Kernel search for location problems

M.Grazia Speranza

Department of Economics and Business

University of Brescia

Spanish Network on Location Analysis and Related Problems

Barcelona, November 27, 2015

Exact methods for MILP problems

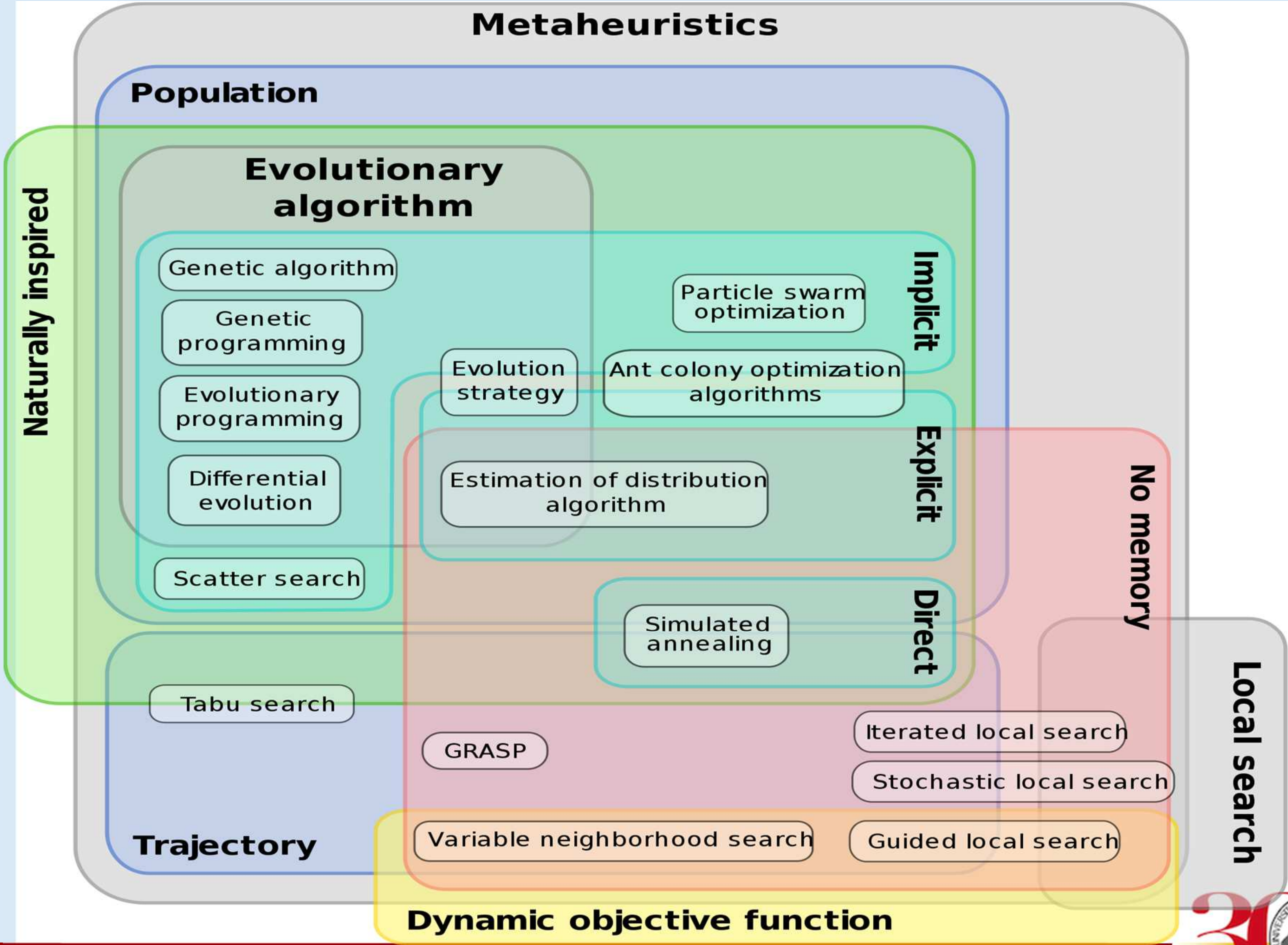
- General
- More and more powerful
- Commercial software

- Inadequate for instances of medium/large size

Heuristics

- Greedy
- Local search
- Metaheuristics:
  - Simulated annealing
  - Tabu search
  - Variable neighborhood search
  - Ant colony optimization
  - Genetic algorithms

Heuristics



Metaheuristics
Population
Naturally inspired
Evolutionary algorithm
Genetic algorithm
Genetic programming
Evolutionary programming
Differential evolution
Scatter search
Evolution strategy
Particle swarm optimization
Ant colony optimization algorithms
Implicit
Estimation of distribution algorithm
Explicit
No memory
Simulated annealing
Direct
Tabu search
GRASP
Iterated local search
Stochastic local search
Local search
Trajectory
Variable neighborhood search
Guided local search
Dynamic objective function

Heuristics

- Needed where exact methods inadequate
- Often designed for a specific problem
- May require great implementation effort

And if the problem slightly changes?

The heuristic needs to be re-designed

This may limit the impact of OR

Heuristics and exact methods

Can we combine the strength of exact methods
with the strength of heuristics?

How?

- What kind of MILPs?
- Which heuristic scheme?
- Tailored or general?

Matheuristics

MILP in heuristics

K. Doerner, V. Schmid, Survey: Matheuristics for rich vehicle routing problems, LNCS, 2010
M. Ball, Heuristics based on mathematical programming, Surveys in Operations Research and Management Science, 2011
L. Bertazzi, M.G. Speranza, Matheuristics for inventory routing problems, in 'Hybrid Algorithms…', Montoya-Torres et al (eds), 2012
C. Archetti, M.G. Speranza, A survey on matheuristics for routing problems, EJCO, 2014

…. often still tailored algorithms

Heuristics for MILP

Goal:

A simple heuristic for any MILP problem
or at least for classes of MILP problems

Must be based on MILP formulation

A general matheuristic

Observations/starting points

- Often in an optimal solution there are few non-zero variables
- Often basic variables in the LP-relaxation are good predictors of non-zero variables in an optimal MILP solution
- Often reduced costs are good predictors of the likelihood of a non-basic variable to be non-zero in a MILP optimal solution

