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 Golden

Robert H. Smith School of Business - University of Maryland

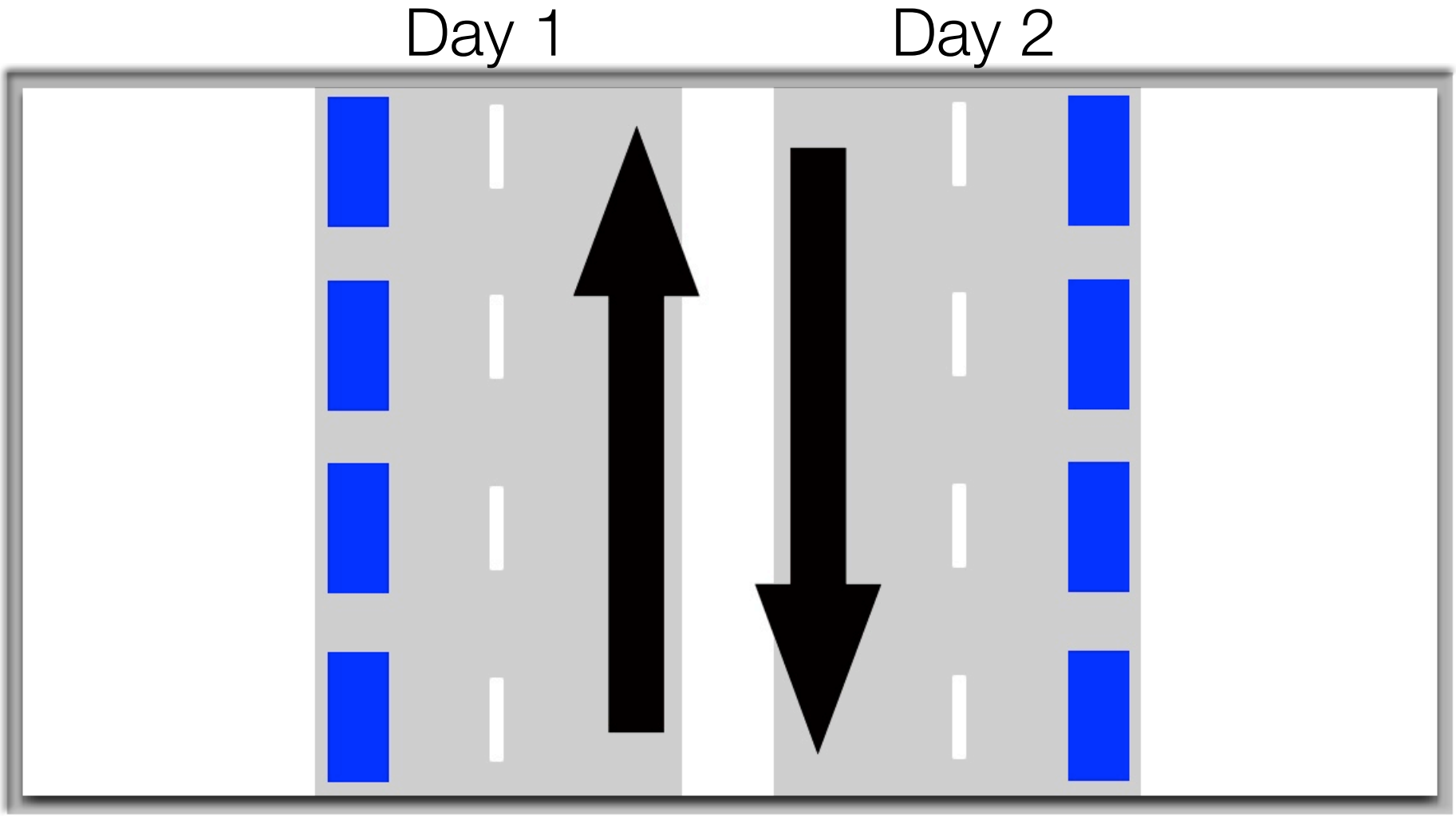
Edward Wasil

Kogod School of Business - American University

Tristan VII
June 2010

Introduction

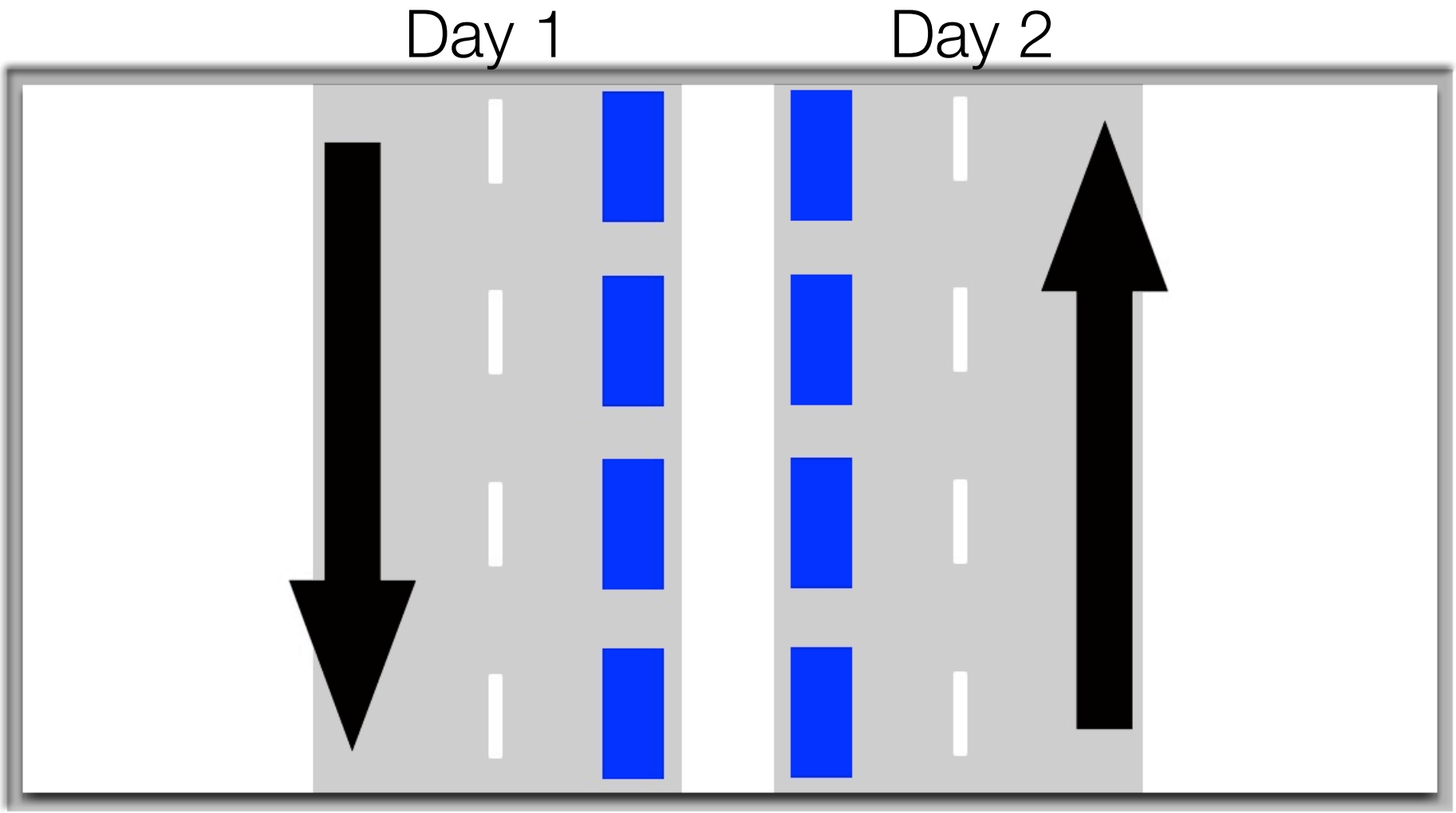
- Washington D.C. wants to sweep a subset of its streets over two days
- A street sweeper cannot sweep a side of a street if there are cars parked on that side
- There are parking restrictions for each street segment that are enforced by parking signs and dictate parking availability and hence sweeping ability
- A common parking constraint (and central motivator) 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



