南京大学 NANJING UNIVERSITY

Websoft

3-13 松弛算法

程龚

你还记得LP吗？

- Standard form

In Section 2.3.2 we introduced the linear programming problem as to minimize

$$c^{\mathsf{T}} \cdot X = \sum_{i=1}^{n} c_i x_i$$

under the constraints

$$A \cdot X = b, \text{ i.e., } \sum_{i=1}^{n} a_{ji} x_i = b_j \text{ for } j = 1, \ldots, m.$$

$$x_i \geq 0 \text{ for } i = 1, \ldots, n \text{ (i.e., } X \in (\mathbb{R}^{\geq 0})^n)$$

for every input instance $A = [a_{ji}]_{j=1,\ldots,m, i=1,\ldots,n}$, $b = (b_1, \ldots, b_m)^{\mathsf{T}}$, and $c = (c_1, \ldots, c_n)^{\mathsf{T}}$ over reals.

你还记得LP吗？

- Canonical form

The **canonical form** of LP is to minimize

$$c^{\mathsf{T}} \cdot X = \sum_{i=1}^{n} c_i \cdot x_i$$

under the constraints

$$AX \geq b, \text{ i.e., } \sum_{i=1}^{n} a_{ji} x_i \geq b_j \text{ for } j = 1, \ldots, m$$

$$x_i \geq 0 \text{ for } i = 1, \ldots, n$$

for every input instance (A, b, c).

你还记得LP吗？

- Standard inequality form

Next, we consider the **standard inequality form** of LP. One has to minimize

$$c^{\mathsf{T}} \cdot X = \sum_{i=1}^{n} c_i x_i$$

under the constraints

$$AX \leq b, \text{ i.e., } \sum_{i=1}^{n} a_{ji} x_i \leq b_j \text{ for } j = 1, \ldots, m, \text{ and}$$

$$x_i \geq 0 \text{ for } i = 1, \ldots, n$$

for a given instance A, b, c.

你还记得IP吗？

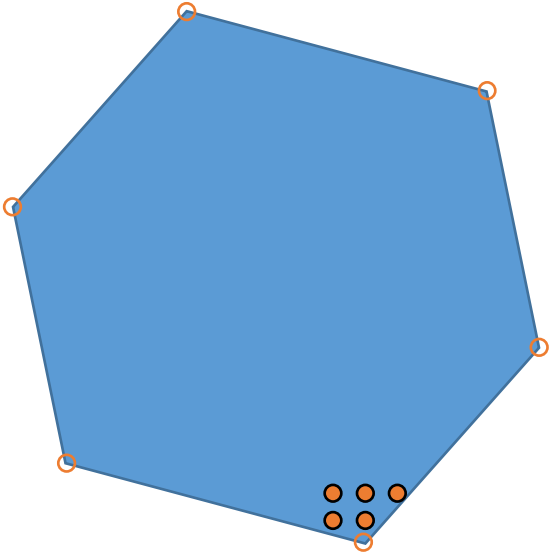
你还记得IP吗？

Remember that the integer linear programming problem is defined in the same way except that all coefficients are integers and the solution $X \in \mathbb{Z}^n$.

为什么IP比LP难？

Remember that the integer linear programming problem is defined in the same way except that all coefficients are integers and the solution $X \in \mathbb{Z}^n$.

为什么IP比LP难？



你能解释这些概念吗？

- Relaxation

你能解释这些概念吗？

- Relaxation

- … the computed optimal solution for LP does not need to be a feasible solution for IP …，这样做有什么用处呢？

你能解释这些概念吗？

- Relaxation

- … the computed optimal solution for LP does not need to be a feasible solution for IP …，这样做有什么用处呢？

First of all, $cost(\alpha)$ is a lower bound[25] on the cost of the optimal solutions with respect to 0/1-LP and IP. Thus, $cost(\alpha)$ can be a good approximation of the cost of the optimal solutions of the original problem. This combined with the prima-dual method in Section 3.7.4 can be very helpful as a precomputation for a successful application of the branch-and-bound method as described in Section 3.4. Another possibility is to use α to compute a solution β that is feasible for 0/1-LP or IP. This can be done, for instance, by (randomized) rounding of real values to the values 0 and 1 or to positive integers. For some problems, such a feasible solution β is of high quality in the sense that it is reasonably close to an optimal solution. Some examples of such successful applications are given in Chapters 4 and 5.

