Unsatisfiability-based optimization in clasp

Benjamin Andres Benjamin Kaufmann Oliver Mattheis
Torsten Schaub

University of Potsdam

Outline

Motivation

- Problem: Package configuration for Linux systems raises difficult optimization problems
- Goal: Explore the applicability of MaxSAT solving techniques for ASP optimization
- Approach: Expand the highly sophisticated MSUnCore Algorithm (Marques-Silva) for ASP usage
- Result: The *unclasp* solver won four out of seven tracks at the 2011 MISC competition

Basic Idea : MSUnCore Algorithm

- try to solve the problem
- if the problem is solvable, the solution is an optimal solution
- else
  - identify an unsatisfiable core
  - if the core is empty, the problem is unsolvable
  - else
    - relax all soft clauses by adding new unique variables
    - add an *one-hot* constraint to the problem
    - try to solve the modified problem

Pseudo Code : MSUnCore Algorithm

Algorithm 1 Iterative UNSAT Core Elimination of msu1

```
T := ∅ {set of all relaxation variables}
while SAT solver returns UNSATISFIABLE do
    let UC be the UNSAT core provided by the SAT solver
    S := ∅ {set of new relaxation variables for UC}
    for all Clause c ∈ UC do
        if c is relaxable then
            Allocate a new relaxation variable v
            c := c ∪ {v}
            S := S ∪ {v}
        end if
    end for
    if S = ∅ then
        return CNF UNSATISFIABLE
    else
        Add clauses enforcing the one-hot constraint for S to the SAT solver
        T := T ∪ S
    end if
end while
R := {v|v ∈ T, v = 1}; k := |R|
return Satisfying Assignment, k, R.
```

MaxSAT in ASP

- ASP optimization and MaxSAT are strongly related
- both consist of a set of hard rules and a linear optimization function
- literals in an ASP optimization rule can be interpreted as soft clauses in MaxSAT
- for ASP, the soft clauses in MaxSAT are relaxed and the usage of relaxation variables is minimized by an optimization rule

Implementation in Clasp

- consider all literals in the minimization rule as false
- try to solve the problem
- if the problem is solvable, the solution is an optimal solution
- else
  - identify an unsatisfiable core
  - if the core is empty, the problem is unsolvable
  - else
    - all literals in the core are removed from the minimization rule
    - a soft *at-most-1* rule over all literals in the core is added
    - existing *at-most-n* rules in the core become *at-most-n+1* rules
    - try to solve the modified problem

Pseudo Code : Implementation in Clasp

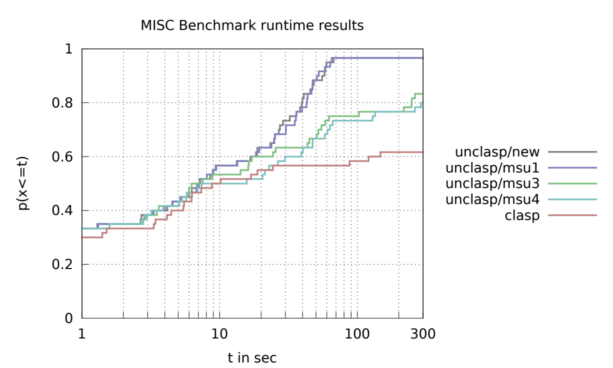
Algorithm 2 Unsatisfiable Core optimization for ASP

```
assumptions := {l | ¬l ∈ minimizationrule}
guardmap := ∅
while ASP solver returns UNSATISFIABLE under assumptions do
    let conflict be the subset of assumptions involved in the root conflict
    if conflict = ∅ then
        return UNSATISFIABLE
    else
        Allocate a new guard variable guard
        Add constraint guard :- 2{l | ¬l ∈ conflict}.
        assumptions := (assumptions \ conflict) ∪ {¬guard}
        guardmap := guardmap ∪ {(guard, conflict, 2)}
        for all Literal literal ∈ conflict do
            if (¬literal, conflict1, bound) ∈ guardmap ∧ bound < |conflict1| then
                Allocate a new guard variable guard1
                Add constraint guard1 :- (bound + 1){l | ¬l ∈ conflict1}.
                assumptions := assumptions ∪ {¬guard1}
                guardmap := guardmap ∪ {(guard1, conflict1, bound + 1)}
            end if
        end for
    end if
end while
return Satisfying Assignment.
```

