

Chapter 10: Iterative Improvement

Step by step and the thing is done.
(Charles Atlas)

| | |
|--|-----------|
| Introduction | 2 |
| Linear Programming | 3 |
| The Linear Programming Program | 3 |
| Terminology | 4 |
| Example of Feasible Region | 5 |
| Example Values of Objective Function | 6 |
| LP Application to Transportation Problem | 7 |
| Salsa Instance | 8 |
| LP Application to Maximum Flow | 9 |
| Example Maximum Flow Instance | 10 |
| Maximum Bipartite Matching | 11 |
| Maximum Matching and Bipartite Graphs | 11 |
| Augmenting Paths | 12 |
| Example of Augmenting Paths | 13 |
| Example of Augmenting Paths, Continued | 14 |
| Stable Marriage Problem | 15 |
| Stable Marriage Problem | 15 |
| Stable Marriage Algorithm | 16 |
| Example of Stable Marriage Algorithm | 17 |
| Example, Continued | 18 |
| Comments on Stable Marriage Algorithm | 19 |

Introduction

Iterative improvement solves problems where:

- The problem is an optimization problem, to find the solution that minimizes or maximizes some value (cost/profit).
- An initial solution can be easily found.
- It can be improved by a sequence of small changes.
- It is returned when no more improvements can be made.

A potential pitfall is a local extremum. No small change makes an improvement, but some sequence of changes can.

Linear Programming

The Linear Programming Program

Linear programming optimizes a linear function subject to linear constraints.

$$\begin{aligned}
 &\text{maximize (or minimize) } c_1x_1 + \dots + c_nx_n \\
 &\text{subject to } a_{i1}x_1 + \dots + a_{in}x_n \leq \text{ (or } \geq \text{ or } =) b_i \\
 &\qquad\qquad\qquad \text{for } i = 1, \dots, m \\
 &\qquad\qquad\qquad \text{and } x_1 \geq 0, \dots, x_n \geq 0
 \end{aligned}$$

For example:

$$\begin{aligned}
 &\text{maximize } 3x + 5y \\
 &\text{subject to } x + y \leq 4 \\
 &\qquad\qquad\qquad x + 3y \leq 6 \\
 &\qquad\qquad\qquad x \geq 0, y \geq 0
 \end{aligned}$$

Terminology

- A *feasible solution* is any (x_1, \dots, x_n) point that satisfies the constraints.
- The *feasible region* is the set of all feasible points.
- The task is to find an *optimal solution*, the feasible solution with the largest value of the *objective function*.
- The *simplex method* starts with an *extreme point* and changes from extreme point to extreme point.
- The *simplex method* is guaranteed to find the optimal solution.

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 4

Example of Feasible Region

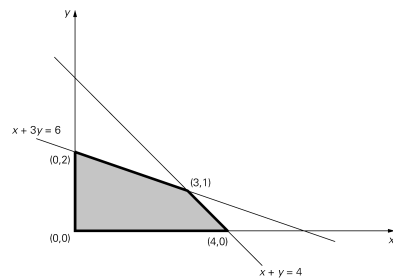


FIGURE 10.1 Feasible region of problem (10.2)

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 5

Example Values of Objective Function

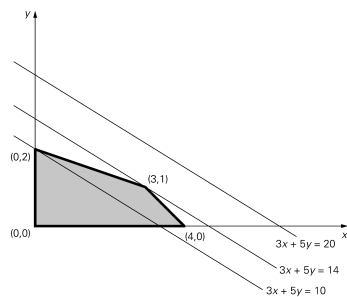


FIGURE 10.2 Solving a two-dimensional linear programming problem geometrically

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 6

LP Application to Transportation Problem

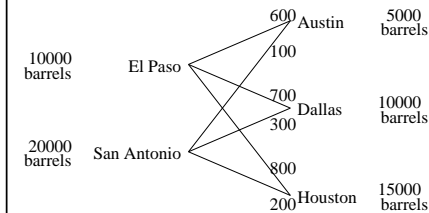
- Suppose that a product needs to be shipped from locations s_1, s_2, \dots to locations t_1, t_2, \dots .
- Each “source” i has a_i units.
- Each “sink” j requires b_j units.
- For each source i and sink j , there is a shipping cost of c_{ij} per unit.
- We need to determine the number of units x_{ij} shipped from each source i to each sink j .
- We want to minimize the shipping cost.
- This can be formulated as a linear programming problem.