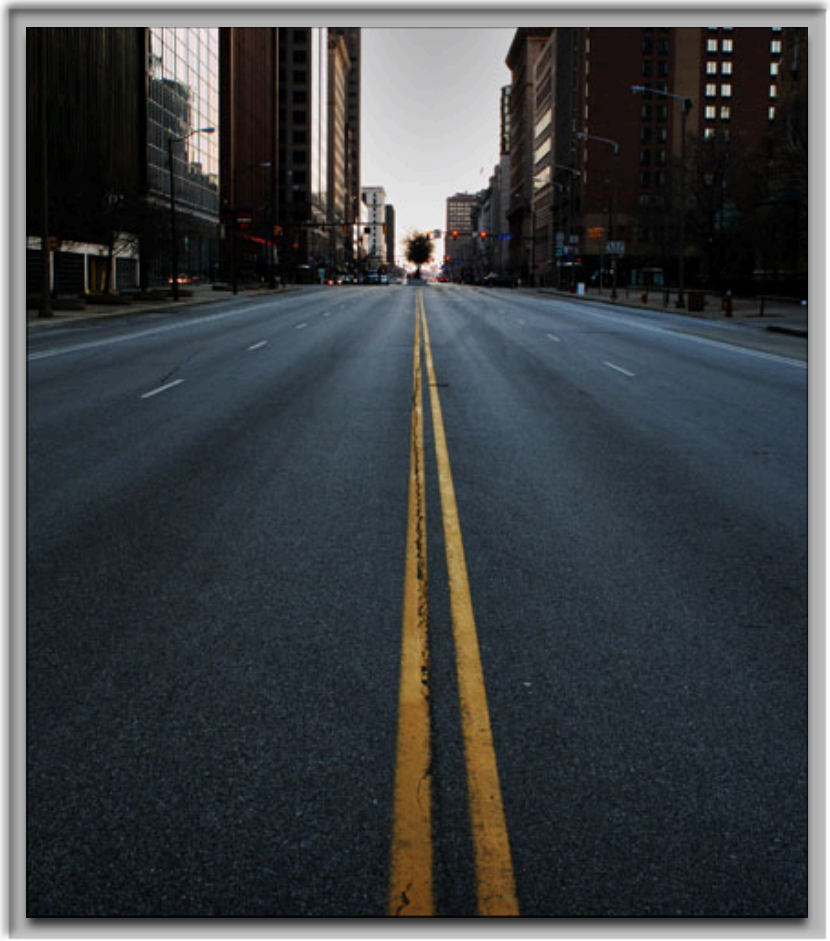
Motivating Parking Constraint

Day 1 - Park on the right, sweep on the left

Day 2 - Park on the left, sweep on the right

Introduction

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

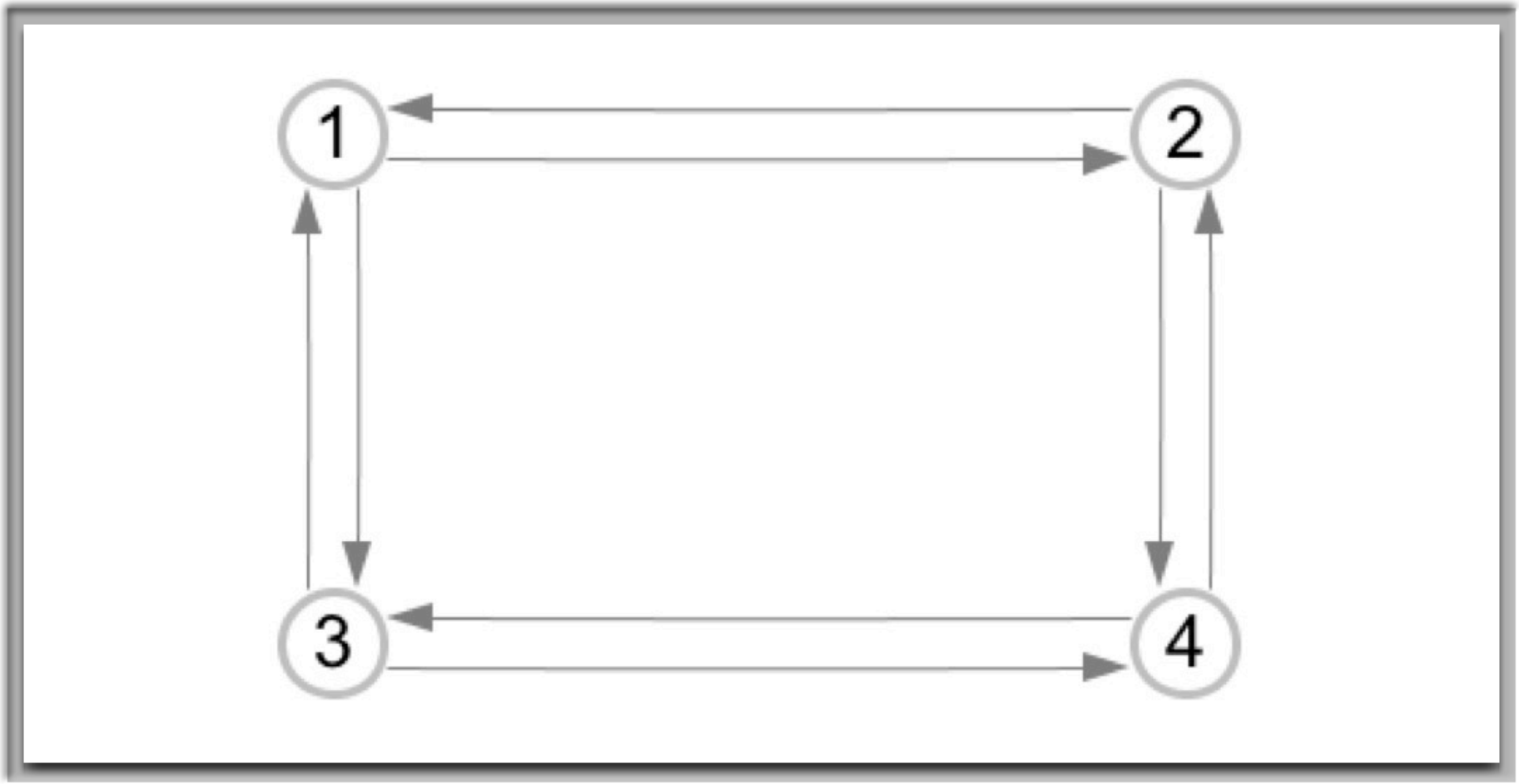


Literature Review

- Bodin and Kursh (1978, 1979) study street sweeping in New York City
  - Deal with time-window parking constraints
- Street sweeping can be modeled as a Directed Rural Postman Problem (DRPP) (Christofides et al. 1986)
  - Our problem is more complicated; the decision variables of interest are choosing appropriate street signs over choosing routing
- Eglese and Murdock (1991) consider bidirectional street sweeping

