Plowing with Multiple Plows

A Variant of the Windy Postman Problem

May 22, 2012 – ODYSSEUS 2012

Benjamin Dussault, Bruce Golden, and Edward Wasil

Overview

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

Background

Chinese Postman Problem (CPP)

- Consider a graph $G=\{V,A\}$ where
  - $V=\{v_i\}$ is the set of vertices
  - $A=\{(v_i,v_j) \mid v_i, v_j \in V, i<j\}$ is the set of arcs
  - 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 direction
- ❖ Goal: Construct a least-cost cycle that visits all arcs in *A* at least once
- ❖ Key Difference: Costs are not symmetric

Background

Downhill Plow Problem (DPP)

- A variant of the Windy Postman Problem that incorporates four costs:
  - The costs of plowing uphill and downhill
  - The costs 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 or to plow the street

Background

Methodology for the CPP, WPP, and DPP

- 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

DPP - IP Formulation

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

Literature Review

- Arc routing is well studied. There are many survey articles including
  - 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

Literature Review

- Min-Max k-CPP
  - Frederickson et al (1978)
  - Ahr and Reinelt (2002)
  - Arh (2004)
- Min-Max k-WRPP
  - Benavent (2009, 2010, 2011, 2012)
- Don’t incorporate four costs like the DPP

Introduction to MDPP

- Generalization of the Downhill Plow Problem
- Min-Max Multiple Plow Downhill Plowing Problem (Min-Max *k*-DPP)
  - *k* Plows
  - Objective: Minimize the maximum route length
- Key attributes:
  - Minimizing cumulative route length and route balancing are competing objectives
  - If multiple plows traverse a street, there is a decision regarding which plows clear the street

Problem Statement

- Consider a graph $G=\{V,A\}$ where
  - $V=\{v_i\}$ is the set of vertices
  - $A=\{(v_i,v_j) \mid v_i, v_j \in V\}$ is the set of arcs
  - 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 *k* cycles, each beginning and ending at a depot, that collectively visit all streets in *A* at least twice (once for each side of the street) so that the length of the longest cycle is minimized
  - Plowing each street an arbitrary number of times (as opposed to twice) 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
    - If multiple plows traverse a street, there is a decision regarding which plows clear the street
    - Ad hoc assignment of clearing duties might make the longest route longer

Solution Methodology

Overview

- Construct a "solution framework" using the solution to the Downhill Postman Problem
  - Solution to the IP gives a number of traversals for each arc
  - Solution serves as a lower bound
- Use solution framework to construct an initial grand cycle using Fleury's Algorithm
- Split grand cycle into *k* routes
- Perform local search on a grand cycle (not split grand cycle) to improve solution
  - Reinitialize and repeat local search

Solution Methodology

Initial Grand Cycle

- A grand cycle can be produced by the solution framework using Fleury's Algorithm
- This grand cycle is guaranteed to traverse (and hence plow) each street twice
- DPP formulation is modified to guarantee that a grand cycle can be split into *k* feasible routes
  - Require that the number of times a grand cycle leaves the depot is greater than *k*

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

{0,1,2,3,2,0,3,0,1,2,0}

Plow 1:

Plow 2:

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

Plow 1:

Plow 2:

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

{0,1,2,3,2,0} {0,3,0} {0,1,2,0}

Plow 1:

Plow 2:

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

{0,3,0}   {0,1,2,0}

Plow 1: {0,1,2,3,2,0}

Plow 2:

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

{0,1,2,0}

Plow 1: {0,1,2,3,2,0}

Plow 2: {0,3,0}

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

Plow 1: {0,1,2,3,2,0}

Plow 2: {0,3,0} {0,1,2,0}

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

Plow 1: {0,1,2,3,2,0}

Plow 2:

Solution Methodology

Grand Cycle Splitting

- A grand cycle is guaranteed to return to the depot *k* times
- Consider following example with 2 plows and unit costs:

Plow 1:{0,1,2,3,2,0}

Plow 2: {0,3,0,1,2,0}

Solution Methodology

Local Search

- We explore the set of all grand cycles that obey the solution framework
- Search nearby grand cycles to find a better one
- Need to
  - Define nearby
  - Specify a fitness function - gives the quality of a grand cycle

