Structure-based Instance Generation

Yuri Malitsky[2,3], Marius Merschformann[1], Barry O'Sullivan[2],
Kevin Tierney[1]

[1] kevin.tierney@upb.de – Assistant Professor
Decision Support & Operations Research
Department of Business Information Systems
University of Paderborn

[2] University College Cork, Ireland
[3] IBM Research, USA

September 29, 2014 – COSEAL Workshop 2014

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Outline

1. Motivation
2. Structures in SAT/Max-SAT Problems
3. The structure-based instance generation approach
4. Application to SAT/Max-SAT
5. Computational results
6. Conclusion

Motivation

Problem:

1. Training portfolios/configuring algorithms is difficult on small datasets.

Motivation

Problem:

1. Training portfolios/configuring algorithms is difficult on small datasets.
2. Industrial datasets tend to be very small

Motivation

Problem:

1. Training portfolios/configuring algorithms is difficult on small datasets.
2. Industrial datasets tend to be very small
3. Thus, often not enough data exists to build models/tune on specific industrial problems

Motivation

Problem:

1. Training portfolios/configuring algorithms is difficult on small datasets.
2. Industrial datasets tend to be very small
3. Thus, often not enough data exists to build models/tune on specific industrial problems

Motivation

Problem:

1. Training portfolios/configuring algorithms is difficult on small datasets.
2. Industrial datasets tend to be very small
3. Thus, often not enough data exists to build models/tune on specific industrial problems

In OR, this is a particular problem: many problems only have a limited number of real-world instances available (between 5 and 50 instances)

Motivation

Problem:

1. Training portfolios/configuring algorithms is difficult on small datasets.
2. Industrial datasets tend to be very small
3. Thus, often not enough data exists to build models/tune on specific industrial problems

In OR, this is a particular problem: many problems only have a limited number of real-world instances available (between 5 and 50 instances)

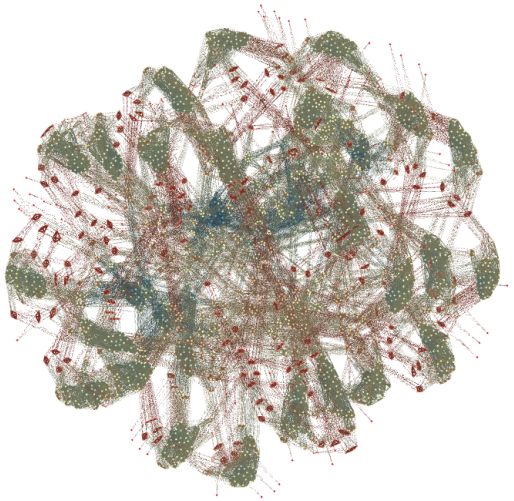
Solution: Generate new instances based on the instances we already have!

Novelty of this work

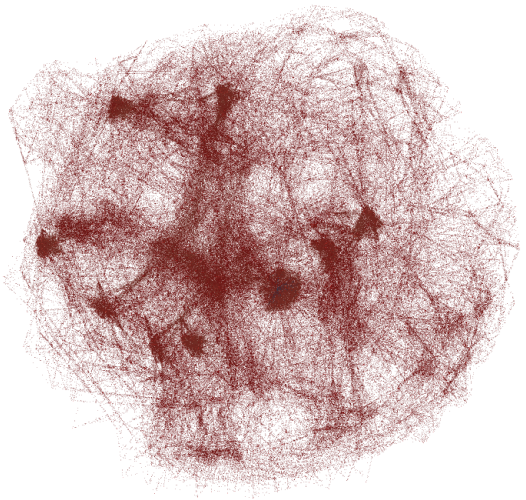
The first instance generation paradigm that...

- Analyzes the structure of existing instances based on their features and transports structures between instances
- Uses instance features as a criteria for accepting/rejecting an instance
- Shows that instances generated are effective within a selection approach

