

# Integer / linear programming

# “Programming”

- A class of mathematical techniques (e.g. dynamic programming) to solve the general optimization problems as

$$\text{Max } f(x), \quad x \in S \subseteq R^n$$

$R^n$ : the set of  $n$ -dimensional vectors of real numbers

$f(x)$ : *objective function*, a real-valued function defined on  $S$   
 $S$  the *constraint set*.

- By choosing  $f$  and  $S$  appropriately, we can model a wide variety of real-life problems in this way.
- Hard problems:  $|S| \sim O(k^n)$ ,  $k > 1$

# Feasibility and optimality

- Any  $x \in S$  is called a *feasible solution*
- If there is an  $x_0 \in S$  such that  $f(x) \geq f(x_0)$  for all  $x \in S$  then  $x_0$  is called an *optimal solution*
- The aim is to find an optimal solution for a given  $f$  and  $S$
- $S$  can be defined by *constraints*

# Example

$$\text{MAX: } 350X_1 + 300X_2 \quad \rightarrow f$$

$$\text{S.T.: } X_1 + X_2 \leq 200$$

$$9X_1 + 6X_2 \leq 1566 \quad \rightarrow \text{constraints}$$

$$12X_1 + 16X_2 \leq 2880$$

$$X_1, X_2 \geq 0$$

$X_1, X_2$  must be integers  $\rightarrow$  integer programming

# Integer (linear) programming

- $f$  &  $S$  are restricted by linear form (functions)
  - Linear programming
- $S$  is restricted to have only integer values
  - Integer programming (IP), often referred to as integer linear programming (ILP)
- mixed integer programming problem: some elements of  $S$  are restricted to integers
- ILP is often harder than the corresponding LP problem

# Linear programming

- $f(x) = c^T x$ ,  $S = \{ x \mid Ax = b, x \geq 0 \}$ 
  - $c$  is an  $n \times 1$  vector,  $A$  is an  $m \times n$  matrix and  $b$  is an  $m \times 1$  vector
- For general  $x$ , these problems can be solved exactly (e.g. simplex technique).  
For integer  $x$ , the problem is *NP*-complete.

# Inequality

- Inequality constraints can easily be introduced by adding an extra variable
  - $\max 2x_1 + 3x_2$  subject to  $x_1 + x_2 \leq 10$  is equivalent to  $\max 2x_1 + 3x_2$  subject to  $x_1 + x_2 + x_3 = 10$
  - For “ $\geq$ ”, we would insert  $(-x_3)$  into the constraint
  - The extra variable is called a slack variable – it does not appear in the objective function. Because this is so straight-forward, many ILP solving programs allow you to express constraints with inequality directly.

# Example: capital budgeting

- A firm has  $n$  projects that it would like to undertake, but due to budget limitations, not all can be selected. In particular, project  $j$  has a value of  $c_j$ , and requires an investment of  $a_{ij}$  in the time period  $i$ ,  $i = 1, \dots, m$ . The capital available in time period  $i$  is  $b_i$ .
- Objective: maximize the total value, subject to given budget constraints

# Example: capital budgeting

A set of variables  $x_j$ , which we interpret as:

- $x_j = 1$ , project  $j$  is selected
- $x_j = 0$ , project  $j$  is not selected

Then the objective function can be formulated as

$$\sum_{j=1}^n c_j x_j$$

The constraints are

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, \dots, m; \quad x_j \leq 1, \quad j = 1, \dots, n$$

# Linear programming problems

$$\begin{array}{ll} \text{maximize} & z = -4x_1 + x_2 - x_3 + x_4 \\ \text{subject to} & -7x_1 + 5x_2 + x_3 + x_4 = 8 \\ & -2x_1 + 4x_2 + 2x_3 - x_4 = 10 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

We will describe LPs that start with the following (canonical) form:

- equality constraints
- nonnegative (right hand side, RHS) variables

# Fundamental theorem

Extreme point (or Simplex filter) theorem:

If the maximum or minimum value of a linear function defined over a polygonal convex region exists, then it is to be found at the boundary of the region

Boundary points: basic feasible solutions

# What does the extreme point theorem imply?

- A finite number of extreme points (bfs) implies a finite number of solutions!
- Hence, search is reduced to a finite set of points
- However, a finite set can still be too large for practical purposes
- **Simplex method provides an** efficient systematic search guaranteed to converge in a finite number of steps.

# Basic feasible solutions

- Each corner point solution of the polyhedron is a basic feasible solution.
- The simplex method is a systematic way of moving from one basic feasible solution to another, always improving the solution, until the optimum solution is obtained.