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 7

Salsa Instance

Suppose we need to ship salsa from El Paso and San Antonio to Austin, Dallas, and Houston.



CS 3343 Analysis of Algorithms

Chapter 10: Slide – 8

LP Application to Maximum Flow

- Suppose we want to maximize the flow of material from a source v_1 to a sink v_n .
- If (v_i, v_j) is an edge, then c_{ij} is its capacity.
- The flow from v_i to v_j is $0 \leq x_{ij} \leq c_{ij}$
- Flow in an intermediate vertex equals flow out.

$$\sum_{i=1}^n x_{ij} = \sum_{k=1}^n x_{jk} \text{ for } j = 2, \dots, n-1$$

- Flow from the source equals flow to the sink.

$$\sum_{j=1}^n x_{1j} = \sum_{j=1}^n x_{jn}$$

- This can be formulated as a LP problem.

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 9

Example Maximum Flow Instance

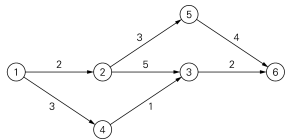


FIGURE 10.4 Example of a network graph. The vertex numbers are vertex “names;” the edge numbers are edge capacities.

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 10

Maximum Bipartite Matching

11

Maximum Matching and Bipartite Graphs

- In an undirected graph, a *matching* is a subset of edges such that no two edges share a vertex.
- A *maximum matching* is a matching with the largest number of edges.
- In a *bipartite graph*, the vertices can be partitioned into V and U such that every edge is between V and U .

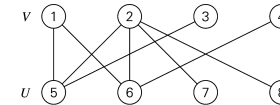


FIGURE 10.8 Example of a bipartite graph

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 11

Augmenting Paths

- Let M be a matching.
- Construct a directed graph where edges in M go from U to V and all other edges go from V to U .
- If there is a larger matching, then there is a path from an unmatched vertex in V to an unmatched vertex in U .
- This path is called an *augmenting path*.
- The new matching subtracts the path edges in M and adds the path edges not in M .
- We can use depth-first search to find augmenting paths. This results in a $O(V^2 + VE)$ algorithm.

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 12

Example of Augmenting Paths

(a)
Augmenting path: 1, 6

(b)
Augmenting path: 2, 6, 1, 7

CS 3343 Analysis of Algorithms Chapter 10: Slide – 13

Example of Augmenting Paths, Continued

(c)
Augmenting path: 3, 8, 4, 9, 5, 10

(d)
Maximum matching

FIGURE 10.9 Augmenting paths and matching augmentations

CS 3343 Analysis of Algorithms Chapter 10: Slide – 14

Stable Marriage Problem

Stable Marriage Problem

- Y is a set of n men. X is a set of n women.
- Each man (woman) has a preference list ordering the women (men) with no ties.
- A matching is *unstable* if there are matches (m_1, w_1) and (m_2, w_2) such that m_1 prefers w_2 over w_1 , and w_2 prefers m_1 over m_2 .
- The stable marriage problem is to find a matching that is *stable*.

| | men's preferences | | | women's preferences | | | ranking matrix | | |
|------|-------------------|-----|-----|---------------------|-----|-----|----------------|-----|-----|
| | 1st | 2nd | 3rd | 1st | 2nd | 3rd | Ann | Lea | Sue |
| Bob: | Lea | Ann | Sue | Ann | Jim | Tom | Bob | 2 | 3 |
| Jim: | Lea | Sue | Ann | Lea | Tom | Bob | Jim | 3 | 1 |
| Tom: | Sue | Lea | Ann | Sue | Jim | Tom | Bob | 3 | 2 |