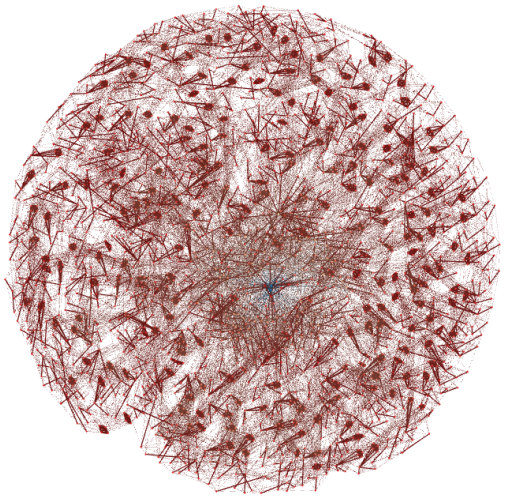
Variable graph visualizations



aes_32_3_keyfind_1

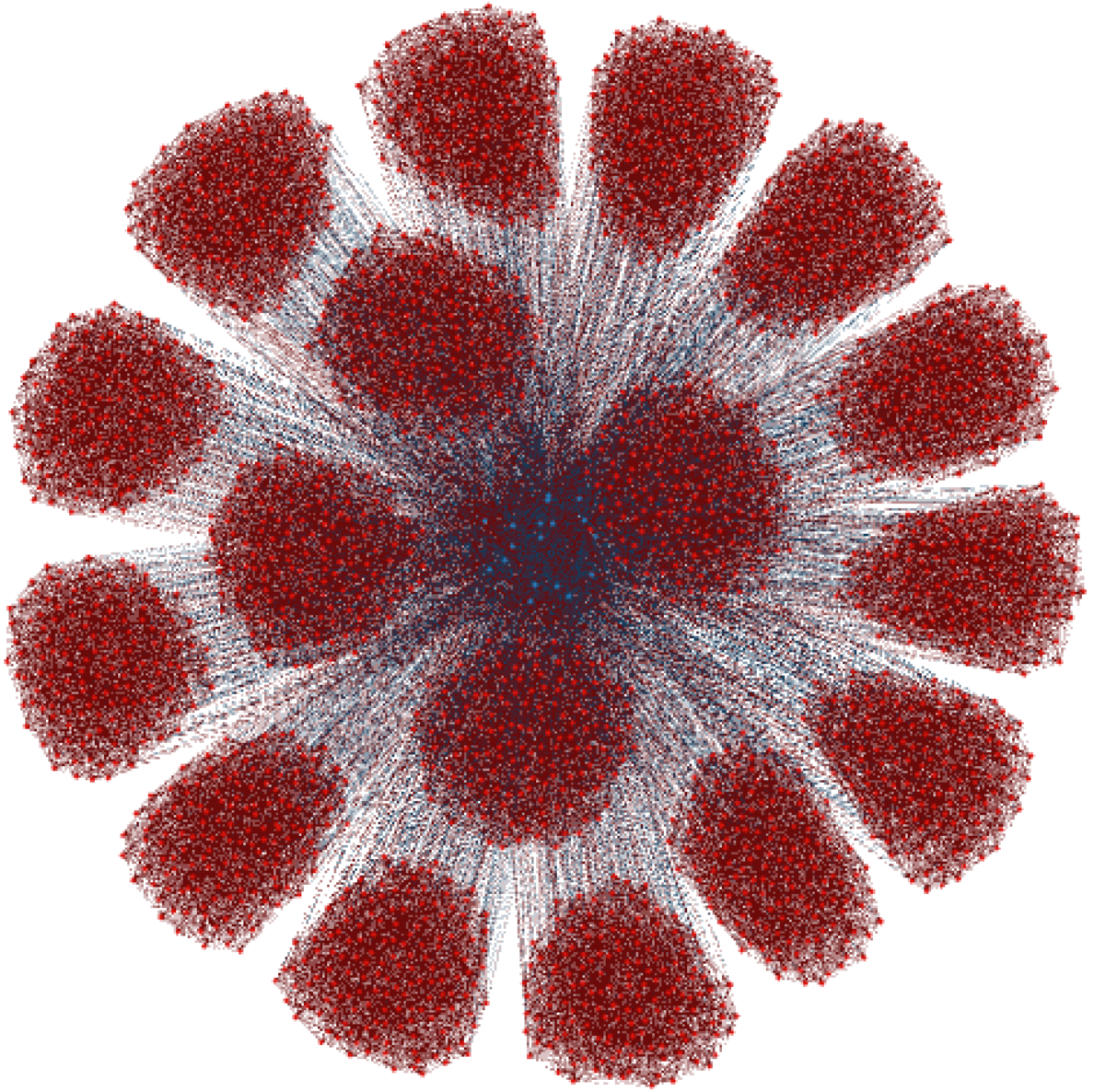


AProVE07-03



eq.atree.braun.8.unsat

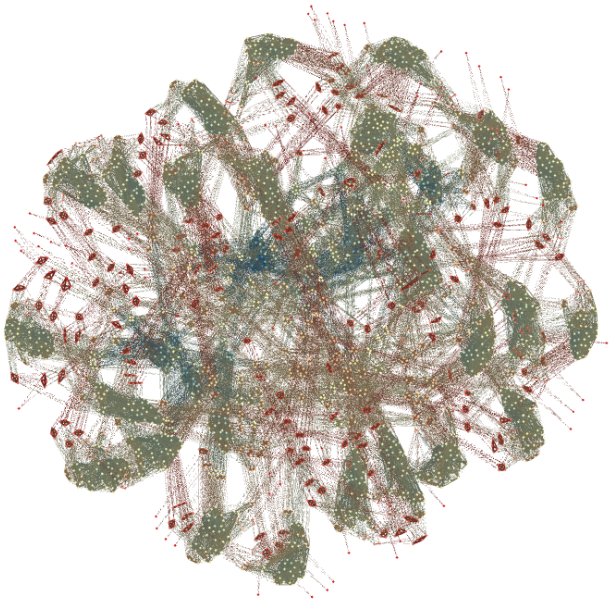
The "SAT Flower": counting-harder-php- reshuffled-07



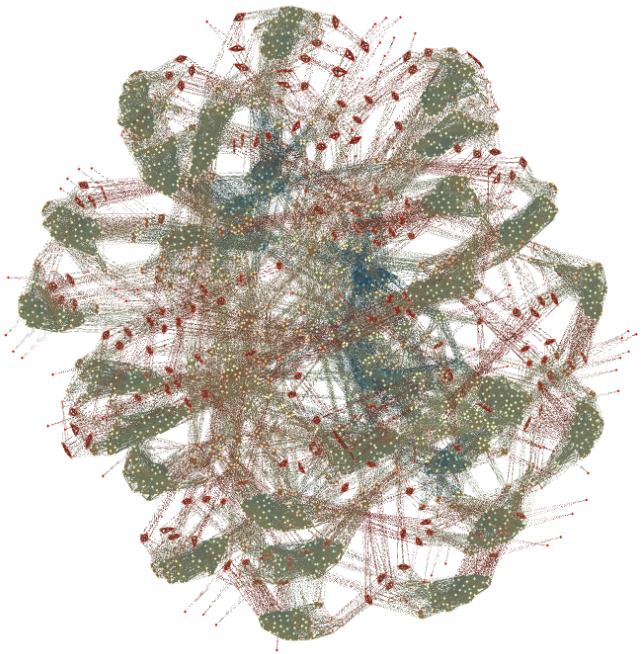
The Structure-based Instance Generation Approach

Instance generation example

Receiver

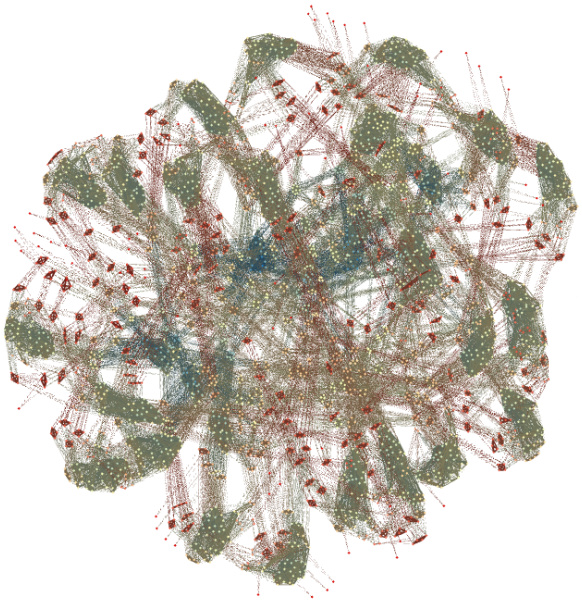


Giver

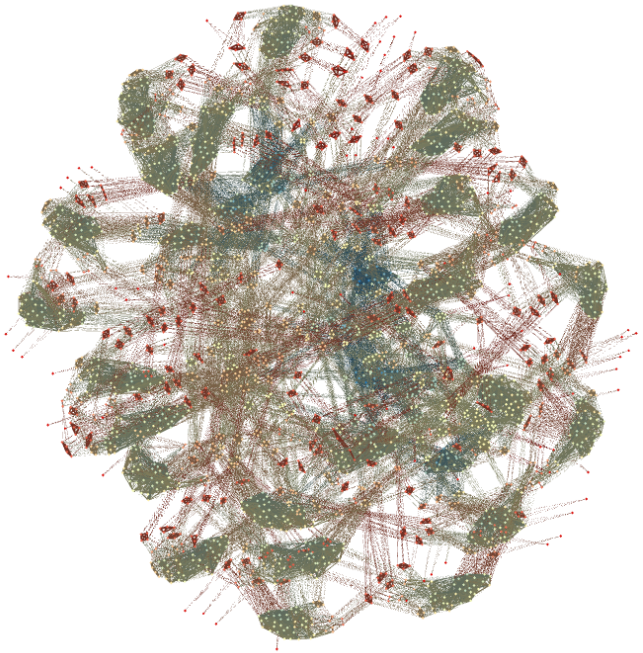


Instance generation example

Receiver (1 Destroy)

