Duplex Encoding of Antibandwidth Feasibility Formulas Submitted to the SAT Competition 2020

Katalin Fazekas [1] Markus Sinnl [2] Armin Biere [1] Sophie Parragh [2]
[1] Institute for Formal Models and Verification [2] Institute of Production and Logistics Management
Johannes Kepler University Linz, Austria

I. The Antibandwidth Problem

This benchmark set is originating in our recent work on the antibandwidth problem (in short ABP) presented in [1]. The ABP is a max-min optimization problem where, for a given graph $G = (V, E)$, the goal is to assign a unique label from the range $[1, \ldots, |V|]$ to each vertex $v \in V$, such that the smallest difference between labels of neighbouring nodes is maximal. Applications of the ABP include for example scheduling, obnoxious facility location, radio frequency assignment.

To solve the ABP, in [2] an iterative solution method was proposed, where each iteration asked whether there exists a labelling to the graph, s.t. the smallest difference between labels of neighbours is greater than k. Finding the highest k where the answer is still affirmative determines the optimal solution of the ABP. In [1] we slightly refined the proposed formalization of [2] s.t. the question of each iteration can be stated as combinations of at-most-one constraints sliding over k-long sequences of binary variables. All in all, a feasibility query for a given k consists of the following constraints:

$$\forall \ell \in \{1, \ldots, |V|\} : \sum_{i \in V} x_i^\ell = 1 \qquad \text{(Labels)}$$

$$\forall i \in V : \sum_{\ell \in \{1,\ldots,|V|\}} x_i^\ell = 1 \qquad \text{(Vertices)}$$

$$\forall \{i, i'\} \in E, 1 \le \lambda \le |V| - k : \qquad \text{(Obj}_k\text{)}$$
$$\sum_{\ell=\lambda}^{(\lambda+k)} x_i^\ell \le 1 \wedge \sum_{\ell=\lambda}^{(\lambda+k)} x_{i'}^\ell \le 1 \wedge \left(\sum_{\ell=\lambda}^{(\lambda+k)} x_i^\ell \le 0 \vee \sum_{\ell=\lambda}^{(\lambda+k)} x_{i'}^\ell \le 0 \right),$$

where binary variables $x_i^\ell = 1$ ($i \in V$, $\ell \in \{1, \ldots, |V|\}$) if and only if vertex i is assigned label ℓ. Constraints (Labels) make sure that each label is used only once and constraints (Vertices) ensure that each node $i \in V$ gets assigned one label. Constraints (Obj$_k$) forbid for each neighbouring node to assign two labels from any k-wide range of labels.

II. SAT Encoding

In [1] we defined a so-called staircase at-most-one constraint set (SCAMO) over a sequence of Boolean variables $X = \langle x_1\, x_2 \cdots x_n \rangle$ for a given width k (where $1 < k \le n$) as

$$\mathrm{SCAMO}(X, k) = \bigwedge_{i=0}^{(n-k)} \left(\sum_{j=i+1}^{(i+k)} x_j \le 1 \right).$$

Then we proposed a linear size SAT encoding of this constraint set. The main idea of the encoding is to slice up the n-long sequence of Boolean variables into M k-long sequences and build up the complete SCAMO constraint set as a combination of smaller at-most-one and at-most-zero constraints, such that these smaller constraints can be efficiently shared and reused. Each smaller at-most-one and at-most-zero constraint is translated to SAT with standard BDD-based methods (see [3], [4]). However, each of these constraints is encoded twice, first considering a variable order $x_{i+1} < x_{i+2} < \ldots < x_{i+k}$ in the BDD construction, then the reverse of that order. The result is an arc-consistent encoding of a $\mathrm{SCAMO}(X, k)$ constraint set with approximately $11 \cdot M \cdot k$ clauses, where $M = \lceil \frac{n}{k} \rceil$.

Consider a feasibility query with value k of the ABP over a graph G, as it was formalized in the previous section. We first encode for each vertex $v \in V$ a $\mathrm{SCAMO}(X, k)$ set, where X is a $|V|$-long sequence of Boolean variables $\langle x_v^{\ell_1}\, x_v^{\ell_2} \cdots x_v^{\ell_{|V|}} \rangle$ representing all possible labels of v. Then, we simply add the disjunction of the corresponding at-most-zero constraints belonging to neighbouring nodes to encode all the constraints of (Obj$_k$). The exactly-one constraints of (Vertices) are encoded as conjunctions of the constructed smaller at-most-one constraints and negations of at-most-zero constraints. The other set of exactly-one constraints in (Labels) is encoded as conjunction of at-least-one constraints and the product encoding of at-most-one constraints introduced in [5].

The resulting formula consists mostly of unit, binary and ternary clauses from the SCAMO constraints. The larger clauses are either $|V|$-long clauses from the at-least-one part of the exactly-one constraints in (Labels), or M-long clauses from the at-least-one part of constraints (Vertices).