# Basic Feasible Solutions

$$\begin{array}{ll} \text{maximize} & z = -4x_1 + x_2 - x_3 + x_4 \\ \text{subject to} & -7x_1 + 5x_2 + x_3 + x_4 = 8 \\ & -2x_1 + 4x_2 + 2x_3 - x_4 = 10 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

Suppose there are  $m$  constraints,  $n$  variables

A **basic solution** is found by setting  $n-m$  variables to 0 and solving the remaining system with  $m$  variables and  $m$  constraints.

- The  $n - m$  variables are called **non-basic variables**
- The  $m$  variables are called **basic variables**

# Basic feasible solutions

maximize

$$z = -4x_1 + x_2 - x_3 + x_4$$

subject to

$$-7x_1 + 5x_2 + x_3 + x_4 = 8$$

$$-2x_1 + 4x_2 + 2x_3 - x_4 = 10$$

$$x_1, x_2, x_3, x_4 \geq 0$$

<b>-z</b>	<b>x<sub>1</sub></b>	<b>x<sub>2</sub></b>	<b>x<sub>3</sub></b>	<b>x<sub>4</sub></b>		
<b>1</b>	<b>-4</b>	<b>1</b>	<b>-1</b>	<b>1</b>	<b>=</b>	<b>0</b>
<b>0</b>	<b>-7</b>	<b>5</b>	<b>1</b>	<b>1</b>	<b>=</b>	<b>8</b>
<b>0</b>	<b>-2</b>	<b>4</b>	<b>2</b>	<b>-1</b>	<b>=</b>	<b>10</b>

# Basic feasible solutions

<b>-z</b>	<b>x<sub>1</sub></b>	<b>x<sub>2</sub></b>	<b>x<sub>3</sub></b>	<b>x<sub>4</sub></b>		
<b>1</b>	<b>-11</b>	<b>6</b>	<b>0</b>	<b>2</b>	<b>=</b>	<b>8</b>
<b>0</b>	<b>-7</b>	<b>5</b>	<b>1</b>	<b>1</b>	<b>=</b>	<b>8</b>
<b>0</b>	<b>12</b>	<b>6</b>	<b>0</b>	<b>-3</b>	<b>=</b>	<b>-6</b>

**Example:** Suppose we want the solution with basic variables  $x_3$  and  $x_4$ , and thus  $x_1$  and  $x_2$  are non-basic.

**We then perform pivot operations.**

# Basic Feasible Solutions

<b>-z</b>	<b>x<sub>1</sub></b>	<b>x<sub>2</sub></b>	<b>x<sub>3</sub></b>	<b>x<sub>4</sub></b>	
<b>1</b>	<b>-3</b>	<b>2</b>	<b>0</b>	<b>0</b>	<b>= 4</b>
<b>0</b>	<b>-3</b>	<b>3</b>	<b>1</b>	<b>0</b>	<b>= 6</b>
<b>0</b>	<b>-4</b>	<b>2</b>	<b>0</b>	<b>1</b>	<b>= 2</b>

**Next pivot on the -3.**

# Basic Feasible Solutions

**Canonical form:** basic variables have a single one in the column.

	reduced costs	→	-z	x <sub>1</sub>	x <sub>2</sub>	x <sub>3</sub>	x <sub>4</sub>	=	4
			1	-3	2	0	0		6
			0	-3	3	1	0		2
			0	-4	2	0	1		2

The basic solution is found by setting non-basic variables to 0. We get  $x_1=0$ ,  $x_2=0$ ,  $x_3=6$ ,  $x_4=2$ .

This solution also satisfies  $x \geq 0$ . It is called a **basic feasible solution**.

# Simplex algorithm

$z = -3x_1 + 2x_2 - 4$

-z	$x_1$	$x_2$	$x_3$	$x_4$	
1	-3	2	0	0	= 4
0	-3	3	1	0	= 6
0	-4	2	0	1	= 2

The entering variable for a max problem is a variable with positive reduced cost.

The pivot element is chosen uniquely in the column of the entering variable so that the next basis is feasible.

# Simplex algorithm

The pivot element is chosen to leave the basis according to pivot rules (e.g. a min ratio rule).

-z	$x_1$	$x_2$	$x_3$	$x_4$	
1	1	0	0	-1	= -2
0	3	0	1	-1.5	= 3
0	-2	1	0	.5	= 1

A pivot is carried out, leading to the next bfs.

Variable  $x_4$  has left the basis.

The new basis consists of  $x_2$  and  $x_3$ .

# Simplex algorithm

-z	$x_1$	$x_2$	$x_3$	$x_4$	
1	0	0	-1/3	-1/2	= -3
0	1	0	1/3	-1/2	= 1
0	0	1	2/3	-1/2	= 3

**Optimality conditions for a maximization problem:  
all reduced costs are non-positive.**

# Simplex algorithm

Pivots are carried out until the bfs is optimal.

