper is structured as follows. In the next section, we explain how to encode the STP into PMSAT. In Section 3, we describe a heuristic algorithm for PMSAT using backbone guided search. Computational results are reported in Section 4. Concluding remarks are drawn in the last section.

2 PMSAT Encoding of STP

Jiang *et al.* [6] suggested to encode STP as a weighted MAX-SAT instance and to solve it using a MAX-SAT solver. However, a solution for the MAX-SAT instance may violate some clauses whose satisfiability is required for the feasibility of the STP solution. We propose to encode STP as a PMSAT instance to formulate independently hard and soft constraints and to solve it using a PMSAT solver. Let $G = (V, E, w)$ be a weighted graph of n nodes $v_1, v_2, \dots, v_n$, and $S \subseteq V$ a set of terminal nodes.

1. For each edge $e_{ij}, 1 \leq i \leq n, 1 \leq j \leq n$, connecting nodes i and j of the graph, introduce a boolean variable x_{ij}. $\mathfrak{I}(x_{ij}) = 1$ if e_{ij} is chosen as part of the Steiner tree.
2. For each variable x_{ij}, construct the clause $c_{ij} = (\neg x_{ij})^{w_{ij}}$ to minimize the cost of including the edge e_{ij} in the tree. $\mathbf{f_B} = \bigwedge \mathbf{c_{ij}}$ are soft clauses.
3. List terminal nodes in an arbitrary order. For some fixed l, generate the possible $k (k \leq l)$ shortest paths between successive pairs of nodes using Dijkstra's algorithm. If no path exists between two terminal nodes, then return no solution.
   Variables $p^1_{ij}, p^2_{ij}, \dots, p^k_{ij}$ denote the k shortest paths between terminal nodes i and j. The reduction is an approximation of the original instance, since only the k shortest paths are generated between pairs of nodes.
4. A solution to STP must contain a path between each pair of terminal nodes. Namely, for each $(v_i, v_j) \in S \times S$, construct a clause $(p^1_{ij} \vee p^2_{ij} \vee \cdots \vee p^k_{ij})$.
   $\mathbf{f_{A_1}} = \bigwedge (\mathbf{p^1_{ij}} \vee \mathbf{p^2_{ij}} \vee \cdots \vee \mathbf{p^k_{ij}})$ are hard clauses.
5. Each path must contain all its edges. Namely, for each path p^k_{ij} containing edges $e_{il}, e_{lm}, \dots, e_{rj}$, construct clauses $(p^k_{ij} \supset x_{il}) \wedge (p^k_{ij} \supset x_{lm}) \wedge \cdots \wedge (p^k_{ij} \supset x_{rj})$ which are equivalent to $(\neg p^k_{ij} \vee x_{il}) \wedge (\neg p^k_{ij} \vee x_{lm}) \wedge \cdots \wedge (\neg p^k_{ij} \vee x_{rj})$.
   $\mathbf{f_{A_2}} = \bigwedge ((\neg \mathbf{p^k_{ij}} \vee \mathbf{x_{il}}) \wedge (\neg \mathbf{p^k_{ij}} \vee \mathbf{x_{lm}}) \wedge \cdots \wedge (\neg \mathbf{p^k_{ij}} \vee \mathbf{x_{rj}}))$ are hard clauses.
6. Let $\mathbf{f_A} = \mathbf{f_{A_1}} \wedge \mathbf{f_{A_2}}$. $\mathbf{f} = \mathbf{f_A} \wedge \mathbf{f_B}$ is the PMSAT instance yield.

The number of variables is $|E| + k(|S| - 1)$. The total number of clauses is $O(|E| + kL(|S| - 1))$, where L is the maximum number of edges in pre-computed paths. The reduction is linearly dependent on the number of edges. The reduction is sound as any PMSAT solution yields a valid Steiner tree. Since all hard clauses f_A are satisfied, a path exists between each pair of terminal nodes in the obtained set of nodes. The reduction is incomplete, since the PMSAT instance will not yield a solution if there is no Steiner tree using the k paths generated in step 3.