Solution Methodology

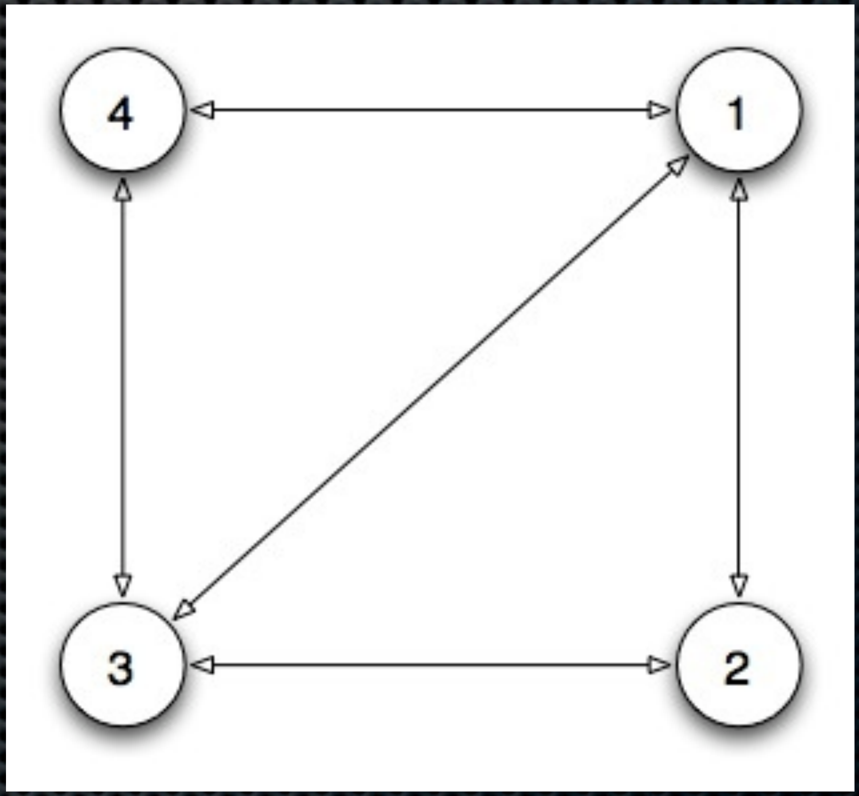
Local Search - Neighborhood

- Similar to the procedure presented in Plowing with Precedence (Dussault et al. ROUTE 2011)
- All grand cycles (which are Eulerian cycles) can be decomposed into sub-cycles
- Definition of neighborhood around a grand cycle *s*, *N(s)*, is the set of all grand cycles that can be obtained by permuting sub-cycles of the grand cycle

