Integer Programming

David C. Parkes

Division of Engineering and Applied Science,

Harvard University

CS 286r–Spring 2002

Motivation

- Very flexible and expressive model for formulation of combinatorial optimization problems
  – combinatorial allocation problem
  – multicast routing problem
- Fast off-the-shelf solvers available
  – CPLEX, OSL, etc.
- Strong theoretical foundations
  – characterization of tractable special cases
- Connection to LP via relaxation/reformulation

Basic Definition

A typical formulation:

$$\begin{aligned} &\max_x \; c^T x \\ \text{s.t.} \quad &Ax \le b \\ &x \ge 0, \quad x \text{ integer} \end{aligned}$$

Note: a *mixed-integer* program has some integer variables and some LP variables.

Almost all problems NP-hard in general, but can often solve many large problems in practice.

Example

Assignment problem: n workers, n jobs. Each worker i has cost c_{ij} for performing job j. Problem is to assign jobs to workers to minimize the total cost.

$$\min_x \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j$$

$$x_{ij} \in \{0, 1\}$$

[this has $n!$ feasible solutions!]

Special Structures

Def. An integral polytope has only integral extremal points.

Lemma. If an IP has an integral polytope, then the LPR of an IP computes an optimal solution.

$P(A) = \{x \ : \ Ax \leq b\}$

- Look for a *useful geometric structure*:
  – totally unimodular 0,± 1 matrix (condition on A)
  – totally dual integral (condition on A and b)
  – balanced 0,1 matrix (condition on A)
- Tractable problem classes:
  – assignment problem, network flows, matching problems, etc.

Total Unimodularity

Thm. The LP $\{\max c^T x \; : \; Ax \leq b, x \in \mathbb{R}^n_+\}$ has an integral optimal solution for all integer vectors b for which it has a finite optimal value if and only if A is totally unimodular.

[necessary] If matrix A is TU, $a_{ij} \in \{+1, -1, 0\}$, for all i, j.

[sufficient] Matrix A is TU if: (i) $a_{ij} \in \{+1, -1, 0\}$; (ii) each column has at most two non-zero coefficients; (iii) the set of rows, M, can be paritioned into (M_1, M_2), s.t. each column j containing two nonzero coefficients satisfies $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$.

e.g., the assignment problem.

Relaxing a Formulation

It can be very useful to solve a relaxed IP formulation, for example to provide an upper-bound on the value of the best-possible solution to a maximization problem. [*we see this in Branch and Bound.*]

Def. Given $\max_x\{c^T x \,:\, x \in X\}$, then define a *relaxation* as $\max_x\{c^T x \,:\, x \in T\}$, where $X \subset T$.

e.g. drop the *integrality requirement*, and formulate as an LP.

Strengthening: Addition of Inequalities

It can be very useful to strengthen an IP formulation, with "valid inequalities", *before* using an LP relaxation to compute bounds (or solve the problem).

Def. Given formulation P_1, $\max_x\{c^T x \,:\, x \in X\}$, then define a *stronger formulation*, P_2, as $\max_x\{c^T x \,:\, x \in T\}$, where if x^* is optimal in P_1 then $x^* \in T$, but $T \subset X$.

Principle: introduce any valid inequalities that we know which might be active in an optimal solution.

note: cuts can be generated automatically, via combinatorial (Padberg) and/or geometric reasoning (Gomory).

Strengthening: Lifting Formulations

- Introduce additional variables (or “auxiliary variables”), typically a large number
- Geometrically, the original problem is viewed as the projection of a higher dimensional but simpler polyhedron

we see an example of this in the combiantorial allocation problem.

Strengthening is useful:

- Can allow the identification of tractable special cases, and/or efficient general purpose algorithms.
- With an integral LP formulation of an IP we can use primal-dual based methods to design iterative mechanisms.

Example: Combinatorial Allocation Problem

Given a set $\mathcal{G}$ of items, and a valuation, $v_i(S) \geq 0$ for each $S \subseteq \mathcal{G}$, and each agent i, a straightforward IP formulation is:

$$\max_{x_i(S)} \sum_S \sum_i x_i(S) v_i(S) \qquad \text{[CAP]}$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \qquad \forall i$$

