UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

OAR Lib:
An Open Source Arc Routing Library

COLLEGE OF COMPUTER, MATHEMATICAL, & NATURAL SCIENCES

Oliver Lum
Research Advisor: Bruce Golden
Course Advisors: Radu Balan, Kayo Ide

What is OAR Lib?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- An open-source java library aimed at new operations researchers in the field of arc routing.
- An architecture for future software development in routing and scheduling.
- Design Philosophy: Usability First, Performance Second
- Well-Documented

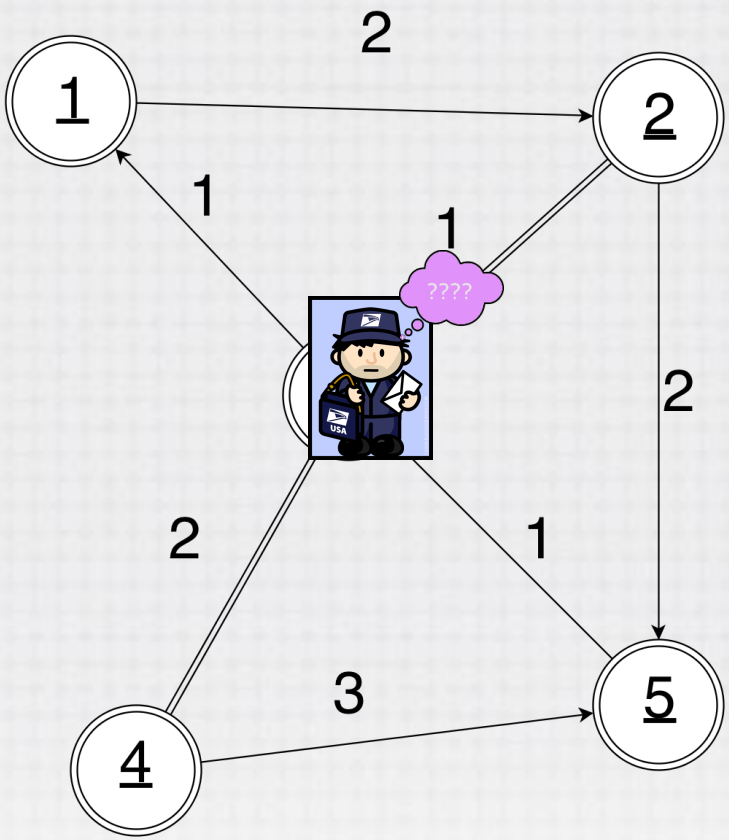
Why Java?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

1. Industry-standard
2. Balances developer-friendliness with speed
3. Java Native Interface (JNI) allows for C/C++ interfacing
4. Portable
5. Massive-Scale Parallelism Possible
6. Ease of Coding

What's the Problem?

- Similar to the well-studied Vehicle Routing Problems (VRPs)
- Seeks a minimum cost traversal of a graph's arcs:
  - Chinese Postman Problem
    - Undirected
    - Directed
    - Mixed
    - Windy
    - Rural Variants

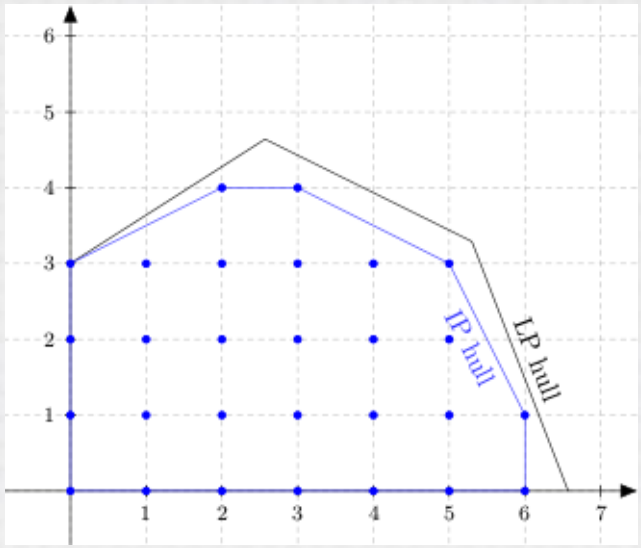


Applications

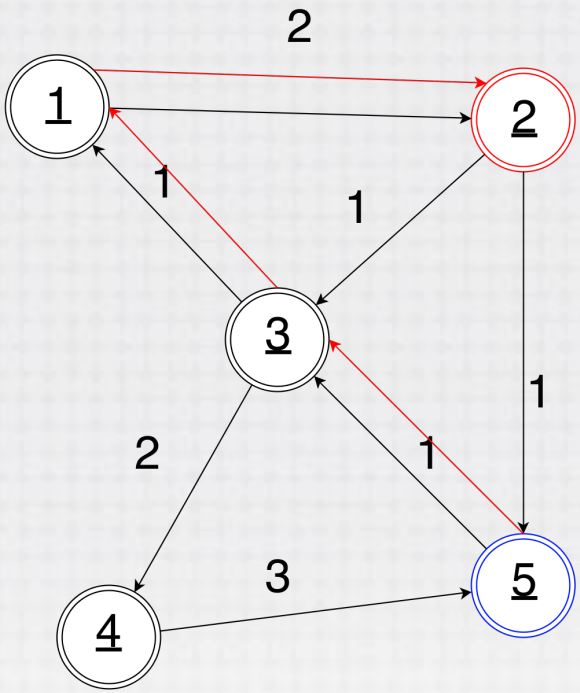
- Ubiquitous:
  - Package Delivery
  - Snow Plowing
  - Military Patrols
- Various interesting wrinkles:
  - Time-Windows
  - Close-Enough
  - Turn Penalties
  - Asymmetric Costs

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

IP Solvers



Constructive Heuristics



IP Formulation [10]

$$\text{minimize} \sum_{i \text{ or } j \in \{D^+ \cup D^-\}} c_{ij} x_{ij}$$

subject to:

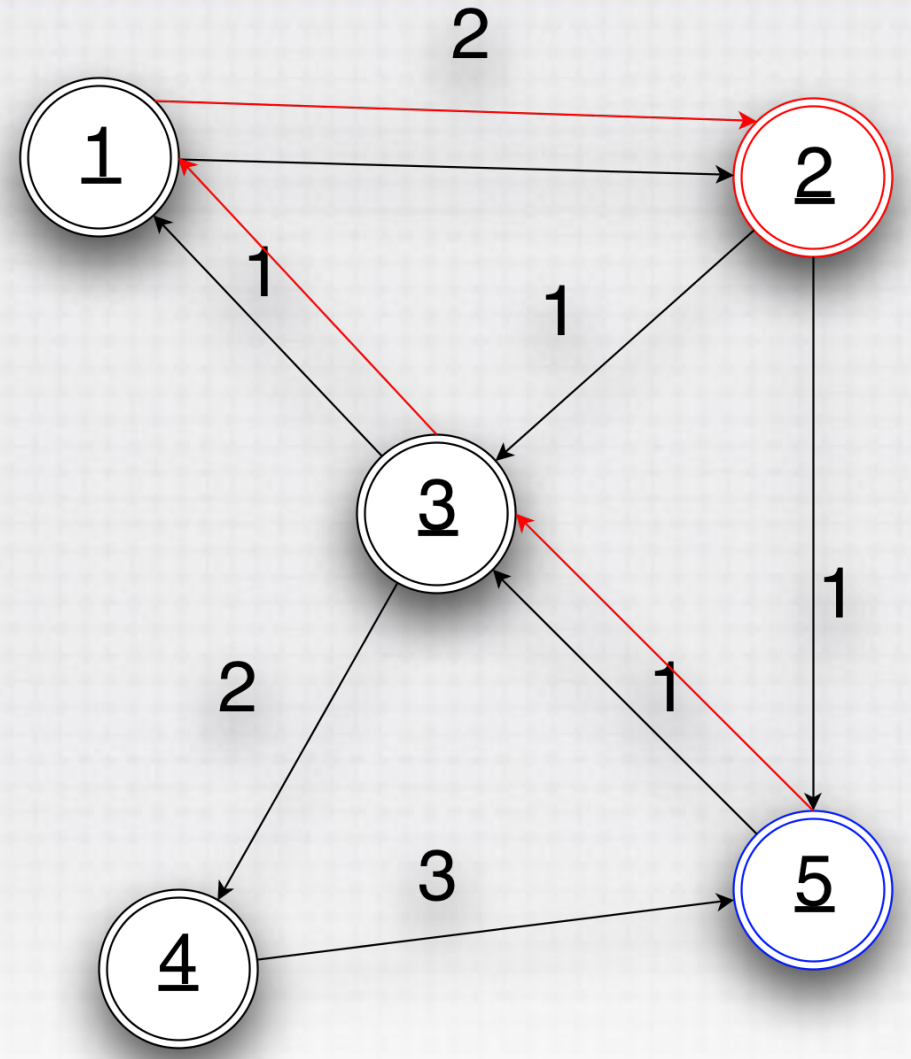
$$\sum_{j \in D^+} x_{ij} = -\delta(i), \ \forall i \in D^-$$

$$\sum_{i \in D^-} x_{ij} = \delta(j), \quad \forall j \in D^+$$

$$x_{ij} \in \mathbb{Z}_+^0$$

- Cost function: C
- f_{ij} : represents number of additional times we traverse the shortest path from i to j.
- $\delta(v)$: outdegree – indegree of vertex v.
- $D\uparrow+$: set of vertices with excess outgoing arcs.
- $D\uparrow-$: set of vertices with excess incoming arcs.

