Computation in MIP

Andrea Lodi
University of Bologna, Italy
andrea.lodi@unibo.it

January 11, 2008 (50 A.G.) @ 12^{th} Aussois Combinatorial Optimization Workshop

Outline, Assumptions and Notation

- We consider a general Mixed Integer Program in the form:

$$\min\{c^T x : Ax \geq b, x \geq 0,\ x_j \text{ integer},\ j \in \mathcal{I}\} \tag{1}$$

where matrix A does not have a special structure.

Outline, Assumptions and Notation

- We consider a general Mixed Integer Program in the form:

$$\min\{c^T x : Ax \geq b, x \geq 0,\ x_j \text{ integer},\ j \in \mathcal{I}\} \tag{1}$$

  where matrix A does not have a special structure.
- Thus, the problem is solved through branch-and-bound and the bounds are computed by iteratively solving the LP relaxations through a general-purpose LP solver.
  (Unfortunately, the talk does not cover LP advances.)

Outline, Assumptions and Notation

- We consider a general Mixed Integer Program in the form:

$$\min\{c^T x : Ax \geq b, x \geq 0,\ x_j \text{ integer},\ j \in \mathcal{I}\} \tag{1}$$

  where matrix A does not have a special structure.
- Thus, the problem is solved through branch-and-bound and the bounds are computed by iteratively solving the LP relaxations through a general-purpose LP solver.
  (Unfortunately, the talk does not cover LP advances.)
- The implicit question the talk tries to answer is:

  what does a general-purpose MIP solver contain/do?

Outline, Assumptions and Notation

- We consider a general Mixed Integer Program in the form:

$$\min\{c^T x : Ax \geq b, x \geq 0, \ x_j \text{ integer}, \ j \in \mathcal{I}\} \tag{1}$$

  where matrix A does not have a special structure.

- Thus, the problem is solved through branch-and-bound and the bounds are computed by iteratively solving the LP relaxations through a general-purpose LP solver.
  (Unfortunately, the talk does not cover LP advances.)
- The implicit question the talk tries to answer is:

  what does a general-purpose MIP solver contain/do?

- The talk is DIS-organized as follows:

Outline, Assumptions and Notation

- We consider a general Mixed Integer Program in the form:

$$\min\{c^T x : Ax \geq b, x \geq 0, \; x_j \text{ integer}, \; j \in \mathcal{I}\} \tag{1}$$

  where matrix A does not have a special structure.
- Thus, the problem is solved through branch-and-bound and the bounds are computed by iteratively solving the LP relaxations through a general-purpose LP solver.
  (Unfortunately, the talk does not cover LP advances.)
- The implicit question the talk tries to answer is:

  what does a general-purpose MIP solver contain/do?

- The talk is DIS-organized as follows:
  1. MIP solvers, Evolution:
     (a) A performance perspective
     (b) An application perspective

Outline, Assumptions and Notation

- We consider a general Mixed Integer Program in the form:

$$\min\{c^T x : Ax \geq b, x \geq 0,\ x_j \text{ integer},\ j \in \mathcal{I}\} \tag{1}$$

  where matrix A does not have a special structure.
- Thus, the problem is solved through branch-and-bound and the bounds are computed by iteratively solving the LP relaxations through a general-purpose LP solver. (Unfortunately, the talk does not cover LP advances.)
- The implicit question the talk tries to answer is:

  what does a general-purpose MIP solver contain/do?

- The talk is DIS-organized as follows:
  1. MIP solvers, Evolution:
     (a) A performance perspective
     (b) An application perspective
  2. MIP solvers, Challenges:
     (a) A performance perspective
     (b) An application perspective

MIP Evolution, early days

- Despite quite some work on basically all aspects of IP and in particular on cutting planes, the early days of general-purpose MIP solvers were mainly devoted to develop fast and reliable LP solvers used within good branch-and-bound schemes.

MIP Evolution, early days

- Despite quite some work on basically all aspects of IP and in particular on cutting planes, the early days of general-purpose MIP solvers were mainly devoted to develop fast and reliable LP solvers used within good branch-and-bound schemes.

- Remarkable exceptions are:
  - 1983 Crowder, Johnson & Padberg: PIPX, pure 0/1 MIPs

  - 1987 Van Roy & Wolsey: MPSARX, mixed 0/1 MIPs

MIP Evolution, early days

- Despite quite some work on basically all aspects of IP and in particular on cutting planes, the early days of general-purpose MIP solvers were mainly devoted to develop fast and reliable LP solvers used within good branch-and-bound schemes.
- Remarkable exceptions are:
  - 1983 Crowder, Johnson & Padberg: PIPX, pure 0/1 MIPs
  - 1987 Van Roy & Wolsey: MPSARX, mixed 0/1 MIPs
- When do the early days end?
  Or equivalently, when does the current generation of MIP solvers begin?

MIP Evolution, early days

- Despite quite some work on basically all aspects of IP and in particular on cutting planes, the early days of general-purpose MIP solvers were mainly devoted to develop fast and reliable LP solvers used within good branch-and-bound schemes.
- Remarkable exceptions are:
  - 1983 Crowder, Johnson & Padberg: PIPX, pure 0/1 MIPs
  - 1987 Van Roy & Wolsey: MPSARX, mixed 0/1 MIPs
- When do the early days end?
  Or equivalently, when does the current generation of MIP solvers begin?
- It looks like a major (crucial) step to get to nowadays MIP solvers has been the ultimate proof that cutting planes generation in conjunction with branching could work in general, i.e., after the success in the TSP context:
  - 1994 Balas, Ceria & Cornuéjols: lift-and-project
  - 1996 Balas, Ceria, Cornuéjols & Natraj: gomory cuts revisited

MIP Evolution, Cplex numbers

- Bob Bixby & Tobias Achterberg performed the following interesting experiments which compare all Cplex versions starting from Cplex 1.2, the first one having MIP capability.

