Interdiction Branching

Stefano Smriglio

University of L'Aquila
stefano.smriglio@univaq.it

Andrea Lodi, Ted K. Ralphs, Fabrizio Rossi

ISMP, Berlin - August 2012

Outline

- well-known: branching on a variable dichotomy
- generalization: branching on a variable set
- powerful branching sets: improving solution covers
- minimal covers and bilevel programming
- interdiction branching
- computational experience

Notation

$$\max\{c^\top x \mid Ax \leq b,\ x_j \in \{0,1\}\ \forall j \in I^n\}$$

$$A \in \mathbb{Q}^{m \times n}, c \in \mathbb{Q}^n_+, b \in \mathbb{Q}^m, I^n = \{1, \ldots, n\}$$

- $\mathcal{F}$ set of feasible solutions
- $\bar{x} \in \mathcal{F}$ be the *incumbent* solution
- $\bar{z} = c^\top \bar{x}$ incumbent value

subproblem a:

- F_1^a (F_0^a) indices of variables fixed to one (to zero)
- $N^a = I^n \setminus (F_0^a \cup F_1^a)$ indices of free variables
- $\mathcal{F}^a$ set of the feasible solutions

Branching on a variable (dichotomy)

$\mathbf{x}^*$ current fractional solution, x_j^* fractional

$$x_j \leq \lfloor x_j^* \rfloor \vee x_j \geq \lceil x_j^* \rceil$$

- variable bounds yield fast reoptimization
- "local" objective: improve LP bound at child subproblems

- *strong branching* [Applegate et al. 95]
- cheaper variants (e.g. restrict to variable subsets)
  [Linderoth and Savelsbergh 99]
  [Achterberg, Koch and Martin 04]
  [Achterberg 07][Fischetti and Monaci 10]
- other estimation/bounding methods (e.g. *pseudocosts*, [Bénichou et al 71])

Advanced methods

Well-known difficulties:

variable branching may produce unbalanced trees

choices at the top of the enumeration tree are crucial

many ideas to do better:

- lookahead branching

  [Glankwamdee and Linderoth 11]
- branching on general disjunctions

  [Owen and Mehrotra 01][Karamanov and Cornuéjols 05]

  [Cornuejols, Liberti and Nannicini 09]
- methods based on logical information

  [Achterberg 07] [Karzan, Nemhauser and Savelsbergh 09]

Branching on a *variable set*

Choose an index set $S \subseteq N^a$ as well as an ordering of its elements $(i_1, \ldots, i_{|S|})$.

Partition $\mathcal{F}^a$ into $|S| + 1$ subproblems:

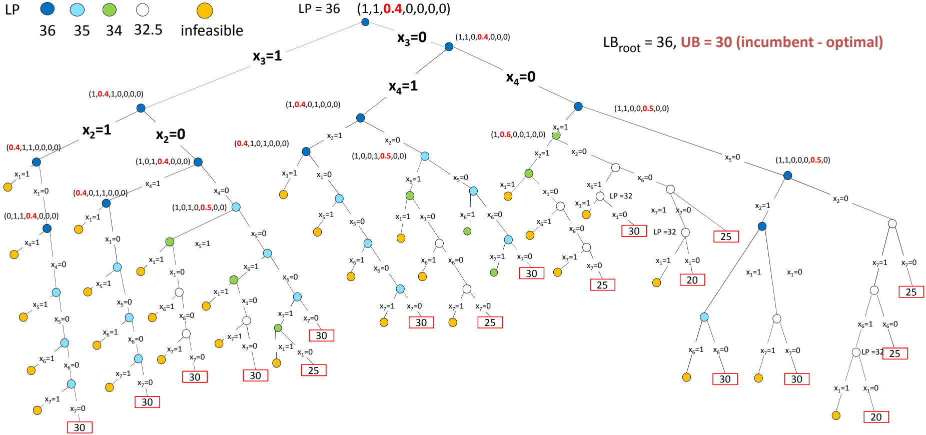
$$
\begin{aligned}
& x_{i_1} = 1 \vee \\
& (x_{i_2} = 1 \wedge x_{i_1} = 0) \vee \\
& \vdots \\
& (x_{i_{|S|}} = 1 \wedge x_{i_1} = 0 \wedge \ldots \wedge x_{i_{|S|-1}} = 0) \vee \\
& \sum_{i \in S} x_i = 0
\end{aligned}
$$

$|S| = 1$ yields branching on variable

What's different?