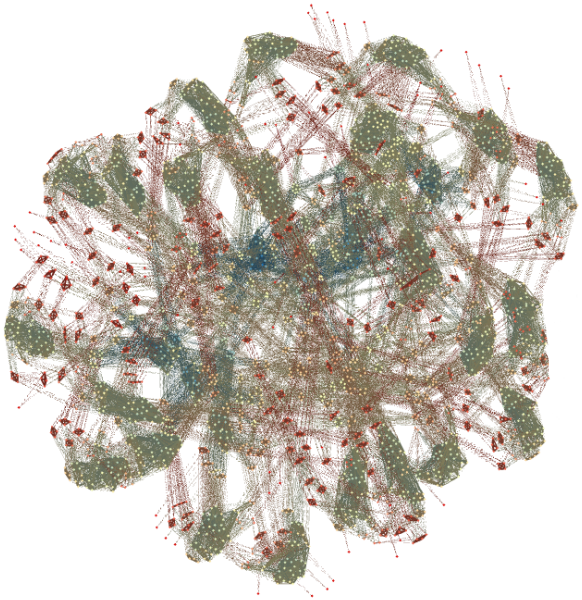


Giver

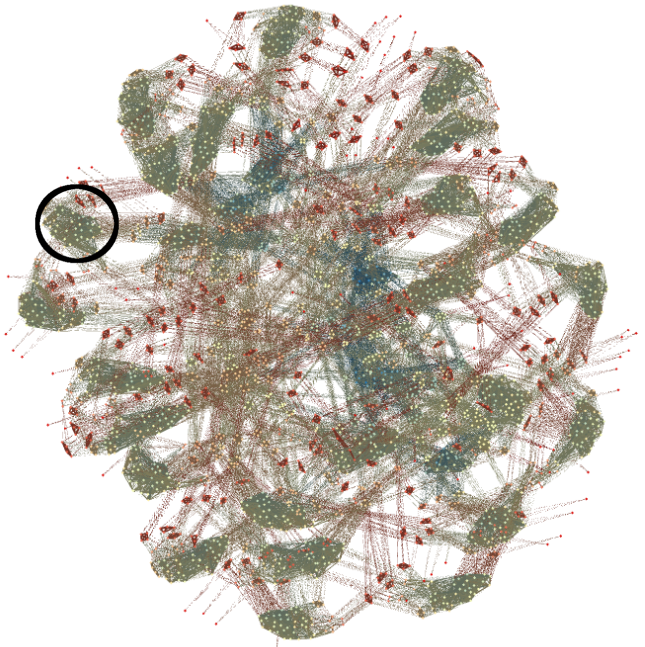


Instance generation example

Receiver (1 Destroy)

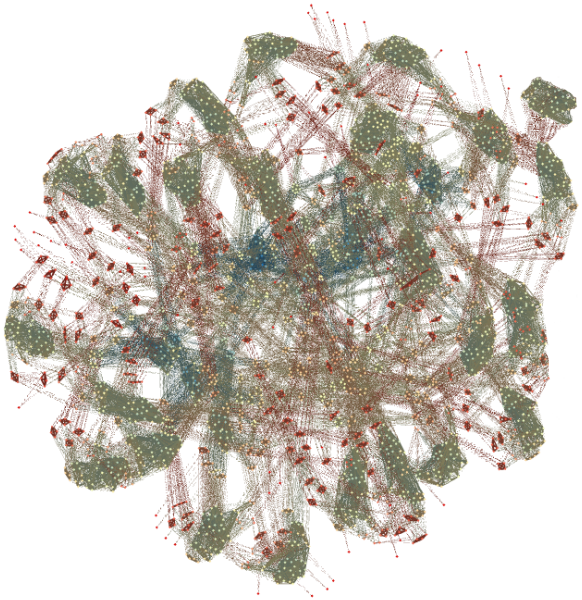


Giver (Identify structure)

