Multi-Period Street Scheduling and Sweeping

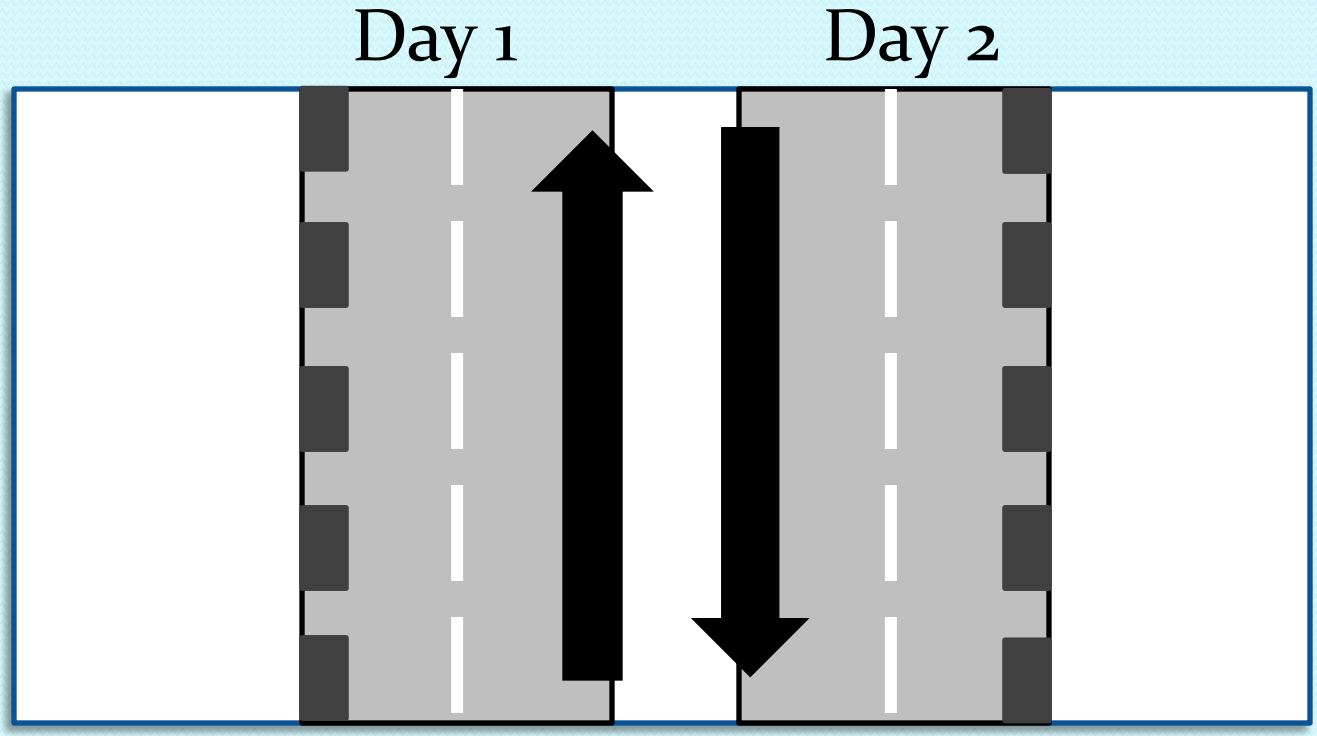
Benjamin Dussault
Department of Applied Mathematics and Scientific Computation – University of Maryland
Carmine Cerrone
Department of Mathematics and Computer Science – University of Salerno
Bruce L. Golden
R.H. Smith School of Business - University of Maryland
Edward Wasil
Kogod School of Business – American University

ICS Santa Fe
January 2013

Introduction

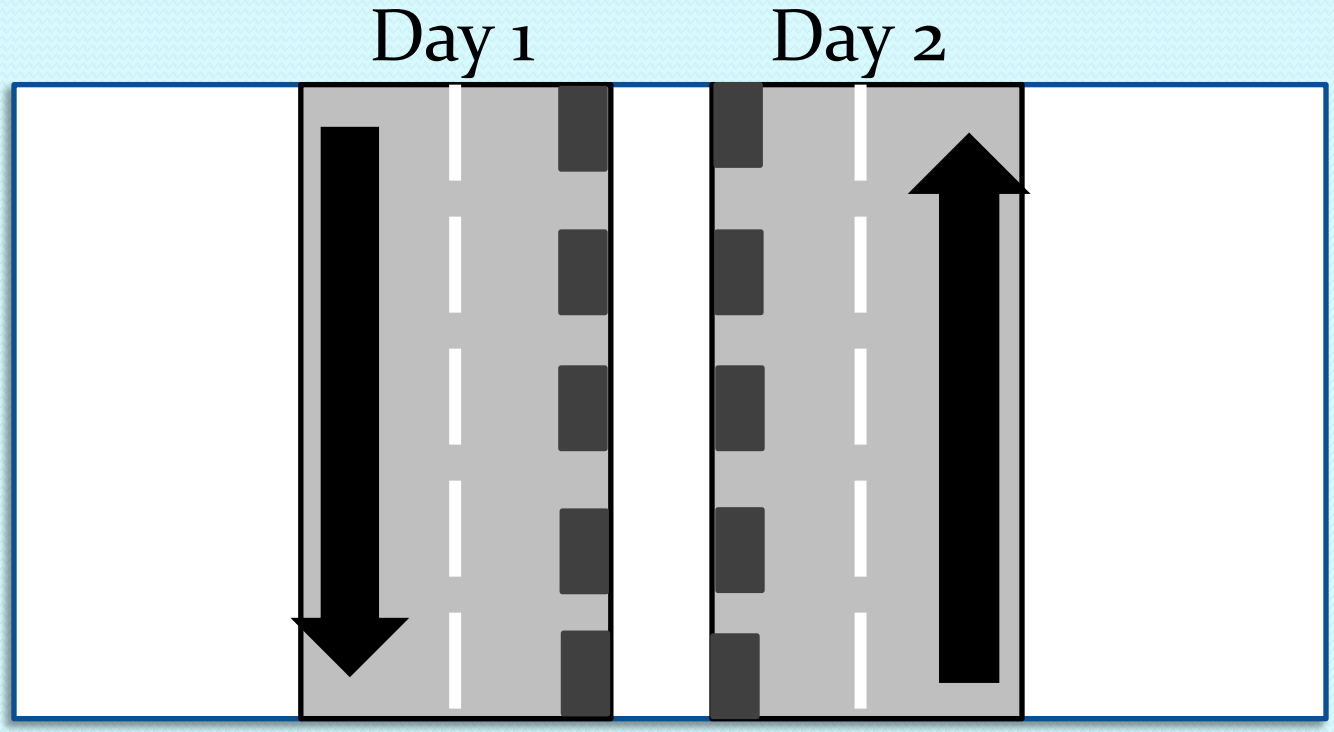
- Washington, DC wants to have a subset of its streets swept over two days
- A street sweeper cannot sweep a side of a street if there are cars parked on that side
- There are parking restrictions for each street segment that are enforced by parking signs and they dictate parking availability and hence sweeping ability
- A common parking constraint is the requirement that parking be available on one side of a street at all times