3 Heuristic Search for PMSAT

Backbone variables are a set of literals which are true in every model of a SAT instance. The backbone of a PMSAT instance is the set of assignments of values to variables which are the same in every possible optimal solution. They are proven to influence hardness in optimization and approximation [13]. Heuristic search methods could improve their performance by using backbone information. We propose to solve a PMSAT using a heuristic local search algorithm that takes advantage of a pseudo-backbone sampled using information extracted from local minima. Our method is inspired by a heuristic sampling method for generating assignments for a local search for MAX-SAT [14].

Let Ω be a set of assignments on X, the set of Boolean variables, $A(x_i)$ the value of the variable x_i in the assignment A, and $C(A)$ the contribution of A defined as the total number of satisfied clauses in f_A and f_B: $C(A) = |f_A| \cdot \#sat_{f_A}(A) + \#sat_{f_B}(A)$, where $\#sat_{f_A}(A)$ and $\#sat_{f_B}(A)$ denote the number of satisfied clauses in f_A and f_B, respectively. A multiplier coefficient $|f_A|$ is added to $C(A)$ to underline the priority of satisfying clauses of f_A. A variable frequency p_i of positive occurrences of x_i in all assignments of Ω is defined as $p_i = (\sum_{A\in\Omega} C(A) \cdot A(x_i)) / \sum_{A\in\Omega} C(A)$. $A(x_i) = 1$ with the probability p_i. Let $X(\alpha)$ denote the set of variables which appear in the set of clauses α. The main steps of the algorithm BB_PMSAT are outlined in Figure 1.

```
procedure BB_PMSAT
input: A formula $F = f_A \wedge f_B$ in CNF containing $n$ variables $x_1,\ldots,x_n$,
        MaxTries1, MaxTries2, MaxSteps.
output: A solution $A$ for $F$, or “Not found” if $f_A$ is not satisfiable.
begin
        for $t = 0$ to $|\Omega| - 1$ do
          $A \leftarrow$ WalkSAT_MAXSAT($F$,$X(F)$,MaxTries=1,MaxSteps);
          If $A$ satisfies $F$ then return $A$;
          $\Omega[t] \leftarrow A$;
        end for
        $A \leftarrow$ BB_WalkSAT_MAXSAT($F$,$X(F)$, $\Omega$, MaxTries1, MaxSteps);
        if $A$ satisfies $F$ then return $A$;
        if ($\exists f_{A_{SAT}}, f_{A_{UNSAT}} | f_A = f_{A_{SAT}} \wedge f_{A_{UNSAT}}$) and ($A$ satisfies $f_{A_{SAT}}$)
          and ($X(f_{A_{SAT}}) \cap X(f_{A_{UNSAT}}) = \emptyset$)
        then $f \leftarrow f_{A_{UNSAT}}, X(f) \leftarrow X(f_{A_{UNSAT}})$;
        else $f \leftarrow f_A, X(f) \leftarrow X(f_A)$;
        end if
        $A_f \leftarrow$ BB_WalkSAT($f$,$X(f)$, $\Omega$, MaxTries2, MaxSteps);
        if $A_f$ satisfies $f$ then update $A$ and return $A$;
        return “Not found”;
end
```