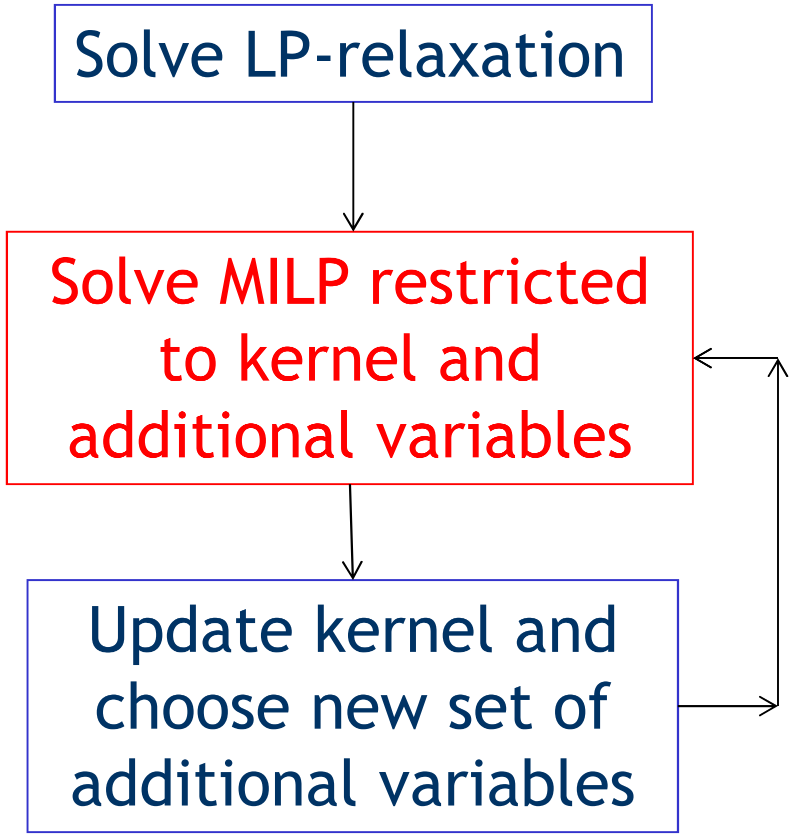
Kernel search

Basic concepts:

Kernel = set of 'promising' (likely to be non-zero) variables

MILPs restricted to $\underbrace{\textbf{kernel and some more variables}}_{\text{marginally wrong (few variables missing)}}$

Kernel search – general scheme



Experience with Kernel Search

Portfolio optimization
Mansini, Speranza, EJOR (1999)
Angelelli, Mansini, Speranza, JCOA (2010)

Multi-dimensional Knapsack Problem
Angelelli, Mansini, Speranza, C&OR (2010)

Index tracking
Guastaroba, Speranza, EJOR (2012)

Capacitated Facility Location Problem
Guastaroba, Speranza, JOH (2013)

BILP problems (Single source CFLP)
Guastaroba, Speranza, EJOR (2014)

Bi-objective enhanced index tracking
Guastaroba, Filippi, Speranza, Omega (2015)

Experience with Kernel Search

Portfolio optimization
Mansini, Speranza, EJOR (1999)

One continuous or integer variable associated with each binary variable

Initial intuitions:
- few securities in an optimal portfolio
- identification of one set of promising variables (kernel) and restricted MILP

Angelelli, Mansini, Speranza, JCOA (2010)

Concept of bucket

Experience with Kernel Search

Multi-dimensional Knapsack Problem
Angelelli, Mansini, Speranza, C&OR (2010)

A pure binary problem

All variables explored

Increasing size of the kernel

Experience with Kernel Search

Index tracking
Guastaroba, Speranza, EJOR (2012)

Binary and other continuous variables

A limited number of variables explored

Variables can be removed from the kernel

Experience with Kernel Search

Capacitated Facility Location Problem
Guastaroba, Speranza, JOH (2013)

Extension
to problems with
binary variables and
a set of continuous
variables associated
with each binary
variable

Experience with Kernel Search

BILP problems
(Single source CFLP)
Guastaroba, Speranza, EJOR (2014)

Extension to any Binary Problem

Some hard variable fixing

Experience with Kernel Search

Bi-objective enhanced index tracking

Guastaroba, Filippi, Speranza, submitted (2014)

Extension to a multi-objective problem

This talk

Multi-dimensional Knapsack Problem
Angelelli, Mansini, Speranza, C&OR (2010)

Capacitated Facility Location Problem
Guastaroba, Speranza, JOH (2013)

BILP problems (Single source CFLP)
Guastaroba, Speranza, EJOR (2014)

Multi-dimensional Knapsack Problem (MKP)