MIP Evolution, Cplex numbers

- Bob Bixby & Tobias Achterberg performed the following interesting experiments which compare all Cplex versions starting from Cplex 1.2, the first one having MIP capability.
- 1,734 MIP instances, time limit of 30,000 CPU seconds, computing times as geometric means normalized wrt Cplex 11.0 (equivalent if within 10%).

Cplex versions	year	better	worse	time
11.0	2007	0	0	1.00
10.0	2005	201	650	1.91
9.0	2003	142	793	2.73
8.0	2002	117	856	3.56
7.1	2001	63	930	4.59
6.5	1999	71	997	7.47
6.0	1998	55	1060	21.30
5.0	1997	45	1069	22.57
4.0	1995	37	1089	26.29
3.0	1994	34	1107	34.63
2.1	1993	13	1137	56.16
1.2	1991	17	1132	67.90

MIP Evolution, Cplex numbers (cont.d)

- On a slightly larger set of 1,852 MIPs (including some models in which older versions encountered numerical troubles), the experiment highlights the version-to-version improvement in the number of solved problems.

MIP Evolution, Cplex numbers (cont.d)

- On a slightly larger set of 1,852 MIPs (including some models in which older versions encountered numerical troubles), the experiment highlights the version-to-version improvement in the number of solved problems.

Cplex versions	year	# optimal	% optimal	v-to-v % improvement
11.0	2007	1,243	67.1%	7.8%
10.0	2005	1,099	59.3%	3.5%
9.0	2003	1,035	55.9%	2.6%
8.0	2002	987	53.3%	2.5%
7.1	2001	941	50.8%	4.3%
6.5	1999	861	46.5%	13.4%
6.0	1998	613	33.1%	1.0%
5.0	1997	595	32.1%	1.8%
4.0	1995	561	30.3%	4.4%
3.0	1994	479	25.9%	6.2%
2.1	1993	365	19.7%	4.7%
1.2	1991	278	15.0%	—

MIP Evolution, Cplex numbers (cont.d)

- On a slightly larger set of 1,852 MIPs (including some models in which older versions encountered numerical troubles), the experiment highlights the version-to-version improvement in the number of solved problems.

Cplex versions	year	# optimal	% optimal	v-to-v % improvement
11.0	2007	1,243	67.1%	7.8%
10.0	2005	1,099	59.3%	3.5%
9.0	2003	1,035	55.9%	2.6%
8.0	2002	987	53.3%	2.5%
7.1	2001	941	50.8%	4.3%
6.5	1999	861	46.5%	13.4%
6.0	1998	613	33.1%	1.0%
5.0	1997	595	32.1%	1.8%
4.0	1995	561	30.3%	4.4%
3.0	1994	479	25.9%	6.2%
2.1	1993	365	19.7%	4.7%
1.2	1991	278	15.0%	—

- Does anybody remember which was the key feature of Cplex v. 6.5?

MIP Evolution, nowadays key features

- The current generation of MIP solvers incorporates key ideas developed continuously during the first 50 years of Integer Programming:

MIP Evolution, nowadays key features

- The current generation of MIP solvers incorporates key ideas developed continuously during the first 50 years of Integer Programming:
  - Cutting plane generation:
    Gomory Mixed Integer cuts, Mixed Integer Rounding, cover cuts, flow covers, 0/1 cuts, . . .

MIP Evolution, nowadays key features

- The current generation of MIP solvers incorporates key ideas developed continuously during the first 50 years of Integer Programming:
  - Cutting plane generation:
    Gomory Mixed Integer cuts, Mixed Integer Rounding, cover cuts, flow covers, $0/1$ cuts, . . .
  - Sophisticated branching strategies:
    strong branching, pseudo-cost branching, diving and hybrids

MIP Evolution, nowadays key features

- The current generation of MIP solvers incorporates key ideas developed continuously during the first 50 years of Integer Programming:
  - Cutting plane generation:
    Gomory Mixed Integer cuts, Mixed Integer Rounding, cover cuts, flow covers, 0/1 cuts, . . .
  - Sophisticated branching strategies:
    strong branching, pseudo-cost branching, diving and hybrids
  - Primal heuristics:
    rounding heuristics (from easy to complex), local search, . . .

MIP Evolution, nowadays key features

- The current generation of MIP solvers incorporates key ideas developed continuously during the first 50 years of Integer Programming:
  - Cutting plane generation:
    Gomory Mixed Integer cuts, Mixed Integer Rounding, cover cuts, flow covers, 0/1 cuts, . . .
  - Sophisticated branching strategies:
    strong branching, pseudo-cost branching, diving and hybrids
  - Primal heuristics:
    rounding heuristics (from easy to complex), local search, . . .
  - Preprocessing:
    probing, bound strengthening, propagation

MIP Evolution, nowadays key features

- The current generation of MIP solvers incorporates key ideas developed continuously during the first 50 years of Integer Programming:

  - Cutting plane generation:
    Gomory Mixed Integer cuts, Mixed Integer Rounding, cover cuts, flow covers, 0/1 cuts, . . .

  - Sophisticated branching strategies:
    strong branching, pseudo-cost branching, diving and hybrids

  - Primal heuristics:
    rounding heuristics (from easy to complex), local search, . . .

  - Preprocessing:
    probing, bound strengthening, propagation

- Moreover, the MIP computation has reached such an effective and stable quality to allow the solution of sub-MIPs in the algorithmic process, the MIPping approach. [Fischetti & Lodi]
  These sub-MIPs are solved both for cutting plane generation and in the primal heuristic context.

