Weighted-Sequence Problem

ASP vs CASP and Declarative vs Problem Oriented Solving

Yuliya Lierler, Shaden Smith, Miroslaw Truszczynski, Alex Westlund
University of Kentucky

14th International Symposium on
Practical Aspects of Declarative Languages
Philadelphia, January 23-24, 2012

January 23, 2012

Introduction I

Our goals:

- to evaluate ASP vs CASP solving technology
- to evaluate modeling approaches in ASP and CASP

Introduction II

The problem studied:

- the weighted-sequence (WS) problem
- inspired by the problem of finding an optimal join order in the cost-based query optimizer of ORACLE
- the issue: variables with large domains

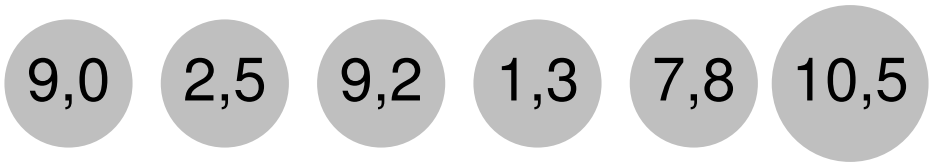
Experiments involved instances that are

- easy and difficult to ground
- with loose and tight constraints

Our ultimate goal is to use ASP and CASP technology to assist in finding an optimal join order.

Problem Statement I

- Given: a set of nodes and an integer m — *maximum cost*.
- Each node is a pair (*weight,cardinality*).

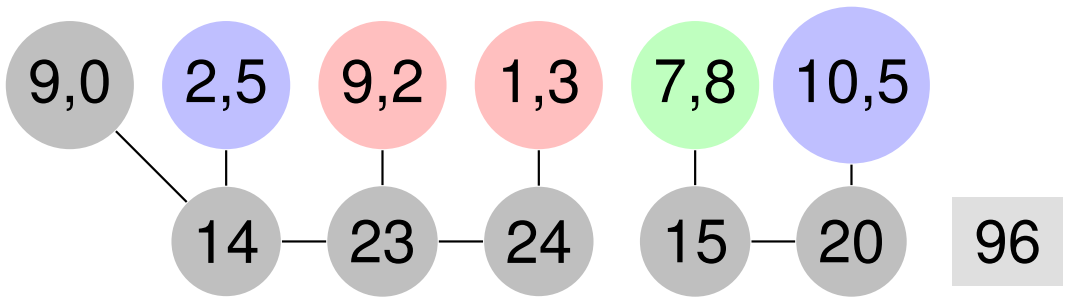


Problem Statement II

- The objective is to find a permutation and color assignment with a *total cost* at most m.
- The cost of every colored node n_i is defined by

$$cost(n_i) = \begin{cases} weight(n_i) + card(n_i) & \text{if } n_i \text{ is green} \\ cost(n_{i-1}) + weight(n_i) & \text{if } n_i \text{ is red} \\ cost(n_{i-1}) + card(n_i) & \text{if } n_i \text{ is blue} \end{cases}$$

- The *total cost* of a sequence is calculated by summing the costs of each colored node.



ASP Introduction

- ASP is a declarative programming paradigm intended to solve difficult (NP-hard) combinatorial search problems with a growing list of applications.

ASP Introduction II

Rules

$$a_0 \leftarrow a_1, \ldots, a_m, \mathit{not}\ a_{m+1}, \ldots, \mathit{not}\ a_n,$$

