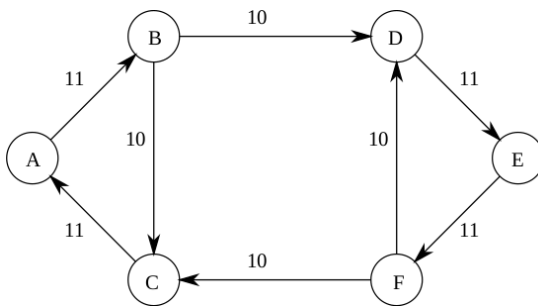


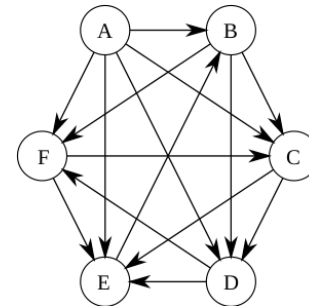
# Inspectable Incrementality in Minimum Feedback Arc Set Solving

Colin Shea-Blymyer  
sheablyc@oregonstate.edu  
Oregon State University  
Corvallis, Oregon, USA

Jeffrey M. Young  
youngjef@oregonstate.edu  
Oregon State University  
Corvallis, Oregon, USA



(a) A three cycle graph which demonstrates that greedily selecting the minimum weighted edge will not provide the minimum feedback arc set.



(b) A four cycle graph where each cycle depends on the edge (E,B).

## ABSTRACT

Abstract here

## KEYWORDS

datasets, neural networks, gaze detection, text tagging

### ACM Reference Format:

Colin Shea-Blymyer and Jeffrey M. Young. 2021. Inspectable Incrementality in Minimum Feedback Arc Set Solving. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 INTRODUCTION

The minimum feedback arc set problem is a canonical NP-Complete problem given by [1]. Worse still, the problem is APX-hard yet finding the minimum feedback arc set of a directed graph is desirable for many domains: such as certain rank-choice voting systems, tournament ranking systems, and dependency graphs in general.

Solutions to the minimum feedback arc set problem are commonly implemented in widely used graph libraries yet all suffer from a distinct flaw. While each implementation provides a solution, the implementations only do so without allowing the user to inspect intermediate steps; which might contain useful domain information. For example, the graph Figure 1b displays a directed graph which encodes a single win tournament system. Each vertex is a player

and each edge encodes a win over a contestant, for example we see that player E beat contestant B. Observe that the Figure 1b contains cycles which implies that there is not a linear ordering amongst the contestants of the tournament, and so any ranking of the contestants would violate a win of one contestant over another. Removing the minimum feedback arc set would remove the cycles, which yields a linear order in the tournament that violates the fewest number of wins. In this example, intermediate results are edges which compose the minimum feedback arc set, thus if one had access to the intermediate results one might choose to rematch the opponents rather than remove the win. Crucially, the result of the rematch may change the final minimum feedback arc set. Such a procedure could therefore increase the confidence in the results of the tournament.

To allow for *inspectable incrementality* we propose a novel direction in solving the minimum feedback arc set problem based on recent advances in satisfiability solving (SAT) and satisfiability-modulo theories (SMT) solving. Our approach is to utilize an SMT solver to detect cycles and minimize the weight of the feedback arc set. Incrementality in this approach is given through use of an *incremental* SMT solver and generation of *unsatisfiable cores*. A minimum unsatisfiable core is the minimum set of clauses in a SAT or SMT formula which prevent a SAT or SMT solver run from finding a satisfiable assignment. An incremental solver provides the end-user the ability to add or remove constraints and thereby direct the solver during runtime. Incrementality in our approach is crucial as it the end-user decision points to interact with the solution process. Thus, an end-user might observe a unsatisfiable core which corresponds to a cycle and deliberately resample *only* the edges in the discovered cycle.

The approach has benefits ...

We make the following contributions:

- (1) Gadgets

but is it novel?

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

Conference'17, July 2017, Washington, DC, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

add citation

cite Viggo Kann thesis

cite

- (2) Evaluation of unsatisfiable cores for this problem domain  
 (3) empirical evaluation of this novel method

again, is  
it novel?

## 2 BACKGROUND AND RELATED WORK

In this section, we provide background on the minimum feedback arc set problem, incremental SAT and SMT solving, and unsatisfiable cores. We begin with preliminaries on graphs and feedback arc sets.

On a directed graph  $G(V, E)$  a *feedback arc set*  $S$  is a set of edges in  $E$ , such that removing them from the graph  $G$  results in an acyclic graph. The minimum feedback arc set problem finds the  $S^*$  of the minimum size. For weighted graphs, one might desire  $S^*$  to have the minimum total weight of any  $S$ . Finding this  $S^*$  is the minimum weighted feedback arc set problem.

Hi Mike! This is a special note environment just for you. From this point we plan on giving an overview of the necessary interface for sat and smt solving. We're planning on adapting this from Jeff's PhD thesis, so it won't be from whole cloth.

## 3 GADGETS

**Listing 1: Cycle check gadget. `v_from` and `v_to` are symbolic variables in the solver. `name` is the constraint label in the solver. The cycle check replaces each edge with the constraint that previous vertices must be a lesser integer value than future vertices**

```
def cycle_check(solver, v_from, v_to, name):
    solver.assert_and_track(v_from < v_to, name)
```

We have a working implementation of a cycle detection gadget, strategies for edge relaxation in the unsatisfiable core, and incremental constraint solving in the SAT and SMT solver. See for the source code.

## 4 CASE STUDIES

To evaluate this approach, we construct a prototype SMT-enabled algorithm and assess the prototype on Erdős-Rényi graphs. Erdős-Rényi graphs are generated according to two parameters:  $p$ , a metric of connectedness for the graph, and  $k$  is the number of vertices in the graph. In addition, we add a parameter  $s$  which represents the size of the range of possible weights on edges in a graph.

With these parameters we employ random generation to construct sample graphs. We are interested in the individual effect  $k$ ,  $p$  and  $s$  have on runtime, in addition to the interactions effects between each parameter. Consider the case where  $p$  and  $k$  are left unbound, yet  $s$  is set to 1. This is the specific case where solving the minimum weighted feedback arc set problem solves the minimum feedback arc set problem. Thus, by setting  $s$  to 1 we generate graphs to solve the minimum feedback arc set problem. Consider the case where  $s$  is larger than one. In this case, edges must possess positive weights and the weighted minimum feedback arc set may be different than the minimum feedback arc set.

Hi Mike. The rest of the section is describing each case individually and then research questions. We don't have these spelled out right now. What we hypothesize is that at some combination of  $k$ ,  $p$  and  $s$  the solver runtime will become infeasible. Then we can make determinations on the viability of this method in different domains. For example, most tournaments will be a bounded number of participants, and score ranges, but may have maximum connectivity. Thus, we would be able to make a conclusion on the viability of our method for the tournament domain.

## 5 RESULTS AND DISCUSSION

It was good enough to graduate