A Branch-and-Cut approach to solve the Hamiltonian Cycle Problem

Marco Colombo
University of Edinburgh

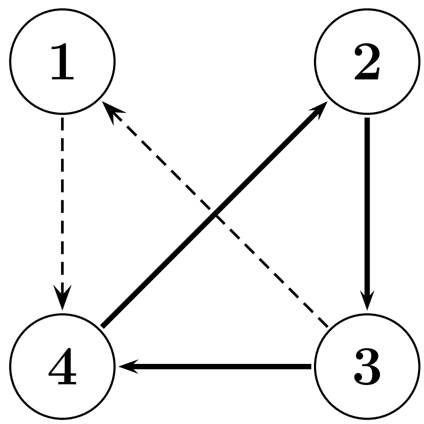
10 March 2004

Contents

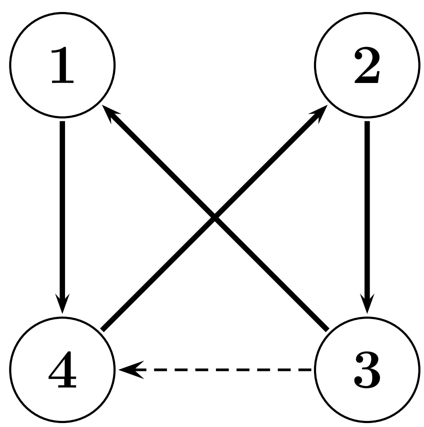
- ▷ Hamiltonian Cycle Problem
- ▷ Cutting Planes and Branch-and-cut
- ▷ Nonconvex quadratic formulation
- ▷ Implementation

Two definitions

A **cycle** is a sequence $v_1, \ldots, v_k$ of distinct vertices such that $(v_i, v_{i+1}) \in E$ and $v_k = v_1$.



Given a graph G, a **Hamiltonian cycle** is a cycle that includes every vertex of G, and each vertex appears exactly once in the cycle.



HCP and TSP

The **Hamiltonian Cycle Problem** asks to find a Hamiltonian cycle in a graph or to state that such a cycle does not exist.

The **Travelling Salesman Problem** asks to find the minimum distance Hamiltonian cycle in a weighted graph.

Main differences:

- ▷ TSP usually involves undirected graphs (symmetric TSP).
- ▷ TSP usually involves highly connected graphs (often complete).

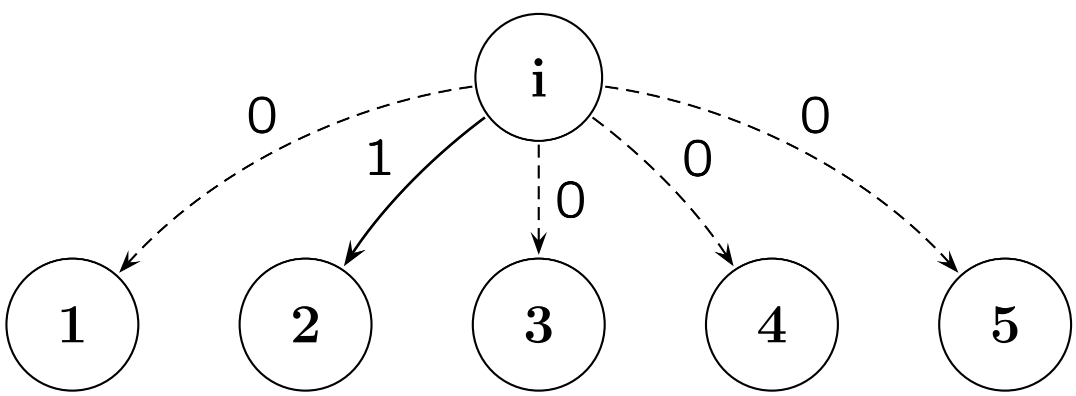
While HCP tries to find “a” Hamiltonian cycle in a graph that contains few of them, TSP tries to find “the” Hamiltonian cycle of minimum distance in a graph that has many of them.