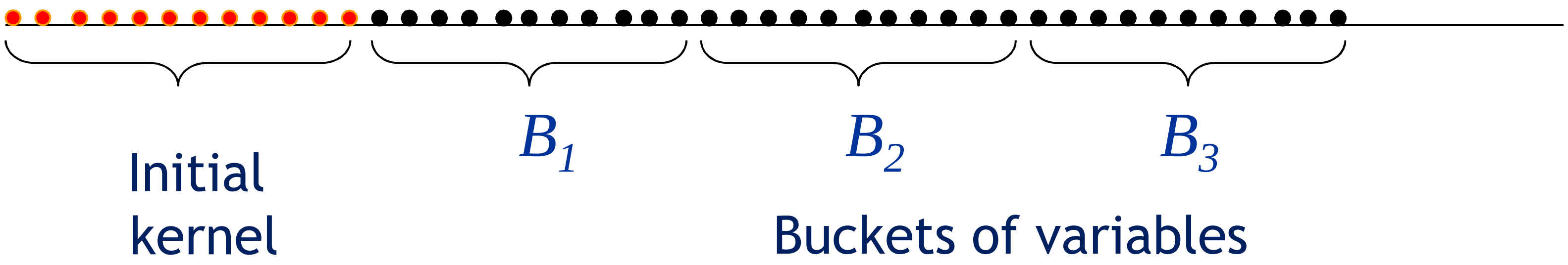
$$\max \sum_{j=1}^{n} p_j z_j$$

$$\sum_{j=1}^{n} w_{ij} z_j \le c_i \quad i = 1, \ldots, m$$

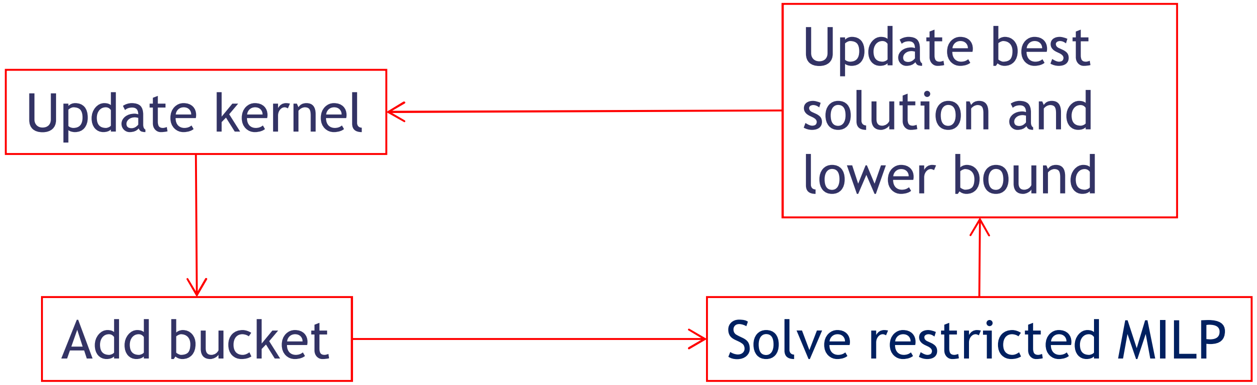
$$z_j \in \{0, 1\} \quad j = 1, \ldots, n$$

- Pure binary problem, one set of binary variables
- Strongly NP-hard problem
- Playground for heuristics and metaheuristics
- Benchmark instances available

