Plowing with Precedence

A Variant of the Windy Postman Problem

April 22, 2012 – POMS 2012

Benjamin Dussault, Bruce Golden, Chris Gröer, and Edward Wasil

Overview

- Background
  - The Chinese Postman Problem and the Windy Postman Problem
  - The Levitating Plow Problem
- Literature Review
- Introduction
- Problem Statement
- Problem Formulation
- Solution Methodology
- Results
- Conclusions

Background

Chinese Postman Problem (CPP)

- Consider a graph $G=\{V,A\}$ where
  - $V=\{v_i\}$
  - $A=\{(v_i,v_j) \mid v_i, v_j \in V, i<j\}$
  - c_{ij} = Cost of traversing arc (v_i,v_j)
  - $c_{ij} = c_{ji}$
- Goal: Construct a least-cost cycle that visits all arcs in *A* at least once

Background

Windy Postman Problem (WPP)

- A variant of the Chinese Postman Problem
- The graph is Windy, i.e., it is harder to traverse in one direction on an arc as opposed to the other
- Goal: Construct a least-cost cycle that visits all arcs in *A* at least once
- Key Difference: Costs are not symmetric

Background

Levitating Plow Problem (LPP)

- Motivates Plowing with Precedence and is used in our solution methodology
- A variant of the Windy Postman Problem that incorporates four costs:
  - The cost of plowing uphill and downhill
  - The cost of deadheading uphill and downhill
- The plow can deadhead at any time
  - When considering a street that is not plowed, the plow has the option to deadhead the street
  - Requires levitation over the snow (coming soon to a plow near you)

Background

Methodology for the CPP, WPP and LPP

- Key observation: If a graph is Eulerian, then an optimal cycle can be produced by Fleury's Algorithm
- Therefore, it is sufficient to convert the instance graph to an Eulerian graph in an optimal way
- Possible methods
  - Integer programming
  - Add least-cost paths between odd-degree nodes

Background

LPP - IP Formulation

- Adapt IP formulation from the Windy Postman Problem
- Essential variables:
  - x_{ij} = the number of times (i,j) is plowed
  - y_{ij} = the number of times (i,j) is deadheaded
- Essential constraints:
  - Plow each street twice
  - Degree matching for each node
- While the LPP is NP-hard, the IP is easily solved by commercial solvers

Literature Review

- Arc Routing is well studied. There are many survey articles:
  - Assad and Golden (1995)
  - Eiselt et al. (1995a, 1995b)
  - Dror (2000)
- Perrier et al. (2006, 2007) provide a four-part survey of winter road maintenance covering:
  - System Design
  - Models and Algorithms
  - Vehicle Routing and Depot Location
  - Vehicle Routing and Fleet Sizing

Introduction

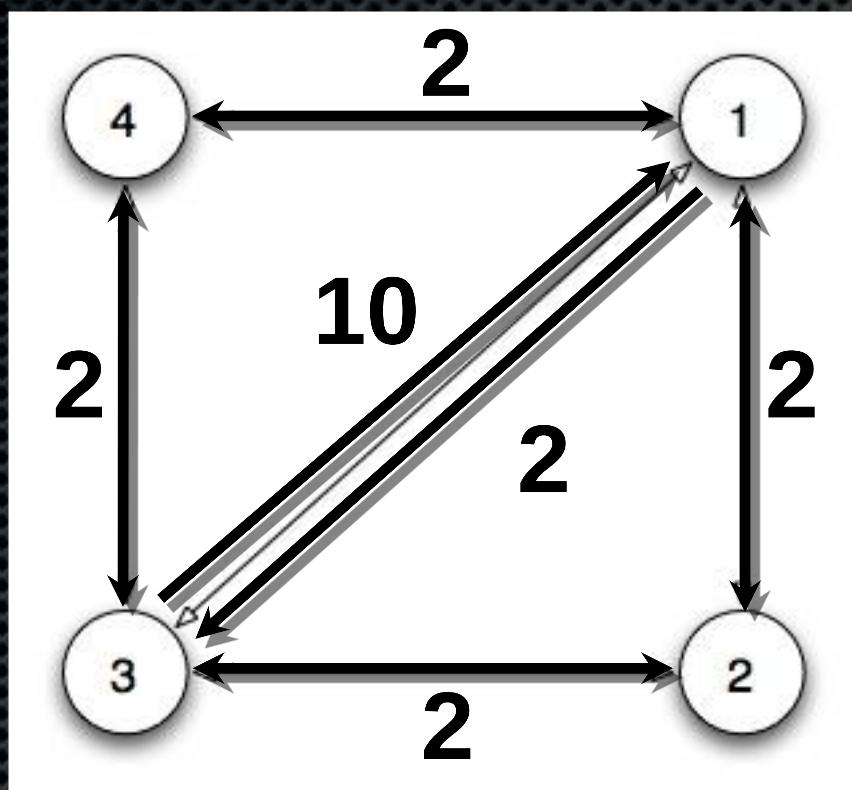
- Variant of the Levitating Plow Problem
  - Levitating plows are not real
  - If a plow encounters an unplowed street, it must plow it
- Therefore, the option of deadhead traversal is only available *after a street is plowed*
- Introduces the concept of precedence: the potential choices and associated costs of traversing a street depends on the preceding tour

Introduction

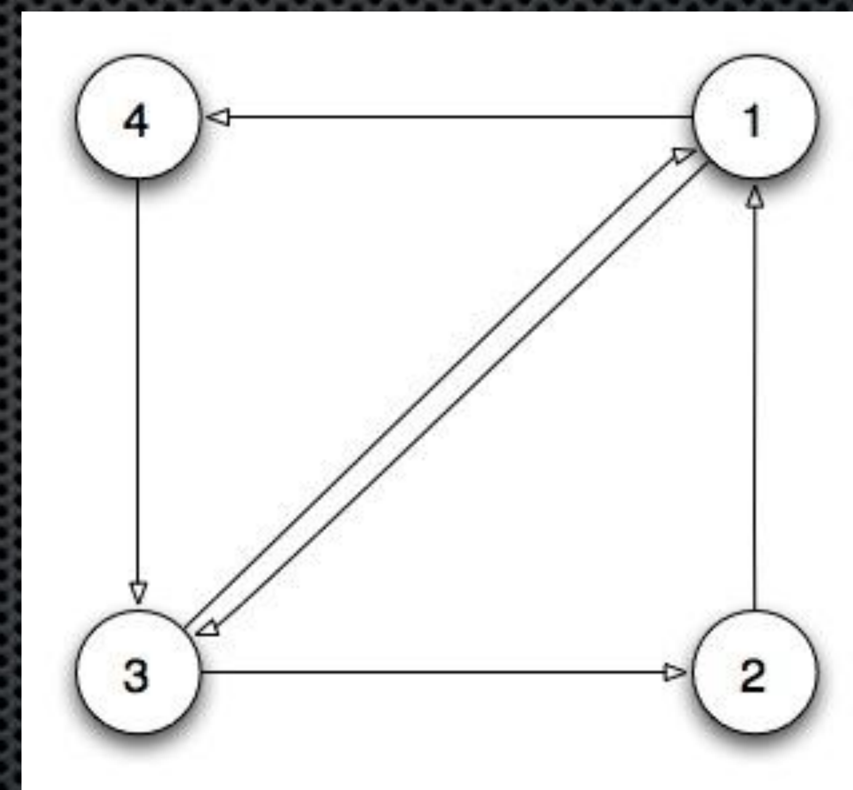
- The concept of precedence requires a fundamentally different solution methodology than those used in WPP literature
- An Eulerian graph yields many Eulerian cycles
  - Equivalent in WPP
  - Not equivalent in Plowing with Precedence