III. Generated Instances

To evaluate in [1] our proposed SAT-based solution method for the ABP, we implemented a C++ tool called `DuplexEncoder`. It takes as input a graph and a lower (LB) and an upper bound (UB) of the antibandwidth. For each value k starting from LB, it encodes the ABP as we presented in the previous section and invokes a SAT solver on it to decide

Supported by the Austrian Science Fund (FWF) under projects W1255-N23, S11408-N23 and by the LIT AI Lab funded by the State of Upper Austria.

feasibility. If the formula is SAT, it moves to the next k, if it is UNSAT, the previous value was optimal and stops.

We experimented in [1] on the 24 graphs of the Harwell-Boeing Sparse Matrix Collection [6]. Our benchmark set was generated with `DuplexEncoder` from 12 graphs of [6], mostly where we could not solve the ABP in 1800 seconds in [1]. For each of these graphs first we considered every consecutive k values in a very wide range around the value of $(\mathrm{LB} + \mathrm{UB})/2$ (see [1] for each value of LB and UB) and generated the corresponding SAT formula of the ABP for each.

From the resulting formulas we identified the "interesting" problems based on the description of the expected benchmarks on the homepage of the SAT competition. First we dropped all those formulas that Minisat [7] with default settings could solve in less than a minute. The remaining 539 formulas were tried to be solved in less than an hour with CaDiCaL [8], [9] on our cluster with Intel Xeon E5-2620 v4 @ 2.10GHz CPUs. CaDiCaL solved all in all 121 problems (91 SAT and 30 UNSAT) successfully and the required solving times of these instances ranged between few seconds and one hour with a very balanced distribution.

Due to our source AB problem, we know that whenever a formula with a specific k-value is satisfiable, all the other formulas with smaller k from the same graph must be SAT as well. Further, an unsatisfiable k means that every formula with larger k of the same graph is UNSAT as well. Based on these observations, we identified another 44 problem instances that must be satisfiable, but CaDiCaL could not solve. Since most of the unsatisfiable formulas were immediately solved with Minisat, we could not find further ones with this approach.

Further, for most of the graphs we collected two more yet unsolved instances that are hopefully not too far in difficulty from the solved ones. More precisely, for each graph where it was possible, we considered the highest k-value where the answer was SAT and picked the next formula (i.e. $k + 1$). Similarly, we considered the lowest formula where the answer was UNSAT and picked the next one (i.e. $k - 1$). For the resulting 22 formulas we do not know whether they are satisfiable or not, but hope that sooner or later they will be solved.

All in all, we included in our submission the 121 "interesting" problems together with the 44 unsolved satisfiable formulas (having at the end 135 SAT and 30 UNSAT problems) and the 22 unsolved, completely unknown problems. The result is a benchmark set of 187 problem instances, where approximately 12% is unknown whether satisfiable or not, 72% is SAT, the remaining 16% is UNSAT and CaDiCaL can solve approximately 65% of the problems in one hour. The file name of each submitted problem follows the `abw-[source-graph].w[`k `of query].cnf` pattern. The source code of the `DuplexEncoder` tool from [1] and the script that was used to generate the benchmarks are both available at http://fmv.jku.at/duplex/.

REFERENCES

[1] K. Fazekas, M. Sinnl, A. Biere, and S. Parragh, "Duplex encoding of staircase at-most-one constraints for the antibandwidth problem," in *CPAIOR*, ser. LNCS. Springer, 2020, to appear.

[2] M. Sinnl, "A note on computational approaches for the antibandwidth problem," *CoRR*, vol. abs/1910.03367, 2019. [Online]. Available: http://arxiv.org/abs/1910.03367

[3] I. Abío, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and V. Mayer-Eichberger, "A new look at BDDs for pseudo-boolean constraints," *J. Artif. Intell. Res.*, vol. 45, pp. 443–480, 2012. [Online]. Available: https://doi.org/10.1613/jair.3653

[4] N. Eén and N. Sörensson, "Translating pseudo-boolean constraints into SAT," *JSAT*, vol. 2, no. 1-4, pp. 1–26, 2006. [Online]. Available: https://satassociation.org/jsat/index.php/jsat/article/view/18

[5] J. Chen, "A new sat encoding of the at-most-one constraint," *Proc. Constraint Modelling and Reformulation*, 2010.

[6] E. Rodriguez-Tello, H. Romero-Monsivais, J. Ramrez-Torres, and F. Lardeux, "Harwell-boeing graphs for the CB problem," https://www.researchgate.net/publication/272022702_Harwell-Boeing_graphs_for_the_CB_problem, 2015.

[7] N. Eén and N. Sörensson, "An extensible sat-solver," in *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds., vol. 2919. Springer, 2003, pp. 502–518. [Online]. Available: https://doi.org/10.1007/978-3-540-24605-3_37

[8] A. Biere, "CaDiCaL at the SAT Race 2019," in *Proc. of SAT Race 2019 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Series of Publications B, M. Heule, M. Järvisalo, and M. Suda, Eds., vol. B-2019-1. University of Helsinki, 2019, pp. 8–9.

[9] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020," in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, M. Heule, M. Järvisalo, M. Suda, M. Iser, and T. Balyo, Eds., 2020.