Double Description Method in Cost Partitioning

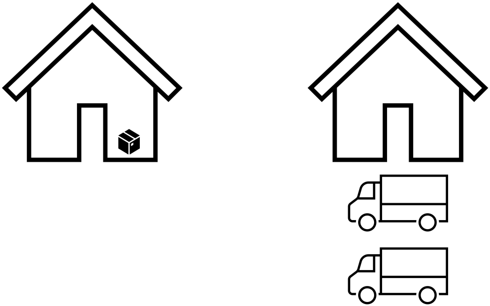
Raphael Kübler

University of Basel, Switzerland

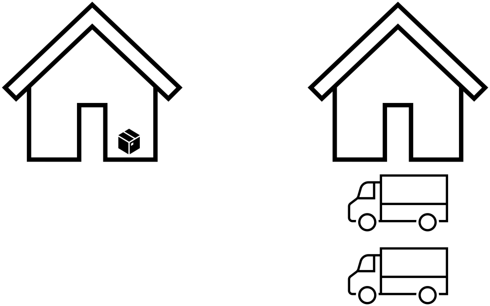
November 11, $2 \times 3 \times 337$

Planning

Logistics Example



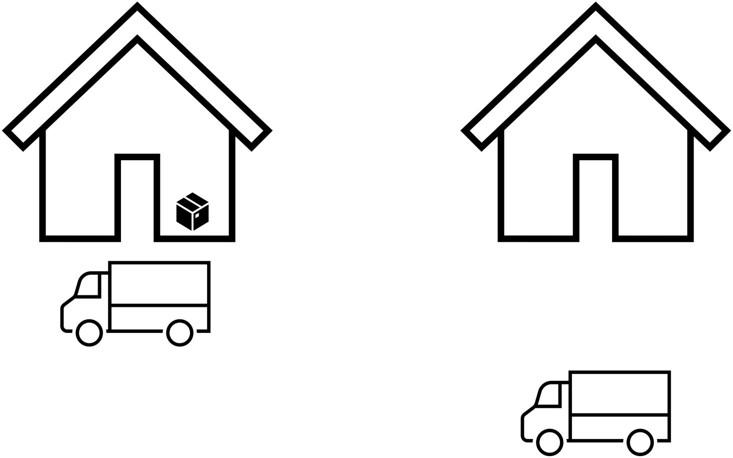
Logistics Example



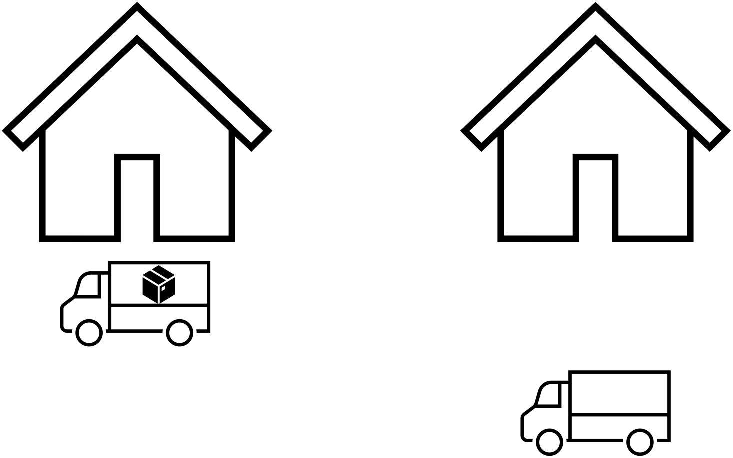
(informal) task description

actions: trucks can drive from one location to the other and (un-)load package
goal: find sequence of actions such that package is at other location

