# Disjoint Path Problem

## an application of Lagrangian Relaxation

This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dlbricker@icaen.uiowa.edu

author

# Application

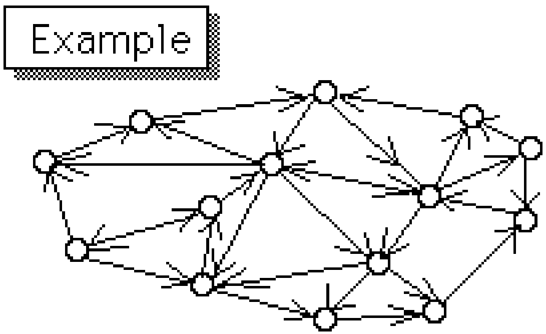
A set of products is to be scheduled on a machine.

(Example: scheduling steel to be rolled (producing varying grades, widths, thicknesses, etc.) in a hot strip mill.)

For some pairs (i,j) of products, no major setup is required if product j immediately follows product i.

We wish to sequence the products so as to minimize the number of major setups required.

Represent the products by nodes in a network, with arc from node i to node j if node j requires no major setup when if follows node i.



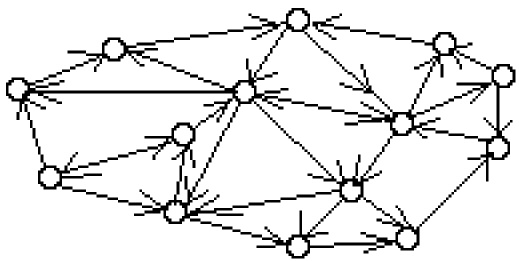

The nodes on a path through the network correspond to a sequence of products which can be produced with a single major setup.

Any two such paths should be ***disjoint***, i.e., should share no common products.

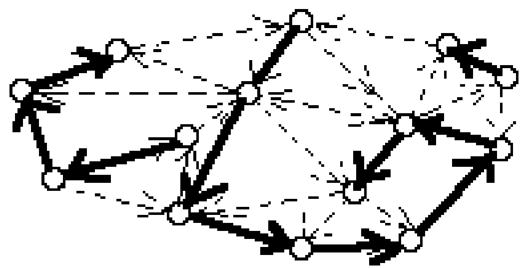
# The Disjoint Path Problem:

Find the minimum number of disjoint paths which span all the nodes of a directed graph.

Example



A feasible solution



(3 paths!)

## PROBLEM STATEMENT:

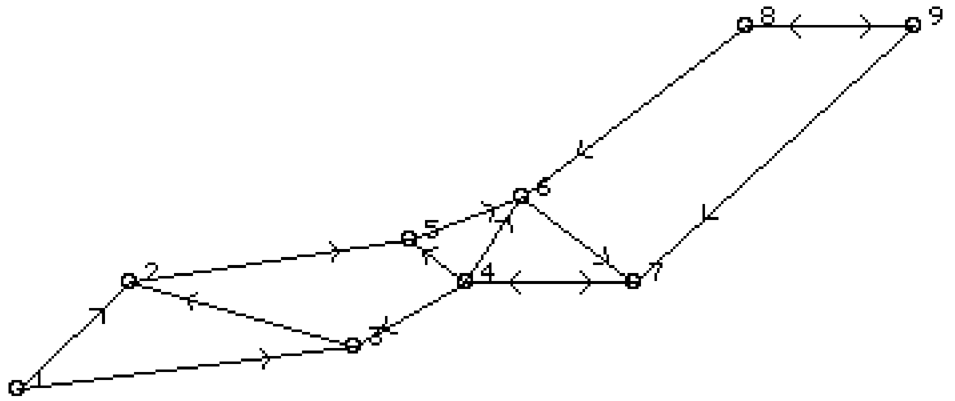
Given a directed graph (digraph) $G = (N,A)$

where $N = \{1, 2, \ldots n\}$ = set of nodes

$A$ = set of arcs ( $A \subseteq N \times N$ )