Directed Solver [10]



OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. **What's In the Library?**
7. Example
8. Further Work
9. Special Thanks
10. References

IP Formulation [5]

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

$$\text{minimize} \quad \sum_{(i,j)\in E} c_{ij}x_{ij}$$

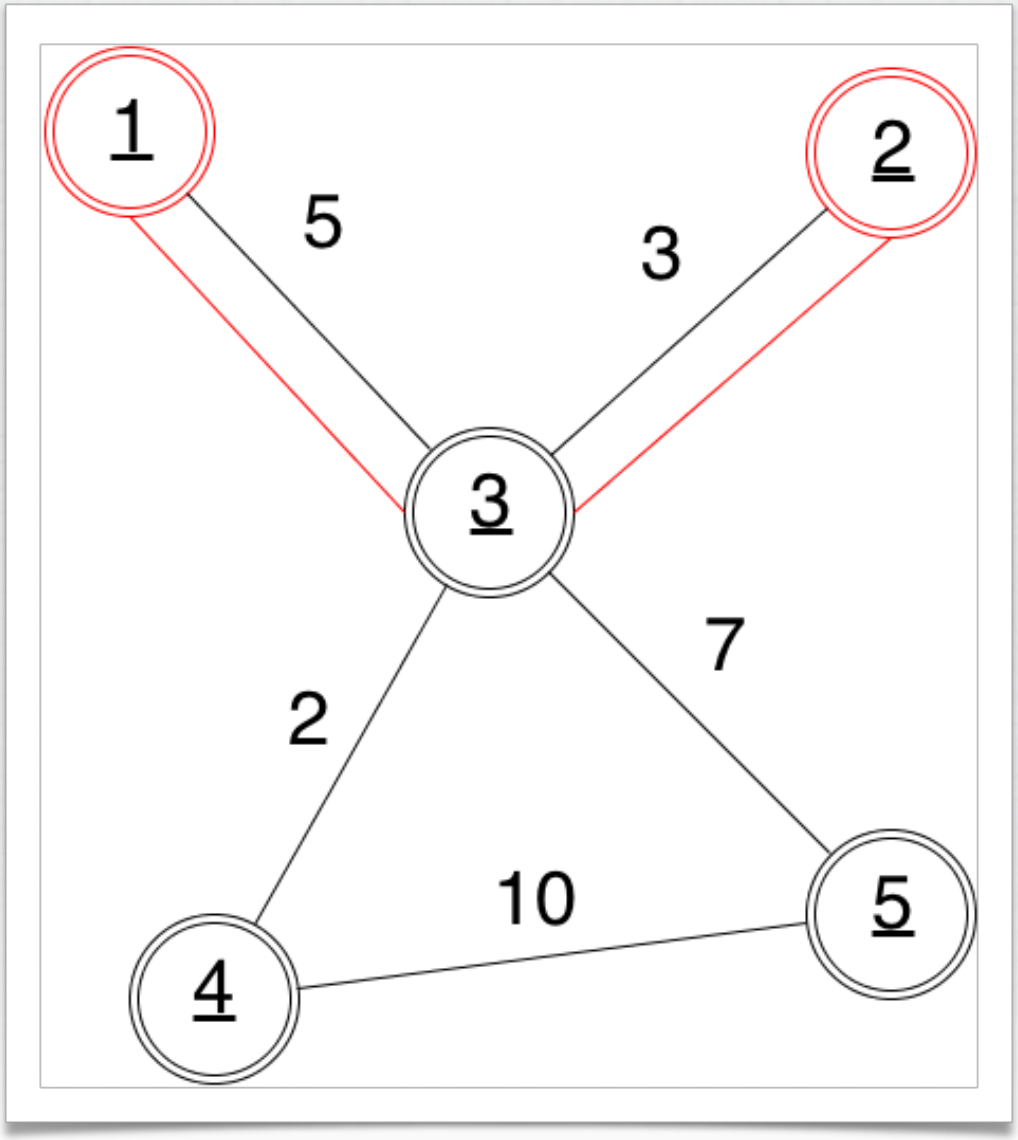
$$\text{subject to:}$$

$$\sum_{(i,j)\in E_v} (x_{ij}+1) \equiv 0 \ mod\ 2, \forall v \in V$$

$$x_{ij} \in \mathbb{Z}_{+}^{0}$$

- Cost function: C
- Edge set: E
- Vertex set: V
- t_e : represents number of additional times we traverse edge e.
- $\delta(v)$: set of edges incident on v.

Undirected Solver [5]



OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. **What's In the Library?**
7. Example
8. Further Work
9. Special Thanks
10. References

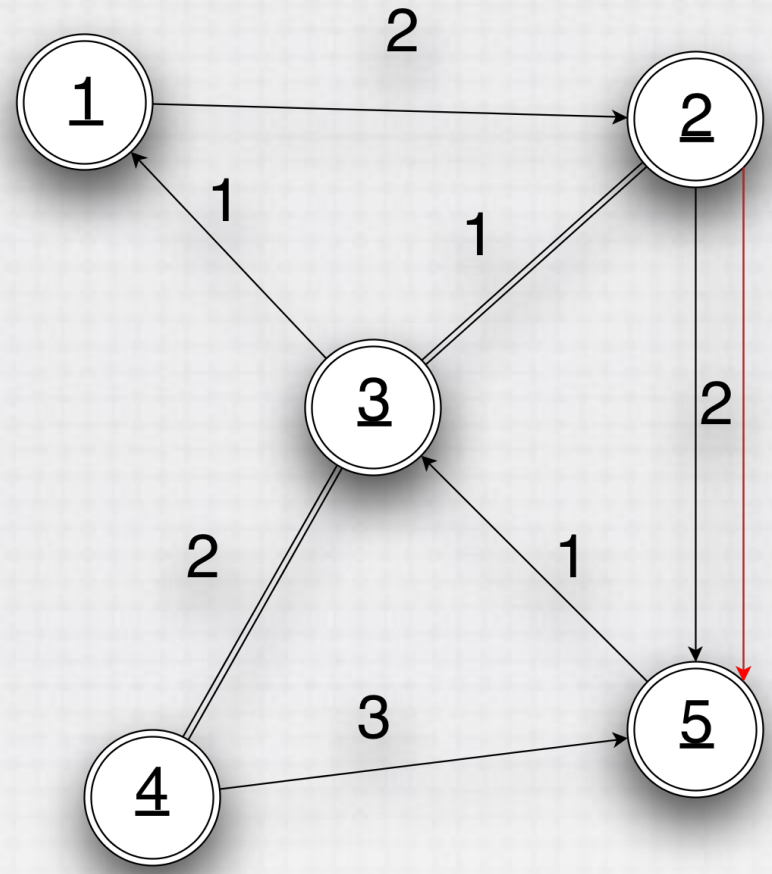
IP Formulation

$$\text{minimize} \quad \sum_{s \in \{A \cup \hat{E} \cup \check{E}\}} c_s x_s$$

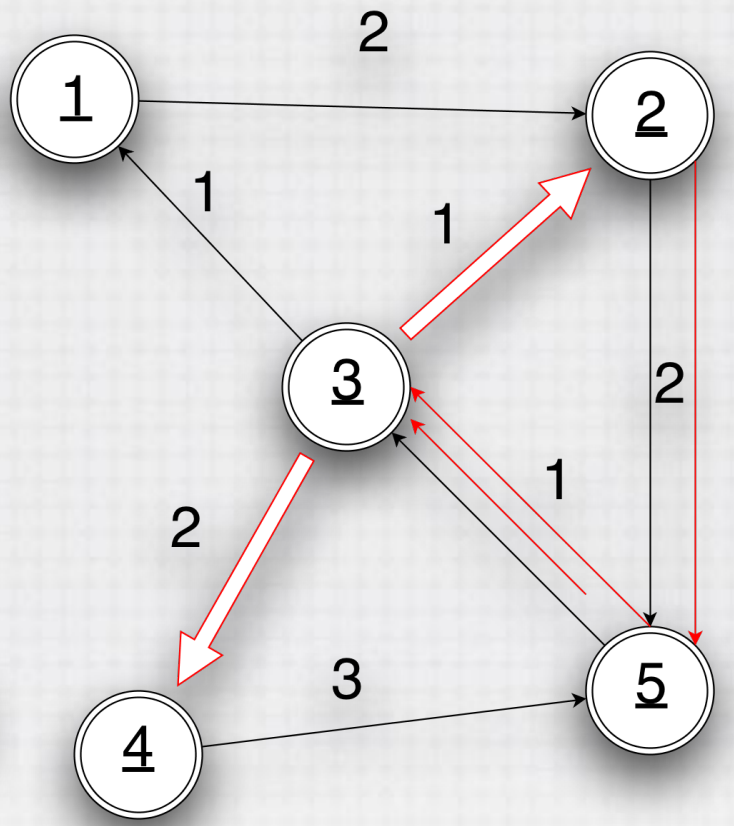
subject to:

$$\begin{aligned}
y'_e + y'_{\bar{e}} &\geq 1, && \forall e \in E \\
x_s &= y'_s + y_s, && \forall s \in A \cup \hat{E} \cup \check{E} \\
\sum_{s \in S_v^+} x_s - \sum_{s \in S_v^-} x_s &= 0, && \forall v \in V \\
y'_a &= 1, && \forall a \in A \\
y'_e &\in \{0,1\}, && \forall e \in \hat{E} \cup \check{E} \\
y_s &\in \mathbb{Z}_+^0
\end{aligned}$$

- Cost function: C
- x_s : represents number of times we traverse link s.
- $\delta\uparrow+(v)$: arcs / directed edges going out of v.
- $y\downarrow e\uparrow'$: 1 if we orient edge e from i to j, o oth.
- $y\downarrow s$: the number of additional times we traverse link s.