-z	$x_1$	$x_2$	$x_3$	$x_4$		
1	0	0	-1/3	-1/2	=	-3
0	1	0	1/3	-1/2	=	1
0	0	1	2/3	-1/2	=	3

$$z = -x_3/3 - x_4/2 + 3$$

This new bfs is optimal. Increasing  $x_3$  or  $x_4$  makes the solution worse.

# Duality in linear programming

- Every primal problem there exists a corresponding dual problem
  - Primal Maximization  $\rightarrow$  Dual Minimization
  - Primal Minimization  $\rightarrow$  Dual Maximization

# Duality in linear programming

- Primal has  $n$  choice variable and  $m$  constraints and dual has  $m$  variables and  $n$  constraints
- Right hand side elements ( $b_i$ ) in the primal correspond to coefficient of the objective function of the dual
- The  $a_{ij}$  constraint coefficients become  $a_{ji}$  in dual

# Primal and dual in matrix form

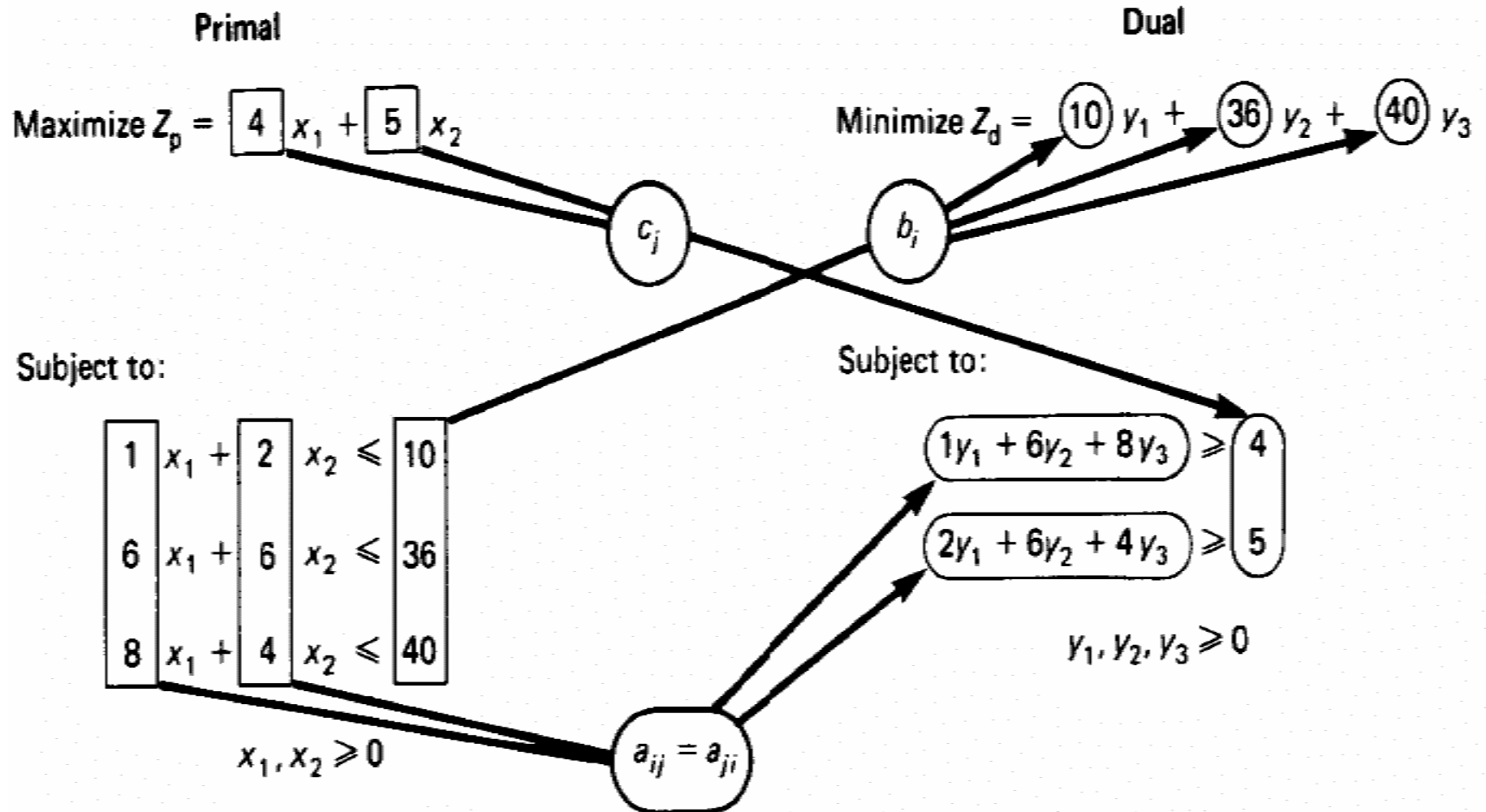
- Primal

$$\begin{aligned} \max Z &= c'x \\ \text{s. t. } Ax &\leq r \\ x &\geq 0 \end{aligned}$$

