Encoding Selection for Solving Hamiltonian Cycle Problems with ASP

Liu Liu and Mirosław Truszczyński

Department of Computer Science
University of Kentucky

Answer Set Programming (ASP)

- Answer Set Programming (ASP) is a declarative formalism for solving hard search and optimization problems.
- Problems are modeled by answer set programs.
- Instances to a problem are modeled by sets of ground atoms.
- Answer sets represent solutions to the problem for that instance.

Ease of modeling, but

- Unlikely a single solver will uniformly outperform all others.
- A wrong parameter configuration of solver may make an excellent solver run "forever".
- Workarounds
  - solver selection — inspired by algorithm selection (Rice, 1976)
  - portfolio solving — claspfolio (Hoos et al., 2014)
  - automated parameter configuration — SMAC by (Hutter et al., 2011).
- The key idea:
  learn instance-driven performance models and use them, given an instance, to select a solver (or a parameter configuration) that might perform well on that instance.

Encodings affect runtime

- It is common for a problem to have several equivalent encodings in ASP.
- These encodings have the same answer sets but vary in runtime.
- No encoding is uniformly better than others when evaluated on broad classes of problem instances.
- Establish program rewriting heuristics to generate better performing programs, e.g., symmetry breaking, projection.

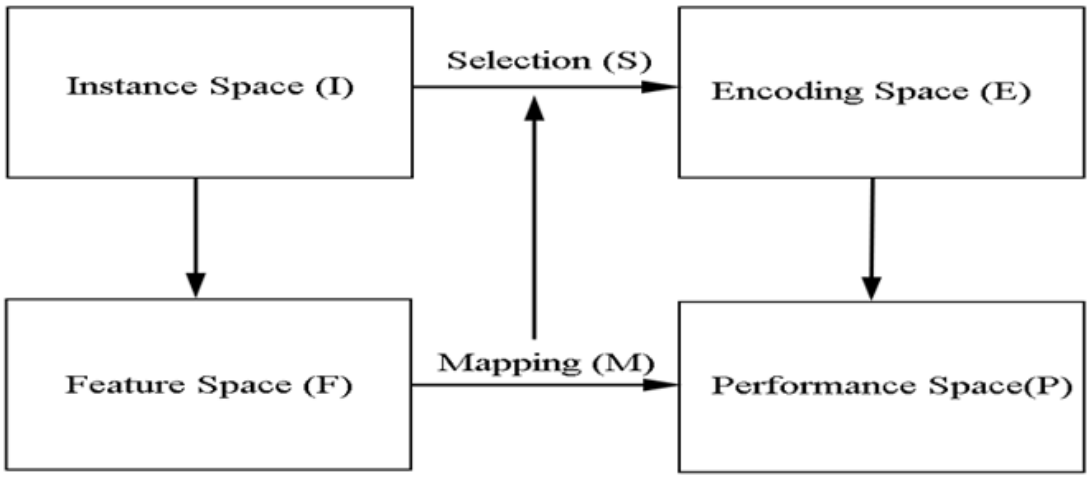
  queen(R,C1),queen(R,C2),row(R),col(C1),col(C2),C1!=C2.

  queen(R,C1),queen(R,C2),row(R),col(C1),col(C2),C1<C2.
- Even with heuristics, still no uniformly best encoding emerges.
- A workaround: Generate several encodings and then select the encoding likely to perform the best on a given instance via machine learning techniques.

Encoding selection

The idea based in algorithm selection by Rice (1976).

- Problem instances to solve (I)
- Encoding candidates to select from (E)
- Generate features for the problems (F)
- Run some instances to have performance history data (P)
- Build mapping from features of instances to performance data (M)
- Select the best encoding for a given instance based on the mapping (S)



We select hamiltonian cycle(HC) problems

- An important problem on graphs.
- Often used in the past ASP competition for benchmarks.
- Related to other problems, eg., travelling salesman problem and wire routing problems.