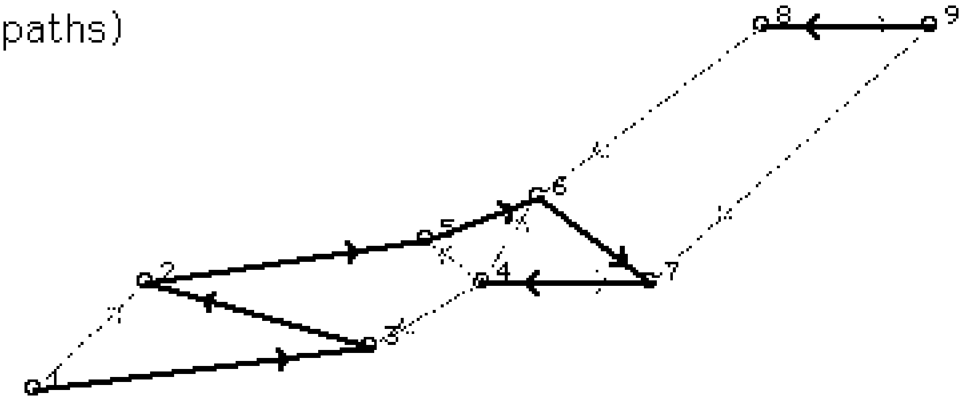
Find the minimum number of paths such that every node $i \in N$ lies on one (and only one) path

# Example:



# The optimal solution:

(2 paths)



# Mathematical Programming Model

Define the variables

$$X_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is included on a path} \\ 0 & \text{otherwise} \end{cases}$$

Clearly $X_{ij} = 1$ for at most one j for each i
and $X_{ij} = 1$ for at most one i for each j

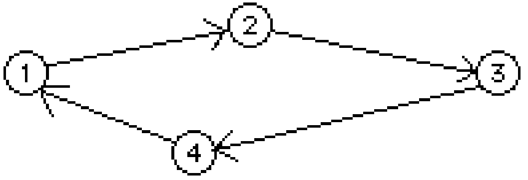
*That is, at most one arc enters node j,*
*and at most one arc leaves node i*

Thus, we have the constraints

$$\sum_{j=1}^{n} X_{ij} \leq 1 \quad \text{for each } i \varepsilon N$$

$$\sum_{i=1}^{n} X_{ij} \leq 1 \quad \text{for each } j \varepsilon N$$

However, the above constraints permit circuits, e.g.,



We must add the constraint that the edges of the subgraph indicated by X form a "forest", i.e., a collection of trees.

*(A tree is a subgraph containing no cycle.)*

In order to facilitate defining the objective function (which is to be the number of paths) in terms of X,