- Dual

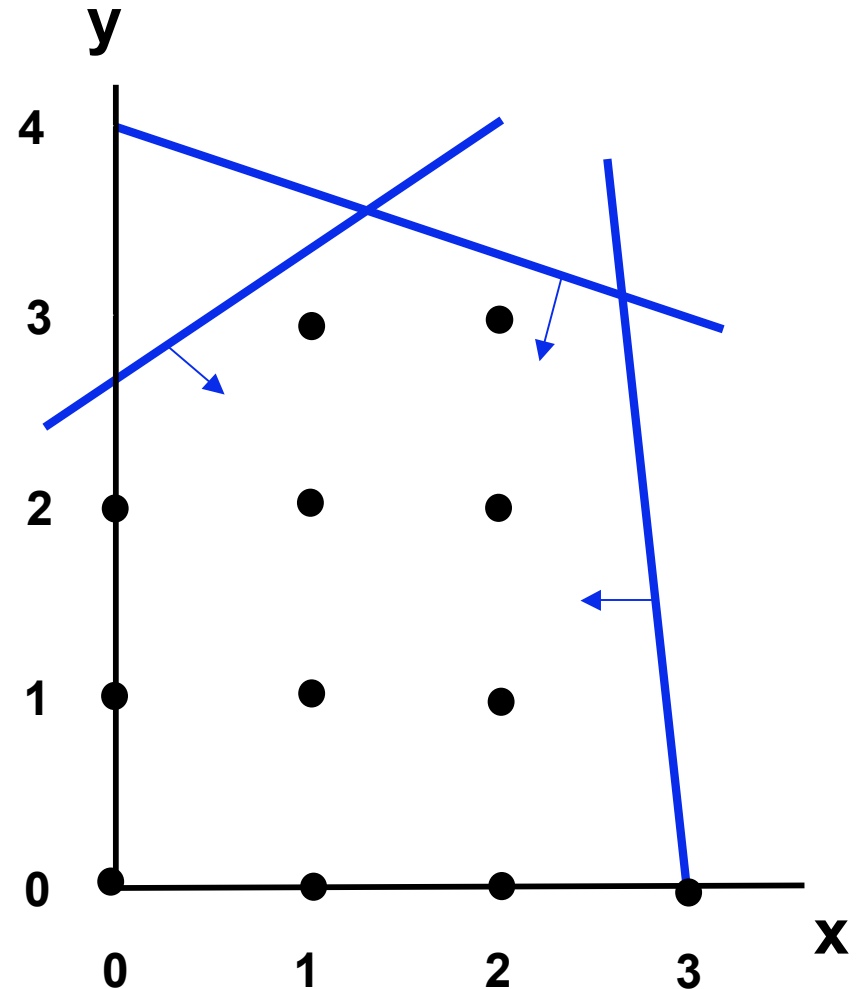
$$\begin{aligned} \max Z^* &= r'y \\ \text{s. t. } A'y &\geq c \\ y &\geq 0 \end{aligned}$$

# Primal and dual relationship



# LP $\rightarrow$ Integer Programming

- Feasible region is a set of discrete points.
- Can't be assured a corner point solution.
- IP is NP-hard problem (By reduction from Satisfiability)
- Solving it as an LP provides a relaxation and a bound on the solution.