你能解释这些概念吗？

- Relaxation

- … the computed optimal solution for LP does not need to be a feasible solution for IP …，这样做有什么用处呢？

(1) *Reduction*
Express a given instance x of an optimization problem U as an input instance $I(x)$ of 0/1-LP or IP.
(2) *Relaxation*
Consider $I(x)$ as an instance of LP and compute an optimal solution α to $I(x)$ by an algorithm for linear programming.
(3) *Solving the original problem*
Use α to either compute an optimal solution for the original problem, or to find a high-quality feasible solution for the original problem.

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

Thus, the task is to maximize

$$\sum_{i=1}^{n} c_i x_i$$

under the constraints

$$\sum_{i=1}^{n} w_i x_i \leq b, \text{ and}$$

$$x_i \in \{0, 1\} \text{ for } i = 1, \ldots, n.$$

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

$$\text{minimize} \sum_{i=1}^{m} x_i$$

under the following n linear constraints

$$\sum_{j \in \text{Index}(k)} x_j \geq 1 \quad \text{for } k = 1, \ldots, n.$$

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

$$
\begin{aligned}
\text{minimize} \quad & t \\
\text{subject to} \quad & \sum_{i \in M} x_{ij} = 1, && j \in J \\
& \sum_{j \in J} x_{ij} p_{ij} \leq t, && i \in M \\
& x_{ij} \in \{0, 1\}, && i \in M,\ j \in J
\end{aligned}
$$

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

$$\text{maximize} \sum_{j=1}^{m} z_j$$

subject to the following $2m + n$ constraints

$$z_j - \sum_{i \in \text{In}^+(F_j)} x_i - \sum_{l \in \text{In}^-(F_j)} (1 - x_l) \leq 0 \text{ for } j = 1, \ldots, m$$

$$x_i \in \{0, 1\} \text{ for } i = 1, \ldots, n$$

$$z_j \in \{0, 1\} \text{ for } j = 1, \ldots, m.$$

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

minimize

$$\sum_{i=1}^{n} c(v_i) \cdot x_i.$$

$x_i \in \{0, 1\}$

$x_i + x_j \geq 1$ for every $\{v_i, v_j\} \in E$

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

Now, the task is to maximize

$$\sum_{e \in E} x_e$$

under the $|V|$ constraints

$$\sum_{e \in E(v)} x_e \leq 1 \quad \text{for every } v \in V,$$

and the following $|E|$ constraints

$$x_e \in \{0, 1\} \quad \text{for every } e \in E.$$

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

$$\max \sum_{i=1}^{n} w_i x_i,$$

$$\text{s.t. } x_i + x_j \leq 1, \ \forall \ (i,j) \in \overline{E},$$

$$x_i \in \{0,1\}, \ i = 1, \ldots, n.$$

你会将一些常见问题reduce到0/1-LP吗？

- KP
- SCP
- MS
- MAX-SAT
- WEIGHT-VCP
- MMP
- WEIGHT-CL
- TSP

将TSP reduce到0/1-LP

minimize the tour length $\sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}$.

Without further constraints, the $\{x_{ij}\}_{i,j}$ will however effectively range over all subsets of the set of edges, which is very far from the sets of edges in a tour, and allows for a trivial minimum where all $x_{ij} = 0$. Therefore, both formulations also have the constraints that there at each vertex is exactly one incoming edge and one outgoing edge, which may be expressed as the $2n$ linear equations

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \text{ for } j = 1, \ldots, n \text{ and } \sum_{j=1, j \neq i}^{n} x_{ij} = 1 \text{ for } i = 1, \ldots, n.$$

These ensure that the chosen set of edges locally looks like that of a tour, but still allow for solutions violating the global requirement that there is *one* tour which visits all vertices, as the edges chosen could make up several tours each visiting only a subset of the vertices; arguably it is this global requirement that makes TSP a hard problem. The MTZ and DFJ formulations differ in how they express this final requirement as linear constraints.