Our work

- Propose several equivalent encodings of the problem (E).
- Contribute several classes of structured instances and selected hard instances near the phase transition (I).
- Identify instance features that are relevant to the property of graph instances, combining with the features extracted by ASP feature generator *claspre* [1] (F).
- Build machine learning models to predict the behavior of each encoding based on their performance history, and then select encodings that lead to significant performance gains (P, M, S).

[1] `https://potassco.org/labs/claspre/`

Hamiltonian cycle instance format

- The HC problem takes directed graphs as input instances.
- An instance is given by the lists of nodes and edges(links), represented as ground atoms over a unary predicate node/1 and a binary predicate link/2.

```
node(1..5).
link(1,5).
link(3,2).
...
```

Hamiltonian cycle encodings

- The HC problem imposes constraints on a set of selected edges:
- Exactly one edge leaving each node.
- Exactly one edge entering each node.
- Every node must be reachable from every other node by a path of selected edges.

```
%encoding
{ hpath(X,Y) : link(X,Y) }=1:-node(X).
{ hpath(X,Y) : link(X,Y) }=1:-node(Y).
:- not reach(X,Y),node(X),node(Y)
%Define reachability
reach(X,Y):- hpath(X,Y).
reach(X,Z) :- reach(X,Y),hpath(Y,Z).
```

Encoding rewriting

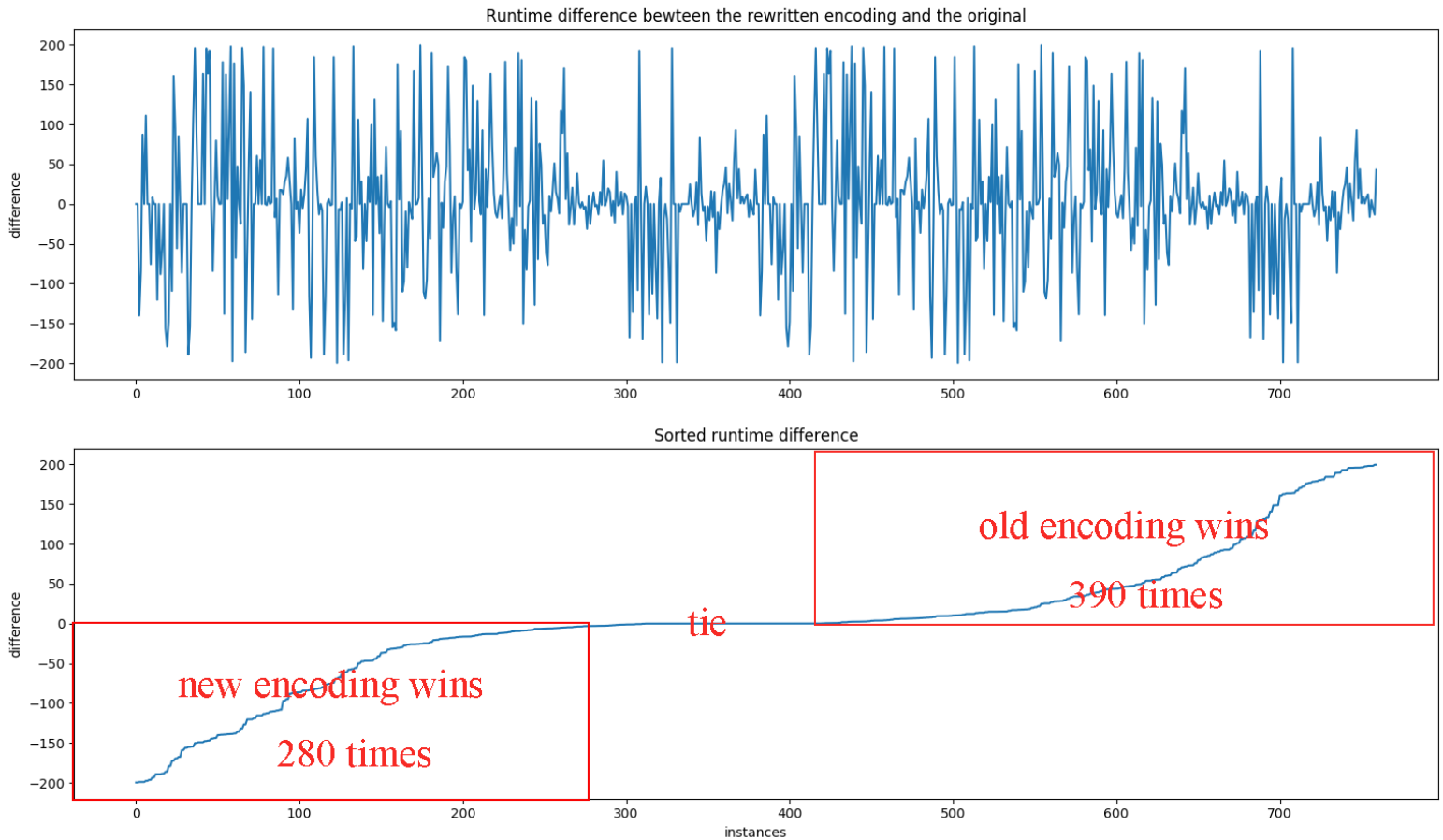
- Reachability can be modeled in a different way
- Nodes reachable from each other -> nodes reachable from node 1 (including 1).

```
%encoding rewriting
{ hpath(X,Y) : link(X,Y) }=1:-node(X).
{ hpath(X,Y) : link(X,Y) }=1:-node(Y).
:- not reach(X),node(X)
%Define reachability from 1
reach(X):- hpath(1,X).
reach(Y) :- reach(X),hpath(X,Y).
```

