Introduction to
Optimization Method

Dis$\cdot$*count*

School of Management
University of Science and Technology of China

Sept 1, 2020

Summary

Dynamic Programming

0-1 Knapsack Problem

The most common problem being solved is the 0-1 knapsack problem, which restricts the number x_i of copies of each kind of item to zero or one. Given a set of n items numbered from 1 up to n, each with a weight w_i and a value v_i, along with a maximum weight capacity W,

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

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \le W \text{ and } x_i \in \{0, 1\}.$$

0-1 Knapsack Problem

Assume $w_1, w_2, \ldots, w_n, W$ are strictly positive integers. Define $m[i, w]$ to be the maximum value that can be attained with weight less than or equal to w using items up to i. We can define $m[i, w]$ recursively as follows:

m[0, w]=0

m[i, w]=m[i-1, w] if w$_i > w$

m[i, w]=$\max(m[i-1, w], m[i-1, w-w_i] + v_i)$ if $w_i \leqslant w$.

0-1 Knapsack Problem

Assume $w_1, w_2, \ldots, w_n, W$ are strictly positive integers. Define $m[i, w]$ to be the maximum value that can be attained with weight less than or equal to w using items up to i. We can define $m[i, w]$ recursively as follows:

m[0, w]=0

m[i, w]=m[i-1, w] if w$_i > w$

m[i, w]$=\max(m[i-1,\ w],\ m[i-1, w-w_i]+v_i)$ if $w_i \leqslant w$.

0-1 Knapsack Problem

The solution can then be found by calculating $m[n, W]$. To do this efficiently, we can use a table to store previous computations. This solution will therefore run in $O(nW)$ time and $O(nW)$ space.

Weight Limit (i):	0	1	2	3	4	5	6	7	8	9	10	11
$w_1 = 1$ $v_1 = 1$	0	1	1	1	1	1	1	1	1	1	1	1
$w_2 = 2$ $v_2 = 6$	0	1	6	7	7	7	7	7	7	7	7	7
$w_3 = 5$ $v_3 = 18$	0	1	6	7	7	18	19	24	25	25	25	25
$w_4 = 6$ $v_4 = 22$	0	1	6	7	7	18	22	24	28	29	29	40
$w_5 = 7$ $v_5 = 28$	0											

Optimal substructure

- Fibonacci sequence

$$fib(n) = fib(n-1) + fib(n-2)$$

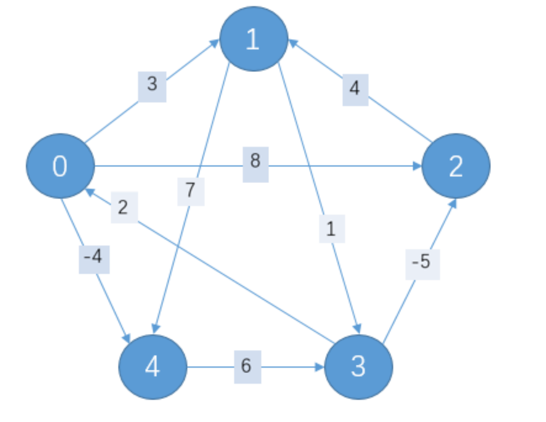
- Dijkstra's algorithm for the shortest path problem

$$d[y] = \min_x \{d[y], d[x] + w(x, y)\}$$

How to define the status and stage of problems is essential.

Shortest path problem

Given a directed graph (V, A) with source node s, target node t, and cost w_{ij} for each edge (i, j) in A, consider the program with variables x_{ij}.



Shortest path problem

- Integer programming formulation:

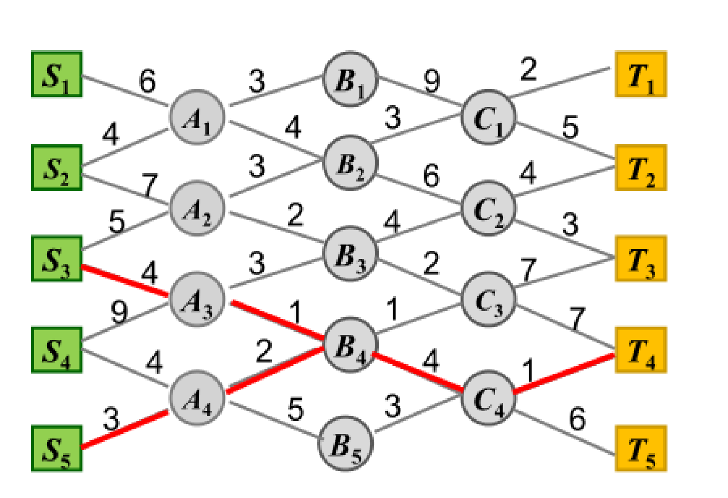
$$\min \sum_{(i,j)\in A} w_{ij} x_{ij}$$