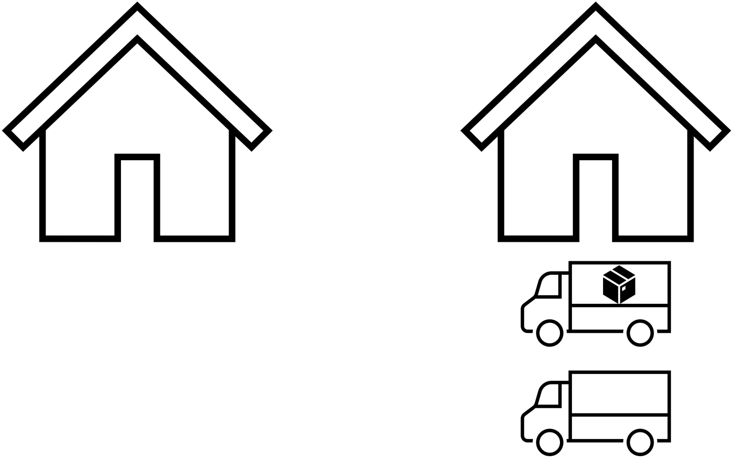
Logistics Example



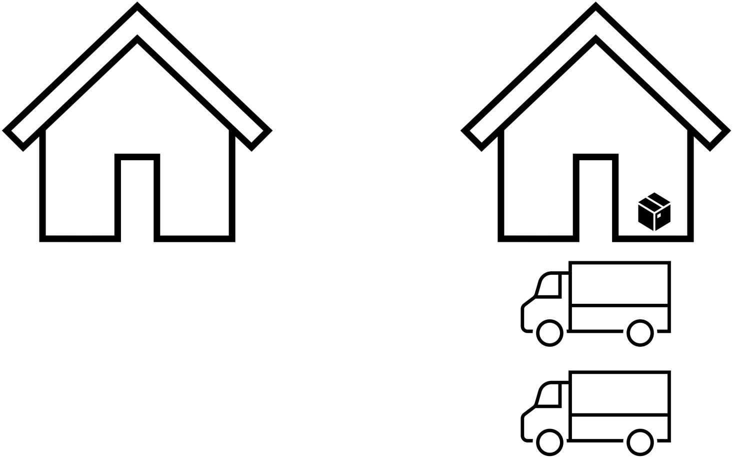
Logistics Example



Logistics Example



Logistics Example



Finding Plans

Heuristic Search

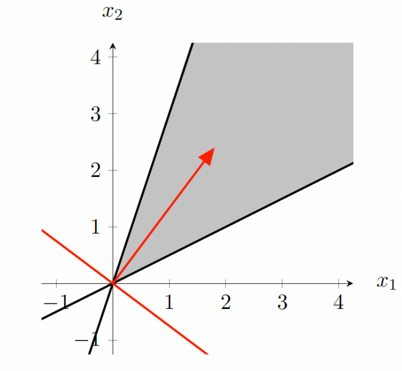
- cost partitioning over abstraction heuristics
- calculating (optimal) cost partitioning involves solving a large linear program $\Rightarrow$ computationally expensive

Linear Programs

Example

$$\max \quad \frac{3}{4}x_1 + x_2 \quad \text{s.t.}$$
$$\frac{1}{2}x_1 - x_2 \leq 0$$
$$-3x_1 + x_2 \leq 0$$

Solution Space



Linear Programs

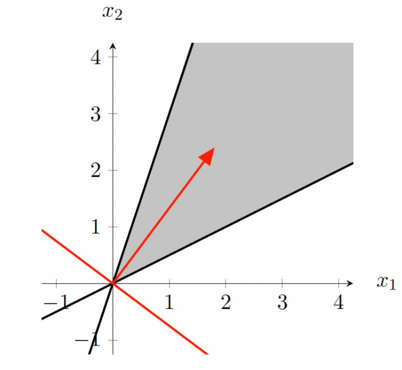
Example

$$\max \quad \frac{3}{4}x_1 + x_2 \quad \text{s.t.}$$
$$\frac{1}{2}x_1 - x_2 \leq 0$$
$$-3x_1 + x_2 \leq 0$$

Generating Rays

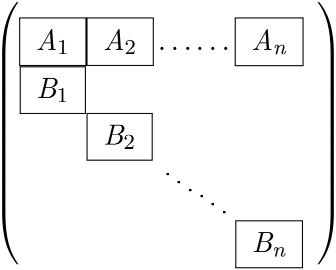
Every solution can be written as a finite sum of generating rays

Solution Space



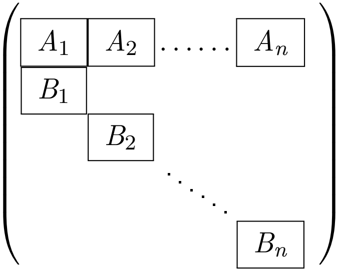
Decomposition

Dantzig-Wolfe Decomposition



- solves linear programs with special structure
- starts with linear program that uses less columns
- iteratively adds columns that improve solution
- to know which columns to add, it solves the pricing problem

Dantzig-Wolfe in Cost Partitioning



- B_i's are the abstraction heuristics used in cost partitioning
- A_i's form the cost partitioning constraints

