

Solvers for Mixed Integer Programming

600.325/425 Declarative Methods - J. Eisner 1

Relaxation: A general optimization technique

- **Want:**
 - $x^* = \text{argmin } f(x)$ subject to $x \in S$
 - S is the feasible set
- **Start by getting:**
 - $x_1 = \text{argmin } f(x)$ subject to $x \in T$
 - where $S \subseteq T$
 - T is a larger feasible set, obtained by dropping some constraints
 - Makes problem easier if we have a large # of constraints or difficult ones
 - If we're lucky, it happens that $x_1 \in S$
 - Then $x^* = x_1$, since
 - x_1 is a feasible solution to the original problem
 - no feasible solution better than x_1 (no better $x \in S$ since none anywhere in T)
 - Else, add some constraints back (to shrink T) and try again, getting x_2
 - $x_1, x_2, x_3, \dots \rightarrow x^*$ as T closes in on S

600.325/425 Declarative Methods - J. Eisner 2

Relaxation: A general optimization technique

- **Want:**
 - $x^* = \text{min } f(x)$ subject to $x \in S$
 - S is the feasible set
- **Start by getting:**
 - $x_1 = \text{min } f(x)$ subject to $x \in T$
 - where $S \subseteq T$
 - T is a larger feasible set, obtained by dropping some constraints
 - Makes problem easier if we have a large # of constraints or difficult ones

Integrity constraints: if we drop all of these, we can just use simplex. "LP relaxation of the ILP problem."

- Else, add some constraints back (to shrink T) and try again

But how can we add integrity constraints back? (simplex relies on having dropped them all)

600.325/425 Declarative Methods - J. Eisner 3

Rounding doesn't work

round to nearest int (3,3)? No, infeasible.
round to nearest feasible int (2,3) or (3,2)? No, suboptimal.
round to nearest integer vertex (0,4)? No, suboptimal.

Function to maximize: $f(x, y) = 2x + 5y$
Optimum LP solution $(x, y) = (2.8, 3.3)$
Pareto optima: $(0, 4), (2, 3), (3, 2), (4, 1)$
Optimum ILP solution $(x, y) = (4, 1)$

Really do have to add the integrity constraints back somehow, and solve a new optimization problem.

image adapted from Jop Sibeyn 600.325/425 Declarative Methods - J. Eisner 4

Cutting planes: add new linear constraints

Decreasing cost

(a) (b) (c)

Figure 14-2 Illustration of a cutting-plane algorithm. (a) The continuous optimum x_1 . (b) The new x_2 after one cut. (c) The solution of the original ILP after two cuts.

- New linear constraints can be handled by simplex algorithm
- But will collectively rule out non-integer solutions

600.325/425 Declarative Methods - J. Eisner 5
figure adapted from Papadimitriou & Steiglitz

Add new linear constraints: Cutting planes

Decreasing cost

(a) (b) (c)

- Can ultimately trim back to a new polytope with only integer vertices
 - This is the "convex hull" of the feasible set of the ILP
- Since it's a polytope, it can be defined by linear constraints!
 - These can replace the integrality constraints
- Unfortunately, there may be exponentially many of them ...
 - But hopefully we'll only have to add a few (thanks to relaxation)

600.325/425 Declarative Methods - J. Eisner 6
figure adapted from Papadimitriou & Steiglitz

Example

$$x_1 + 4x_2 + x_3 \geq 10$$

$$4x_1 + 2x_2 + 2x_3 \geq 13$$

$$x_1 + x_2 - x_3 \geq 0$$

$x_1, x_2, x_3 \geq 0$ and integer.

$$x_1 + 4x_2 + x_3 \geq 10$$

$$x_1 + 3x_2 + x_3 \geq 9$$

$$2x_1 + 4x_2 + x_3 \geq 13$$

$$x_1 + x_2 + x_3 \geq 5$$

$$2x_1 + x_2 + x_3 \geq 7$$

$$x_1 + 2x_2 \geq 5$$

$$2x_1 + x_2 \geq 4$$

$$x_1 + x_2 - x_3 \geq 0$$

$$x_1, x_2, x_3 \geq 0.$$

No integrality constraints!
But optimal solution is the same.

