

# The Minimum Label Spanning Tree Problem and Some Variants

Yupei Xiong, *Univ. of Maryland*

Bruce Golden, *Univ. of Maryland*

Edward Wasil, *American Univ.*

---

Presented at Oklahoma State University  
October 7, 2005

# Outline of Lecture

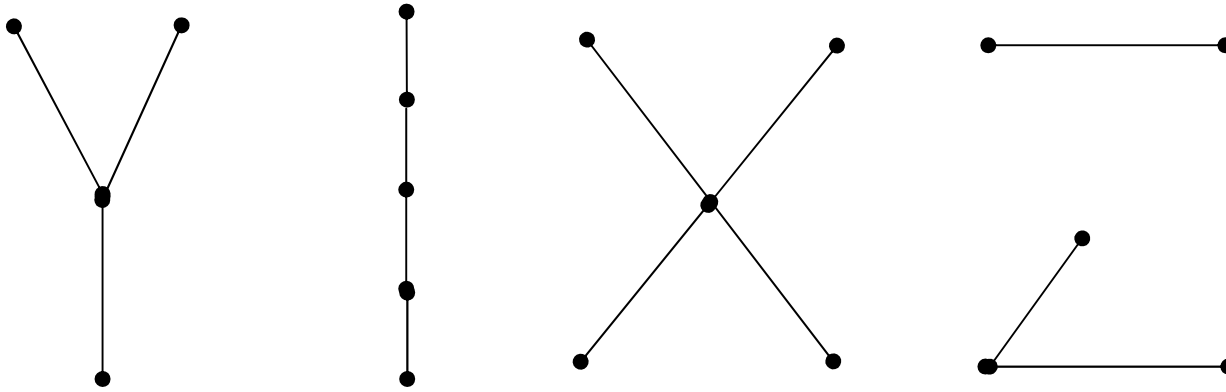
---

- 10 - Minute Introduction to Graph Theory and Complexity
- Introduction to the MLST Problem
- A GA for the MLST Problem
- Four Modified Versions of the Benchmark Heuristic
- A Modified Genetic Algorithm
- Results and Conclusions

# Defining Trees

---

- A graph with no cycles is *acyclic*
- A *tree* is a connected acyclic graph

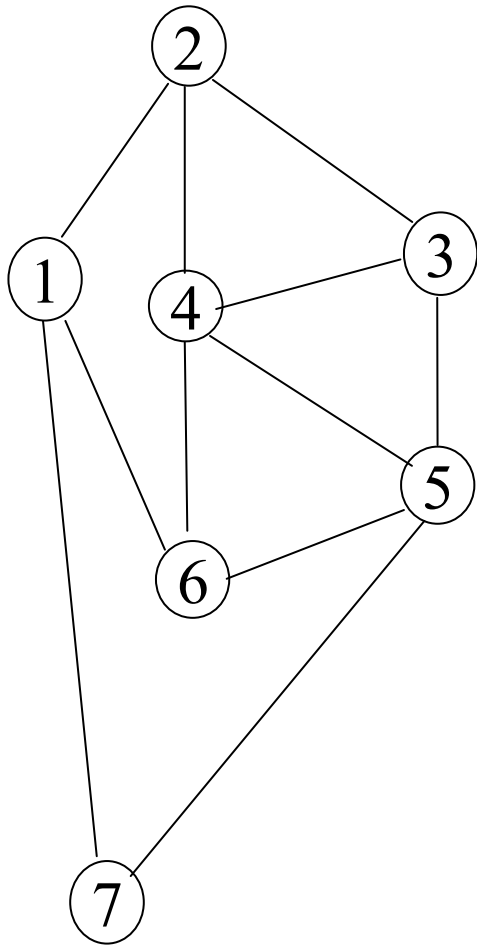


Some examples of trees

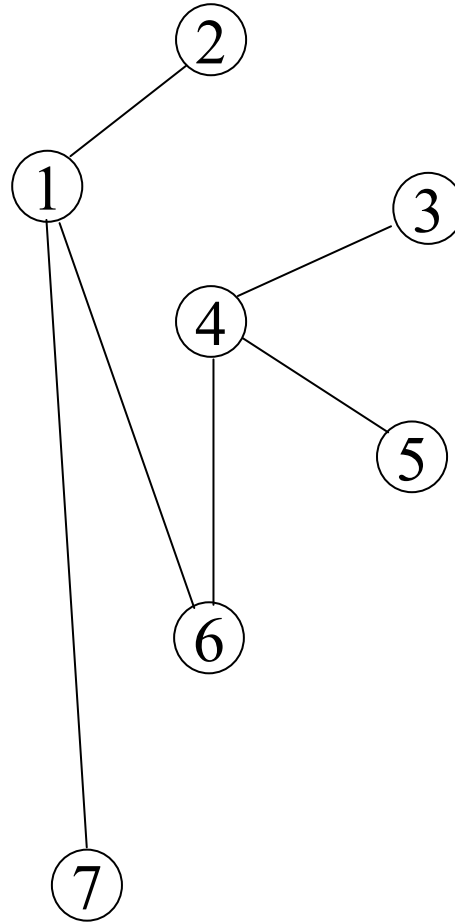
- A *spanning tree* of a graph  $G$  contains all the nodes of  $G$