Introduction

Deadhead costs = 1



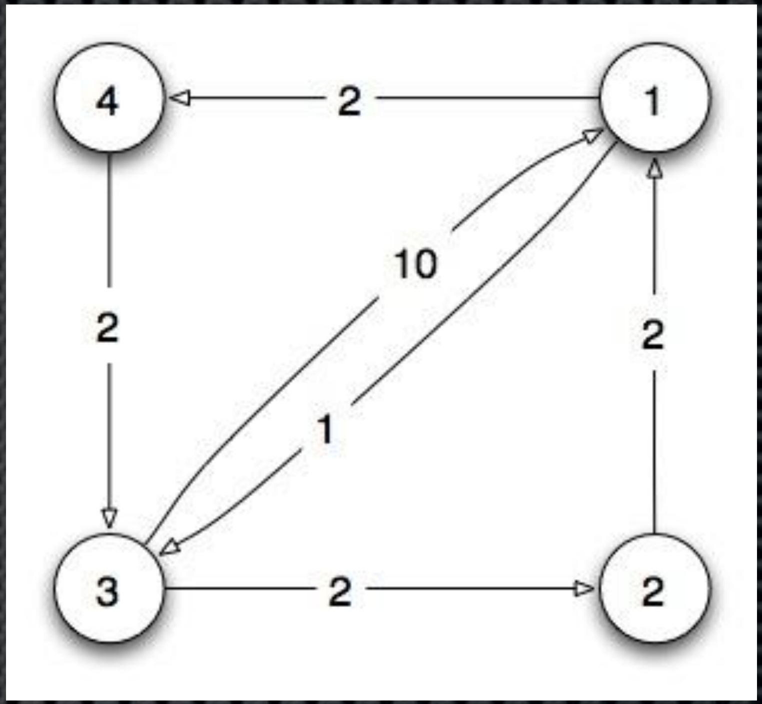
Original Instance

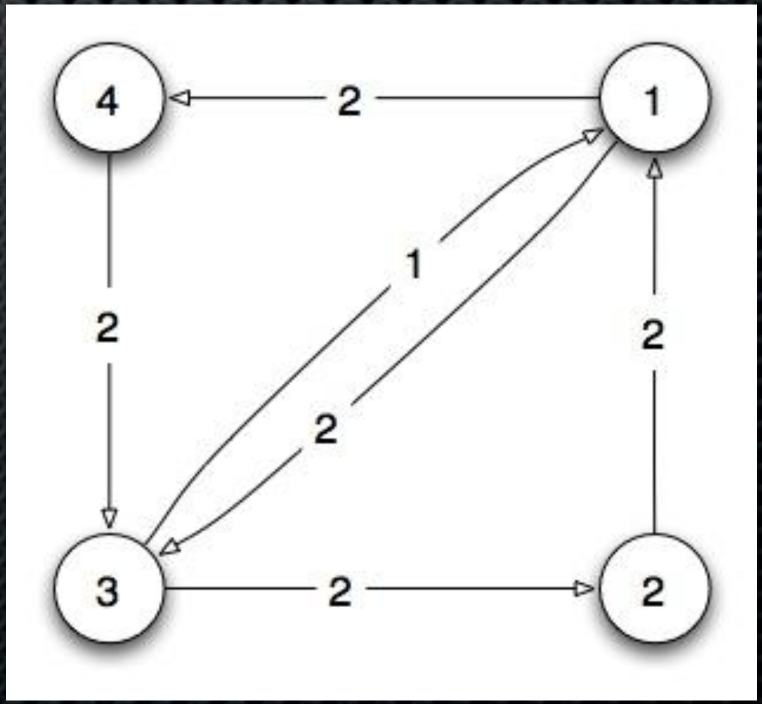


Induced Eulerian Graph

Introduction

- Many Eulerian cycles:
  - {1,4,3,1,3,2,1}
    - Plow arc (3,1) before (1,3)
    - Cost = 19
  - {1,3,2,1,4,3,1}
    - Plow arc (1,3) before (3,1)
    - Cost = 11





Problem Statement

- Consider a graph $G=\{V,A\}$ where
  - $V=\{v_i\}$
  - $A=\{(v_i,v_j) \mid v_i, v_j \in V\}$
  - c_{ij}^+ = Cost of plowing arc (v_i,v_j)
  - c_{ij}^- = Cost of deadheading arc (v_i,v_j)
  - $c_{ij}^+ >> c_{ji}^+ >> c_{ij}^- \geq c_{ji}^-$
- Goal: To construct a least-cost cycle that visits all streets in *A* at least twice (once for each side of the street) and begins and ends at a depot (required to incorporate precedence)
  - Plowing each street once (as in the previous example) is easily handled
  - Plowing each street an arbitrary number of times is easily handled

Problem Statement

- Undirected arcs allow plowing against the flow of traffic
  - Practically, streets are closed for plowing
- Good solutions will attempt to plow downhill on both sides of the street
- Allows for the possibility of:
  - Plowing downhill
  - Then deadheading uphill
  - Then plowing downhill

Problem Formulation

- Requires an index *t* to incorporate precedence
- Essential elements:
  - x_{ijt} = 1 if plow (*i,j*) at time t, 0 otherwise
  - y_{ijt} =1 if deadhead (*i,j*) at time t, 0 otherwise
  - φ_{ijt} =1 if (*i,j*) is first plowed at time t, 0 otherwise
- Essential constraints:
  - Eulerian cycle continuity (arc entering node *i* at time *t* requires arc leaving node *i* at time *t+1*)
  - Forbid deadhead on (*i,j*) until (*i,j*) or (*j,i*) is plowed
- Large number of variables and constraints (~8000 and 19000 respectively, for an instance with 10 arcs and 7 nodes)