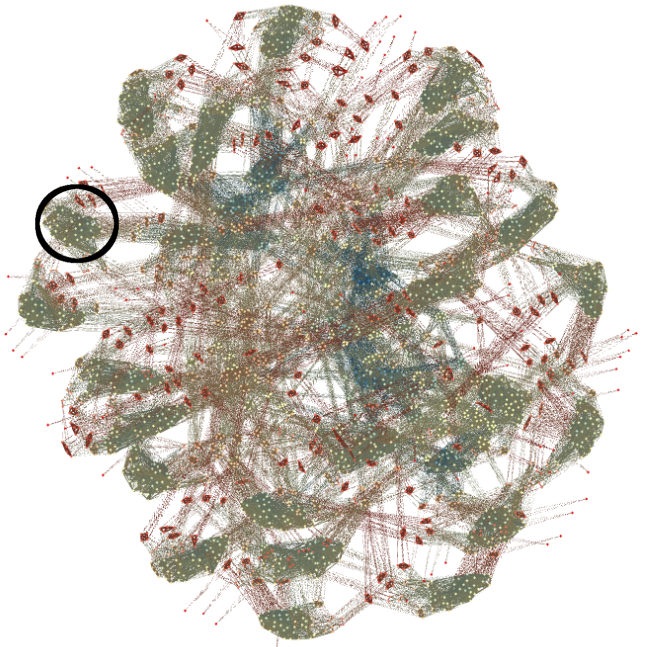


Instance generation example

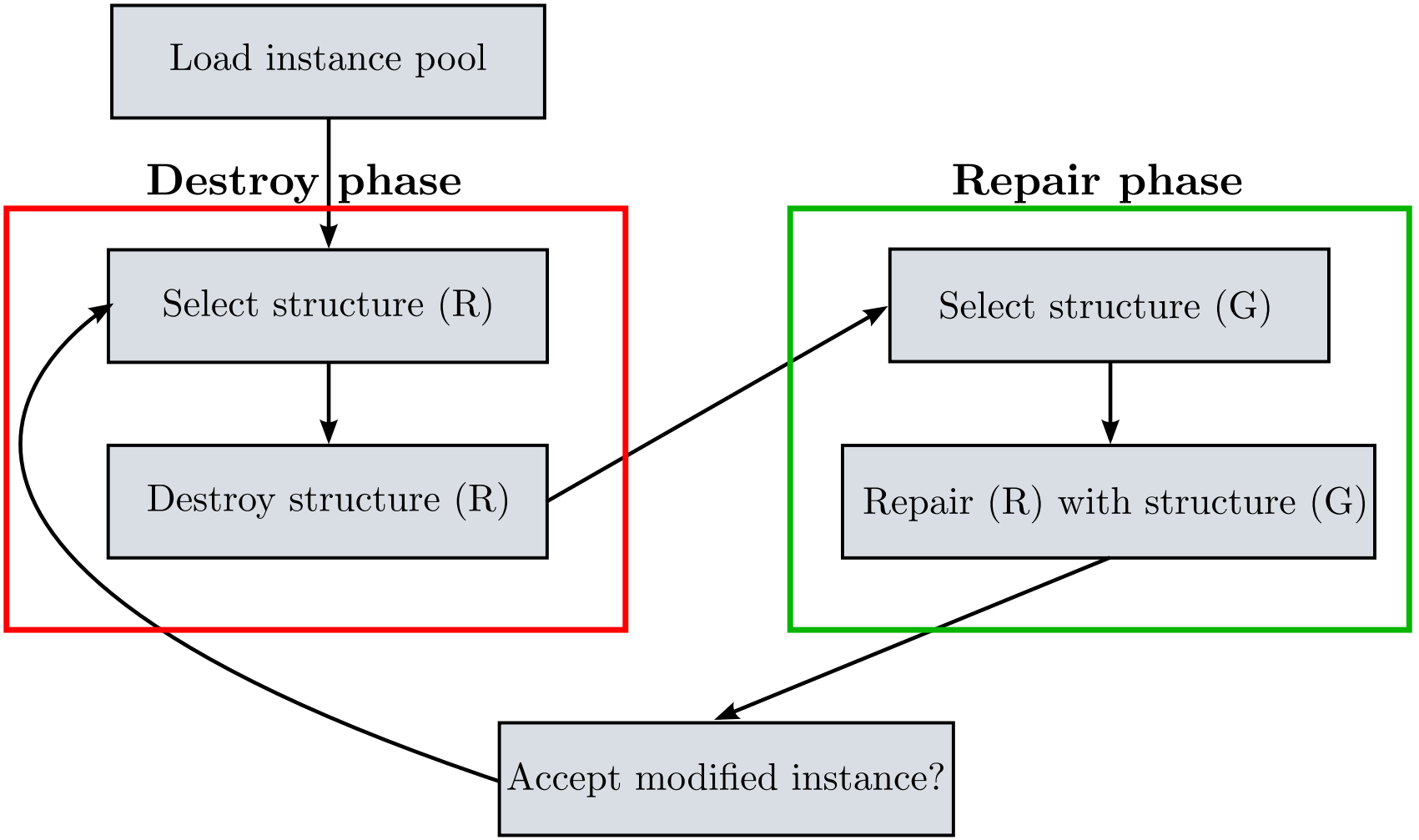
Receiver (Repair with structure)



Giver (Identify structure)



Instance generation algorithm



Reliance on instance features

Structure detection

- Structures should be connected to the instance features

Reliance on instance features

Structure detection

- Structures should be connected to the instance features
- e.g. in SAT: “modules” of variables (variable graph connectivity)

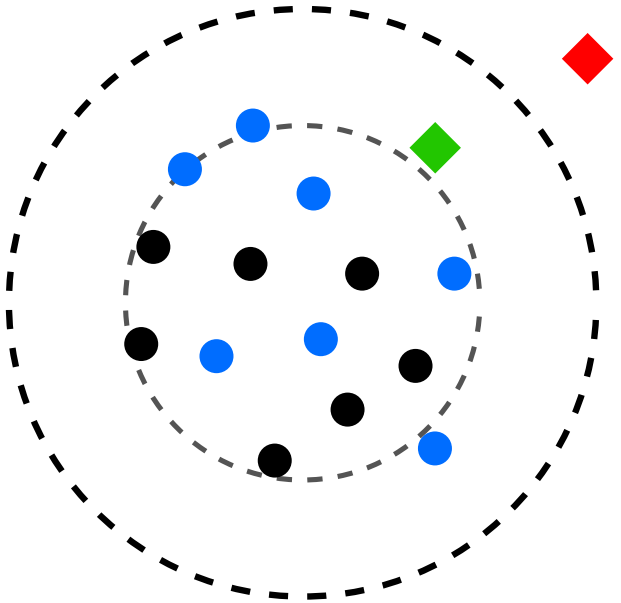
Reliance on instance features

Structure detection

- Structures should be connected to the instance features
- e.g. in SAT: “modules” of variables (variable graph connectivity)

Instance acception

- Ensure the distance of the new instance is not “too far” away from the pool of instances it was created from.



Related work

Numerous SAT/Max-SAT Instance generators:

- Too many to list them all; most generate instances based on probability distributions fine tuned to try to make the instances more industrial-like.

Main related work:

Morphing (Gent, Hoos, Prosser, Walsh. AAAI 1999):

- Goal is to "connect" the structures of two instances; it is a much more invasive process
- Through our destroy/repair process, we modify the underlying instance differently

Why SAT/Max-SAT

- Simple instance structure for testing the generation technique

Why SAT/Max-SAT

- Simple instance structure for testing the generation technique
- Dearth of real-world data, especially for Max-SAT