# Spanning Trees

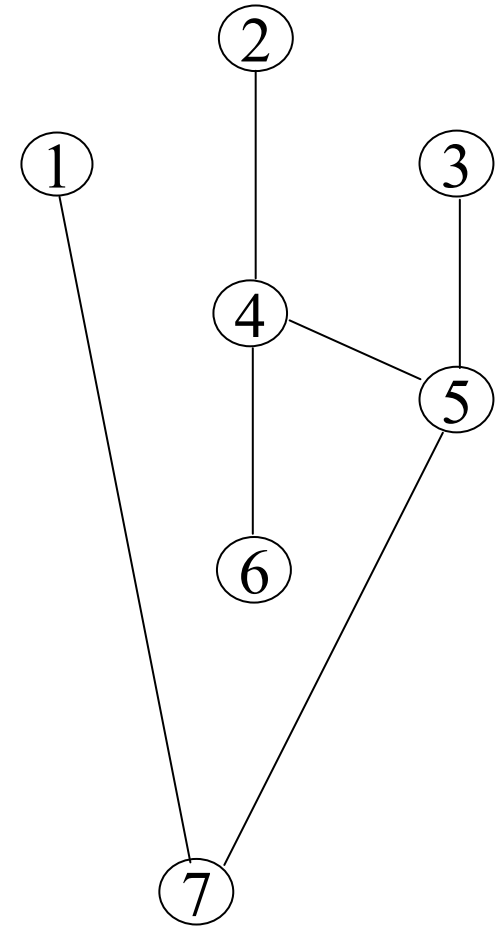
---



Graph G



A spanning tree of G



Another spanning tree of G

# Minimal Spanning Trees

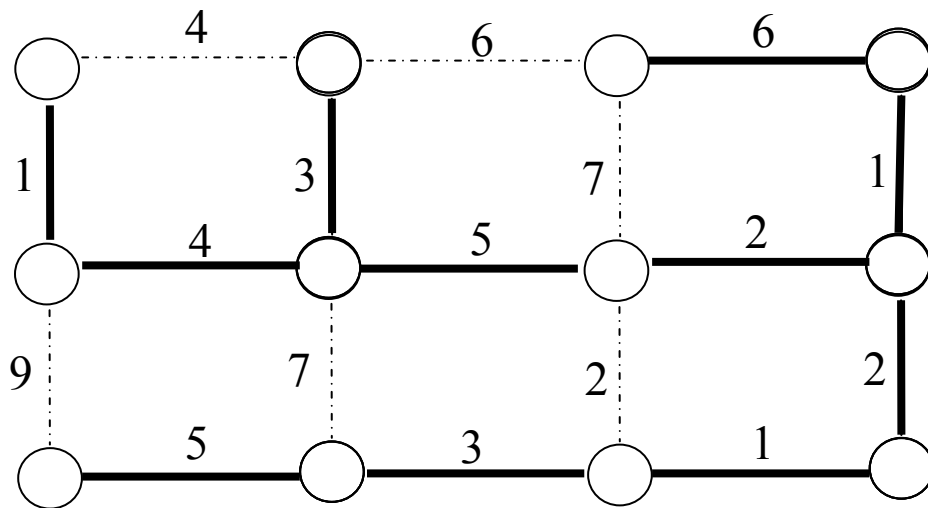
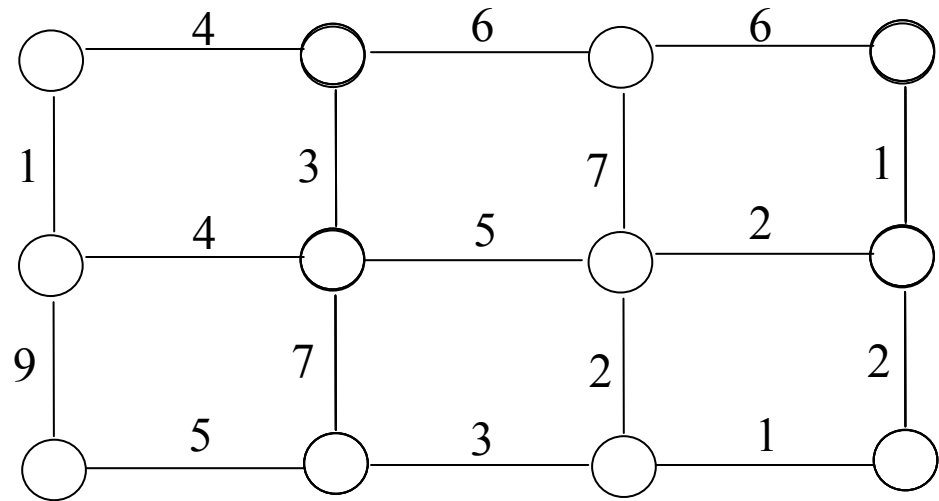
---

A network problem for which there is a simple solution method is the selection of a minimum spanning tree from an undirected network over  $n$  cities

- The cost of installing a communication link between cities  $i$  and  $j$  is  $c_{ij} = c_{ji} \geq 0$
- Each city must be connected, directly or indirectly, to all others, and this is to be done at minimum total cost
- Attention can be confined to trees, because if the network contains a cycle, removing one link of the cycle leaves the network connected and reduces cost

# A Minimal Spanning Tree

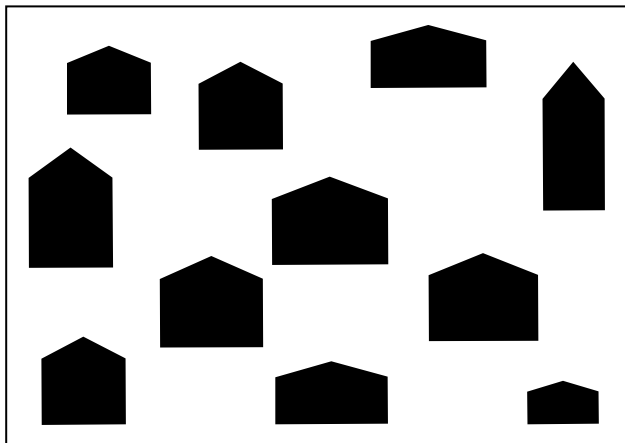
Original Network



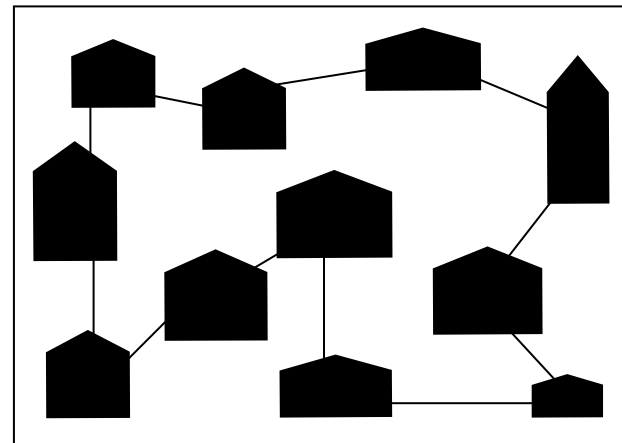
Minimum Spanning Tree

# The Traveling Salesman Problem

- Imagine a suburban college campus with 140 separate buildings scattered over 800 acres of land
- To promote safety, a security guard must inspect each building every evening
- The goal is to sequence the 140 buildings so that the total time (travel time plus inspection time) is minimized
- This is an example of the well-known TSP



