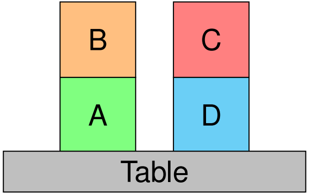
Planning using Lifted Task Representations

Augusto B. Corrêa
augusto.blaascorrea@unibas.ch

5th of December, 2019

Planning in Blocksworld



Objects: *A*, *B*, *C*, *D*, *Table*

Predicates: *on*(*?X*, *?Y*), *clear*(*?X*)

State: Set of ground atoms

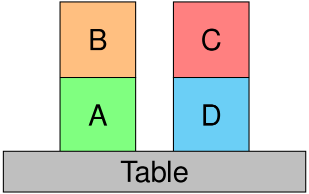
Goal: Stack *C* right above *B*

- i.e., *on*(*C*, *B*)

s_0:

on(*A*, *Table*)
on(*B*, *A*)
on(*D*, *Table*)
on(*C*, *D*)
clear(*B*)
clear(*C*)

Planning in Blocksworld



Objects: $A, B, C, D, Table$

Predicates: $on(?X, ?Y), clear(?X)$

State: Set of ground atoms

Goal: Stack C right above B

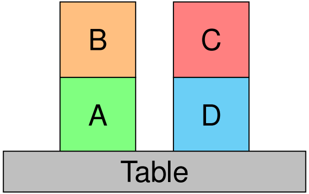
- i.e., $on(C, B)$

Modify state = Apply an action

s_0:

$on(A, Table)$
$on(B, A)$
$on(D, Table)$
$on(C, D)$
$clear(B)$
$clear(C)$

Planning in Blocksworld



Action schema *move(?X, ?Y, ?Z)*

- Preconditions:
  clear(?X), *clear(?Z)*, *on(?X,?Y)*,
  $?X \neq ?Y \neq ?Z$.
- Effects:
  on(?X,?Z), $\neg$*clear(?Z)*, $\neg$*on(?X,?Y)*.

s_0:

on(*A*, *Table*)
on(*B*, *A*)
on(*D*, *Table*)
on(*C*, *D*)
clear(*B*)
clear(*C*)

Ground action *move(C, D, B)* achieves the goal

How to obtain ground actions?

Grounding as a Bottleneck

For n blocks, there are $O(n^3)$ ground actions

Grounding as a Bottleneck

For n blocks, there are $O(n^3)$ ground actions

Even worse

- Organic Synthesis has action schemas with > 10 parameters

  Instance $\#11$ has $71 \cdot 10^{12}$ ground actions

  Solution length of only 2

Grounding as a Bottleneck

For n blocks, there are $O(n^3)$ ground actions

Even worse

- Organic Synthesis has action schemas with > 10 parameters
  - Instance $\#11$ has $71 \cdot 10^{12}$ ground actions
  - Solution length of only 2

There are better methods

- Most popular: Fast Downward grounding algorithm (Helmert 2009)
- It can only ground 8 instances of Organic Synthesis in 16 GB of memory.

Lifted Planning

What we consider lifted planning

- Planning without grounding
- Grounded atoms to represent states

How we plan in this thesis

- Heuristic Search
- Use database techniques to generate successors

Database Theory Background

Database Theory Background

- **Unnamed Relation:** Tables without column names
- **Database:** Set of unnamed relations
- **Relation:** Table with column names (attributes)
- Rows of these tables will be called as tuples

T	
0	0
0	1
1	1

T(X,Y)	
X	**Y**
0	0
0	1
1	1

Relational Algebra Operations

Selection (σ)

T(X,Y)	
X	**Y**
0	0
0	1
1	1

$\sigma_{X=Y}(T(X,Y))$	
X	**Y**
0	0
1	1

Relational Algebra Operations

Projection (π)

T(X,Y)	
X	**Y**
0	0
0	1
1	1

$\pi_Y(T(X, Y))$
Y
0
1

Relation Algebra Operations

Join ($\bowtie$) and semi-join ($\ltimes$)

T(X,Y)	
X	**Y**
0	0
0	1
1	1

R(Y, Z)	
Y	**Z**
0	2
0	5
2	3

$T(X, Y) \bowtie R(Y, Z)$		
X	**Y**	**Z**
0	0	2
0	0	5

$T(X, Y) \ltimes R(Y, Z)$	
X	**Y**
0	0

Semi-join can work as a filter to guarantee global consistency

Query

What are the values of Y that occur simultaneously in $T(X, Y)$ and $R(Y, Z)$?