- How can we find these new constraints??

example from H. P. Williams 600.325/425 Declarative Methods - J. Eisner 7

Chvátal cuts

$$x_1 + 4x_2 + x_3 \geq 10$$

$$4x_1 + 2x_2 + 2x_3 \geq 13$$

$$x_1 + x_2 - x_3 \geq 0$$

$x_1, x_2, x_3 \geq 0$ and integer.

$$\xrightarrow{\text{divide by 2}} 2x_1 + x_2 + x_3 \geq 6$$

Must be integer because of integrality constraints on x_1, x_2, x_3

round

$$2x_1 + x_2 + x_3 \geq 7$$

$$+ 5 * (x_1 + 4x_2 + x_3 \geq 10)$$

$$+ (x_1 + x_2 - x_3 \geq 0)$$

$$7x_1 + 21x_2 + 7x_3 \geq 57$$

divide by 7 and round

$$x_1 + 3x_2 + x_3 \geq 9$$

- Add integer multiples of constraints, divide through, and round using integrality
- This generates a new (or old) constraint
- Repeat till no new constraints can be generated
- Generates the convex hull of the ILP!
- But it's impractical

example from H. P. Williams 600.325/425 Declarative Methods - J. Eisner 8

Gomory cuts

- Chvátal cuts:**
 - Can generate the convex hull of the ILP!
 - But that's impractical
 - And unnecessary (since we just need to find optimum, not whole convex hull)
- Gomory cuts:**
 - Only try to cut off current relaxed optimum that was found by simplex
 - "Gomory cut" derives such a cut from the current solution of simplex

figure adapted from Papadimitriou & Steiglitz 600.325/425 Declarative Methods - J. Eisner 9

Branch and bound: Disjunctive cutting planes!

Minimise $2x_1 + 7x_2 + 2x_3$

$$x_1 + 4x_2 + x_3 \geq 10$$

$$4x_1 + 2x_2 + 2x_3 \geq 13$$

$$x_1 + x_2 - x_3 \geq 0$$

$x_1, x_2, x_3 \geq 0$ and integer.

For each leaf, why is it okay to stop there?

figure from H. P. Williams

Remember branch-and-bound from constraint programming?

figure thanks to Tobias Achterberg 600.325/425 Declarative Methods - J. Eisner 11

Branch and bound: Pseudocode

Or set \hat{c}, \hat{x} to a known feasible solution.

In notation, \hat{c} is upper bound (feasible but poor objective) - decreases globally
 \hat{v} is lower bound (good objective but infeasible) - increases down the tree

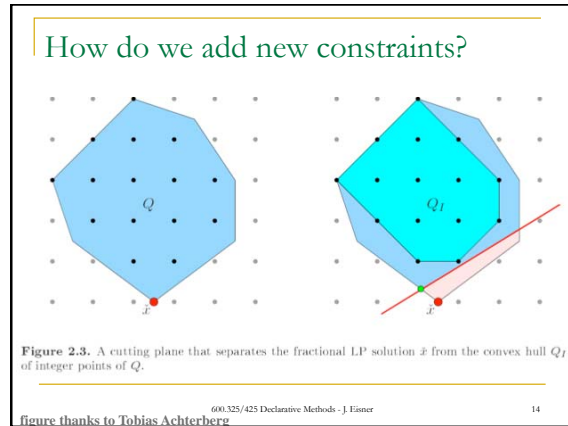
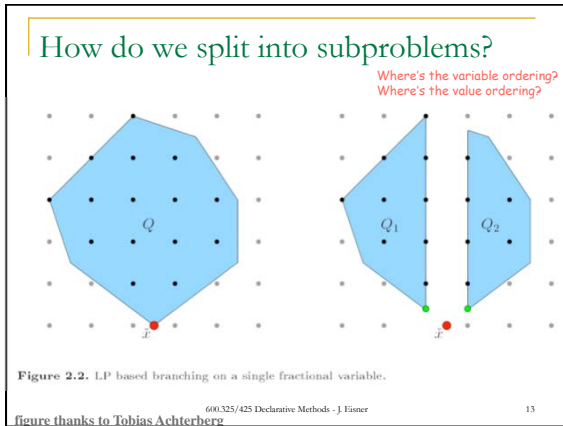
Input: Minimization problem instance R . May simplify ("presolve") it first.