$$\sum_{S: j \in S} \sum_i x_i(S) \leq 1, \qquad \forall j$$

$$x_i(S) \in \{0, 1\}, \qquad \forall i, S$$

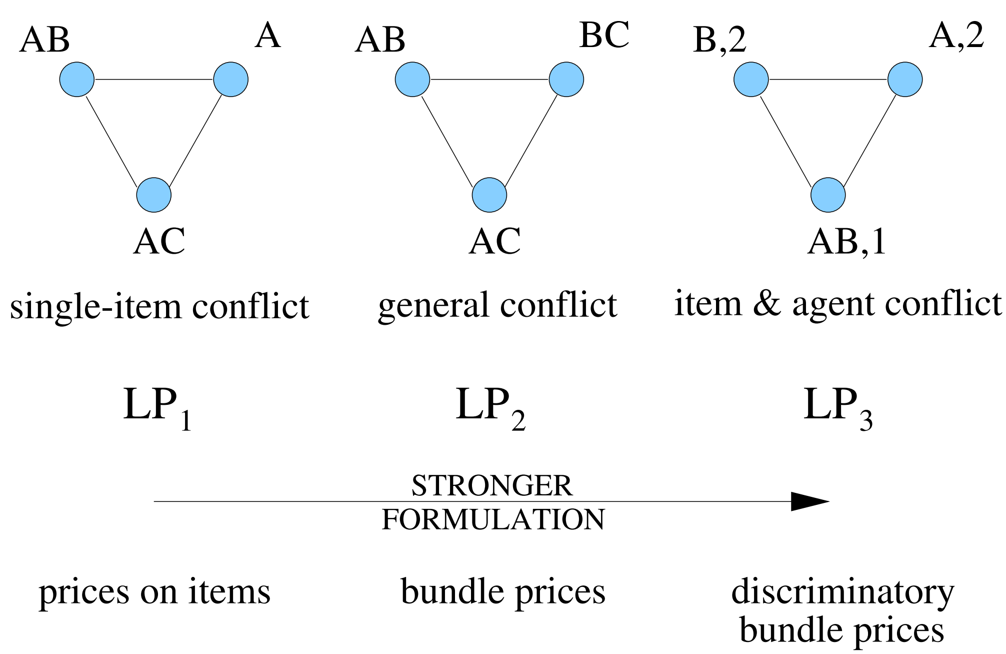
CAP is NP-hard because Weighted Set-Packing reduces to CAP.

Interesting directions: (a) identify tractable special cases; (b) introduce approximations; (c) lift the formulation and design primal-dual methods.

Lifting the Formulation

(Bikchandani & Ostroy 98)

	Variables
Primal	allocation of items to agents
Dual	non-linear, possibly non-anonymous prices



(a) Second-Order Formulation

(i) Introduce Π, the set of all possible partitions of items, and for any partition $k \in \Pi$, write $S \in k$ when bundle S is part of the partition.

(ii) Introduce variables $y(k) \geq 0$, where $y(k)$ indicates the level with which partition k is selected in the solution.

$$\max_{x_i(S), y(k)} \sum_S \sum_i x_i(S) v_i(S) \qquad \text{[LP}_2\text{]}$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \qquad \forall i \qquad (\text{LP}_2\text{-1})$$

$$\sum_i x_i(S) \leq \sum_{k: S \in k} y(k), \qquad \forall S \qquad (\text{LP}_2\text{-2})$$

$$\sum_{k \in \Pi} y(k) \leq 1 \qquad (\text{LP}_2\text{-3})$$

$$x_i(S), y(k) \geq 0, \qquad \forall i, S, k$$

(b) Third-order Formulation

(i) Introduce Γ, the set of all allocations, such that $\gamma \in \Gamma$ defines both a partition and an assignment of bundles; write $[i, S] \in \gamma$ to indicate that S is assigned to agent i.

(ii) Introduce variables, $y(\gamma) \geq 0$, to indicate the level with which allocation γ is selected.

$$\max_{x_i(S), y(\gamma)} \sum_S \sum_i x_i(S) v_i(S) \qquad \text{[LP}_3\text{]}$$

$$\text{s.t.} \quad \sum_S x_i(S) \leq 1, \quad \forall i \qquad \text{(LP}_3\text{-1)}$$

$$x_i(S) \leq \sum_{\gamma:[i,S]\in\gamma} y(\gamma), \quad \forall i, S \qquad \text{(LP}_3\text{-2)}$$

$$\sum_{\gamma\in\Gamma} y(\gamma) \leq 1 \qquad \text{(LP}_3\text{-3)}$$

$$x_i(S), y(\gamma) \geq 0, \quad \forall i, S, \gamma$$

CAP: Examples

	A	B	AB
Agent 1	0	0	3
Agent 2	2^*	0	2
Agent 3	0	2^*	2

	A	B	C	AB	BC	AC	ABC
Agent 1	60	50	50	200^*	100	110	250
Agent 2	50	60	50	110	200	100	255
Agent 3	50	50	75^*	100	125	200	250