Motivating Parking Constraint



Day 1 – Park on the left, sweep on the right
Day 2 – Park on the right, sweep on the left

Motivating Parking Constraint



Day 1 – Park on the right, sweep on the left
Day 2 – Park on the left, sweep on the right

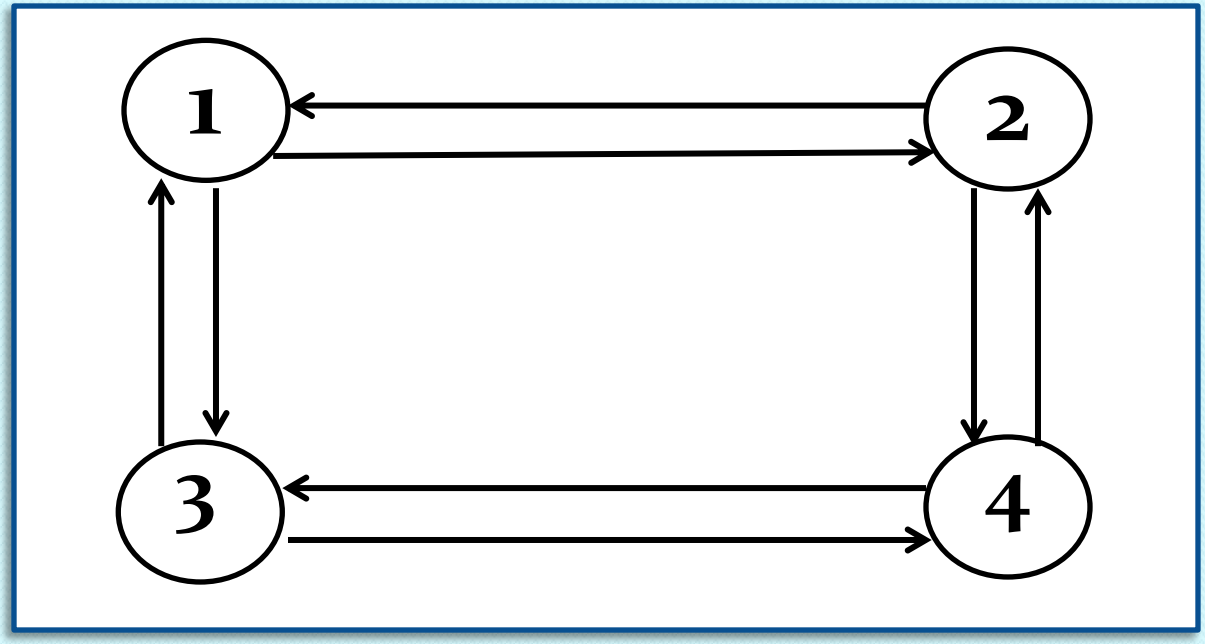
Parking Constraints

- Some streets have no parking at all
  - Here we can sweep either side of the street at any time
  - There are no parking signs in this case

