The Label-Constrained Minimum Spanning Tree (LCMST) Problem

Yupei Xiong, Sysmind LLC

Bruce Golden, University of Maryland

Edward Wasil, American University

Si Chen, University of Maryland

INFORMS Meeting, College Park, MD

March 29, 2008

LCMST Problem

- **Problem:** Given an undirected labeled graph G = (V,E,L) and a positive integer K, where V is the set of nodes, E is the set of edges and L is the set of labels. Each edge in E has a cost and is colored by a label in L. The goal of the LCMST problem is to find the minimum-cost spanning tree of G with at most K distinct labels.
- The LCMST is NP-hard.
- The LCMST problem is somewhat more realistic than the minimum labeling spanning tree (MLST) problem.

Some Observations

- A feasible solution of the LCMST problem is a label set C which satisfies two conditions:

  1. C induces a connected subgraph and spans all the nodes.

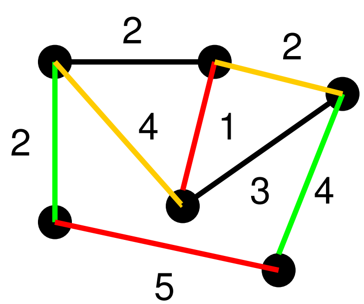
  2. C contains at most K labels.

- For each feasible solution, the minimum-cost spanning tree of the induced subgraph can be obtained by Prim's algorithm in polynomial running time.

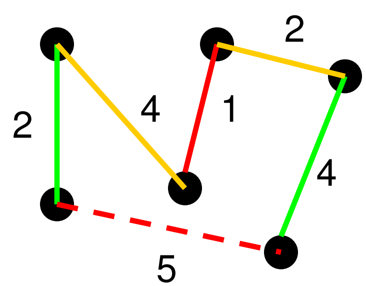
- A cost function $f: 2^L \rightarrow R^+$ can be well-defined. If A is a feasible solution, then f(A) is the MST cost of the induced subgraph. If A is not a feasible solution, $f(A) = \infty$.

A Small Example (K = 3)

Input Graph

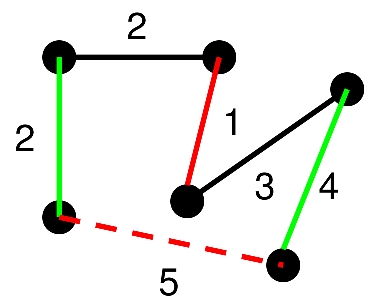


Solution A



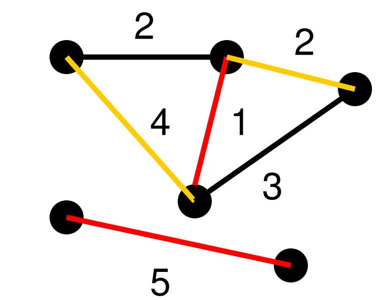
$f(A) = 13$

Solution B



$f(B) = 12$

Solution C



$f(C) = \infty$

Local Search 1 (LS1)

- LS1 begins with an arbitrary feasible solution A with K labels. We can assume that $A=\{a_1,a_2,\ldots,a_K\}$.
- We replace the label a_1 by some label b_1 in the set L, such that $f(A-\{a_1\}+\{b_1\}) = \min \{f(A-\{a_1\}+\{b\}: b \text{ in } L\}$. Then we set $A := A-\{a_1\}+\{b_1\}$. It is possible that $b_1=a_1$.
- We improve A by replacing $a_2,\ldots,a_K$ by $b_2,\ldots,b_K$, respectively. This is called a Replacing Loop (RL).
- We continue RLs until no improvement can be made.
- We output the final label set A.

Local Search 2 (LS2)

- LS2 begins with an arbitrary feasible solution A with K labels. We can assume that $A=\{a_1,a_2,\ldots,a_K\}$.
- We add a label a_{K+1} in L-A to A. Then we remove some label a_r in A, such that

  $f(A+\{a_{K+1}\}-\{a_r\}) = \min \{f(A+\{a_{K+1}\}-\{a_t\}: 1 \leqslant t \leqslant K+1\}$.
  Then we set $A := A+\{a_{K+1}\}-\{a_r\}$. It is possible that $a_r=a_{K+1}$
- We improve A by adding each label in L-A and removing a bad label. This is called a Replacing Loop (RL).
- We continue RLs until no improvement can be made.
- We output the final label set A.