In addition to the x_{ij} variables as above, there is for each $i = 1, \ldots, n$ a dummy variable u_i that keeps track of the order in which the cities are visited, counting from city 1; the interpretation is that $u_i < u_j$ implies city i is visited before city j. For a given tour (as encoded into values of the x_{ij} variables), one may find satisfying values for the u_i variables by making u_i equal to the number of edges along that tour, when going from city 1 to city i.

Because linear programming favours non-strict inequalities ($\geq$) over strict ($>$), we would like to impose constraints to the effect that

$$u_j \geq u_i + 1 \text{ if } x_{ij} = 1.$$

Merely requiring $u_j \geq u_i + x_{ij}$ would *not* achieve that, because this also requires $u_j \geq u_i$ when $x_{ij} = 0$, which is not correct. Instead MTZ use the $n(n-1)$ linear constraints

$$u_j + (n-1) \geq u_i + nx_{ij} \text{ for all distinct } i, j \in \{2, \ldots, n\}$$

where the constant term $n - 1$ provides sufficient slack that $x_{ij} = 0$ does not impose a relation between u_j and u_i.

The way that the u_i variables then enforce that a single tour visits all cities is that they increase by (at least) 1 for each step along a tour, with a decrease only allowed where the tour passes through city 1. That constraint would be violated by every tour which does not pass through city 1, so the only way to satisfy it is that the tour passing city 1 also passes through all other cities.

The MTZ formulation of TSP is thus the following integer linear programming problem:

$$\begin{aligned}
\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} : & \\
x_{ij} \in \{0, 1\} \qquad & i, j = 1, \ldots, n; \\
\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad & j = 1, \ldots, n; \\
\sum_{j-1, j \neq i}^{n} x_{ij} = 1 \qquad & i = 1, \ldots, n; \\
u_i - u_j + n x_{ij} \leq n - 1 \qquad & 2 \leq i \neq j \leq n; \\
1 \leq u_i \leq n \qquad & 1 \leq i \leq n.
\end{aligned}$$

The first set of equalities requires that each city is arrived at from exactly one other city, and the second set of equalities requires that from each city there is a departure to exactly one other city. The last constraints enforce that there is only a single tour covering all cities, and not two or more disjointed tours that only collectively cover all cities.

你能解释这些概念吗？

- Rounding

你能解释这些概念吗？

- Rounding

use an optimal solution α of the relaxed problem instance to get a reasonable feasible (integral or Boolean) solution β of the original problem instance.

你能解释这些概念吗？

- Rounding

use an optimal solution α of the relaxed problem instance to get a reasonable feasible (integral or Boolean) solution β of the original problem instance.

- 什么是一种好的rounding？

你能解释这些概念吗？

- Rounding

use an optimal solution α of the relaxed problem instance to get a reasonable feasible (integral or Boolean) solution β of the original problem instance.

- 什么是一种好的rounding？

the obtained rounded integral solution is a feasible solution of the original input instance and that the cost has not been changed too much in the rounding processes.

你理解这个算法了吗？

Algorithm 3.7.4.2. Input: An instance $(X, \mathcal{F})$, $X = \{a_1, \ldots, a_n\}$, $\mathcal{F} = \{S_1, \ldots, S_m\}$ of SCP(k).

Step 1: "Reduction"
Express $(X, \mathcal{F})$ as an instance $I(X, \mathcal{F})$ of 0/1-LP in the way described above.