Variant 1

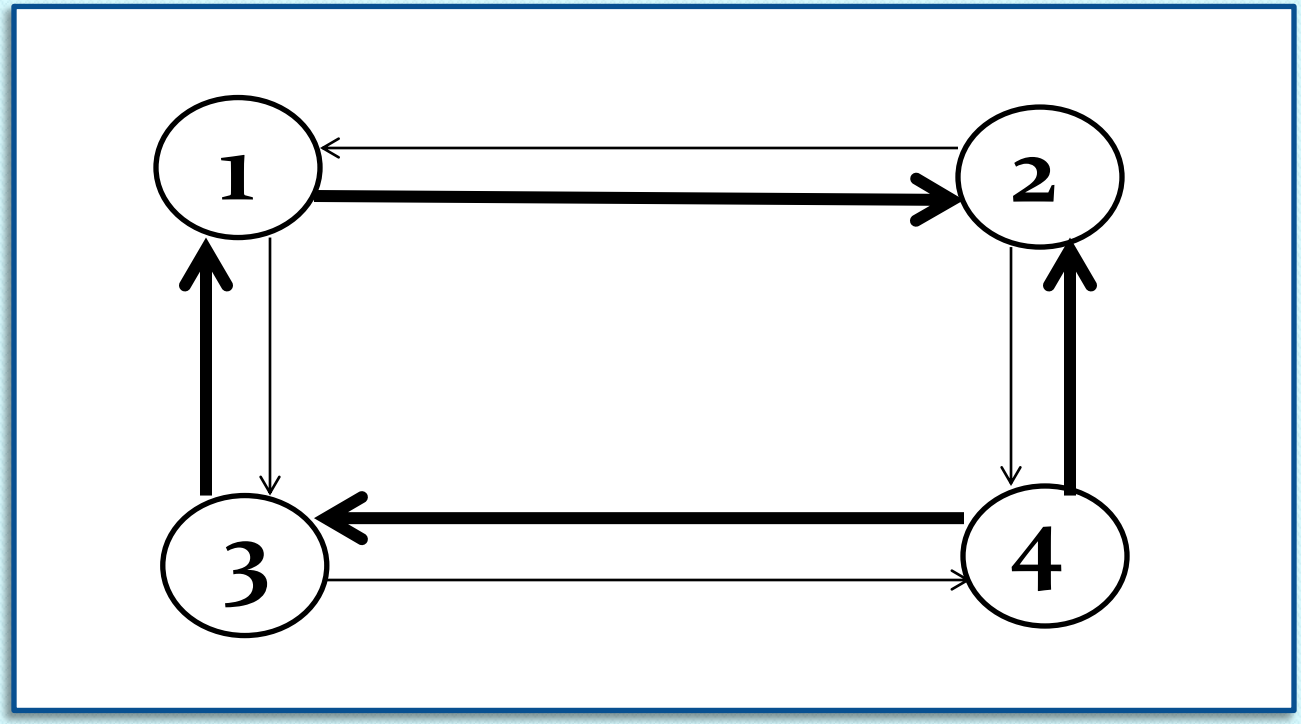
- Suppose the city decides to redo all parking signs (reschedule No Parking) while still obeying parking constraints
- How should the city redo the signs to minimize the length of the path traveled by the street sweeper?
- If we redo the signs, we can decrease the distance traveled by the sweeper

Variant 1 - Example



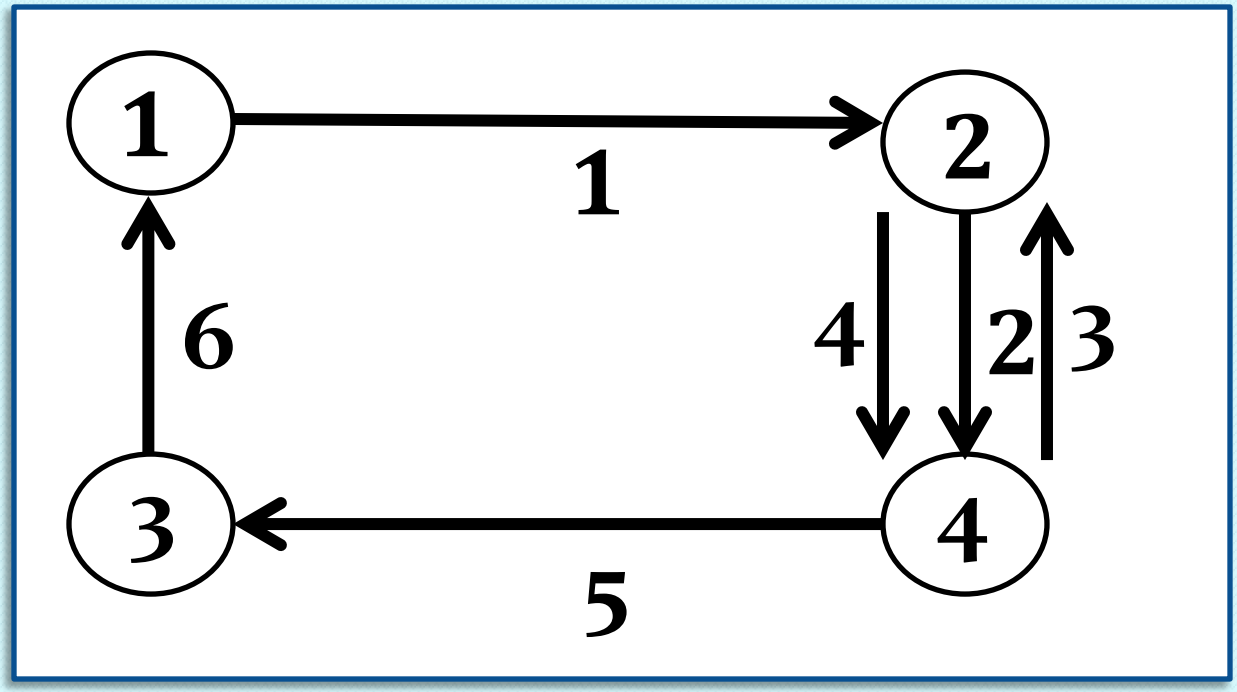
All street sides must be swept and available parking is required on one side