Even Degree

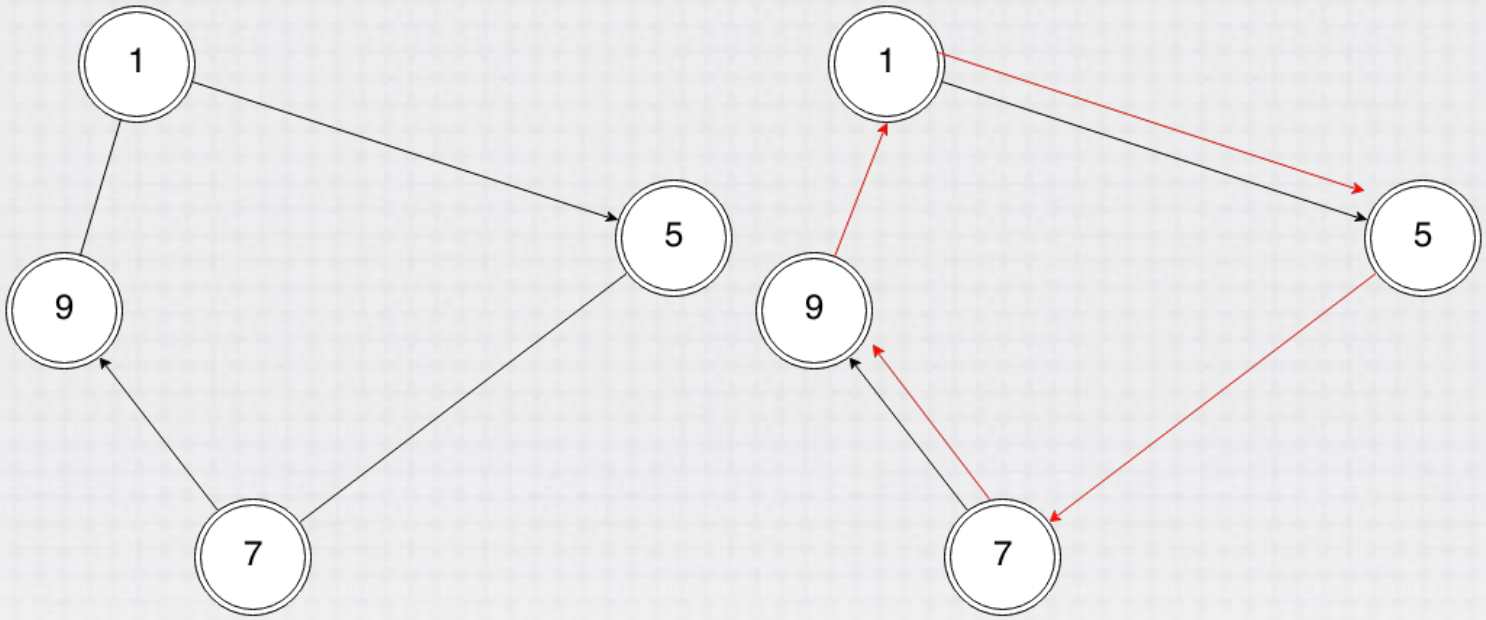


In-Out Degree

OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. **What's In the Library?**
7. Example
8. Further Work
9. Special Thanks
10. References

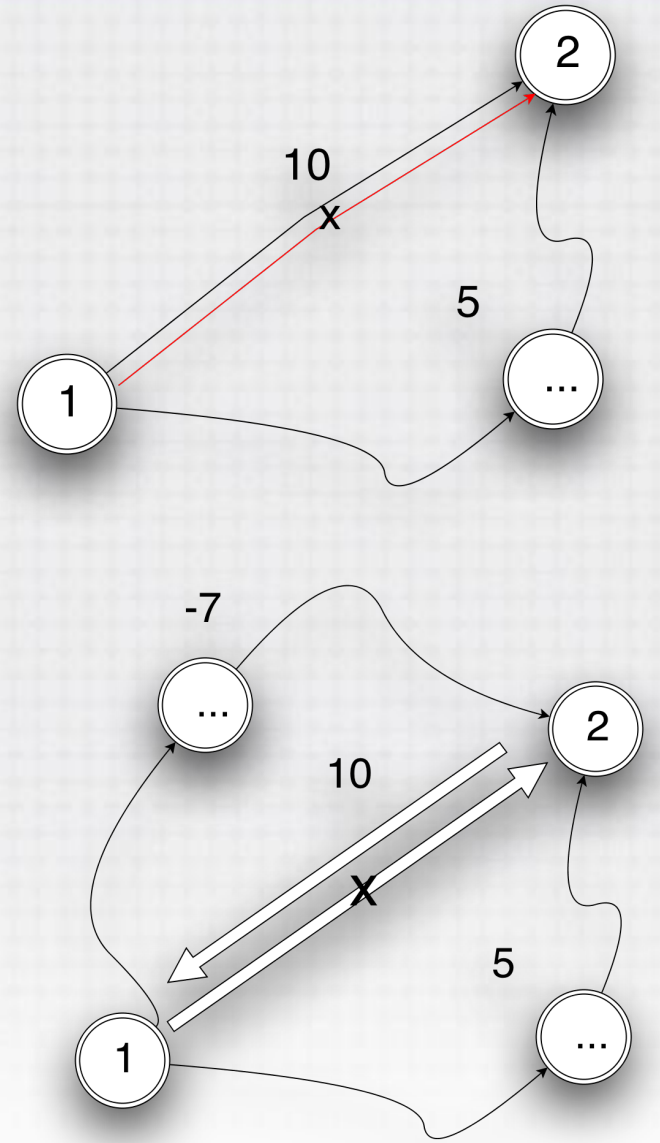
Mixed 1 [8]