T(X,Y)	
X	**Y**
0	0
0	1
1	1

R(Y,Z)	
Y	**Z**
0	2
0	5
2	3

Q(Y)
Y
0

Queries can be solved using relational algebra

$$Q(Y) := \pi_Y(T(X, Y) \bowtie R(Y, Z))$$

Conjunctive Queries

Logical perspective:

$$(\exists X)(\exists Z)\, T(X, Y) \wedge R(Y, Z).$$

Conjunctive Queries

Logical perspective:

$$(\exists X)(\exists Z)\, T(X, Y) \wedge R(Y, Z).$$

Some queries can be expressed using the following fragment

$$(\exists Z_1) \ldots (\exists Z_m) \psi(X_1, \ldots, X_n, Z_1, \ldots, Z_n),$$

where $\psi(X_1, \ldots, X_n, Z_1, \ldots, Z_n)$ is a conjunction of relations

Conjunctive Queries

Logical perspective:

$$(\exists X)(\exists Z)\, T(X, Y) \wedge R(Y, Z).$$

Some queries can be expressed using the following fragment

$$(\exists Z_1)\ldots(\exists Z_m)\psi(X_1, \ldots, X_n, Z_1, \ldots, Z_n),$$

where $\psi(X_1, \ldots, X_n, Z_1, \ldots, Z_n)$ is a conjunction of relations

Conjunctive queries are queries that can be represented as above

- More common notation:

$$Q(Y) :- \; T(X, Y), R(Y, Z).$$

- It can be solved using only selection, projection, and join*

Tractability of Conjunctive Queries

- Intermediate relations can have an exponential number of tuples
- In general, no efficient method exists

Tractability of Conjunctive Queries

- Intermediate relations can have an exponential number of tuples
- In general, no efficient method exists
- Some queries are computable in time polynomial in the input and output
  - Tractability depends on the structure

Acyclicity

- Every query Q has an associated hypergraph H_Q
  - Every free variable is a node
  - Every relation in the body is a hyperedge containing the nodes of its variables
- If H_Q is acyclic, then computing Q is tractable
  - Full reducer: Eliminate tuples not participating in the answer of Q

Acyclicity and Full Reducer

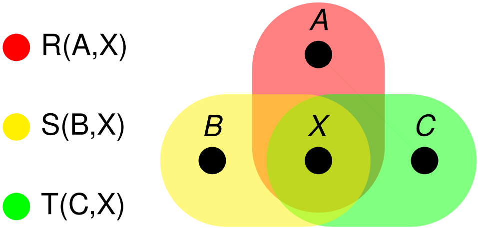
Acyclicity and Full Reducer

Idea: Filter out all “dangling tuples” in advance

Acyclicity and Full Reducer

Idea: Filter out all “dangling tuples” in advance

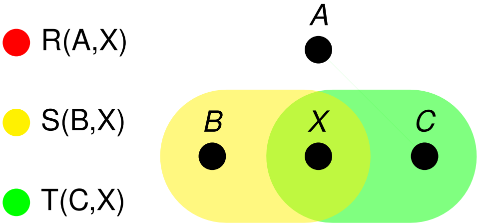
$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$



Acyclicity and Full Reducer

Idea: Filter out all “dangling tuples” in advance

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$



$$R(A, X) := R(A, X) \ltimes S(B, X)$$

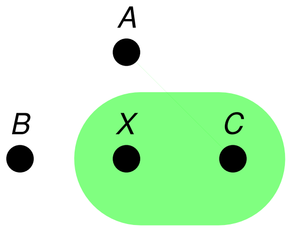
$$S(B, X) := S(B, X) \ltimes R(A, X)$$

Acyclicity and Full Reducer

Idea: Filter out all "dangling tuples" in advance

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$

- R(A,X)
- S(B,X)
- T(C,X)