An Example

- We are given an undirected complete labeled graph with 20 nodes and 20 labels. Let K=5. The following table shows the solution and the corresponding value in one replacing loop for LS1 and LS2.

LS1		LS2	
Label Set A	f(A)	Label Set B	f(B)
10,7,14,0,9	4995.21	10,7,14,0,9	4995.21
10,4,14,0,9	4548.53	10,14,0,9,2	4750.22
10,4, 8,0,9	4492.95	10,14,9,2,4	4441.40
10,4,8,14,9	4142.57	10,14,9,4,8	4142.57

Genetic Algorithm (GA)

- Each chromosome of GA is a feasible solution, i.e., a label set with K labels that induces a connected subgraph.
- Queen-bee crossover is applied. The crossover operation is always taken between the queen-bee chromosome and another chromosome.
- The idea of LS1 is applied in the mutation operation.
- The crossover rate is 100%. The mutation rate is 100%.

Crossover

- Given two parents P and Q. Let $R = P \cup Q$ and G_R be the subgraph induced by R. Let C be an empty label set.
- We proceed with Prim's algorithm, beginning with an arbitrary node.
- We grow a tree by adding the minimum-cost edges. If an edge has a new label, we add it to C. So we also grow the label set C.
- When $|C| = K$, we continue Prim's algorithm restricted to the subgraph G_C induced by C.

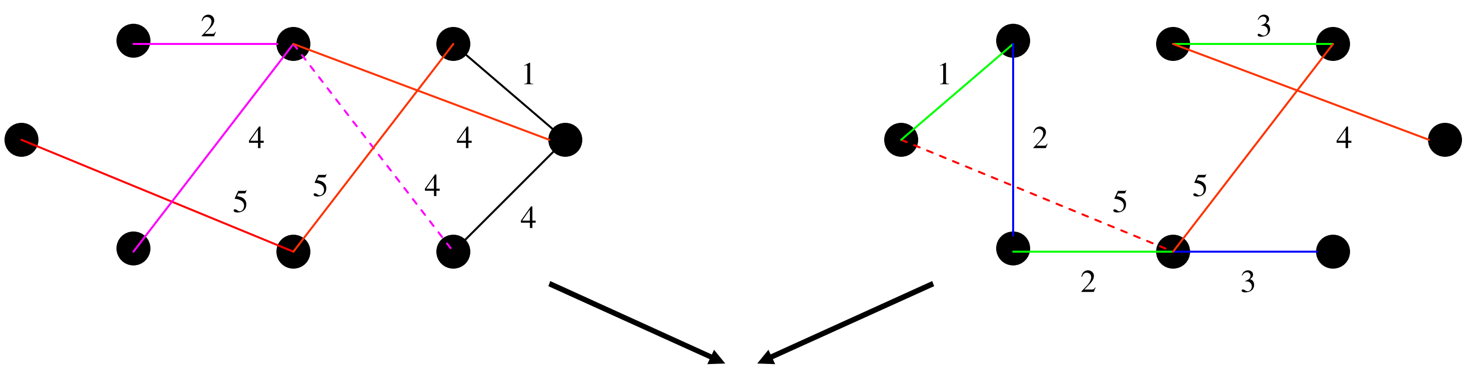
Crossover (continued)

- If C is not a feasible solution, then we run Prim's algorithm again beginning with another node. When the bound K is not too small, we can always obtain a feasible solution C.
- Finally, we output the best solution of P, Q, and C as the child of crossover.

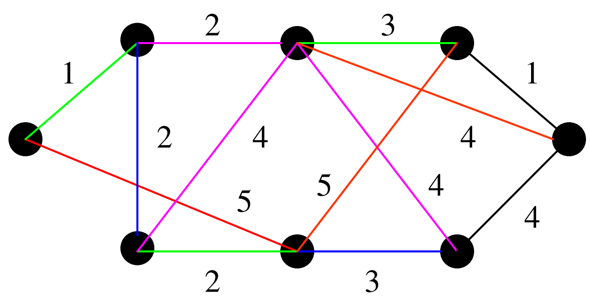
An Example (K = 3)