# Duality theory

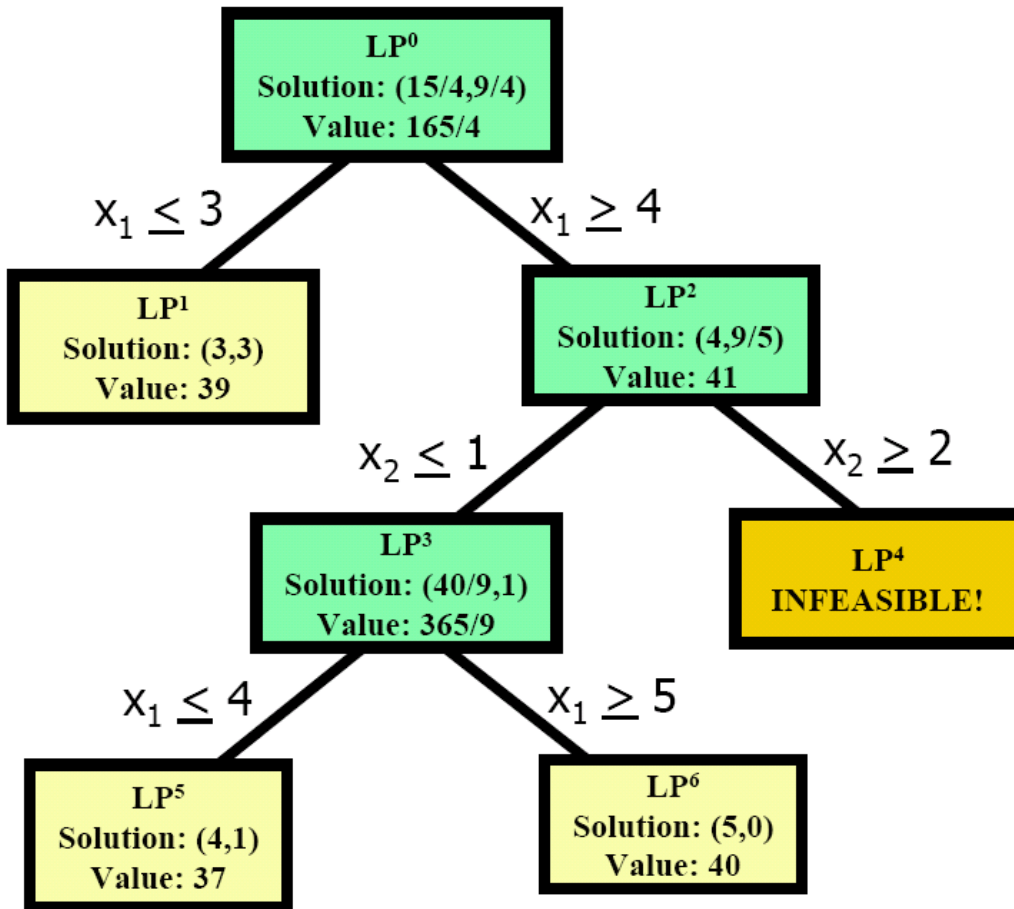
- **(Weak duality theorem)** the objective function value of the dual (*min*) at any feasible solution  $\geq$  the objective function value of the primal (*max*) at any feasible solution.
- **(Strong duality theorem)** if the primal has an optimal solution,  $x^*$ , then the dual also has an optimal solution,  $y^*$ , such that  $c'x^* = r'y^*$ .

# Solutions to IP

- Package
  - GNU Linear Programming Kit (GLPK)  
<http://www.gnu.org/software/glpk/> → free
  - Cplex: mathematical optimizer  
<http://www.ilog.com/products/cplex/> → commercial
- Enumeration
  - Small size of problems
- Branch and bound (tree-like search)
- Specially designed algorithms
  - LP relaxation

# IP Solver = LP Solver + B&B

- Solve the LP relaxation of the IP
- If the LP solution has a non-integer variable, branch on that variable and solve the 2 resultant LPs
- Traverse the tree recursively until:
  - All terminals exposed (unsolvable LPs are terminal); or
  - A sub-LP solution is worse than an existing integer solution



**IP:**

Maximize  $8x_1 + 5x_2$

S.t.  $x_1 + x_2 \leq 6$

$9x_1 + 5x_2 \leq 45$

$x_1, x_2 \in \mathbb{Z}^+$



**LP Relaxation:**

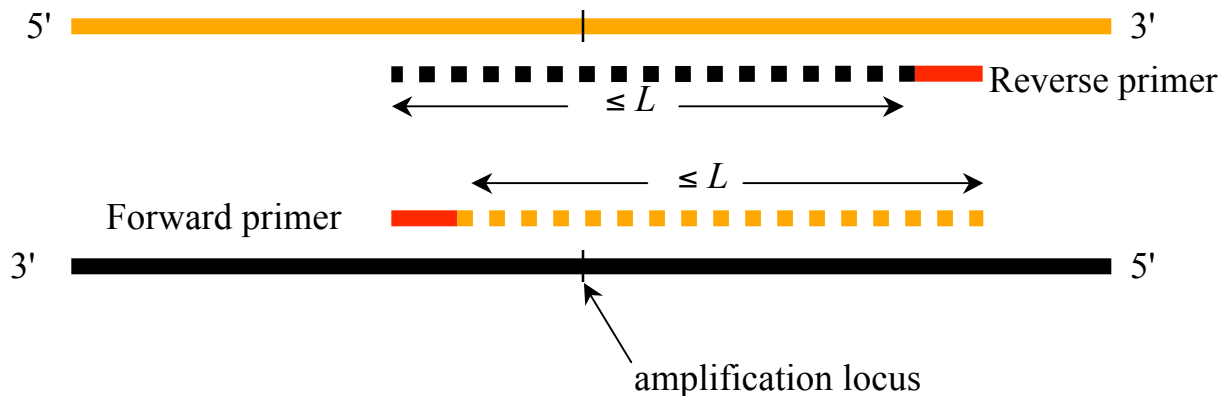
Maximize  $8x_1 + 5x_2$

S.t.  $x_1 + x_2 \leq 6$

$9x_1 + 5x_2 \leq 45$

$x_1, x_2 \geq 0$

# Primer pair selection problem



- **Given:**

- Genomic sequence around amplification locus
- Primer length  $k$
- Amplification upper-bound  $L$