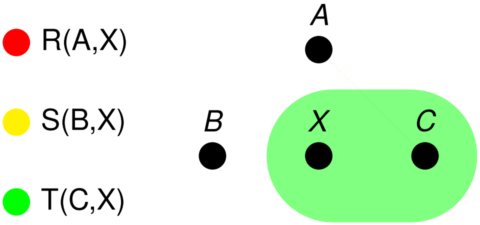


$$R(A, X) := R(A, X) \ltimes S(B, X)$$
$$S(B, X) := S(B, X) \ltimes T(C, X)$$
$$T(C, X) := T(C, X) \ltimes S(B, X)$$
$$S(B, X) := S(B, X) \ltimes R(A, X)$$

Acyclicity and Full Reducer

Idea: Filter out all “dangling tuples” in advance

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$

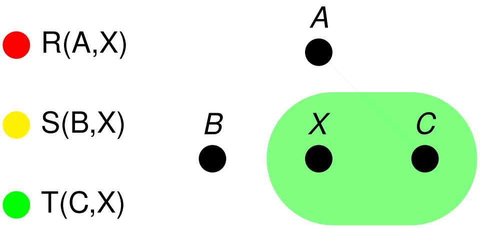


$$R(A, X) := R(A, X) \ltimes S(B, X)$$
$$S(B, X) := S(B, X) \ltimes T(C, X)$$
$$T(C, X) := T(C, X) \ltimes S(B, X)$$
$$S(B, X) := S(B, X) \ltimes R(A, X)$$
$$Q(A, B, C, X) := (T(C, X) \bowtie S(B, X)) \bowtie R(A, X)$$

Acyclicity and Full Reducer

Idea: Filter out all "dangling tuples" in advance

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$



$$
\begin{aligned}
R(A, X) &:= R(A, X) \ltimes S(B, X) \\
S(B, X) &:= S(B, X) \ltimes T(C, X) \\
T(C, X) &:= T(C, X) \ltimes S(B, X) \\
S(B, X) &:= S(B, X) \ltimes R(A, X) \\
Q(A, B, C, X) &:= (T(C, X) \bowtie S(B, X)) \bowtie R(A, X)
\end{aligned}
$$

Intermediate relations are monotonic $\Longrightarrow$ $Q(A, B, C, X)$ is the largest relation

$\Longrightarrow$ Polynomial in the input and output

Planning as Database Progression

Planning as Database Progression

- States as databases
- One unnamed relation per predicate
- Tuple (a, b) is in table of a predicate P if $P(a, b)$ is true in the state
- Applying an action to a state = Update the database

$on(A, Table)$
$on(B, A)$
$on(D, Table)$
$on(C, D)$
$clear(B)$
$clear(C)$

on	
A	Table
B	A
D	Table
C	D

clear
B
C

Successor Generation

Successor Generation

Preconditions of *move(?X, ?Y, ?Z)*:

$$\textit{clear}(?X),\ \textit{clear}(?Z),\ \textit{on}(?X, ?Y), ?X \neq ?Y \neq ?Z.$$

Successor Generation

Preconditions of *move(?X, ?Y, ?Z)*:

$$clear(?X),\ clear(?Z),\ on(?X, ?Y), ?X \neq ?Y \neq ?Z.$$

Objects instantiating *?X, ?Y, ?Z* are the tuples in

$$Q(?X, ?Y, ?Z) :- clear(?X),\ clear(?Z),\ on(?X, ?Y), ?X \neq ?Y \neq ?Z.$$

Instantiating of action schemas = Conjunctive query over the preconditions

Are the schemas in the IPC acyclic?

Precondition with acyclic hypergraph $\implies$ Efficient successor generation

Are the schemas in the IPC acyclic?

Precondition with acyclic hypergraph $\implies$ Efficient successor generation

Benchmark	**Schemas**	**Acyclic**	**Avg. Proportion**
IPC 1998-2018	59520	56668 (95.8%)	83.4%
Org. Synthesis – Original	760	65 (8.6%)	8.6%

Are the schemas in the IPC acyclic?