Variant 1 - Example



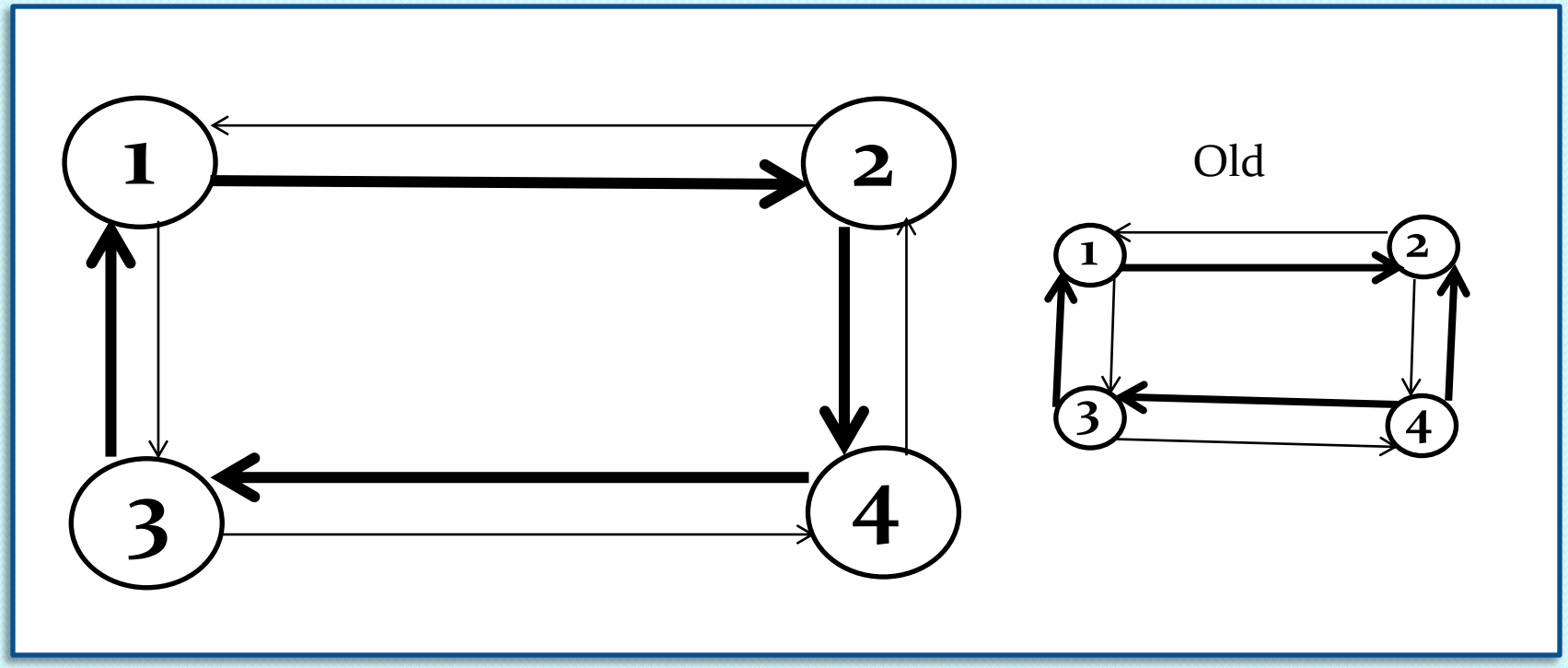
Bold edges are street sides that must be swept on even days

Variant 1 - Example



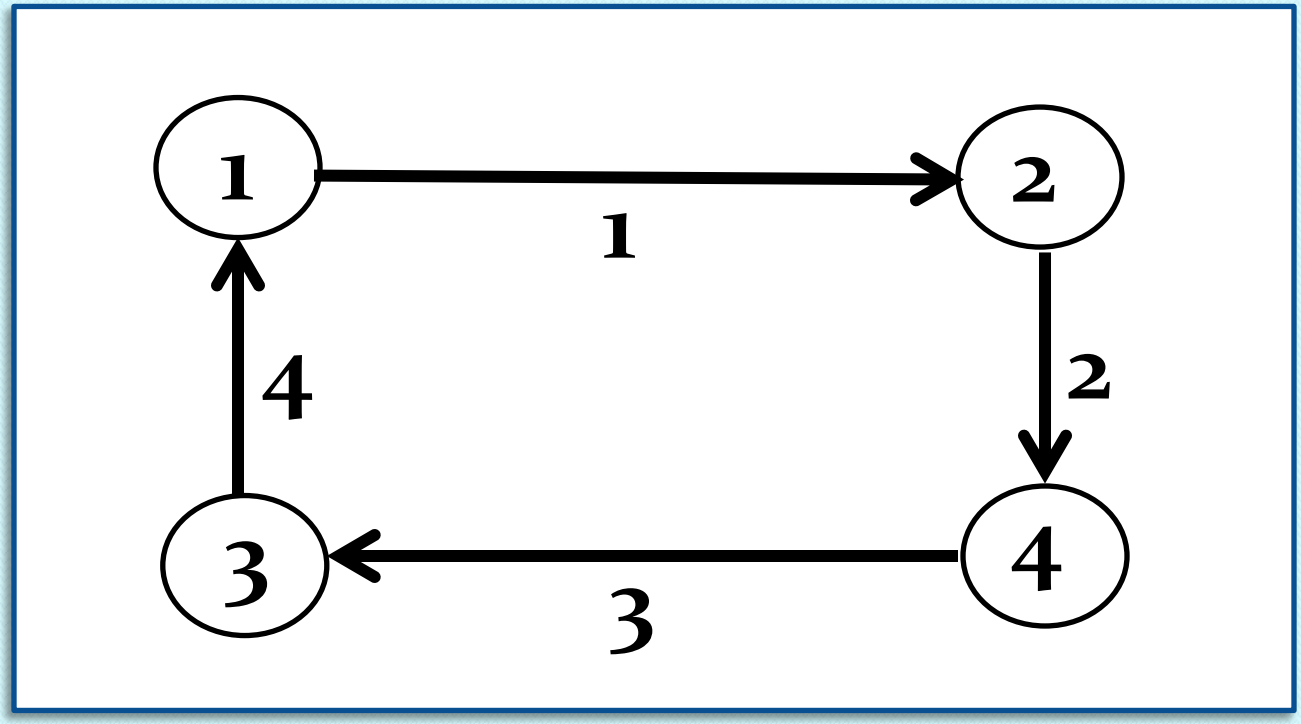
Schedule leads to an undesirable sweeping route on even days

Variant 1 - Example



New schedule: bold edges are street sides that must be swept on even days

Variant 1 - Solved



Optimal sweeping route on even days (on odd days the route is the reverse)