Solution Methodology

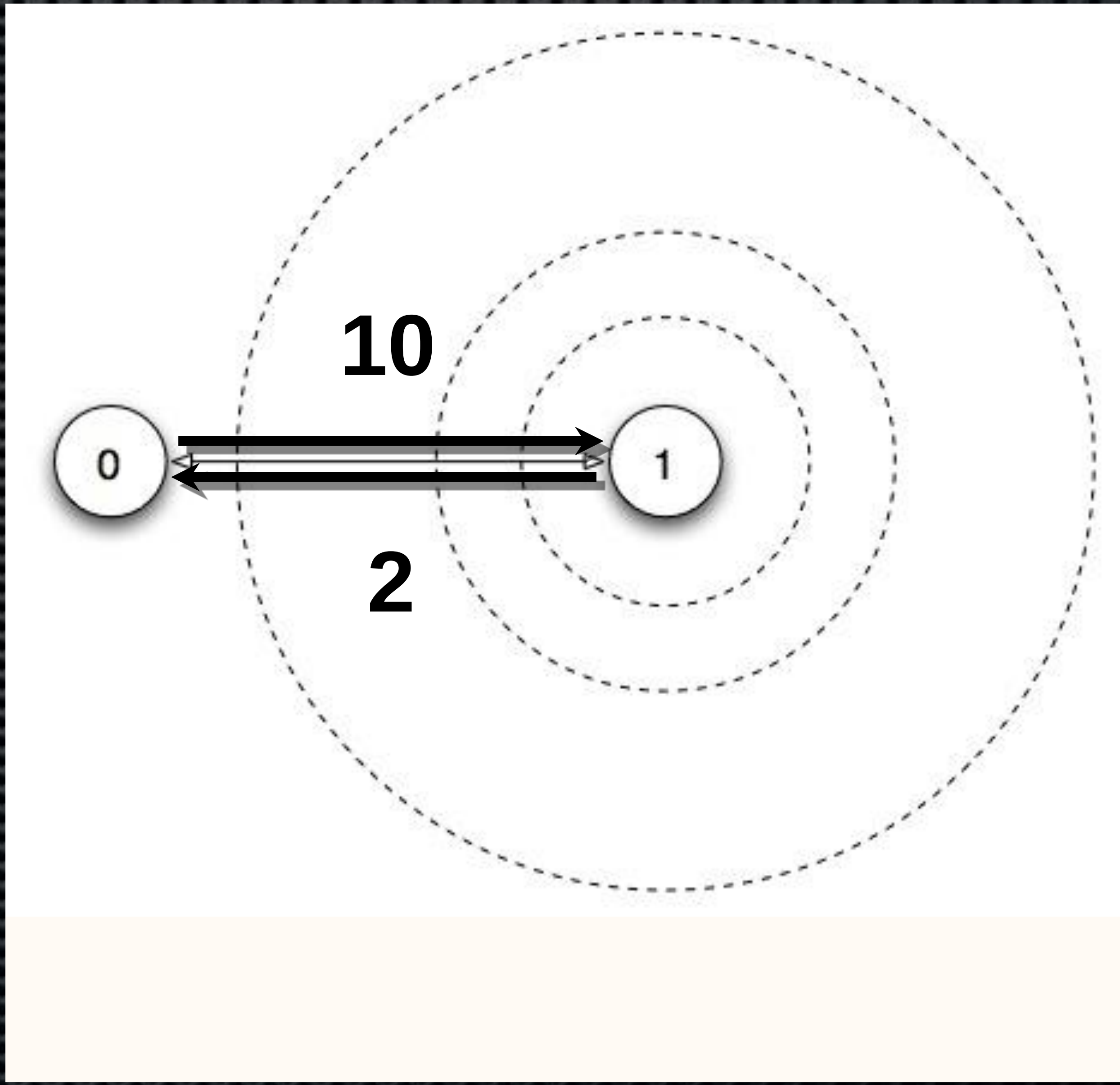
Overview

- Construct a “solution framework” using the solution to Levitating Postman Problem
  - Solution to IP gives a number of traversals for each arc
  - Solution serves as a lower bound
- Use solution framework to construct initial solution using Fleury’s Algorithm
- Perform local search on a solution
  - Reinitialize and repeat local search
- Prune a solution to obtain the final solution

Solution Methodology

Solution Framework

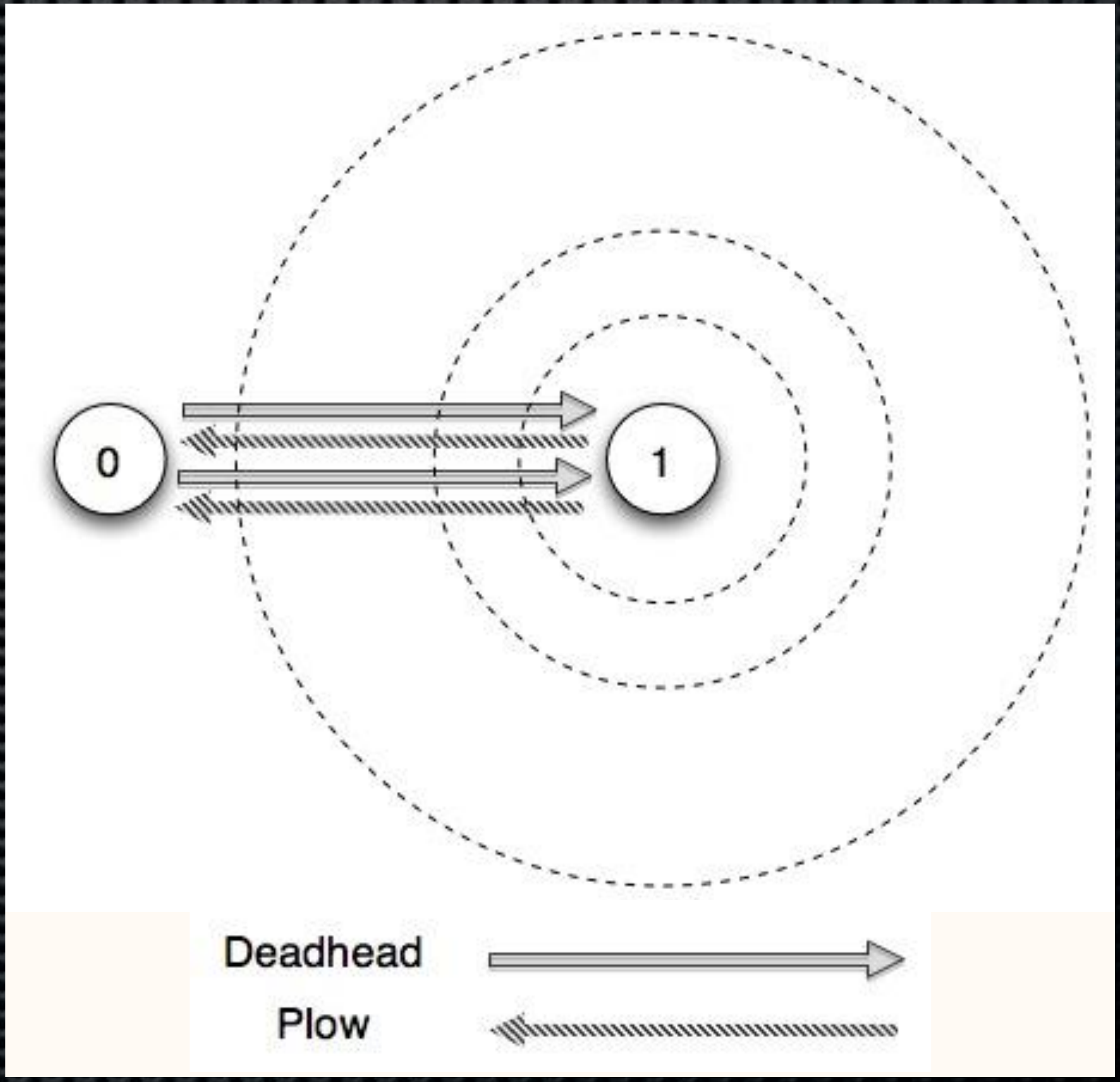
- Circles on graph indicate elevation
- It is possible that no cycle will yield the objective function of the solution framework
- Let the cost of (0,1) be 10 and the cost of (1,0) be 2
- Let the deadhead cost be 1



Solution Methodology

Solution Framework

- Solution framework seeks to plow downhill twice
- Plowing uphill is unavoidable, hence the solution framework forbiding it is infeasible
- Solution framework has objective value of 6
- Optimal cycle (0,1,0) has cost 12



Solution Framework

Solution Methodology

Initial Solution

- A cycle can be produced by the solution framework using Fleury's Algorithm
- This cycle is guaranteed to traverse (and hence plow) each street twice
- Not guaranteed to have a cost that is the same as the lower bound of the solution framework (previous example)
- Seek to improve a cycle using a local search heuristic

Solution Methodology

Local Search

- We explore the set of all Eulerian cycles that obey the solution framework
- Search nearby cycles to find a better one
- Requires:
  - Definition of neighborhood - define nearby
  - Fitness function - gives the quality of a cycle
    - In our case, the fitness is the cost of the cycle

Solution Methodology
Local Search

- Solution Fitness:

For each arc, decide to plow based on the following:

if arc has been plowed twice
→ then don't plow
else if arc hasn't been plowed at all
→ then plow
else if going downhill
→ then plow
else if cycle isn't going downhill later
→ then plow
else don't plow