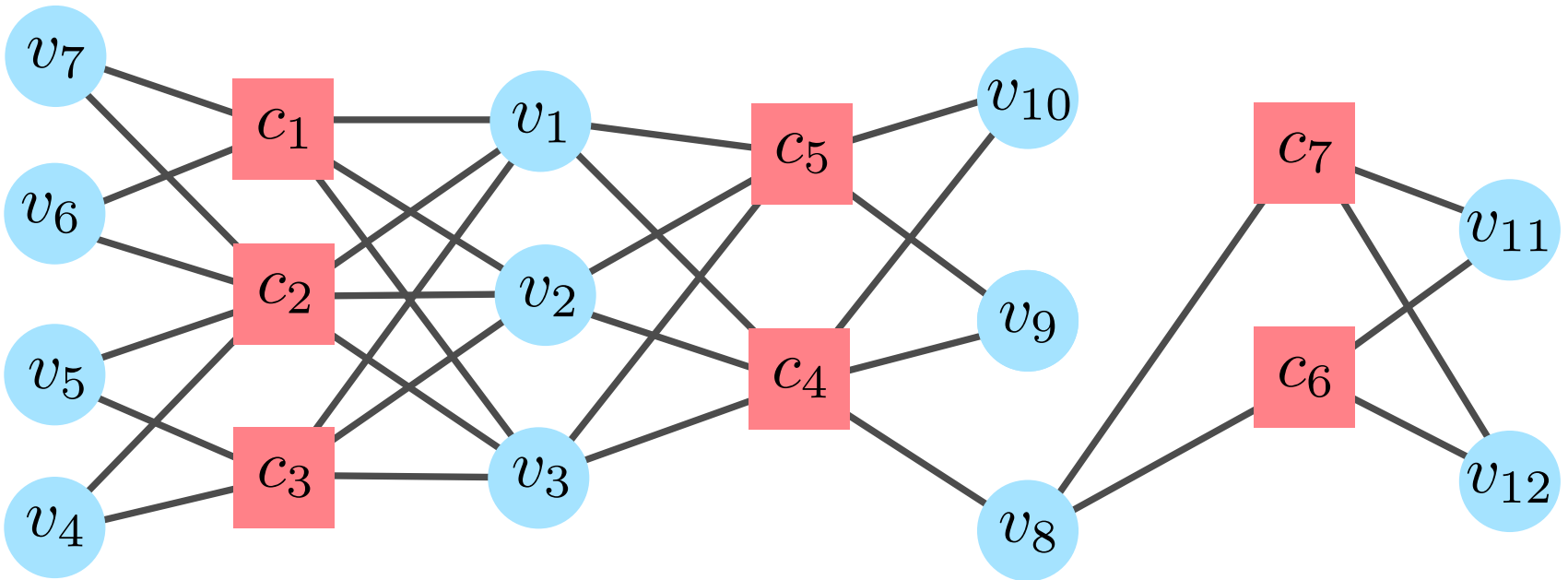
Why SAT/Max-SAT

- Simple instance structure for testing the generation technique
- Dearth of real-world data, especially for Max-SAT
- Well-known and studied instance features

Application to SAT/Max-SAT

Variable based heuristic

- Identifies variables in sub-components of the instance
- Uses the average number of clauses each variable is in and selects a variable with a number of clauses near this value

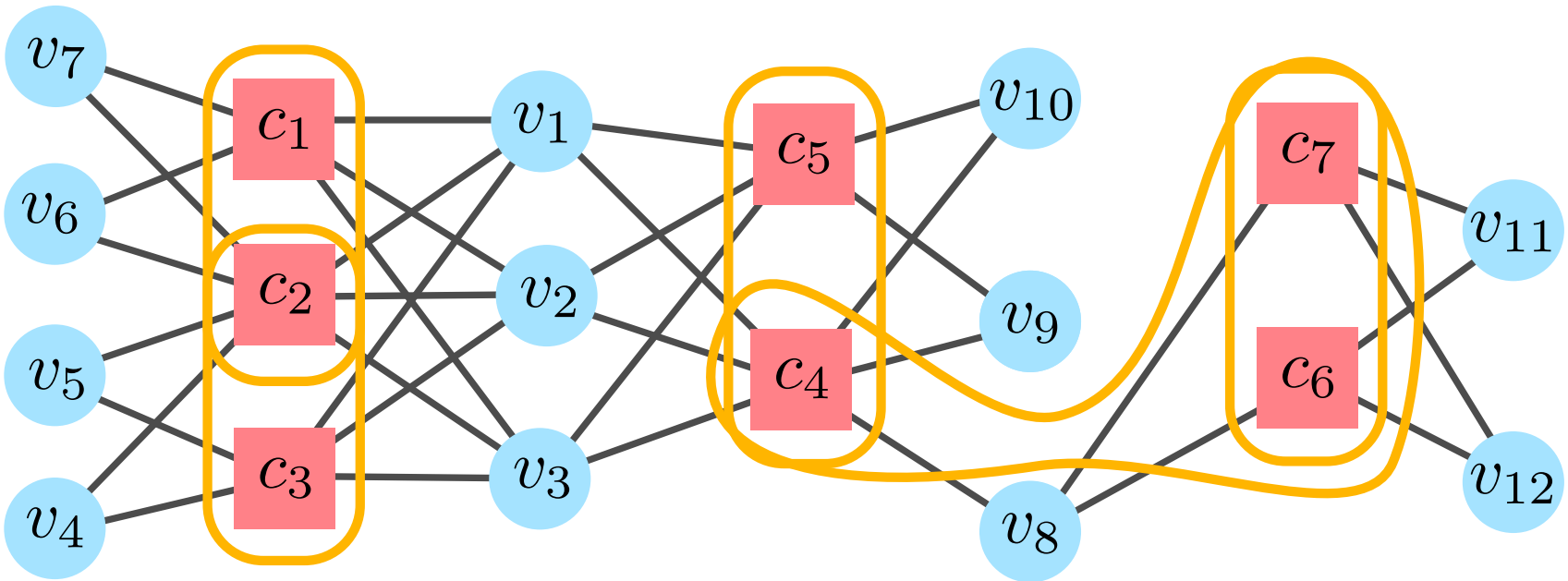


Average clauses/variable: 2.8;

Application to SAT/Max-SAT

Variable based heuristic

- Identifies variables in sub-components of the instance
- Uses the average number of clauses each variable is in and selects a variable with a number of clauses near this value

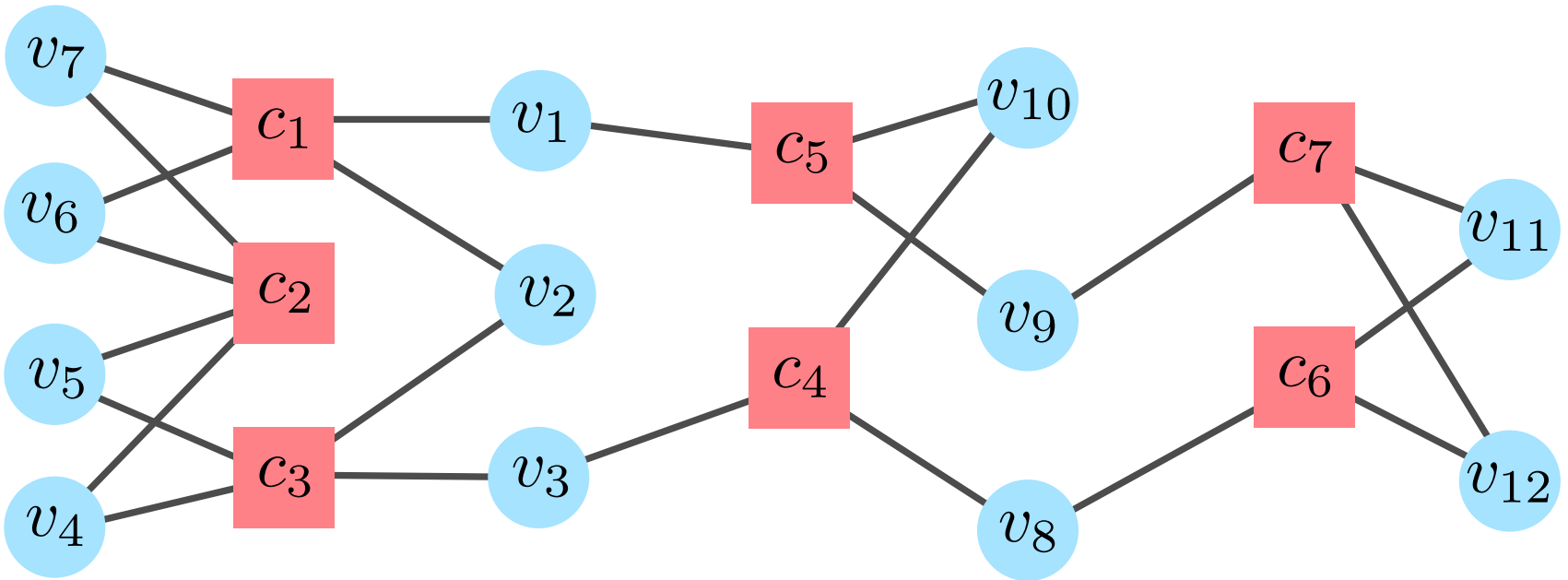


Average clauses/variable: 2.8; Identified sub-components

Application to SAT/Max-SAT

Clause based heuristic

- Identifies clauses in sub-components of the instance
- Uses the average number of variables shared by each clause and selects a clause with a number of variables near this value and returns it and its neighbors