Solution Methodology

Local Search - Neighborhood

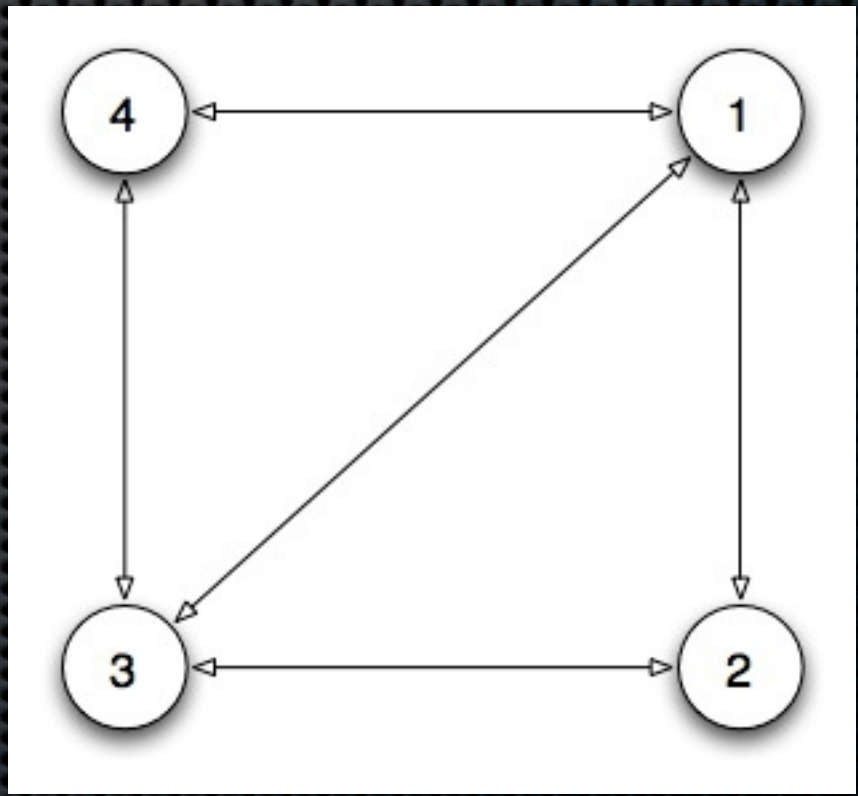
{1,2,3,1,2,3,4,1,3,4,1}



Solution Methodology

Local Search - Neighborhood

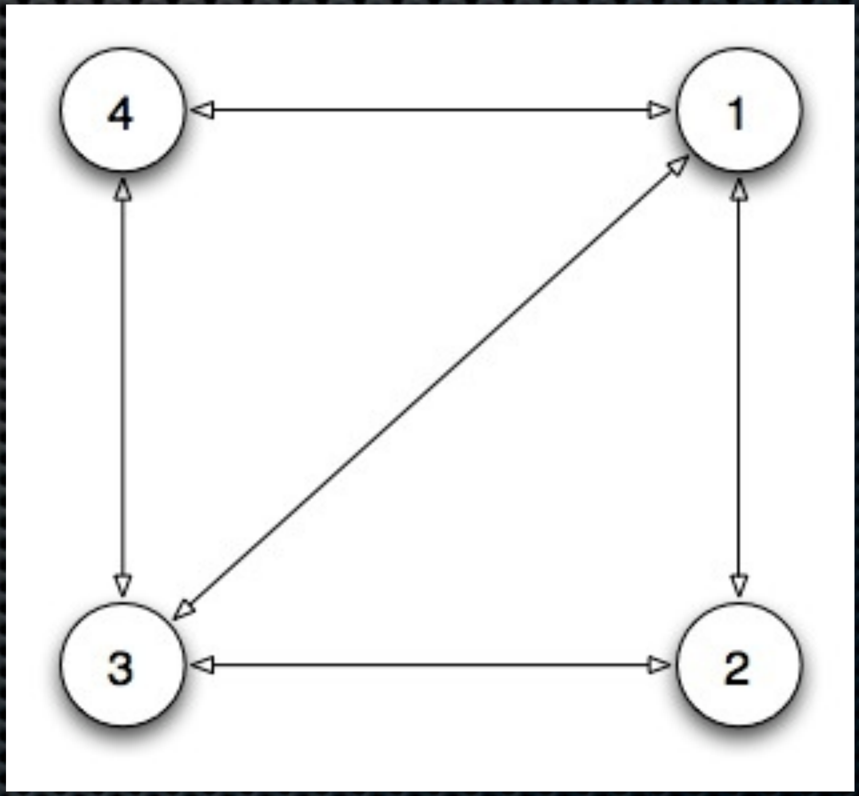
{1,2,3,1,2,3,4,1,3,4,1}



Solution Methodology

Local Search - Neighborhood

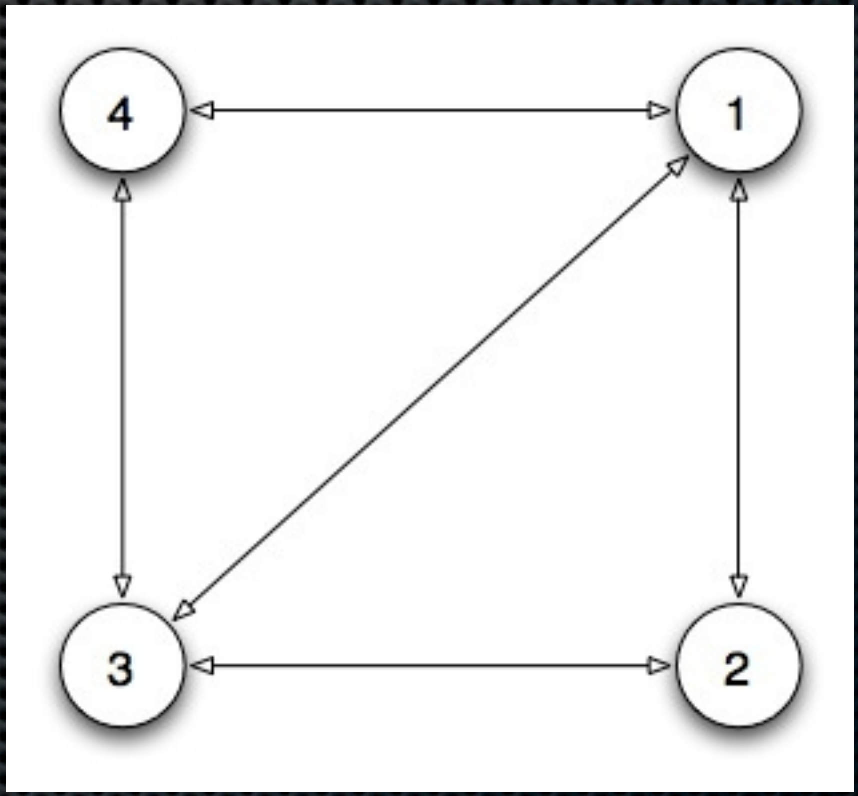
{1,2,3,1,2,3,4,1,3,4,1}



Solution Methodology

Local Search - Neighborhood

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

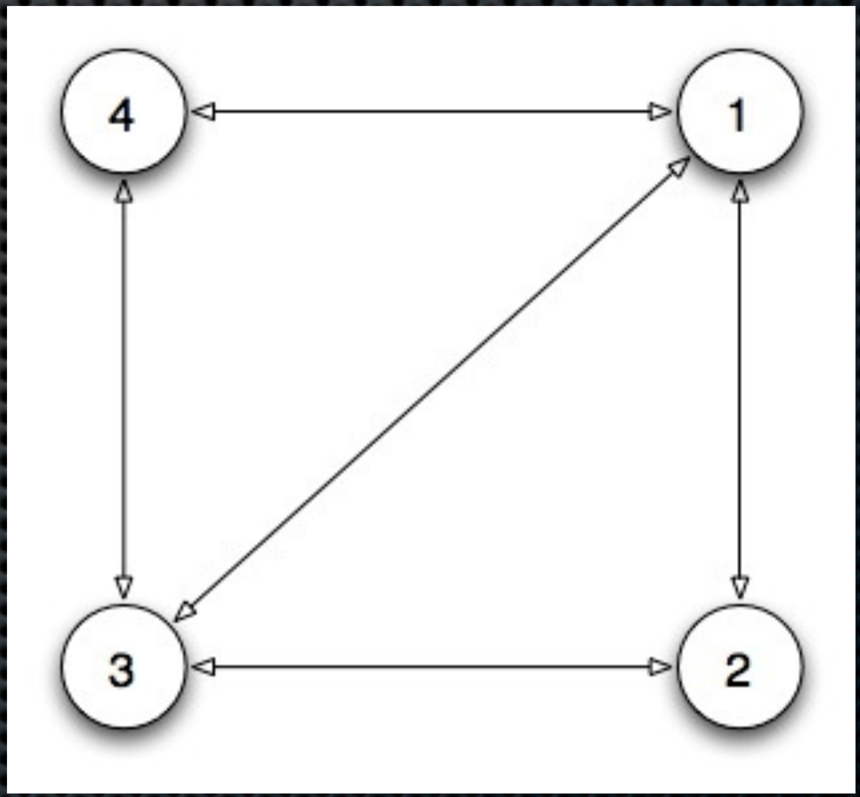


Solution Methodology

Local Search - Neighborhood

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