Original problem



Possible solution

# Analysis of Algorithms

---

## ■ Definitions

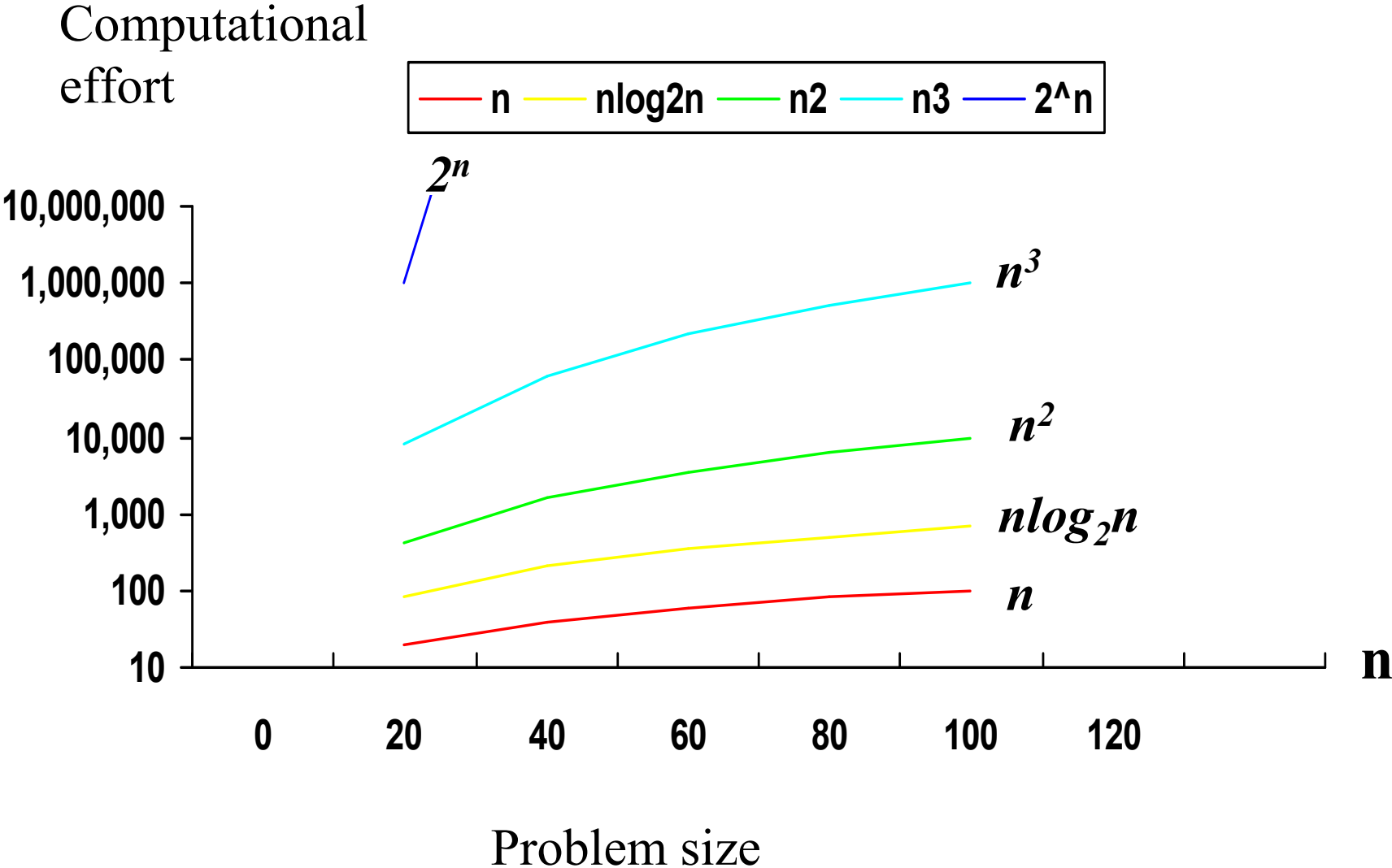
- Algorithm- method for solving a class of problems on a computer
- Optimal algorithm –verifiable optimal solution
- Heuristic algorithm –feasible solution

## ■ Performance Measures

- Number of basic computations / Running time
- Computational effort
  - Problem size
  - Player one
  - Player two



# Computational Effort as a Function of Problem Size



# Good vs. Bad Algorithms

---

- Terminology
  - Researchers have emphasized the importance of finding polynomial time algorithms, by referring to all such polynomial algorithms as inherently good
  - Algorithms that are *not* polynomially bounded, are labeled inherently bad
- Good Optimal Algorithms Exist for these Problems
  - Transportation problem
  - Minimal spanning tree problem
  - Shortest path problem
  - Linear programming

# High Quality Heuristic Algorithms

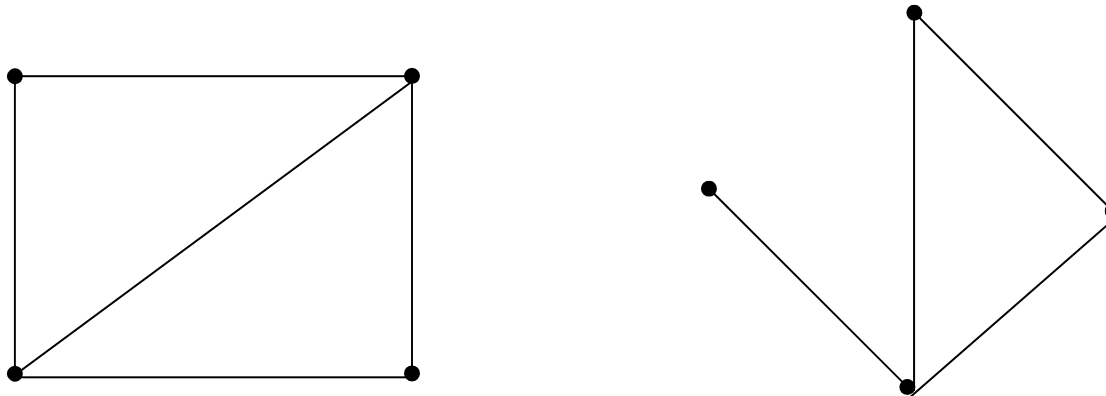
---

- Good Optimal Algorithms Don't Exist for these Problems
  - Traveling salesman problem (TSP)
  - Minimum label spanning tree problem (MLST)
- Why Focus on Heuristic Algorithms?
  - For the above problems, optimal algorithms are not practical
  - Efficient, near optimal heuristics are needed to solve real-world problems
  - The key is to find fast, high-quality heuristic algorithms