- **Find:** Forward and reverse primers of length  $k$  that hybridize within a distance of  $L$  of each other and optimize amplification efficiency (**melting temperatures, secondary structure, cross hybridization, etc.**)

# Primer set selection: multiplex experiment

- Spotted microarray synthesis [Fernandes and Skiena'02]
  - Need unique pair for each amplification product, but primers can be re-used to minimize cost
  - Potential to reduce #primers from  $O(n)$  to  $O(n^{1/2})$  for  $n$  products

# Primer set selection: application

- SNP Genotyping
  - Thousands of SNPs that must be genotyped using hybridization based methods (e.g., SBE)
  - Selective PCR amplification needed to improve accuracy of detection steps (whole-genome amplification not appropriate)
  - No need for unique amplification!
  - Primer minimization is critical
    - Fewer primers to buy
    - Fewer multiplex PCR reactions

# Primer set selection problem

- **Given:**

- Genomic sequences around each amplification locus
- Primer length  $k$
- Amplification upperbound  $L$

- **Find:**

- Minimum size set of primers  $S$  of length  $k$  such that, for each amplification locus, there are two primers in  $S$  hybridizing to the forward and reverse sequences within a distance of  $L$  of each other
- Uniqueness constraint:  $S$  should contain a unique pair of primers amplifying each locus

# Selection with uniqueness Constraints

- Can be modeled as minimum multicolored subgraph problem:
  - Vertices of the graph correspond to candidate primers
  - add edge colored by color  $i$  between two primers if they amplify  $i$ -th SNP and do not amplify any other SNP
  - Goal is to find minimum size set of vertices inducing edges of all colors
- NP-hard problem
- Trivial approximation algorithm: select 2 primers for each SNP
  - $O(n^{1/2})$  approximation since at least  $n^{1/2}$  primers required by every solution

# Integer program formulation

- Variable  $x_u$  for every vertex (candidate primer)  $u$ 
  - $x_u$  set to 1 if  $u$  is selected, and to 0 otherwise
- Variable  $y_e$  for every edge  $e$ 
  - $y_e$  set to 1 if corresponding primer pair selected to amplify one of the SNPs
- Objective: minimize sum of  $x_u$
- Constraints:
  - for each  $i$ , sum of  $\{y_e : e \text{ amplifying SNP } i\} \geq 1$
  - $2y_e \leq x_u + x_v$  for every  $e$  incident to  $u$  &  $v$

# Protein structure prediction

- Protein functions determined by 3D structures
- About 30,000 protein structures in PDB (Protein Data Bank)
- Experimental determination of protein structures time-consuming and expensive
- Many protein sequences available