Variant 2

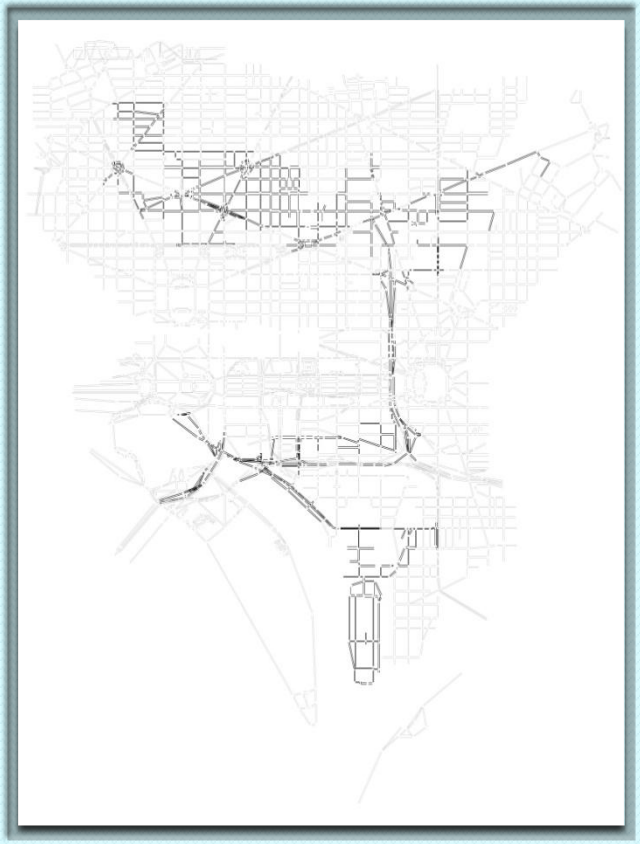
- The city has street signs already in place
- How can the city change a small number of street signs to allow for a schedule that substantially decreases the distance traveled by the sweeper?
- There are costs associated with changing existing street signs (e.g., the actual cost of new signs and sign replacement, the confusion caused by new signs)

Problem Statement

- The city is represented by a directed graph G
- G is strongly connected
- Each street has two edges representing the two sides of the street
  - The edges go in opposite directions if the street is two way
  - Same direction if the street is one way
- The street sweeper is responsible for some subset of the streets (not necessarily connected)

Washington, DC Example

Bold edges must be swept



Problem Statement

- Each street (and associated edge pairs) has one of the following constraints, determined ahead of time:
  - Neither side is swept during the two days. Travel along either side will result in deadheading
  - One side does not need to be swept and the other does. Travel along the former will result in deadheading, and the latter may be swept on either day
  - Both sides need to be swept, but cannot be swept on the same day. This is the situation where parking is required to be available on one side of the street at all times
  - Both sides need to be swept, but they may be swept on the same day

A Feasible Schedule

- A feasible schedule assigns a sweeping day to each edge that requires sweeping, in a way that obeys the parking constraints
- A schedule completely determines the parking signs
- We want to determine a feasible schedule that produces the Eulerian cycle with the smallest deadhead distance (by solving the DRPP with a simple heuristic)
- This combined problem is NP-hard since it is a generalization of the DRPP

Solution Methodology

- We use a genetic algorithm (GA) that acts on a population of feasible schedules

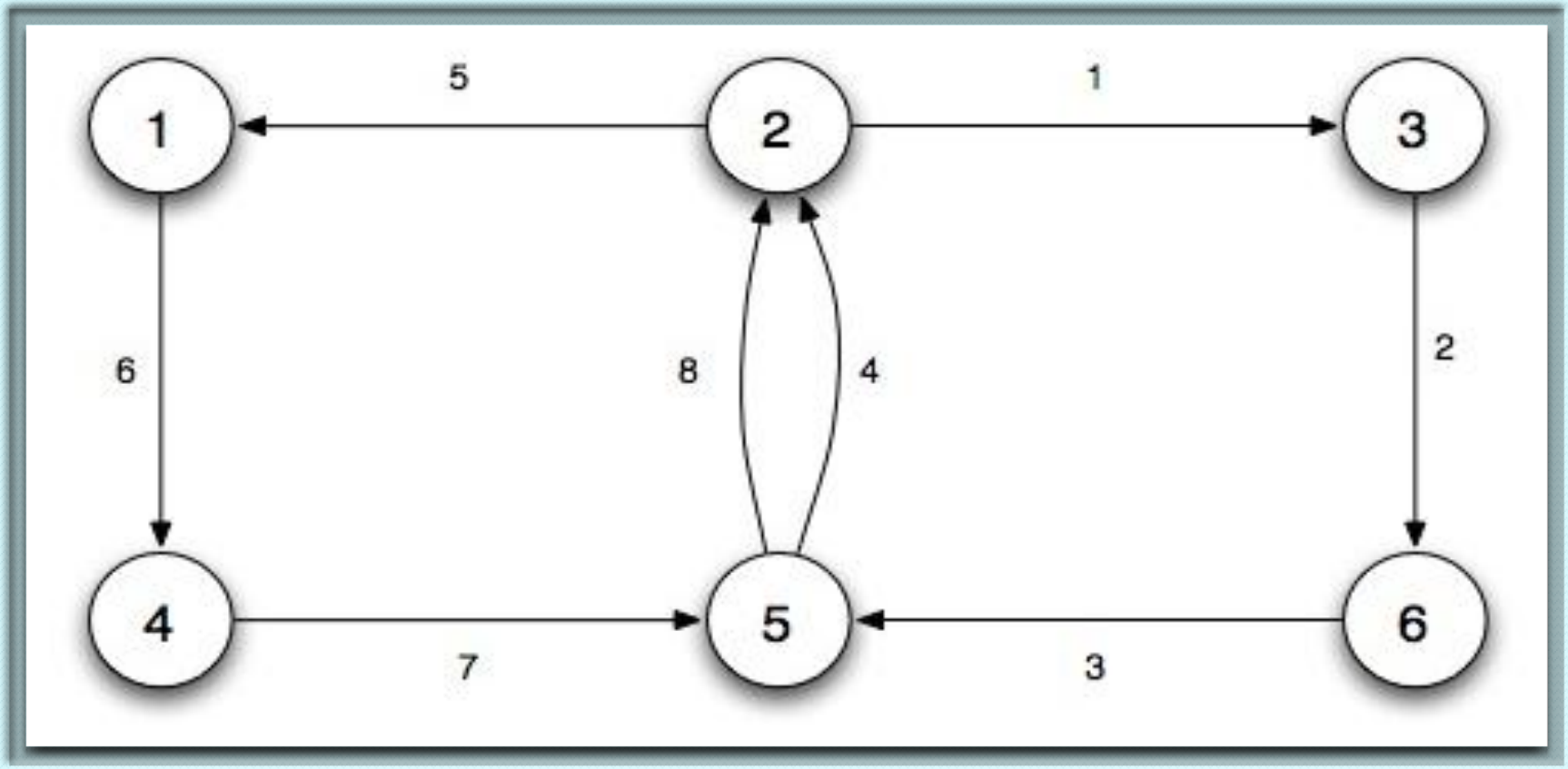
- We use a simple and fast heuristic for the DRPP to produce a cycle whose length serves as a fitness function in the GA