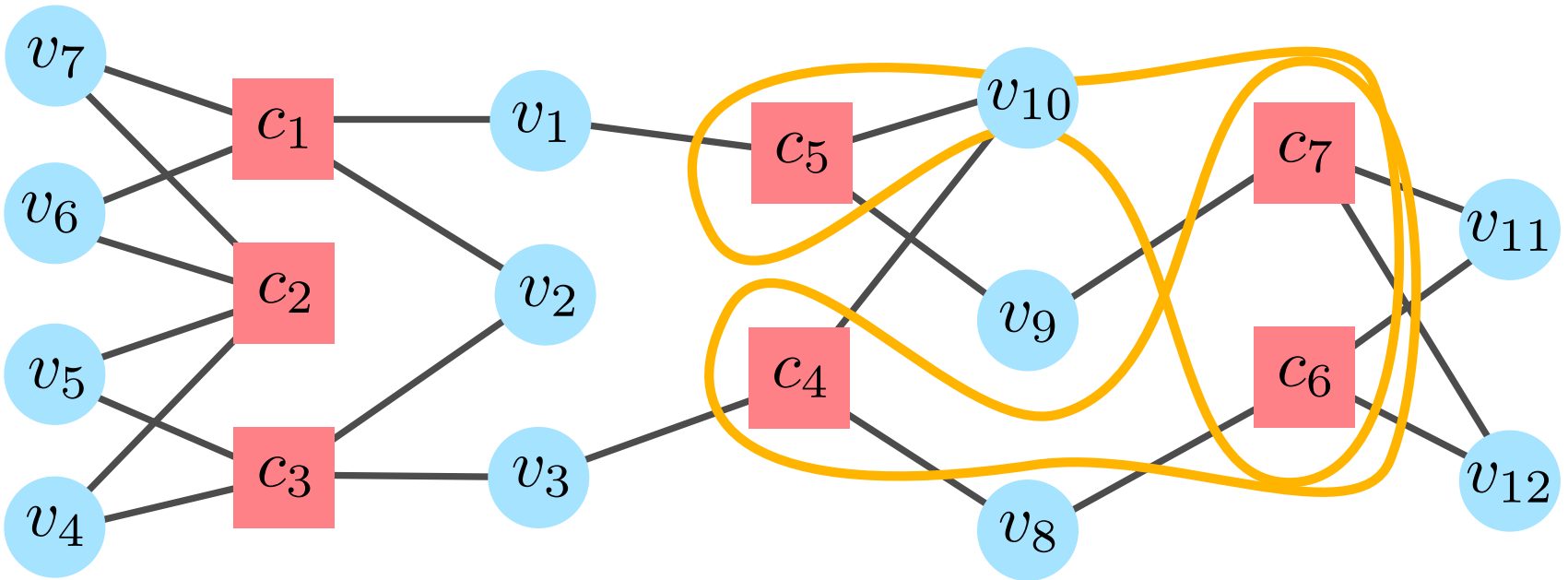


Average variable/clause: 3.4;

Application to SAT/Max-SAT

Clause based heuristic

- Identifies clauses in sub-components of the instance
- Uses the average number of variables shared by each clause and selects a clause with a number of variables near this value and returns it and its neighbors

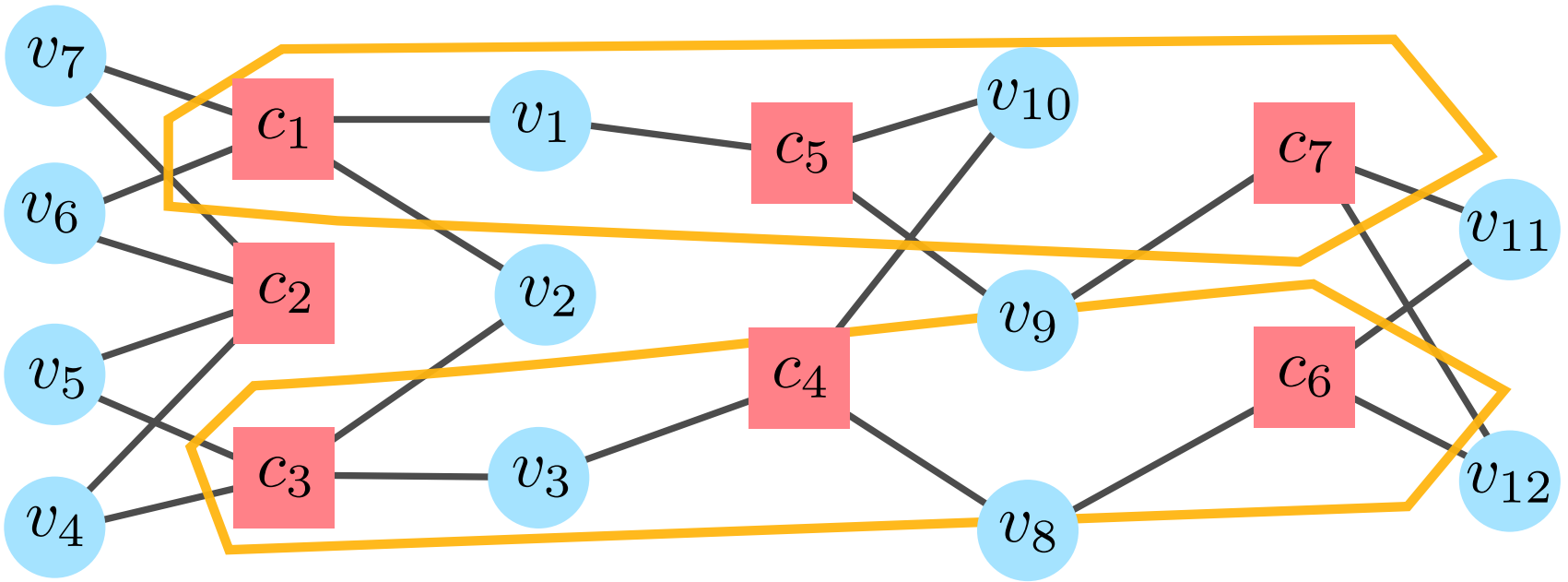


Average variable/clause: 3.4; Identified sub-components

Application to SAT/Max-SAT

Clause based heuristic

- Identifies clauses in sub-components of the instance
- Uses the average number of variables shared by each clause and selects a clause with a number of variables near this value and returns it and its neighbors