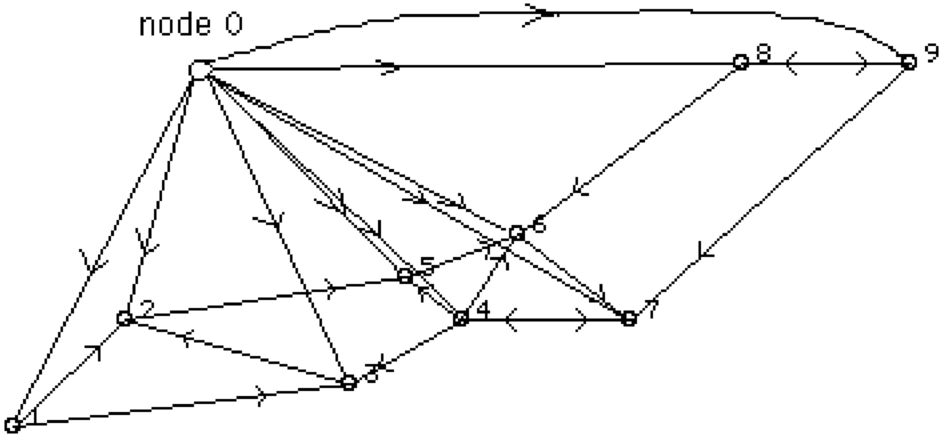
Define a new node 0

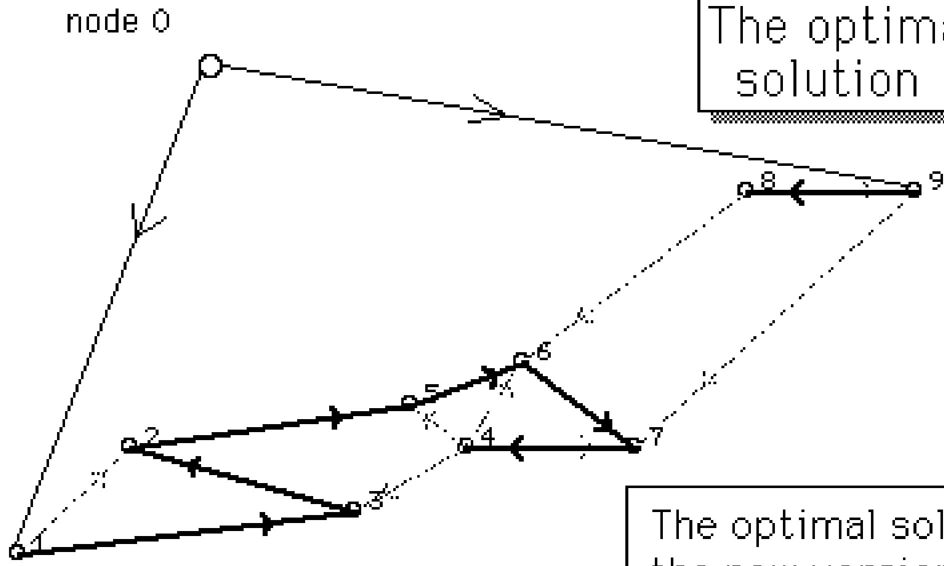
Let $G' = (N', A')$ where

$$N' = N \cup \{0\}$$

$$A' = A \cup \{ (0,1), (0,2), \ldots (0,n)\}$$

Let $$X_{0i} = \begin{cases} 1 & \text{if node } i \text{ is the beginning of a path} \\ 0 & \text{otherwise} \end{cases}$$


node 0
1
2
3
4
5
6
7
8
9




The optimal solution

The optimal solution to the new version of the problem is a spanning tree!

## The Optimization Problem:

**Minimize** $\sum_{j=1}^{n} X_{0j}$

**subject to**

$X \ \varepsilon \ \mathcal{T}$ = **set of all spanning trees of G'**

$\sum_{j=1}^{n} X_{ij} \leq 1$ **for each** $i \varepsilon N$

$\sum_{i=0}^{n} X_{ij} = 1$ **for each** $j \varepsilon N$

$X_{ij} \ \varepsilon \ \{0,1\}$ **for each** $(i,j) \ \varepsilon \ A'$

*Note that no inequality limits out-degree of node 0*

**Minimize** $\sum_{j=1}^{n} X_{0j}$

**subject to**

$X \in \mathcal{T}$ = **set of all spanning trees of G'**

$\sum_{j=1}^{n} X_{ij} \leq 1$ **for each** $i \in N$

$\sum_{i=0}^{n} X_{ij} = 1$ **for each** $j \in N$

*These are essentially constraints of an assignment problem!*

$X_{ij} \in \{0,1\}$ **for each** $(i,j) \in A'$

This problem appears to be a good candidate for Lagrangian Relaxation because of its structure:

- If we relax the spanning tree constraint, we obtain a relaxation which is an assignment problem
- If we relax the assignment constraints, we obtain a relaxation which is a minimum spanning tree problem

*However, because the spanning tree constraint is not easily written as a system of explicit linear constraints, relaxing them is problematic!*

## Variable "splitting"

For each variable $X_{ij}$ of the problem, define a variable $Y_{ij}$

Require that X be a spanning tree,
that Y be a feasible assignment,
and that $X_{ij} = Y_{ij}$ for each i & j

$$\text{Minimize} \quad \alpha \sum_{j=1}^{n} X_{0j} + (1-\alpha) \sum_{j=1}^{n} Y_{0j}$$

**subject to**

$$X \in \mathcal{T}$$

$$\sum_{j=1}^{n} Y_{ij} \leq 1 \quad \text{for each } i \in N$$

$$\sum_{i=0}^{n} Y_{ij} = 1 \quad \text{for each } j \in N$$

$$Y_{ij} \in \{0,1\} \quad \text{for each } (i,j) \in A'$$

$$X_{ij} = Y_{ij} \quad \text{for each } (i,j) \in A'$$

for some specified weight $\alpha$ which distributes the cost between the two sets of variables $(0 \leq \alpha \leq 1)$

$$\text{Minimize} \quad \alpha \sum_{j=1}^{n} X_{0j} + (1-\alpha) \sum_{j=1}^{n} Y_{0j}$$

subject to

$X \;\varepsilon\; \mathcal{T}$

$$\sum_{j=1}^{n} Y_{ij} \le 1 \quad \text{for each } i \varepsilon N$$

$$\sum_{i=0}^{n} Y_{ij} = 1 \quad \text{for each } j \varepsilon N$$

$$Y_{ij} \;\varepsilon\; \{0,1\} \quad \text{for each } (i,j) \,\varepsilon\, A$$

$$X_{ij} = Y_{ij} \quad \text{for each } (i,j) \,\varepsilon\, A'$$

*We now relax these constraints!*

# The Lagrangian Relaxation:

$$\text{Minimize } \alpha \sum_{j=1}^{n} X_{0j} + (1-\alpha) \sum_{j=1}^{n} Y_{0j} + \sum_{i=0}^{n} \sum_{j=1}^{n} \lambda_{ij} (X_{ij} - Y_{ij})$$