MIP Evolution, an application viewpoint

- Solving a MIP to optimality is only one aspect of using MIP solver for applications, sometimes not the most important one.

MIP Evolution, an application viewpoint

- Solving a MIP to optimality is only one aspect of using MIP solver for applications, sometimes not the most important one.
  Nowadays MIP solvers include useful tools for complex algorithmic design and data and model analysis. Some of them are:

MIP Evolution, an application viewpoint

- Solving a MIP to optimality is only one aspect of using MIP solver for applications, sometimes not the most important one.
  Nowadays MIP solvers include useful tools for complex algorithmic design and data and model analysis. Some of them are:

  - automatic tuning of the parameters:
    the number of parameters (corresponding to different algorithmic options) makes the hand-tuning complex but it guarantees great flexibility

MIP Evolution, an application viewpoint

- Solving a MIP to optimality is only one aspect of using MIP solver for applications, sometimes not the most important one.
  Nowadays MIP solvers include useful tools for complex algorithmic design and data and model analysis. Some of them are:
  - automatic tuning of the parameters:
    the number of parameters (corresponding to different algorithmic options) makes the hand-tuning complex but it guarantees great flexibility
  - multiple solutions:
    allow flexibility and support for decision making and, as side effect, improve primal heuristics

MIP Evolution, an application viewpoint

- Solving a MIP to optimality is only one aspect of using MIP solver for applications, sometimes not the most important one.
  Nowadays MIP solvers include useful tools for complex algorithmic design and data and model analysis. Some of them are:

  - automatic tuning of the parameters:
    the number of parameters (corresponding to different algorithmic options) makes the hand-tuning complex but it guarantees great flexibility

  - multiple solutions:
    allow flexibility and support for decision making and, as side effect, improve primal heuristics

  - detection of sources of infeasibility in the models:
    real-world models are often over constrained and sources of infeasibility must be removed
    [Amaldi; Chinneck]

MIP Evolution, an application viewpoint

- Solving a MIP to optimality is only one aspect of using MIP solver for applications, sometimes not the most important one.
  Nowadays MIP solvers include useful tools for complex algorithmic design and data and model analysis. Some of them are:
  - automatic tuning of the parameters:
    the number of parameters (corresponding to different algorithmic options) makes the hand-tuning complex but it guarantees great flexibility
  - multiple solutions:
    allow flexibility and support for decision making and, as side effect, improve primal heuristics
  - detection of sources of infeasibility in the models:
    real-world models are often over constrained and sources of infeasibility must be removed
    [Amaldi; Chinneck]
  - callbacks:
    allow flexibility to accommodate the user code so as to take advantage of specific knowledge

MIP Challenges

- Some difficult MIPs:
  - bad modeling:
    * the model has numerical difficulties
    * the MIP modeling capability is not sufficient wrt the real problem
  - large problems
  - knapsack constraints with huge coefficients and general integer variables with large bounds
  - scheduling models with disjunctive constraints and fundamental continuous variables

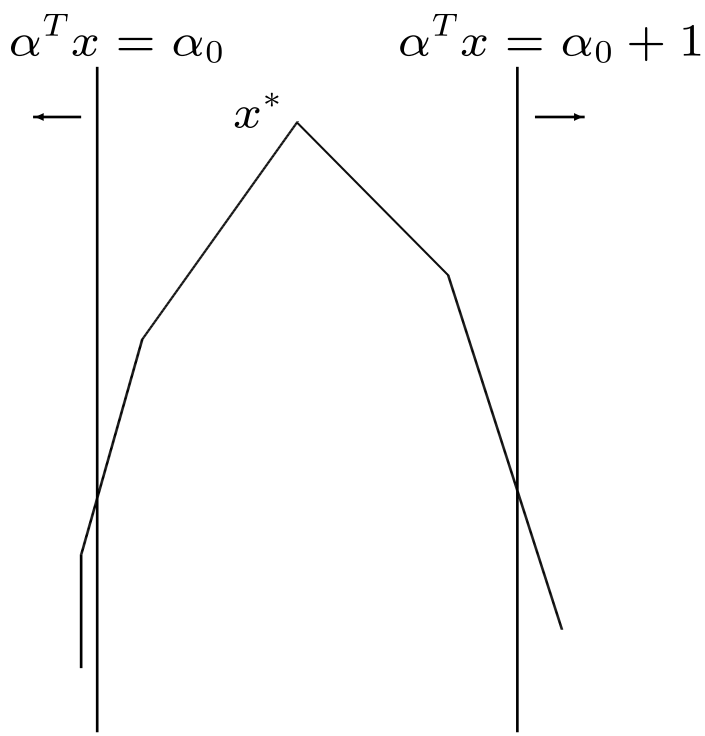
MIP Challenges, performance

- The performance of MIP solvers can/must be improved in many different directions.