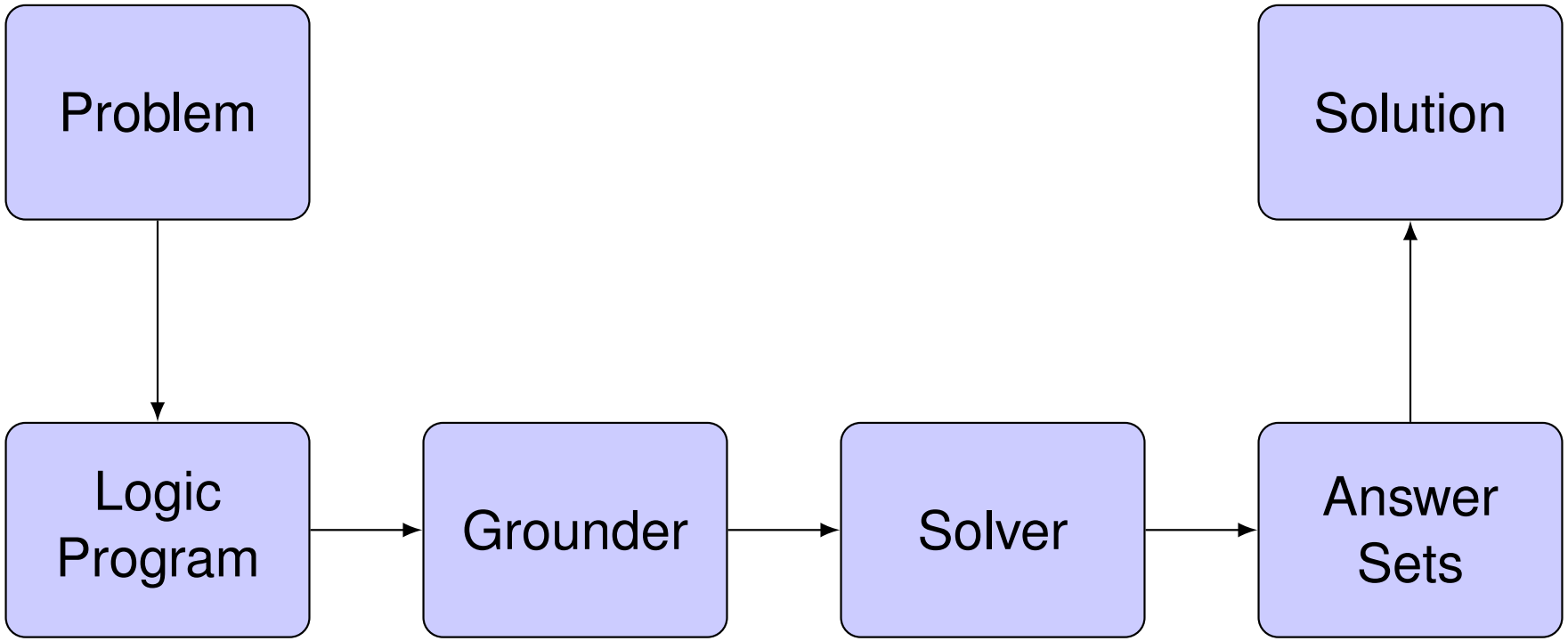
- Rules impose constraints: the meaning of a basic rule is, informally, if all a_i's $i = 1, ..., m$ are established, and none of the a_i's, $i = m + 1, ..., n$ can be established, a must hold.
- Formally, the meaning is defined by the answer-set semantics.

Choice Rules

$$\{a, b, c\} \leftarrow \mathit{condition}$$

- Choice rules allow us to generate assignments which contain any combination of the atoms in the head.
- Useful for describing the space of assignments to search through.

ASP Introduction III



- Grounding variables over large domains is a bottleneck of ASP.
- CASP – A hybrid approach which avoids grounding of such variables but collects the constraints they are involved with and uses a constraint solver to solve them.

Encodings

- Several encodings of the weighted-sequence problem were developed.
- We study the behavior of multiple encodings: from declarative to problem oriented.
- Two families of encodings were developed: *Declarative* (DECL) and *Sequence* (SEQ).

Declarative

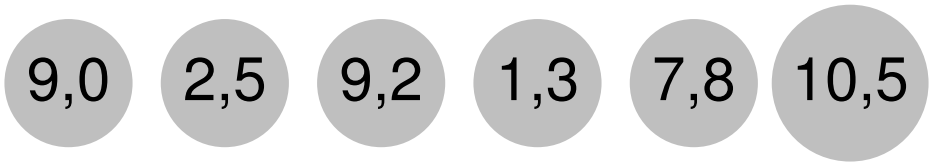
- The *declarative* encoding, or DECL, is a strict interpretation of the problem statement.
- A sequence of n nodes is generated using choice rules.
- Each node is assigned a color using choice rules.
- The cost of the sequence is then calculated and made sure to be at most m.