subject to

$$X \in \mathcal{T}$$

$$\sum_{j=1}^{n} Y_{ij} \le 1 \quad \text{for each } i \in N$$

$$\sum_{i=0}^{n} Y_{ij} = 1 \quad \text{for each } j \in N$$

$$Y_{ij} \in \{0,1\} \quad \text{for each } (i,j) \in A'$$

# The Lagrangian Relaxation:

$$\text{Minimize } \sum_{j=1}^{n} (\alpha + \lambda_{0j}) X_{0j} + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{ij} X_{ij}$$

$$+ \sum_{j=1}^{n} (1 - \alpha - \lambda_{0j}) Y_{0j} - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{ij} Y_{ij}$$

**subject to**

$$X \in \mathcal{T}$$

$$\sum_{j=1}^{n} Y_{ij} \le 1 \quad \text{for each } i \in N$$

$$\sum_{i=0}^{n} Y_{ij} = 1 \quad \text{for each } j \in N$$

$$Y_{ij} \in \{0,1\} \quad \text{for each } (i,j) \in A'$$

The Lagrangian Relaxation separates into two subproblems:

**Minimum Spanning Tree Problem:**

$$\Phi_X(\lambda) = \textbf{minimum} \quad \sum_{j=1}^{n} (\alpha + \lambda_{0j}) X_{0j} + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{ij} X_{ij}$$

**subject to**

$$X \in \mathcal{T}$$

# Assignment Problem

$$\Phi_Y(\lambda) = \text{minimum} \sum_{j=1}^{n} (1-\alpha-\lambda_{0j}) Y_{0j} - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{ij} Y_{ij}$$

**subject to**

$$\sum_{j=1}^{n} Y_{ij} \leq 1 \quad \text{for each } i \varepsilon N$$

$$\sum_{i=0}^{n} Y_{ij} = 1 \quad \text{for each } j \varepsilon N$$

$$Y_{ij} \ \varepsilon \ \{0,1\} \quad \text{for each } (i,j) \ \varepsilon \ A'$$

For any matrix $\lambda$ of Lagrangian multipliers, the sum of the optimal values of the two subproblems provides a lower bound on the optimal value of the original problem:

$$\Phi(\lambda) = \Phi_X(\lambda) + \Phi_Y(\lambda) \leq Z^*$$

**The Lagrangian Dual:**

$$\Phi^* = \text{Maximum } \Phi(\lambda)$$

The search for the optimal dual variables ( $\lambda$ ) can be performed by ***subgradient optimization***

The subgradient of the dual objective, $\Phi(\lambda)$ is the matrix $\Delta = \{ \delta_{ij} \}$ where $\delta_{ij} = (X_{ij} - Y_{ij})$

This is the direction in which to change $\lambda$

$$\lambda_{new} = \lambda_{old} + \tau \frac{\left(Z^{*} - \Phi(\lambda_{old})\right)}{\|\Delta\|^2} \Delta$$

$\tau \; \varepsilon \; (0,2]$ *is a stepsize parameter*

It may be that the optimal values of X and Y for the subproblems are never feasible paths.

For this reason, it is worthwhile to seek a feasible solution (which provides an upper bound) by means of a heuristic.

Two heuristic algorithms have been designed:

- a "greedy" algorithm
- a random-search algorithm

# The "greedy" algorithm proceeds as follows:

Initially, the path set P is empty ($P \leftarrow \varnothing$)

(a) If all nodes lie on a path, stop. Else, begin a new path by selecting the node $i^*$ which minimizes $\lambda_{0i}$.
Let $P \leftarrow P \cup \{(0,i^*)\}$

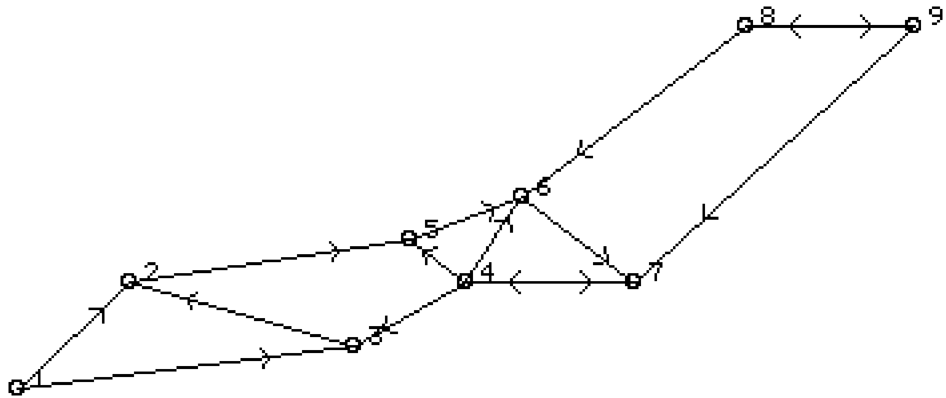
(b) If $\{(i,j): j$ does not lie on a path$\}$ is empty, go to step (a).
Otherwise, let $j^* \leftarrow \text{argmin}\{\lambda_{ij} : j \text{ does not lie on a path}\}$

