*Proceedings of SAT Competition 2018: Solver and Benchmark Descriptions*, volume B-2018-1 of Department of Computer Science Series of Publications B, University of Helsinki 2018.

# GrandTour$^{\mathbf{obs}}$ Puzzle as a SAT Benchmark

Md Solimul Chowdhury
*Department of Computing Science*
*University of Alberta*
Edmonton, Alberta, Canada
mdsolimu@ualberta.ca

Martin Müller
*Department of Computing Science*
*University of Alberta*
Edmonton, Alberta, Canada
mmueller@ualberta.ca

Jia-Huai You
*Department of Computing Science*
*University of Alberta*
Edmonton, Alberta, Canada
jyou@ualberta.ca

***Abstract*—GrandTour[1] is a puzzle game, which is usually played in a rectangular grid of points, where a player is posed with the challenge to find a closed loop that goes through each point exactly once. In GrandTour, the player is allowed to connect any pair of points to solve the game. We consider a slight variation of the GrandTour puzzle game, where a player is allowed to connect a restricted subset of pairs of points, as there are obstacles between some of the points. We call this puzzle GrandTour with obstacles or GrandTour$^{\text{obs}}$. For SAT competition-2018, we provide 20 SAT instances of the GrandTour$^{\text{obs}}$ puzzle.**

## I. Encoding and Instance Generation

*a)* **GrandTour$^{\mathbf{obs}}$** *as Hamiltonian Cycle Problem:* The problem of determining whether a Hamiltonian cycle exists in a given graph is a well-known NP-complete problem. This problem deals with the following question: is there a cycle in a given graph, in which each vertex is visited exactly once? While solving a **GrandTour** puzzle is equivalent to finding a Hamiltonian cycle in a complete graph[2] induced from its completely connected puzzle grid, solving **GrandTour$^{\mathbf{obs}}$** problem is equivalent to finding a Hamiltonian cycle in an incomplete graph, which is induced from its incompletely connected puzzle grid. If a Hamiltonian cycle exists in the graph induced from the puzzle grid of a **GrandTour$^{\mathbf{obs}}$** puzzle, then the puzzle is solvable, otherwise it is not.

*b) Problem Encoding:* A ground normal logic program $P$ can be converted into a SAT formula $S$, from the Clark's completion $C$ of $P$. For example, the Clark's completion of $P = \{a \leftarrow b., a \leftarrow c.\}$ is $C = \{a \leftarrow b \vee c.\}$ and $C$ can be expressed into the SAT formula $S$ in CNF form as: $(\neg b \wedge \neg c) \vee a \equiv (\neg b \vee a) \wedge (\neg c \vee a)$. This technique is utilized in ASSAT [1], a solver for answer set programs using a SAT solver as the underlying inference engine. Given a ground normal logic program $P$, ASSAT computes its stable models by using a SAT solver. As part of its solving process, ASSAT produces a SAT formula $S$ from $P$. We exploit this feature of ASSAT to generate SAT instances of the **GrandTour$^{\mathbf{obs}}$** problem from the normal logic program encoding of Hamiltonian cycle as proposed in [2].

*c) Instance Generation:* For the **GrandTour$^{\mathbf{obs}}$**, a puzzle grid of size $x \times y$ can be induced from a graph with $n$ nodes, where $n$ is an even number and $n = x * y$. The website for ASSAT[3] contains 32 normal logic program instances for the Hamiltonian cycle problem from two graphs with 60 and 50 vertexes, respectively. We treat these instances as the **GrandTour$^{\mathbf{obs}}$** instances of grid size $10 \times 6$ and $10 \times 5$, respectively.

TABLE I: Details of the 20 SAT instances for the **GrandTour$^{\mathbf{obs}}$** benchmark; An instance $gto\_p\mathbf{x}c\mathbf{y}$ is based on a grid which has $x$ points and $y$ pairs of points are connected.

| Problem/Grid Size | Variables/Clauses | MiniSAT cpuTime (s) | MiniSAT Results |
|---|---|---|---|
| gto_p60c229 / $10 \times 6$ | 1011/3473 | 101.95 | UNSAT |
| gto_p60c231 / $10 \times 6$ | 1015/3493 | 351.36 | UNSAT |
| gto_p60c241 / $10 \times 6$ | 1046/3673 | 481.81 | UNSAT |
| gto_p60c239 / $10 \times 6$ | 1038/3627 | 717.90 | UNSAT |
| gto_p60c231_1 / $10 \times 6$ | 1019/3517 | 833.89 | UNSAT |
| gto_p60c243 / $10 \times 6$ | 1054/3721 | 881.04 | UNSAT |
| gto_p60c233 / $10 \times 6$ | 1023/3533 | 1177.71 | UNSAT |
| gto_p60c234 / $10 \times 6$ | 1027/3577 | 1897.77 | UNSAT |
| gto_p60c235 / $10 \times 6$ | 1031/3583 | 3360.01 | UNSAT |
| gto_p60c238 / $10 \times 6$ | 1043/3649 | 4232.67 | UNSAT |
| gto_p60c295 / $10 \times 6$ | 635/3330 | 5000 | UNKNOWN |
| gto_p60c343 / $10 \times 6$ | 729/4563 | 5000 | UNKNOWN |
| gto_p50c291 / $10 \times 5$ | 607/3813 | 5000 | UNKNOWN |
| gto_p50c345 / $10 \times 5$ | 703/5413 | 5000 | UNKNOWN |
| gto_p50c311 / $10 \times 5$ | 623/4137 | 5000 | UNKNOWN |
| gto_p50c312 / $10 \times 5$ | 643/4325 | 5000 | UNKNOWN |
| gto_p50c314 / $10 \times 5$ | 635/4319 | 5000 | UNKNOWN |
| gto_p50c314_1 / $10 \times 5$ | 639/4339 | 5000 | UNKNOWN |
| gto_p50c345 / $10 \times 5$ | 605/3664 | 5000 | UNKNOWN |
| gto_p50c307 / $10 \times 5$ | 613/4036 | 5000 | UNKNOWN |

These 32 instances are known to be hard for SAT solvers. In our experiment, MiniSat could not solve any of those within 5000 seconds. So, these instances are not *interesting* for the SAT competition-2018. As per the requirement of the benchmark submission, to generate 10 *interesting* SAT instances, we took a graph $G$, namely *hard.1.graph* in the ASSAT website, and repeated the following steps until we get 10 *interesting* instances: (i) randomly remove a few arcs to produce a new graph $G'$, (ii) generate the SAT instance $S$ from the normal logic program encoding of the Hamiltonian cycle program along with $G'$ by using ASSAT, and (iii) test $S$ with MiniSat to see if it is *interesting*. The first 10 rows in Table 1 show these *interesting* instances. The next 10 rows give the details of the 10 hard instances taken from the ASSAT website.

[1] http://curiouscheetah.com/Museum/Puzzle/Grandtour
[2] In a complete graph, each pair of vertexes are connected by an edge.
[3] http://assat.cs.ust.hk/hardsat.html

## References

[1] F. Lin and Y. Zhao, "ASSAT: computing answer sets of a logic program by SAT solvers", Artif. Intell. , vol. 157, pp. 115–137, 2004.
[2] I. Niemelä, "Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm", Ann. Math. Artif. Intell., vol. 25, pp.241–273, 1999.