# Threading – Energy Function

MTYKLILNGKTKGETTTEAVDAATAEKVFQYANDNGVDGEWTYTE

how preferable to put  
two particular residues  
nearby:  $E_p$

alignment gap  
penalty:  $E_g$

compatibility with local  
secondary structure  
prediction:  $E_{ss}$

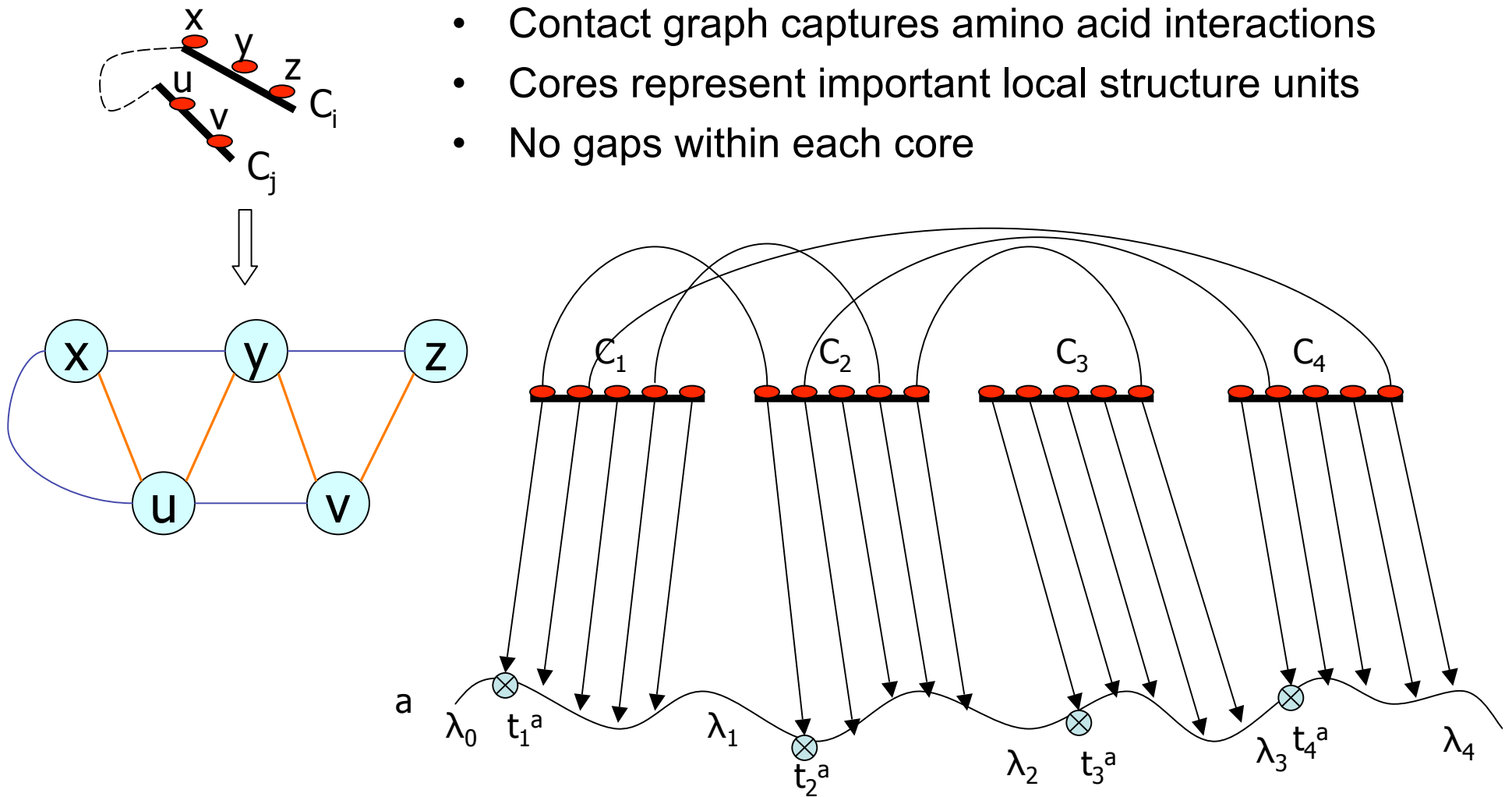


how well a residue fits  
a structural  
environment:  $E_s$

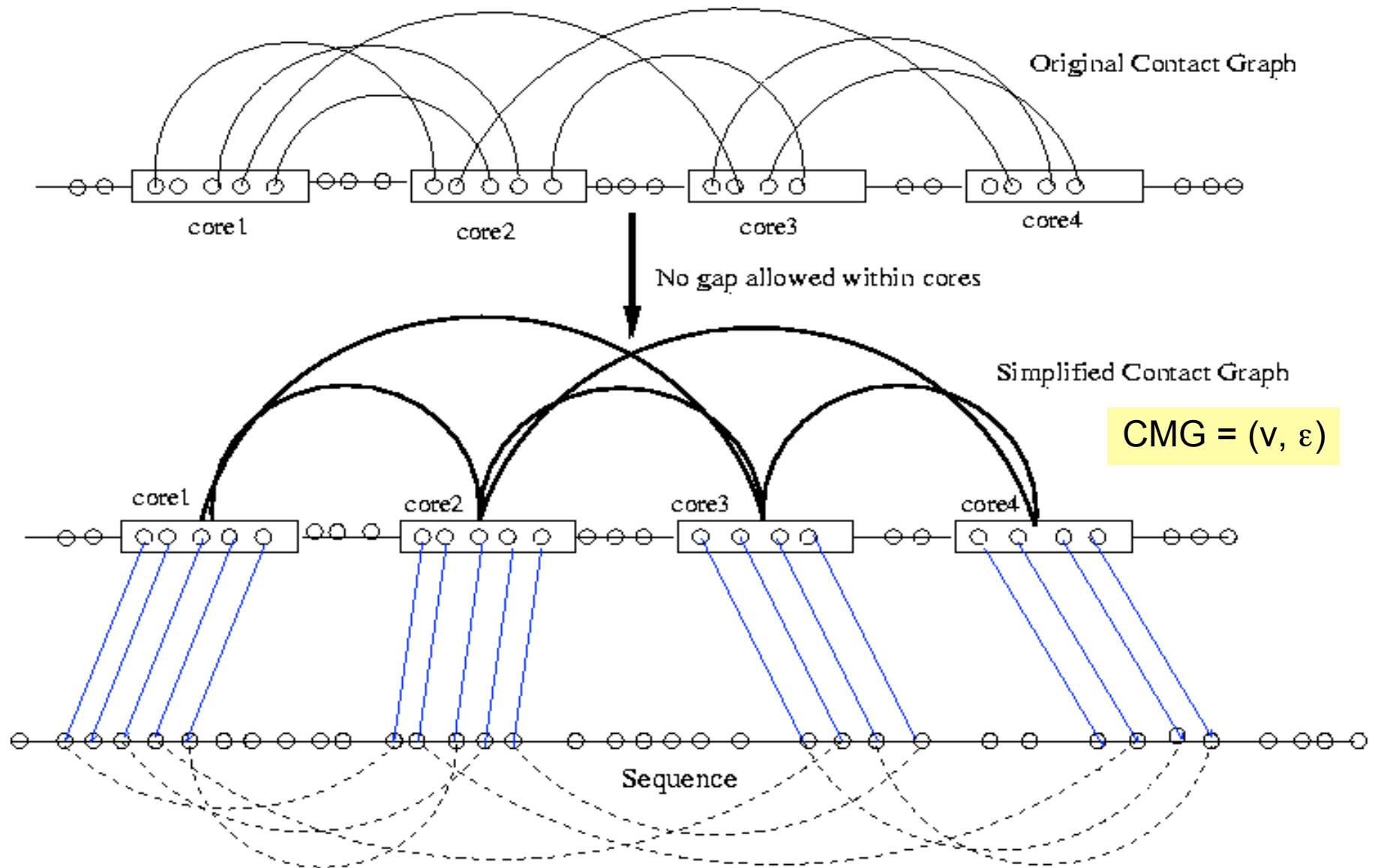
how often a residue  
mutates to the  
template residue:  $E_m$