Average variable/clause: 3.4; Identified sub-components

Application to SAT/Max-SAT

Destroy

- Destroying is simple:
  1. Remove all of the selected clauses
  2. Perform bookkeeping

Application to SAT/Max-SAT

Repair

Goal: Map variables in sub-component to variables in the instance

Application to SAT/Max-SAT

Repair

Goal: Map variables in sub-component to variables in the instance

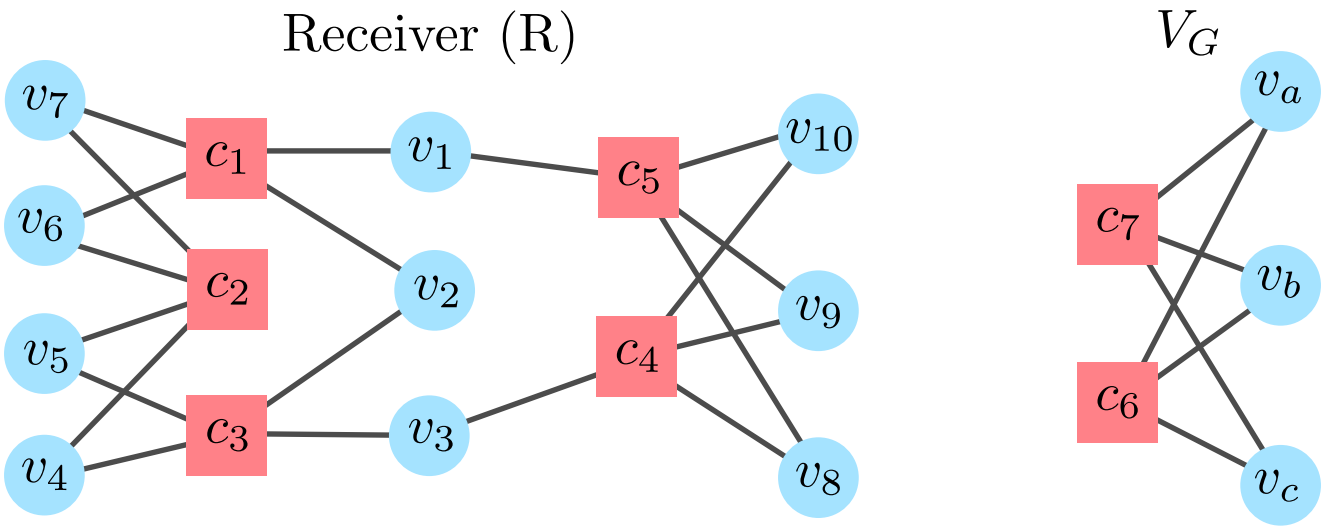
- Compute three features for each variable in the receiver and new component:
  1. # clauses / # instance clauses
  2. # variable positive clauses / # total variable clauses
  3. Average # variables for each clause the variable is in

Application to SAT/Max-SAT

Repair

Goal: Map variables in sub-component to variables in the instance

- Compute three features for each variable in the receiver and new component:
  1. # clauses / # instance clauses
  2. # variable positive clauses / # total variable clauses
  3. Average # variables for each clause the variable is in



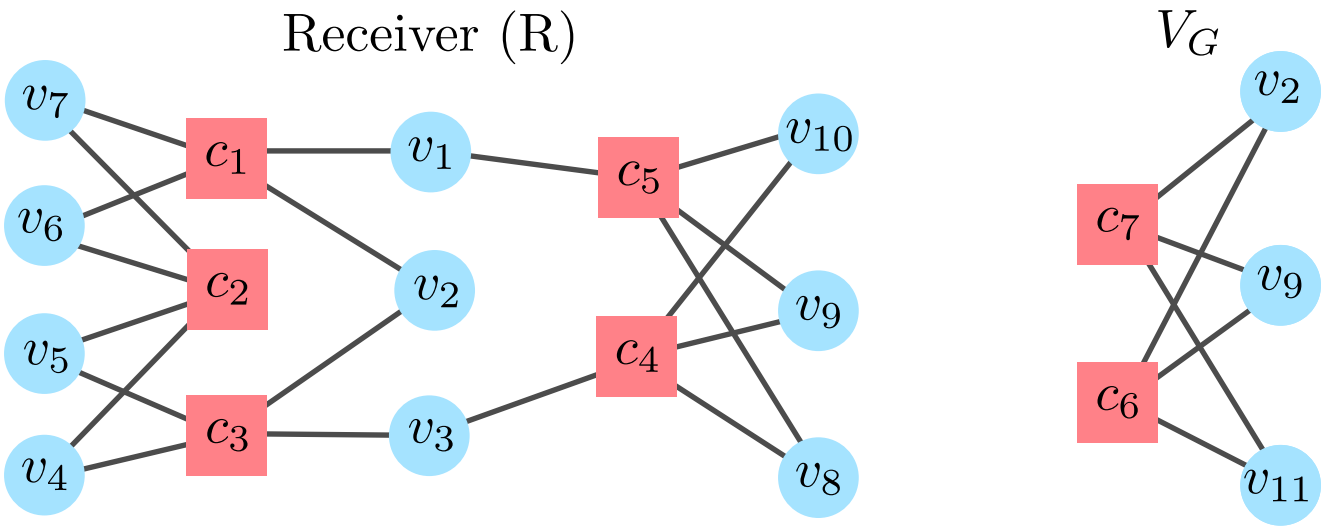
Then, map variables in V_G to variables with similar features, or with small probability, make it a new variable.

Application to SAT/Max-SAT

Repair

Goal: Map variables in sub-component to variables in the instance

- Compute three features for each variable in the receiver and new component:
  1. # clauses / # instance clauses
  2. # variable positive clauses / # total variable clauses
  3. Average # variables for each clause the variable is in