IP perspective

G $= (V, E)$ has m nodes and n edges:

- Choose m edges to be in the cycle and all others $n - m$ edges to be left out.
- Associate a binary variable x_{ij} to each arc:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is used in the HC} \\ 0 & \text{otherwise} \end{cases}$$



- Use the node–arc incidence matrix:

$$A = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & -1 & 0 \\ 1 & 0 & 0 & 1 & -1 \end{bmatrix}$$

IP Formulation

- ▷ One arc enters and one arc leaves each node:

$$Ax = 0$$

- ▷ There are m edges in a cycle:

$$\sum_{i,j \in V} x_{ij} = m$$

- ▷ Each edge can be either used or not:

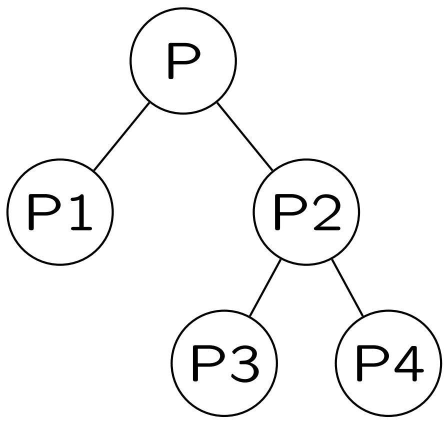
$$x_{ij} \in \{0, 1\}$$

This problem is **intractable** for nontrivial sizes.

- ▷ Relax the integrality constraint and solve the LP relaxation.
- ▷ Deal with nonintegrality...
- ▷ It would be easier if there were less integer variables.

Branch and bound

Partition the problem into smaller problems until these can be solved. This is done by fixing the value of one variable at a time, usually the one with most fractional value.



Bounding procedures allow to remove a subproblem without having to solve it when it is proven that it cannot possibly contain a better solution than the current one found so far.

Can we do something better than dealing with one variable at a time?

Cutting planes (Gomory)

A cutting plane is a constraint with these two properties:

- ▷ Any feasible integer point will satisfy the cut.
- ▷ The optimal solution of the current linear programming relaxation will violate the cut.

This can be embedded in an iterative algorithm:

- ▷ Solve the LP relaxation of the integer problem.
- ▷ If the optimal solution is integer, it solves the IP as well.
- ▷ Generate a cutting plane and append it to the existing constraints.
- ▷ Go back to the first step.

Branch-and-cut

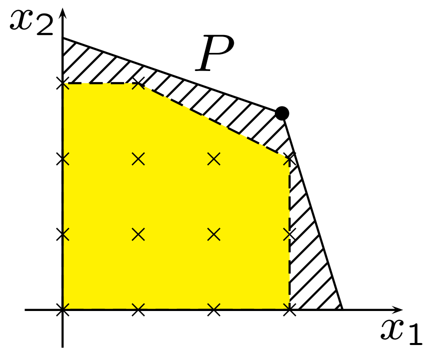
Relies on the same idea as Branch-and-bound: divide the problem into smaller and smaller problems until these can be solved.

The branching does not happen on a variable at a time. Instead (disjunctive) cutting planes are added to the problem.

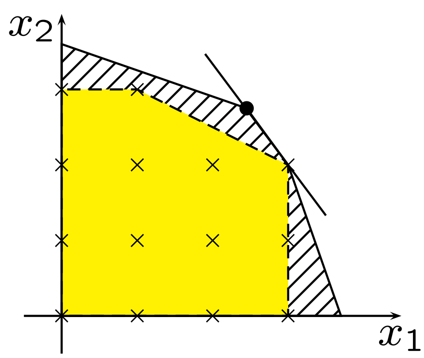
Therefore, each node in the branching tree generates two sons:

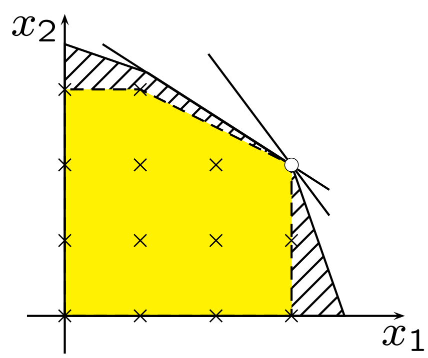
- ▷ $P_1 = P \cap h_1, \qquad h_1 = a^T x \leq b - 1$
- ▷ $P_2 = P \cap h_2, \qquad h_2 = a^T x \geq b$

Some pictures

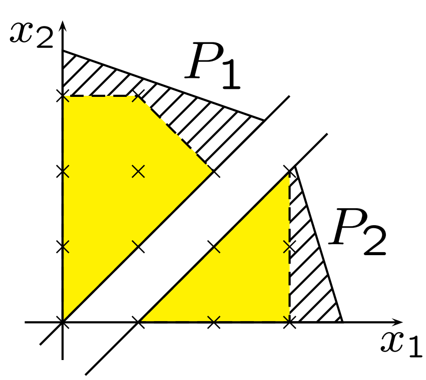


Two Gomory cutting planes:





Two disjunctive cuts:



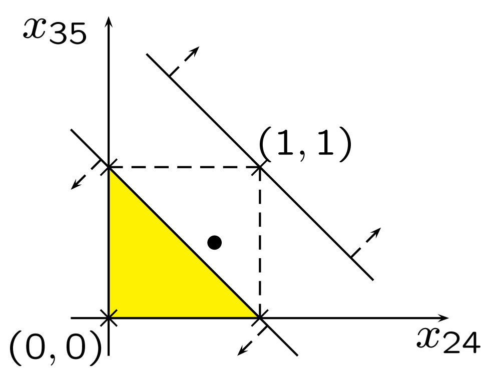
Example for BIP

Suppose the current solution contains (among others) these two fractional variables:

$$x_{24} = 0.7 \qquad x_{35} = 0.5$$

Introduce two cuts:

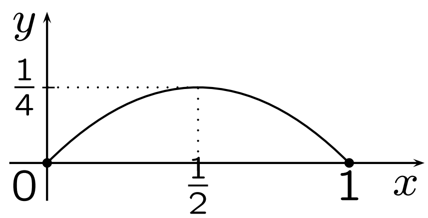
$$x_{24} + x_{35} \leq 1 \quad \text{and} \quad x_{24} + x_{35} \geq 2.$$



The cuts remove noninteger points and leave all integer points untouched.

Nonconvex QP approach

Relax the integrality constraint on variable x and express it as a continuous variable $0 \le x \le 1$ such that $x(1-x) = 0$.



In the interval $[0, 1]$ the function is non-negative, and attains its minimum at the extreme points, which have **integer** coordinates.

- ▷ This setup is **nonconvex**.
- ▷ It's not possible to use the simplex.
- ▷ Also IPMs struggle if the nonconvexity is too large.

Exploiting the nonconvex QP approach

$\mathcal{A}(i) = \{j \mid (i,j) \in E\}$: (out)-neighbours of i.

Choose only one node among those in $\mathcal{A}(i)$:

$$\sum_{j \in \mathcal{A}(i)} x_{ij} = 1.$$

Now consider the following quantity:

$$\left(\sum_{j \in \mathcal{A}(i)} x_{ij} \right)^2 - \sum_{j \in \mathcal{A}(i)} x_{ij}^2 = \sum_{k \neq l} x_{ik} x_{il} \geq 0$$

- ▷ If more than one variable has positive value, the term is positive.
- ▷ When **exactly one** variable is positive, the term in zero.
- ▷ Minimize this term!