Pricing Problem

Minimize $c(y) - h$ subject to
$h \leq$ heuristic i under cost c

$\Rightarrow$ one pricing problem per abstraction

Pricing Problem in Cost Partitioning

Constraints

$d_0 = 0$

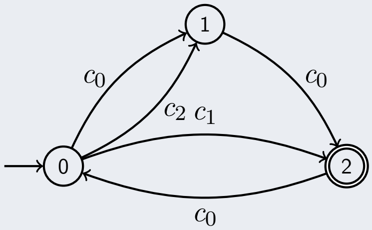
$d_t \leq d_s + c_\ell$ for all transitions from state s to t with cost c_ℓ

$h \leq d_{s^*}$ for all goal states s^*

Example

$$d_0 = 0$$
$$d_0 \leq d_2 + c_0$$
$$d_1 \leq d_0 + c_0$$
$$d_1 \leq d_0 + c_2$$
$$d_2 \leq d_0 + c_1$$
$$d_2 \leq d_1 + c_0$$
$$h \leq d_2$$

Graph



Finding Generating Rays

Methods

Double Description

Used to calculate the generating rays of linear constraints

Fourier-Motzkin Elimination

Used to project out variables of linear constraints

Strategies

Pricing Problem (constraints over h,c,d)

project out distance variables (with FM) →

constraints over h,c

↓ *compute gen. rays (with DD)*

generating rays over h,c.d

project out distance variables (ignore) →

generating rays over h,c

↓ *compute gen. rays (with DD)*

Projection - A Closer Look

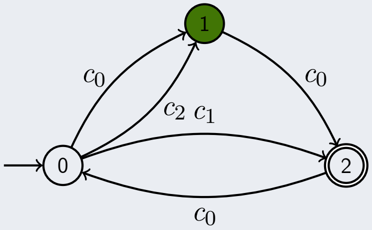
Fourier-Motzkin Elimination

1. choose variable to project out

Example

$$d_0 = 0$$
$$d_0 \leq d_2 + c_0$$
$$d_1 \leq d_0 + c_0$$
$$d_1 \leq d_0 + c_2$$
$$d_2 \leq d_0 + c_1$$
$$d_2 \leq d_1 + c_0$$
$$h \leq d_2$$

Graph



Projection - A Closer Look

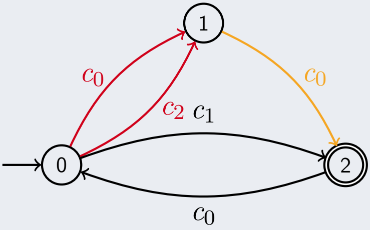
Fourier-Motzkin Elimination

2. group constraints with respect to chosen variable

Example

$$d_0 = 0$$
$$d_0 \leq d_2 + c_0$$
$$d_1 \leq d_0 + c_0$$
$$d_1 \leq d_0 + c_2$$
$$d_2 \leq d_0 + c_1$$
$$d_2 \leq d_1 + c_0$$
$$h \leq d_2$$

Graph



Projection - A Closer Look

Fourier-Motzkin Elimination

3. combine constraints from different groups ⇒ new constraints without chosen variables

Example

$$d_0 = 0$$

$$d_0 \leq d_2 + c_0$$

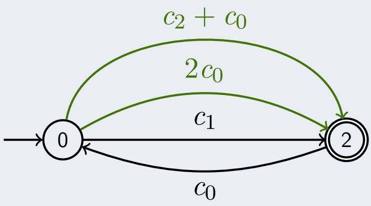
$$d_2 \leq d_0 + c_0 + c_0$$

$$d_2 \leq d_0 + 2c_2$$

$$d_2 \leq d_0 + c_1$$

$$h \leq d_2$$

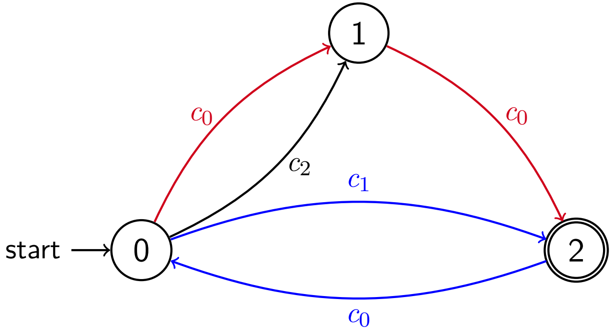
Graph



Potential Shortcut for Projection

Observation

- nodes get eliminated ⇒ new edges
- edges represent open or closed walks in the original graph



Potential Shortcut for Projection

Applying Fourier-Motzkin elimination to all distance variables $\Rightarrow$ constraints represent open walks from start to goal or closed walks in the original graph...