Step 2: "Relaxation"
Relax $I(X, \mathcal{F})$ to an instance LP$(X, \mathcal{F})$ of LP by relaxing $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$ for every $i = 1, \ldots, m$.
Solve LP$(X, \mathcal{F})$ by an algorithm for linear programming.
Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ [i.e., $x_i = \alpha_i$] be an optimal solution for LP$(X, \mathcal{F})$.

Step 3: "Solving the original problem"
Set $\beta_i = 1$ iff $\alpha_i \geq 1/k$.

Output: $\beta = (\beta_1, \ldots, \beta_m)$.

$$\text{minimize} \sum_{i=1}^{m} x_i$$

under the following n linear constraints

$$\sum_{j \in \text{Index}(k)} x_j \geq 1 \quad \text{for } k = 1, \ldots, n.$$

你理解这个算法了吗?

Algorithm 3.7.4.2. Input: An instance $(X, \mathcal{F})$, $X = \{a_1, \ldots, a_n\}$, $\mathcal{F} = \{S_1, \ldots, S_m\}$ of SCP(k).

Step 1: "Reduction"
Express $(X, \mathcal{F})$ as an instance $I(X, \mathcal{F})$ of 0/1-LP in the way described above.

Step 2: "Relaxation"
Relax $I(X, \mathcal{F})$ to an instance LP$(X, \mathcal{F})$ of LP by relaxing $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$ for every $i = 1, \ldots, m$.
Solve LP$(X, \mathcal{F})$ by an algorithm for linear programming.
Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ [i.e., $x_i = \alpha_i$] be an optimal solution for LP$(X, \mathcal{F})$.

Step 3: "Solving the original problem"
Set $\beta_i = 1$ iff $\alpha_i \geq 1/k$.

Output: $\beta = (\beta_1, \ldots, \beta_m)$.

- 为什么β是可行解?
- 与最优解的cost相差多少?

$$\text{minimize} \sum_{i=1}^{m} x_i$$

under the following n linear constraints

$$\sum_{j \in \text{Index}(k)} x_j \geq 1 \quad \text{for } k = 1, \ldots, n.$$

你理解这个算法了吗？

Algorithm 3.7.4.2. Input: An instance $(X, \mathcal{F})$, $X = \{a_1, \ldots, a_n\}$, $\mathcal{F} = \{S_1, \ldots, S_m\}$ of SCP(k).

Step 1: "Reduction"
Express $(X, \mathcal{F})$ as an instance $I(X, \mathcal{F})$ of 0/1-LP in the way described above.

Step 2: "Relaxation"
Relax $I(X, \mathcal{F})$ to an instance LP$(X, \mathcal{F})$ of LP by relaxing $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$ for every $i = 1, \ldots, m$.
Solve LP$(X, \mathcal{F})$ by an algorithm for linear programming.
Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ [i.e., $x_i = \alpha_i$] be an optimal solution for LP$(X, \mathcal{F})$.

Step 3: "Solving the original problem"
Set $\beta_i = 1$ iff $\alpha_i \geq 1/k$.

Output: $\beta = (\beta_1, \ldots, \beta_m)$.

- 为什么β是可行解?
- 与最优解的cost相差多少?

$$\beta_i \leq k \cdot \alpha_i$$

$$cost(\beta) \leq k \cdot cost(\alpha)$$

$$\text{minimize} \sum_{i=1}^{m} x_i$$

under the following n linear constraints

$$\sum_{j \in \text{Index}(k)} x_j \geq 1 \quad \text{for } k = 1, \ldots, n.$$

你能给出WEIGHT-VCP的relaxation算法吗？

- 如何rounding？
  - 为什么是可行解？
  - 与最优解的cost相差多少？

minimize

$$\sum_{i=1}^{n} c(v_i) \cdot x_i.$$

$x_i \in \{0, 1\}$

$x_i + x_j \geq 1$ for every $\{v_i, v_j\} \in E$

你能给出MMP的relaxation算法吗？

- 如何rounding？
  - 为什么是可行解？
  - 与最优解的cost相差多少？

Now, the task is to maximize

$$\sum_{e \in E} x_e$$

under the $|V|$ constraints

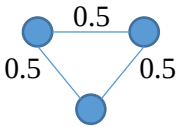
$$\sum_{e \in E(v)} x_e \leq 1 \quad \text{for every } v \in V,$$