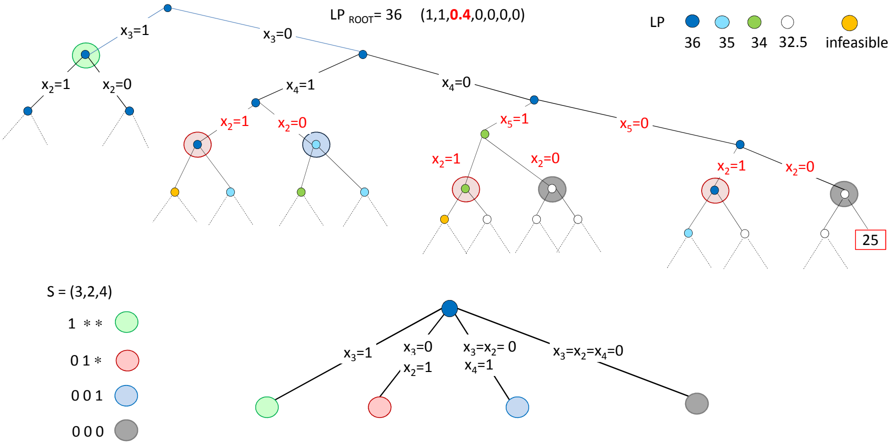
$$\max_{\mathbf{x}\in\{0,1\}^7} \quad 15x_1 + 15x_2 + 15x_3 + 15x_4 + 10x_5 + 10x_6 + 10x_7$$

$$\text{s.t.} \quad 5x_1 + 5x_2 + 5x_3 + 5x_4 + 4x_5 + 4x_6 + 4x_7 \leq 12$$

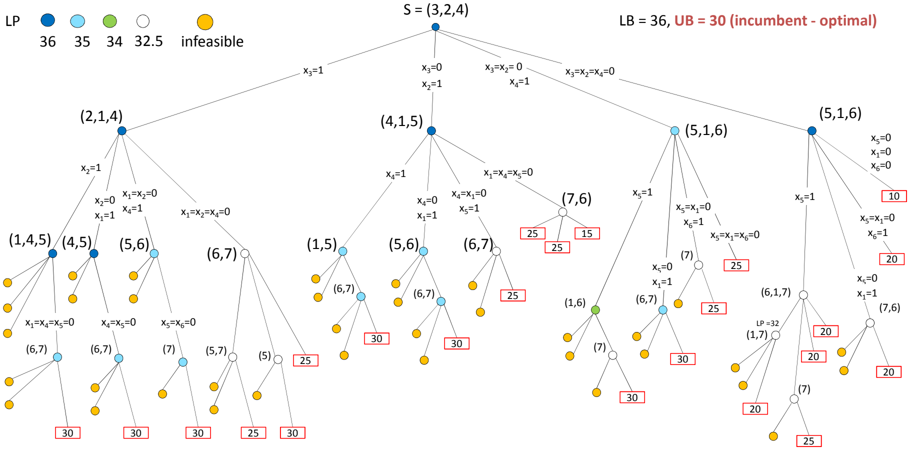


What's different?