$$\text{s.t.} \sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & \text{if } i = s; \\ -1, & \text{if } i = t; \\ 0, & \text{otherwise.} \end{cases}$$

$$x \in \{0, 1\} \text{ and for all } i.$$

Shortest path problem

- Find the shortest path from s to t.

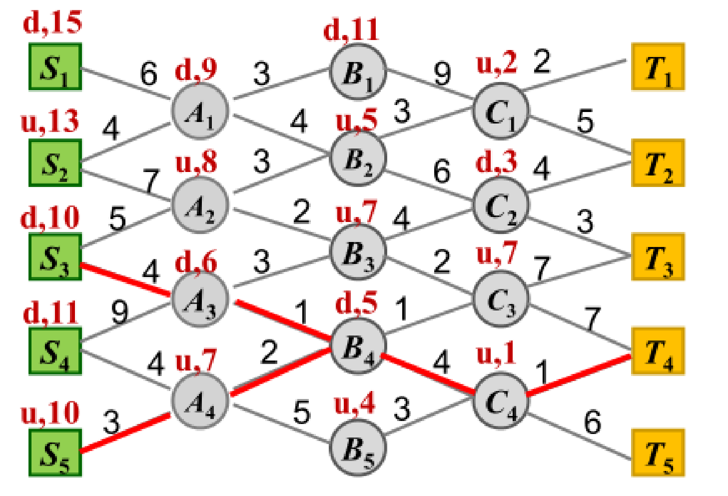


Shortest path problem

$$\begin{aligned}
&Stage1 \quad F(C_l) = \min_m\{C_lT_m\}\\
&Stage2 \quad F(B_k) = \min_l\{B_kC_l + F(C_l)\}\\
&Stage3 \quad F(A_j) = \min_k\{A_jB_k + F(B_k)\}\\
&Stage4 \quad F(S_i) = \min_j\{S_iA_j + F(A_j)\}
\end{aligned}$$

Shortest path problem

- Find the shortest path from s to t.



Integer & Linear Programming

Integer Programming

- Shortest path problem

$$\min \sum_{(i,j)\in A} w_{ij}x_{ij}$$

$$\text{s.t.} \sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & \text{if } i = s; \\ -1, & \text{if } i = t; \\ 0, & \text{otherwise.} \end{cases}$$

$$x \in \{0,1\} \text{ and for all } i.$$

- Maximum flow problem
- Assignment problem

Integer Programming

- Shortest path problem

$$\min \sum_{(i,j)\in A} w_{ij}x_{ij}$$

$$\text{s.t.} \sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & \text{if } i = s; \\ -1, & \text{if } i = t; \\ 0, & \text{otherwise.} \end{cases}$$

$$x \in \{0,1\} \text{ and for all } i.$$

- Maximum flow problem
- Assignment problem

Totally Unimodular Matrix

- Every entry in A is 0, $+1$, or -1;
- Every column of A contains at most two non-zero (i.e., $+1$ or -1) entries;
- If two non-zero entries in a column of A have the same sign, then the row of one is in B, and the other in C;
- If two non-zero entries in a column of A have opposite signs, then the rows of both are in B, or both in C.

TU Matrix

- Totally unimodular matrices are extremely important in polyhedral combinatorics and combinatorial optimization since they give a quick way to verify that a linear program is integral (has an integral optimum, when any optimum exists).
- Specifically, if A is TU and b is integral, then linear programs of forms like $\{\min cx \mid Ax \geq b, x \geq 0\}$ or $\{\max cx \mid Ax \leq b\}$ have integral optima, for any c. Hence if A is totally unimodular and b is integral, every extreme point of the feasible region (e.g. $\{x \mid Ax \geq b\}$) is integral and thus the feasible region is an integral polyhedron.