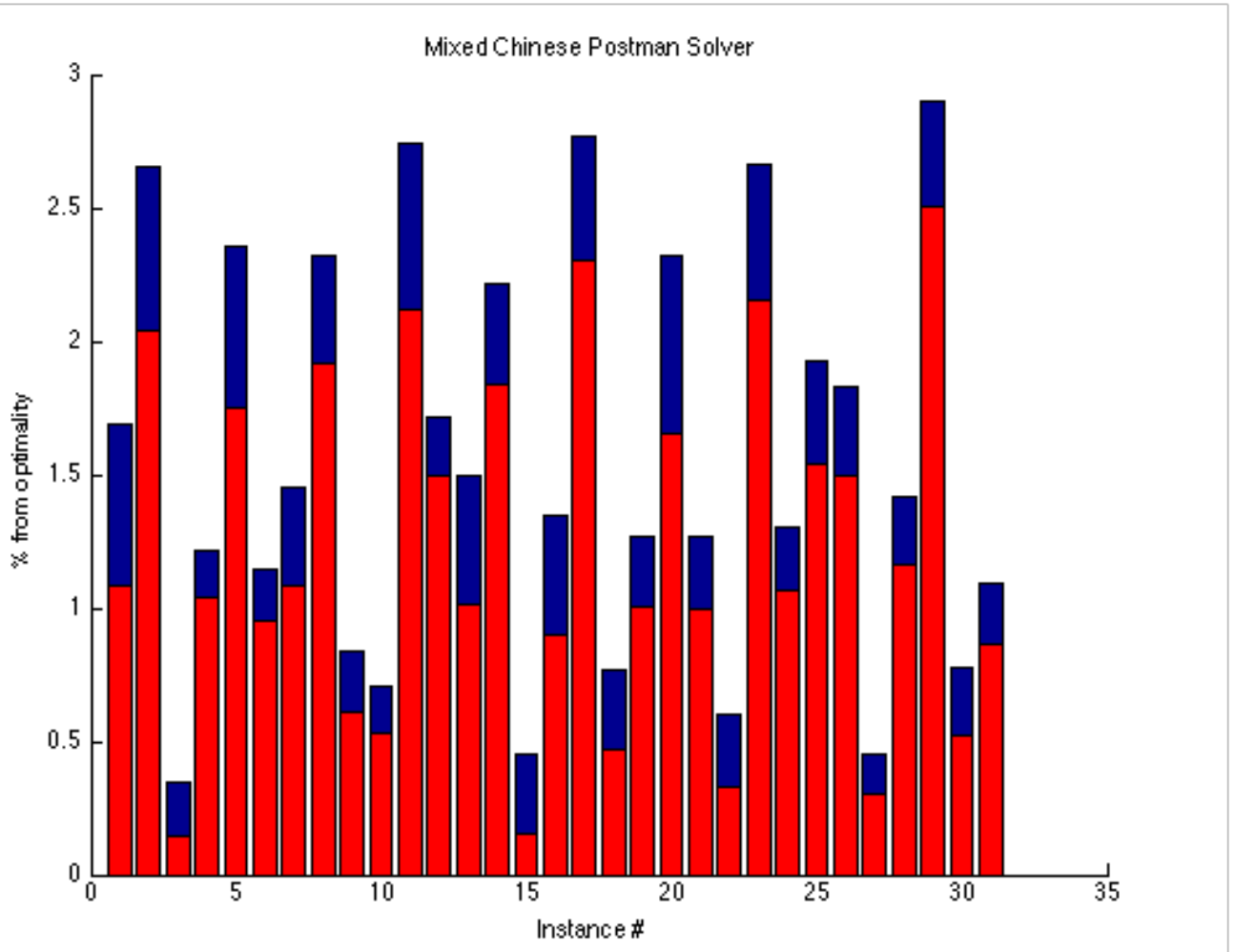
Even Parity

Mixed 2

- There are two ideas encapsulated by the algorithm:
  - Wherever possible, replace an added arc with a ‘shortest path’ if it reduces cost.
  - Wherever possible, reverse the direction of a directed edge and add two ‘shortest paths’ from i to j if it reduces cost.



Computational Results



OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. **What's In the Library?**
7. Example
8. Further Work
9. Special Thanks
10. References

IP Formulation [9]

$$\text{minimize} \quad \sum_{e^+ \in E^+} c_{e^+} x_{e^+} + \sum_{e^- \in E^-} c_{e^-} x_{e^-}$$

subject to:

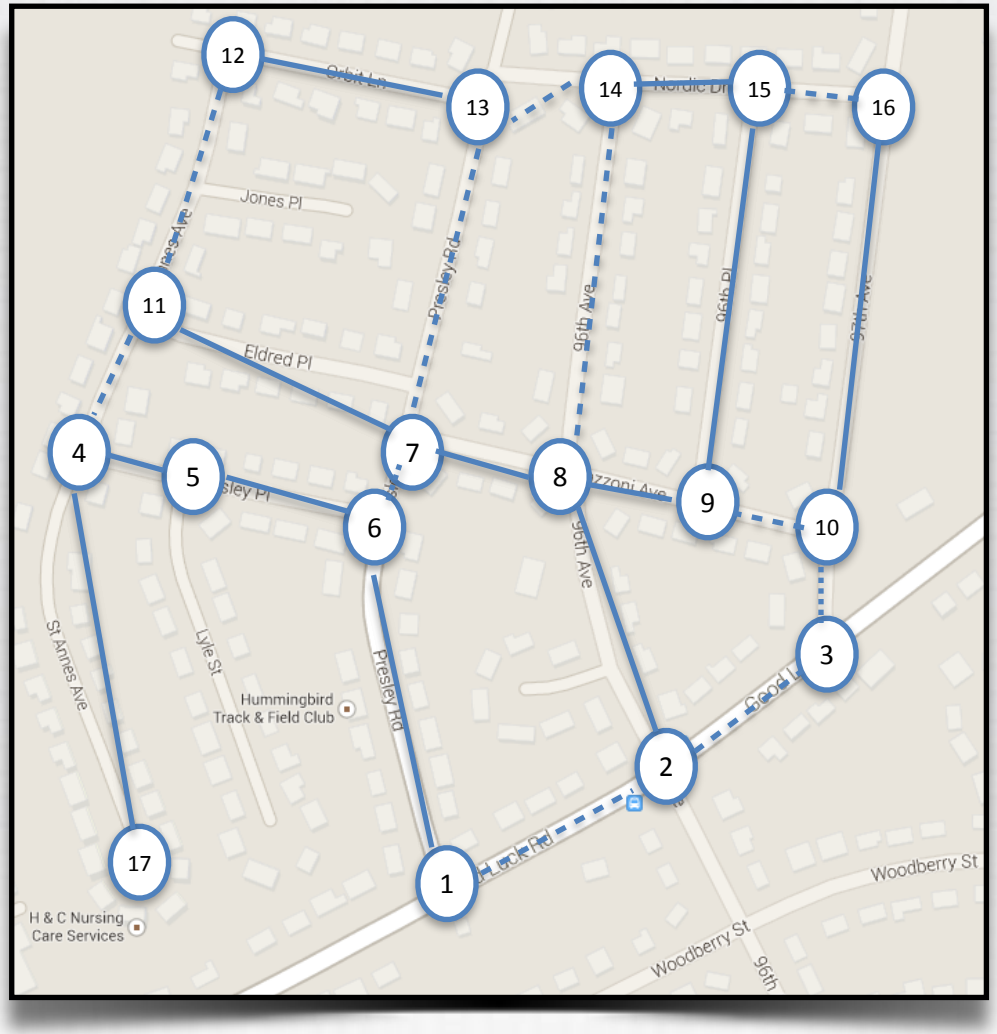
$$\sum_{e^+ \in E^+} x_{e^+} - \sum_{e^- \in E^-} x_{e^-} = 0, \quad \forall v \in V$$

$$x_{e^+} + x_{e^-} \geq 1, \quad \forall e \in E$$

$$x_{e^+}, x_{e^-} \in \mathbb{Z}_+^0, \quad \forall e \in E$$

- Cost function: C
- x_{e^+} : represents number of times we traverse edge e from i to j.
- $\delta(v)$: arcs / directed edges going into v.

Example [1]

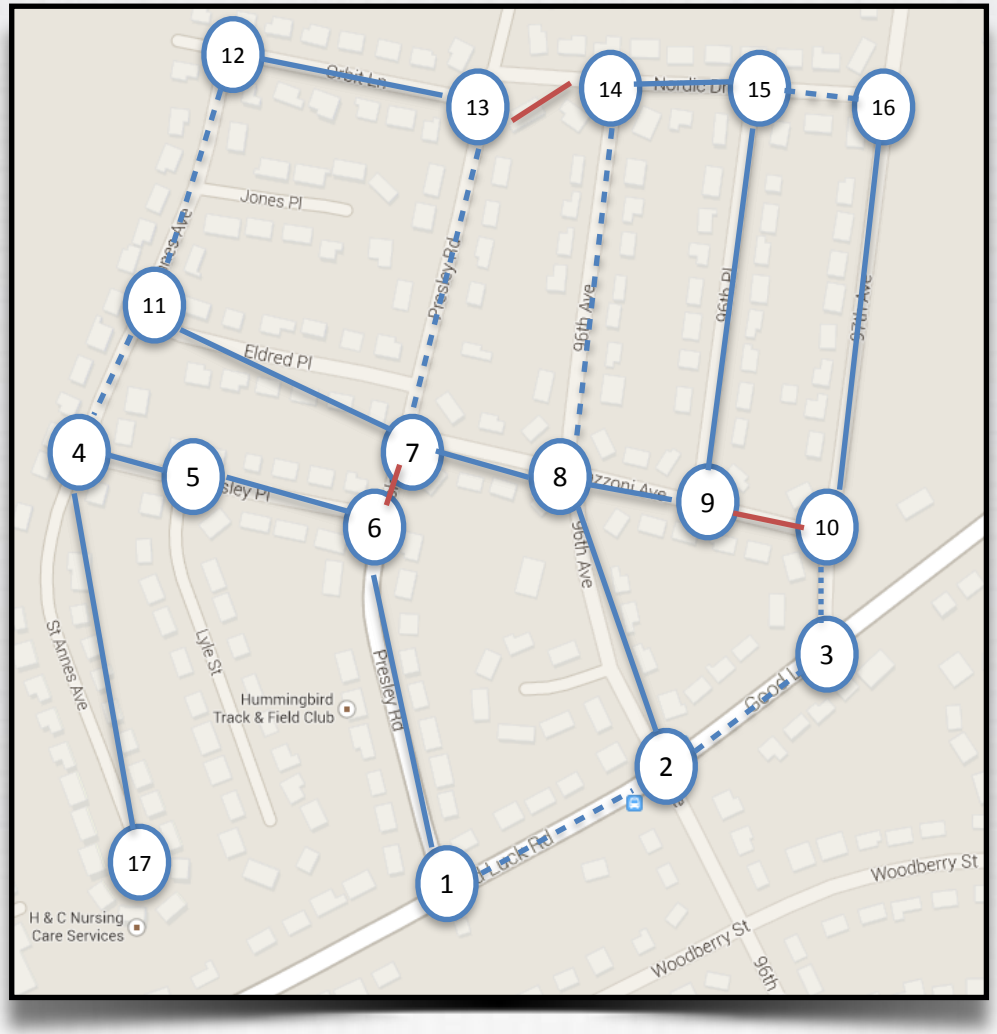


- Rural Postman Problem
  - Set of required / non-required edges

OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. What's In the Library?
7. **Example**
8. Further Work
9. Special Thanks
10. References