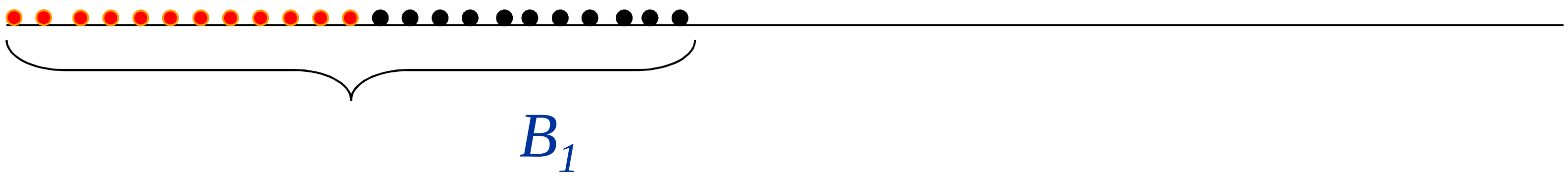
Kernel search – initialization phase



Kernel search – iterative phase

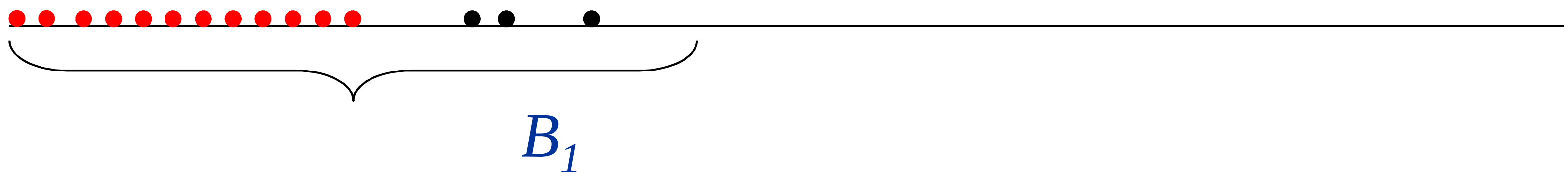


Kernel search – iterative phase



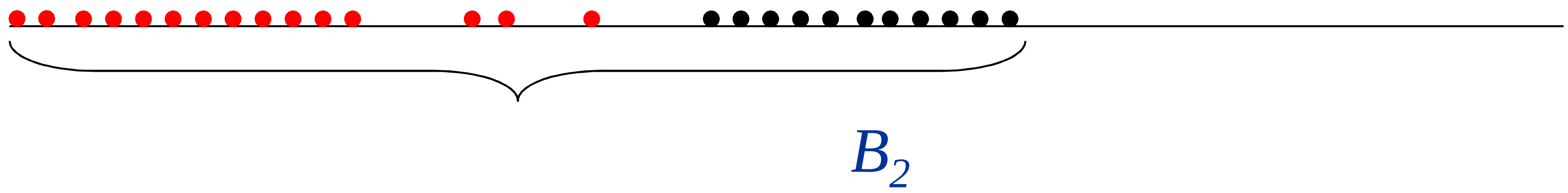
Restricted MILP

Kernel search – iterative phase



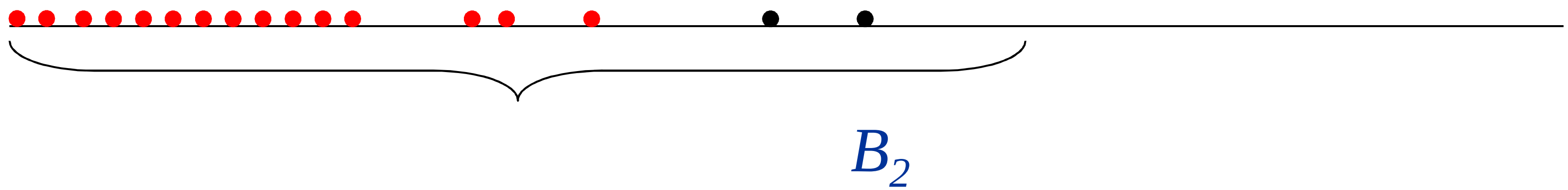
Updated kernel

Kernel search – iterative phase



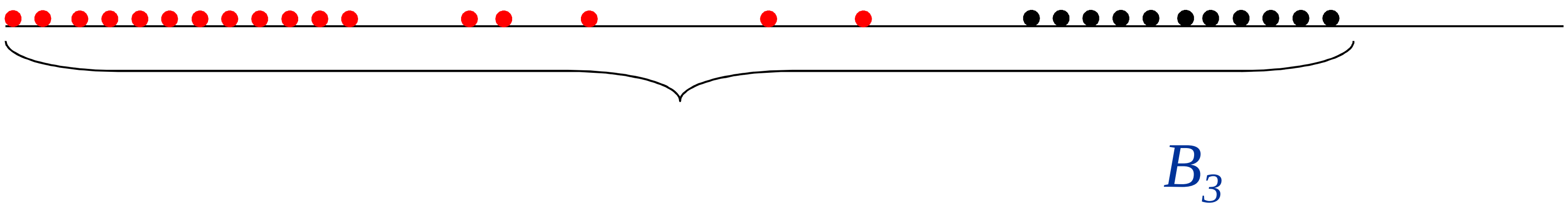
Restricted MILP

Kernel search – iterative phase



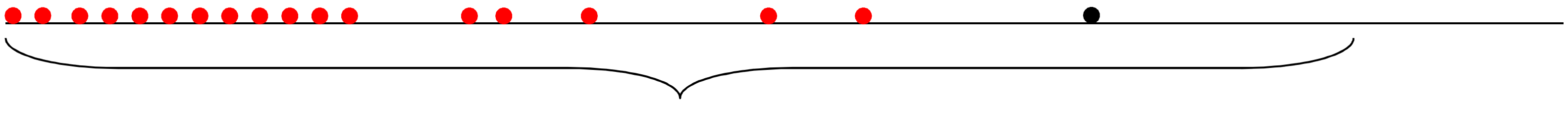
Updated kernel

Kernel search – iterative phase



Restricted MILP

Kernel search – iterative phase



B_3

Updated kernel

MKP - Kernel search

- Initial kernel — Basic LP variables
- Sorting of variables — Reduced costs
- Creation of buckets — Small or big?

  Fixed or variable?

  Disjoint or overlapping?

MKP: Kernel search

- At each restricted MILP:
  - Selection of at least one item from the current bucket
  - Improvement of the current best solution
- Increasing size of the kernel
- Time limit on the solution of each restricted MILP
- All buckets explored

MKP: Computational results

- MILP Solver: CPLEX 9.0
- PC AMD Athlon ™ 2000+ Pentium 3GHz, RAM 2GB
- Fixed-bucket-I(1), Fixed-bucket-I(0.1); Fixed-bucket-I(0.2)
  - Time limit: 1 hour
  - Buckets of length equal to the number of basic variables, 10% of the number, 20% of the number

MKP: Chu-Beasley instances

270 benchmark instances:

- n = 100, 250, 500; m = 5, 10, 30
  - 30 instances for each pair n,m
  - w_{ij} integer drawn in $U(0,1000)$
  - $c_i = \alpha \sum_j w_{ij}$ tightness ratio α=0.25, 0.50 and 0.75
  - $p_j = \alpha \sum_i w_{ij} / m + 500\, q_j$ where q_j drawn in $U(0,1)$

MKP: Instances solved to optimality

Optimal solutions:

n=100, m=5,10,30

'easy' instances

n=250, m=5,10

n=500, m=5

Vimont, Boussier, Vasquez, JOCO (2008)

n=500, m=10

Boussier et al, VI ALIO/EURO, 2008

MKP: Best known solutions

Best known solutions:

n=250, m=30

n=500, m=30

Chu and Beasley, JOC (1998)

Vasquez and Vimont, EJOR (2005)

Very large computational times (days)

MKP: Computational results

average % deviations

from optimal or best known solutions

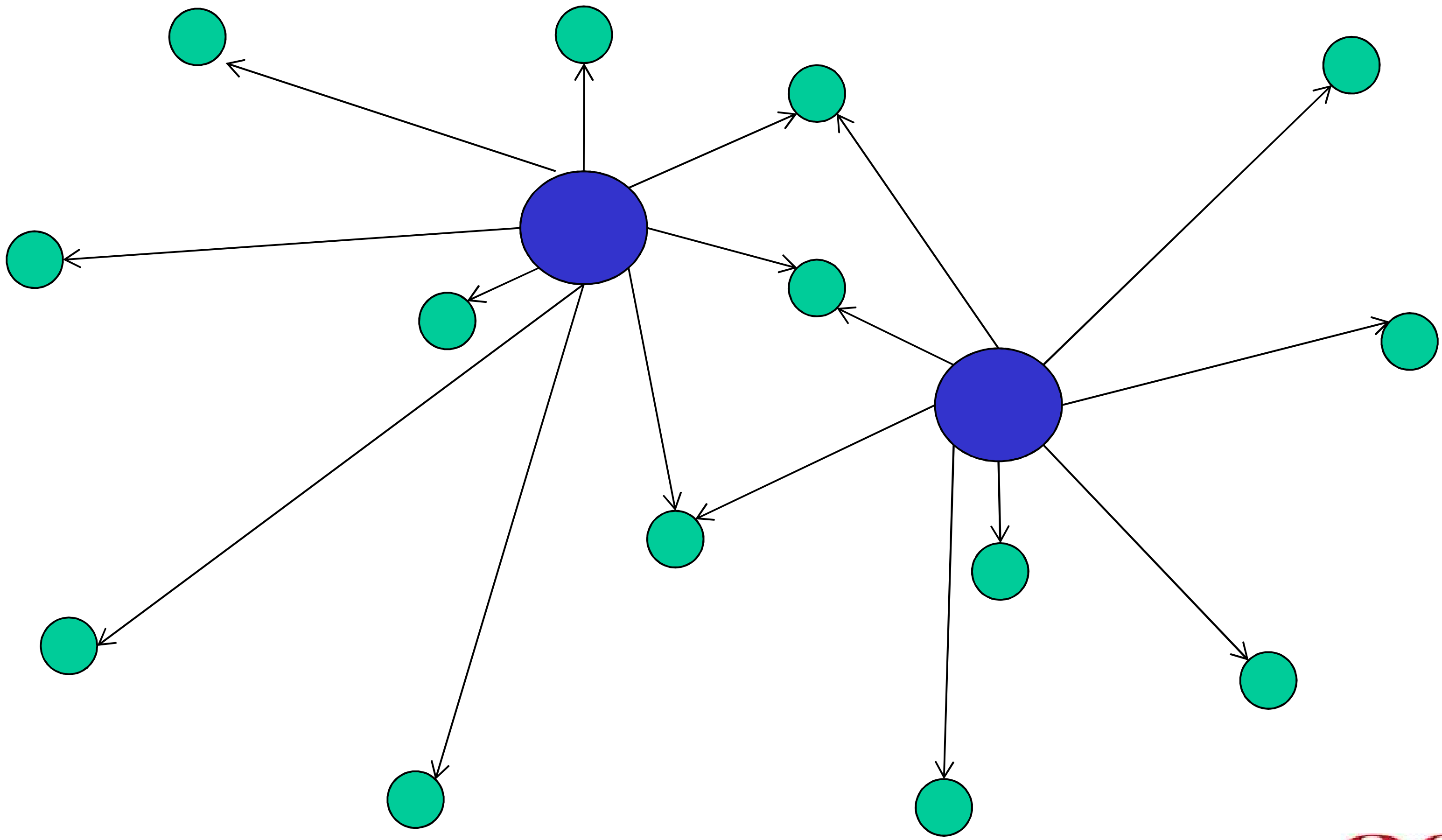
n	m	F-B-I(1)	F-B-I(0.2)	F-B-I(0.1)
250	5	0	0.003	0.008
250	10	0.001	0.003	0.009
250	30	0.013	0.026	0.025
All		**0.005**	**0.011**	**0.014**
500	5	0.002	0.004	0.004
500	10	0.019	0.021	0.020
500	30	0.062	0.062	0.047
All		**0.028**	**0.029**	**0.024**