(c) Let $P \leftarrow P \cup \{(i^*,j^*)\}$ and $i^* \leftarrow j^*$.
Return to step (b).

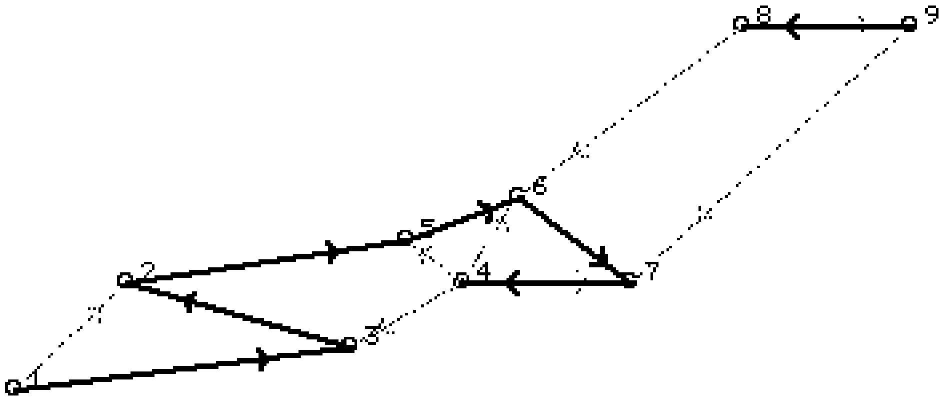
**The random search algorithm finds several trial solutions, each constructed as in the greedy algorithm except:**

In step (b), the choice of the next node to add to the path is random, with probability depending upon the current value of the Lagrange multipliers ($\lambda_{ij}$). *(Probabilities vary inversely as the multipliers, so that the choice tends to be "greedy".)*

# Randomly-generated problem (N=9)

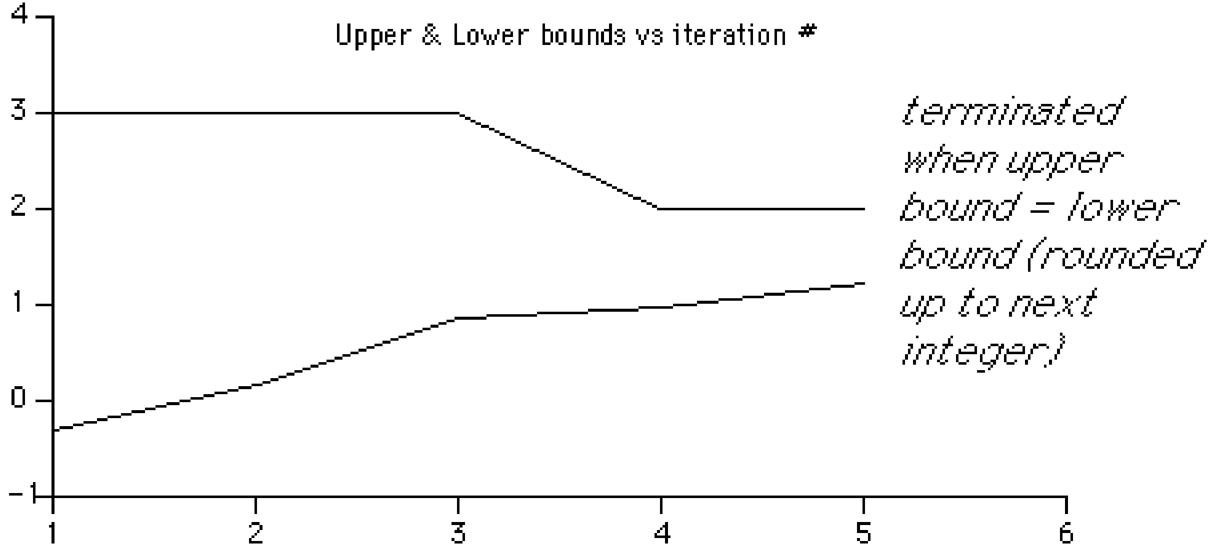


# The optimal solution:



# Results of Lagrangian dual search

(Spanning tree & assignment subproblems)