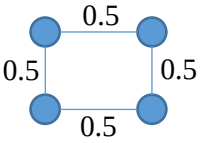
and the following $|E|$ constraints

$$x_e \in \{0, 1\} \quad \text{for every } e \in E.$$

你能给出MMP的relaxation算法吗？

- 如何rounding？
  - 为什么是可行解？
  - 与最优解的cost相差多少？





Now, the task is to maximize

$$\sum_{e \in E} x_e$$

under the $|V|$ constraints

$$\sum_{e \in E(v)} x_e \leq 1 \quad \text{for every } v \in V,$$

and the following $|E|$ constraints

$$x_e \in \{0, 1\} \quad \text{for every } e \in E.$$

你能给出MAX-SAT的relaxation算法吗?

- 如何rounding?
  - 为什么是可行解?
  - 与最优解的cost相差多少?

$$\text{maximize} \sum_{j=1}^{m} z_j$$

subject to the following $2m + n$ constraints

$$z_j - \sum_{i \in \mathrm{In}^+(F_j)} x_i - \sum_{l \in \mathrm{In}^-(F_j)} (1 - x_l) \leq 0 \text{ for } j = 1, \ldots, m$$
$$x_i \in \{0, 1\} \text{ for } i = 1, \ldots, n$$
$$z_j \in \{0, 1\} \text{ for } j = 1, \ldots, m.$$

你能给出MAX-SAT的relaxation算法吗？

- 如何rounding？
  - 为什么是可行解？
  - 与最优解的cost相差多少？
- 考虑：按概率随机rounding

$$\text{maximize} \sum_{j=1}^{m} z_j$$

subject to the following $2m + n$ constraints

$$z_j - \sum_{i \in \text{In}^+(F_j)} x_i - \sum_{l \in \text{In}^-(F_j)} (1 - x_l) \leq 0 \text{ for } j = 1, \ldots, m$$
$$x_i \in \{0, 1\} \text{ for } i = 1, \ldots, n$$
$$z_j \in \{0, 1\} \text{ for } j = 1, \ldots, m.$$

你能给出WEIGHT-VCP的relaxation算法吗？

- 如何rounding？
  - 为什么是可行解？
  - 与最优解的cost相差多少？
- 也考虑：按概率随机rounding？

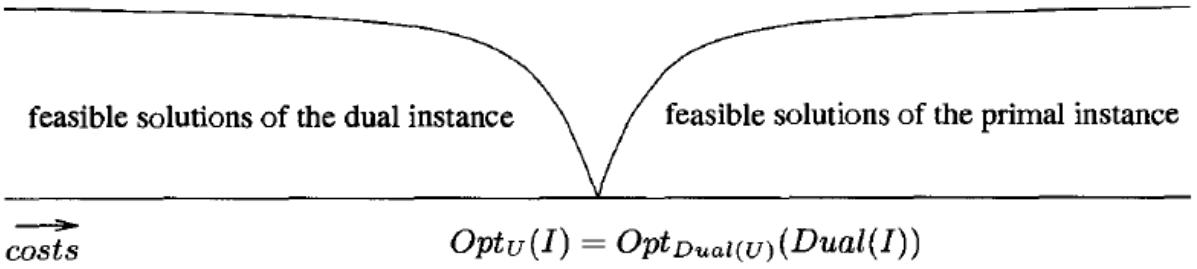
$$\text{minimize} \quad \sum_{i=1}^{n} c(v_i) \cdot x_i.$$

$$x_i \in \{0,1\}$$

$$x_i + x_j \geq 1 \text{ for every } \{v_i, v_j\} \in E$$

你能解释这些概念吗？

- Primal problem
- Dual problem
  - 在福特-法尔克森算法中是如何体现的？



你能解释这些概念吗？

- Primal problem
- Dual problem
  - 在福特-法尔克森算法中是如何体现的？

(i) $Dual(U)$ is a maximization [minimization] problem that can be obtained by converting every instance I of U to an instance $Dual(I)$ of $Dual(U)$. There should exist an efficient algorithm for computing $Dual(I)$ for a given I.