MKP: Computational results

Number of equal (improved) solutions
with respect to Vasquez and Vimont (2005)

n	m	F-B-I(1)	F-B-I(0.2)	F-B-I(0.1)
250	5	30(12)	28(12)	24(11)
250	10	29(20)	26(20)	26(16)
250	30	26(20)	25(19)	26(20)
500	5	23(0)	19(0)	19(0)
500	10	8(0)	6(0)	5(1)
500	30	0(0)	0(0)	1(0)

Capacitated Facility Location Problem



Capacitated Facility Location Problem

$$\min z = \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} f_j y_j$$

$$s.t. \quad \sum_{i \in I} x_{ij} \le s_j y_j \qquad j \in J$$

$$\sum_{j \in J} x_{ij} = d_i \qquad i \in I$$

$$x_{ij} \le d_i \qquad i \in I, j \in J$$

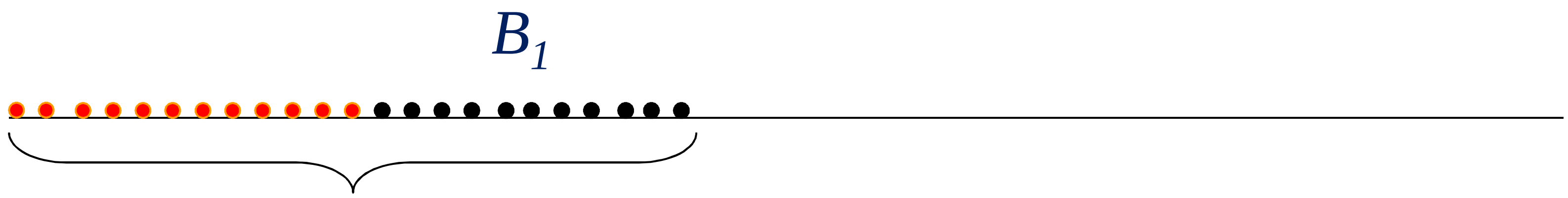
$$x_{ij} \ge 0 \qquad i \in I, j \in J$$

$$y_j \in \{0,1\} \qquad j \in J$$

CFLP: Kernel search

- Kernel includes subsets of x (customers) for selected y (locations)
- A variable y can be removed from the kernel if not selected by p previous MILPs
- Only a subset of buckets is explored

Kernel search – iterative phase

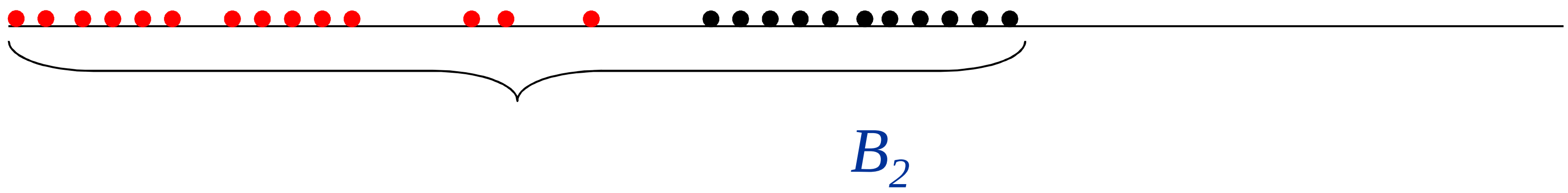


Restricted MILP

Kernel search – iterative phase

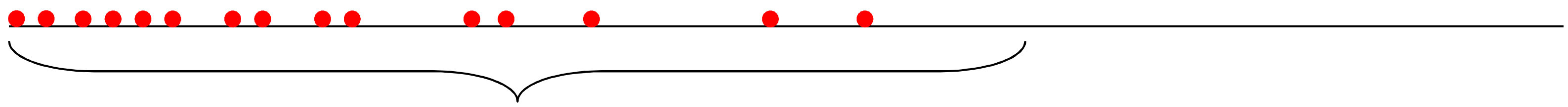
Updated kernel

Kernel search – iterative phase



Restricted MILP

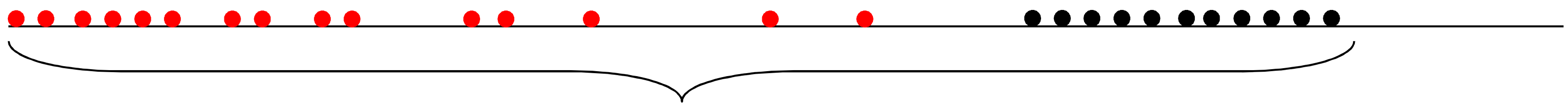
Kernel search – iterative phase



B_2

Updated kernel

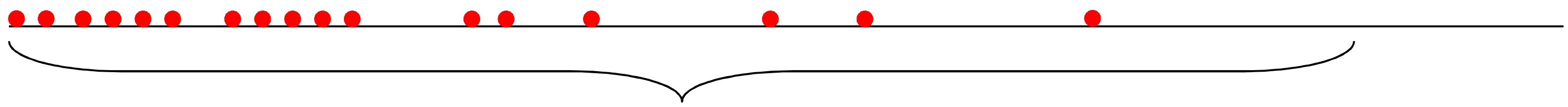
Kernel search – iterative phase



B_3

Restricted MILP

Kernel search – iterative phase



B_3

Updated kernel

CFLP: Instances

49 instances from the OR-library

Optimal solutions are known

100 instances from Avella and Boccia (2009)

Optimal solutions are known for 98 out of 100 instances

295 instances from Avella et al. (2009) Only heuristic solutions

- Test Bed A: 150 instances with fixed costs two orders of magnitude bigger than the other costs
- Test Bed B: 145 instances with fixed costs one order of magnitude bigger than the other costs

150 instances generated as in Avella et al. (2009) with fixed costs

and other costs of the same order of magnitude

(new instances)