Output: Optimal solution x^* with value c^* , or conclusion that R has no solution, indicated by $c^* = \infty$.

- Initialize $\mathcal{L} := \{R\}$, $\hat{c} := \infty$. Simplify if desired by propagation; then relax by dropping integrality constraints [init]
- If $\mathcal{L} = \emptyset$, stop and return $x^* = \hat{x}$ and $c^* = \hat{c}$. [abort]
- Choose $Q \in \mathcal{L}$, and set $\mathcal{L} := \mathcal{L} \setminus \{Q\}$. [select]
- Solve a relaxation Q_{relax} of Q . If Q_{relax} is empty, set $\hat{c} := \infty$. Otherwise, let \hat{x} be an optimal solution of Q_{relax} and \hat{c} its objective value. [solve]
- If $\hat{c} \geq \hat{c}$, goto Step 2. [bound]
- If \hat{x} is feasible for R , set $\hat{x} := \hat{x}$, $\hat{c} := \hat{c}$, and goto Step 2. [check]
- Split Q into subproblems $Q = Q_1 \cup \dots \cup Q_k$, set $\mathcal{L} := \mathcal{L} \cup \{Q_1, \dots, Q_k\}$, and goto Step 2.

Can insert a stochastic local search here to try to find a feasible solution x^* near \hat{x} , to improve upper bound c^* further
pseudocode thanks to Tobias Achterberg

Branch&cut: add new constraints here (cutting planes, conflict clauses, or pick from huge set ("row generation"))
Branch&price: add new non-0 vars picked from huge set ("column gener.")



Variable & value ordering heuristics (at a given node)

- **Priorities:** User-specified var ordering
- **Most fractional branching:** Branch on variable farthest from int
- Branch on a variable that should tighten (hurt) the LP relaxation a lot
 - **Strong branching:** For several candidate variables, try rounding them and solving the LP relaxation (perhaps incompletely).
 - **Penalties:** If we rounded x up or down, how much would it tighten objective just on next iteration of dual simplex algorithm? (Dual simplex maintains an overly optimistic cost estimate that relaxes integrality and may be infeasible in other ways, too.)
 - **Pseudo-costs:** When rounding this variable in the past, how much has it *actually* tightened the LP relaxation objective (on average), per unit increase or decrease?
- Branching on SOS1 and SOS2

600.325/425 Declarative Methods - J. Eisner

Warning

- If variables are unbounded, the search tree might have infinitely many nodes!

figures from H. P. Williams

Warning

- If variables are unbounded, the search tree might have infinitely many nodes!
- Fortunately, it's possible to compute bounds ...
 - Given an LP or ILP problem (min $c \cdot x$ subj. to $Ax=b, x \geq 0$)
 - Where all numbers in A, b, c are integers; n vars, m constraints
 - If there's a finite optimum x , each x_i is \leq a bound whose log is
 - $O(m^2 \log m \log(\text{biggest integer in } A \text{ or } b))$ [for LP]

Intuition for LP: Only way to get LP optima far from the origin is to have slopes that are close but not quite equal ... which requires large ints.

figures from Papadimitriou & Steiglitz

Warning

- If variables are unbounded, the search tree might have infinitely many nodes!
- Fortunately, it's possible to compute bounds ...
 - Given an LP or ILP problem (min $c \cdot x$ subj. to $Ax=b, x \geq 0$)
 - Where all numbers in A, b, c are integers; n vars, m constraints
 - If there's a finite optimum x , each x_i is \leq a bound whose log is
 - $O(m^2 \log m \log(\text{biggest integer in } A \text{ or } b))$ [for LP]
 - $O(\log n + m(\log n + \log(\text{biggest int in } A, b, \text{ or } c)))$ [for ILP]

For ILP: A little trickier. (Could ILP have huge finite optimum if LP is unbounded? Answer: no, then ILP unbounded too.)

figures from Papadimitriou & Steiglitz

Reducing ILP to 0-1 ILP

- Given an LP or ILP problem (min c.x subj. to Ax=b, x>=0)
- Where all numbers in A,b,c are integers; n vars, m constraints
- If there's a finite optimum x, each x_i is ≤ a bound whose log is
 - O(log n + m(log n + log (biggest int in A, b, or c))) [for ILP]

If log bound=100, then e.g. 0 ≤ x₅ ≤ 2¹⁰⁰

Remark: This bound enables a polytime reduction from ILP to 0-1 ILP

- Remember: Size of problem = length of encoding, not size of #s
- Can you see how?
- Hint: Binary numbers are encoded with 0 and 1
- What happens to linear function like ... + 3 x₅ + ... ?