Solution Methodology

Local Search

- All Eulerian cycles can be decomposed into sub-cycles
- Definition of neighborhood around a solution *s*, *N(s)*: the set of all cycles that can be obtained by a combination of the following moves
  - Sub-cycles in the cycle are permuted
  - Sub-cycles in the cycle are reversed

Plowing with Precedence

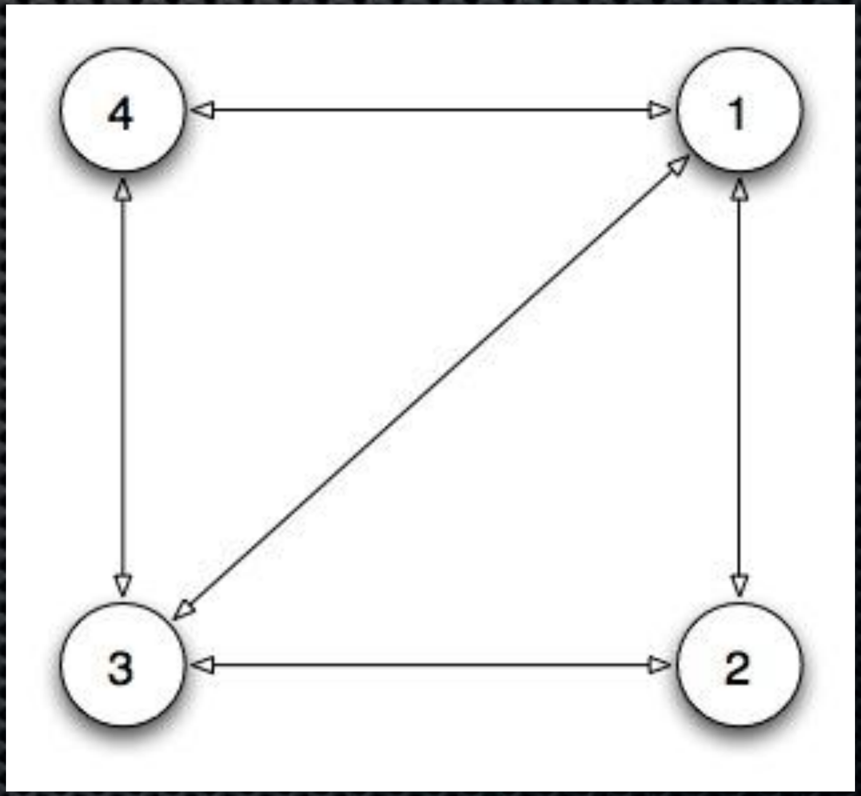
Solution Methodology - Local Search

{1,2,3,1,2,3,4,1,3,4,1}

{1,2,3,4,1,3,4,1,2,3,1}

↕

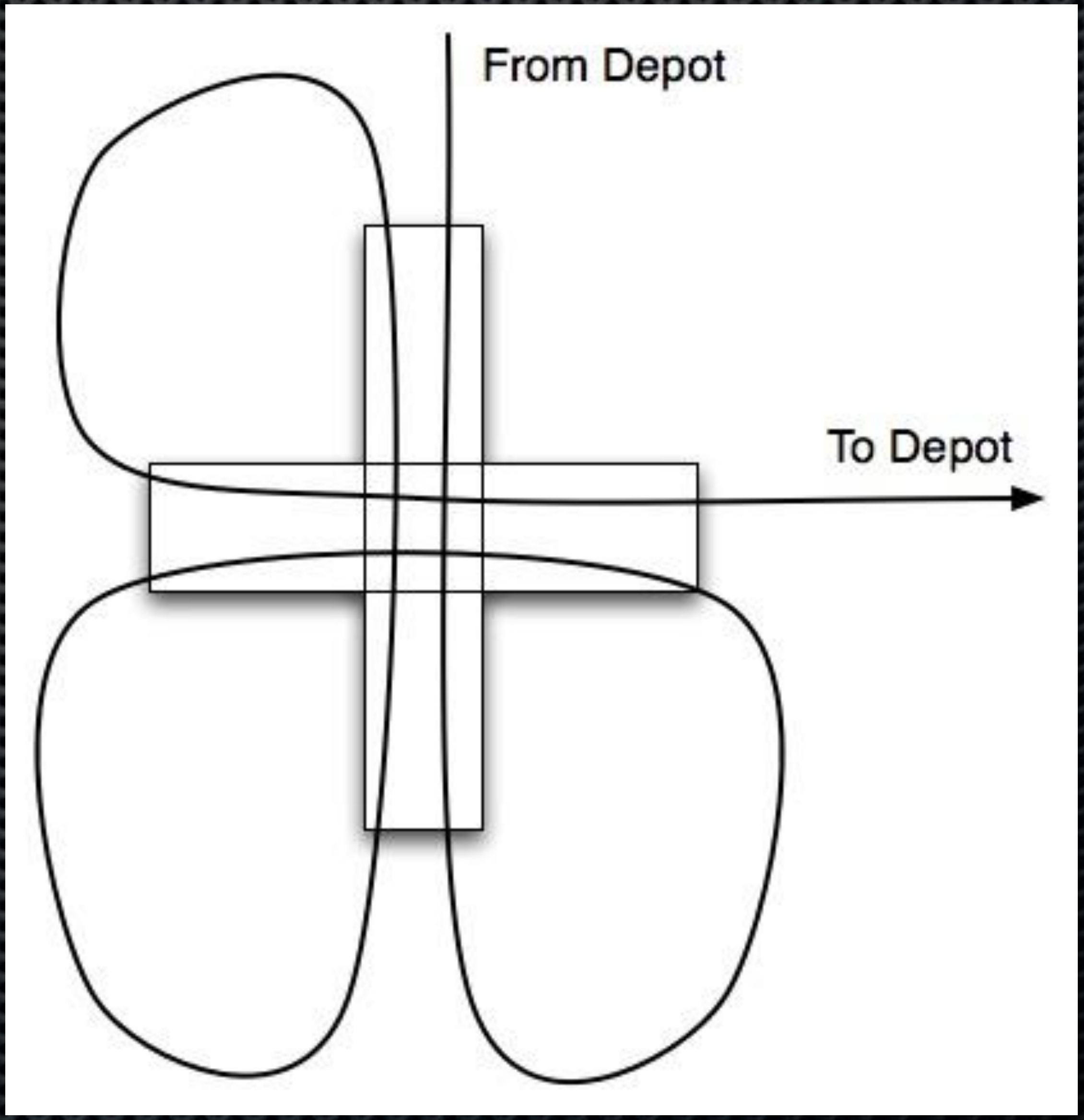
{1,2,3,4,1,3,4,1,3,2,1}



Solution Methodology

Local Search

- The number of permutations is large: *n!* for *n* cycles
- To limit the size of the neighborhood, if *n>4*, we limit the set of permutations to *4!+n* for linear growth
- Most intersections have four or fewer cycles



