The Minimum Labeling Spanning Tree Problem: Heuristic and Metaheuristic Approaches

Bruce Golden
Yupei Xiong
Edward Wasil

OR Workshop in Honor of Paolo Carraresi
Università di Pisa, March 3, 2004
Introduction

- The Minimum Labeling 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
Where did we start?

- The MLST Problem is NP-hard
- Several heuristics had been proposed
- One of these, MVCA (version 2), 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.

- See [Chang & Leu, 1997], [Krumke & Wirth, 1998], [Wan, Chen & Xu, 2002], and [Bruggemann, Monnot & Woeginger, 2003].
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 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 and spans $V$. 
How MVCA Works

Input

Intermediate Solution

Solution
Worst-Case Results

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

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

   \[
   \frac{\text{MVCA}}{\text{OPT}} \leq H_b = \sum_{i=1}^{b} \frac{1}{i} < 1 + \ln b
   \]
   where \( b = \text{max label frequency} \), and
   \( H_b = b^{th} \) harmonic number
A Perverse-Case Example

- **MVCA**
  - It might add colors associated with 3 polygons first
  - Then, it must add colors associated with 4 rays

- **Optimal Solution**
  - Add colors associated with 4 rays first
  - Then, add color associated with 1 polygon
Given an integer $k \geq 3$, build a graph $G = (V, E)$ with $k^2$ nodes and $2k - 1$ labels, where the edges form $k - 1$ concentric polygons and $k$ rays (as below for $k = 3$)

\[
\frac{\text{MVCA}}{\text{OPT}} = \frac{7}{5}
\]

In the previous example, we had $k = 4$ and
Some Observations

- In general, the result \( \frac{\text{MVCA}}{\text{OPT}} = \frac{2k-1}{k+1} \), which approaches 2 for large \( k \), is possible.

- The labels associated with the rays in the perverse-case example are cut labels.

- Definition: A label \( c \) is called a cut label if the removal of all edges with label \( c \) disconnects graph \( G \).

- These labels must be in any solution.

- Next, we show that the Xiong, Golden, Wasil worst-case bound is tight.
A Worst-Case Example
A Worst-Case Example - continued
A Worst-Case Example - continued
A Worst-Case Example - continued

Suppose we implement MVCA by adding cut labels first? Is the bound still tight?
More Observations

- For the worst-case example with $b = 3$ and $n = 19$,
  \[
  \frac{\text{MVCA}}{\text{OPT}} = \frac{11}{6} = 1 + \frac{1}{2} + \frac{1}{3} = H_3
  \]

- 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 \( s[0], s[1], \ldots, s[p-1] \) 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:
  
  \[
  t[j] = \text{crossover} \{ s[j], s[(j+k) \mod p] \}
  \]
  
  \[
  t[j] = \text{mutation} \{ t[j] \}
  \]
  
  \[
  s[j] = \text{the better solution of } s[j] \text{ and } t[j]
  \]

  End For

- Run until generation \( p-1 \) and output the best solution from the final generation
Crossover Schematic (p = 4)

Generation 0

Generation 1

Generation 2

Generation 3
**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 its labels
  
  c. Add labels of \( S \) to \( T \), until \( T \) represents a connected subgraph \( H \) of \( G \) that spans \( V \)
  
  d. Output \( T \)
An Example of Crossover

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

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

\[ T = \{ \} \]

\[ S = \{ a, b, c, d \} \]

Ordering: a, b, c, d
An Example of Crossover

\[ T = \{ \text{a} \} \]

\[ T = \{ \text{a}, \text{b} \} \]

\[ T = \{ \text{a}, \text{b}, \text{c} \} \]
Mutation

- Given a solution S, find a mutation T
- Description of Mutation
  
  a. Randomly select a label $c$ not in S and let $S = S \cup \{ c \}$
  
  b. Sort S in decreasing order of the frequency of its labels
  
  c. From the last label on the above list to the first, try to remove one label from S while preserving a connected subgraph $H$ of $G$ that spans $V$
  
  d. Continue until no longer possible
  
  e. Call this solution T and output T
An Example of Mutation

\[ S = \{ \text{a, b, c} \} \]

\[ S = \{ \text{a, b, c, d} \} \]

Ordering: \text{a, b, c, d}
An Example of Mutation

Remove \{ d \}

\[ S = \{ a, b, c \} \]

Remove \{ a \}

\[ S = \{ b, c \} \]

\[ T = \{ b, c \} \]
Computational Results

- 81 combinations: \( n = 20 \) to \( 200 \) / \( l = 20 \) to \( 250 \) / density = \( .2, .5, .8 \) / \( p = 20 \)

- 20 sample graphs for each combination

- The average number of labels is compared

- GA beats MVCA in 53 of 81 cases (16 ties, 12 worse)

- No instance required more than 2 seconds CPU time on a Pentium 4 PC with 1.80 GHz and 256 MB RAM
Conclusions & Future Work

- The GA is fast, conceptually simple, and powerful
- It contains a single parameter
- We think GAs can be applied successfully to a host of other NP-hard problems
- Future work
  - More extensive computational testing
  - Add edge costs
  - Examine other variants