600.325/425 Declarative Methods - J. Eisner 19

Totally Unimodular Problems

- There are some ILP problems where nothing is lost by relaxing to LP!
 - "some mysterious, friendly power is at work" -- Papadimitriou & Steiglitz
 - All vertices of the LP polytope are integral anyway.
 - So regardless of the cost function, the LP has an optimal solution in integer variables (& maybe others)
 - No need for cutting planes or branch-and-bound.
 - This is the case when A is a **totally unimodular** integer matrix, and b is integral. (c can be non-int.)

$$A\vec{x} \leq \vec{b} \quad (\text{or } A\vec{x} = \vec{b})$$

600.325/425 Declarative Methods - J. Eisner 20

Totally Unimodular Cost Matrix A

- A square matrix with determinant +1 or -1 is called **unitary**.
- A unitary integer matrix is called **unimodular**. Its inverse is integral too!
 - (follows easily from A⁻¹ = adjoint(A) / det(A))
 - Matrices are like numbers, but more general. Unimodular matrices are the matrix generalizations of +1 and -1: you can divide by them without introducing fractions.
- A **totally unimodular** matrix is one whose square submatrices (obtained by crossing out rows or columns) are all either unimodular (det=±1) or singular (det=0).
 - Matters because simplex inverts non-singular square submatrices.

600.325/425 Declarative Methods - J. Eisner 21

Some Totally Unimodular Problems

- The following common **graph problems** pick a subset of edges from some graph, or assign a weight to each edge in a graph.
 - Weighted bipartite matching
 - Shortest path
 - Maximum flow
 - Minimum-cost flow
- Their cost matrices are totally unimodular.
 - They satisfy the conditions of a superficial test that is sufficient to guarantee total unimodularity.
 - So, they can all be solved right away by the simplex algorithm or another LP algorithm like primal-dual.
 - All have well-known direct algorithms, but those can be seen as essentially just special cases of more general LP algorithms.

600.325/425 Declarative Methods - J. Eisner 22

Some Totally Unimodular Problems

- The following common **graph problems** pick a subset of edges from some graph ...
 - Weighted matching in a bipartite graph

max $\sum_{ij} c_{ij}x_{ij}$ with x_{ij} binary (x_{ij} ≥ 0)

subto $(\forall i) \sum_j x_{ij} \leq 1$ each top/bottom node has at most one edge

$(\forall j) \sum_i x_{ij} \leq 1$ one edge

If we formulate as Ax ≤ b, x ≥ 0, the A matrix is totally unimodular:

	x _{iA}	x _{iB}	x _{iiB}	x _{iiiC}	x _{ivB}	x _{ivC}
(i=A)	1	1	0	0	0	0
(i=B)	0	0	1	0	0	0
(i=ii)	0	0	0	1	0	0
(i=iii)	0	0	0	0	1	0
(i=iv)	0	0	0	0	0	1
(j=A)	1	0	0	0	0	0
(j=B)	0	1	1	0	1	0
(j=C)	0	0	0	1	0	1

Sufficient condition: Each column (for edge x_{ij}) has at most 2 nonzero entries (for i and j). These are both +1 (or both -1) and are in different "halves" of the matrix. (Also okay if they are +1 and -1 and are in same "half" of the matrix.)

drawing from Edwards M T et al. Nucl. Acids Res. 2005;33:3253-3262 (Oxford Univ. Press)

Some Totally Unimodular Problems

- The following common **graph problems** pick a subset of edges from some graph ...
 - Shortest path from s to t in a directed graph

min $\sum_{ij} c_{ij}x_{ij}$ with x_{ij} binary

subto $\sum_j x_{sj} = 1, \sum_j x_{jt} = 1$

$(\forall j \notin \{s, t\}) \sum_i x_{ij} = \sum_k x_{jk}$

Can formulate as Ax = b, x ≥ 0 so that A matrix is totally unimodular:

	x _{sA}	x _{sB}	x _{AB}	x _{BC}	x _{CA}	x _{CB}	x _{ct}
(s)	1	1	0	0	0	0	0
(j=A)	-1	0	1	0	0	0	0
(j=B)	0	0	-1	1	0	1	0
(j=C)	0	-1	0	-1	1	0	1
(t)	0	0	0	0	0	0	-1

Q: Can you prove that every feasible solution is a path?
A: No: it could be a path plus some cycles.
But then can reduce cost by throwing away the cycles. So optimal solution has no cycles.