Solution Methodology

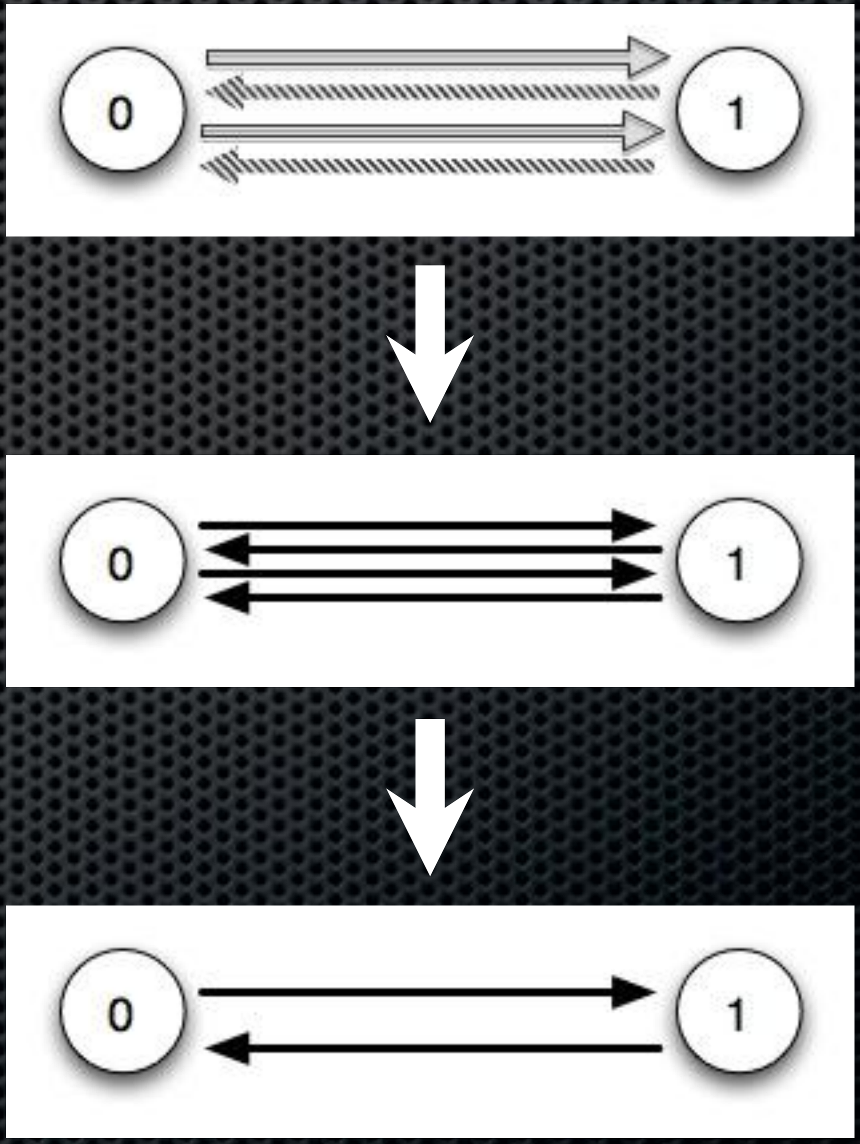
Reinitialization

- Local search is deterministic and depends on the initial solution
- We reinitialize to produce new initial solutions for local search
- This is done by permuting cycles around different nodes randomly a large number of times
- The best solution produced in 15 runs of local search and reinitialization is retained

Solution Methodology

Pruning

- It is possible that a cycle will have sub-cycles that have only deadhead moves
- These cycles can be pruned to obtain a lower-cost cycle that still plows each street twice
- Pruning is done at the end of local search plus reinitialization phase



Solution Methodology

Lower Bounds

- Linear Program (LP) relaxation
  - Difficult to solve in a reasonable amount of time
  - Removed some constraints to speed up the LP
  - Obtained bounds are very tight
- LPP in solution framework
  - Does not incorporate precedence at all
  - Outperforms the LP relaxation

Computational Results

- We test our algorithm on 45 modified Windy Rural Postman Problems given in Corberan et al. (2007)
  - Remove Rural concept
  - Existing costs are interpreted as plowing costs
  - Randomly generate deadhead costs
- Instances are characterized by:
  - Number of nodes (7 to 196)
  - Number of arcs (10 to 316)
  - Average cost deviation - average discrepancy in cost between plowing up and plowing down (4% to 80%)

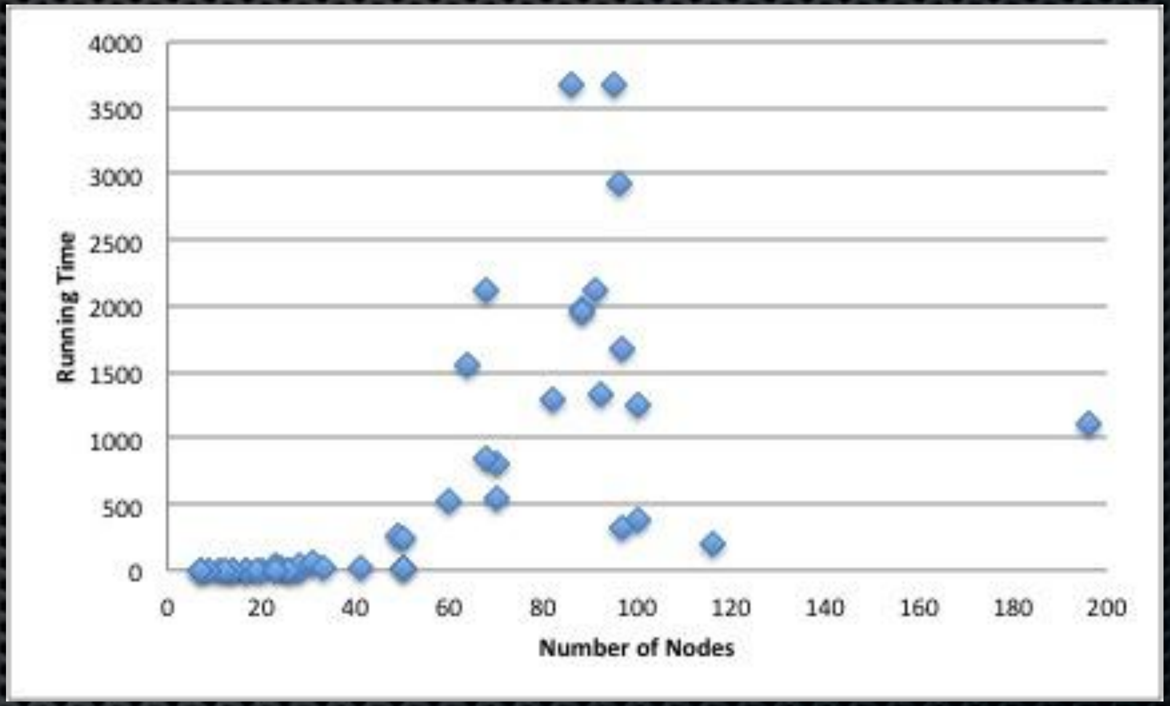
Computational Results

- Our IP formulation for Plowing with Precedence is large, so we only solve the smallest of instances (up to 9 nodes) to optimality with Gurobi
- We compare the solution produced by our heuristic to the lower bound given by the solution framework
  - If the heuristic solution matches lower bound, then we know we have the optimal solution
- Our heuristic performs very well
  - Produces the optimal solution to 24 of 45 instances
  - Average deviation of 0.17% from the lower bound