**total energy:**  $w_m E_m + w_s E_s + w_p E_p + w_g E_g + w_{ss} E_{ss}$

# Threading – Formulation



# Threading – Formulation





# Threading with integer programming

*Minimize*

$$E = W_m E_m + W_s E_s + W_p E_p + W_g E_g + W_{ss} E_{ss}$$

*s.t.*

$$x_{(i,l)} + x_{(i+1,k)} \leq 1, k \notin R[i, i+1, l] \quad \leftarrow \text{Cores are aligned in order}$$

$$y_{(i,l)(j,k)} \leq x_{i,l}, k \in R[i, j, l]$$

$$y_{(i,l)(j,k)} \leq x_{j,k}, l \in R[j, i, k]$$

$$y_{(i,l)(j,k)} \geq x_{i,l} + x_{j,k} - 1$$

Each y variable is 1 if and only if its two x variables are 1 –

x and y represent exactly the same threading

$$\sum_{l \in D[i]} x_{i,l} = 1 \quad \leftarrow \text{Each core has only one alignment position}$$

$$x_{i,l}, y_{(i,l)(j,k)} \in \{0,1\}$$

# Energy function is linear

$$E_m = \sum_{i=1}^M \sum_{l \in D[i]} \left[ x_{i,l} \times \sum_{r=0}^{\text{len}_i-1} \text{Mutation}(\text{head}_i + r, l + r) \right],$$

- Sequence substitution score

$$E_s = \sum_{i=1}^M \sum_{l \in D[i]} \left[ x_{i,l} \times \sum_{r=0}^{\text{len}_i-1} \text{Fitness}(\text{head}_i + r, j + r) \right],$$

- Fitness of aa in each position (example, hydrophobicity)

$$E_{ss} = \sum_{i=1}^M \sum_{l \in D[i]} \left[ x_{i,l} \times \sum_{r=0}^{\text{len}_i-1} \text{SS}(\text{head}_i + r, j + r) \right],$$

- Agreement with secondary structure prediction

$$E_p = \sum_{1 \leq i \leq j \leq M, (c_i, c_j) \in \varepsilon(\text{CMG})} \sum_{l \in D[i]} \sum_{k \in R[i, j, l]} y_{(i,l), (j,k)} P(i, j, l, k),$$

- Pairwise interaction between two cores

$$P(i, j, l, k) = \sum_{u=0}^{\text{len}_i-1} \sum_{v=0}^{\text{len}_j-1} \times \delta(\text{head}_i + u, \text{head}_j + v) \text{Pair}(l + u, k + v)$$

$$E_g = \sum_{i=1}^M \sum_{l \in D[i]} \sum_{k \in R[i, i+1, l]} y_{(i,l), (i+1,k)} G(i, l, k),$$

- Gap between two successive cores