24

Some Totally Unimodular Problems

- The following common **graph problems** pick a subset of edges from some graph ...
 - Shortest path from s to t in a directed graph
 - edge scores from drawing

$$\min \sum_{i,j} c_{ij} x_{ij}$$

with x_{ij} binary

Not needed! (just need $x_{ij} \geq 0$)

$$\text{subject to } \sum_j x_{sj} = 1, \sum_j -x_{jt} = -1$$

$$(\forall j \notin \{s, t\}) \sum_i -x_{ij} + \sum_k x_{jk} = 0$$

Can formulate as $Ax = b, x \geq 0$ so that **A matrix is totally unimodular:**

Sufficient condition: Each column (for edge x_{ij}) has at most 2 nonzero entries (for i and j). These are +1 and -1 and are in the same "half" of the matrix. (Also okay to be both +1 or both -1 and be in different "halves.")

	x_{sA}	x_{sC}	x_{AB}	x_{AC}	x_{CA}	x_{Bt}	x_{Ct}
(s)	1	1	0	0	0	0	0
(j=A)	-1	0	1	0	0	0	0
(j=B)	0	0	-1	1	0	1	0
(j=C)	0	-1	0	-1	1	0	1
(t)	0	0	0	0	0	-1	-1

(this "half" is empty, can divide rows into "halves" in any way that satisfies sufficient condition)

Some Totally Unimodular Problems

- The following common **graph problems** pick a subset of edges from some graph ...
 - Maximum flow (previous problems can be reduced to this)
 - Minimum-cost flow
- Cost matrix is rather similar to those on the previous slides, but with additional "capacity constraints" like $x_{ij} \leq k_{ij}$
- Fortunately, if A is totally unimodular, so is A with I (the identity matrix) glued underneath it to represent the additional constraints

Solving Linear Programs

600.325/425 Declarative Methods - J. Eisner 27

Canonical form of an LP

- $\min c \cdot x$ subject to $Ax \leq b, x \geq 0$
- m constraints (rows)
- n variables (columns) (usually $m < n$)

So x specifies a linear combination of the columns of A .

600.325/425 Declarative Methods - J. Eisner 28

Fourier-Motzkin elimination

- Variable elimination on a set of inequalities
 - [blackboard example]
- To eliminate variable z , first take each inequality involving z and solve it for z . This gives $z \leq \alpha_1, z \leq \alpha_2, \dots, z \geq \beta_1, z \geq \beta_2, \dots$
 - The α_i and β_j are linear functions of the other vars a, b, \dots, y
- Replace these inequalities by $\alpha_1 \geq \beta_1, \alpha_1 \geq \beta_2, \alpha_2 \geq \beta_1, \alpha_2 \geq \beta_2, \dots$
 - Equivalently, $\min \alpha_i \geq \max \beta_j$. These equations are true of an assignment a, b, \dots, y iff it can be extended with a consistent value for z .
 - Similar to resolution of CNF-SAT clauses in Davis-Putnam algorithm! Impractical since, just like resolution, may square the # of constraints. ☹
- Repeat to eliminate variable y , etc.
- If one of our equations is "**a = [linear cost function]**," then at the end, our only inequalities are lower and upper bounds on a .
 - Now it's easy to min or max a ! Then back-solve to get b, c, \dots, z in turn.