Experimental Results

- highly efficient on certain subsets of MISC instances
- mixed results with optimization problems from the asparagus benchmark collection
  - comparable results with unweighted problems with small cores (15-puzzle, sokoban and clique)
  - superior results with problems with a large number of optimization variables and large cores (labyrinthpath, minimum postage stamp problem (mpsp), weight bounded dominating set instances (wbds))
  - does not scale as well as clasp's approach on weighted problems (companyctrl, opendoors, fastfood)

MISC : Experimental Results



ASP (unweighted) : Experimental Results

	new	msu1	msu3	msu4	clasp
15-puzzle					
init1	9,58	9,42	9,53	9,45	6,74
init1simple	0,93	0,93	0,93	0,94	2,02
init1simple2	0,42	0,42	0,42	0,42	1,48
init2	81,84	82,74	82,27	81,81	73,85
init3	13,38	13,28	13,44	13,43	16,81
sokoban					
dimitr_yo51s10	0,24	0,24	0,25	0,25	0,21
dimitr_yo51s14	1,65	1,65	1,66	1,66	1,06
dimitr_yo51s17	2,54	2,55	2,55	2,58	1,59
dimitr_yo52s10	1,67	1,67	1,67	1,67	0,65
dimitr_yo55s10	0,81	0,81	0,81	0,83	0,49
clique					
gen10_25	0,02	0,02	0,02	0,02	0,02
gen200_8000	4,77	300,00	21,35	0,73	0,77
gen300_20000	**28,87**	**300,00**	**143,45**	**4,17**	**3,94**
gen75_1000	0,06	8,19	0,12	0,07	0,05
gen100_2000	0,14	300,00	0,51	0,16	0,08

ASP (unweighted) : Experimental Results

	new	msu1	msu3	msu4	clasp
labyrinthpath					
l10_10_01	1,88	1,89	1,88	1,86	18,33
l11_11_01	**2,41**	**1,88**	**2,39**	**2,37**	**300,00**
l12_12_01	**3,44**	**4,82**	**212,90**	**236,66**	**300,00**
l13_13_01	300,00	300,00	300,00	300,00	300,00
l14_14_01	300,00	300,00	300,00	300,00	300,00
mpsp					
mpsp30-2	0,04	0,13	0,08	0,09	0,05
mpsp36-2	0,12	3,35	0,44	1,02	0,30
mpsp48-2	**2,99**	**300,00**	**50,90**	**53,01**	**116,59**
mpsp54-2	**1,96**	**300,00**	**64,83**	**36,83**	**80,59**
mpsp60-2	**17,14**	**300,00**	**300,00**	**300,00**	**300,00**
wbds					
r100_400_11_1	**72,01**	**300,00**	**300,00**	**300,00**	**300,00**
r100_400_11_13	**3,40**	**300,00**	**300,00**	**300,00**	**300,00**
r100_400_11_9	**4,16**	**300,00**	**300,00**	**300,00**	**300,00**
r150_600_11_17	300,00	300,00	300,00	300,00	300,00
r150_600_11_3	300,00	300,00	300,00	300,00	300,00

ASP (weighted) : Experimental Results

	new	msu1	msu3	msu4	clasp
companyctrl					
02-company	0,85	0,88	285,46	0,93	0,74
12-company	3,96	4,02	4,52	4,08	3,91
22-company	1,46	1,52	300,00	1,52	1,13
32-company	0,96	0,94	300,00	0,92	0,85
42-company	1,54	1,51	1,76	1,71	1,52
opendoors					
level_00	0,09	0,10	0,09	0,11	0,11
level_05	0,19	0,26	0,19	0,22	0,20
level_10	0,37	0,37	0,37	0,62	0,28
level_17	24,73	6,76	24,17	71,98	4,92
level_28	**300,00**	**300,00**	**300,00**	**300,00**	**18,64**
fastfood					
a5.16.dl	300,00	300,00	32,36	12,15	13,02
a5.4.dl	300,00	300,00	300,00	3,84	1,15
fa8.17.dl	300,00	300,00	24,42	8,72	5,92
a8.8.dl	**300,00**	**300,00**	**300,00**	**276,82**	**300,00**
a9.11.dl	300,00	300,00	28,94	7,29	6,15

Summary

- introducing a top-down optimization strategy specialized for unweighted, hierarchic optimization in ASP
- able to solve a number of problems faster than traditional branch-and-bound strategies
- a useful addition to clasp's portfolio of optimization strategies

- Sourceforge
  `http://potassco.sourceforge.net/labs.html#unclasp`
- Google+
  `https://plus.google.com/102537396696345299260`