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

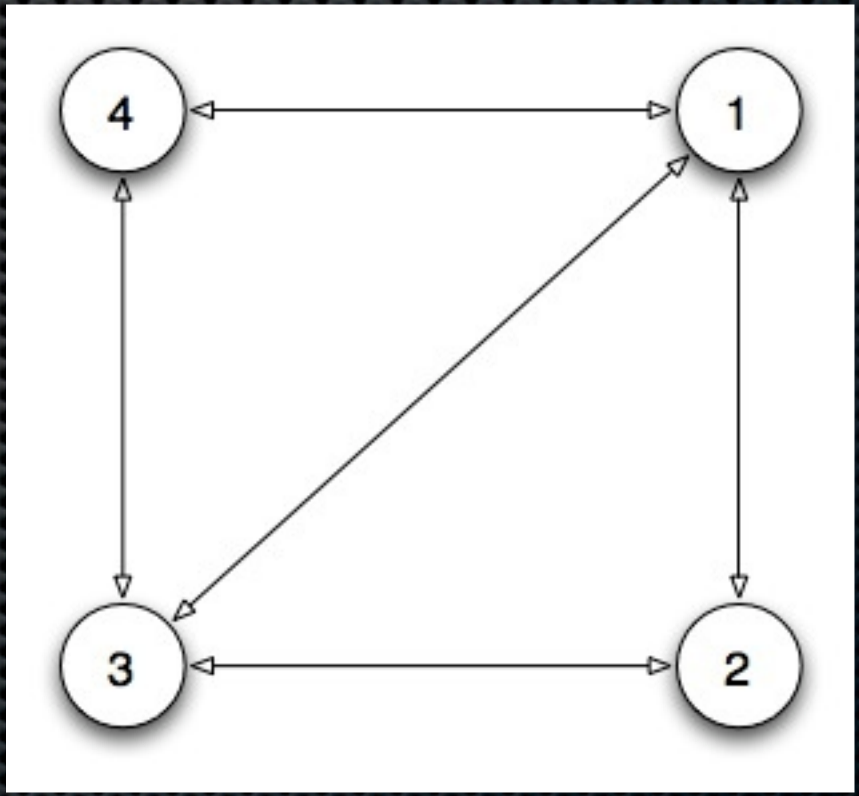


Solution Methodology

Local Search - Neighborhood

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

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

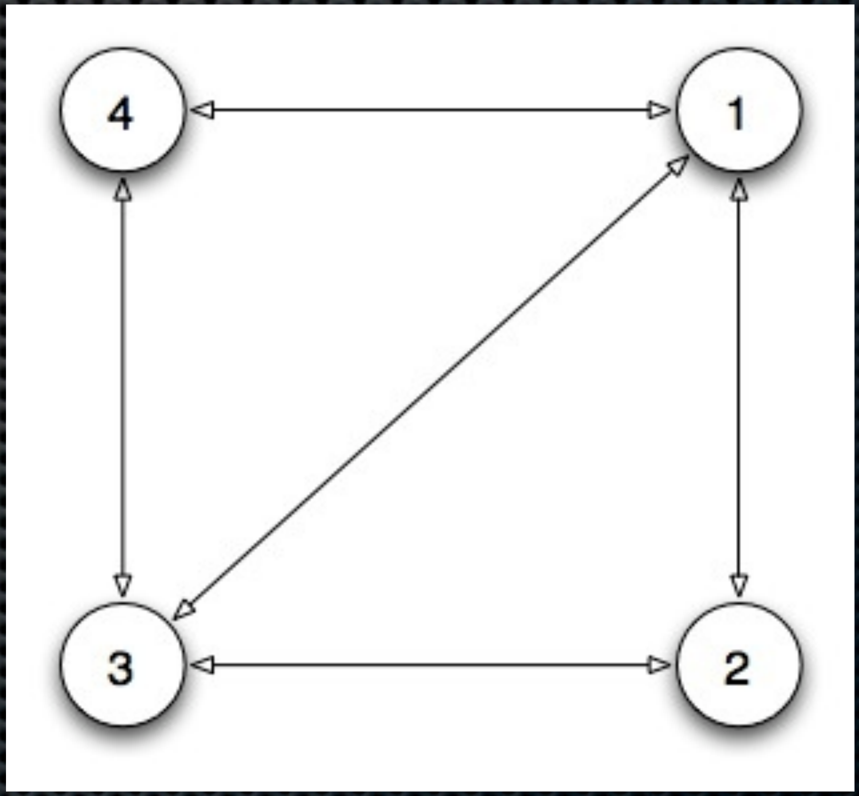


Solution Methodology

Local Search - Neighborhood

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

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

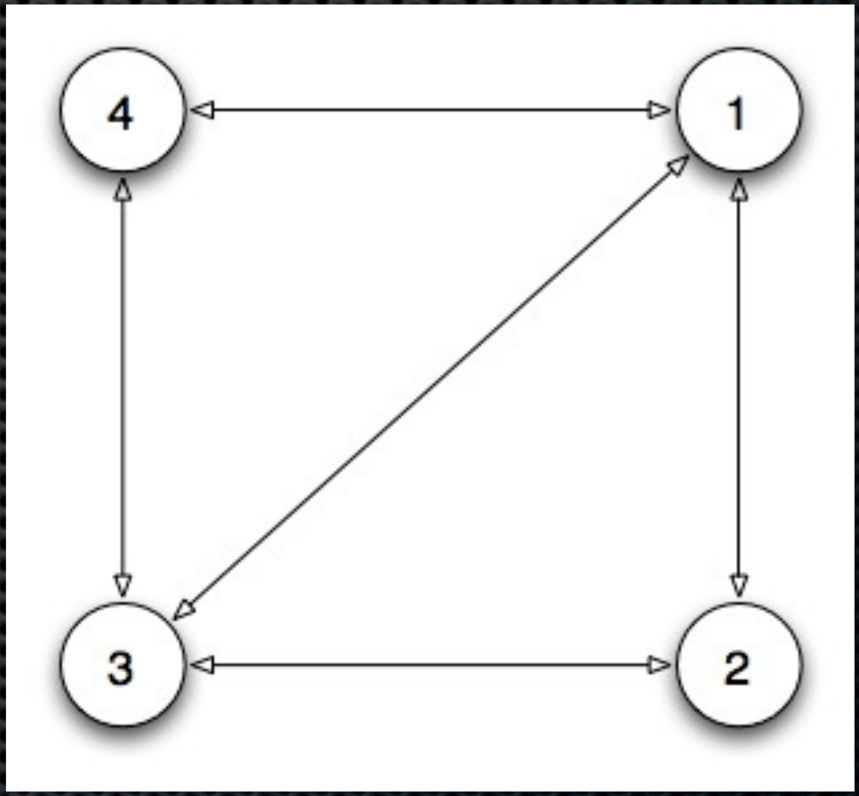


Solution Methodology

Local Search - Neighborhood

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

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