(ii) For every instance I of U, the cost of any feasible solution for U is not smaller [not greater] than the cost of any feasible solution for $Dual(U)$, i.e.,

$$cost(\alpha) \geq cost(\beta) \ [cost(\alpha) \leq cost(\beta)]$$
for all $\alpha \in \mathcal{M}(I)$ and for all $\beta \in \mathcal{M}(Dual(I))$.

(iii) For every instance I of U,

$$Opt_U(I) = Opt_{Dual(U)}(Dual(I)).$$

你能解释这些概念吗？

- Primal problem
- Dual problem
  - 在福特-法尔克森算法中是如何体现的？
- LP-duality
  - LP的primal problem和dual problem有何异同？

The **canonical form** of LP is to minimize

$$c^{\mathsf{T}} \cdot X = \sum_{i=1}^{n} c_i \cdot x_i$$

under the constraints

$$AX \geq b, \text{ i.e., } \sum_{i=1}^{n} a_{ji} x_i \geq b_j \text{ for } j = 1, \ldots, m$$

$$x_i \geq 0 \text{ for } i = 1, \ldots, n$$

for every input instance (A, b, c).

$$\text{maximize } \sum_{j=1}^{m} b_j y_j$$

under the constraints

$$\sum_{j=1}^{m} a_{ji} y_j \leq c_j \quad \text{for } i = 1, \ldots, n$$

$$y_j \geq 0 \quad \text{for } j = 1, \ldots, m.$$

你理解这个定理及其证明了吗？

Theorem 3.7.4.6 (Weak LP-Duality Theorem). *Let I be an instance of* LP *in the canonical form as given by (3.16) and (3.17). Let $Dual(I)$ be the dual instance of I given by (3.18) and (3.19). Then, for every feasible solution $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathcal{M}(I)$ and every feasible solution $\beta = (\beta_1, \ldots, \beta_m) \in \mathcal{M}(Dual(I))$,*

$$cost(\alpha) = \sum_{i=1}^{n} c_i \alpha_i \geq \sum_{j=1}^{m} b_j \beta_j = cost(\beta),$$

$$i.e., \; Opt_{\mathrm{LP}}(I) \geq Opt_{\mathrm{LP}}(Dual(I)).$$

The **canonical form** of LP is to minimize

$$c^{\mathrm{T}} \cdot X = \sum_{i=1}^{n} c_i \cdot x_i$$

under the constraints

$$AX \geq b, \text{ i.e., } \sum_{i=1}^{n} a_{ji} x_i \geq b_j \text{ for } j = 1, \ldots, m$$

$$x_i \geq 0 \text{ for } i = 1, \ldots, n$$

for every input instance (A, b, c).

$$\text{maximize } \sum_{j=1}^{m} b_j y_j$$

under the constraints

$$\sum_{j=1}^{m} a_{ji} y_j \leq c_j \quad \text{for } i = 1, \ldots, n$$

$$y_j \geq 0 \quad \text{for } j = 1, \ldots, m.$$

Proof. Since $\beta \in \mathcal{M}(Dual(I))$ is a feasible solution for $Dual(I)$,

$$\sum_{j=1}^{m} a_{ji} \beta_j \leq c_i \quad \text{for } i = 1, \ldots, n. \tag{3.20}$$

Since $\alpha_i \geq 0$ for $i = 1, \ldots, n$, the constraints (3.20) imply

$$cost(\alpha) = \sum_{i=1}^{n} c_i \alpha_i \geq \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ji} \beta_j \right) \alpha_i. \tag{3.21}$$