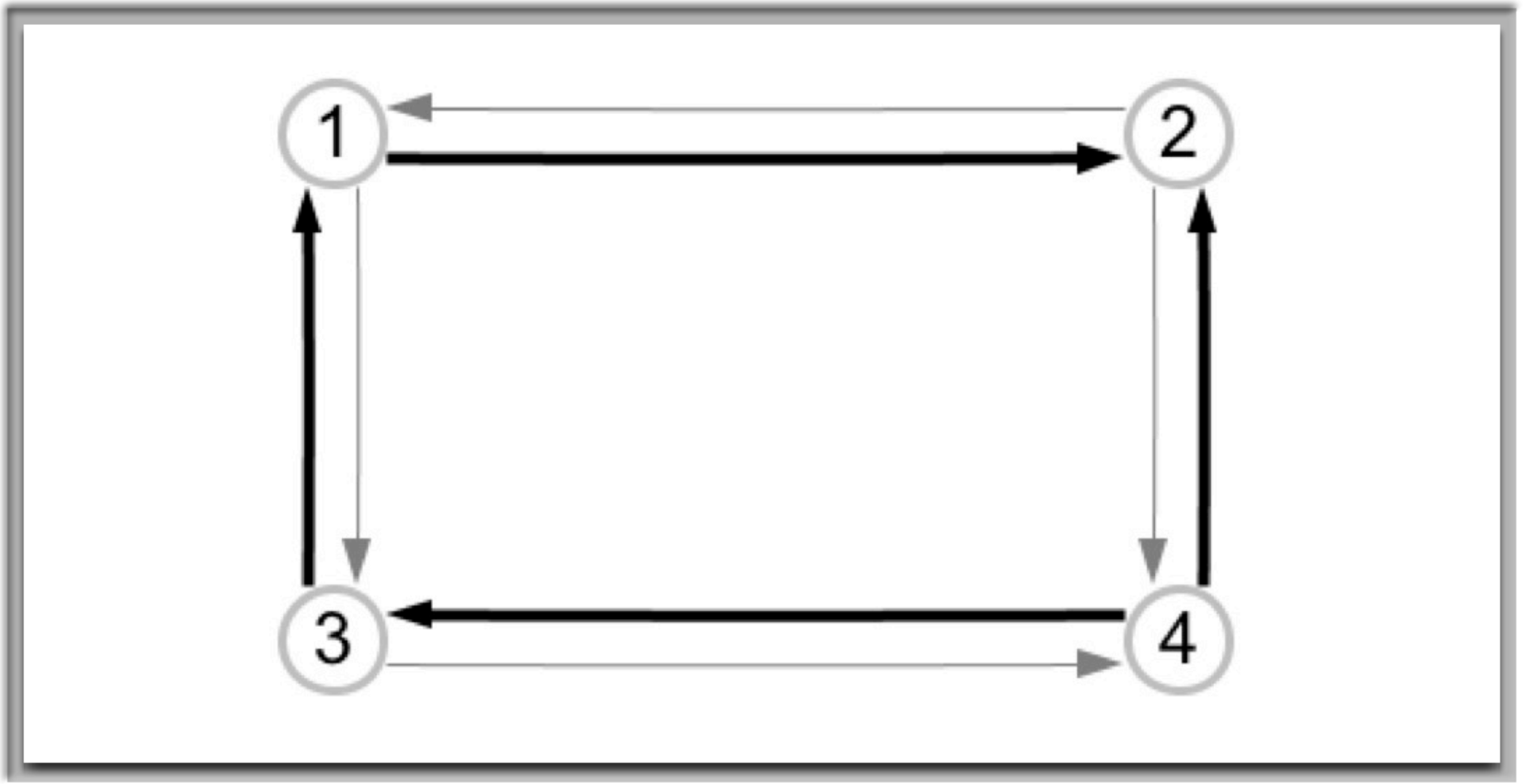
Problem 1

- Suppose the city decides to redo all parking signs while still obeying parking constraints
- How should the city redo the signs to minimize the length of the path traveled by the street sweeper?
- Problem 1: Solve for the optimal routing given existing parking constraints
- Relaxing the signs can only decrease the distance traveled by the sweeper