# One More Concept from Graph Theory

---

- A disconnected graph consists of two or more connected graphs
- Each of these connected subgraphs is called a component



A disconnected graph with two components

# Introduction

---

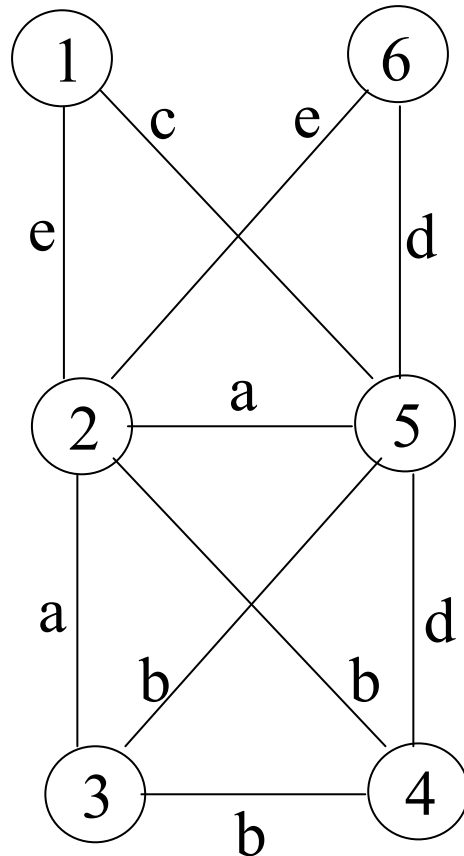
- The Minimum Label Spanning Tree (MLST) Problem
  - Communications network design
  - Edges may be of different types or media (e.g., fiber optics, cable, microwave, telephone lines, etc.)
  - Each edge type is denoted by a unique letter or color
  - Construct a spanning tree that minimizes the number of colors

# Introduction

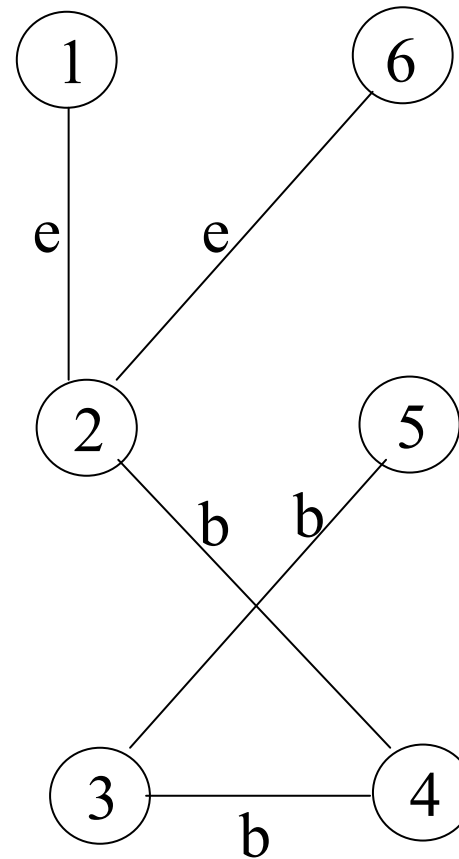
---

- A Small Example

Input



Solution



# Literature Review

---

- Where did we start?
  - Proposed by Chang & Leu (1997)
  - The MLST Problem is NP-hard
  - Several heuristics had been proposed
  - One of these, MVCA (maximum vertex covering algorithm), was very fast and effective
  - Worst-case bounds for MVCA had been obtained

# Literature Review

---

- An optimal algorithm (using backtrack search) had been proposed
- On small problems, MVCA consistently obtained nearly optimal solutions
- A description of MVCA follows



# Description of MVCA

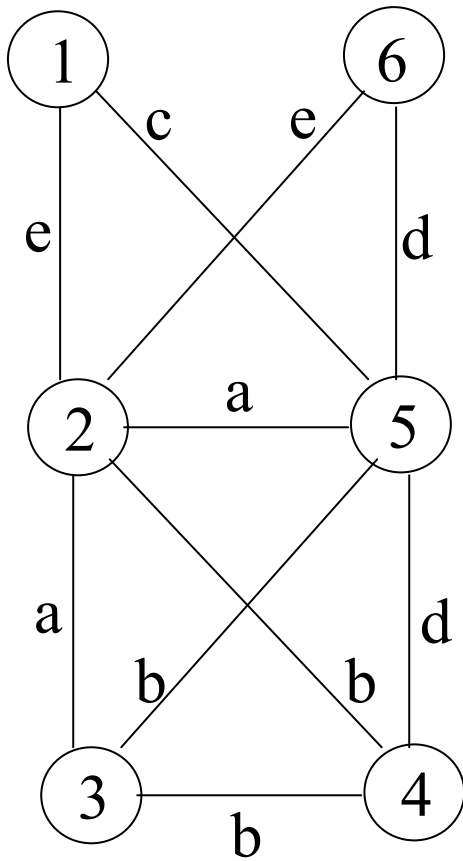
---

0. Input:  $G (V, E, L)$ .
1. Let  $C \leftarrow \{ \}$  be the set of used labels.
2. repeat
3.     Let  $H$  be the subgraph of  $G$  restricted to  $V$  and edges with labels from  $C$ .
4.     for all  $i \in L - C$  do
5.         Determine the number of connected components when inserting all edges with label  $i$  in  $H$ .
6.     end for
7.     Choose label  $i$  with the smallest resulting number of components and do:  $C \leftarrow C \cup \{i\}$ .
8. Until  $H$  is connected.