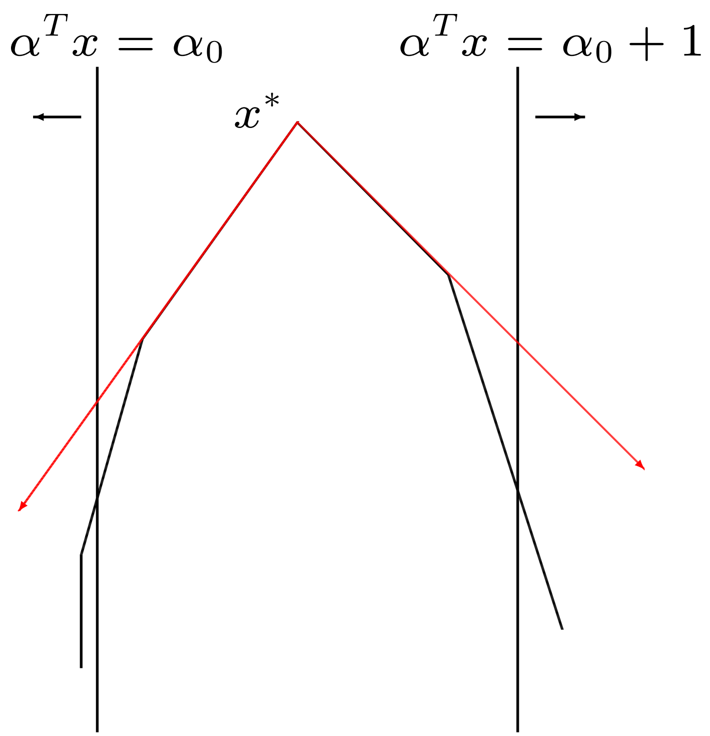
MIP Challenges, performance

- The performance of MIP solvers can/must be improved in many different directions. Among them my favorite ones are:
  - branching vs cutting
  - sophisticated techniques for general integer and continuous variables
  - performance variability
  - revisiting good "old" methods
  - cutting planes exploitation
  - symmetric MIPs