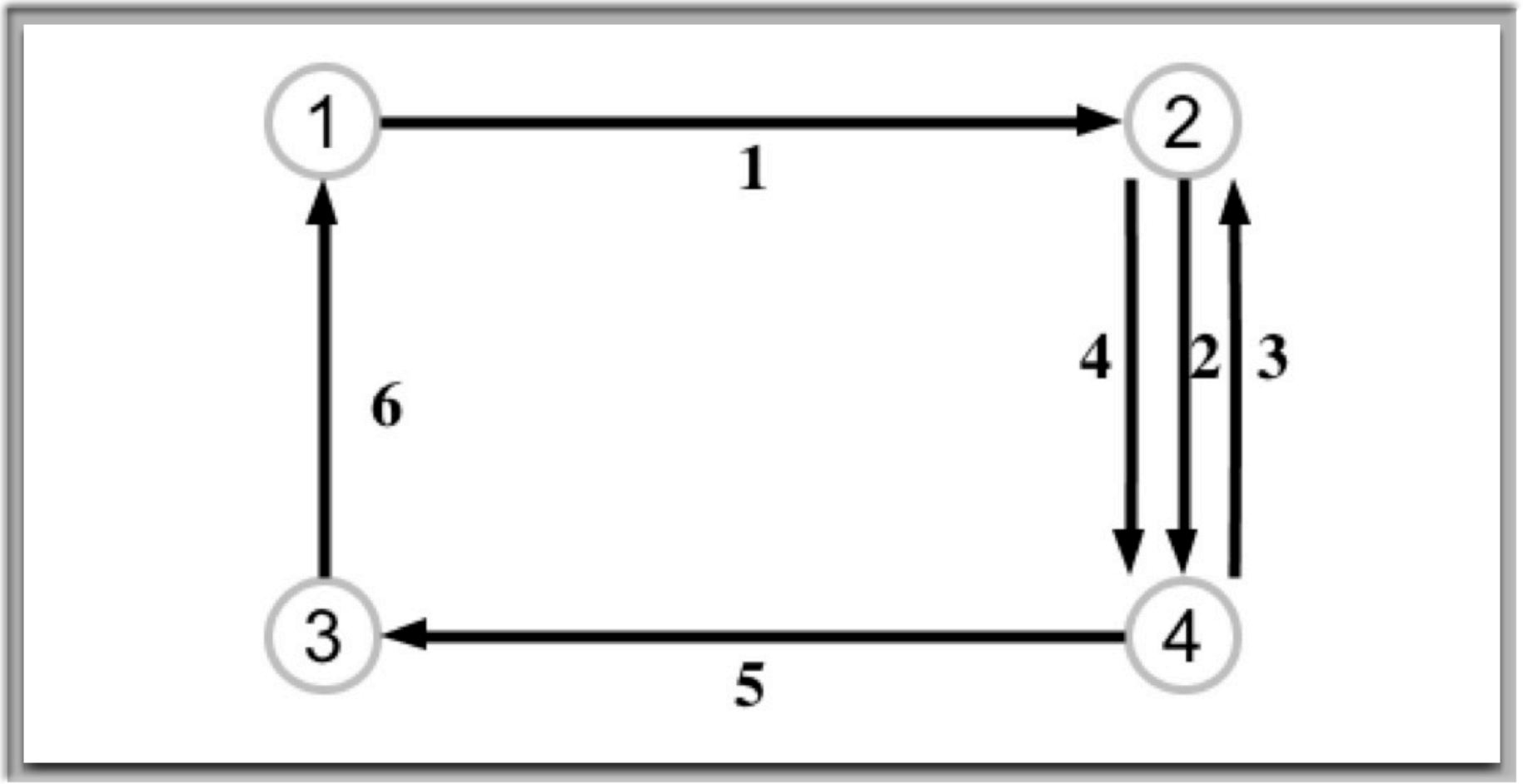
Problem 1 - Example

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



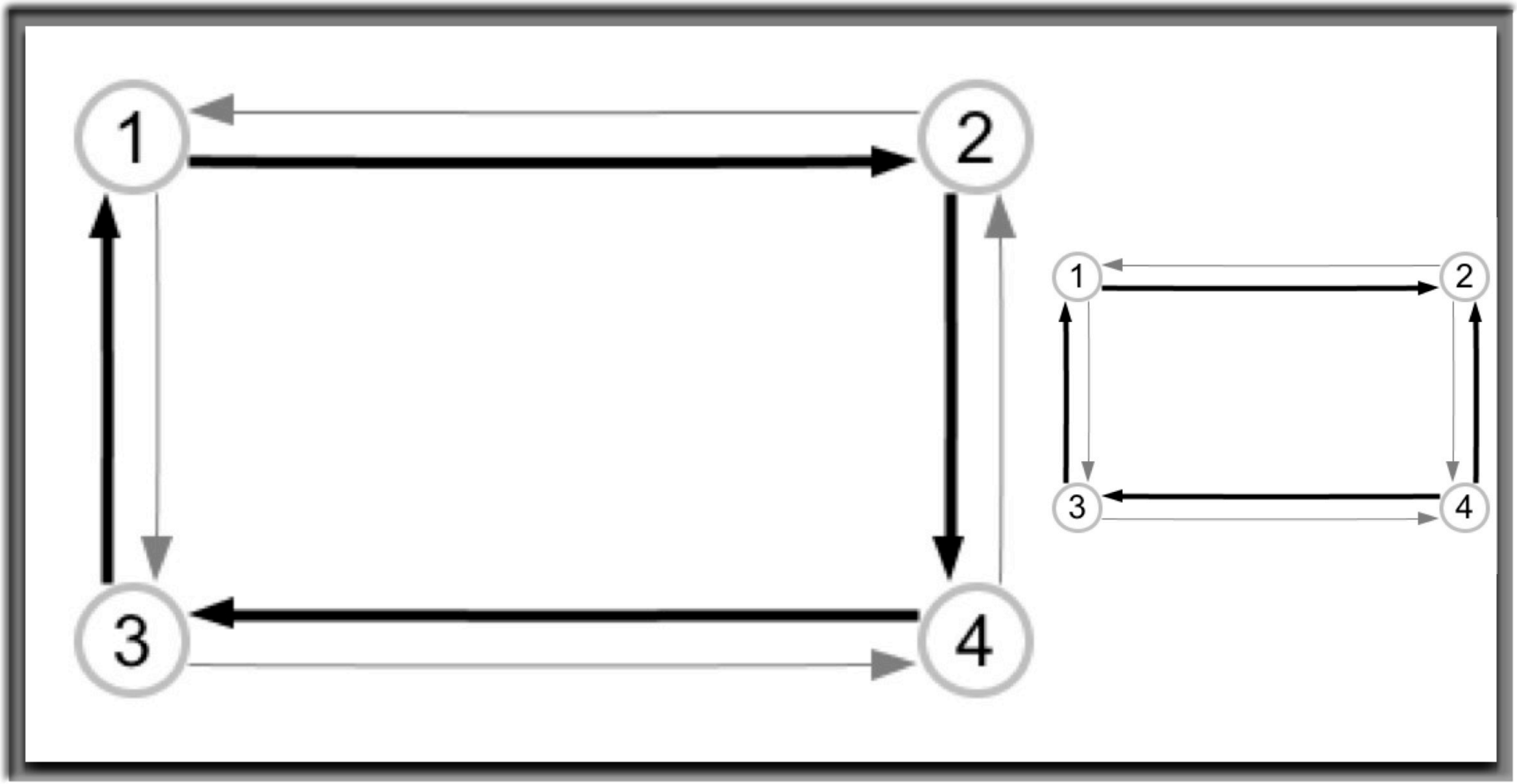
Problem 1 - Example