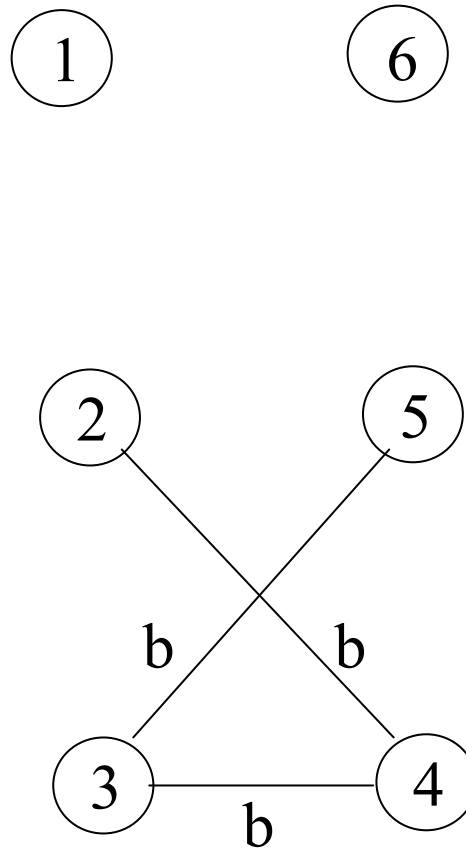
# How MVCA Works

---

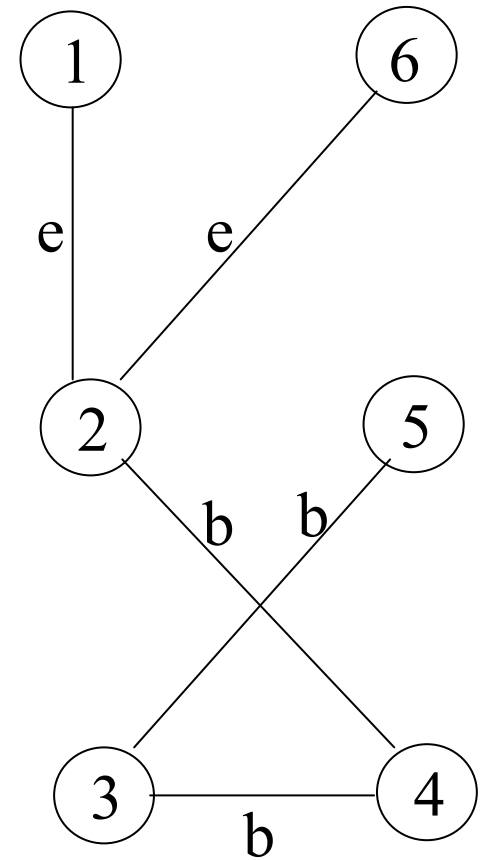
**Input**



**Intermediate  
Solution**



**Solution**



# Worst-Case Results

---

1. Krumke, Wirth (1998):

$$\frac{\text{MVCA}}{\text{OPT}} \leq 1 + 2 \ln n$$

2. Wan, Chen, Xu (2002):

$$\frac{\text{MVCA}}{\text{OPT}} \leq 1 + \ln(n-1)$$

3. Xiong, Golden, Wasil (2005):

$$\frac{\text{MVCA}}{\text{OPT}} \leq H_b = \sum_{i=1}^b \frac{1}{i} < 1 + \ln b$$

where  $b = \max$  label frequency, and

$H_b = b^{\text{th}}$  harmonic number

# Some Observations

---

- The Xiong, Golden, Wasil worst-case bound is tight
- Unlike the MST, where we focus on the edges, here it makes sense to focus on the labels or colors
- Next, we present a genetic algorithm (GA) for the MLST problem

# Genetic Algorithm: Overview

---

- Randomly choose  $p$  solutions to serve as the initial population
- Suppose  $s[0], s[1], \dots, s[p-1]$  are the individuals (solutions) in generation 0
- Build generation  $k$  from generation  $k-1$  as below

For each  $j$  between 0 and  $p-1$ , do:

$t[j] = \text{crossover} \{ s[j], s[(j+k) \bmod p] \}$

$t[j] = \text{mutation} \{ t[j] \}$

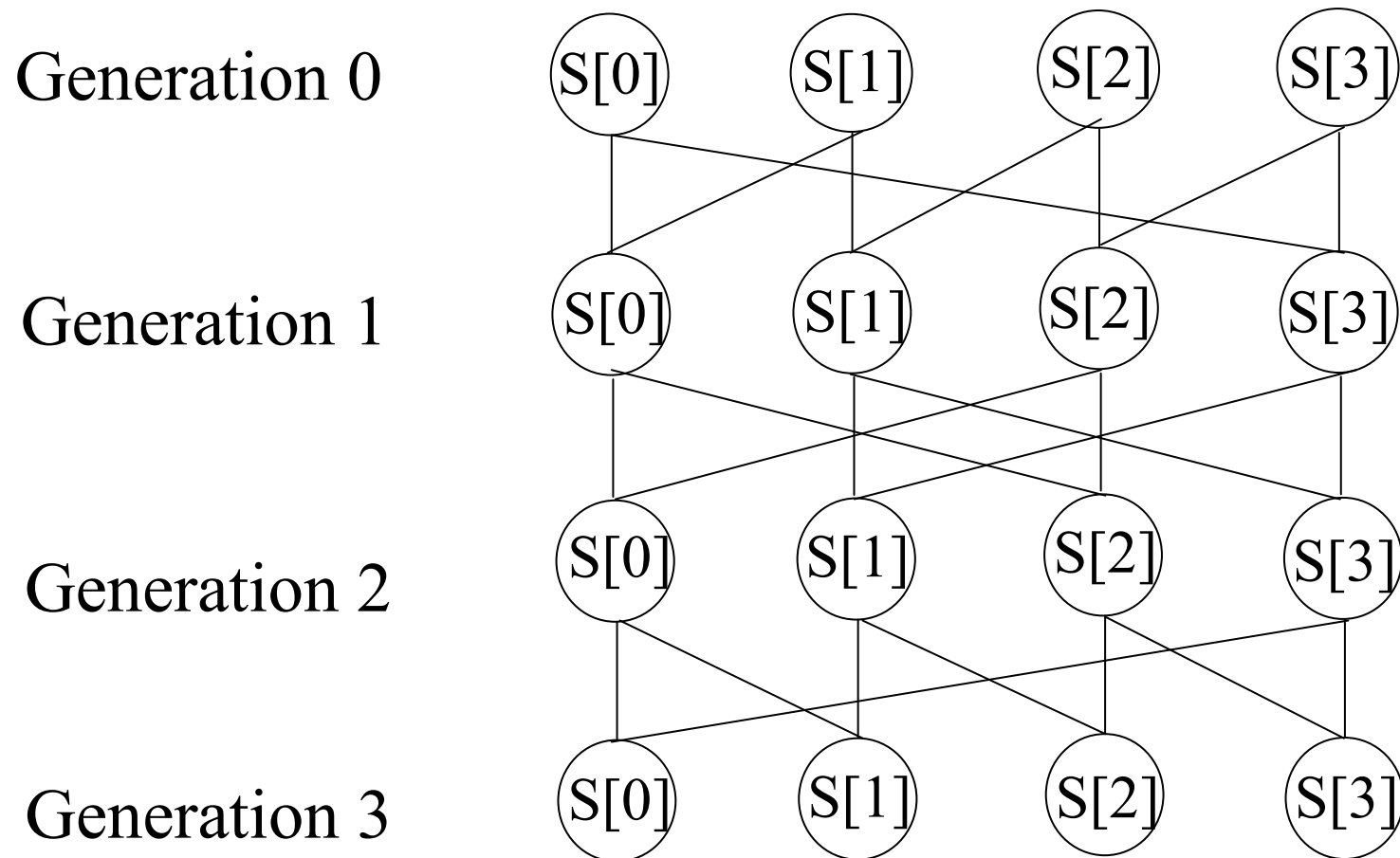
$s[j] = \text{the better solution of } s[j] \text{ and } t[j]$