Similarly, because $\alpha \in \mathcal{M}(I)$,

$$\sum_{i=1}^{n} a_{ji} \alpha_i \geq b_j \quad \text{for } j = 1, \ldots, m$$

and so[30]

$$cost(\beta) = \sum_{j=1}^{m} b_j \beta_j \leq \sum_{j=1}^{m} \left(\sum_{i=1}^{n} a_{ji} \alpha_i \right) \beta_j. \tag{3.22}$$

Since

$$\sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ji} \beta_j \right) \alpha_i = \sum_{j=1}^{m} \left(\sum_{i=1}^{n} a_{ji} \alpha_i \right) \beta_j,$$

the inequalities (3.21) and (3.22) imply

$$cost(\alpha) \geq cost(\beta).$$

你理解这个定理及其证明了吗?

Theorem 3.7.4.6 (Weak LP-Duality Theorem). *Let I be an instance of* LP *in the canonical form as given by (3.16) and (3.17). Let $Dual(I)$ be the dual instance of I given by (3.18) and (3.19). Then, for every feasible solution $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathcal{M}(I)$ and every feasible solution $\beta = (\beta_1, \ldots, \beta_m) \in \mathcal{M}(Dual(I))$,*

$$cost(\alpha) = \sum_{i=1}^{n} c_i \alpha_i \geq \sum_{j=1}^{m} b_j \beta_j = cost(\beta),$$

i.e., $Opt_{\mathrm{LP}}(I) \geq Opt_{\mathrm{LP}}(Dual(I))$.

Theorem 3.7.4.8 (LP-Duality Theorem). *Let I and $Dual(I)$ be instances of* LP *as described in Theorem 3.7.4.6. Then*

$$Opt_{\mathrm{LP}}(I) = Opt_{\mathbf{LP}}(Dual(I)).$$

你会构造LP-duality了吗？

The **canonical form** of LP is to minimize

$$c^{\mathsf{T}} \cdot X = \sum_{i=1}^{n} c_i \cdot x_i$$

under the constraints

$$AX \geq b, \text{ i.e., } \sum_{i=1}^{n} a_{ji} x_i \geq b_j \text{ for } j = 1, \ldots, m$$

$$x_i \geq 0 \text{ for } i = 1, \ldots, n$$

for every input instance (A, b, c).

$$\text{maximize } \sum_{j=1}^{m} b_j y_j$$

under the constraints

$$\sum_{j=1}^{m} a_{ji} y_j \leq c_j \quad \text{for } i = 1, \ldots, n$$

$$y_j \geq 0 \quad \text{for } j = 1, \ldots, m.$$

你会构造LP-duality了吗？

- In the case of the standard maximization form

maximize $f(x_1, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

under the constraints

$$\begin{aligned} \textstyle\sum_{i=1}^{n} a_{ji}x_i \le b_j \quad & \text{for } j = 1, \ldots, m, \\ x_i \ge 0 \quad & \text{for } i = 1, \ldots, n, \end{aligned}$$

minimize $b_1y_1 + b_2y_2 + \ldots + b_my_m$

under the constraints

$$\begin{aligned} \textstyle\sum_{j=1}^{m} a_{ji}y_j \ge c_i \quad & \text{for } i = 1, \ldots, n, \\ y_j \ge 0 \quad & \text{for } j = 1, \ldots, m. \end{aligned}$$

The **canonical form** of LP is to minimize

$$c^{\mathsf{T}} \cdot X = \sum_{i=1}^{n} c_i \cdot x_i$$

under the constraints

$$AX \ge b, \text{ i.e., } \sum_{i=1}^{n} a_{ji}x_i \ge b_j \text{ for } j = 1, \ldots, m$$

$$x_i \ge 0 \text{ for } i = 1, \ldots, n$$

for every input instance (A, b, c).