Another Perspective

Recall the simplex method for linear programming.

$$Bx = b$$
$$x^* = (B^{-1}b, 0)$$

How to obtain the inverse of B?
Cramer's rule:

$$B^{-1} = B^*/\text{det}(B)$$

Simplex Method

- Feasible region(Convex polytope)
- Basic feasible solution(Extreme point)
- Basic variables(Identity matrix)
- Entering variable selection
- Leaving variable selection
- Pivot operation
- Reduced costs

Another Perspective

The simplex method is an iteration process.

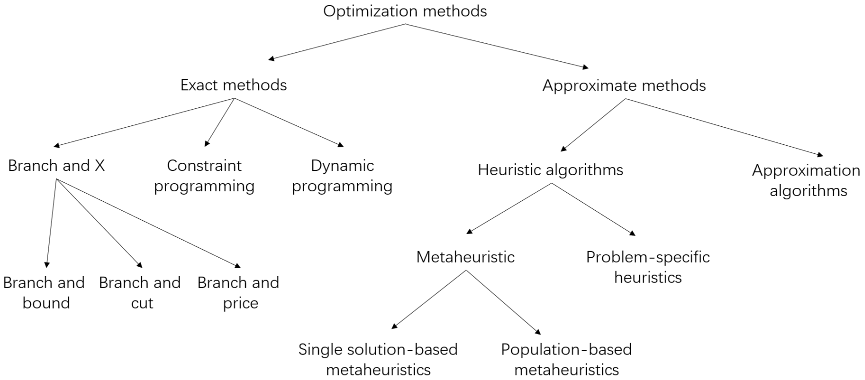
$$x' = x\theta\lambda$$

How to obtain the inverse of B?

Cramer's rule:

$$B^{-1} = B^*/\mathsf{det}(B)$$

General Optimization Methods



General Optimization Methods

Exact Methods

Exact Methods

- Branch and X
    1. Branch and bound
    2. Branch and cut
    3. Branch and price