Potential Shortcut for Projection

Applying Fourier-Motzkin elimination to all distance variables $\Rightarrow$ constraints represent open walks from start to goal or closed walks in the original graph...

... but it gets even better

Potential Shortcut for Projection

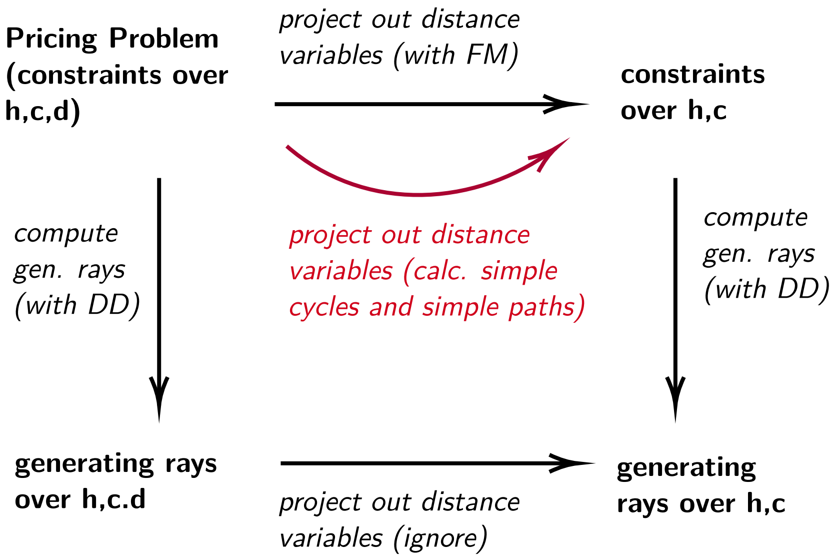
Applying Fourier-Motzkin elimination to all distance variables ⇒ constraints represent open walks from start to goal or closed walks in the original graph...

... but it gets even better

Theorem

The constraint system representing all simple paths and simple cycles has the same solution space as the original pricing problem.

Strategies


Pricing Problem (constraints over h,c,d)
project out distance variables (with FM)
constraints over h,c
project out distance variables (calc. simple cycles and simple paths)
compute gen. rays (with DD)
compute gen. rays (with DD)
generating rays over h,c.d
project out distance variables (ignore)
generating rays over h,c

Experiments

Setup

- 1590 tasks from International Planning Competition
- considered projections onto one or two variables for every task
- run involved calculating all generating rays for all projections onto one or two variables of one task
- time limit of 5 min per run
- memory limit of 2 GiB per run
- experiment run on Intel Xeon Silver 4114

Number Of Solved Tasks

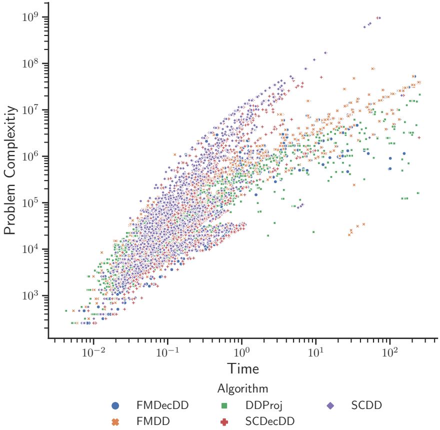
Sys1	Solved	Time Limit Reached	Memory Limit Reached
DDProj	1398	**192**	–
FMDD	1560	9	**21**
SCDD	**1590**	–	–

Sys2	Solved	Time Limit Reached	Memory Limit Reached
DDProj	163	**1427**	–
FMDD	196	327	**1107**
SCDD	351	**1013**	226

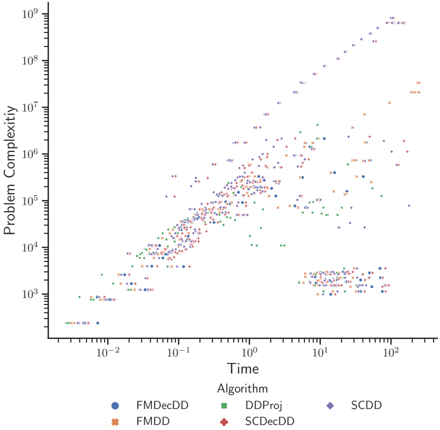
Complexity vs. Runtime

Idea: complexity can be measured by multiplying number of constraints (at the beginning) by the number of variables