Declarative Illustration

Given:

- A set of nodes {(9,0), (2,5), (9,2), (1,3), (7,8), (10,5)}
- The maximum cost is 100.

ASP rules are included that “generate” the space of all sequences of nodes.

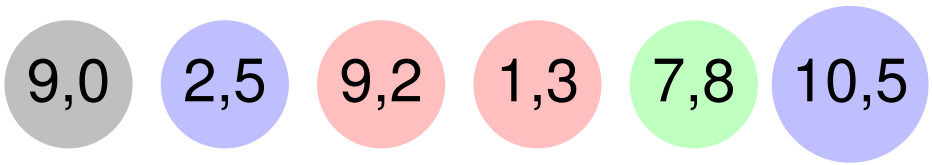


Declarative Illustration

Given:

- A set of nodes {(9,0), (2,5), (9,2), (1,3), (7,8), (10,5)}
- The maximum cost is 100.

Additional rules are added to expand the space to cover all colored sequences.

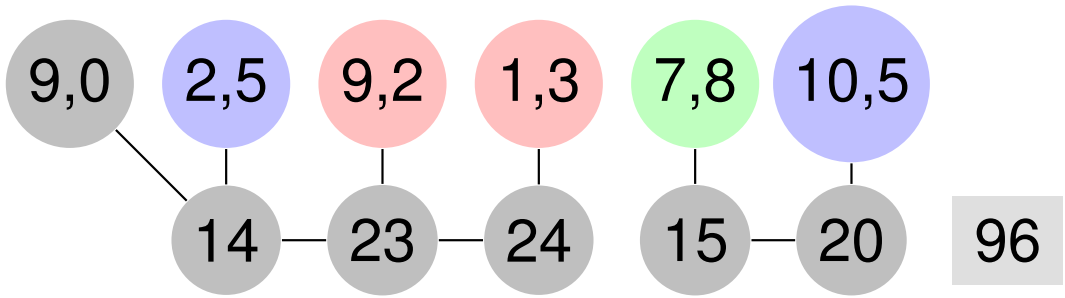


Declarative Illustration

Given:

- A set of nodes {(9,0), (2,5), (9,2), (1,3), (7,8), (10,5)}
- The maximum cost is 100.

Finally, a segment that is a Horn program is included to compute the cost.



Sequence

- The *sequence* encoding (SEQ) is the result of an observation:

Given an arbitrary sequence of nodes it is possible to determine a coloring such that there is no color assignment of lower cost.

$$cost(n_i) = \begin{cases} weight(n_i) + card(n_i) & \text{if } n_i \text{ is green} \\ cost(n_{i-1}) + weight(n_i) & \text{if } n_i \text{ is red} \\ cost(n_{i-1}) + card(n_i) & \text{if } n_i \text{ is blue} \end{cases}$$

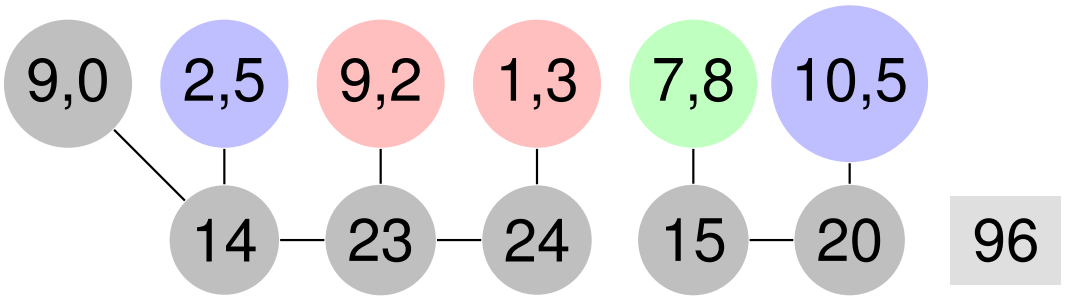
- Let us define a node n_i's *value* $val(n_i)$ as