Tested instances

Instances		# Inst.	|J|	|I|
OR-Library Beasley (1988)	OR-Library-1	13	16	50
	OR-Library-2	12	25	50
	OR-Library-3	12	50	50
	OR-Library-4	12	100	1000
TBED1 Avella and Boccia (2009)	*TBED1* -1	20	300	300
	TBED1 -2	20	300	1500
	TBED1 -3	20	500	500
	TBED1 -4	20	700	700
	TBED1 -5 *	20	1000	1000
Test Bed A Avella *et al*. (2009)	Test Bed A-1	30	800	4400
	Test Bed A-2	30	1000	1000
	Test Bed A-3	30	1000	4000
	Test Bed A-4	30	1200	3000
	Test Bed A-5	30	2000	2000
Test Bed B Avella *et al*. (2009)	Test Bed B-1	25	800	4400
	Test Bed B-2	30	1000	1000
	Test Bed B-3	30	1000	4000
	Test Bed B-4	30	1200	3000
	Test Bed B-5	30	2000	2000
Test Bed C generated as in Avella *et al*. (2009)	Test Bed C-1	30	800	4400
	Test Bed C-2	30	1000	1000
	Test Bed C-3	30	1000	4000
	Test Bed C-4	30	1200	3000
	Test Bed C-5	30	2000	2000

CFLP: Computational results

Instances	# Inst.	*B-KS* # Opt.	*B-KS* CPU (sec.)
OR-Library-1	13	13	0.3
OR-Library-2	12	12	0.6
OR-Library-3	12	12	0.9
OR-Library-4	12	12	2158.8

Instances	# Inst.	# Opt.	Worst Gap %	CPU (sec.)
TBED1 -1	20	20	0.00%	*57.8*
TBED1 -2	20	20	0.00%	*68.7*
TBED1 -3	20	19	0.02% **	*225.7*
TBED1 -4	20	20	0.00%	*795.8*
TBED1 -5	20	20	0.00%	*1745.9*

**Iterative version found the optimal solution

CFLP: Computational results

Instances	# Inst.	Impr. %	# Impr.	Opt. Gap %	CPU (sec.)
Test Bed A-1	30	-0.22	26	0.33	1349.9
Test Bed A-2	30	-0.13	26	0.1	336.6
Test Bed A-3	29	-0.34	28	0.27	1539.8
Test Bed A-4	29	-0.3	29	0.18	1571.0
Test Bed A-5	29	-0.09	28	0.07	1382.7

		B-KS			
Instances	# Inst.	Impr. %	# Impr.	Opt. Gap %	CPU (sec.)
Test Bed B-1	25	-1.39	25	0.33	1497.2
Test Bed B-2	30	-0.13	27	0.34	1409.5
Test Bed B-3	29	-0.82	29	0.34	1519.9
Test Bed B-4	30	-0.38	30	0.36	1727.2
Test Bed B-5	30	-0.43	27	0.4	2073.4

CFLP: Computational results

Instances	# Inst.	Opt. Gap %	CPU (sec.)
Test Bed C-1	30	1.51	265.9
Test Bed C-2	30	3.57	1358.4
Test Bed C-3	30	2.09	465.6
Test Bed C-4	30	2.94	1001.2
Test Bed C-5	30	4.65	1833.2

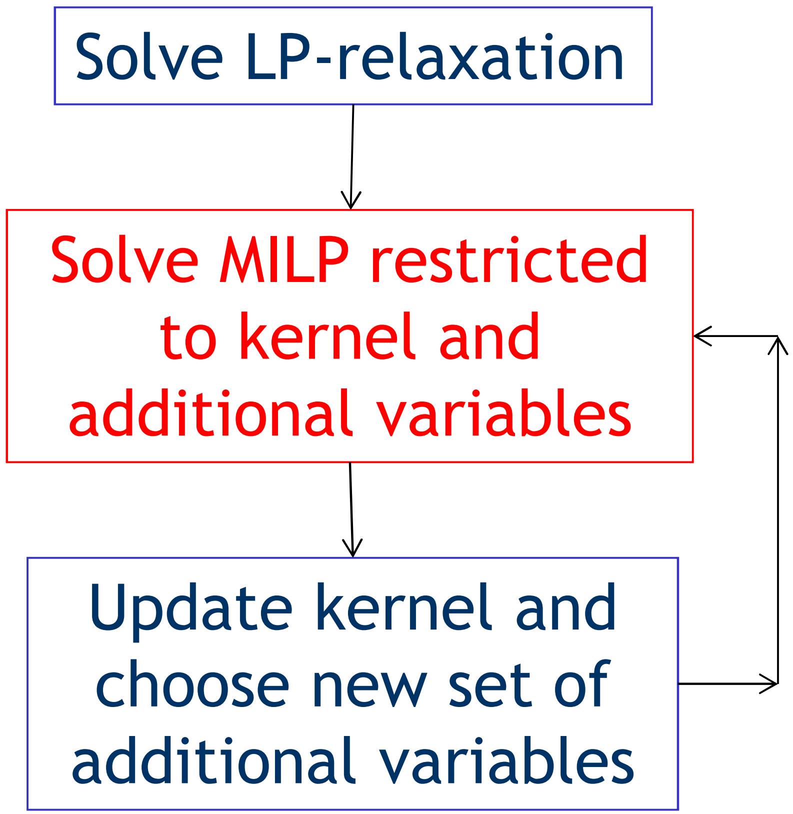
CFLP: A summary

- B-KS found the optimal solution 146 times out of 147
- B-KS improved best known solution for 275 instances out of 293
- Improvements: on average 0.425%, max 5.07%
- The few errors are very small (max 0.46%)