Fig. 1 The BB_PMSAT procedure

In a first phase, the PMSAT instance is solved as a MAX-SAT instance using a variation of the procedure WalkSAT [12] for MAX-SAT (WalkSAT_MAXSAT) to initialize the pseudo-backbone Ω with reached local minima. Next, both formulas f_A and f_B are solved together as a MAX-SAT instance using a variation of the procedure BB_WalkSAT for MAX-SAT (BB_WalkSAT_MAXSAT). Figure 2 presents the procedure BB_WalkSAT for SAT. It integrates a pseudo-backbone estimation using variable frequencies p_i to generate initial assignments. The pseudo-backbone Ω is updated at each time a new local minimum is encountered. The second phase of the algorithm is performed if the best assignment found in the previous phase does not satisfy f_A (a PMSAT instance F is satisfied iff f_A is satisfied). In such case, it is recycled to try to satisfy f_A using BB_WalkSAT guided by the information in Ω. If the best assignment found does not satisfy f_A, then it is recycled to a model of f_A using Ω.

```
procedure BB_WalkSAT
input: A formula $F$ in CNF containing $n$ variables $x_1,\dots,x_n$,
       $\Omega$, MaxTries, MaxFlips.
output: A satisfying assignment $A$ for $F$, or “Not found”.
begin
       for $try = 1$ to MaxTries do
         Calculate $p_i, (i = 1,n)$ using $\Omega$;
         $A \leftarrow$ best assignment for $n$ variables among $t$ randomly
         created assignments in $\Omega$ using $p_i$;
         for $flip = 1$ to MaxFlips do
           if $A$ satisfies $F$ then return $A$;
           $c \leftarrow$ an unsatisfied clause chosen at random;
           if there exists a variable $x$ in $c$ with break value = 0
           then
             $A \leftarrow A$ with $x$ flipped;
           else
             with probability $p$
                $x \leftarrow$ a variable in $c$ chosen at random;
             with probability $(1-p)$
                $x \leftarrow$ a variable in $c$ with smallest break value;
             $A \leftarrow A$ with $x$ flipped;
           end if
         end for
         if $(A \notin \Omega)$ and $(\exists \Omega[k] | C(\Omega[k]) < C(A))$ then $\Omega[k] \leftarrow A$;
       end for
       return “Not found”
end
```

Fig. 2 The BB_WalkSAT procedure

4 Computational Experience

The computing platform used to perform the experiments is a 3.40 GHz Intel Pentium D Processor with 1 GB of RAM running Linux. Programs are coded in C language. We compared the BB_PMSAT results with the optimal solutions of a test problems' set of the OR-Library (series D and E). Series D consists of 20 problems with 1000 nodes, arcs varying from 1250 to 25000, and terminals from 5 to 500. Series E consists of 20 problems of 2500 nodes, arcs varying from 3250 to 62500, and terminals from 5 to 1250. In order to test the effectiveness of the proposed approach, we compared BB_PMSAT results with those obtained with the Tabu Search method called Full Tabusteiner (F-Tabu) from Gendreau *et al.* [5], which is one of the best heuristic approach for the STP in terms of solution quality. BB_PMSAT was also compared to one of the best Genetic Algorithms (GA-E) that has solved STP, which is due to Esbensen [3].

Table 1 Results for series D of STP.

			WalkSAT		BB_PMSAT	
Instance	GA-E(%)	F-Tabu (%)	(%)	CPU secs	(%)	CPU secs
D1-20	0.58	0.10	4.19	4.50	0.11	3.14
E1-20	0.42	0.31	4.65	10.67	1.19	8.29
Best approach	18/40	28/40	16/40		28/40	

PMSAT and MAX-SAT instances were generated from STP instances using the reduction described in Section 2. The number k of pre-computed paths between each pairs of nodes was fixed to 10. The total number of tries for each run of BB_PMSAT was shared between its two phases. Let r be the first phase length ratio of the total run length and pb the ratio of pseudo-backbone size to the number of variables n. BB_PMSAT was tested using the following parameter settings: $r = 0.6$, $pb = 0.5$ (values of r and pb are recommended in [9]), $MaxFlips = 10^5$, and $MaxTries = 100$ (shared between $MaxTries1$ and $MaxTries2$). WalkSAT was tested using a noise parameter $p = 0.5$ (recommended by the authors [12]) and the same values of $MaxFlips$ and $MaxTries$ used in BB_PMSAT.

Table 1 shows the mean results in terms of solution quality (in error percentage w.r.t. the optimum) for the series D and E of STP and their comparison with the Tabu Search method F-Tabu [5] and the Genetic Algorithm GA-E [3]. The results reported for WalkSAT and BB_PMSAT include average CPU time required over 10 runs. CPU times of the methods GA-E and F-Tabu are omitted because of a difference in the evaluation of the processing times. BB_PMSAT and F-Tabu were the best approaches in 28 times and gave clearly better solutions than GA-E (18 times) and WalkSAT (16 times). In terms of solution quality, the average results given by BB_PMSAT and F-Tabu for series D were comparable. However, for series E, F-Tabu outperformed BB_PMSAT. We expect that greater exploration of the parameters of BB_PMSAT may yield still better results.