GA Overview

- Chromosomes are schedules
- Begin with an initial population of randomly constructed schedules
- Iterate
  - Breed each schedule with another schedule
  - Mutate some solutions
- Repeat until no new best solution is found for *N* iterations

Breeding Process

- The induced Eulerian cycle of a schedule is highly sensitive to the schedule
- To construct a good breeding algorithm, we note that most Eulerian cycles can be decomposed into sub-cycles
- A good Eulerian cycle will have low-deadhead sub-cycles
- Our breeding algorithm tries to swap sub-cycles between schedules

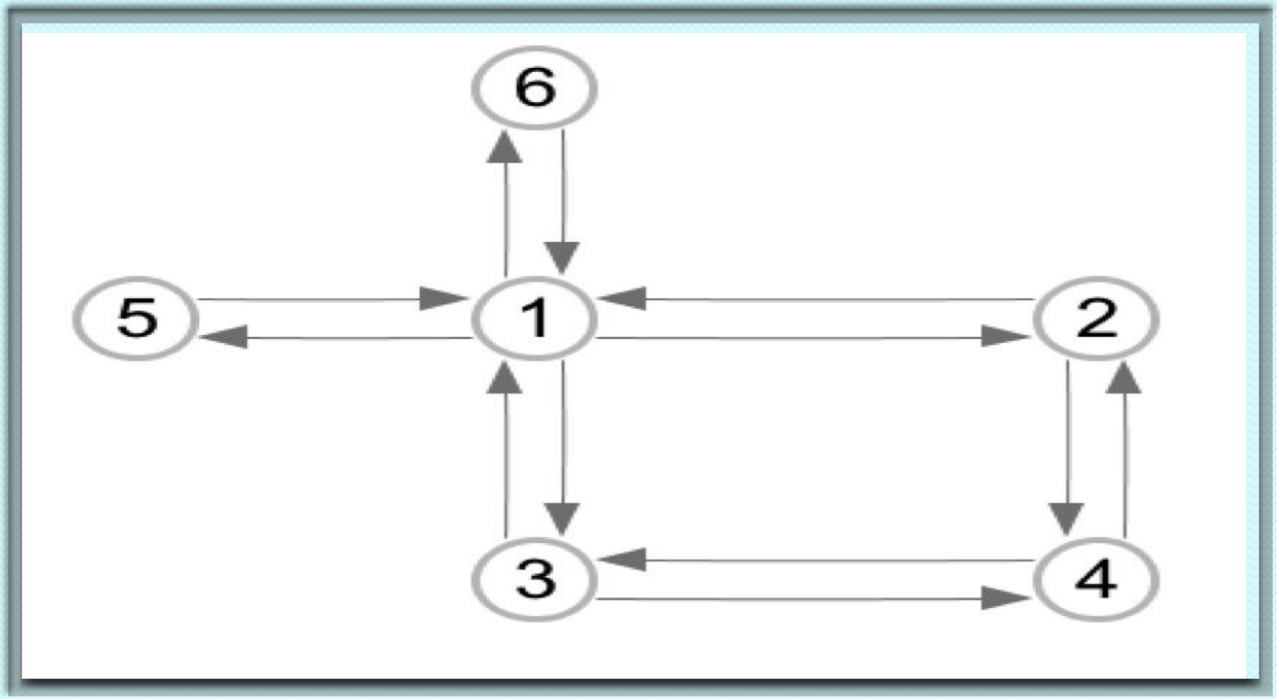


The cycle above can be decomposed into two cycles given by edges 1 to 4 and 5 to 8