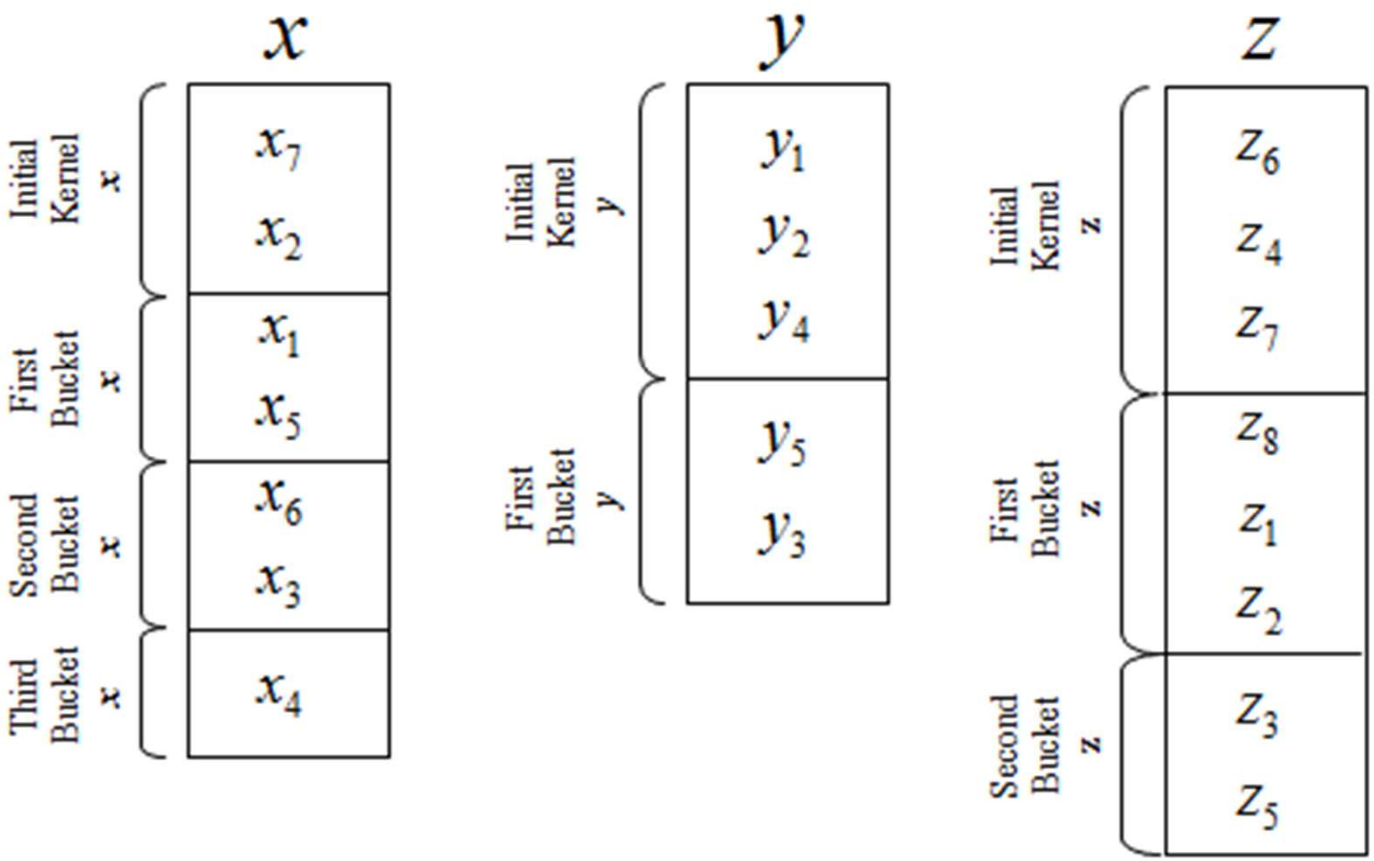
Binary Linear Programming (BILP)

$$\max\ ax + by + cz$$
$$Ax + By + Cz \leq b$$
$$x_i \in \{0, 1\} \quad i = 1, \ldots, n_x$$
$$y_j \in \{0, 1\} \quad j = 1, \ldots, n_y$$
$$z_k \in \{0, 1\} \quad k = 1, \ldots, n_k$$