Precondition with acyclic hypergraph $\Longrightarrow$ Efficient successor generation

Benchmark	Schemas	Acyclic	Avg. Proportion
IPC 1998-2018	59520	56668 (95.8%)	83.4%
Org. Synthesis – Original	760	65 (8.6%)	8.6%

- Many preconditions have cyclicity caused because of inequalities
- Considering acyclicity with inequalities increases proportion to 86.7%
- Organic Synthesis: 8.6% $\rightarrow$ 91.5%
  - FPT algorithm for acyclic queries with inequalities

Existentially Quantified Variables

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$

Precondition: $R(A, X), S(B, X), T(C, X)$

Existentially Quantified Variables

$$Q(A, B, C, X) :\!- \; R(A, X), S(B, X), T(C, X)$$

Precondition: $R(A, X), S(B, X), T(C, X)$
Effect: $P(X)$

Existentially Quantified Variables

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$

Precondition: $R(A, X), S(B, X), T(C, X)$
Effect: $P(X)$

- Different instantiations of A, B, and C for a same X lead to a same successor
- Interested in the values of X. Other variables can be existentially quantified

Existentially Quantified Variables

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$

Precondition: $R(A, X), S(B, X), T(C, X)$
Effect: $P(X)$

- Different instantiations of A, B, and C for a same X lead to a same successor
- Interested in the values of X. Other variables can be existentially quantified

$Q(X) = \pi_X(Q(A, B, C, X)) \implies$ Not polynomial in the output size anymore!

Existentially Quantified Variables

$$Q(A, B, C, X) :- R(A, X), S(B, X), T(C, X)$$

Precondition: $R(A, X), S(B, X), T(C, X)$
Effect: $P(X)$

- Different instantiations of A, B, and C for a same X lead to a same successor
- Interested in the values of X. Other variables can be existentially quantified

$Q(X) = \pi_X(Q(A, B, C, X)) \implies$ Not polynomial in the output size anymore!

Yannakakis' algorithm: Full reducer + join program interleaved with projections

- Project variables out as soon as possible
- Polynomial in the output and input sizes again (with overhead)

Experimental Results

- IPC Benchmark (1056 instances, 53 domains)
  - STRIPS domains with inequalities
- Hard-to-ground Benchmark (418 instances, 6 domains)
  - Organic Synthesis: Original, MIT, and Alkene
  - Genome Edit Distance: Split and non-split
  - Pipesworld-Tankage (non-spit)
- 30 minutes and 16 GiB
- Source code is available online

Methods

- Successor generators based on join programs
  - J^R: Randomly ordered
  - J: PDDL Order
  - $J^<$: Increasing arity
- Successor generators based on acyclicity of preconditions
  - $FR^{SJ,<}$: Full reducer + Join program by arity
  - Y: Full reducer + Yannakakis' algorithm
  - Cyclic preconditions: “partial reducer” + Join program by arity
- Compare to L-RPG and Fast Downward 19.06

What is the impact of the full reducer?

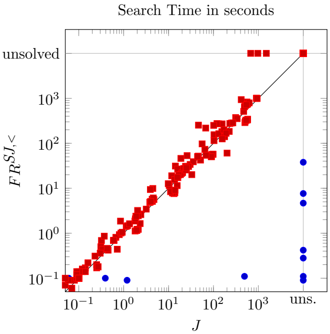
IPC Benchmark	**# of Inst.**	J^R	J	$J^<$	$FR^{SJ,<}$	FD
organic-synthesis-opt18	20	2	11	10	**19**	8
Total	1560	352.3	454	443	**464**	586

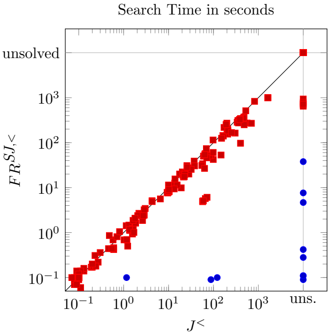
BFS in the IPC benchmark

What is the impact of the full reducer?