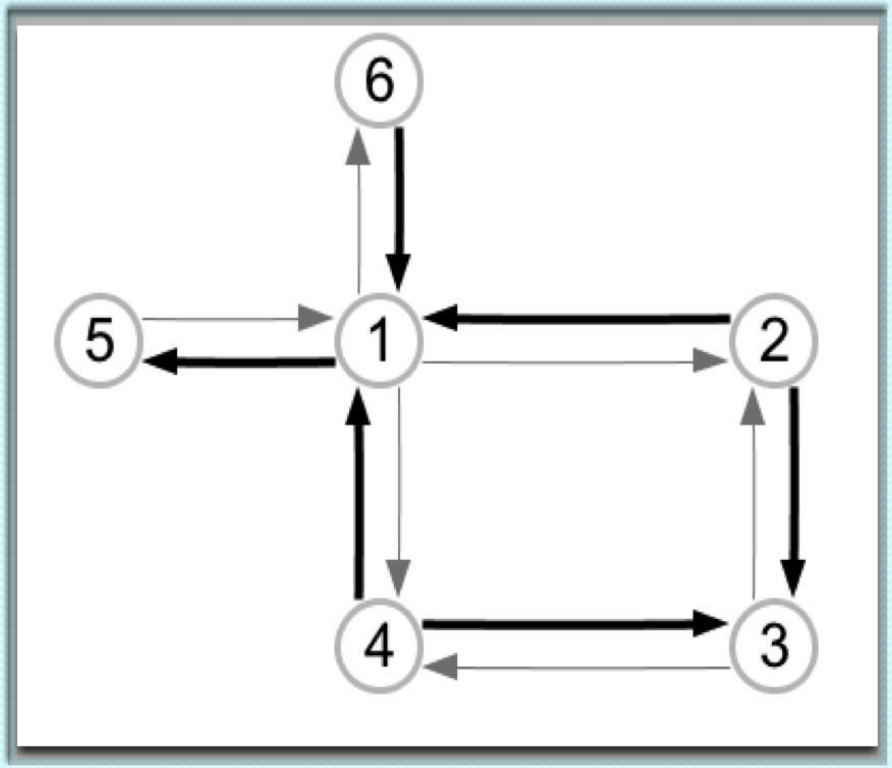
Breeding Process - continued

- A good schedule is one that yields low-deadhead cycles
- Our breeding algorithm determines a random sub-cycle allowed by one schedule on a given day and inserts it into a second schedule on the same day

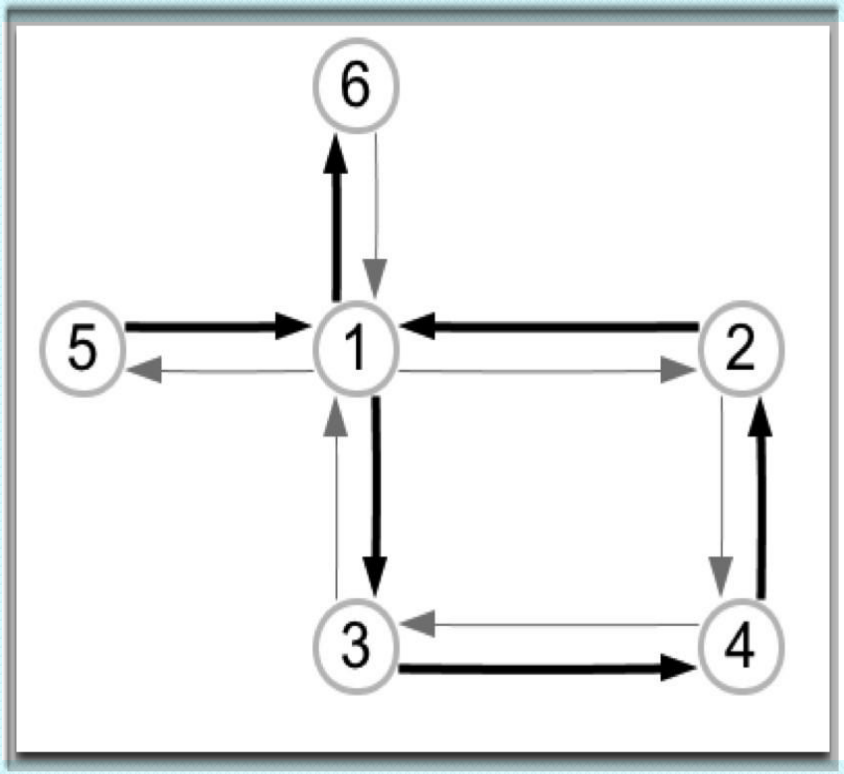
An Example of Breeding



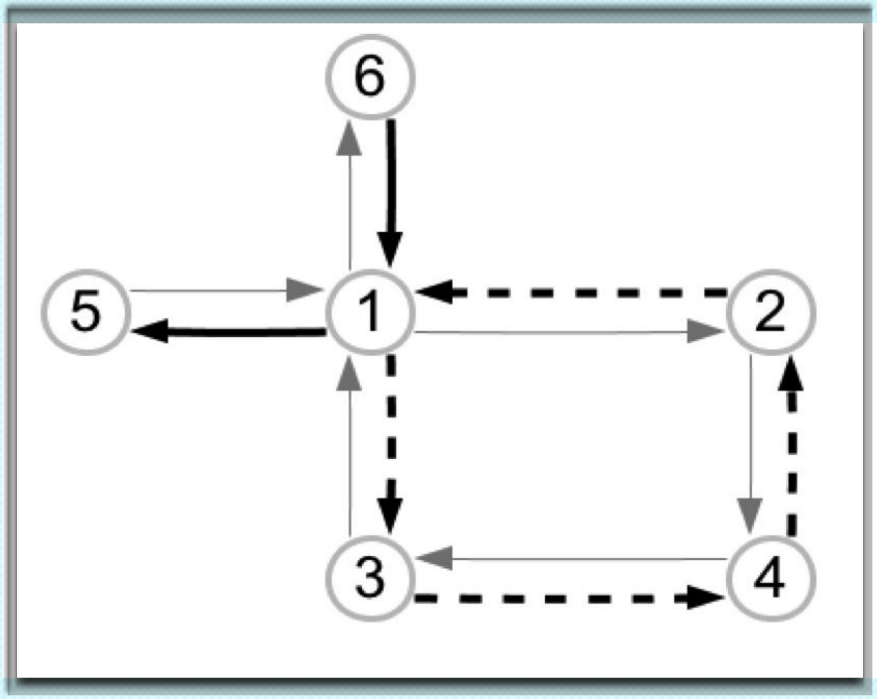
Portion of city graph – each street must have available parking on one side each day