CS 3343 Analysis of Algorithms Chapter 10: Slide – 15

Stable Marriage Algorithm

```

algorithm StableMarriage( $Y, X, P$ )
  // Returns a stable matching
  // Input:  $n$  men  $Y$ ,  $n$  women  $X$ , preferences  $P$ 
  // Output: Stable matching  $M$ 
   $M \leftarrow \emptyset$ 
  while there are free men
     $m \leftarrow$  a free man
     $w \leftarrow$  next woman on  $m$ 's preference list
    if  $w$  is free then add  $(m, w)$  to  $M$ 
    else if  $w$  prefers  $m$  to current match  $m'$  then
      remove  $(m', w)$  from  $M$ 
      add  $(m, w)$  to  $M$ 
  return  $M$ 
  
```

CS 3343 Analysis of Algorithms Chapter 10: Slide – 16

Example of Stable Marriage Algorithm

| | Ann | Lea | Sue | |
|---------------|-----|------|-------------|-------------|
| Free men: | Bob | 2, 3 | <u>1, 2</u> | 3, 3 |
| | Jim | 3, 1 | 1, 3 | 2, 1 |
| Bob, Jim, Tom | Tom | 3, 2 | 2, 1 | 1, 2 |
| | Ann | Lea | Sue | |
| Free men: | Bob | 2, 3 | <u>1, 2</u> | 3, 3 |
| | Jim | 3, 1 | 1, 3 | 2, 1 |
| Jim, Tom | Tom | 3, 2 | 2, 1 | 1, 2 |
| | Ann | Lea | Sue | |
| Free men: | Bob | 2, 3 | <u>1, 2</u> | 3, 3 |
| | Jim | 3, 1 | 1, 3 | <u>2, 1</u> |
| Jim, Tom | Tom | 3, 2 | 2, 1 | 1, 2 |

Bob proposed to Lea
Lea accepted

Jim proposed to Lea
Lea rejected

Jim proposed to Sue
Sue accepted

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 17

Example, Continued

| | Ann | Lea | Sue | |
|-----------|-----|-------------|-------------|-------------|
| Free men: | Bob | 2, 3 | <u>1, 2</u> | 3, 3 |
| | Jim | 3, 1 | 1, 3 | <u>2, 1</u> |
| Tom | Tom | 3, 2 | 2, 1 | <u>1, 2</u> |
| | Ann | Lea | Sue | |
| Free men: | Bob | 2, 3 | 1, 2 | 3, 3 |
| | Jim | 3, 1 | 1, 3 | <u>2, 1</u> |
| Tom | Tom | 3, 2 | <u>2, 1</u> | 1, 2 |
| | Ann | Lea | Sue | |
| Free men: | Bob | <u>2, 3</u> | 1, 2 | 3, 3 |
| | Jim | 3, 1 | 1, 3 | <u>2, 1</u> |
| Bob | Tom | 3, 2 | <u>2, 1</u> | 1, 2 |

Tom proposed to Sue
Sue rejected

Tom proposed to Lea
Lea replaced Bob with Tom

Bob proposed to Ann
Ann accepted

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 18

Comments on Stable Marriage Algorithm

- Unstable match (m_1, w_1) and (m_2, w_2) cannot happen.
- If m_1 and w_2 preferred each other, then w_2 would be asked before w_1 and w_2 would have dumped m_2 .
- Each preference of each man is considered once, so this algorithm can be $O(n^2)$ with efficient data structures.
- The algorithm where the men's preferences are considered in order is *man-optional*.

CS 3343 Analysis of Algorithms

Chapter 10: Slide – 19