Example [1]

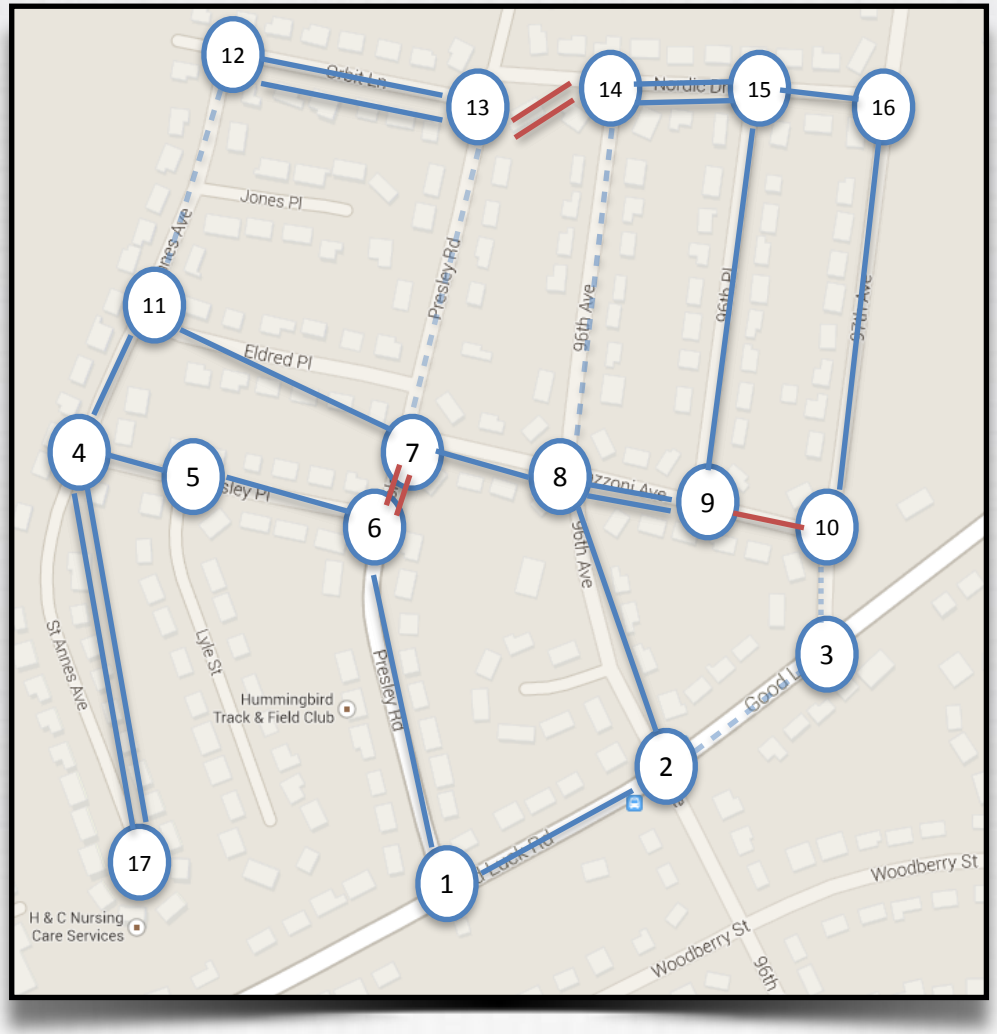


- Rural Postman Problem
  - Set of required / non-required edges
  - Solve an MST over the required components

OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. What's In the Library?
7. **Example**
8. Further Work
9. Special Thanks
10. References

Example [1]

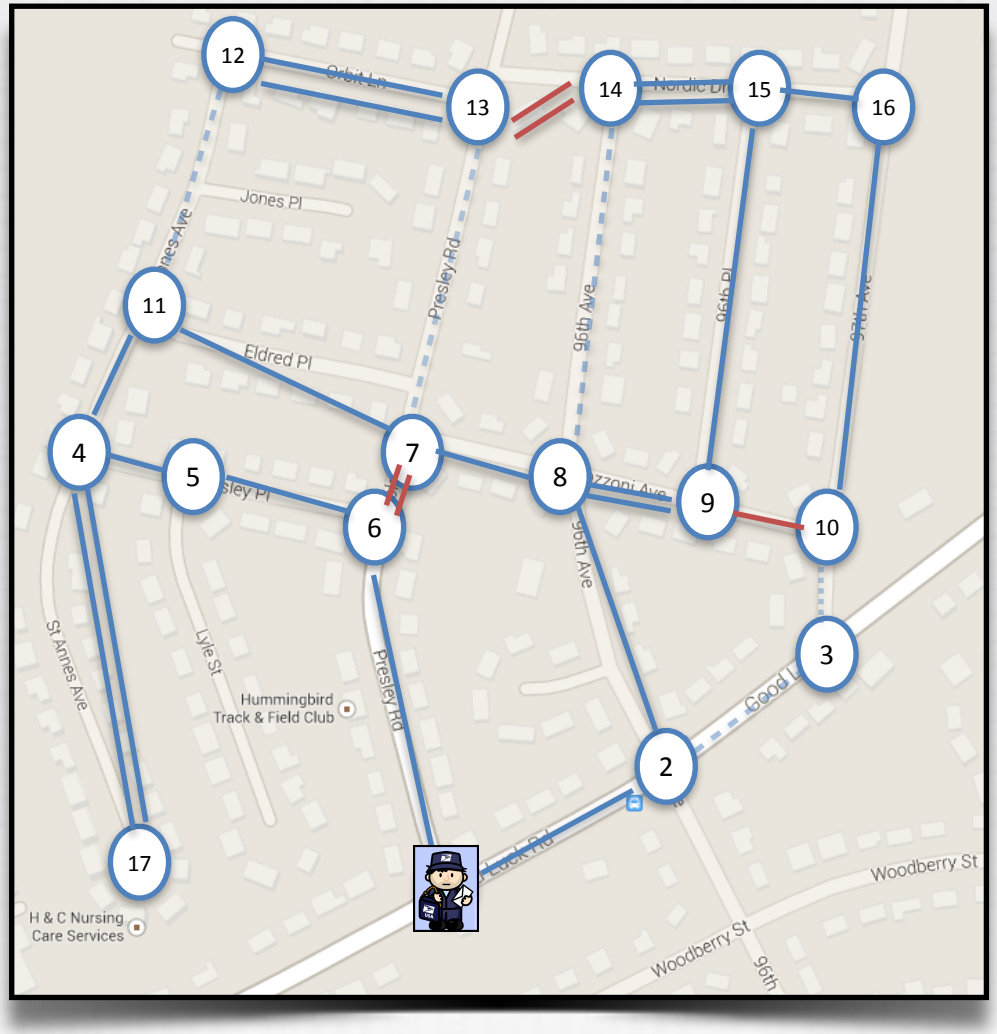


- Rural Postman Problem
  - Set of required / non-required edges
  - Solve an MST over the required components
  - Solve a min-cost matching over the remaining odd vertices

OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. What's In the Library?
7. **Example**
8. Further Work
9. Special Thanks
10. References

Example [1]



- Rural Postman Problem
  - Set of required / non-required edges
  - Solve an MST over the required components
  - Solve a min-cost matching over the remaining odd vertices
  - Solve a min-cost flow to determine the optimal tour

OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. What's In the Library?
7. **Example**
8. Further Work
9. Special Thanks
10. References

Example

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Code Snippet:

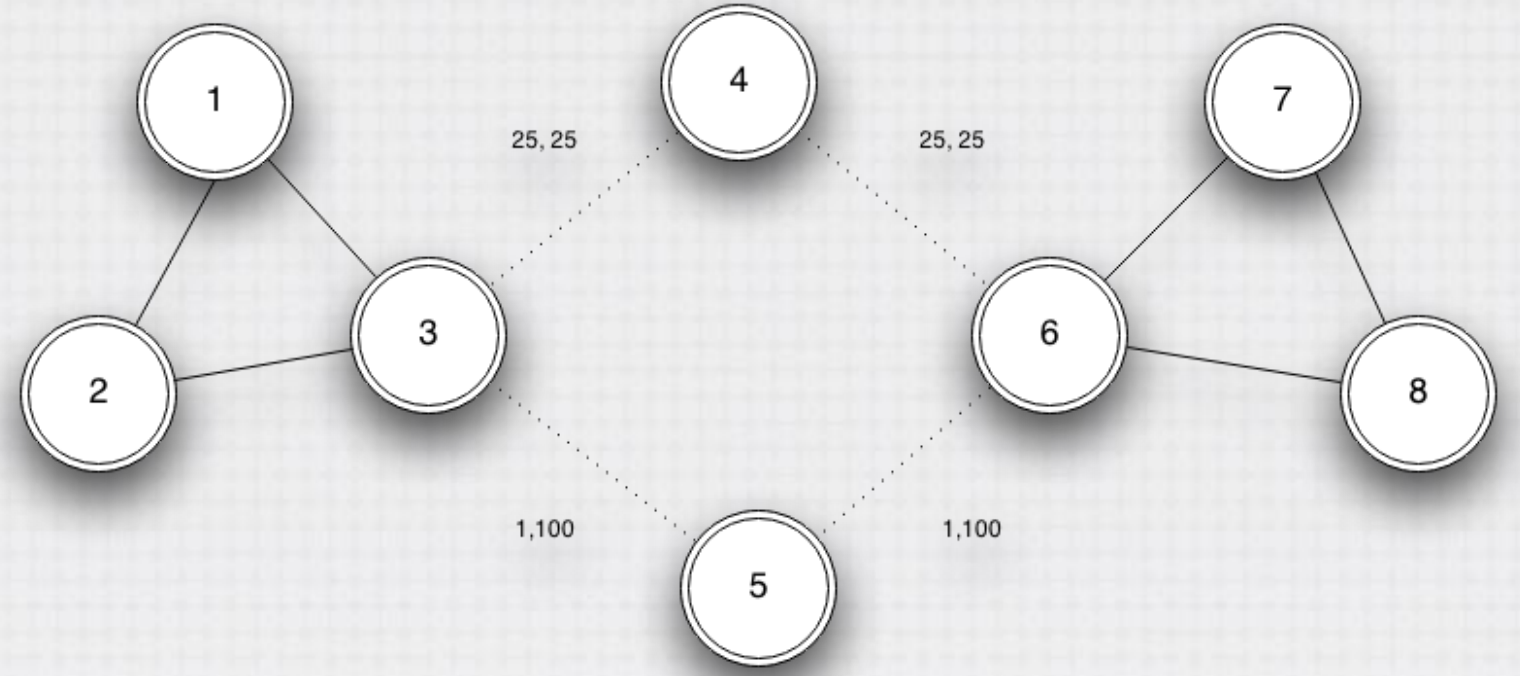
```
WindyGraph neighborhood = new WindyGraph(17);
neighborhood.addEdge(1, 2, 5, 5, false);
neighborhood.addEdge(2, 3, 4, 4, false);
neighborhood.addEdge(4, 17, 7, 7, true);
neighborhood.addEdge(1, 6, 6, 6, true);
neighborhood.addEdge(2, 8, 5, 5, true);
neighborhood.addEdge(3, 10, 3, 3, false);
neighborhood.addEdge(4, 5, 2, 2, true);
neighborhood.addEdge(5, 6, 3, 3, true);
neighborhood.addEdge(6, 7, 1, 1, false);
neighborhood.addEdge(7, 8, 3, 3, true);
neighborhood.addEdge(8, 9, 2, 2, true);
neighborhood.addEdge(9, 10, 2, 2, false);
neighborhood.addEdge(4, 11, 3, 3, false);
neighborhood.addEdge(7, 11, 5, 5, true);
neighborhood.addEdge(11, 12, 5, 5, false);
neighborhood.addEdge(7, 13, 7, 7, false);
neighborhood.addEdge(8, 14, 8, 8, false);
neighborhood.addEdge(9, 15, 7, 7, true);
neighborhood.addEdge(10, 16, 8, 8, true);
neighborhood.addEdge(12, 13, 4, 4, true);
neighborhood.addEdge(13, 14, 3, 3, false);
neighborhood.addEdge(14, 15, 3, 3, true);
neighborhood.addEdge(15, 16, 2, 2, false);

WindyRPP testProblem = new WindyRPP(neighborhood);
WRPPSolver_Win testSolver = new WRPPSolver_Win(testProblem);
Route ans = testSolver.trySolve();
System.out.println(ans.toString());
```

```
Cost is: 91
6-7-11-4-17-4-5-6-1-2-8-9-15-14-13-12-13-14-15-16-10-9-8-7-6
```

Graph Initialization:

- Can be done by hand (left)
- Can be parsed from a text file using GraphReader object
- Modular Graph / Problem / Solver architecture provides

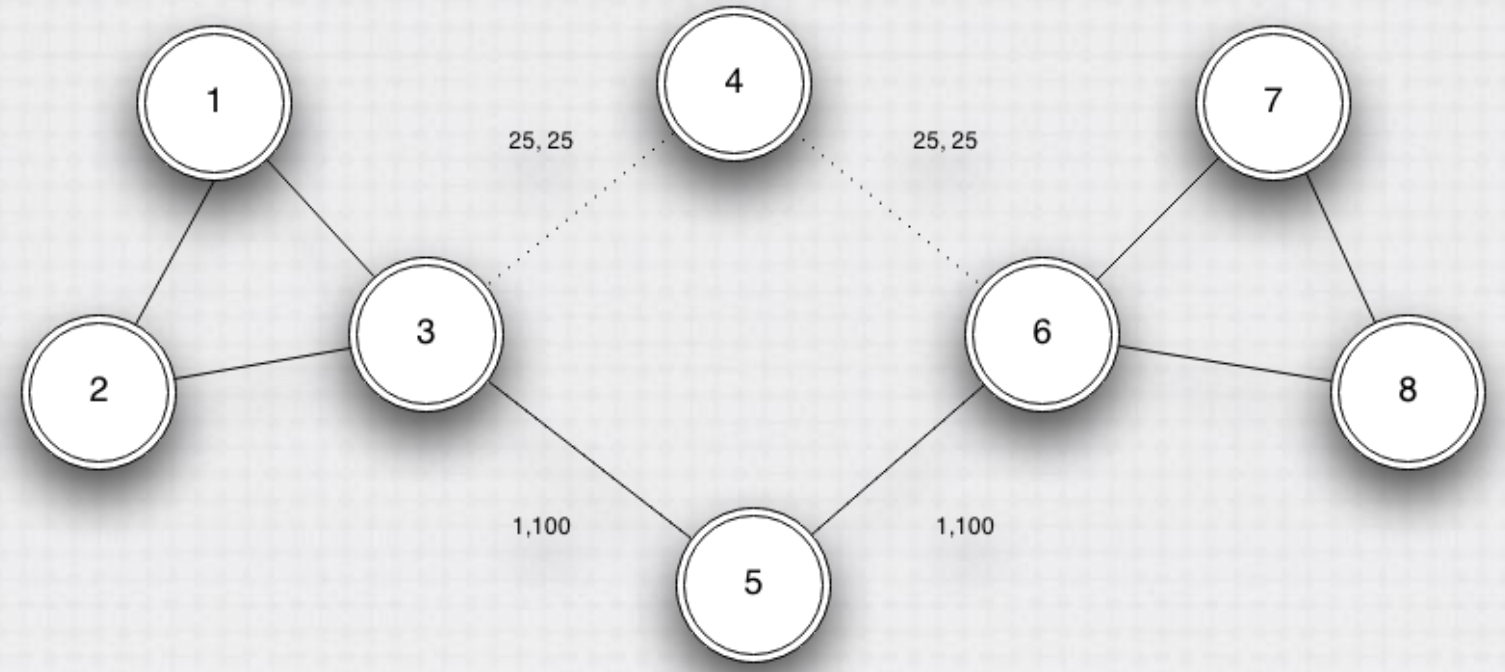
Windy Rural 2 [1]



OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. **What's In the Library?**
7. Example
8. Further Work
9. Special Thanks
10. References