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



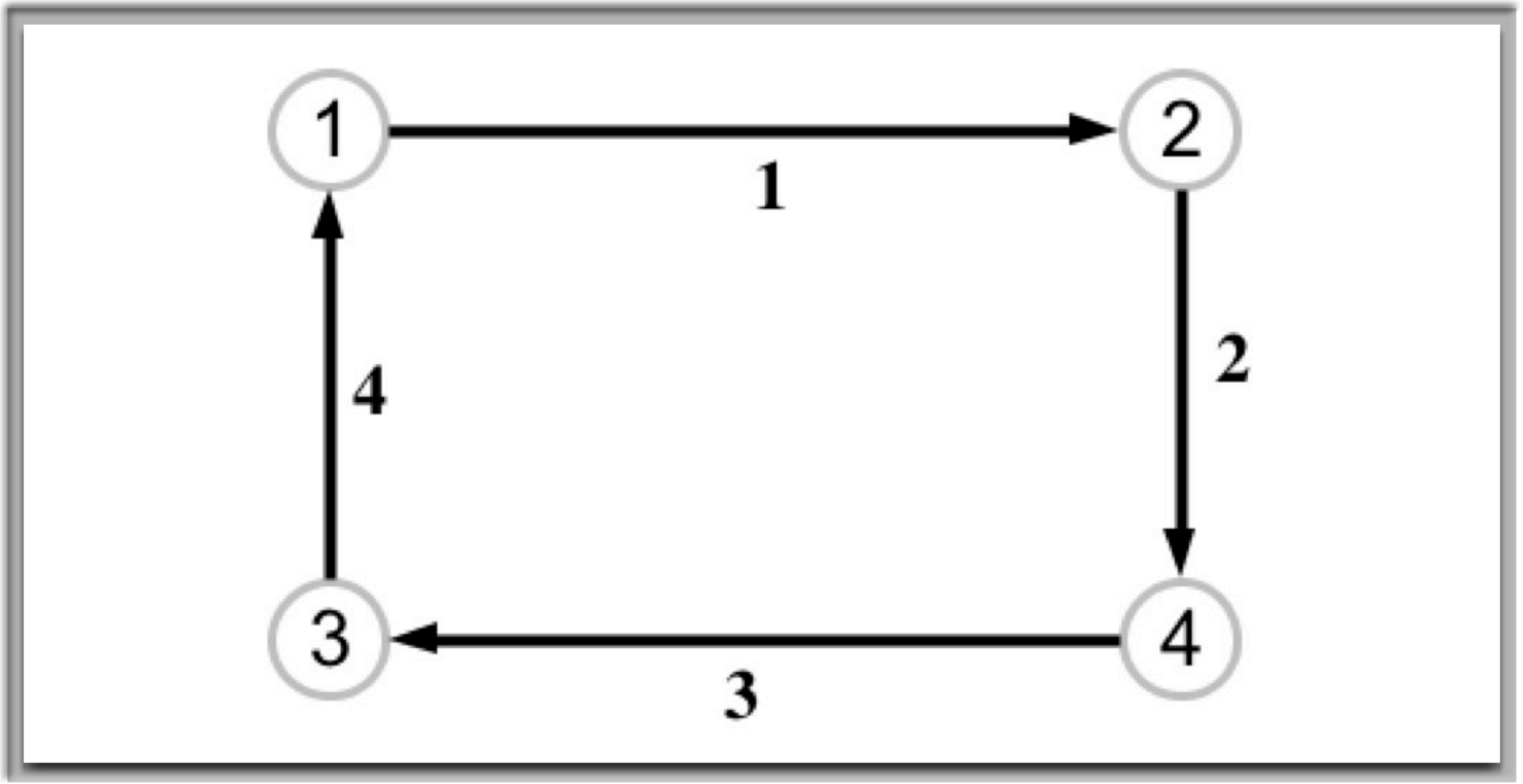
Problem 1 - Example

Schedule induces undesirable sweeping route on even days



Problem 1 - Example

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



Problem 1 - Example

Induces optimal sweeping route on even days (and odd days as well)