Solution P, f(P) = 25

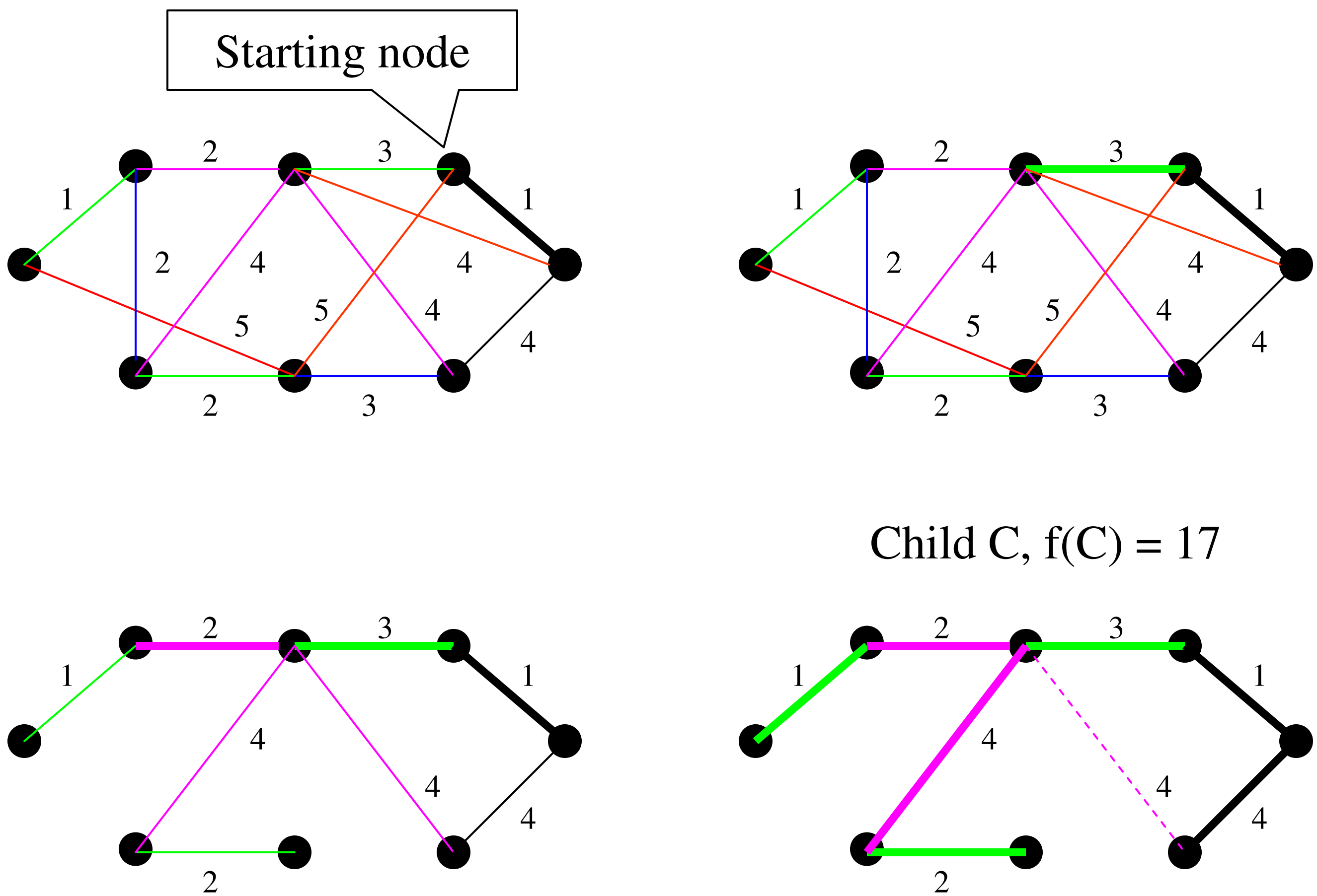
Solution Q, f(Q) = 20



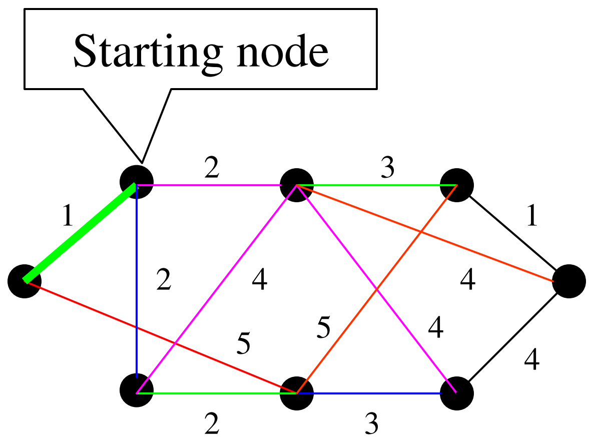
R = P ∪ Q

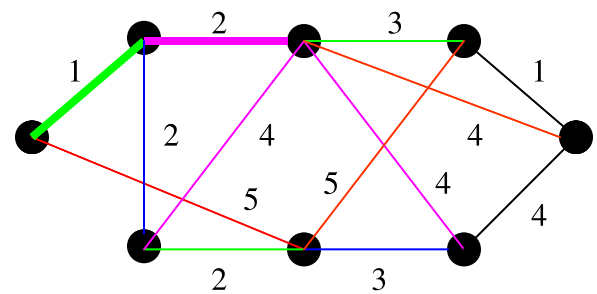


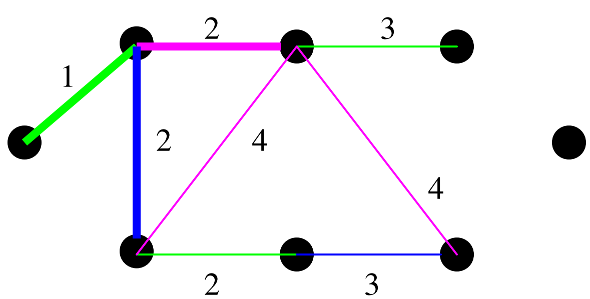
An Example (Case 1)



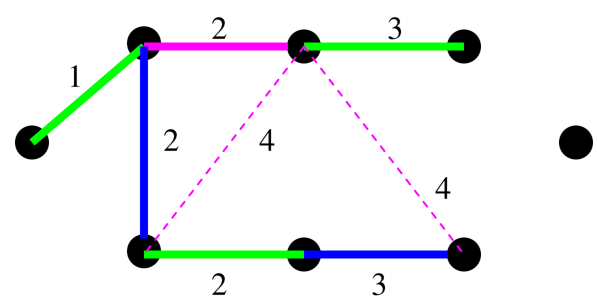
An Example (Case 2)







Child C, f(C) = ∞

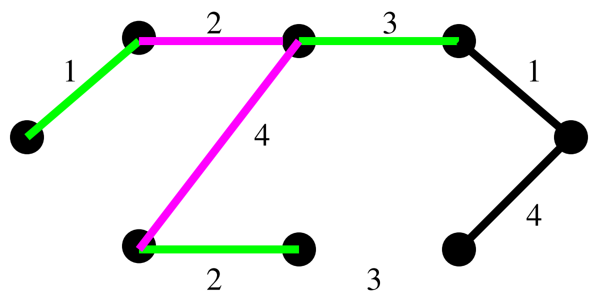


Mutation

- We are given a chromosome $P = \{p_1, p_2, \ldots, p_K\}$.
- We select a random label b in L-P.
- If P can be improved by replacing some p_i by b, then we set $Q = P - \{p_i\} + \{b\}$. So Q is the new chromosome after the mutation operation.
- If P can not be improved by b, then we maintain P to the next generation.
- In the final generation, the queen-bee chromosome is improved by one round of RL in LS1.

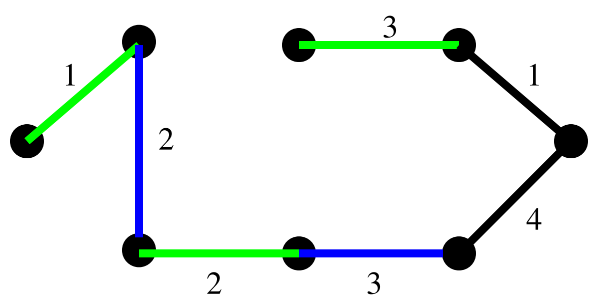
An Example (K = 3)

Solution P, f(P) = 17



Solution Q, f(Q) = 16

Replace pink with blue.



Build Generations

- Suppose there are p chromosome in the first generation.
- We find the queen-bee chromosome, say QB, in each generation.
- We make p-1 crossover operations between QB and the other p-1 chromosomes. Then we do mutation operations for each of the p-1 children. Thus we get p-1 new chromosomes, along with QB, we get the next generation with p chromosomes.
- We stop building generations if the QBs of three consecutive generations do not change.