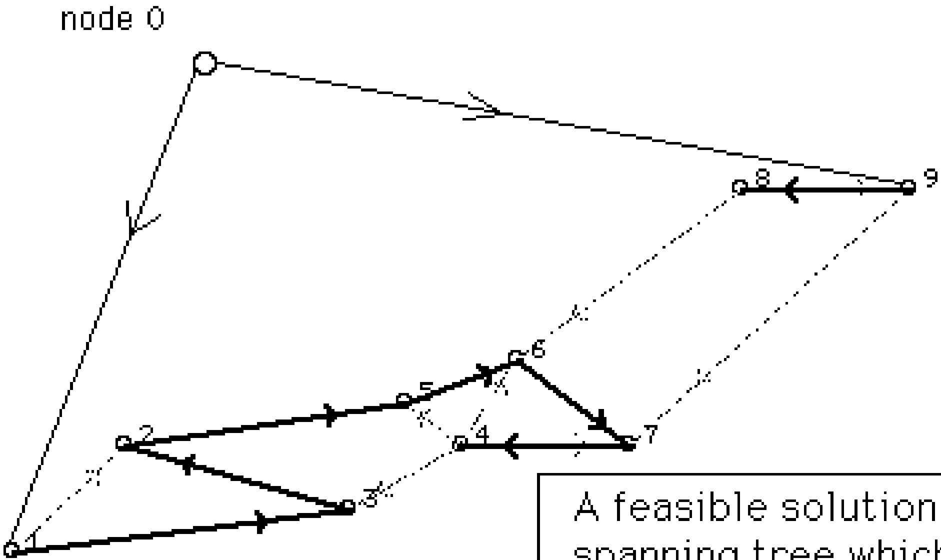
## *Other relaxations are possible:*

#2

Relax, in addition to those relaxed in the approach just presented, the constraint on the in-degree of each node:

$$\sum_{i=0}^{n} Y_{ij} = 1 \quad \textbf{for each } j \in N$$

The subproblem in Y is then a simple GUB (generalized upper bound, or "multiple choice") problem.




A feasible solution is a directed spanning tree which is known as an "arborescence": the number of arcs entering any node is at most 1

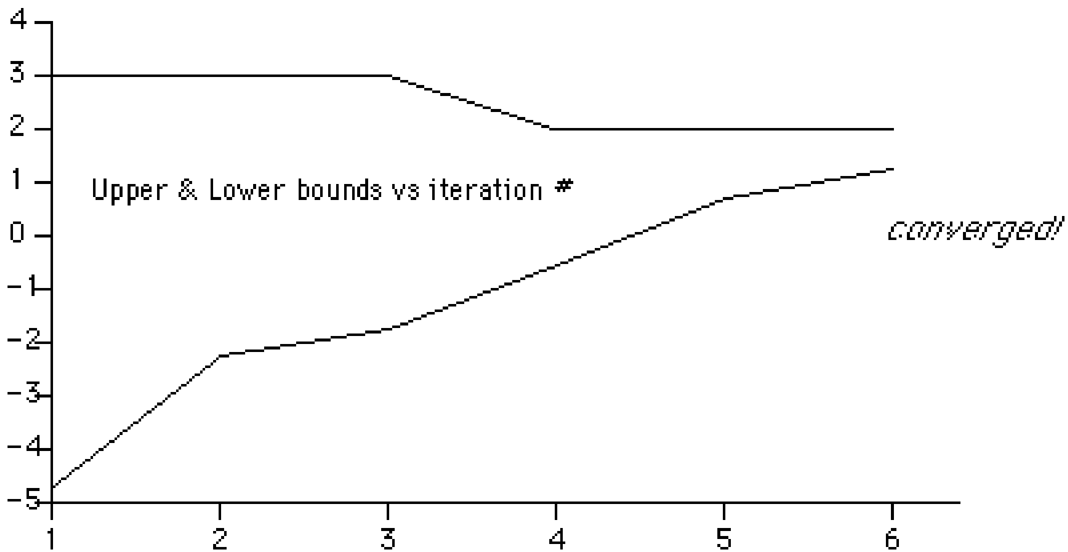
#3

Replace the constraint that X is a tree with the stronger constraint that X is an "arborescence" (a directed tree with in-degrees of the nodes ≤ 1.) Then relax as in #2.