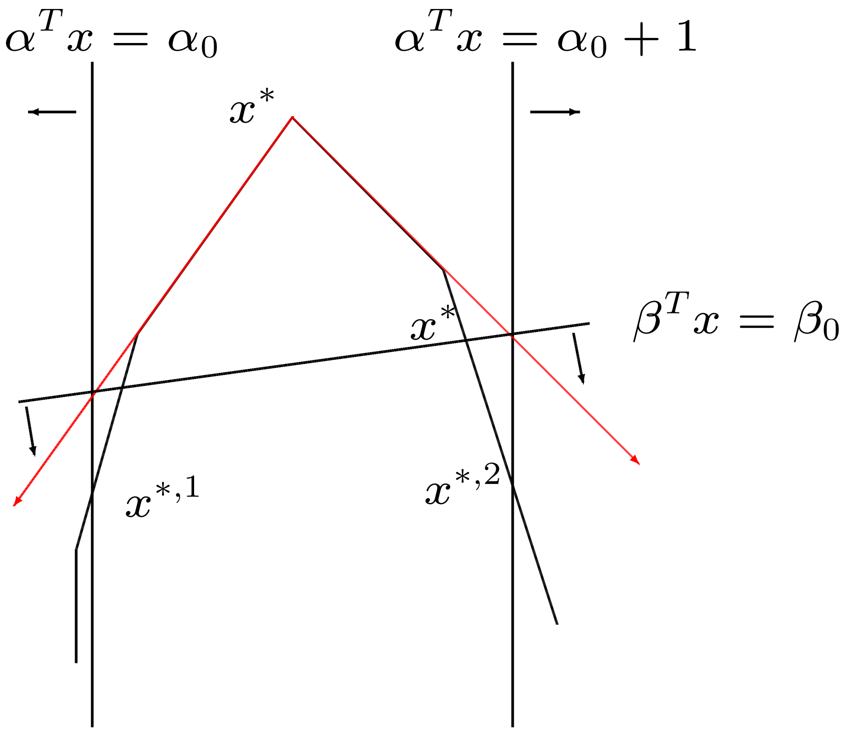
MIP Challenges: branching vs cutting



MIP Challenges: branching vs cutting

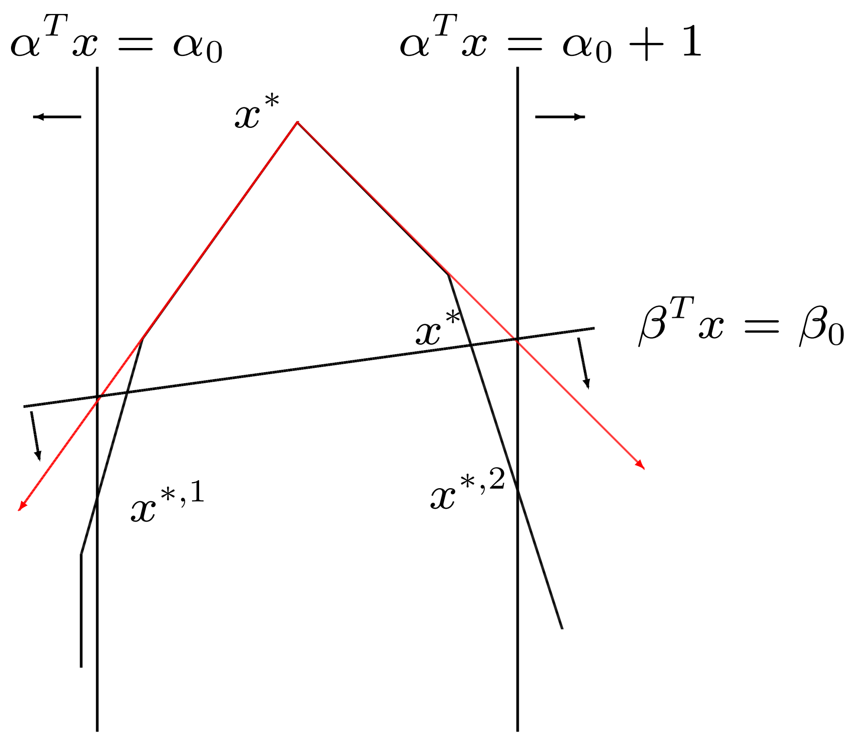


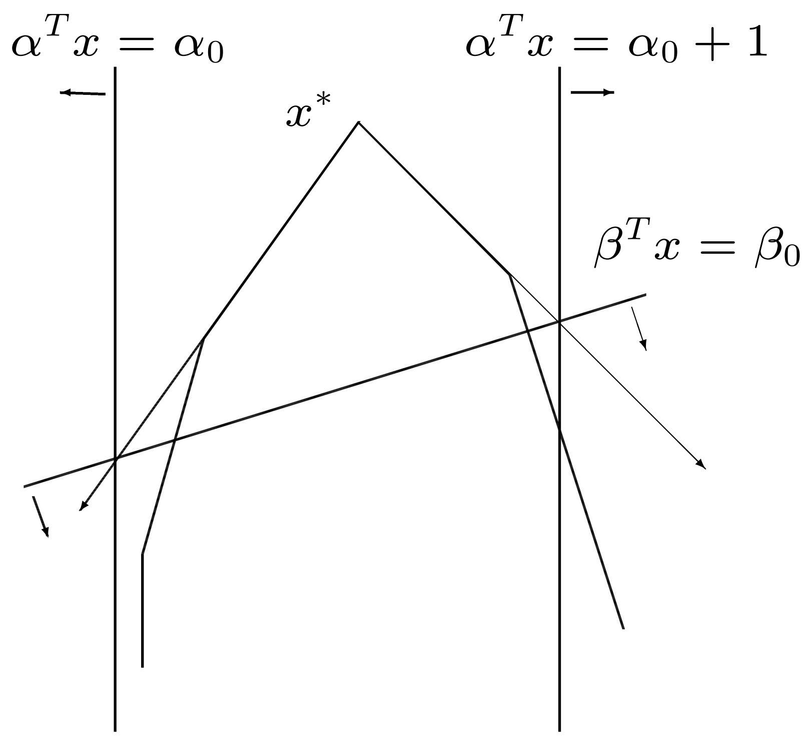
MIP Challenges: branching vs cutting



first wisdom

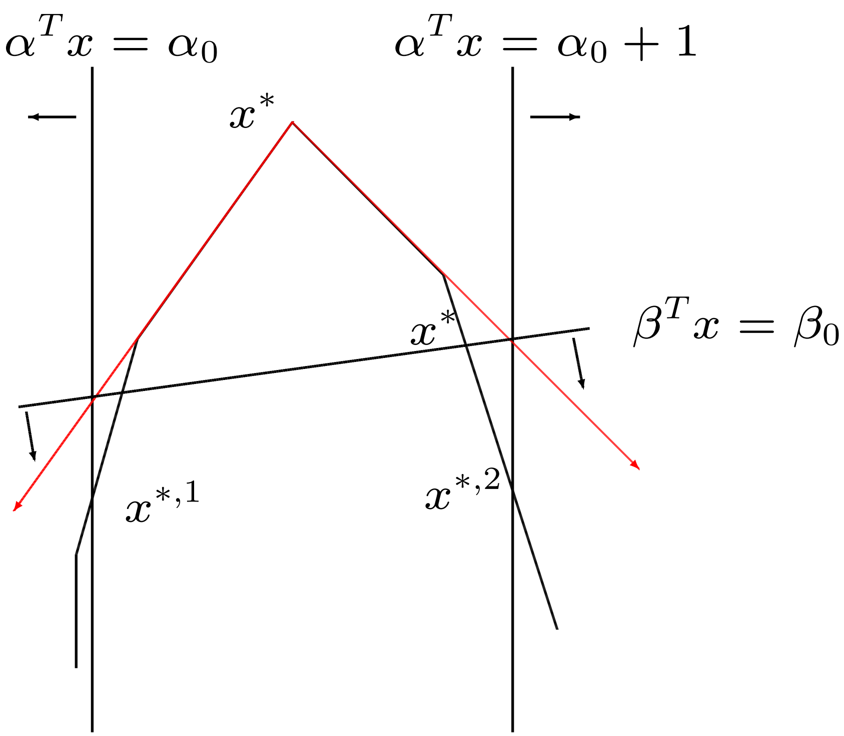
MIP Challenges: branching vs cutting

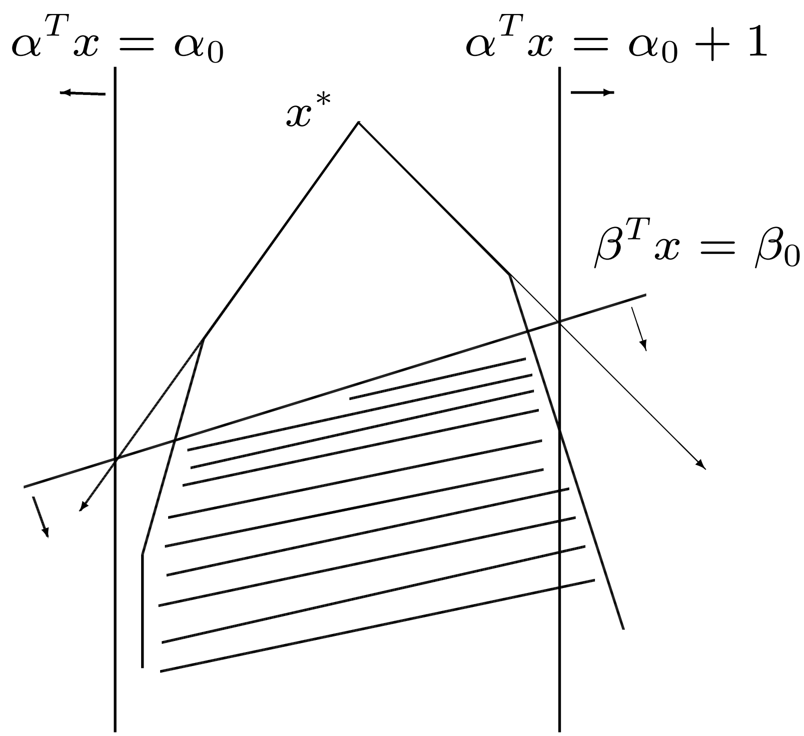




first wisdom

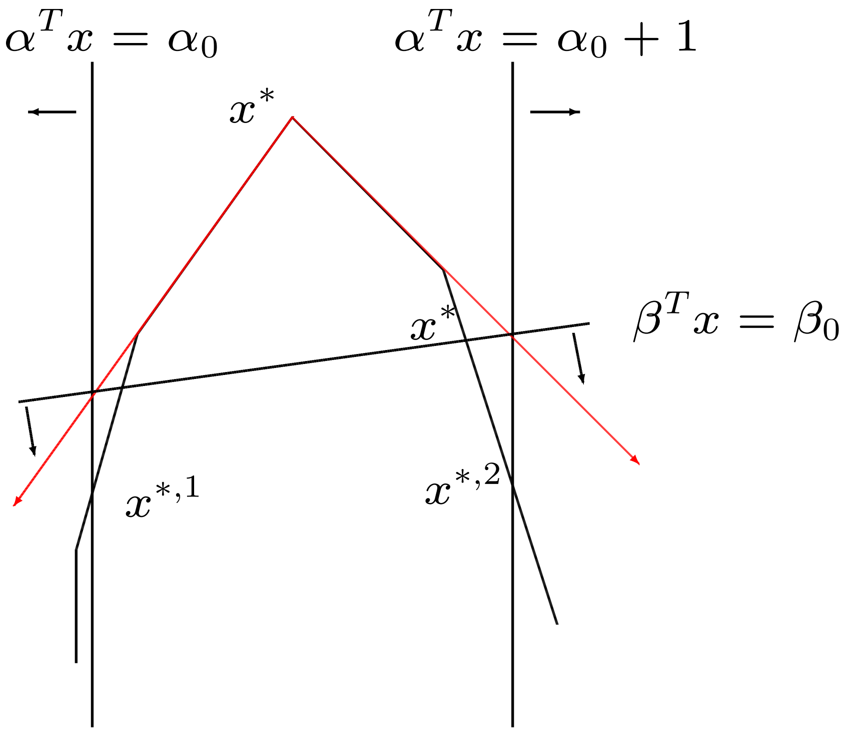
MIP Challenges: branching vs cutting



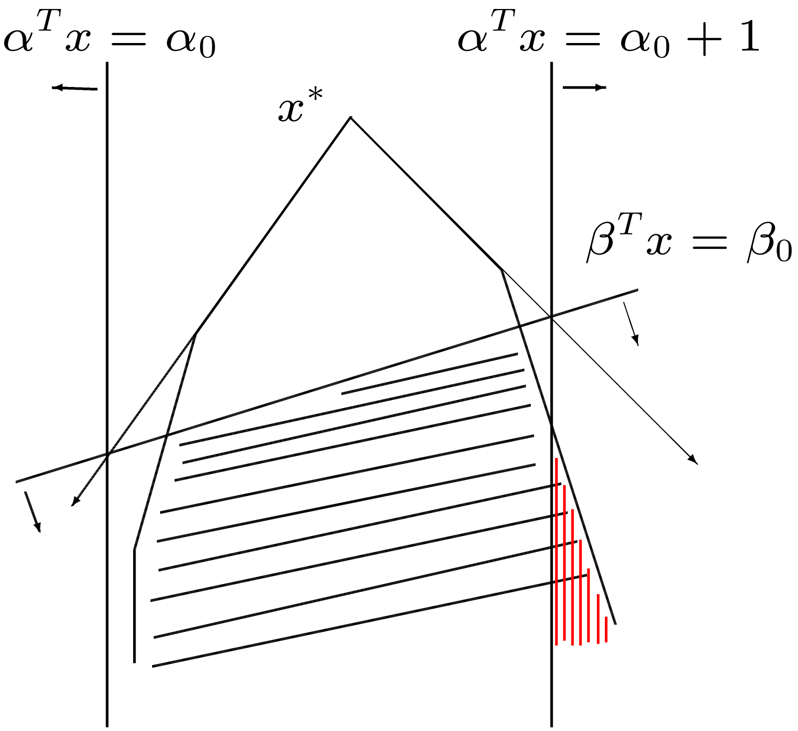


first wisdom

MIP Challenges: branching vs cutting



first wisdom



second wisdom

MIP Challenges, branching vs cutting (cont.d)

- The previous slide highlights a possibility of using traditional cutting plane theory in the branching context. [Karamanov & Cornuéjols]

MIP Challenges, branching vs cutting (cont.d)

- The previous slide highlights a possibility of using traditional cutting plane theory in the branching context. [Karamanov & Cornuéjols]
- It seems that a better coordination of these two fundamental ingredients of the MIP solvers is crucial for strong improvements.

MIP Challenges, branching vs cutting (cont.d)

- The previous slide highlights a possibility of using traditional cutting plane theory in the branching context. [Karamanov & Cornuéjols]
- It seems that a better coordination of these two fundamental ingredients of the MIP solvers is crucial for strong improvements.

- In the context of hard knapsack constraints branching on variables is not effective while (pure) basis reduction methods have proved to be very powerful. [Eisenbrand; Aardal; Pataki; . . .]

MIP Challenges, branching vs cutting (cont.d)

- The previous slide highlights a possibility of using traditional cutting plane theory in the branching context. [Karamanov & Cornuéjols]
- It seems that a better coordination of these two fundamental ingredients of the MIP solvers is crucial for strong improvements.