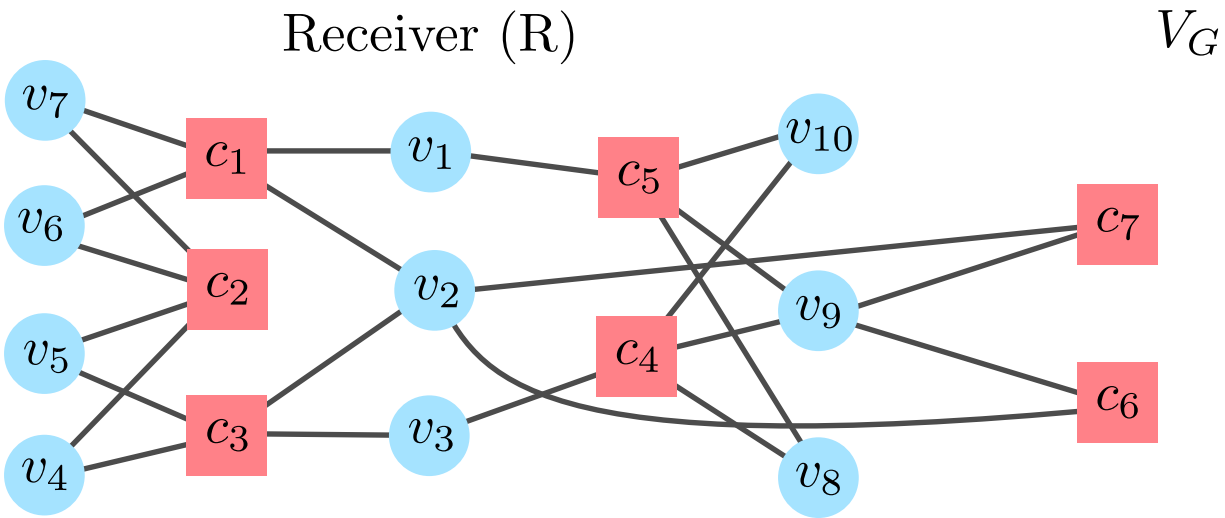
Then, map variables in V_G to variables with similar features, or with small probability, make it a new variable.

Application to SAT/Max-SAT

Repair

Goal: Map variables in sub-component to variables in the instance

- Compute three features for each variable in the receiver and new component:
  1. # clauses / # instance clauses
  2. # variable positive clauses / # total variable clauses
  3. Average # variables for each clause the variable is in



Then, map variables in V_G to variables with similar features, or with small probability, make it a new variable.

Application to SAT/Max-SAT

Acceptance criterion

Max-SAT & SAT

- Compute instance features and check if the distance to the cluster center is within 3 standard deviations of the mean for each feature

Application to SAT/Max-SAT

Acceptance criterion

Max-SAT & SAT

- Compute instance features and check if the distance to the cluster center is within 3 standard deviations of the mean for each feature

SAT

- Potential problem with SAT: Small changes to the instance can make it trivial to solve

Application to SAT/Max-SAT

Acceptance criterion

Max-SAT & SAT

- Compute instance features and check if the distance to the cluster center is within 3 standard deviations of the mean for each feature

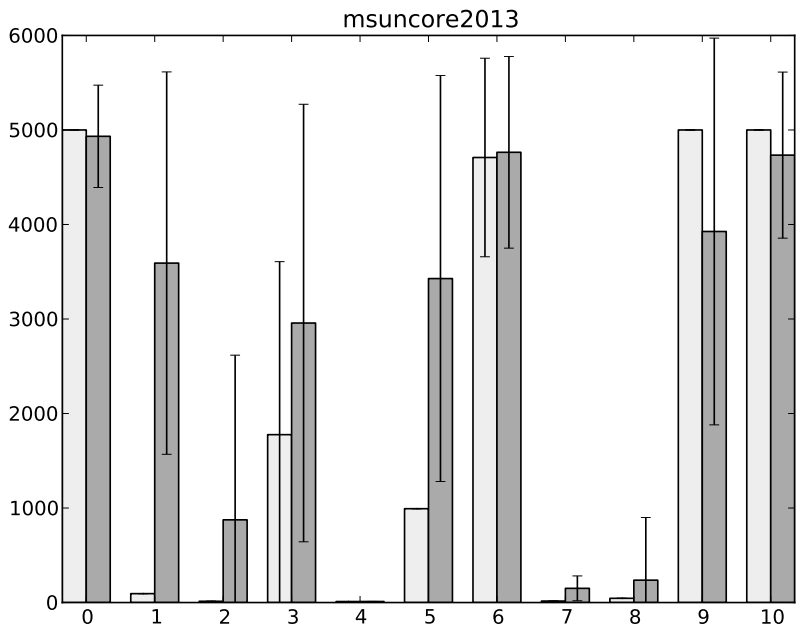
SAT

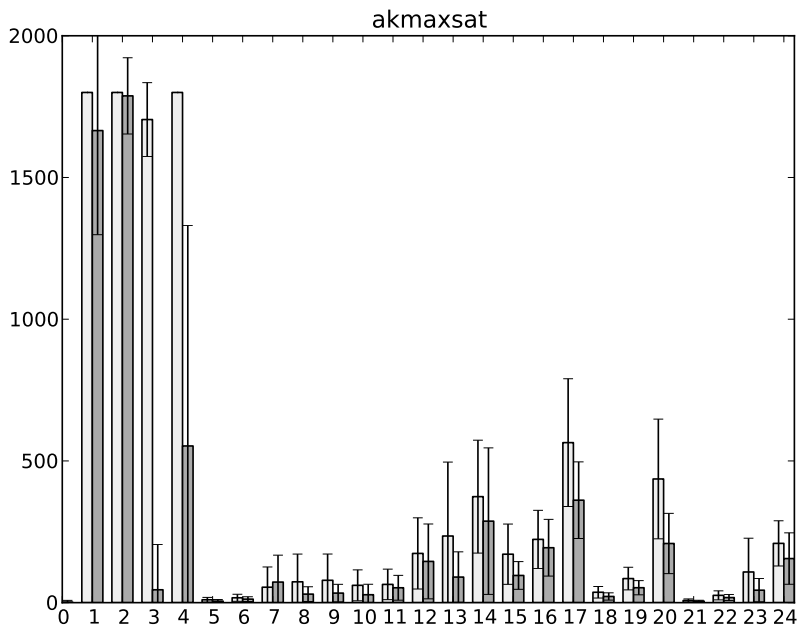
- Potential problem with SAT: Small changes to the instance can make it trivial to solve
- Solution: Run a SAT solver for 30 seconds; if the instance is solved, reject it.

Computational results

Max-SAT

We generated Max-SAT instances using the industrial instances in the Max-SAT 2013 competition.





Above: Average runtime (s) for an industrial solver and random solver on clusters of generated instances from the industrial and random instances, respectively.

Computational results

Max-SAT (2)

Model	Original		Generated	
	Avg. (s)	# Unsolved	Avg. (s)	# Unsolved
Best Single	735	2	735	2
Random Forest	988	5	599	2
SVM (radial)	734	2	591	1
VBS	184	0	184	0

Several selection techniques training a portfolio of industrial Max-SAT solvers (10-fold cross validation).

Computational results

SAT

	Average	PAR10	Solved
Best Single	453	3,872	157
Random (300)	541	5,107	144
Crafted (300)	386	4,090	154
Industrial (46)	348	3,426	161
Generated (300)	502	3,463	162
Generated (1,500)	437	3,049	166
VBS	364	364	195

Conclusion and future work

- We generate industrial-like instances from a pool of existing instances
- Currently, this is possible for the SAT and Max-SAT problems
- We intend to expand this work to mixed-integer programming and possibly other problems