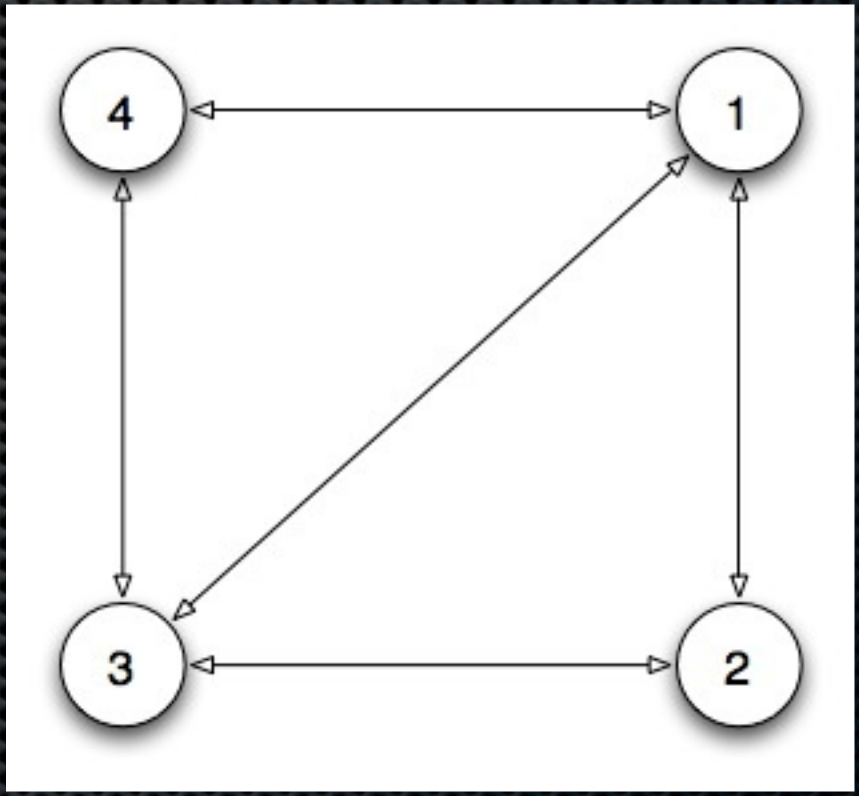


Solution Methodology

Local Search - Neighborhood

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

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

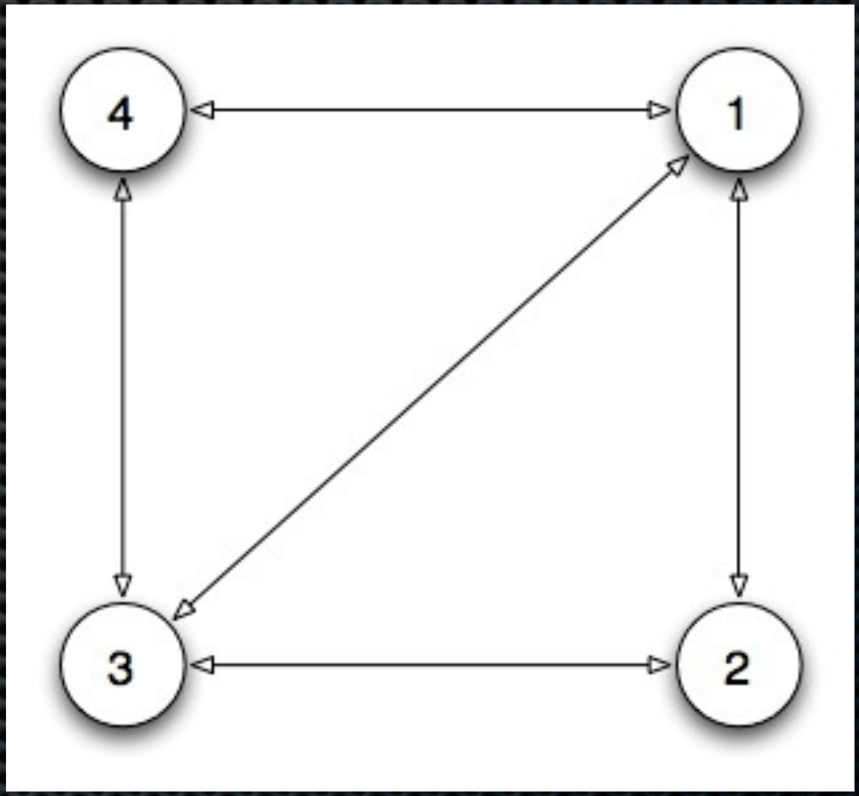


Solution Methodology

Local Search - Neighborhood

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

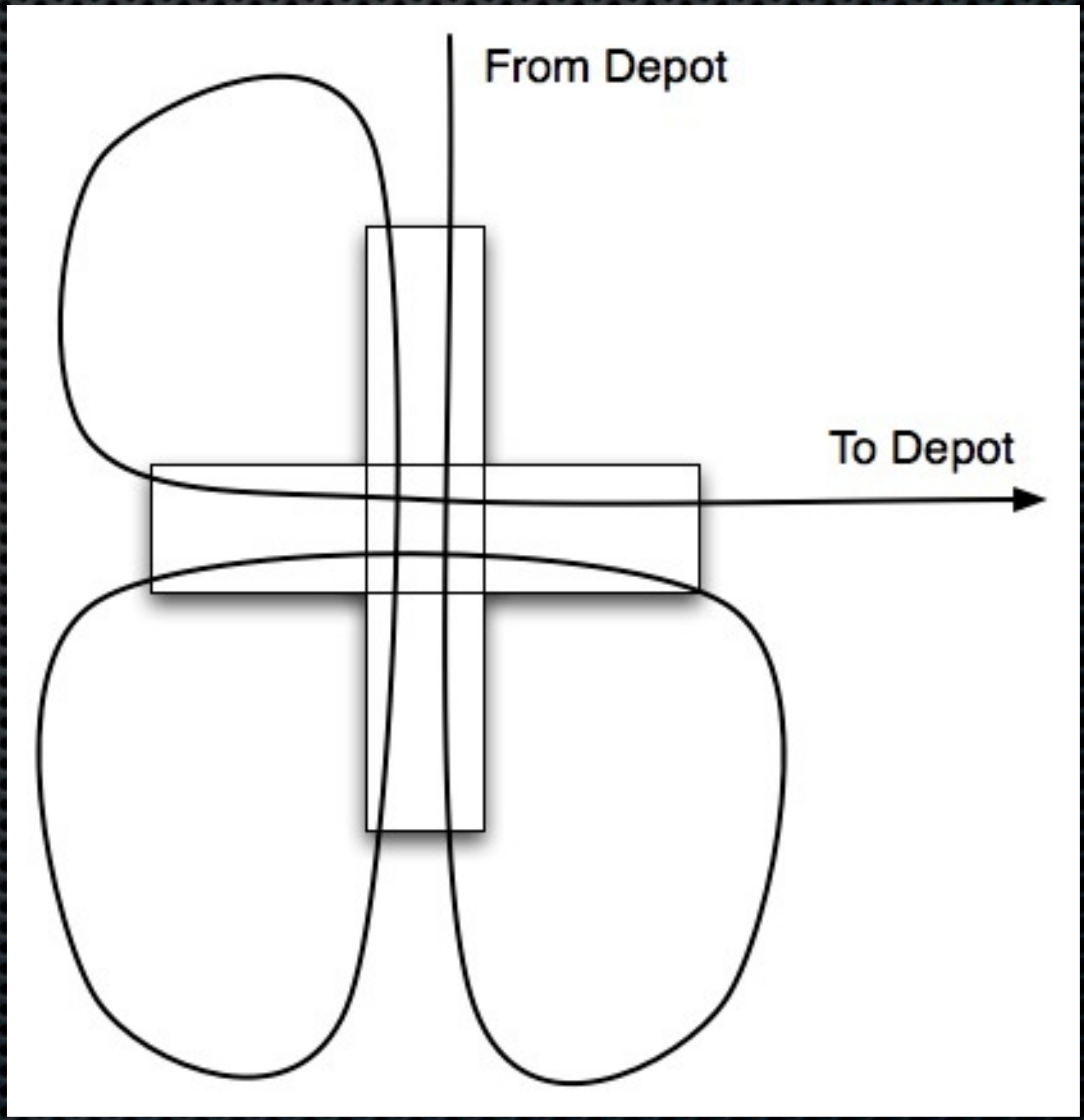
{1,2,3,4,1,3,4,1,2,3,1}



Solution Methodology

Local Search - Neighborhood

- 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

Local Search - Fitness

- For each arc, clearing duties must be assigned to the plows that traverse it