Let $S = (3, 2, 4)$ the set of branching variables at the first two levels. Let's branch on set S:



n-ary tree



LP
36
35
34
32.5
infeasible
S = (3,2,4)
LB = 36, UB = 30 (incumbent - optimal)
x3=1
x3=0 x2=1
x3=x2= 0 x4=1
x3=x2=x4=0
(2,1,4)
(4,1,5)
(5,1,6)
(5,1,6)

Any (true) progress by now?

		pruned by		
branch type	# eval. subproblems	infeasibility	optimality	bound
variable	109	34	21	0
set	87	36	23	0

subproblem evaluation: solving LP vs. computing a branching set (several LPs, or estimation method...): overall workload comparable

- n-ary branching cannot be reproduced by a sequence of binary branchings *unless we branch on integer variables!*
- earlier detection of feasible solutions and infeasible subproblems

- new option: look at global objectives and exploit additional information!

The lesson of CO problems

In many $0-1$ MIPs the down branch $x_j = 0$ is weaker than the up branch $x_j = 1$.

(!) branching on a variable set naturally exploits this fact, except that in the rightmost subproblem ($\sum_{i \in S} x_i = 0$)

- Balas and Yu (Stable Set problem, '86): exploit $\bar{z}$ to compute S such that the rightmost subproblem needs not be generated (our starting point)
- Bienstock and Zuckerberg characterized S with strong properties for the Set Covering problem
- Our idea: design branching method for $0-1$ MIPs combining LP information with the knowledge of feasible solutions

Improving solution cover

- Given some value $z \in \Re$, denote $\mathcal{F}^a(z) = \{x \in \mathcal{F}^a : c^T x > z\}$ the set of z-improving solutions at subproblem a
- index $i \in I^n$ is said to *cover* a solution $\hat{x} \in \mathcal{F}$ if $\hat{x}_i = 1$
- index set S covers a set of solutions X if every solution in X is covered by at least one index in S.

Definition

A *z-improving solution cover* (z-ISC) at subproblem a is an index set $S^a(z) = \{i_1, \ldots, i_{|S^a(z)|}\} \subseteq N^a$ covering $\mathcal{F}^a(z)$.

	fixed		free			
	F_0^a	F_1^a	N^a			
	0	1	1	0	0	0
$\mathcal{F}^a(z)$	0	1	0	1	0	1
	0	1	0	0	1	1
	0	1	1	1	1	0
z-ISC			*		*	*

Branching on a z-ISC

Suppose a solution $x \in \mathcal{F}$ is known with value $z = c^T x$. If $S = S^a(z)$ is a z-ISC, the rightmost subproblem is dominated:

$$x_{i_1} = 1 \vee \ldots \vee (x_{i_{|S|}} = 1 \wedge x_{i_1} = 0 \wedge \ldots \wedge x_{i_{|S|-1}} = 0) \boxed{\vee \sum_{i \in S} x_i = 0}$$

Minimal z-ISCs show a very strong property:

Theorem

If S is a minimal z-ISC, then each term of the disjunction is satisfied by at least one improving solution $\tilde{x} \in \mathcal{F}^a(z)$ □

Characterization of ISCs

Theorem

A nonempty index set $S \subseteq N^a$ *is a* z*-ISC at* a *if and only if*

$$\max_{x \in \{0,1\}^n} \{c^\top x \mid x \in \mathcal{F}^a, x_i = 0 \text{ for all } i \in S\} \leq z.$$

□

Corollary

Let $\tilde{x} \in \mathcal{F}^a$*,* $\tilde{z} = c^T \tilde{x}$*, and* $F_0(\tilde{x}) = \{i \in N^a : \tilde{x} = 0\}$ *be the set of free variables fixed to zero at* $\tilde{x}$*. Then,* $F_0(\tilde{x})$ *is a* $\tilde{z}$*-ISC at* a*.* □

Any feasible solution to subproblem a yields a "nice" (set) branching disjunction

Interdiction branching problem (IBP)

Look now at the incumbent value $\bar{z}$. A smallest $\bar{z}$-ISC can be computed by a (binary-binary) bilevel program.