End For

- Run until generation  $p-1$  and output the best solution from the final generation

# Crossover Schematic (p = 4)

---



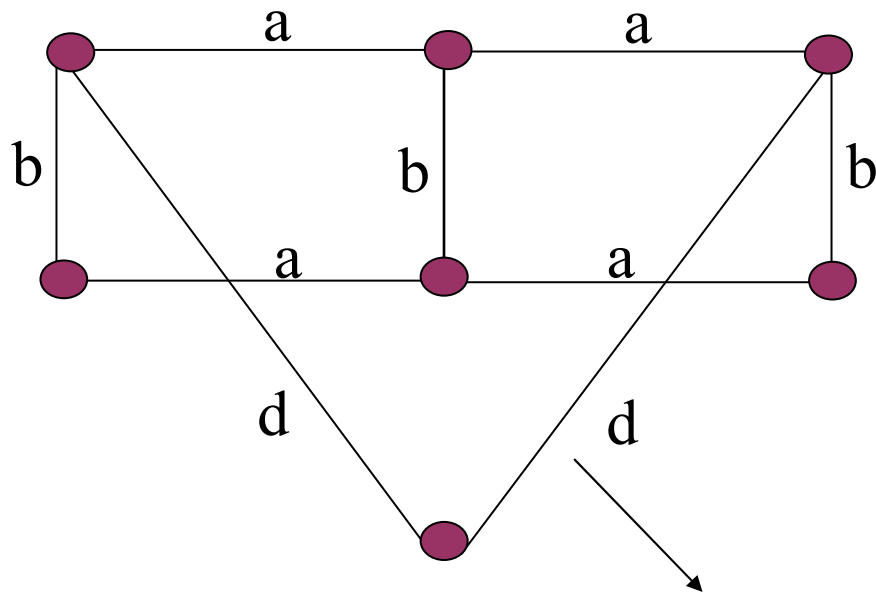
# Crossover

---

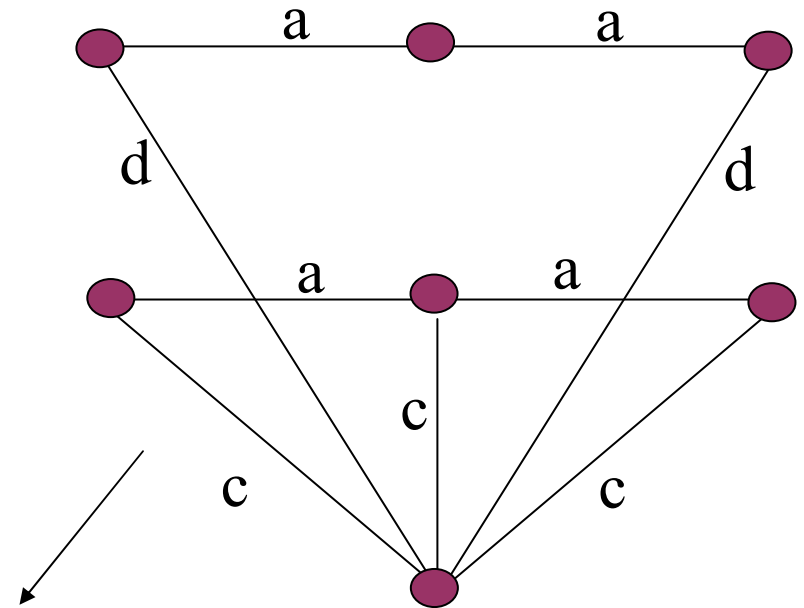
- Given two solutions  $s [ 1 ]$  and  $s [ 2 ]$ , find the child  $T = \text{crossover} \{ s [ 1 ], s [ 2 ] \}$
- Define each solution by its labels or colors
- Description of Crossover
  - a. Let  $S = s [ 1 ] \cup s [ 2 ]$  and  $T$  be the empty set
  - b. Sort  $S$  in decreasing order of the frequency of labels in  $G$
  - c. Add labels of  $S$ , from the first to the last, to  $T$  until  $T$  represents a feasible solution
  - d. Output  $T$

# An Example of Crossover

$s[1] = \{a, b, d\}$



$s[2] = \{a, c, d\}$



$T = \{ \}$

$S = \{a, b, c, d\}$

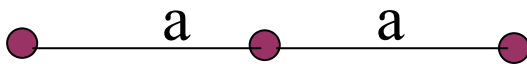
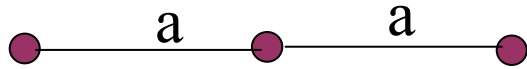
Ordering: a, b, c, d