Schedule 1 on even days - note substantial deadhead required



Schedule 2 on even days – note the low-deadhead cycle 1-3-4-2-1



The good structure of schedule 2 is inserted into schedule 1

Results for Variant 1

- We compare the GA to a CPLEX solution of an integer program (IP) of the problem and to a naive local search procedure (LS)
  - LS changes one element of the schedule at a time and keeps improvements
- Two types of test instances: 1) randomly generated to look like a street network and 2) obtained from actual data of Washington, DC
  - Number of nodes in random instances are 25, 50, 100, and 225 (20, 20, 10, and 10 instances, respectively)

Variant 1 Results

- CPLEX applied to 40 small instances (25 and 50 nodes)
  - CPLEX hits a time limit of 7200 seconds on 25 instances
  - Fails to obtain a feasible solution to 2 instances
  - Produces optimal solution to 15 instances

Variant 1 Results - continued

- GA and LS accuracy on 40 small instances
  - GA obtains good solutions with respect to CPLEX when CPLEX is able to obtain a solution, on average 0.18% worse
  - GA outperforms LS by an average of 1.1% over all problems

Variant 1 Results

- CPLEX running times on small instances
  - Average of 548 seconds, not including instances where CPLEX hits the time limit
- GA: 5.4 seconds on average
- LS: 1.2 seconds on average

Variant 1 Results - continued

- CPLEX applied to 20 large instances (100 and 225 nodes)
  - Hits time limit of 7200 seconds on 19 instances
  - Fails to obtain a feasible solution on 18 instances
  - CPLEX is not a viable way of solving large instances

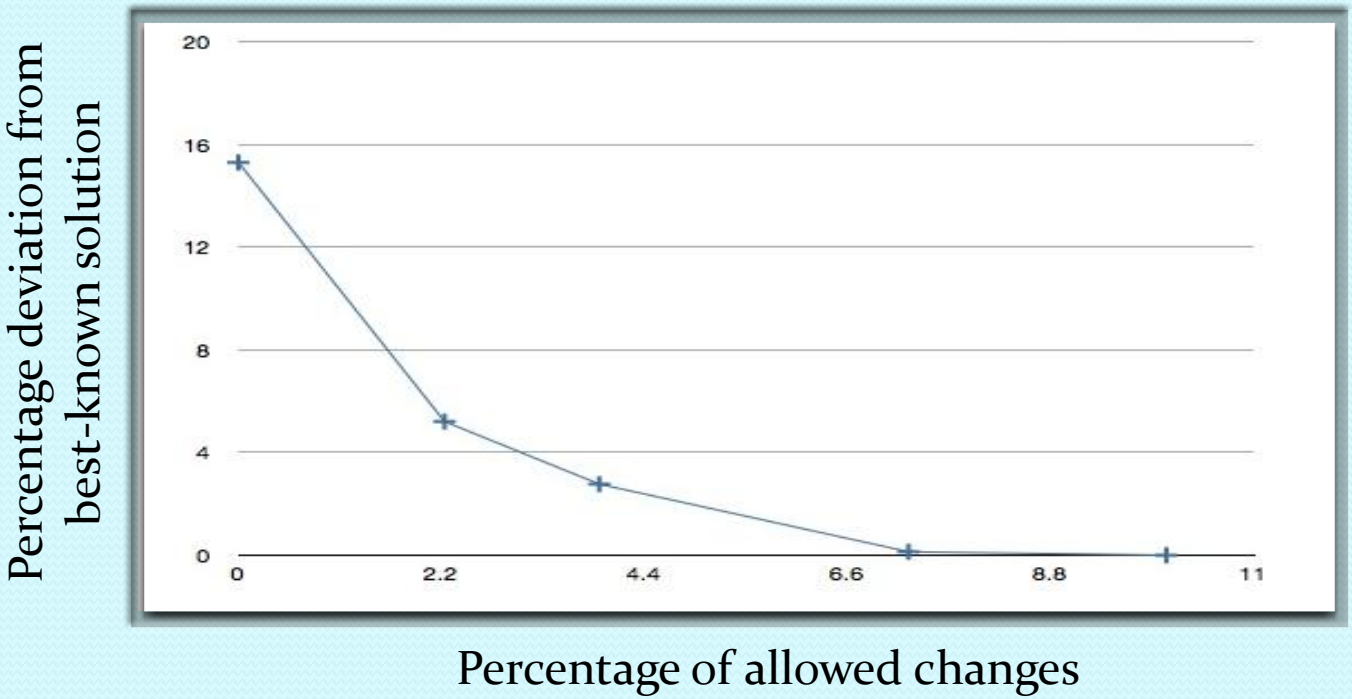
Variant 1 Results - continued

- GA and LS on 20 large instances
  - GA outperforms LS by an average of 0.37% with respect to route length
  - GA outperforms LS with respect to running time
    - 44 vs. 34.4 seconds for 100 nodes
    - 708.7 vs. 3006.2 seconds for 225 nodes

Variant 2 Results

- Iteratively increase the number of allowed street sign changes (n) and compare the resulting fitness values
- Measure
  - Deviation from best solution as a function of n

Variant 2 Results



Percentage deviation from best-known solution vs. percentage of allowed street sign changes

Variant 2 Results - continued

- Convergence of the fitness function in Variant 2 to best-known result obtained from Variant 1 occurs in nearly all test instances
- Average deviation
  - 0.004% when 12% of the signs are changed
  - 0.927% when 7% of the signs are changed
- One does not lose solution quality when beginning with an initial solution and then modifying it

Conclusions

- We described two new street scheduling and sweeping problems encountered in Washington, DC
- We developed a novel genetic algorithm
  - Accurate
  - Fast running times
- Interesting managerial results for Variant 2