Computational Results

Running Time

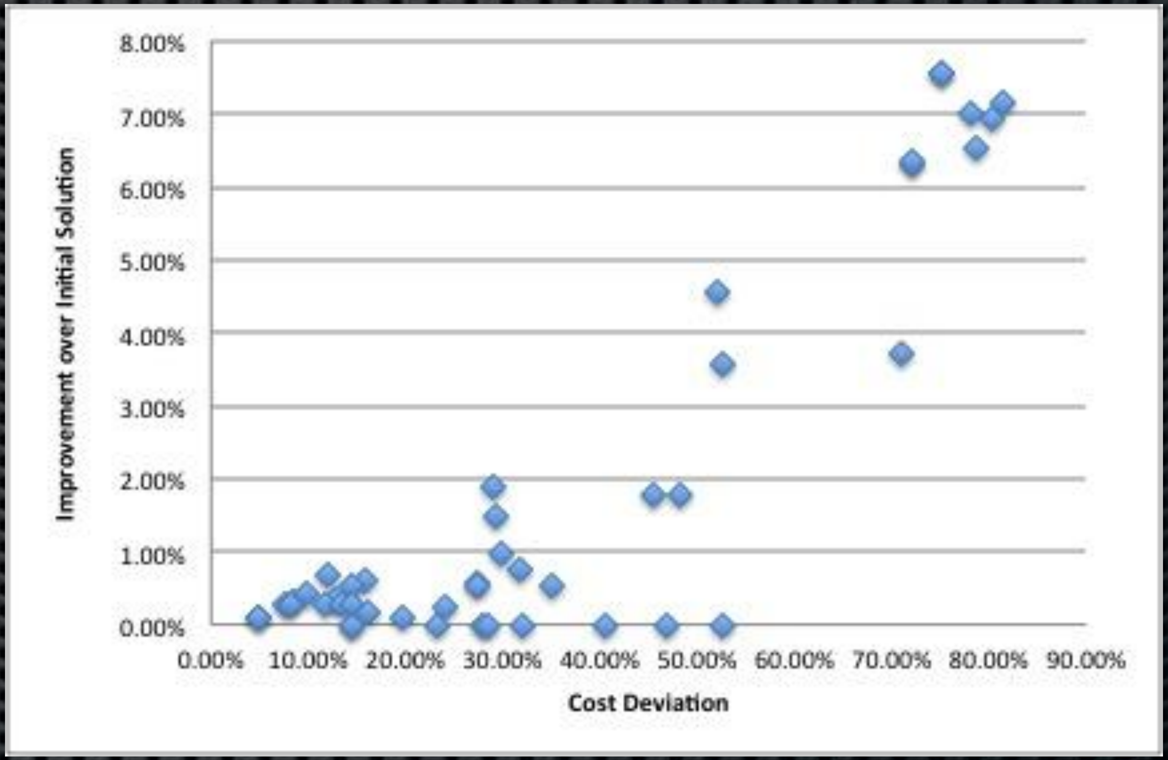
- All tests were performed on a single thread of a 1.86 GHz Intel Core2Duo processor
- Min = 0.156 seconds
- Max = 3686 seconds
- Average = 687 seconds



Computational Results

Improvement over Initial Solution

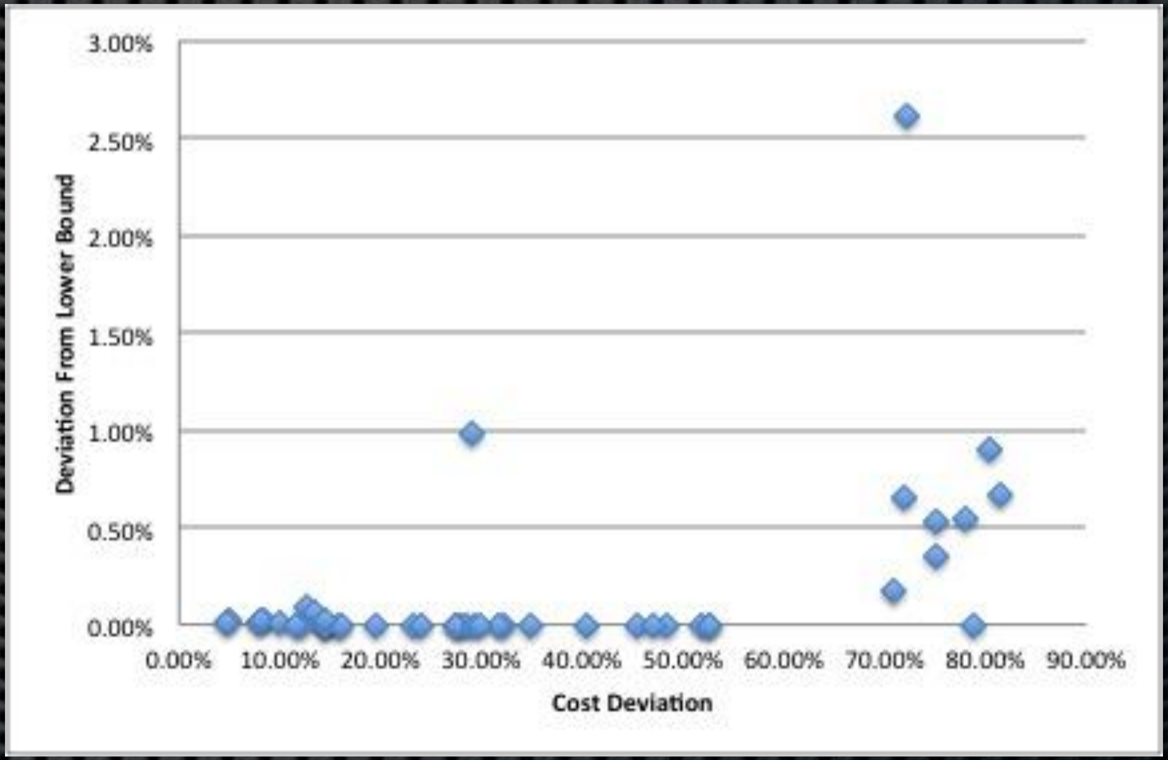
- Compare final solution cost against the initial solution cost
- 1.8% average improvement
- Measure percentage improvement vs. Average cost deviation



Computational Results

Deviation from Lower Bound

- Cost deviation is largest driving factor in deviation from lower bound
- 0.17% average deviation from the lower bound
- Deviation from the lower bound increases as cost deviation increases
- Want to investigate further