- Apply equivalence rewriting techniques, e.g. rewriting aggregate rules.

Influlence of rewriting

The following graph shows the runtime difference of two equivalent encodings when test on 784 instances (Instances will be explained later on).



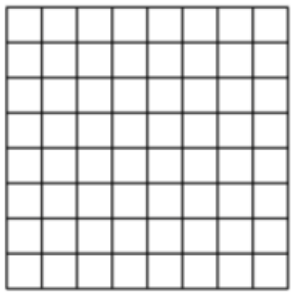
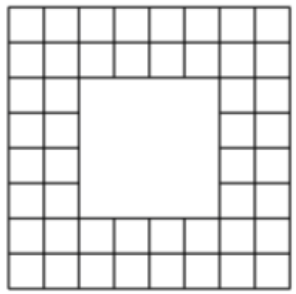
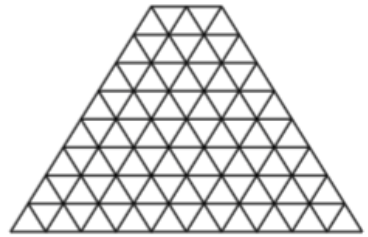
Performance of individual encoding and the oracle

- Six encoding candidates are selected, based on (784) instances solved percentage, average solved runtime and the number of wins.
- Oracle 98% vs. Encoding1 82.3%

Encoding	Solved Percentage%	Average Solved Runtime	Number of Wins
Encoding 1	82.3	84.1	102
Encoding 2	71.8	46.6	126
Encoding 3	55.3	29.7	110
Encoding 4	76.2	42.9	155
Encoding 5	55.4	31.9	120
Encoding 6	77.4	47.7	151
Oracle	98.0	22.8	