SYS1



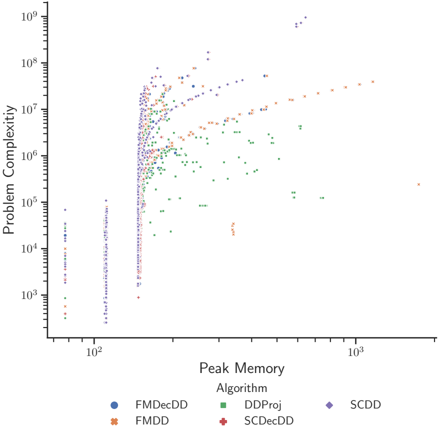
SYS2



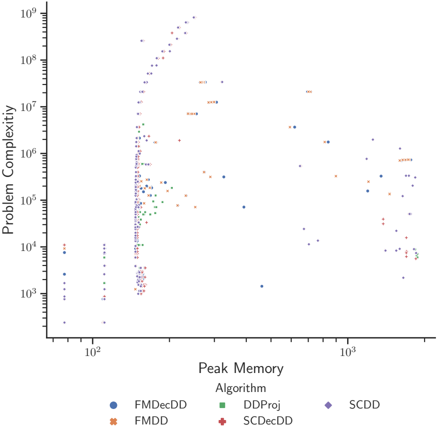
Complexity vs. Peak Memory Consumption

Idea: complexity can be measured by multiplying number of constraints (at the beginning) by the number of variables

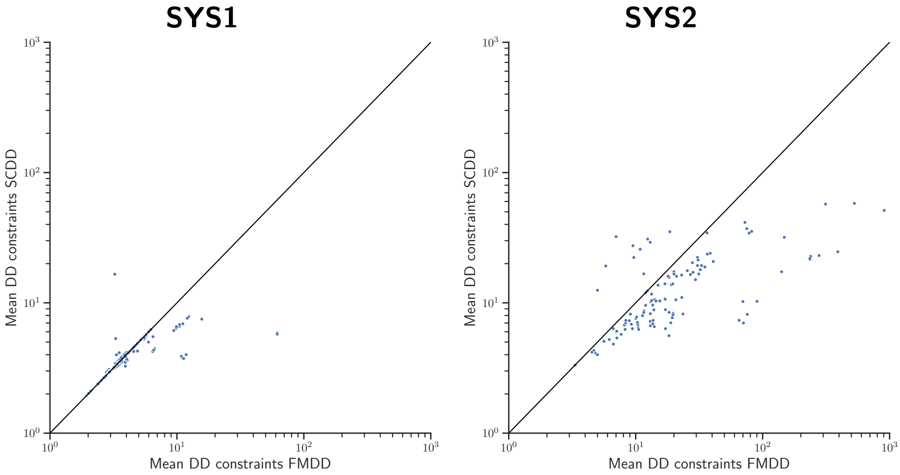
SYS1



SYS2



Redundant Constraints



Conclusion and Outlook

Conclusion

- projecting first seems to be the superior strategy
- projecting by calculating simple paths and simple cycles seems to outperform (naive) Fourier-Motzkin
- our measure of complexity seems to be a good indicator for runtime but not for peak memory consumption

Outlook

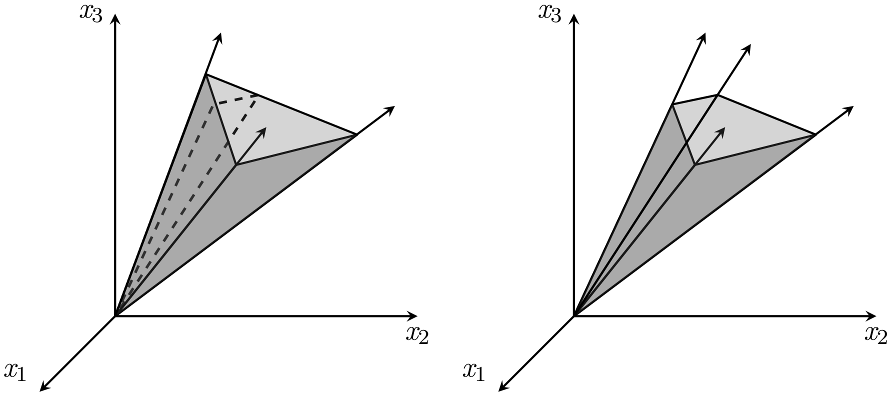
- Does precomputing the generating rays improve performance of cost partitioning?
- detecting and removing redundant constraints
- interesting generating rays

Pricing Problem

Minimize $c(y) - h$ subject to
$h \leq$ heuristic i under cost c

- decomposing solution space

Double Description Method



Decomposition Polyhedral Cones