600.325/425 Declarative Methods - J. Eisner 29

From Canonical to Standard Form

- $\min c \cdot x$ subject to $Ax = b, x \geq 0$
- m constraints (rows)
- $n+m$ variables (columns) (sometimes # vars is called n , even in standard form; it's usually $>$ # constraints; the reason we call it $n+m$ will become clear)

600.325/425 Declarative Methods - J. Eisner 30


Canonical vs. Standard Form

Eliminate last m vars

$$Ax \leq b$$

$$x \geq 0$$

m inequalities
+ n inequalities
(n variables)

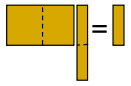


Cost of origin is easy to compute (it's a const in cost function). Eliminating a different set of m variables (picking a different basis) would rotate/reflect/squish the polytope & cost hyperplane to put a different vertex at origin, aligning that vertex's n constraints with the orthogonal $x \geq 0$ hyperplanes. This is how simplex algorithm tries different vertices!

$$Ax = b$$

$$x \geq 0$$

m equalities
+ n+m inequalities
(n+m variables)



vertex
(defined by intersecting n of the constraints, each of which reduces dimensionality by 1)

bfs
(defined by selecting n of the variables to be 0)

Pick n of the n+m equalities to be tight

Simplex algorithm

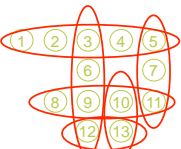
- How do we pick which column to phase in (i.e., which adjacent vertex to move to)?
 - How to avoid cycling back to an old bfs (in case of ties)?
- Alternative and degenerate solutions?
- What happens with unbounded LPs?
- How do we find a first bfs to start at?
 - Simplex phase I: Add "artificial" slack/surplus variables to make it easy to find a bfs, then phase them out via simplex. (Will happen automatically if we give the artificial variables a high cost.)
 - Or, just find any basic solution; then to make it feasible, phase out negative variables via simplex.
- Now continue with phase II. If phase I failed, no bfs exists for original problem, because:
 - The problem was infeasible (incompatible constraints, so quit and return UNSAT).
 - Or the m rows of A aren't linearly independent (redundant constraints, so throw away the extras & try again).

vertex
(defined by intersecting n-m of the constraints, each of which reduces dimensionality by 1)

bfs
(defined by selecting n-m of the variables to be 0)

Pick n-m of the m equalities to be tight

Recall: Duality for Constraint Programs



original ("primal") problem:
one variable per letter,
constraints over up to 5 vars

1	2	3	4	5
	6		7	
8	9	10	11	
	12	13		

transformed ("dual") problem:
one var per word, 2-var constraints.
Old constraints → new vars!
Old vars → new constraints!

600.325/425 Declarative Methods - J. Eisner 39
slide thanks to Rina Dechter (modified)

Duality for Linear Programs (canonical form)

Primal problem

$$\max c \cdot x$$

$$Ax \leq b$$

$$x \geq 0$$

(m)
(n)

↔ dualize ↔

Dual problem

$$\min b \cdot y$$

$$A^T y \geq c$$

$$y \geq 0$$

(n)
(m)

- The form above assumes $(\max, s) \Leftrightarrow (\min, z)$.
- Extensions for LPs in general form:
 - Any reverse constraints $((\max, z)$ or $(\min, s)) \Leftrightarrow$ negative vars
 - Any equality constraints \Leftrightarrow unbounded vars (can simulate with pair of constraints \Leftrightarrow pair of vars)

600.325/425 Declarative Methods - J. Eisner 40

Dual of dual = Primal

Primal problem

$$\max c \cdot x$$

$$Ax \leq b$$

$$x \geq 0$$

(m)
(n)

↔ dualize ↔

Dual problem

$$\min b \cdot y$$

$$A^T y \geq c$$

$$y \geq 0$$

(n)
(m)

Just negate A, b, and c

Equivalent to primal

$$\min (-c) \cdot x$$

$$(-A)x \geq b$$

$$x \geq 0$$

(m)
(n)

↔ dualize ↔

Equivalent to dual

$$\max (-b) \cdot y$$

$$(-A^T)y \leq (-c)$$

$$y \geq 0$$

(n)
(m)

600.325/425 Declarative Methods - J. Eisner 41

Primal & dual "meet in the middle"

Primal problem

$$\max c \cdot x$$

$$Ax \leq b$$

$$x \geq 0$$

(m)
(n)

↔ ↔

Dual problem

$$\min b \cdot y$$

$$A^T y \geq c$$

$$y \geq 0$$

(n)
(m)

- Turns out that for any feasible solutions x and y, $c \cdot x \leq b \cdot y$.
- So if $c \cdot x = b \cdot y$, both must be optimal!
 - (Remark: For nonlinear programming, there's a generalization where the constants in the dual constraints are partial derivatives of the primal constraint and cost function. The equality condition is then called the Kuhn-Tucker condition.)
- For LP, the converse is true: optimal solutions always have $c \cdot x = b \cdot y$!
 - Not true for nonlinear programming or ILP.