*(The algorithm to compute a minimum spanning arborescence is $O(n^4)$. In practice, execution time for the APL code is about 15 times that for the spanning tree problem, for a 20-node problem.)*

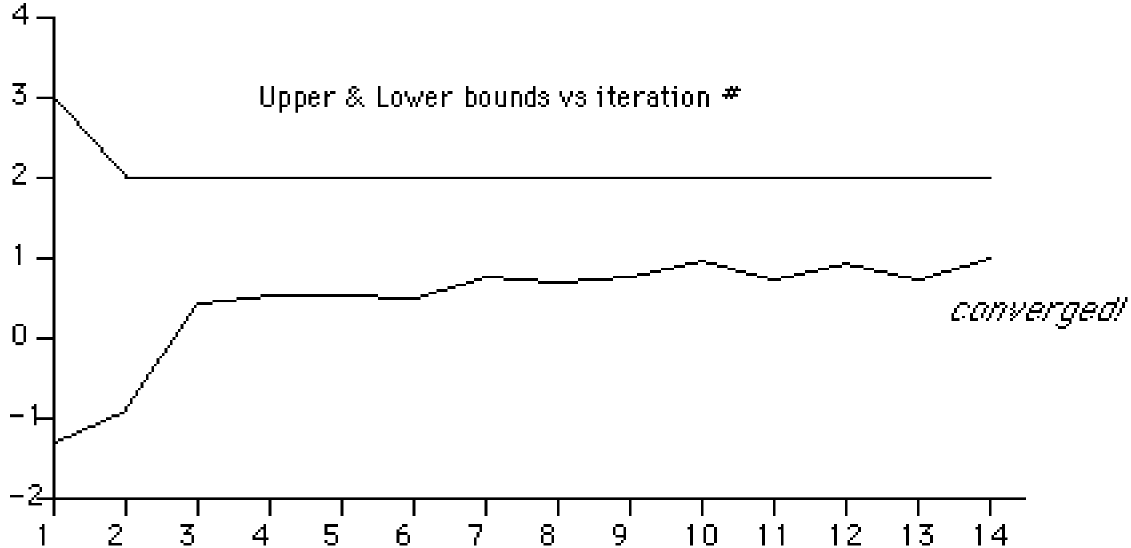
# Using relaxation #2 (spanning tree & GUB problems)