BILP: Kernel search



Individual buckets



BILP: Kernel search

Current kernel

=

Previous current kernel

+

New promising variables

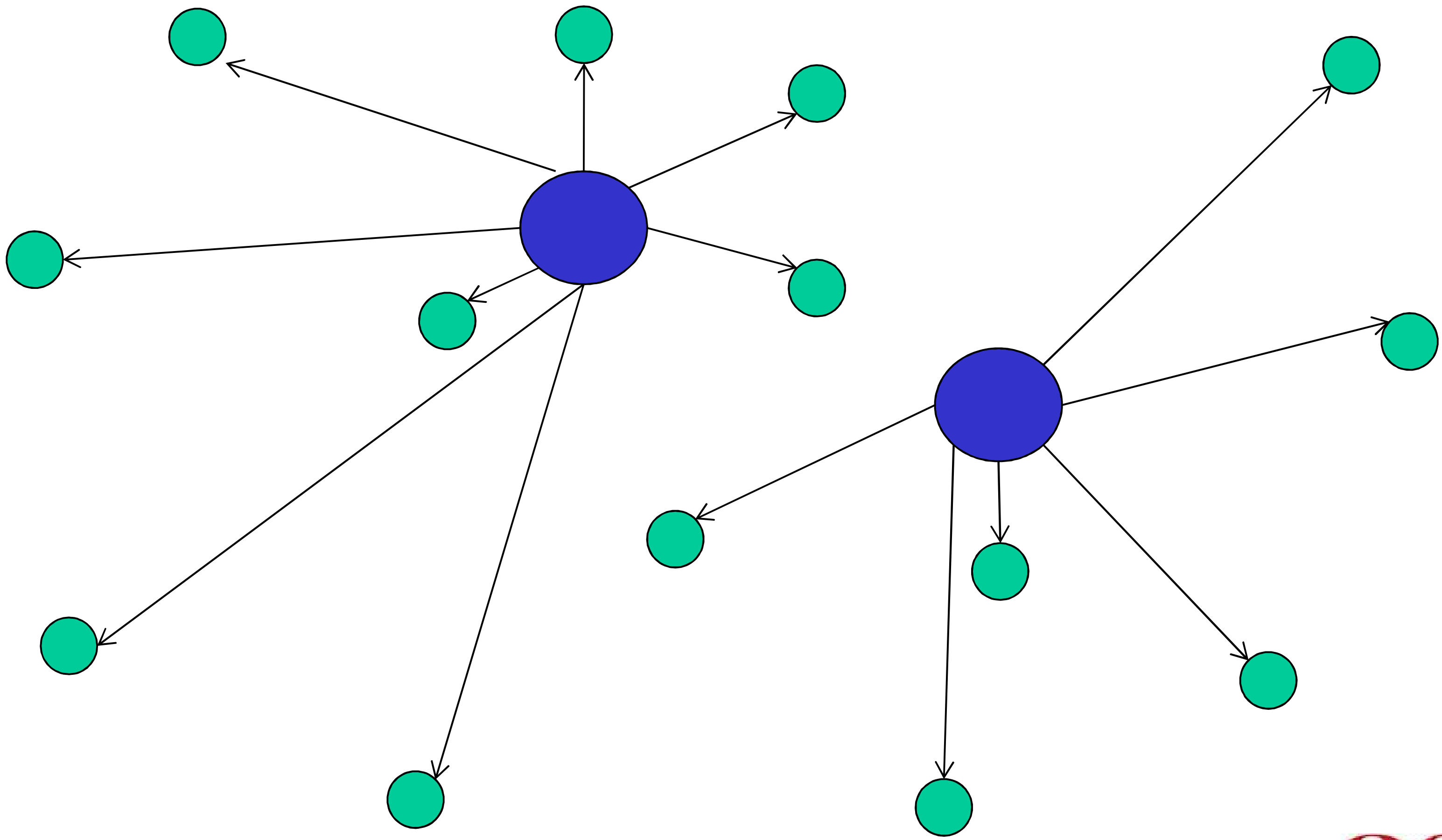
-

No longer promising variables

Hard variable fixing

- Kernel search (0-1)
  - Variables with value 1 in the LP are fixed to 1
  - Variables with value 0 in the LP are fixed to 0

Single Source CPLP



Single source CFLP

$$\min\ z = \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} f_j y_j$$

$$s.t. \quad \sum_{i\in I} d_i x_{ij} \le s_j y_j \qquad j\in J$$

$$\sum_{j\in J} x_{ij} = 1 \qquad i\in I$$

$$x_{ij} \le y_j \qquad i\in I,\ j\in J$$

$$x_{ij} \in \{0,1\} \qquad i\in I,\ j\in J$$

$$y_j \in \{0,1\} \qquad j\in J$$

Single source CFLP: Instances

		# instances	|J|	|I|	
OR-library	OR1 OR2 OR3	8 8 8	16 25 50	50 50 50	CPLEX12.2
OR-library	OR4	12	100	1000	**Heuristic solutions:** Ahuja et al, MS (2004) Chen and Ting, TRE (2008)
Holmberg et al	H1 H2 H3 H4 H5	12 12 16 15 16	10 20 30 10-30 30	50 50 150 70-100 200	**Optimal solutions:** Holmberg et al, EJOR (1999) Yang et al, EJOR (2012) **Heuristic solutions:** Ahuja et al, MS (2004) Chen and Ting, TRE (2008)
Yang et al	Y1 Y2 Y3 Y4	5 5 5 5	30 60 60 80	200 200 300 400	**Optimal solutions:** Yang et al, EJOR (2012)
New	T1 T2 T3 T4 T5	20 20 20 20 20	300 300 500 700 1000	300 1500 500 700 1000	Not available

A preliminary test

			y variables =1 in LP, 0 in ILP	y variables =0 in LP, 1 in ILP
p4	10	50	0	0
p19	20	50	1	1
p32	30	150	0	0
p45	20	80	0	0
p47	10	90	0	0
p49	30	70	3	2
p52	10	100	0	0
p53	20	100	0	0
p66	30	200	0	0
p71	30	200	0	0

Holmberg et al instances

Computational results

		Kernel Search		
			Worst	CPU
Data Set	# Inst.	Gap %	Gap %	(sec.)
OR1	8	0.00	0.00	0.29
OR2	8	0.00	0.00	0.39
OR3	8	0.00	0.00	0.62

CPLEX12.2

Computational results

		Ahuja et al			Chen and Ting			Kernel Search		
Data Set	# Inst.	# Opt.	Gap %	Worst Gap %	# Opt.	Gap %	Worst Gap %	# Opt.	# Impr.	CPU (sec.)
OR4	12	5	0.08%	0.33%	4	0.06%	0.20%	12	7	34.665
H1	12	12	0.00%	0.00%	12	0.00%	0.00%	12	0	0.321
H2	12	12	0.00%	0.00%	12	0.00%	0.00%	12	0	0.379
H3	16	10	0.07%	0.42%	13	0.02%	0.18%	16	3	2.434
H4	15	13	0.01%	0.15%	12	0.06%	0.74%	15	2	0.536
H5	16	11	0.02%	0.14%	11	0.03%	0.19%	16	3	2.319

Optimal solutions for H1-H5:
Holmberg et al, EJOR (1999)
Yang et al, EJOR (2012)

For OR4 optimality is guaranteed by the lower bound

Computational results

		Kernel Search				Kernel Search(0-1)			
Data Set	# Inst.	# Opt.	Gap %	Worst Gap %	CPU (sec.)	# Opt.	Gap %	Worst Gap %	CPU (sec.)
Y1	5	5	0.00%	0.00%	411.282	1	1.02%	2.02%	44.273
Y2	5	4	0.00%	0.01%	1640.424	1	0.67%	2.96%	368.735
Y3	5	4	0.00%	0.01%	597.056	1	1.61%	3.24%	184.533
Y4	5	5	0.00%	0.00%	1409.110	0	0.51%	1.46%	369.761

Optimal solutions:
Yang et al, EJOR (2012)

Computational results

		Kernel Search			Kernel Search(0-1)		
			Worst	CPU		Worst	CPU
Data Set	# Inst.	Gap %	Gap %	(sec.)	Gap %	Gap %	(sec.)
TB1	20	0.76%	2.22%	2206.957	0.98%	2.29%	408.213
TB2	20	0.07%	0.25%	334.705	0.27%	0.75%	186.527
TB3	20	0.76%	2.04%	4190.283	0.89%	2.09%	673.563
TB4	20	0.93%	2.29%	5244.693	1.03%	2.70%	854.165
TB5	20	1.07%	3.11%	6533.149	1.10%	2.67%	968.126

Large instances
Gaps with respect to bound from LP-relaxation

Computational results - CPLEX

- Setting A: MIPEmphasis was set to feasibility
- Setting B: RINS heuristic every 20 nodes
- Setting C: LBHeur is set on (local branching)

On instances OR4:

- KS and CPLEX find optimal solutions
- average time of KS: 35s
- average time of any CPLEX setting: about 110s

On instances TB:

	KS	KS(0-1)	CPLEX A	CPLEX B	CPLEX C
Gap	0.64%	0.78%	1.00%	0.86%	1.19%
Time(s)	3701	618	4698	4575	4767

Conclusions

- Kernel search has been implemented in a straightforward way
- A general heuristic for (classes of) MILP problems is possible
- Ad hoc heuristics would remain valuable, like exact methods remain
- We are working at solving MIPLIB instances