Define binary "deactivation" variables:
$y_i = 1$ if index i is in the cover and $y_i = 0$ otherwise.

$$z_{IBP} = \min_{\mathbf{y} \in \{0,1\}^{|N^a|}} \sum_{i \in N^a} y_i$$

$$\sum_{i \in N^a} c_i x_i \leq \bar{z} - \sum_{i \in F_1^a} c_i \qquad \text{obj bound}$$

$$x \in \arg\max_{\mathbf{x} \in \mathcal{F}^{\mathbf{a}}} c^\top x$$

$$x_i + y_i \leq 1, \; i \in N^a \qquad \text{interdiction const.}$$

Interdiction branching: general scheme

Input:	Subproblem $a = (F_1^a, F_0^a)$, incumbent value $\bar{z}$.
Output:	A set of child subproblems
STEP 1.	Solve IBP $\rightarrow S^a(\bar{z})$
STEP 2.	**if** $z_{IBP} = 0$ ($\Rightarrow \mathcal{F}^a(\bar{z}) = \emptyset$), **then** prune a. STOP **else** choose an ordering $(i_1, \ldots, i_{\|S^a(\bar{z})\|})$ of $S^a(\bar{z})$ branch on $(i_1, \ldots, i_{\|S^a(\bar{z})\|})$

Theorem

The number of subproblems generated by interdiction branching is at most $\max(0, 2|\mathcal{F}^0(\bar{z})| - 1)$.

Solving IBP exactly is too demanding, but any relaxation of the second-level problem still yields an $\bar{z}$-ISC

Back to the LP relaxation

Theorem

A nonempty index set $S \subseteq N^a$ is a $\bar{z}$-ISC at a **if**

$$\max\{c^\top x \mid x \in \mathcal{F}^{\mathbf{a}}_{\mathbf{LP}}, x_i = 0 \textit{ for all } i \in S\} \leq \bar{z}$$

□

Relax lower-level integrality: still get a $\bar{z}$-ISC (but loose minimality)

$$z_{IBLP} = \min_{\mathbf{y} \in \{\mathbf{0},\mathbf{1}\}^{|\mathbf{N^a}|}} \sum_{i \in N^a} y_i$$
$$\sum_{i \in N^a} c_i x_i \leq \bar{z} - \sum_{i \in F_1^a} c_i$$
$$x \in \arg\max_{\mathbf{x} \in \mathcal{F}^{\mathbf{a}}_{\mathbf{LP}}} c^\top x$$
$$x_i + y_i \leq 1, \; i \in N^a$$

Algorithms for IBLP

- MIP (big-M)reformulation:

$$\begin{aligned} \min \sum_{i \in I^n} y_i & \\ b^\top u + \mathbf{1}^\top w \leq \bar{z} - \sum_{i \in F_1^a} c_i & \\ u^\top A_i + w_i \geq c_i - M y_i & \qquad i \in N^a \\ u, w \geq 0, \quad y \in \{0,1\}^n & \end{aligned}$$

- (straightforward) heuristic:

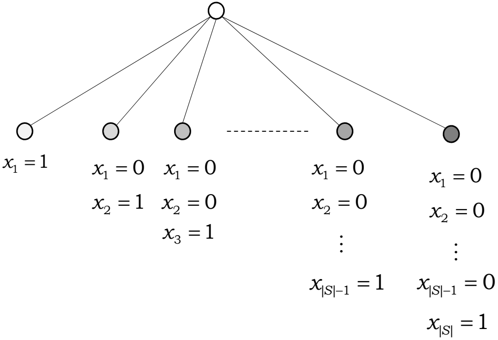
Initialize $S := \emptyset$
while $(\widetilde{z} := \{\max c^\top x : x \in \mathcal{F}_{LP}, x_j = 0, j \in S\} > \bar{z})$
if $\widetilde{x}$ is integer **then** update incumbent - STOP
else find fractional $\widetilde{x}_j$ with max reduced cost;
$S := S \cup \{j\}$
Return S

On the role of LP relaxation

if LP bound $\leq \bar{z}$ we have $z_{IBLP} = 0$ and the node is pruned

otherwise the LP information is still exploited:

- **to choose the branching set**
- **to rank the branching variables**: non-increasing pseudocosts yield a more balanced tree, as less constrained children have top variables fixed to 1



Computational experience

Two CO problems: **knapsack**, **stable set**

in both cases weak LP relaxations

- MIP Solver: CPLEX12.2

  `CPX_PARAM_MIPSEARCH=CPX_MIPSEARCH_TRADITIONAL`:
  traditional branch-and-cut search
  `CPX_PARAM_THREADS=1`: sequential mode
- Computer: 2.8 GHz Intel Quad Core - 24 GB RAM - Linux
- CPU time limit = 2 hours
- initial incumbent computed by Cplex at the root node

Experiment I: Hard knapsack problems

- Purposes:
  - compare 0-1 branching to IB
  - investigate the effect of LP strengthening
- test-bed: *strongly correlated spanner instances* (Pisinger 05)
  - *spanner* set (v, m) of v items with weights in $[1, R]$ and profits $p_j = w_j + R/10$ normalized by $m + 1$.
  - n items generated by picking k in (v, m) and setting $(d \cdot p_k, d \cdot w_k)$, $d \in [1, m]$
- subproblem evaluation:
  - IBLP solved by HEU - `CPX_PARAM_PRESLVND=-1`: no node presolve to avoid interference
  - branching set ranking: nondecreasing item weights

Hard knapsack problems

$m = 2$ - 20 instances/row

n	Cplex cuts OFF 0–1		Cplex cuts OFF IB		Cplex cuts ON 0–1		Cplex cuts ON IB		%Δ(time/#sub) 0–1	%Δ(time/#sub) IB
50	(20)	62.98	(20)	53.33	(20)	39.41	(20)	28.83	37.4	45.9
		1,761,896.95		602,378.60		1,257,841.45		360,106.85	28.6	40.2
60	(19)	421.00	(19)	388.93	(19)	252.77	(19)	197.75	40.0	49.2
		8,629,773.32		4,155,480.05		6,654,547.21		2,377,041.68	22.9	42.8
70	(15)	76.11	(17)	24.11	(16)	531.98	(17)	367.60	-599.0	-1424.7
		1,974,794.93		229,331.47		16,104,171.69		4,321,404.50	-715.5	-1784.3
80	(13)	422.47	(14)	312.62	(13)	154.85	(14)	125.08	63.3	60.0
		11,081,012.15		3,142,002.54		4,552,826.54		1,395,661.54	58.9	55.6
90	(9)	5.38	(9)	1.60	(9)	5.38	(9)	1.60	0.0	0.0
		129,098.56		12,118.44		129,098.56		12,118.44	0.0	0.0
100	(11)	113.00	(11)	20.00	(11)	113.00	(11)	20.00	0.0	0.0
		2,668,660.45		168,958.91		2,668,660.45		168,958.91	0.0	0.0

(# solved)	av. CPU sec. (solved)
	# av. subproblems (solved)

Hard knapsack problems

$m = 3$ - 20 instances/row

	Cplex cuts OFF				Cplex cuts ON				%Δ(time/#sub)	
n	0–1		IB		0–1		IB		0–1	IB
50	(20)	461.72	(20)	400.03	(19)	118.65	(20)	77.49	74.3	80.6
	18,136,555.70		6,064,691.55		3,705,276.47		971,255.95		79.6	84.0
60	(19)	379.79	(19)	169.37	(19)	403.16	(19)	159.47	-6.2	5.8
	14,390,370.89		2,408,517.37		12,056,479.58		1,897,376.26		16.2	21.2
70	(17)	688.23	(17)	300.94	(17)	678.88	(17)	288.35	1.4	4.2
	25,163,713.53		3,969,008.53		20,126,021.82		3,247,175.41		20.0	18.2
80	(17)	137.04	(18)	77.41	(17)	308.04	(18)	89.41	-124.8	-15.5
	4,972,139.94		968,274.35		9,309,853.12		966,178.47		-87.2	0.2
90	(12)	0.36	(13)	0.53	(12)	0.30	(12)	0.30	16.7	43.4
	5,770.17		3,737.83		1,131.08		676.33		80.4	81.9
100	(13)	2.79	(15)	2.64	(13)	3.95	(13)	1.64	-41.6	37.9
	88,137.38		28,888.23		100,859.31		14,329.77		-14.4	50.4