- Different grand cycles produce different routes when split, resulting in a different set of plows that traverse a given arc
- Optimal plowing assignment for a given split grand cycle is determined by using an IP
- IP is too computationally intensive for a local search
- Random allocation was found to produce near optimal solutions

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 randomly permuting cycles around different nodes a large number of times
- Return the best solution produced in 15 runs of local search and reinitialization

Solution Methodology

Lower Bounds

- DPP in solution framework provides the length of an optimal route for a single plow
- The best scenario is for this optimal route to be split evenly amongst the *k* plows
- Computational results are compared against this lower bound
  - If we match the lower bound then we have obtained the optimal solution

Computational Results

- We test our algorithm on 20 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 (64 to 196)
  - Number of arcs (116 to 316)
- Performed tests for 2, 3, 4, and 5 plows

Computational Results

- Our heuristic performs very well
  - 0.09% average deviation from lower bound for 2 plows
  - 0.49% average deviation for 3 plows
  - 0.74% average deviation for 4 plows
  - 1.92% average deviation for 5 plows
  - Max deviation over all instances is 4.67%
- Optimal solution for at least 14 out of 80 total instances

Conclusions

- Introduced the Min-Max *k*-MDPP variant of the WPP
- Our heuristic generates high-quality solutions
- Managerial implications
  - Lower bound, obtained by splitting a single optimal tour into *k* equal-length routes, is often tight
  - Route length and route balancing are *not* competing objectives
- Future work
  - Incorporate turn penalties
  - Incorporate precedence, where an unplowed street must be plowed before it can be deadheaded