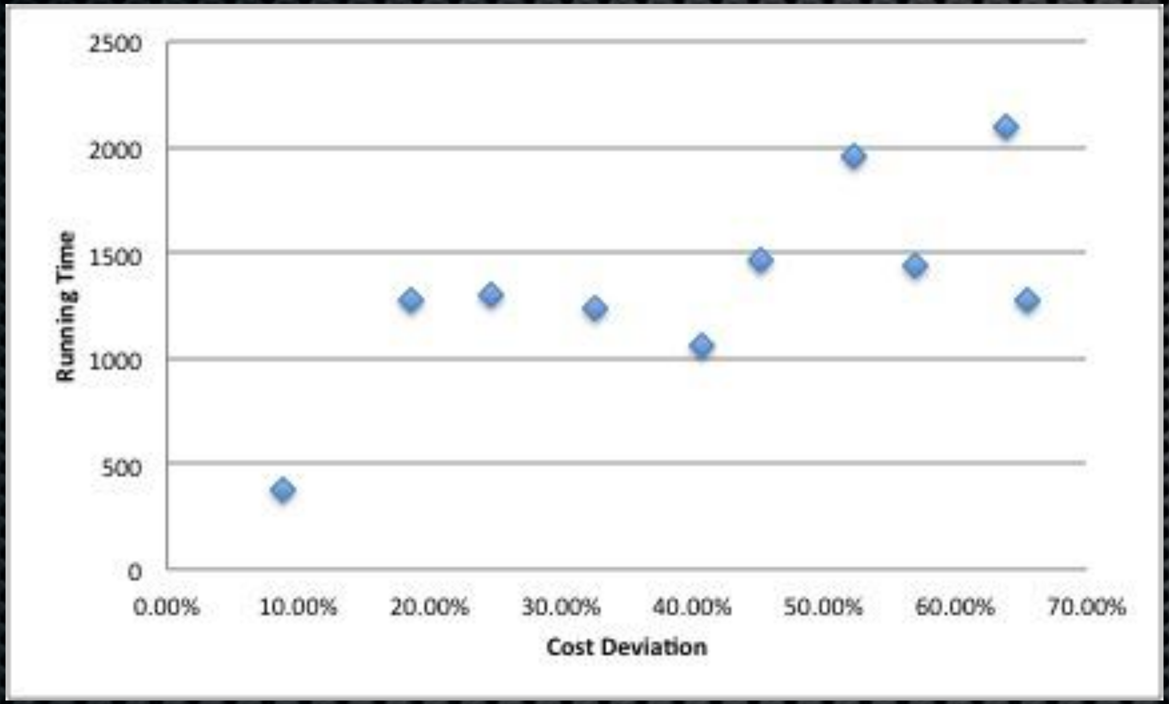


Computational Results

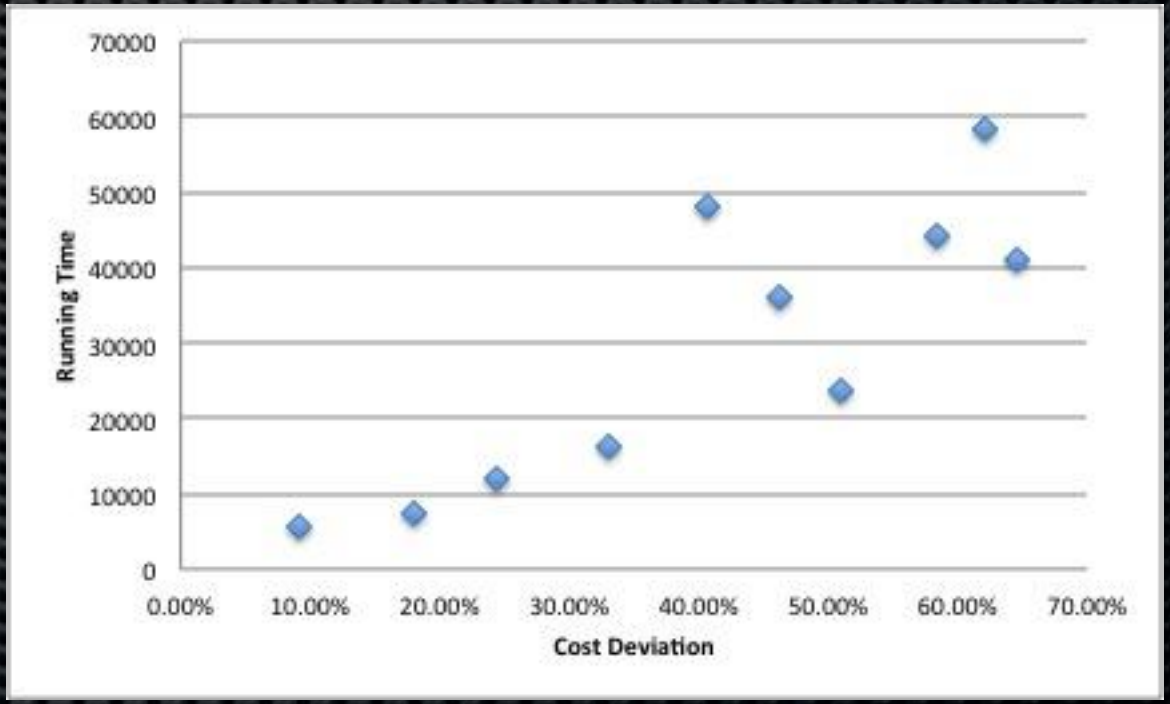
- We selected two large instances (116 and 196 nodes) and constructed several new instances that:
  - Preserved the same graph
  - Average cost deviation ranged from 10% to 70%
- Compare the effects of average cost deviation on:
  - Running Time
  - Percentage Improvement
  - Deviation from Lower Bound