(# solved)	av. CPU sec. (solved)
	# av. subproblems (solved)

Experiment II: Stable set

Unstructured SS problems represent difficult IP-s, hard to be solved by IP algorithms; combinatorial algorithms (based on the Balas-Yu branching rule) often performs better.

- Purposes:
  - compare IB to 0-1 branching
  - compare IB to state-of-the-art combinatorial algorithms
  - investigate the effect of branching set ranking
- test-bed:

  DIMACS challenge benachmark graphs
  - formulation based on *clique inequalities* [Rossi, S. 01]
- subproblem evaluation:
  - IBLP solved by Cplex MIP heuristic
  - ranking by non-increasing vertex degree

DIMACS benchmark stable set problems

Graph name	0-1 branching Time (sec)	Sub (#)	Time/sub (msec)	Interdiction Branching Time (sec)	IBLP (sec)	Sub (#)	Time/sub (msec)
brock200_1	1,551.96	370,372	4.19	1,028.37	547.91	57,603	9.81
brock200_2	81.50	11,769	6.92	63.20	21.41	3,441	9.69
brock200_3	213.70	33,142	6.45	130.35	55.03	5,973	12.22
brock200_4	475.79	104,572	4.55	308.49	136.27	19,150	8.79
C125.9	5.19	6,093	0.85	33.61	28.25	1,754	10.95
c-fat200-5	6.30	47	134.04	8.02	0.45	27	157.25
DSJC125.1	3.44	4,259	0.81	40.13	36.35	1,505	14.37
DSJC125.5	4.34	1,307	3.32	4.88	2.32	417	6.42
mann_a9	0.01	5	2.00	0.02	0.01	3	6.67
mann_a27	3.32	10,755	0.31	44.82	42.67	1,176	22.69
keller4	16.91	6,429	2.63	18.44	13.10	955	10.84
p_hat300-1	56.05	5,198	10.78	43.25	14.00	2,674	8.52
p_hat300-2	167.59	8,576	19.54	117.24	51.03	2,064	29.88
p_hat300-3	22,449.20	1,104,172	20.33	6,307.62	2937.63	116,737	29.79
san200_0.7_2	4.42	822	5.38	4.00	1.85	176	12.46
san200_0.9_3	2.88	595	4.84	13.49	9.73	744	10.82
sanr200_0.7	1,067.42	203,077	5.26	626.92	307.15	36,429	9.43
sanr200_0.9	6,223.24	2,054,850	3.03	5,318.50	4,295.58	203,835	15.13

Further experimental findings

- **vertex-deg vs. pseudocosts**. Ranking $|S|$ by non-increasing pseudocosts slightly worsen the results but don't change the overall judgement
- **robustness w.r.t. node selection**. IB looks much less sensitive to different node selection strategies. DFS performs poorly with $0-1$ branching, while is competitive with IB
- **IB vs. combinatorial algorithms**. In some cases # evaluated subproblems much smaller than [Mannino and Sassano 96], [Sewell 96]. IB takes much longer CPU times.
- **IB vs. specialized branch-and-cut algorithms** Competitive with [Rossi, S. 01], [Rebennack et al. 09] both in the number of evaluated subproblems and CPU times.

Conclusions

Promising features of IB:

- Robust to weak LP relaxations
- Robust to different MIP strategies
- Flexible in choosing the branching disjunction
- Customizable by additional knowledge of the problem

Work in progress:

- Specialized algorithms for IBP, IBLP
- Experiments on the MIPLIB
- Improved implementation