$$val(n_i) = min(weight(n_i), cardinality(n_i))$$

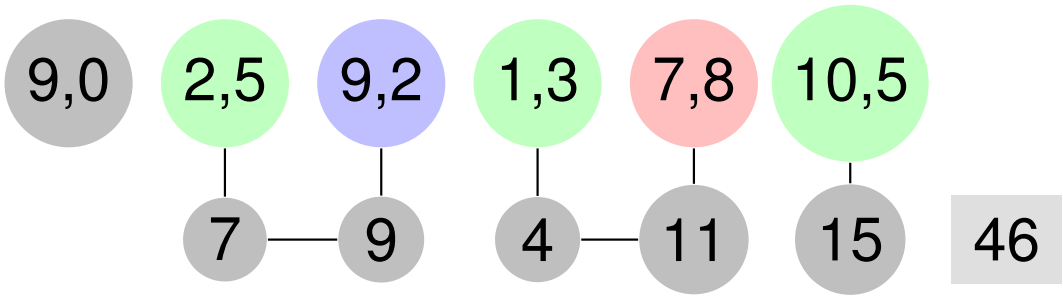
$$color(n_i) = \begin{cases} green & \text{if } w(n_i) + c(n_i) \leq cost(n_{i-1}) + val(n_i) \\ red & \text{otherwise, if } weight(n_i) \leq cardinality(n_i) \\ blue & \text{otherwise} \end{cases}$$

Sequence Illustration

DECL

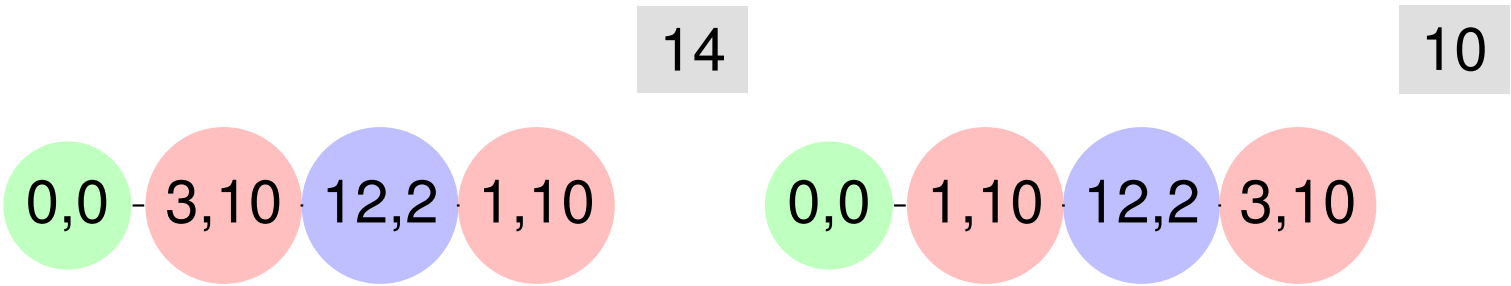


SEQ



Sequence+

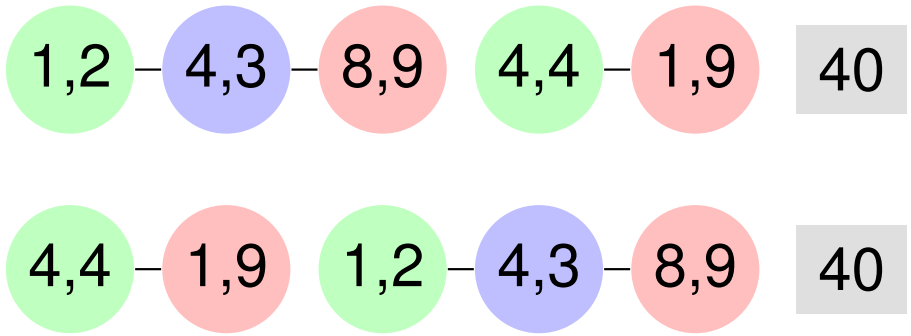
- Let n and n' be two consecutive *non-green* nodes in our sequence M. If $val(n') < val(n)$, then the resulting sequence M' has a total cost of lower than that of M.



- We call this encoding SEQ+.

Sequence++

- Since green nodes only rely upon their own weight and cardinality to calculate cost, any permutation of segments starting with green nodes results in the same cost.
- Thus, by sorting all segments starting with green nodes we are able to further restrict our solutions by **breaking symmetries**.