Exact Methods

- Branch and X
  1. Branch and bound
  2. Branch and cut
  3. Branch and price

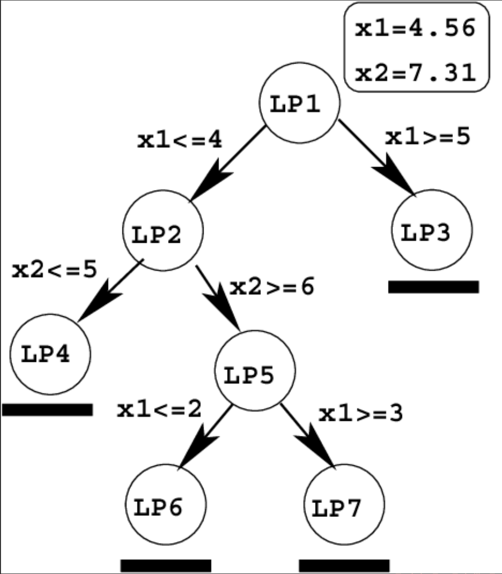
Exact Methods

- Branch and X
  1. Branch and bound
  2. Branch and cut
  3. Branch and price

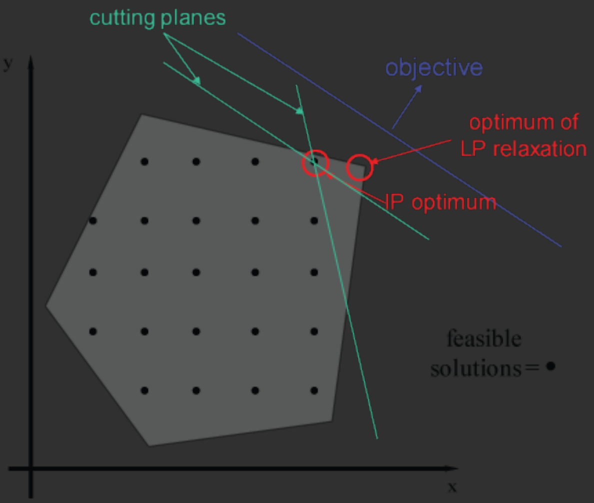
Exact Methods

- Branch and X
  1. Branch and bound
  2. Branch and cut
  3. Branch and price

Branch and bound



Cutting Plane Method

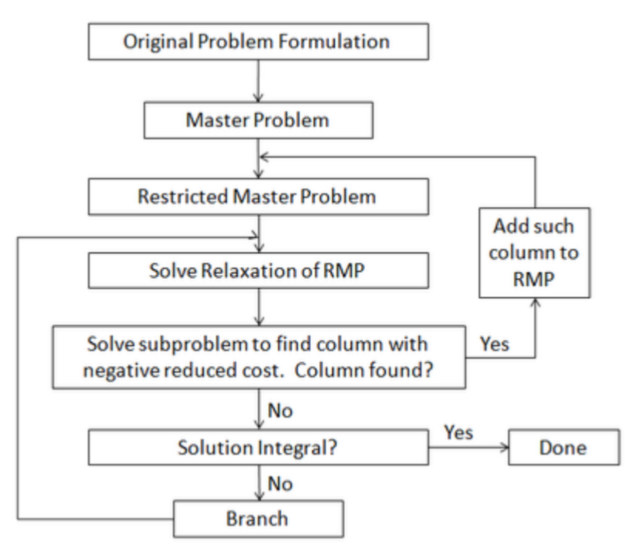


Cutting Planes

Gomory's Cut

- Using the simplex method: $x_i + \sum \bar{a}_{i,j} x_j = \bar{b}_i$ where x_i are the basic variables and the x_j's are the nonbasic variables.
- Rewrite this equation: integer parts(left) and the fractional parts(right): $x_i + \sum \lfloor \bar{a}_{i,j} \rfloor x_j - \lfloor \bar{b}_i \rfloor = \bar{b}_i - \lfloor \bar{b}_i \rfloor - \sum (\bar{a}_{i,j} - \lfloor \bar{a}_{i,j} \rfloor) x_j$.
- Right side is less than 1 and the left side is an integer, therefore the inequality: $\bar{b}_i - \lfloor \bar{b}_i \rfloor - \sum (\bar{a}_{i,j} - \lfloor \bar{a}_{i,j} \rfloor) x_j \leq 0$ must hold for any integer point in the feasible region.
- The inequality above is a cut with the desired properties. Introducing a new slack variable x_k for this inequality, a new constraint is added to the linear program, namely $x_k + \sum (\lfloor \bar{a}_{i,j} \rfloor - \bar{a}_{i,j}) x_j = \lfloor \bar{b}_i \rfloor - \bar{b}_i$, $x_k \geq 0$, x_k an integer.

Branch and price



Exact Methods

- Branch and X
- Dynamic programming
- Constraint programming
- Enumeration method

Exact Methods

- Branch and X
- Dynamic programming
- Constraint programming
- Enumeration method

Exact Methods

- Branch and X
- Dynamic programming
- Constraint programming
- Enumeration method

General Optimization Methods
Approximate Methods

Approximate Methods

- Heuristic algorithms
  1. Metaheuristic
     - Single solution-based metaheuristics
     - Population-based metaheuristics
  2. Problem-specific heuristics
- Approximate algorithms

Approximate Methods

- Heuristic algorithms
  1. Metaheuristic
     * Single solution-based metaheuristics
     * Population-based metaheuristics
  2. Problem-specific heuristics
- Approximate algorithms

Approximate Methods

- Heuristic algorithms
  1. Metaheuristic
     - Single solution-based metaheuristics
     - Population-based metaheuristics
  2. Problem-specific heuristics
- Approximate algorithms

Approximate Methods

- Heuristic algorithms
  1. Metaheuristic
     * Single solution-based metaheuristics
     * Population-based metaheuristics
  2. Problem-specific heuristics
- Approximate algorithms

Approximate Methods

- Heuristic algorithms
  1. Metaheuristic
     * Single solution-based metaheuristics
     * Population-based metaheuristics
  2. Problem-specific heuristics
- Approximate algorithms

Combination Optimization

Combination Optimization

what

- item1
- item2

what

- item3
- item4

Convex Optimization

Convex Optimization

Acknowledgments

The author is extremely thankful to Prof. Liu for the short, yet wonderful, conversations about this seminar.

References

de Figueiredo, D. G. *Análise de Fourier e Equações Diferenciais Parciais.* 5th ed. (IMPA, 2018).

Fleming, H. *George Green e Suas Funções.* http://www.hfleming.com/green.pdf.

Panofsky, W. K. H. & Phillips, M. *Classical Electricity and Magnetism.* 2nd ed. (Addison-Wesley Publishing Company, Inc., 1962).

Shankar, R. *Principles of Quantum Mechanics.* 2nd ed. (Springer, 1994).

The End