- In the context of hard knapsack constraints branching on variables is not effective while (pure) basis reduction methods have proved to be very powerful. [Eisenbrand; Aardal; Pataki; ...]
- On the other hand, a tight integration of basis reduction techniques within MIP solvers has not yet been achieved. One possibility for such an integration is the use of partial reformulations but an intriguing option is exploiting these reformulations to generate cuts in the original space of variables. [Aardal & Wolsey]

MIP Challenges, branching vs cutting (cont.d)

- The previous slide highlights a possibility of using traditional cutting plane theory in the branching context. [Karamanov & Cornuéjols]
- It seems that a better coordination of these two fundamental ingredients of the MIP solvers is crucial for strong improvements.

- In the context of hard knapsack constraints branching on variables is not effective while (pure) basis reduction methods have proved to be very powerful. [Eisenbrand; Aardal; Pataki; . . .]
- On the other hand, a tight integration of basis reduction techniques within MIP solvers has not yet been achieved. One possibility for such an integration is the use of partial reformulations but an intriguing option is exploiting these reformulations to generate cuts in the original space of variables. [Aardal & Wolsey]

- Finally, branching on appropriate disjunctions has been recently proposed in the context of highly symmetric MIPs. [Ostrowsky, Linderoth, Rossi & Smriglio]

MIP Challenges, performance (cont.d)

- A very important class of MIPs is 0/1 IPs. Many of the sophisticated techniques already discussed have been originally proposed for this class and eventually extended to general MIPs.

MIP Challenges, performance (cont.d)

- A very important class of MIPs is 0/1 IPs. Many of the sophisticated techniques already discussed have been originally proposed for this class and eventually extended to general MIPs.
- For example, branching on variables is particularly natural and effective in the 0/1 case while it is not when general integer variables play a central role.

MIP Challenges, performance (cont.d)

- A very important class of MIPs is 0/1 IPs. Many of the sophisticated techniques already discussed have been originally proposed for this class and eventually extended to general MIPs.
- For example, branching on variables is particularly natural and effective in the 0/1 case while it is not when general integer variables play a central role.
- Another example are the models in which continuous variables are important: for those variables MIP solvers do not do much (heuristics, strengthening, . . .).

MIP Challenges, performance (cont.d)

- A very important class of MIPs is 0/1 IPs. Many of the sophisticated techniques already discussed have been originally proposed for this class and eventually extended to general MIPs.
- For example, branching on variables is particularly natural and effective in the 0/1 case while it is not when general integer variables play a central role.
- Another example are the models in which continuous variables are important: for those variables MIP solvers do not do much (heuristics, strengthening, . . .).
- A (urgent) MIP challenge is definitely dealing with general integer and continuous variables with special-purpose techniques.

MIP Challenges, performance (cont.d)

- A very important class of MIPs is 0/1 IPs. Many of the sophisticated techniques already discussed have been originally proposed for this class and eventually extended to general MIPs.
- For example, branching on variables is particularly natural and effective in the 0/1 case while it is not when general integer variables play a central role.
- Another example are the models in which continuous variables are important: for those variables MIP solvers do not do much (heuristics, strengthening, . . .).
- A (urgent) MIP challenge is definitely dealing with general integer and continuous variables with special-purpose techniques.

- Cutting planes generation has been a key step for the success of MIP solvers but: are we using cuts in the best way?

MIP Challenges, performance (cont.d)

- A very important class of MIPs is 0/1 IPs. Many of the sophisticated techniques already discussed have been originally proposed for this class and eventually extended to general MIPs.
- For example, branching on variables is particularly natural and effective in the 0/1 case while it is not when general integer variables play a central role.
- Another example are the models in which continuous variables are important: for those variables MIP solvers do not do much (heuristics, strengthening, . . .).
- A (urgent) MIP challenge is definitely dealing with general integer and continuous variables with special-purpose techniques.

- Cutting planes generation has been a key step for the success of MIP solvers but: are we using cuts in the best way?By far not!

MIP Challenges, performance (cont.d)

- A very important class of MIPs is 0/1 IPs. Many of the sophisticated techniques already discussed have been originally proposed for this class and eventually extended to general MIPs.
- For example, branching on variables is particularly natural and effective in the 0/1 case while it is not when general integer variables play a central role.
- Another example are the models in which continuous variables are important: for those variables MIP solvers do not do much (heuristics, strengthening, . . .).
- A (urgent) MIP challenge is definitely dealing with general integer and continuous variables with special-purpose techniques.

- Cutting planes generation has been a key step for the success of MIP solvers but: are we using cuts in the best way?By far not!
- Fundamental questions about the use of cutting planes remain open:
  - stabilization issues
  - cut selection
  - cut interaction

MIP Challenges, performance (cont.d)

- The interaction of key ingredients presented before has side effects: positive and negative ones.

MIP Challenges, performance (cont.d)

- The interaction of key ingredients presented before has side effects: positive and negative ones.
- On the positive side, improvements in LP performance explicitly speed up node throughput, but implicitly help because one can now do more strong branching and MIPping.

MIP Challenges, performance (cont.d)

- The interaction of key ingredients presented before has side effects: positive and negative ones.
- On the positive side, improvements in LP performance explicitly speed up node throughput, but implicitly help because one can now do more strong branching and MIPping.
- On the negative side, finding a (near-)optimal solution very early in the search tree explicitly improves the quality of the primal bound but might sometimes hurt in proving optimality (or at least does not help).