- We call this encoding SEQ++.

Clingo and Clingcon Encodings

- CLINGCON encodings differ from CLINGO's: some functional predicates are replaced by constraint atoms.
- These predicates contain the cost of nodes in our sequence and resultingly can deal with possibly large integer domains.

Clingo:

$$seqCost(P, Cost) \leftarrow \quad posCost(P, C'),\ seqCost(P-1, C),\ Cost = C + C', coloredPos(P).$$

Clingcon:

$$seqCost(P) =^{\$} \quad posCost(P) + seqCost(P-1) \ \leftarrow \ coloredPos(P)$$

Experimental Setup

- Four classes of instances were developed: *Easy*, *Hard*, *Optimal*, and *Unsatisfiable*.
- We developed *small* and *large* sets of instances for each of these classes.
- In *small* instances, we limit weights and cardinalities of nodes to $[0, 12]$.
- In *large* instances, we limit weights and cardinalities of nodes to $[0, 100]$.
- 30 random problem instances were developed for each class.
- By lowering the maximum cost, we make the problem tighter.

Small Instances

	CLINGO				CLINGCON			
Instance	DECL	SEQ	SEQ+	SEQ++	DECL	SEQ	SEQ+	SEQ++
Easy	0.88	0.75	0.81	0.86	0.02	0.06	0.05	0.15
Hard	4.01	1.19	1.77	2.97	**to=7**	9.50	4.34	5.04
Optimal	26.28	15.75	20.41	15.04	**to=27**	253.30	203.75	34.57
Unsat	180.62	193.79	162.88	27.88	**to=30**	**to=25**	**to=17**	128.63

- On easy instances, CLINGO was outperformed by CLINGCON due to CLINGO's larger need for grounding.
- However, the roles reverse on hard instances and CLINGO is faster for the remaining small tests.

Large Instances

	CLINGO				CLINGCON			
Instance	DECL	SEQ	SEQ+	SEQ++	DECL	SEQ	SEQ+	SEQ++
Easy	21.76	15.38	15.35	16.73	0.01	0.07	0.07	0.08
Hard	22.75	13.96	14.41	23.36	**to=4**	1.76	1.18	0.72
Optimal	**to=1**	97.38	**to=1**	101.95	**to=24**	46.58	23.49	5.07
Unsat	**to=12**	**to=12**	**to=10**	248.81	**to=30**	189.38	92.29	10.43

- In all classes but DECL, CLINGCON appears to scale better than CLINGO.

Comparing Sizes of Ground Programs

Table: Small Instances

	CLINGO				CLINGCON			
Instance	DECL	SEQ	SEQ+	SEQ++	DECL	SEQ	SEQ+	SEQ++
Easy	75268	62739	63099	64719	575	539	899	2519
Hard	29326	26588	26948	28568	575	539	899	2519
Optimal	26842	24495	24855	26475	575	539	899	2519
Unsat	26350	24077	24437	26057	575	539	899	2519

Table: Large Instances

	CLINGO				CLINGCON			
Instance	DECL	SEQ	SEQ+	SEQ++	DECL	SEQ	SEQ+	SEQ++
Easy	1546714	1162451	1162619	1163207	383	358	526	1114
Hard	434237	377120	377288	377876	383	358	526	1114
Optimal	350933	308383	308551	309139	383	358	526	1114
Unsat	349336	307052	307220	307808	383	358	526	1114

Conclusions I

Results indicate that:

- CLINGO is less sensitive to “sophisticated” encodings due to its advanced learning abilities.
- Less advanced solvers like CLINGCON benefit from encodings which limit search space.
- SEQ++’s performance shows that breaking symmetries is important for both CLINGO and CLINGCON.

Conclusions II

- As problems scale in size, CLINGCON performs better than CLINGO.
- CLINGCON's ability to avoid a grounding bottleneck makes it a promising solver for problems with large integer domains.
- However, we believe that CLINGCON needs to incorporate learning to a larger degree in order to compete with CLINGO when grounding is not an issue.

Future Work

- We would also like to benchmark CSP/CLP systems on the WS problem.
- Ultimately, we plan to address the issue of the query optimization problem by applying ASP/CASP/CSP technology.

Thank you!