IPC Benchmark	# of Inst.	J^R	J	$J^<$	$FR^{SJ,<}$	FD
organic-synthesis-opt18	20	2	11	10	**19**	8
Total	1560	352.3	454	443	**464**	586

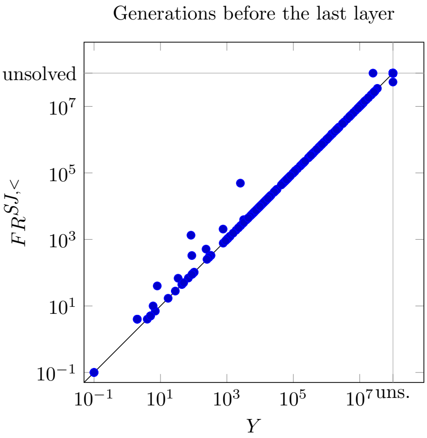
BFS in the IPC benchmark





What if we only consider variables in the effects?

What if we only consider variables in the effects?



- Significant improvement only in Organic Synthesis
- Structure of the task eliminates duplication

What about hard-to-ground domains?

What about hard-to-ground domains?

		BFS			GBFS		
Hard-to-ground Benchmark	**# of Inst.**	$FR^{SJ,<}$	Y	FD	$FR^{SJ,<}$	Y	FD
Genome Edit Distance	312	44	44	**46**	**312**	**312**	**312**
Organic Synthesis	56	**44**	**44**	20	47	**50**	20
Pipesworld Tankage	50	11	10	**14**	**22**	**22**	20
Total	418	**99**	98	80	381	**384**	352

Hard-to-ground domains using BFS and GBFS with goal-count

What about hard-to-ground domains?

Hard-to-ground Benchmark	# of Inst.	BFS $FR^{SJ,<}$	BFS Y	BFS FD	GBFS $FR^{SJ,<}$	GBFS Y	GBFS FD
Genome Edit Distance	312	44	44	**46**	**312**	**312**	**312**
Organic Synthesis	56	**44**	**44**	20	47	**50**	20
Pipesworld Tankage	50	11	10	**14**	**22**	**22**	20
Total	418	**99**	98	80	381	**384**	352

Hard-to-ground domains using BFS and GBFS with goal-count

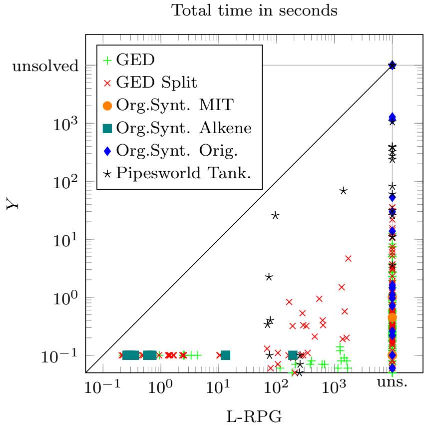
- J and $J^{<}$ have coverage similar to Fast Downward
- Y and $FR^{SJ,<}$ are faster than Fast Downward in almost all instances
  - Fast Downward memory and time consumption is dominated by the translator

What about other lifted planners?

- **L-RPG:** Lifted planner using a lifted version of FF (Ridder 2013)

What about other lifted planners?

- **L-RPG:** Lifted planner using a lifted version of FF (Ridder 2013)



	# of Inst.	GBFS $FR^{SJ,<}$	GBFS Y	L-RPG
GED	312	**312**	**312**	113
Org.Synt.	56	47	**50**	14
Pipes. Tank.	50	**22**	**22**	10
Total	418	381	**384**	137

Conclusion & Future Work

Conclusion:

- New successor generator methods using lifted representations
- Lifted successor generation is tractable in several domains
- Well-suited for domains where grounding is a bottleneck
- Good performance in the hard-to-ground domains tested

Conclusion & Future Work

Conclusion:

- New successor generator methods using lifted representations
- Lifted successor generation is tractable in several domains
- Well-suited for domains where grounding is a bottleneck
- Good performance in the hard-to-ground domains tested

Future Work:

- Lifted heuristics
- Partially-grounded actions to eliminate acyclicity
- Other database techniques