maximize $\sum_{j=1}^{m} b_j y_j$

under the constraints

$$\begin{aligned} \sum_{j=1}^{m} a_{ji}y_j \le c_j \quad & \text{for } i = 1, \ldots, n \\ y_j \ge 0 \quad & \text{for } j = 1, \ldots, m. \end{aligned}$$

你会构造LP-duality了吗？

- In the case of the standard maximization form

maximize $f(x_1, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

under the constraints

$$\begin{aligned} \textstyle\sum_{i=1}^{n} a_{ji}x_i \leq b_j \quad & \text{for } j = 1, \ldots, m, \\ x_i \geq 0 \quad & \text{for } i = 1, \ldots, n, \end{aligned}$$

minimize $b_1y_1 + b_2y_2 + \ldots + b_my_m$

under the constraints

$$\begin{aligned} \textstyle\sum_{j=1}^{m} a_{ji}y_j \geq c_i \quad & \text{for } i = 1, \ldots, n, \\ y_j \geq 0 \quad & \text{for } j = 1, \ldots, m. \end{aligned}$$

maximize $\displaystyle\sum_{e \in E} x_e$

under the constraints

$$\sum_{e \in Inc(v)} x_e \leq 1 \quad \text{for every } v \in V,$$

$$x_e \geq 0.$$

MMP

minimize $\displaystyle\sum_{v \in V} y_v$

under the constraints

$$\begin{aligned} y_u + y_w \geq 1 \quad & \text{for every } \{u, w\} \in E \\ y_v \geq 0. & \end{aligned}$$

MIN-VCP

Primal-dual method

Primal-dual scheme

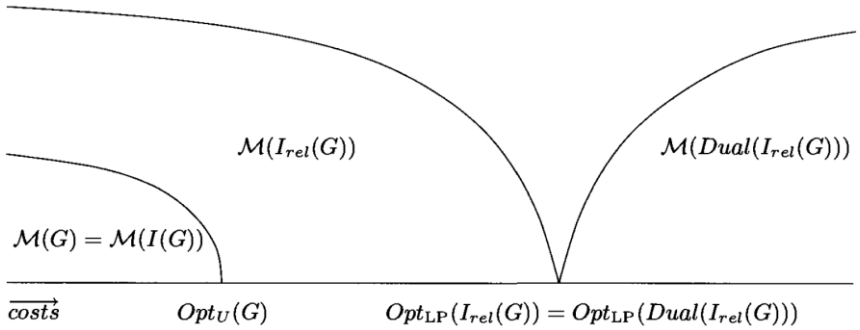
Input: An input instance G of an optimization problem U.

Step 1: Express G as an instance $I(G)$ of IP and relax $I(G)$ to an instance $I_{rel}(G)$ of LP.

Step 2: Construct $Dual(I_{rel}(G))$ as an instance of LP.

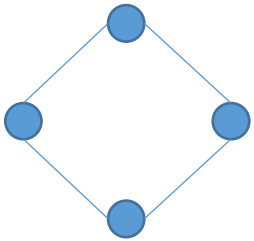
Step 3: Try to solve at once $Dual(I_{rel}(G))$ as an instance of LP and $I(G)$ as an instance of IP.

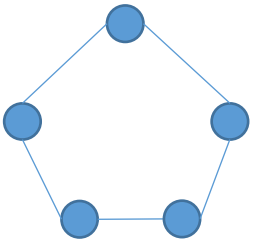
Output: A feasible solution α for $I(G)$ and $Opt_{\mathrm{LP}}(Dual(I_{rel}(G)))$.



广义的relaxation

- 将最长哈密尔顿圈问题 松弛为 最大权匹配问题
  - 如何将最大权匹配 改造为 最/较长哈密尔顿圈?
  - 改造后的cost如何?





OT

- 除IP外，广义的“松弛-修正”思想还可用于解决其它问题，例如TSP的松弛修正算法、最短超串问题的松弛修正算法等，请调研至少2种算法（其中至多1种来自上述例子），结合例子介绍算法的设计与分析，重点阐述其中的“松弛-修正”思想。