Overall, BB_PMSAT found more optimal solutions than WalkSAT on all instances in less average CPU time. Indeed, the average CPU time achieved by BB_PMSAT and WalkSAT on all the problems is 5.71 secs and 7.58 secs, respectively. These positive results can demonstrate the superiority of the PMSAT encoding and the use of BB_PMSAT search procedure in comparison to the MAX-SAT encoding and the use of WalkSAT procedure for STP. BB_PMSAT's overall performance is comparable to the Tabu Search method F-Tabu.

5 Conclusions

In this paper we have examined a logic-based method to solve STP. We have considered MAX-SAT and Partial MAX-SAT encodings of STP. Empirical evaluation has been conducted on these encodings using two heuristic algorithms: BB_PMSAT and WalkSAT. BB_PMSAT relies on a pseudo-backbone sampling to guide the search trajectory through near-optimal solutions. We have reported some computational results showing that BB_PMSAT can solve large instances of STP. It appears that solving STP as a PMSAT using BB_PMSAT is more effective than solving it as a MAX-SAT using WalkSAT. Results are compared to those of specialized STP algorithms (F-Tabu and GA-E). The performance of BB_PMSAT is better than that of GA-E and close to that of F-Tabu in terms of solution quality . We have tested larger STP instances and obtained good results. However, the lack of space prevents us to present them and to discuss the choice of *k*, the number of precomputed shortest paths. We can conclude that the reduction of STP into PMSAT and the use of BB_PMSAT represent an effective means of solving STP.

References

1. Cha, B., Iwama, K., Kambayashi, Y., Miyazaki, S.: Local search algorithms for Partial MAXSAT. In Proc. AAAI-97, (1997), 263–268
2. Cook, S.A.: The complexity of theorem proving procedures. In Proc. 3rd ACM Symposium of the Theory of Computation, (1971) 263–268
3. Esbensen, H.: Computing near-optimal solutions to the Steiner problem in graphs using a genetic algorithm. Networks 26, (1995) 173–185
4. Fu, Z., Malik, S.: On solving the Partial MAX-SAT problem. In Proc. SAT'06, LNCS 4121, (2006) 252–265
5. Gendreau, M., Larochelle, J.-F., Sansò, B.: A tabu search heuristic for the Steiner tree problem. Networks 34(2), (1999) 162–172
6. Jiang, Y., Kautz, H.A., Selman, B.: Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. In Proc. 1st Inter. Joint Workshop on Artificial Intelligence and Operations Research, (1995)
7. Kahng, A.B., Robins, G.: On optimal interconnections for VLSI. Kluwer Publishers, (1995)
8. Karp, R.M.: Reducibility among combinatorial problems. In E. Miller and J.W. Thatcher, eds, Complexity of Computer Computations, Plenum Press, (1972) 85–103
9. Menaï, M.B., Batouche, M.: A backbone-based co-evolutionary heuristic for Partial MAX-SAT. In Proc. EA-2005, LNCS 3871, (2006) 155–166, Springer-Verlag
10. Nguyen, U.T.: On multicast routing in wireless mesh networks. Computer Communications 31(7), (2008), 1385–1399
11. Osborne, L.J., Gillett, B.E.: A comparison of two simulated annealing algorithms applied to the directed Steiner problem on networks. ORSA Journal on Computing 3, (1991), 213–225
12. Selman, B., Kautz, H.A., Cohen, B.: Noise strategies for improving local search. In Proc. AAAI-94, (1994) 337–343
13. Slaney, J., Walsh, T.: Backbones in optimization and approximation. In Proc. IJCAI-01, (2001) 254–259
14. Telelis, O., Stamatopoulos, P.: Heuristic backbone sampling for maximum satisfiability. In Proc. 2nd Hellenic Conference on AI, (2002) 129–139