MIP Formulations

- Notation
  - *E is the set of all edges*
  - *V is the set of all nodes*
  - *n is the total number of nodes*
  - *K is the maximum number of labels allowed in a solution*
  - *E_k is the set of all the edges with color k*
  - *A is the set of all arcs*
  - *c_{ij} is the cost or weight of edge (i, j)*

MIP Formulations

- Variables

$$e_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if arc } < i,j > \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if label } k \text{ is selected} \\ 0 & \text{otherwise.} \end{cases}$$

MIP Formulations (I)

- Single-Commodity Flow Model

$$\min \sum_{(i,j)\in E} c_{ij} e_{ij} \tag{3.A.1}$$

$$\text{subject to} \sum_{(i,j)\in E} e_{ij} = n-1 \tag{3.A.2}$$

$$\sum_{i:(i,j)\in A} f_{ij} - \sum_{l:(j,l)\in A} f_{jl} = 1 \quad \forall j \in V - \{1\} \tag{3.A.3}$$

$$\sum_{i:(i,1)\in A} f_{i1} - \sum_{l:(1,l)\in A} f_{1l} = -(n-1) \tag{3.A.4}$$

MIP Formulations (I)

$$f_{ij} + f_{ji} \leq (n-1) \cdot e_{ij} \quad \forall (i,j) \in E \tag{3.A.5}$$

$$\sum_{(i,j) \in E_k} e_{ij} \leq (n-1) \cdot y_k \quad \forall k \in L \tag{3.A.6}$$

$$\sum_{k \in L} y_k \leq K \tag{3.A.7}$$

$$e_{ij},\ y_k \in \{0,1\} \quad \forall (i,j) \in E \quad \forall k \in L \tag{3.A.8}$$

$$f_{ij} \geq 0 \quad \forall (i,j) \in A. \tag{3.A.9}$$

MIP Formulations (II)

- Multi-Commodity Flow Model

$$\min \sum_{(i,j)\in E} c_{ij} e_{ij} \tag{3.A.10}$$

$$\text{subject to} \sum_{(i,j)\in E} e_{ij} = n-1 \tag{3.A.11}$$

$$\sum_{i:(i,h)\in A} f_{ih}^h - \sum_{l:(h,l)\in A} f_{hl}^h = 1 \quad \forall h \in V - \{1\} \tag{3.A.12}$$

$$\sum_{i:(i,1)\in A} f_{i1}^h - \sum_{l:(1,l)\in A} f_{1l}^h = -1 \quad \forall h \in V - \{1\} \tag{3.A.13}$$

$$\sum_{i:(i,j)\in A} f_{ij}^h - \sum_{l:(j,l)\in A} f_{jl}^h = 0 \quad \forall h \in V - \{1\}, \forall j \neq h \tag{3.A.14}$$

MIP Formulations (II)

$$f_{ij}^h \leq x_{ij} \quad \forall (i,j) \in A, \forall h \in V - \{1\} \tag{3.A.15}$$

$$x_{ij} + x_{ji} \leq e_{ij} \quad \forall (i,j) \in E \tag{3.A.16}$$

$$\sum_{(i,j)\in E_k} e_{ij} \leq (n-1) \cdot y_k \quad \forall k \in L \tag{3.A.17}$$

$$\sum_{k \in L} y_k \leq K \tag{3.A.18}$$

$$e_{ij},\ y_k \in \{0,1\} \quad \forall (i,j) \in E \quad \forall k \in L \tag{3.A.19}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{3.A.20}$$

$$f_{ij}^h \geq 0 \quad \forall (i,j) \in A, \forall h \in V - \{1\}. \tag{3.A.21}$$

Computational Results.

- For each instance, we run LS1 and LS2 5 times and output the best solution. We run GA once and output the best of the final generation.
- For small problems when n = l = 20, 30, 40, 50, we test LS1, LS2, GA and an optimal algorithm (MIP). LS1, LS2 and GA work very well. They reach optimal in many cases. When they do not reach optimal, the gap from optimality is mostly less than 1%.
- For large problems when n $\geqslant$ 100, we test LS1, LS2 and GA. LS1 and LS2 work better than GA. But GA is much faster and the gaps between GA and LS1 or LS2 are less than 1%.

Conclusion

- LC-MST problem is an interesting problem to study. It has more realistic applications.
- The local search methods work very well on the LCMST problem. They take more time and get good results.
- The genetic algorithm is successfully applied to the LCMST problem. The GA is faster and obtains high-quality solutions.

Thank You !