MIP Challenges, performance (cont.d)

- The interaction of key ingredients presented before has side effects: positive and negative ones.
- On the positive side, improvements in LP performance explicitly speed up node throughput, but implicitly help because one can now do more strong branching and MIPping.
- On the negative side, finding a (near-)optimal solution very early in the search tree explicitly improves the quality of the primal bound but might sometimes hurt in proving optimality (or at least does not help).
- This is an example of what we call performance variability: some good features that might not be monotonically helpful.

MIP Challenges, performance (cont.d)

- The interaction of key ingredients presented before has side effects: positive and negative ones.
- On the positive side, improvements in LP performance explicitly speed up node throughput, but implicitly help because one can now do more strong branching and MIPping.
- On the negative side, finding a (near-)optimal solution very early in the search tree explicitly improves the quality of the primal bound but might sometimes hurt in proving optimality (or at least does not help).
- This is an example of what we call performance variability: some good features that might not be monotonically helpful.

  A deeper understanding through sophisticated testing techniques is needed. [Hooker; McGeoch; Margot]

MIP Challenges, performance (cont.d)

- The interaction of key ingredients presented before has side effects: positive and negative ones.
- On the positive side, improvements in LP performance explicitly speed up node throughput, but implicitly help because one can now do more strong branching and MIPping.
- On the negative side, finding a (near-)optimal solution very early in the search tree explicitly improves the quality of the primal bound but might sometimes hurt in proving optimality (or at least does not help).
- This is an example of what we call performance variability: some good features that might not be monotonically helpful.
  A deeper understanding through sophisticated testing techniques is needed. [Hooker; McGeoch; Margot]
- The negative example suggests an additional very crucial question: besides avoiding good primal solutions hurting the optimality proof, how can we use them to have instead a strong speed up?

MIP Challenges, performance (cont.d)

- The interaction of key ingredients presented before has side effects: positive and negative ones.
- On the positive side, improvements in LP performance explicitly speed up node throughput, but implicitly help because one can now do more strong branching and MIPping.
- On the negative side, finding a (near-)optimal solution very early in the search tree explicitly improves the quality of the primal bound but might sometimes hurt in proving optimality (or at least does not help).
- This is an example of what we call performance variability: some good features that might not be monotonically helpful.
  A deeper understanding through sophisticated testing techniques is needed. [Hooker; McGeoch; Margot]

- The negative example suggests an additional very crucial question: besides avoiding good primal solutions hurting the optimality proof, how can we use them to have instead a strong speed up?

- Good "old" methods have been rediscovered and revisited during the years and it is hard to believe that we understand them fully. Recently:
  - strong Benders cutting planes [Fischetti, Salvagnin & Zanette],
  - lexicographic [Zanette, Fischetti & Balas]
  - cutting planes from group relaxation [Gomory; Richard; Dey; Wolsey; Dash & Günlük;. . .]

MIP Challenges, the application viewpoint

- Besides developing additional tools in the spirit of the ones described before (among all possible I would like

  a tool for detecting minimal sources of numerical instability)

MIP Challenges, the application viewpoint

- Besides developing additional tools in the spirit of the ones described before (among all possible I would like

  a tool for detecting minimal sources of numerical instability)

  the main challenge from an application viewpoint seems to be dissemination.

MIP Challenges, the application viewpoint

- Besides developing additional tools in the spirit of the ones described before (among all possible I would like
  a tool for detecting minimal sources of numerical instability)
  the main challenge from an application viewpoint seems to be dissemination.

- More precisely, an interesting direction would be extending the modeling (and solving) capability within the MIP framework.

MIP Challenges, the application viewpoint

- Besides developing additional tools in the spirit of the ones described before (among all possible I would like

  a tool for detecting minimal sources of numerical instability)

  the main challenge from an application viewpoint seems to be dissemination.

- More precisely, an interesting direction would be extending the modeling (and solving) capability within the MIP framework.
- Two successful stories in this direction are:

  1. SCIP (Solving Constraint Integer Programs, [Achterberg]) whose main feature is a tight integration of Constraint Programming and SATisfiability techniques within a MIP solver. It can handle arbitrary (non-linear) constraints in a CP fashion.

MIP Challenges, the application viewpoint

- Besides developing additional tools in the spirit of the ones described before (among all possible I would like

  a tool for detecting minimal sources of numerical instability)

  the main challenge from an application viewpoint seems to be dissemination.

- More precisely, an interesting direction would be extending the modeling (and solving) capability within the MIP framework.
- Two successful stories in this direction are:

  1. SCIP (Solving Constraint Integer Programs, [Achterberg]) whose main feature is a tight integration of Constraint Programming and SATisfiability techniques within a MIP solver. It can handle arbitrary (non-linear) constraints in a CP fashion.

  2. Bonmin (Basic Open-source Nonlinear Mixed INteger programming, [Bonami et al.]) has been developed for Convex MINLP within the framework of the MIP solver Cbc [Forrest].

Computation in MIP

- Before ending the talk, I want to anticipate François question:

Computation in MIP

- Before ending the talk, I want to anticipate François question:

are these cuts valid?

Computation in MIP

- Before ending the talk, I want to anticipate François question:

  are these cuts valid?

- Indeed, a big challenge is accuracy which is a new issue, i.e., an old issue that starts to be very important after realizing that MIP solvers can now really solve the problems.

Computation in MIP

- Before ending the talk, I want to anticipate François question:

  are these cuts valid?

- Indeed, a big challenge is accuracy which is a new issue, i.e., an old issue that starts to be very important after realizing that MIP solvers can now really solve the problems.

- In summary:

Computation in MIP

- Before ending the talk, I want to anticipate François question:

  are these cuts valid?

- Indeed, a big challenge is accuracy which is a new issue, i.e., an old issue that starts to be very important after realizing that MIP solvers can now really solve the problems.

- In summary: still a long way to go!