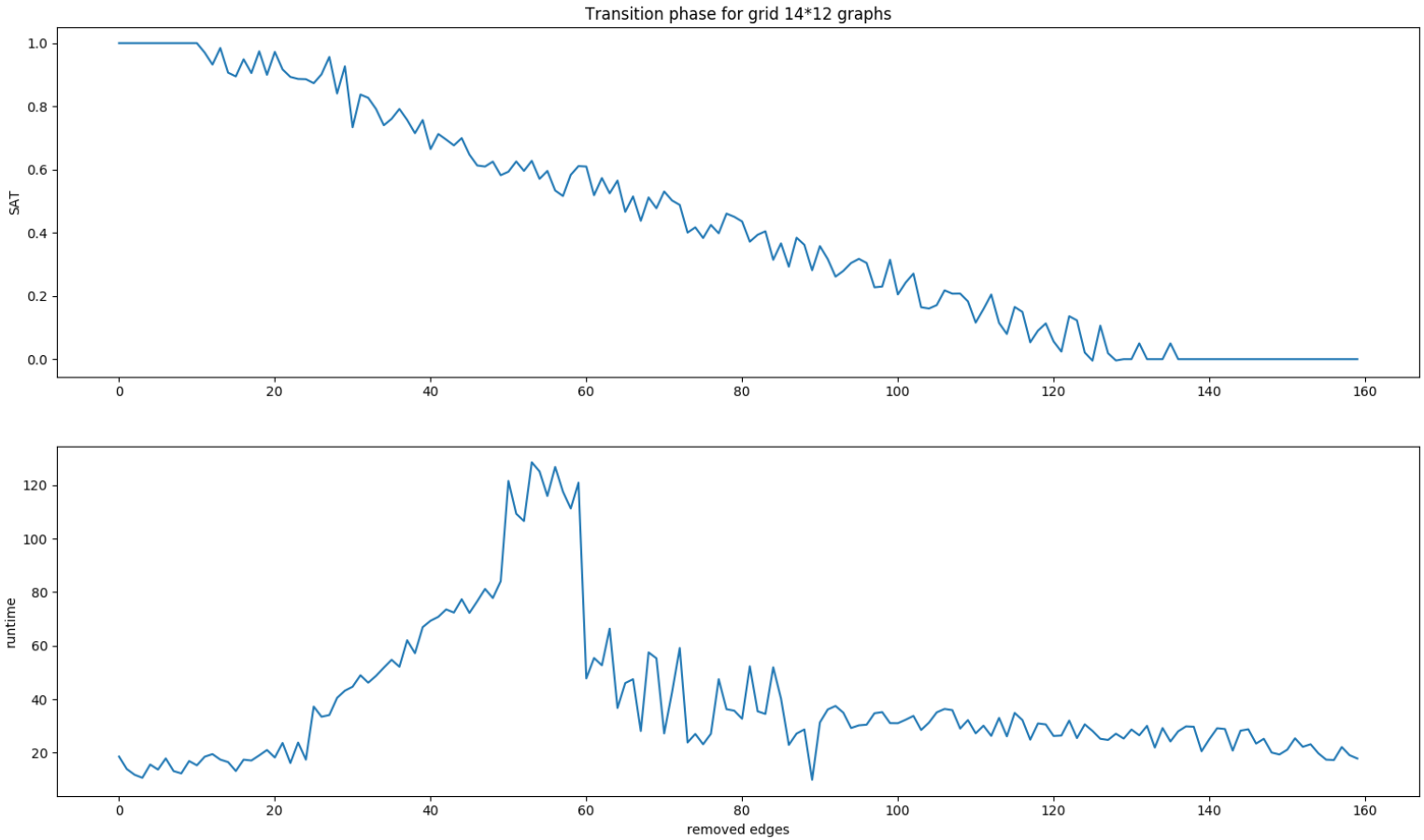
Instances

- Random structured graphs
  - Random graphs turn out quite easy
  - But random structured graphs are hard around the phase transition
- Start with the following structures where instances are always SAT.

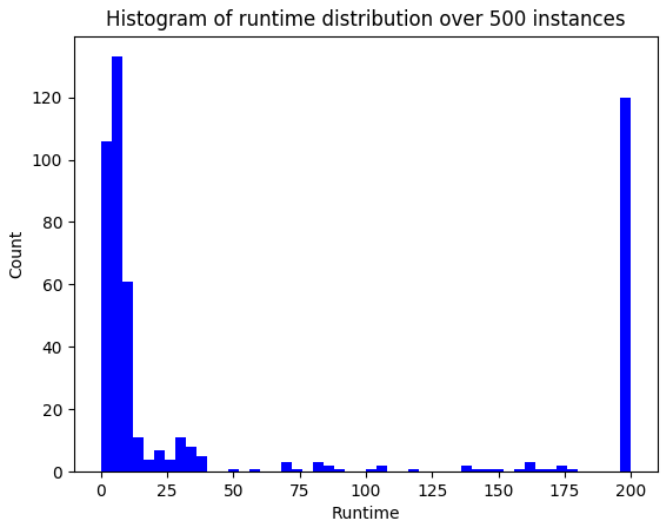
- Remove edges from the graphs (now some SAT, some UNSAT).
- Keep removing edges, until all instances are UNSAT.