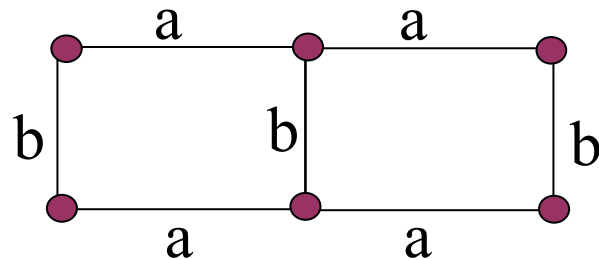
# An Example of Crossover

---

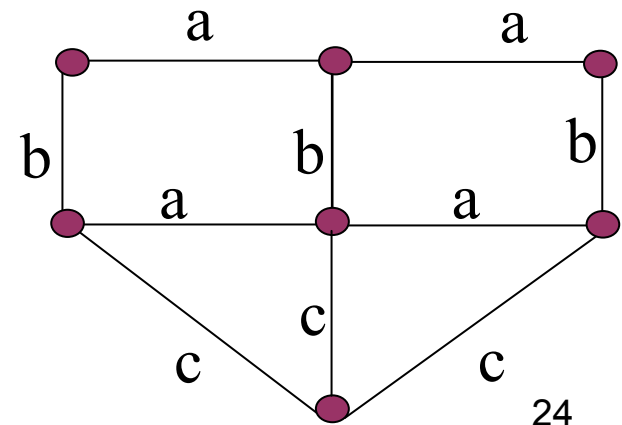
$T = \{ a \}$



$T = \{ a, b \}$



$T = \{ a, b, c \}$



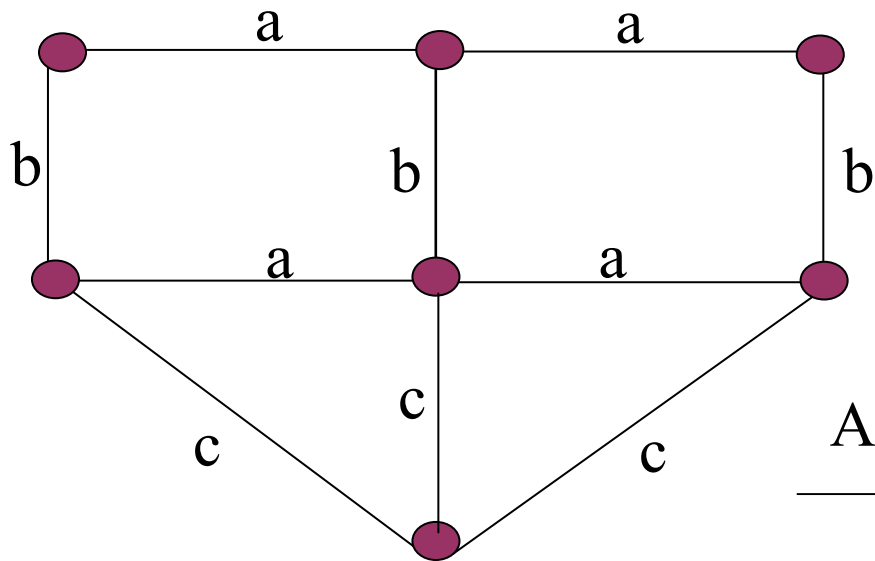
# Mutation

---

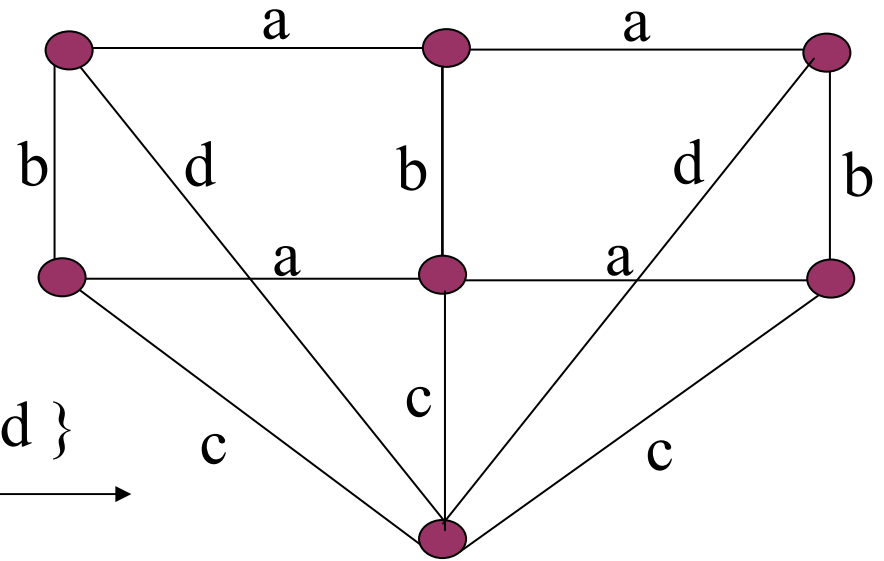
- Given a solution  $S$ , find a mutation  $T$
- Description of Mutation
  - a. Randomly select  $c$  not in  $S$  and let  $T = S \cup c$
  - b. Sort  $T$  in decreasing order of the frequency of the labels in  $G$
  - c. From the last label on the above list to the first, try to remove one label from  $T$  and keep  $T$  as a feasible solution
  - d. Repeat the above step until no labels can be removed
  - e. Output  $T$