Matrix notation

Cross–product term: $\displaystyle\sum_{\substack{k,l\in\mathcal{A}(i)\\ k\neq l}} x_{ik}x_{il}$

Matrix notation:

$$\begin{aligned}(e^T x_i)^2 - x_i^T x_i &= x_i^T e e^T x_i - x_i^T x_i\\ &= x_i^T (ee^T - I) x_i\\ &= x_i^T Q_i x_i\end{aligned}$$

$$Q_i = \begin{bmatrix} 0 & 1 & \cdots & 1\\ 1 & 0 & & 1\\ \vdots & & \ddots & \\ 1 & 1 & & 0\end{bmatrix},\ x_i = \begin{bmatrix} x_{i1}\\ x_{i2}\\ \vdots\\ x_{in_i}\end{bmatrix},\ e = \begin{bmatrix}1\\1\\ \vdots\\ 1\end{bmatrix}$$

Matrix Q_i is an $n_i \times n_i$ matrix ($n_i = |\mathcal{A}(i)|$).

$$x_i^T Q_i x_i \begin{cases} = 0 & \text{at most one } j \in \mathcal{A}(i) \text{ is chosen}\\ > 0 & \text{all other cases}\end{cases}$$

Penalty term

Apply the same procedure to all nodes in the graph:

$$\sum_{i \in V} x_i^T Q_i x_i = x^T Q x,$$

where

$$Q = \begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_m \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}.$$

- ▷ Use $x^T Q x$ as a penalty term to penalize fractional solutions.
- ▷ Objective function:

  $$\min x^T Q x$$

  leads us to assign integer values to the elements of x.
- ▷ Matrix Q is not very sparse.

A sparser reformulation

Introduce an auxiliary variable y_i for each node:

$$y_i = x_i^T e = \sum_{j \in \mathcal{A}(i)} x_{ij}$$

$$x_i^T Q_i x_i = (x_i^T e)(e^T x_i) - x_i^T x_i = y_i^2 - x_i^T x_i$$

$$\tilde{Q}_i = \begin{bmatrix} -1 & & & \\ & \ddots & & \\ & & -1 & \\ & & & 1 \end{bmatrix} \quad \text{and} \quad \tilde{x}_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{in_i} \\ y_i \end{bmatrix}$$

- ▷ Matrix $\tilde{Q}_i$ is much sparser: $n_i + 1$ nonzeros intead of $n_i(n_i - 1)$.
- ▷ $\tilde{Q} = \text{diag}(\tilde{Q}_i)$ is now a $(n+m) \times (n+m)$ matrix.
- ▷ The formulation is separable:

$$\min \sum_{i \in V} x_i^T \tilde{Q}_i x_i = \min x^T \tilde{Q} x$$

Complete problem formulation

$$
\begin{aligned}
&\min \quad \tilde{x}^T \tilde{Q} \tilde{x} \\
\text{s.t.} \quad & Ax = 0 \\
& \sum_{j \in \mathcal{A}(i)} x_{ij} - y_i = 0 \qquad i \in V \\
& y_i = 1 \qquad i \in V \\
& x \geq 0
\end{aligned}
$$

To control nonconvexity:

$$x_i^T Q_i x_i = y_i^2 - \alpha x_i^T x_i$$

▷ Choose a small α (e.g. $\alpha = 0.01$).

Generating cutting planes

Assume that we know how to choose some of the existing arcs to appear in the cut.

- ▷ As we want the flow carried by the arcs in the cut to be integer, the right-hand side must also be an integer number;
- ▷ As we do not allow to remove any integer solution, the difference in right-hand side for the two cuts must be 1.

Evaluate the total flow T shipped through the arcs in the cut from the latest solution of system.