Computational Results

Running Time vs. Average Cost Deviation



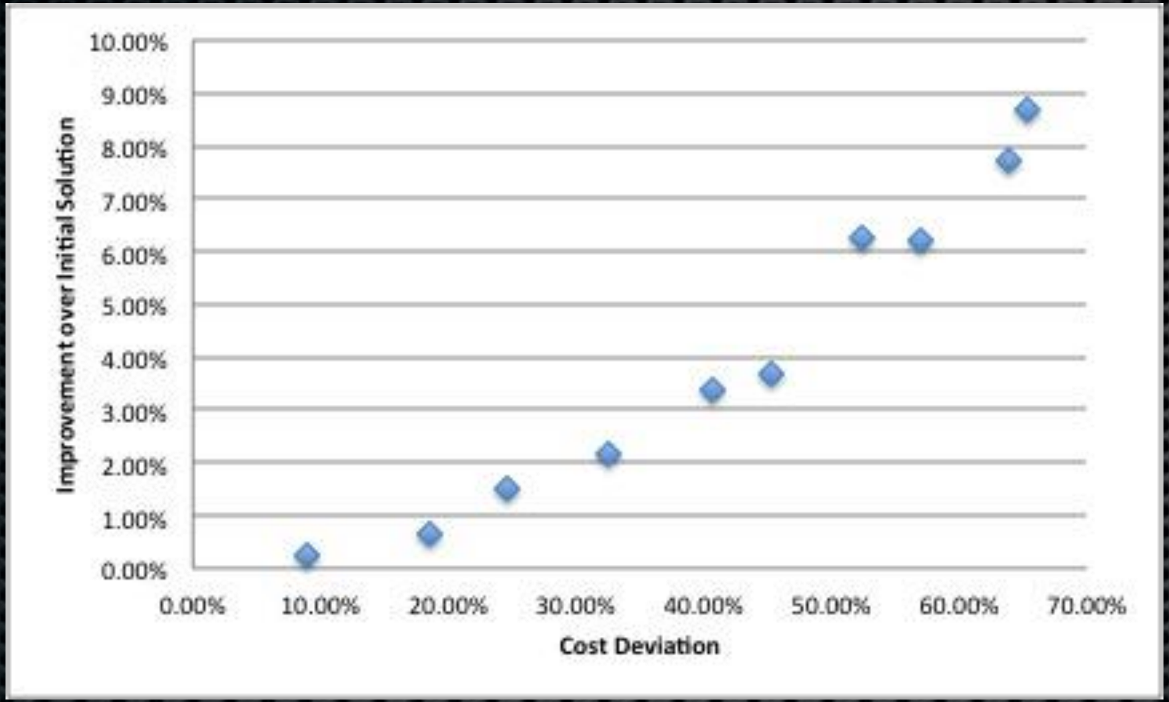
Instance A3101



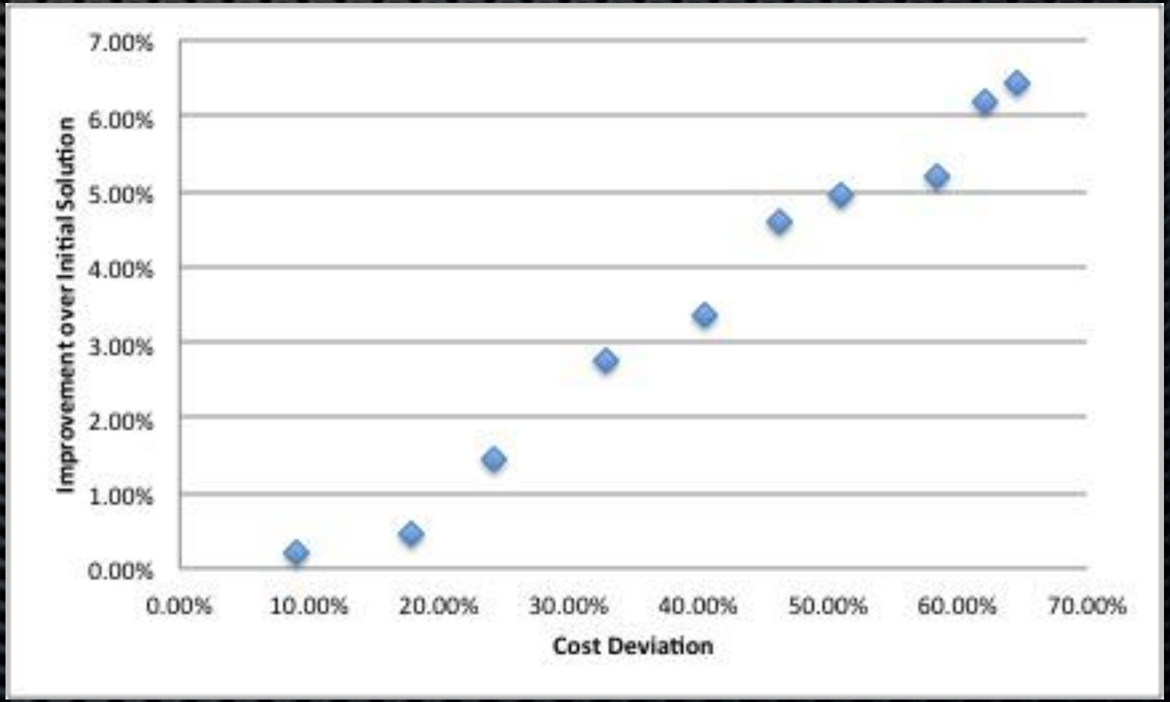
Instance M3101

Computational Results

Percentage Improvement vs. Average Cost Deviation



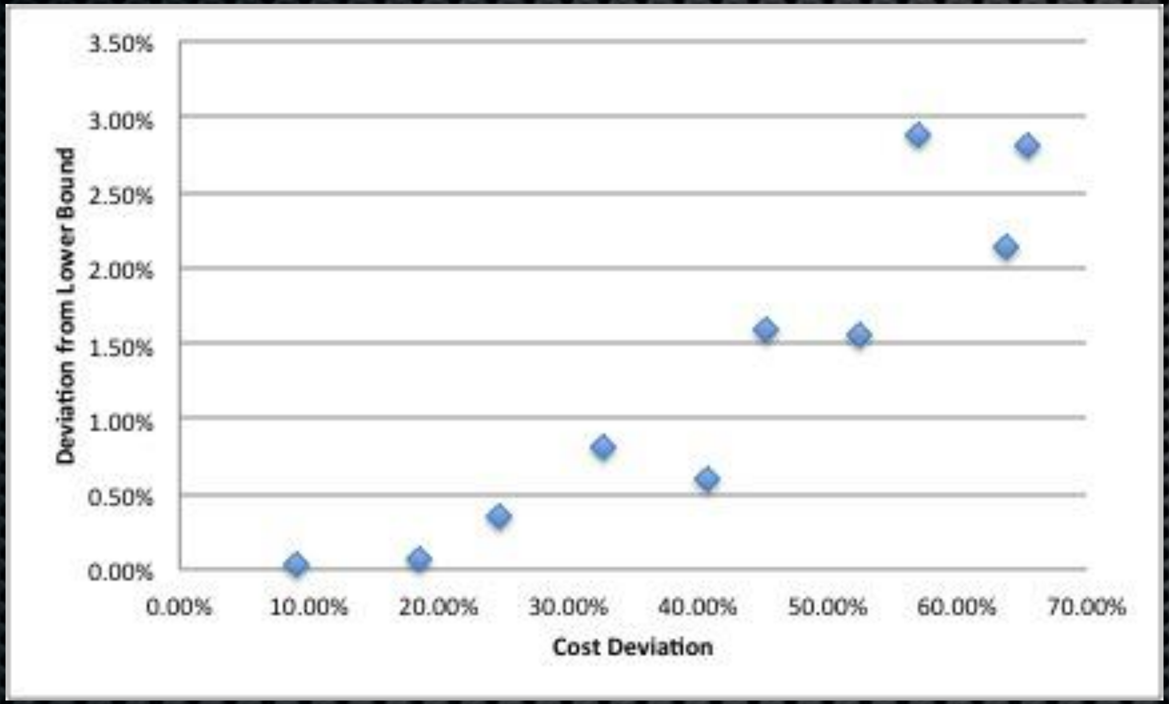
Instance A3101



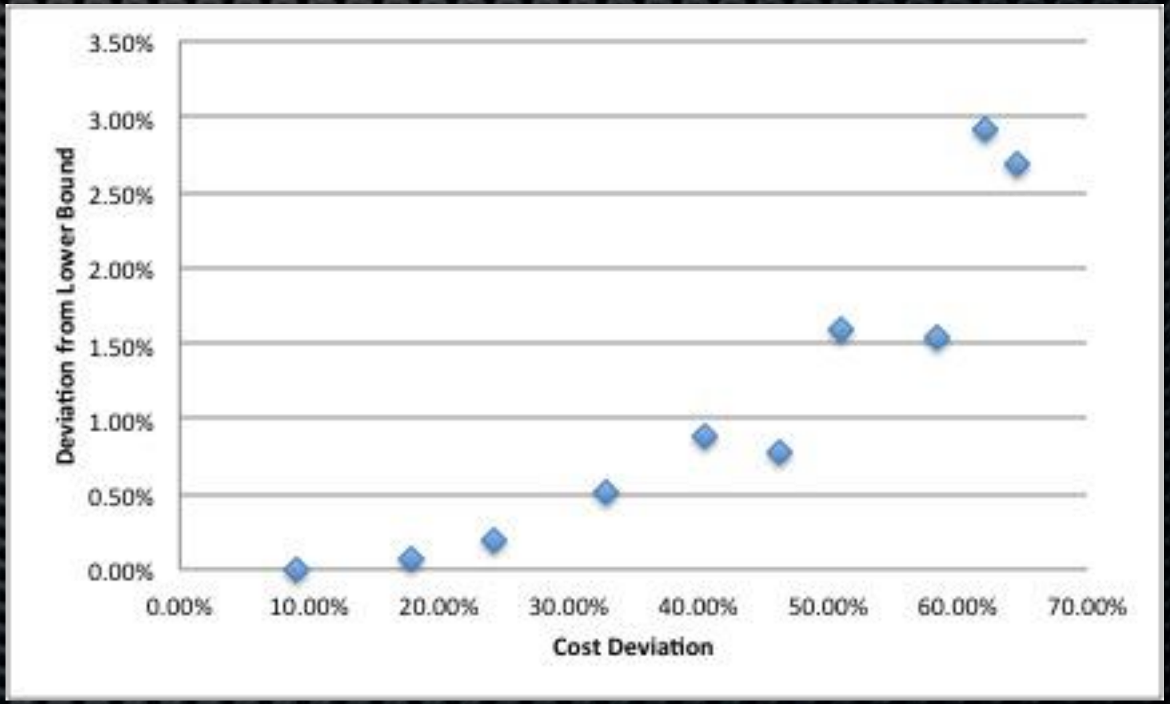
Instance M3101

Computational Results

Deviation from Lower Bound vs. Average Cost Deviation



Instance A3101



Instance M3101

Conclusions

- Introduced the Plowing with Precedence variant of the WPP
- Addressed the practical consideration that the choice of deadheading a street is only available after plowing
- Introduced the concept of precedence to postman problems
- Our heuristic generated very good results, with solutions that are, on average, within 0.17% of the lower bound for instances derived from those in the literature, and 0.49% for all instances
  - Many solutions are optimal
- Observed increases in running time, percentage improvement, and deviation from the lower bound as the average cost deviation increased

Conclusions

- Future work
  - Improve lower bound for large problems
  - Improve upper bound
  - Generalize the concept of precedence: Let the possible choices and costs of traversal be a more general function of the number of times traversed
  - Add multiple plows: When one snow plow clears a street, other plows are able to deadhead that street