Greedy BMC Solver for the Independent Set Reconfiguration Problem

Takahisa Toda[1]

[1]Graduate School of Informatics and Engineering, The University of Electro-Communications

June 27, 2022

1 Solver Description

Our solver, named *greedy BMC solver*, is implemented as the integration of a greedy heuristic search and bounded model checking. If the greedy search finds a reconfiguration sequence in success, the solver returns it, and otherwise, goes into the BMC computation with the inititial state replaced by the end state of the "incomplete" sequence, i.e. the sequence that is not reachable to the target state. In the BMC computation, it first transforms independent-set-reconfiguration problem instances into bounded model checking instances, then applies a bounded model checker with a fixed bound, and obtains reconfiguration sequences as counterexamples to the LTL formulas [**?**]. To boost the performance, the modeling phase into BMC is done by exploiting a small edge clique cover for the input graph. The solver is often powerful in such graphs as having a large number of edges but covered by a small number of cliques, as confirmed in some benchmark instances of this challenge.

For the bounded model checking and the edge clique cover computation, our solver uses the following programs inside it.

- ECC 8: the program for the edge clique cover computation [**?**], obtained from `https://github.com/Pronte/ECC` .
- NuSMV version 2.6.0: the program for the bounded model checking [**?**], obtained from `https://nusmv.fbk.eu/`.

As is the case for bounded model checking, our solver needs a maximum number of transitions to be fixed and it is able to answer yes or no only for the existence of reconfiguration sequences of length less than or equal to the step size. As stated above, our solver is combined with the heuristic search and hence, the obtained sequence are no longer quaranteed to be the shortest.

2 Metric

- (solver#existent)
- (solver#shortest)
- (solver#longest)

3 Computation Environment

- Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz
- 32G bytes memory
- Ubuntu 20.04

4 Usage

ECC8.jar and NuSMV are not included in this submission:

```
~/2022solver# ls bin/
col2nde  compact  ecc2mat  greedy  greedy_bmc-solver_ecc.sh
id-tj-ecc  out2ans_uniform.awk  postproc  st2ltl.awk
```

So, at first, obtain the above two programs and place them in bin.

Check if there exists tmp in /2022solver, where our solver generates various temporary files.

```
~/2022solver$ ls
README.md  bin  example  run.sh  tmp
```

Run run.sh as follows.

```
~/2022solver$ ./run.sh example/hc-toyyes-01.col example/hc-toyyes-01_01.dat
```