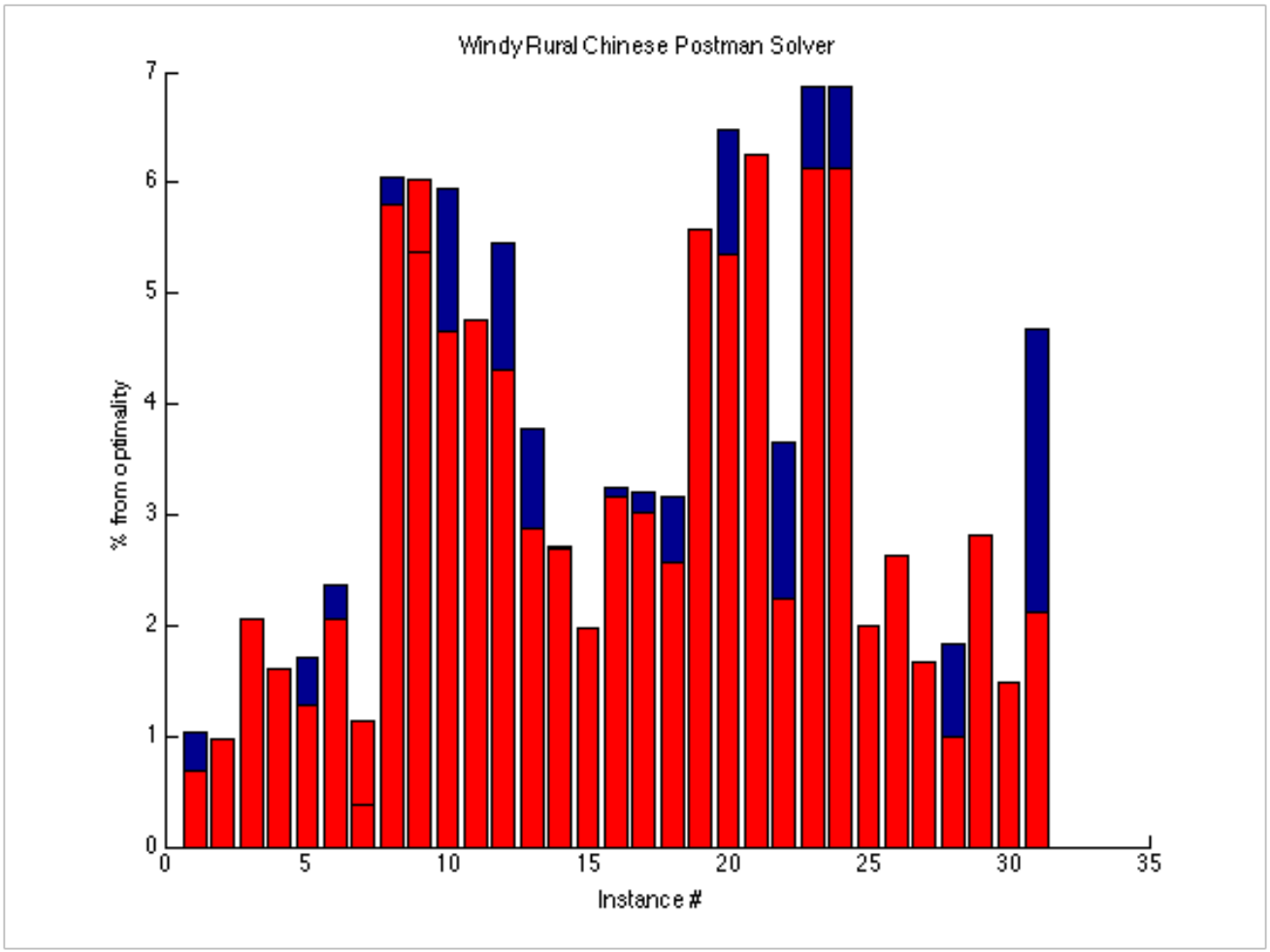
Windy Rural 2 [1]



OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. **What's In the Library?**
7. Example
8. Further Work
9. Special Thanks
10. References

Computational Results



OAR Lib

1. Background
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. **What's In the Library?**
7. Example
8. Further Work
9. Special Thanks
10. References

What's In the Library?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Problem	Classical	Modern
Undirected Chinese Postman Problem	1. Edmonds 2. Exact IP	N/A
Directed Chinese Postman Problem	1. Edmonds 2. Exact IP	N/A
Mixed Chinese Postman Problem	1. Frederickson 2. Exact IP	Yaoyuenyeung
Windy Chinese Postman Problem	1. Win 2. Exact IP	Benavent
Directed Rural Postman Problem	1. Christofides	N/A

What's In the Library?

Common Algorithms:

- Several Shortest Paths implementations
- Min-Cost Flow Implementation
- Hierholzer's Algorithm
- Min-Cost Spanning Tree
- Connected Components Algorithms
- Min-Cost Spanning Arborescence (JNI)
- Min-Cost Matching (JNI)

What's In the Library?

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

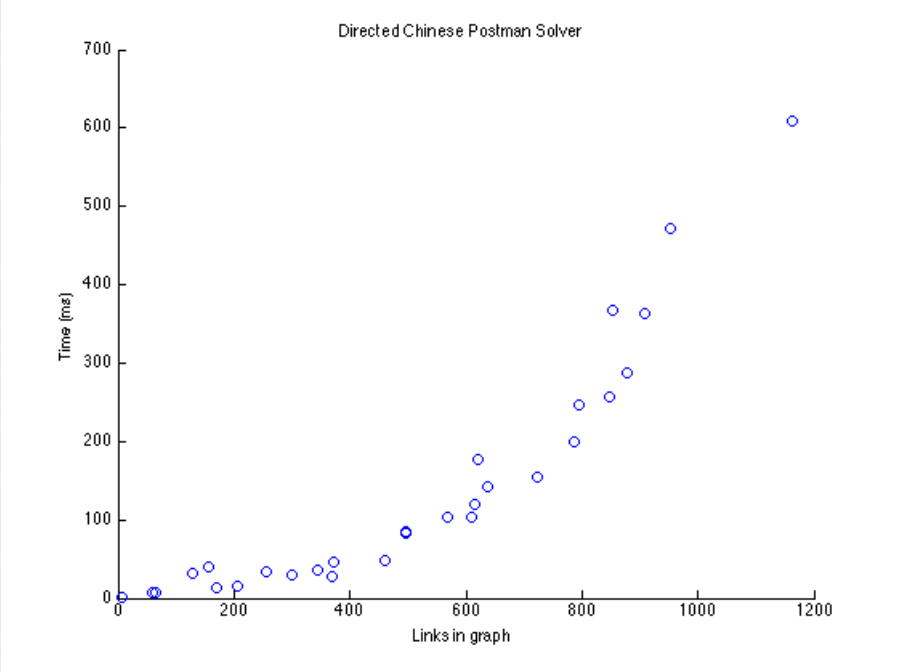
- Example Code Snippets
- Documentation
- Test Data
  - WPP / WRPP Albaida-Madrigueras Instances
  - DRPP Instances from Campos
  - MCPP Instances from Yaoyuenyong
  - Random test instances for UCPP, DCPP

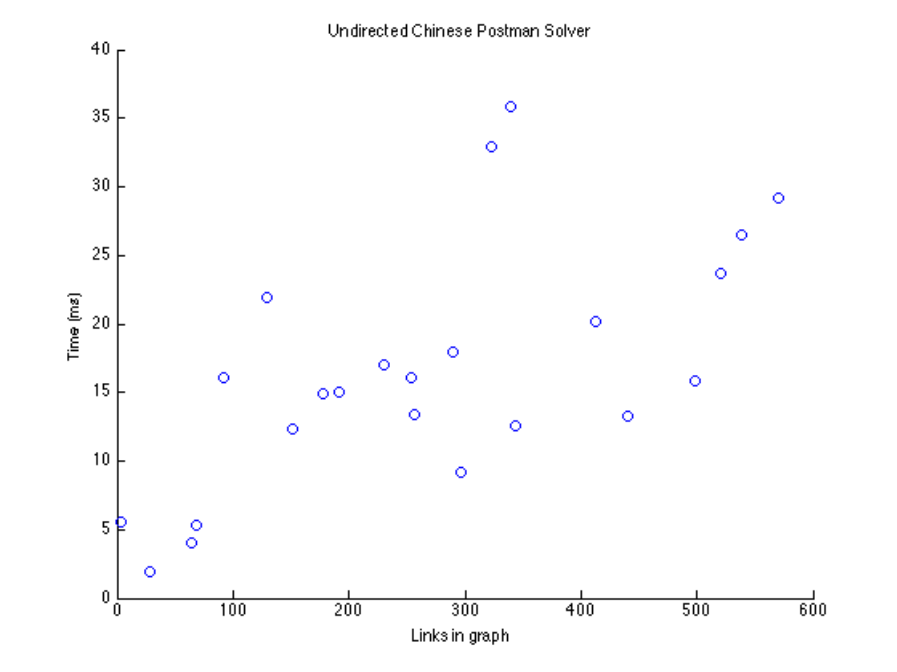
Validation

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

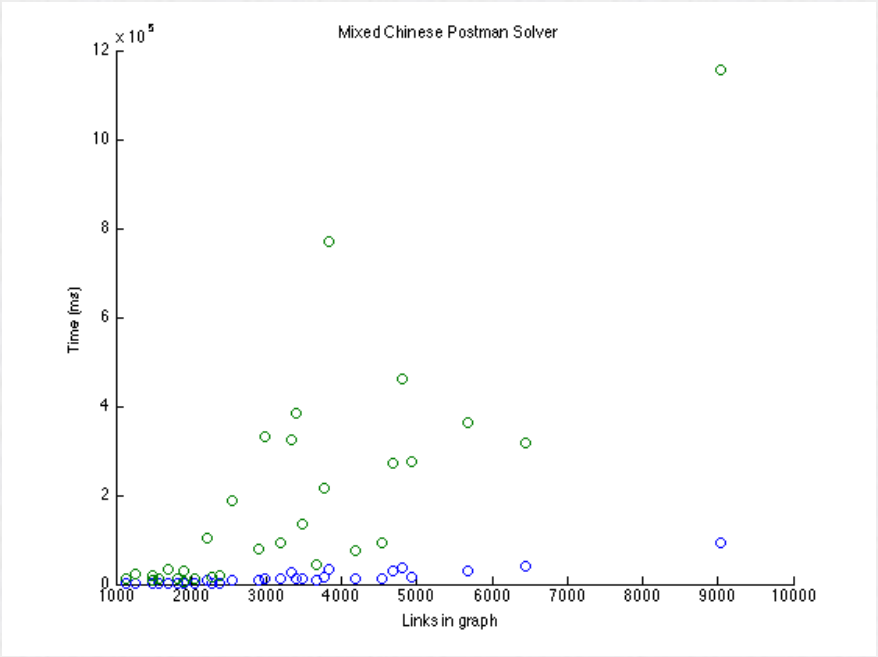
- DCPP: Compare Results to Gurobi Solver
- UCPP: Compare Results to Gurobi Solver
- MCPP1: Validate using the Yaoyuenyong instances, and reproducing the table given in the paper.
- MCPP2: Validate using the Yaoyuenyong instances, and reproducing the table given in the paper.
- MCPP Exact: Validate using the mixed Albaida-Madrigueras instances.
- WPP Exact: Validate using the windy Albaida-Madrigueras instances.
- WRPP 1: Validate using the windy rural Albaida-Madrigueras instances to duplicate average % deviation for the algorithm with / without improvement procedures.
- WRPP 2: Validate using the windy rural Albaida-Madrigueras instances to duplicate average % deviation for the algorithm with / without improvement procedures.
- DRPP: Validate using the Campos DRPP instances.