Discard the current fractional solution by asking:

$$\begin{aligned}\text{arcs in cut} &\geq \lceil T \rceil,\\ \text{arcs in cut} &\leq \lfloor T \rfloor.\end{aligned}$$

A naive cut

Choose two edges such that the sum of their flows is fractional:

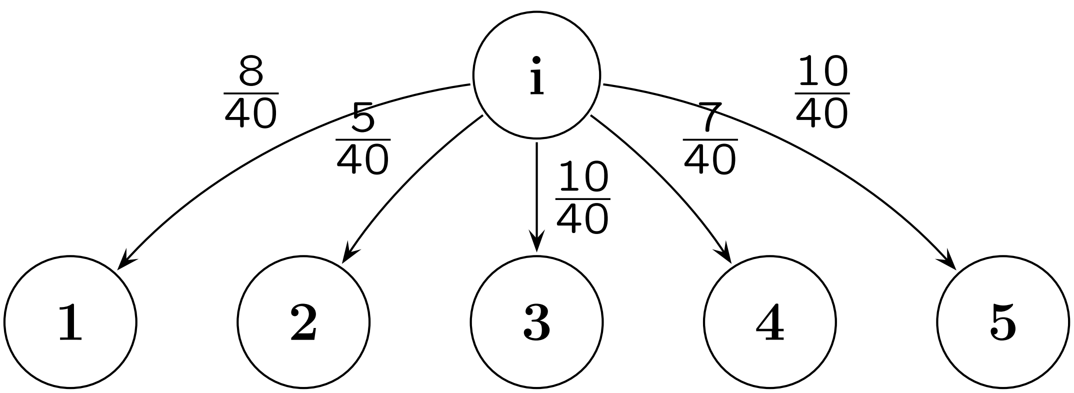
Solution	1	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
Cut	0	1	0	1	0	0	0	0	0	0

Solution	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
Cut	0	1	0	0	0	0	0	0	0	0

- ▷ Easy to implement.
- ▷ Fast.
- ▷ Uses only the current solution.

Cut based on nodes

Choose a node: put some of the outgoing edges in the cut so that the sum of their flow is at least 0.5 but not too close to 1.



- ▷ Other implementations possible.
- ▷ Based on one node or on multiple nodes.
- ▷ Exploits the graph structure and the current solution.

Knight's tour problem

1	2	3	**4**
5	**6**	7	8
9	10	11	**12**
13	14	**15**	16

- ▷ A node for each square of the chess board.
- ▷ An arc between two squares that are linked by a knight's move.

Problem	Nodes	Arcs
`chess8`	64	336
`chess10`	100	576
`chess12`	144	880
`chess14`	196	1248
`chess20`	400	2736
`chess32`	1024	7440

Implementations tested

Formulations:

- ▷ Full QP approach:

$$\min \sum_{i \in V} \tilde{x}_i^T \tilde{Q}_i \tilde{x}_i$$

- ▷ Partial QP approach: $I \subset V$

$$\min \sum_{i \in I} \tilde{x}_i^T \tilde{Q}_i \tilde{x}_i, \quad \text{for some } I \subset V$$

- ▷ Linear approach:

$$\min\ e^T x.$$

Cuts:

- ▷ A naive cut;
- ▷ A cut based on a single node;
- ▷ A cut based on multiple nodes.

Results

NAIVE CUT:

Problem	Partial QP			Linear		
	Prb	Lev	Time	Prb	Lev	Time
chess8	16	14	1	33	31	1
chess10	24	23	2	52	51	2
chess12	53	36	7	87	82	7
chess14	68	51	15	126	125	16
chess20	122	114	80	283	279	100
chess32	342	313	942	890	851	1477

SINGLE NODE:

Problem	Partial QP			Linear		
	Prb	Lev	Time	Prb	Lev	Time
chess8	24	17	1	24	23	1
chess10	35	28	3	55	47	3
chess12	51	45	8	94	77	7
chess14	74	59	17	120	106	16
chess20	193	143	117	278	259	108
chess32	-	-	-	706	687	1238

Conclusions and future work

- ▷ The QP approach provides a boost towards integrality.
- ▷ Disjunctive cuts alone are not enough.

- ▷ Strong cutting planes to tackle larger problems.
- ▷ Branch less!
- ▷ Recovery from subcycles.
- ▷ Heuristic choices for the partial QP.