*(Using greedy heuristic)*

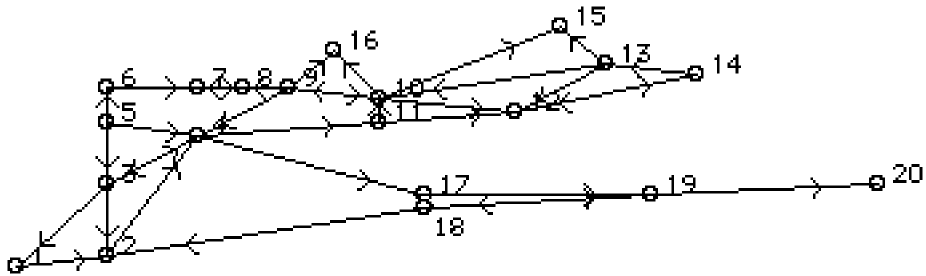


# Using relaxation #3 (spanning arborescence & GUB)

*(Using greedy heuristic)*



# Another randomly-generated problem, with N=20

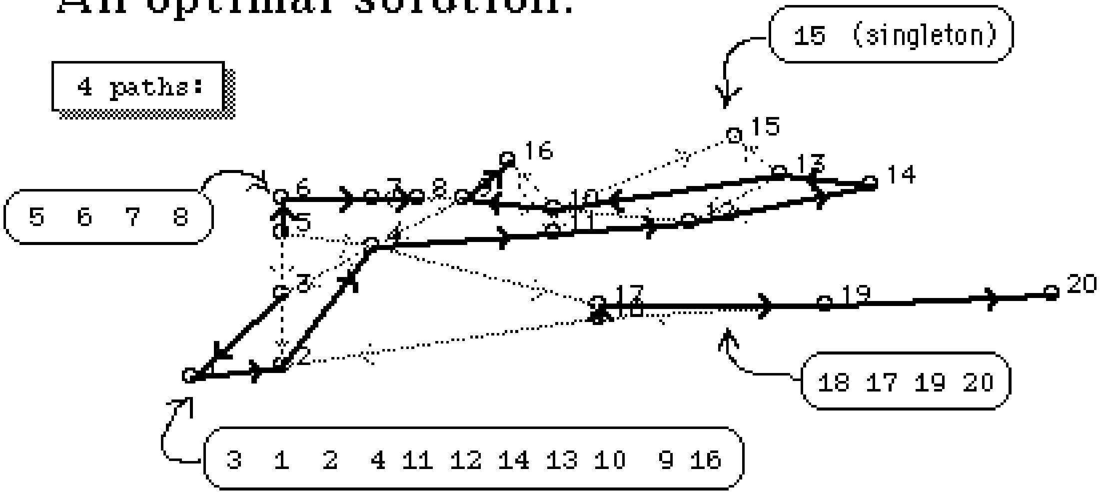


# The Adjacency Matrix:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 18 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# An optimal solution:

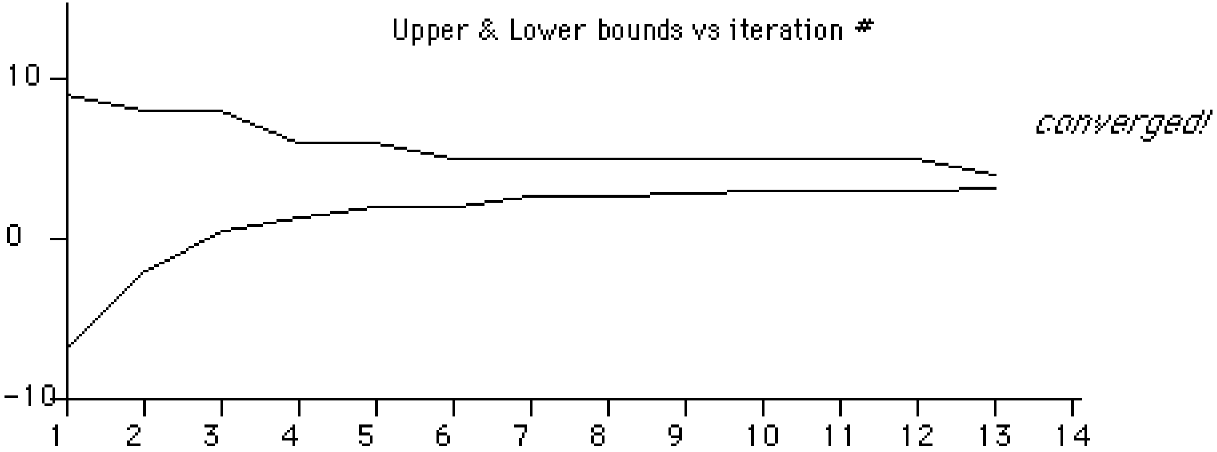
4 paths:




*(The "dummy" node 0 & arcs from it are not shown.)*

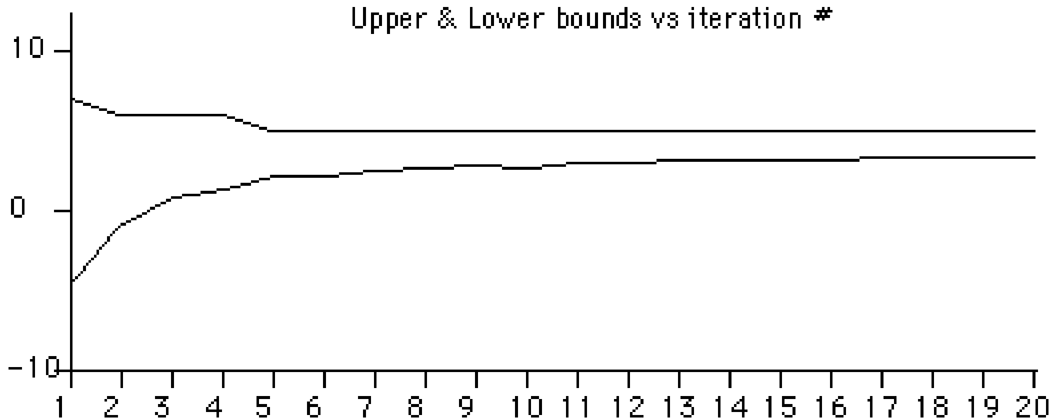
# Relaxation #2 (spanning tree & GUB subproblems)

*(Using random search heuristic with 5 trials)*



# Relaxation #3 (spanning arborescence & GUB subproblems)

*(Using random search heuristic with 5 trials)*




*terminated, with upper bound =5, lower bound ≐3.4*

This limited computational experience suggests that the additional effort required to find the minimum spanning arborescence is not effective.