	A	B	AB
Agent 1	0	0	3^*
Agent 2	2	2	2

CAP: Examples

	prob 1	prob 2	prob 3
LP_1	4	300	
		$x_1(AB) = 0.5$	
		$x_2(BC) = 0.5$	
LP_2	4	275	3.5
			$x_1(AB) = 0.5$
			$x_2(A) = x_3(B) = 0.5$
LP_3	4	275	3
opt	4	275	3
	$(A, B, \emptyset)$	$(AB, \emptyset, C)$	$(AB, \emptyset)$

$$V_{\mathrm{LP},1} \geq V_{\mathrm{LP},2} \geq V_{\mathrm{LP},3} = V_{\mathrm{IP}}$$

Thm. [integrality] The optimal solution to LP_3 is integral.

$$\min_{\pi_i, p_j} \sum_i \pi_i + \sum_j p_j \qquad \text{[DLP}_1\text{]}$$

$$\text{s.}t. \quad \pi_i + \sum_{j \in S} p_j \geq v_i(S), \qquad \forall i, S \qquad (\text{DLP}_1\text{-1})$$

$$\pi_i, p_j \geq 0, \qquad \forall i, j$$

$$\min_{\pi_i, p(S), \pi^s} \sum_i \pi_i + \pi^s \qquad \text{[DLP}_2\text{]}$$

$$\text{s.}t. \quad \pi_i + p(S) \geq v_i(S), \qquad \forall i, S \qquad (\text{DLP}_2\text{-1})$$

$$\pi^s - \sum_{S \in k} p(S) \geq 0, \qquad \forall k \qquad (\text{DLP}_2\text{-2})$$

$$\pi_i, p(S), \pi^s \geq 0, \qquad \forall i, S$$

$$\min_{\pi_i, p_i(S), \pi^s} \sum_i \pi_i + \pi^s \qquad \text{[DLP}_3\text{]}$$

$$\text{s.}t. \quad \pi_i + p_i(S) \geq v_i(S), \quad \forall i, S \qquad (\text{DLP}_3\text{-1})$$

$$\pi^s - \sum_{[i,S] \in \gamma} p_i(S) \geq 0, \quad \forall \gamma \qquad (\text{DLP}_3\text{-2})$$

$$\pi_i, p_i(S), \pi^s \geq 0, \quad \forall i, S$$

Analysis

- LP_3 is integral
- CE prices always exist (perhaps non-linear, non-anonymous)
- CE prices support Vickrey payments at least seller-preferred outcome
- Leads to iBundle, Extend & Adjust, which is an ascending-price Generalized Vickrey Auction [Parkes& Ungar, 02]

Solving Integer Programs

- Partial-enumeration
- LP relaxation
- Heuristics
- Lagrangian relaxations; primal-dual; etc.

Partial Enumeration Techniques: Branch and Bound

Thesis. LPR retains enough structure of IP to be a useful weak representation.

1. Maintain a queue of subproblems [initialized to the masterproblem], and a *current best solution*.
2. At each stage, select a subproblem and *fix* the value of one of the variables.
3. *Branch* on one variable, b, that is undefined in the subproblem: generate two subproblems, one with $x_b = 0$ and one with $x_b = 1$.
4. *Bound* the value of each subproblem [typically via. an LPR].
   – If the LPR is integral then no further enumeration is required, *update* the current best solution.
   – Otherwise: if the bound is less than the current best solution, *prune*; else add the subproblem to the queue.
5. Whenever the feasible solution changes, prune the queue.
6. *Stop* when there are no problems left in the queue.

Partial Enumeration Techniques: Branch and Cut

Essentially the same as branch-and-bound, except at each stage of the search tree, *generate valid inequalities* (cuts) to strengthen the LPR of the subproblem.

1. Branch on a variable in a subproblem.
2. For a child problem, introduce *cuts* to strengthen the bound computed with a LPR, *and generate cuts that are valid everywhere* in the tree.
3. Introduce the cuts to all current subproblems in the queue.

Note: there are general methods (e.g. Gomory& Chvatal) to generate cuts; and also a “cottage industry” in indentifying useful cuts for particular classes of problems.

Multi-cast Steiner Tree Example

[Vazirani& Jain]

- Prize-collecting Steiner tree problem is NP-hard
  – model for multi-cast routing in a general network
- MCST is a 2-approx
- Dual provides a method to compute a cross-monotonic cost-sharing method
- Implement a group-strategyproof, budget-balanced, approximation mechanism for multi-cast routing in a general network.

Summary

- Very fast methods exist to solve integer programs.
- Introducing sufficient valid inequalities into an IP to solve as an LP is useful because:
  – it permits the use of primal-dual methods
  – the constraints provide “price information”
- The geometry of integer program formulations provides a rich mathematical structure to design approximation algorithms for combinatorial optimization problems.