# An Example of Mutation

$S = \{ a, b, c \}$



$S = \{ a, b, c, d \}$



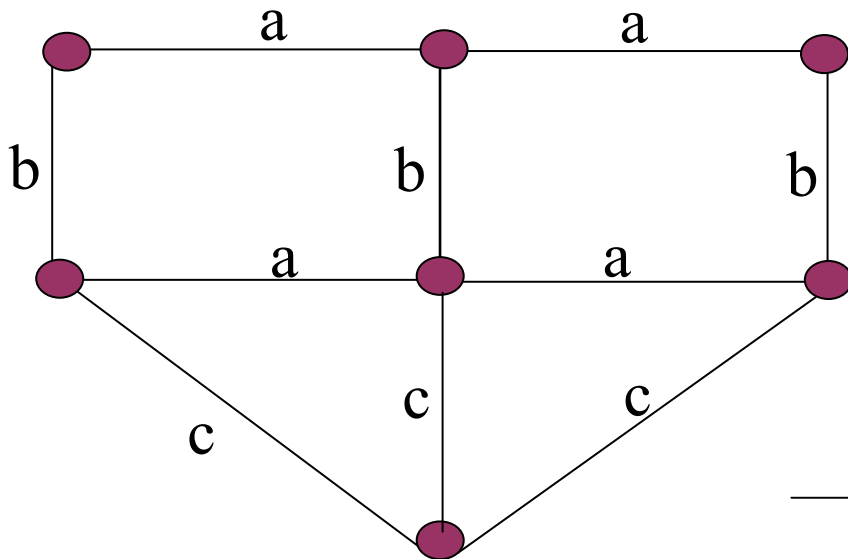
Add { d }

Ordering: a, b, c, d

# An Example of Mutation

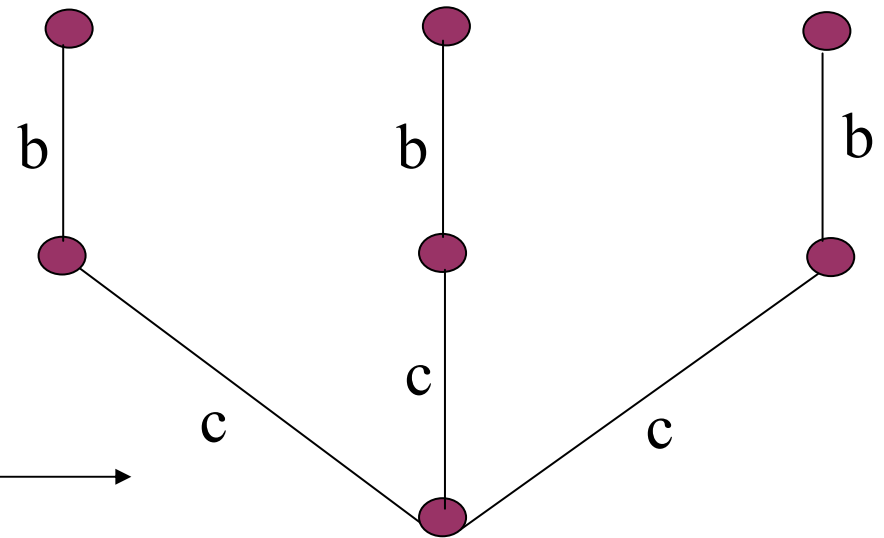
Remove { d }

$S = \{ a, b, c \}$



Remove { a }

$S = \{ b, c \}$



$T = \{ b, c \}$

# Three Modified Versions of MVCA

---

- Voss et al. (2005) implement MVCA using their pilot method
- The results were quite time-consuming
- We added a parameter ( % ) to improve the results
- Three modified versions of MVCA
  - MVCA1 uses % = 100
  - MVCA2 uses % = 10
  - MVCA3 uses % = 30

# MVCA1

---

- We try each label in  $L$  ( $\% = 100$ ) as the first or pilot label
- Run MVCA to determine the remaining labels
- We output the best solution of the  $l$  solutions obtained
- For large  $l$ , we expect MVCA1 to be very slow

# MVCA2 (and MVCA3)

---

- We sort all labels by their frequencies in  $G$ , from highest to lowest
- We select each of the top 10% ( $\% = 10$ ) of the labels to serve as the pilot label
- Run MVCA to determine the remaining labels
- We output the best solution of the  $l/10$  solutions obtained
- MVCA2 will be faster than MVCA1, but not as effective
- MVCA3 selects the top 30% ( $\% = 30$ ) and examines  $3l/10$  solutions
- MVCA3 is a compromise approach

# A Randomized Version of MVCA (RMVCA)

---

- We follow MVCA in spirit
- At each step, we consider the three most promising labels as candidates
- We select one of the three labels
  - The best label is selected with prob. = 0.4
  - The second best label is selected with prob. = 0.3
  - The third best label is selected with prob. = 0.3
- We run RMVCA 50 times for each instance and output the best solution



# A Modified Genetic Algorithm (MGA)

- We modify the crossover operation described earlier
- We take the union of the parents (i.e.,  $S = S_1 \cup S_2$ ) as before
- Next, apply MVCA to the subgraph of  $G$  with label set  $S$  ( $S \subseteq L$ ), node set  $V$ , and the edge set  $E'$  ( $E' \subseteq E$ ) associated with  $S$
- The new crossover operation is more time-consuming than the old one
- The mutation operation remains as before

# Computational Results

---

- 48 combinations:  $n = 50$  to  $200$  /  $l = 12$  to  $250$  / density =  $0.2, 0.5, 0.8$
- 20 sample graphs for each combination
- The average number of labels is compared

# Performance Comparison

	MVCA	GA	MGA	MVCA1	MVCA2	MVCA3	RMVCA	Row Total
MVCA	-	3	0	0	0	0	0	3
GA	30	-	0	1	9	4	2	46
MGA	33	30	-	10	20	16	16	125
MVCA1	35	30	10	-	24	20	18	137
MVCA2	31	20	5	0	-	0	6	62
MVCA3	34	27	8	0	23	-	11	103
RMVCA	35	30	7	3	20	10	-	105

Summary of computational results with respect to accuracy for seven heuristics on 48 cases. The entry  $(i, j)$  represents the number of cases heuristic  $i$  generates a solution that is better than the solution generated by heuristic  $j$ .

# Running Times

---

	MVCA	GA	MGA	MVCA1	MVCA2	MVCA3	RMVCA
$n = 100, l = 125, d = 0.2$	0.05	1.80	7.50	8.25	0.80	2.30	3.85
$n = 150, l = 150, d = 0.5$	0.10	1.85	4.90	11.85	1.15	3.45	4.75
$n = 150, l = 150, d = 0.2$	0.15	3.45	13.55	21.95	2.15	6.35	8.45
$n = 150, l = 187, d = 0.5$	0.15	2.20	6.70	21.70	2.00	6.15	7.50
$n = 150, l = 187, d = 0.2$	0.20	3.95	17.55	39.35	3.60	11.20	11.90
$n = 200, l = 100, d = 0.2$	0.15	3.75	11.40	11.25	1.15	3.35	6.75
$n = 200, l = 200, d = 0.8$	0.25	2.45	5.80	26.70	2.70	8.00	8.65
$n = 200, l = 200, d = 0.5$	0.25	3.45	10.15	38.65	3.90	10.15	12.00
$n = 200, l = 200, d = 0.2$	0.35	6.20	26.65	68.25	6.85	20.35	20.55
$n = 200, l = 250, d = 0.8$	0.30	3.05	7.55	52.25	5.25	15.35	12.95
$n = 200, l = 250, d = 0.5$	0.30	3.95	12.60	69.90	6.80	20.35	16.70
$n = 200, l = 250, d = 0.2$	0.50	6.90	33.15	124.35	12.10	35.80	28.80
Average running time	0.23	3.58	13.13	41.20	4.04	11.90	11.90

Running times for 12 demanding cases (in seconds).

# Conclusions

---

- We presented three modified (deterministic) versions of MVCA, a randomized version of MVCA, and a modified GA
- All five of the modified procedures generated better results than MVCA and GA, but were more time-consuming
- With respect to running time and performance, MGA seems to be the best