Project report

IN4336 Combinatorial Algorithms
Project 3: Graph Coloring: applications, approximations and solvers

David Hoepelman
1521969

Luis Garcia
4062949

ABSTRACT

1. INTRODUCTION

Solvers for NP-Hard problems can be used to solve instances of other NP-Hard problems by polynomially transforming the other problem instance. This is called an encoding. This possibility makes it feasible to make highly-optimized solving algorithms called solvers for certain relatively general NP-Hard problems.

For most popular problems multiple solvers exists, and benchmarks are available or contests are held to compare them. However comparisons between problems are rarer and in this project we will do such a comparison. We want to see if some problems are more suitable to encode other problems in or if the choice of problem to encode to makes little impact.

Another question is whether translations into intermediary problems add significant overhead. While we know that polynomial transformations do not increase the complexity of a NP-Hard problem, it might still have a practical impact in how fast a problem instance can be solved or what the largest solvable problem instance is. With this we want to see if encodings can be viably done in smaller easier steps, or if it is better to encode the problem directly into the target problem.

We formulated the following research question for this project:

RQ1 Is there a speed difference between formalisms when solving an identical problem instance?

RQ2 Are intermediate transformations viable?

RQ2.1 Does an intermediate transformation significantly decrease the speed with which problem instances can be solved?

RQ2.2 Does an intermediate transformation decrease the size of the largest possible problem instances which can be solved?

In answering our research questions we will focus on solvers for *Integer Linear Programming* (ILP), *Satisfiability* (SAT) and *Satisfiability Modulo Theories* (SMT).

As problems to encode we took *Resource Constrained Project Scheduling Problem* (RCPSP) and *Graph Coloring* (GC).

2. ENCODINGS

2.1 Graph Coloring

The graph coloring problem is one of the Karp's 21 original NP-Complete problems [4]. Graph coloring exists in several variants, but they are all easily reducible to the vertex coloring problem. We chose the optimization variant of the vertex coloring problem which is informally defined as given an undirected graph $G = (V, E)$, find the minimal number of colors k needed to color each vertex so that no adjacent vertex shares a color. More formally:

$$\begin{array}{ll}
G = (V, E) & E \subseteq V^2 \\
map : V \rightarrow [1, k] & k \in \mathbb{N} \\
map(v) \neq map(w) & (v, w) \in E \\
VC = min\{k | map : V \rightarrow [1, k]\} &
\end{array}$$

2.1.1 SAT

2.1.2 ILP

The combinatorial problem of graph coloring can also be formulated in 0-1 ILP. An advantage of using ILP is that it can solve optimization problems. The formulation of the graph coloring problem in 0-1 ILP can be done by taking the set of nodes V = 1,...,n and the number of colors k = |V| = n. Then, each color is represented by a binary variable y_k, with k = 1,...,n, that indicatate whether color k is used (y_k = 1) or not (y_k = 0). An objective function is formed by adding up these variables ($\sum_{k=1}^{n} y_k$) to obtain the minimum number of these variable equal to zero. Furthermore, a binary variable x_{i_k} is used for each node-color combination to indicate whether node i has received a color k (x_{ik} = 1) or not (x_{ik} = 0). The resulting 0-1 ILP in formal description:

$$\begin{array}{llrll}
min \sum_{k=1}^{n} y_k & s.t. & (1) & \sum_{k=1}^{n} x_{ik} = 1 & i = 1, ..., n \\
 & & (2) & x_{ik} - y_k \leq 0 & i,k = 1, ..., n \\
 & & (3) & x_{ik} + x_{jk} \leq 1 & (i, j) \in E, k = 1, ..., n \\
 & & (4) & 0 \leq x_{ik}, y_k \leq 1 & \\
 & & (5) & x_{ik}, y_k \in \mathbb{Z} &
\end{array}$$

2.1.3 SMT

2.2 RCPSP

2.2.1 SAT

2.2.2 ILP

3. EXPERIMENTAL SETUP

4. RESULTS

5. RELATED WORK

[2] [1] [14] [12] [3] [5] [7] [10] [8] [9] [13] [6] [11]

6. CONCLUSION

7. REFERENCES

[1] Abío, I., and Stuckey, P. J. Encoding linear constraints into sat. In *Principles and Practice of Constraint Programming* (2014), Springer, pp. 75–91.

[2] Alves, R., Alvelos, F., and Sousa, S. D. Resource constrained project scheduling with general precedence relations optimized with sat. In *Progress in Artificial Intelligence*. Springer, 2013, pp. 199–210.

[3] Horbach, A. A boolean satisfiability approach to the resource-constrained project scheduling problem. *Annals of Operations Research 181*, 1 (2010), 89–107.

[4] Karp, R. M. *Reducibility among combinatorial problems*. Springer, 1972.

[5] Klein, R., and Scholl, A. Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research 112*, 2 (1999), 322–346.

[6] Malaguti, E., and Toth, P. A survey on vertex coloring problems. *International Transactions in Operational Research 17*, 1 (2010), 1–34.

[7] Mingozzi, A., Maniezzo, V., Ricciardelli, S., and Bianco, L. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science 44*, 5 (1998), 714–729.

[8] Ramani, A., Aloul, F. A., Markov, I. L., and Sakallah, K. A. Breaking instance-independent symmetries in exact graph coloring. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings* (2004), vol. 1, IEEE, pp. 324–329.

[9] Schutt, A., Feydy, T., Stuckey, P. J., and Wallace, M. G. Explaining the cumulative propagator. *Constraints 16*, 3 (2011), 250–282.

[10] Sharma, P. C., and Chaudhari, N. S. Polynomial 3-sat encoding for k-colorability of graph. *Special Issue of International Journal of Computer Application on Evolution in Networks and Computer Communications (1)* (2011), 19–24.

[11] Texas University CS395T, . Reducing graph coloring to sat.

[12] Velev, M. N., and Gao, P. Exploiting hierarchical encodings of equality to design independent strategies in parallel smt decision procedures for a logic of equality. In *High Level Design Validation and Test Workshop, 2009. HLDVT 2009. IEEE International* (2009), IEEE, pp. 8–13.

[13] Wille, R., Grosse, D., Soeken, M., and Drechsler, R. Using higher levels of abstraction for solving optimization problems by boolean satisfiability. In *Symposium on VLSI, 2008. ISVLSI'08. IEEE Computer Society Annual* (2008), IEEE, pp. 411–416.

[14] Wu, H., Monahan, R., and Power, J. F. Exploiting attributed type graphs to generate metamodel instances using an smt solver. In *Theoretical Aspects of Software Engineering (TASE), 2013 International Symposium on* (2013), IEEE, pp. 175–182.