600.325/425 Declarative Methods - J. Eisner 42

Primal & dual “meet in the middle”

Not feasible under primal constraints

Max achievable under primal constraints

$c \cdot x$

$$\begin{matrix} \min & b \cdot y \\ A^T y & \geq c \\ y & \geq 0 \end{matrix}$$

$b \cdot y$

Min achievable under dual constraints

Not feasible under dual constraints

Max achievable under dual constraints

$$\begin{matrix} \max & c \cdot x \\ Ax & \leq b \\ x & \geq 0 \end{matrix}$$

$c \cdot x$

Not feasible under dual constraints

$c \cdot x \leq b \cdot y$ for all feasible (x, y) .
(So if one problem is unbounded, the other must be infeasible.)

600.325/425 Declarative Methods - J. Eisner 43

Duality for Linear Programs (standard form)

Primal problem

$$\begin{matrix} \max & c \cdot x \\ Ax + s & = b \\ x & \geq 0 \\ s & \geq 0 \end{matrix}$$

(n struct vars)
(m slack vars)

Dual problem

$$\begin{matrix} \min & b \cdot y \\ A^T y - t & = c \\ y & \geq 0 \\ t & \geq 0 \end{matrix}$$

(m struct vars)
(n surplus vars)

$x \cdot t + s \cdot y = 0$

- Now we have m+n variables and they are in 1-to-1 correspondence.
- At primal optimality:
 - Some m “basic” vars of primal can be ≥ 0 . The n non-basic vars are 0.
- At dual optimality:
 - Some n “basic” vars of dual can be ≥ 0 . The m non-basic vars are 0.
- Complementary slackness: The basic vars in an optimal solution to one problem correspond to the non-basic vars in an optimal solution to the other problem!
- So, if a structural variable in one problem > 0 , then the corresponding constraint in the other problem must be tight (its slack/surplus variable must be 0).
- And if a constraint in one problem is loose (slack/surplus var > 0), then the corresponding variable in the other problem must be 0. (logically equiv. to above)

Why duality is useful for ILP

Instead, let's find bound by dual simplex

Max achievable under LP relaxation

Max achievable for ILP at this node

$c \cdot x$

$$\begin{matrix} \min & b \cdot y \\ A^T y & \geq c \\ y & \geq 0 \end{matrix}$$

$b \cdot y$

Min achieved so far at this node as dual simplex runs

prune early!

Can also find this from dual of LP relaxation

best feasible global solution so far

Optimistic bound poor enough that we can prune this node

$$\begin{matrix} \max & c \cdot x \\ Ax & \leq b \\ x & \geq 0 \\ x & \text{integer} \end{matrix}$$

ILP problem at some node of branch-and-bound tree (includes some branching constraints)

Methods - J. Eisner 45

Understanding Duality [this slide will be expanded]

Drop the names s and t now; use standard form, but call all the variables x and y .

1. The $y_i \geq 0$ are coefficients on a linear sum of the primal constraints. Shows $c \cdot x \leq b \cdot y$, with equality iff complementary slackness holds.
2. Geometric interpretation of the above:
At a primal vertex x , cost hyperplane (shifted to go through the vertex) is a linear combination of the hyperplanes that intersect at that vertex. This is a nonnegative linear combination ($y_i \geq 0$, which is feasible in the dual) iff the cost hyperplane is tangent to the polytope at x (doesn't go through middle; technically, it's a subgradient at x), meaning that x is optimal.
3. “Shadow price” interpretation: Optimal y_i says how rapidly the primal optimum ($\max c \cdot x$) would improve as we relax primal constraint i . (A derivative.) Justify this by Lagrange multipliers.
4. “Reduced cost” interpretation: Each $y_i \geq 0$ is the rate at which $c \cdot x$ would get worse if we phased x_i into the basis while preserving $Ax=b$. This shows that (for an optimal vertex x), if $x_i > 0$ then $y_i = 0$, and if $y_i > 0$ then $x_i = 0$. At non-optimal x , y is infeasible in dual.