Phase transition



Performance data collection and selection

- Collection
  - Performance data: *runtime*
  - *Cutoff time* (200s) is set to terminate an unsuccessful run.
- Selection
  - Combine performance data and find instances that are relatively hard (RT between 40 and 200s) for at least one encoding.
  - Valid instances are rare (<15%). Several thousand instances generated to select 784 reasonably hard ones.



Features

- Encoding-based claspre features
  - *Claspre*[2] is a special version of the solver *clasp*[3] for preprocessing.
  - It extracts static and dynamic features of ground ASP programs while solving them for a short amount of time. (88: Rules, Variables, CDCL solving process, etc.)
  - In order to obtain claspre features of a graph instance, we combine the instance with each of the six encodings and then pass them to *claspre*. (6*88 in total)

[2] https://potassco.org/labs/claspre/

[3] https://potassco.org

Features

- Problem-related graph features
  - Requires the understanding of the problem domain.
  - Related to graph structures (ratio_node_edge, avg_in_degree, avg_out_degree...)
  - Related to the properties of depth-first and breadth-first search trees of the graph as they inform about reachability from a node (dfs_1st_jumpback_depth, dfs_sum_of_choices_along_paths, max_depth_bfs,min_depth_bfs)
  - 42 problem-related graph features in total[4]

[4] https://cs.uky.edu/~lli259/encodingselection/features

Machine Learning Modeling

- The goal of encoding selection is to identify the best encoding, for a given instance
- Regression vs. Classification.
- Our work suggests that the regression approach works better in our problems.
- Regression models: k-nearest neighbor (KNN), decision tree (DT) and random forest (RF).
- Timeouts are replaced with varying penalized average runtime (PARX).
- Data splitting: 80% training data, 20% test data.
- 10-fold cross-validation is used within the 80% training data, and best model are selected based on average validation results in terms of the percentage of solved instances.
- Hyper-parameters tuning: grid search.
- Feature forward selection: In-group individual feature selection and group(13 groups) features selection, 41 features [5] selected from two groups.

[5] These 41 features can be found in our paper.

Experiments

- Four cores, Intel i7-7700 3.60GHz CPU, 16GB RAM, 64-bit 18.04.2 LTS (Bionic Beaver) Ubuntu system.
- The solver used is *clasp*[6] version 3.3.2 with default parameter setting.
- The grounding tool is *gringo*[7] version 4.5.4.

[6] https://potassco.org
[7] https://potassco.org

Test Result

Test result of encoding selection experiment

	Solved %	Avg Solved RT	Wins
Single encoding performance			
Encoding 1	84.0	82.4	25
Encoding 2	71.2	44.0	29
Encoding 3	56.4	30.7	20
Encoding 4	78.8	38.6	28
Encoding 5	57.1	35.4	26
Encoding 6	79.4	48.1	26
Oracle performance			
Oracle	98.7	21.1	
Encoding selection			
Encoding selection (KNN)	92.9	40.2	
Encoding selection (DT)	96.2	42.2	
Encoding selection (RF)	93.6	41.7	

Summary

- We generated several equivalent encodings for the HC problem and then build performance prediction models for them.
- We identified a set of features characterizing problem instances (some of them problem independent, and some reflecting features of the problem of interest).
- We tested the performance of different machine learning regression models and observed the performance gain over any individual encoding.
- We observed the encoding selection approach came very close to the always-select-best oracle in terms of solved instances (not runtime).

Future works

- Running time of our encoding selection approach is higher than the optimal time of the always-select-best oracle:
  - more informative domain-specific features
  - smarter feature selection methods
  - more sophisticated machine learning models
- Combine encoding selection with parameter tuning (ParamILS (Frank Hutter, 2009)).
- Automatically rewrite encodings into their equivalent forms instead of manually.

Thank you