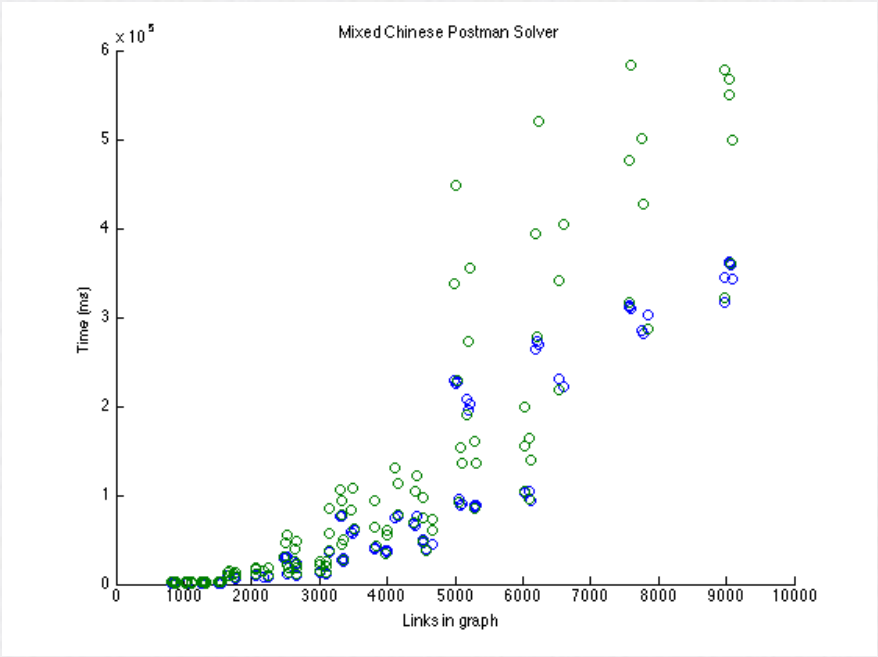
Computational Results





Computational Results





Further Work

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Visualization
- Multi-Vehicle Solvers
- OSM graph ingestion
- More Parsers / Format Conversion
- Decoupled Improvement Procedure Framework
- New Research
- Integrate Faster Open Source Replacements for Common Algorithms
- Suggestions(?)

Schedule

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Required:

- DCPP Solver ✓
- UCPP Solver ✓
- Test Framework ✓
- MCPP Heuristics
  - Frederickson's ✓
  - Shortest Additional Paths ✓
- ~~WPP Heuristics~~ WRPP Heuristics
  - Win's ✓
  - Benavent's ✓
- DRPP Heuristics
  - Christofides' ✓
  - Benavent's - (Alternative Model) ✓
- Final Report (May)

Optional:

- Visualization
- Gurobi Integration
  - DCPP Exact Solver ✓
  - UCPP Exact Solver ✓
  - MCPP Exact Solver ✓
  - WPP Exact Solver ✓
  - WPP Cutting Plane Heuristic ✓
- Performance Optimization (Ongoing)

Deliverables

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

- Code hosted at my personal github at https://github/com/olibear.
  - Test Instances
  - All Code
    - Solvers
    - Graph Architecture
    - Extensible interfaces and abstract classes
    - Common Algorithms
    - Custom Errors
    - Documentation
  - Final Report

Code

Graph Architecture:

- Graph.java
  - DirectedGraph.java
  - UndirectedGraph.java
- Link.java
  - Arc.java
  - Edge.java
  - MultiEdge.java
  - MixedEdge.java
  - WindyEdge.java
- Vertex.java
  - DirectedVertex.java
  - UndirectedVertex.java
  - MixedVertex.java
  - WindyVertex.java

Problems:

- Problem.java
  - UndirectedCPP.java
  - DirectedCPP.java
  - MixedCPP.java
  - WindyCPP.java
  - DirectedRPP.java
  - WindyRPP.java

Solvers:

- SingleVehicleSolver.java
  - DCPPSolver_Edmonds.java
  - DCPPSolver_Gurobi.java
  - UCPPSolver_Edmonds.java
  - UCPPSolver_Gurobi.java
  - MCPPSolver_Gurobi.java
  - MCPPSolver_Frederickson.java
  - MCPPSolver_Yaoyuenyong.java

Code

- WPPSolver_Gurobi.java
- WPPSolver_Gurobi_CuttingPlane.java
- WRPPSolver_Win.java
- WRPPSolver_Benavent_H1.java
- DRPPSolver_Christofides.java

Utilities:

- CommonAlgorithms.java
- Pair.java
- BlossomV.java

Graph I/O:

- GraphGenerator.java
  - DirectedGraphGenerator.java
  - UndirectedGraphGenerator.java
  - MixedGraphGenerator.java
  - WindyGraphGenerator.java
- GraphReader.java

Testing:

- GeneralTestbed.java

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Special Thanks to:

Dr. Bruce Golden

Dr. Angel Corberan

Dr. Zaw Win

Dr. Vincente Campos

Dr. Kriangchai Yaoyuenyong

OAR Lib is available at

https://github.com/Olibear/ArcRoutingLibrary

References

1. Benavent, Enrique, et al. "New heuristic algorithms for the windy rural postman problem." *Computers & operations research* 32.12 (2005): 3111-3128.
2. Campos, V., and J. V. Savall. "A computational study of several heuristics for the DRPP." *Computational Optimization and Applications* 4.1 (1995): 67-77.
3. Christofides, Nicos, et al. "An algorithm for the rural postman problem on a directed graph." *Netflow at Pisa*. Springer Berlin Heidelberg, 1986. 155-166.
4. Dussault, Benjamin, et al. "Plowing with precedence: A variant of the windy postman problem." *Computers & Operations Research* (2012).
5. Edmonds, Jack, and Ellis L. Johnson. "Matching, Euler tours and the Chinese postman." Mathematical programming 5.1 (1973): 88-124.
6. Eiselt, Horst A., Michel Gendreau, and Gilbert Laporte. "Arc routing problems, part I: The Chinese postman problem." *Operations Research* 43.2 (1995): 231-242.
7. Eiselt, Horst A., Michel Gendreau, and Gilbert Laporte. "Arc routing problems, part II: The rural postman problem." *Operations Research* 43.3 (1995): 399-414.
8. Frederickson, Greg N. "Approximation algorithms for some postman problems." Journal of the ACM (JACM) 26.3 (1979): 538-554.
9. Grötschel, Martin, and Zaw Win. "A cutting plane algorithm for the windy postman problem." Mathematical Programming 55.1-3 (1992): 339-358.
10. Thimbleby, Harold. "The directed chinese postman problem." *Software: Practice and Experience* 33.11 (2003): 1081-1096.
11. Win, Zaw. "On the windy postman problem on Eulerian graphs." *Mathematical Programming* 44.1-3 (1989): 97-112.
12. Yaoyuenyong, Kriangchai, Peerayuth Charnsethikul, and Vira Chankong. "A heuristic algorithm for the mixed Chinese postman problem." *Optimization and Engineering* 3.2 (2002): 157-187.

Backup Slides

Background

- Successful software systems are platforms, not projects.
- Extensibility / Flexibility of code should be a pre-requisite, not a feature.
- Standards benefit all developers, especially in the research community.

OAR Lib

1. **Background**
2. What is OAR Lib?
3. Why Java?
4. What's the Problem?
5. Applications
6. What's In the Library?
7. Example
8. Further Work
9. Special Thanks
10. References

Computational Results

UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

Graph	HashMap<Integer, Vertex> internalVertexMap	HashMap<Integer,Vertex> globalVertexMap
	HashMap<Integer, Edge> internalEdgeMap	HashMap<Integer, Edge> globalEdgeMap
Vertex	int mID	int matchID
	int globalID	int mDemand
	boolean demandSet	
Link	int mID	int matchId
	int globalID	int mCapacity
	int mCost	boolean isDirected
	boolean capacitySet	Pair<Vertex> endpoints