Problem 2

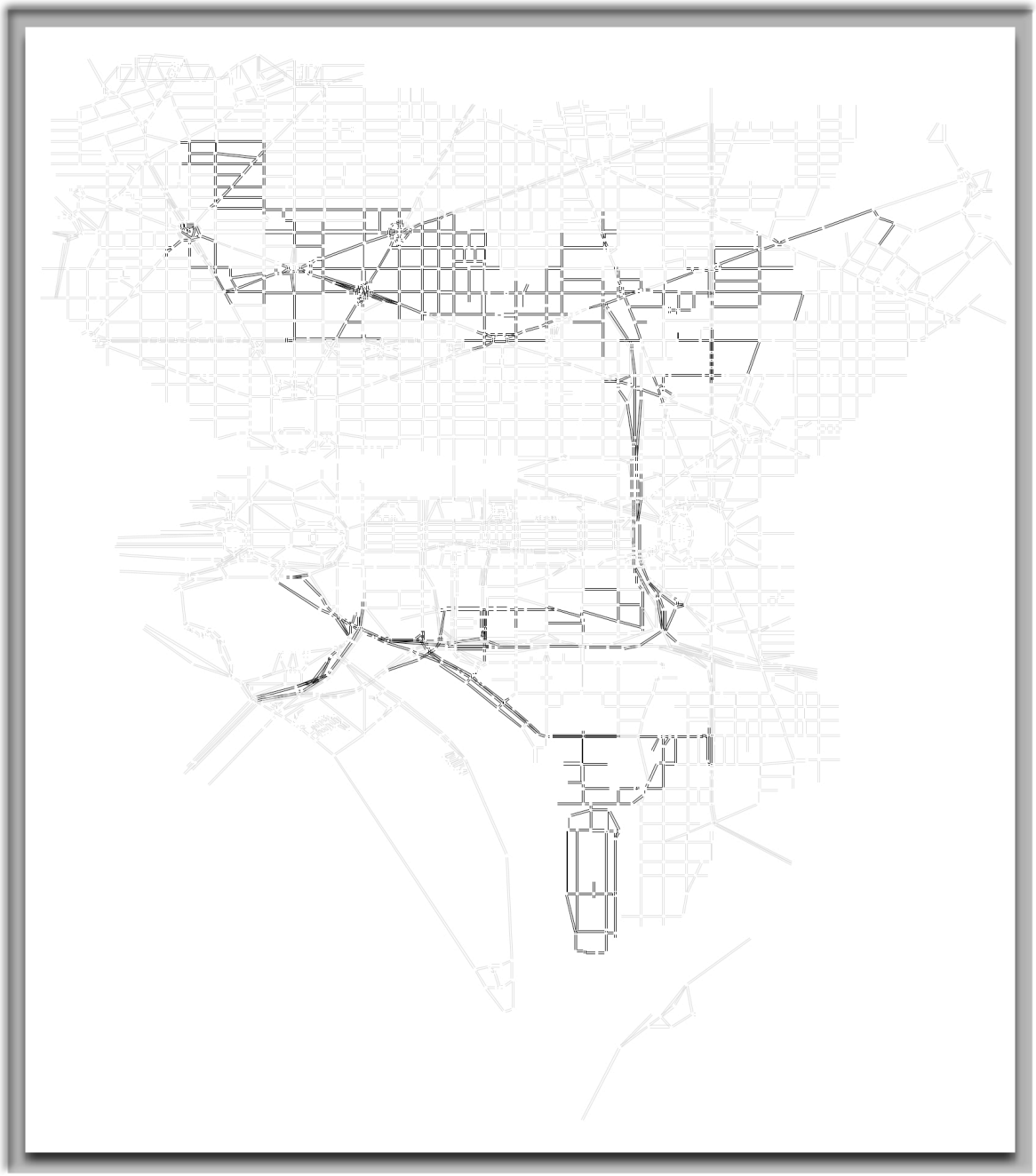
- The city has street signs already in place
- How can the city change a minimal number of street signs to allow for a schedule that maximally decreases the distance traveled by the sweeper?
- There are costs associated with changing existing street signs (e.g., actual cost for new signs, residents confused by new signs)

Problem Statement

- City is represented by a directed graph G
- G is strongly connected
- Each street has by 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
- Street sweeper is responsible for some subset of the streets (not necessarily connected)

Example

Washington DC - bolded edges must be swept



Problem Statement

- Each street (and associated edge pairs) have the following constraints determined ahead of time:
  - Both sides are not 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. This allows for the possibility of both sides to be swept on the same day

Problem Statement

- A feasible schedule assigns a sweeping day to each edge that requires sweeping in a way that obeys the pre-assigned parking constraints
- A schedule completely determines the parking signs so we only concern ourselves with schedules
- We seek to determine a feasible schedule that produces the Eulerian cycle with the smallest deadhead (by solving the DRPP with a simple heuristic)
- This problem is NP-hard since it is a generalization of the DRPP

Solution Methodology

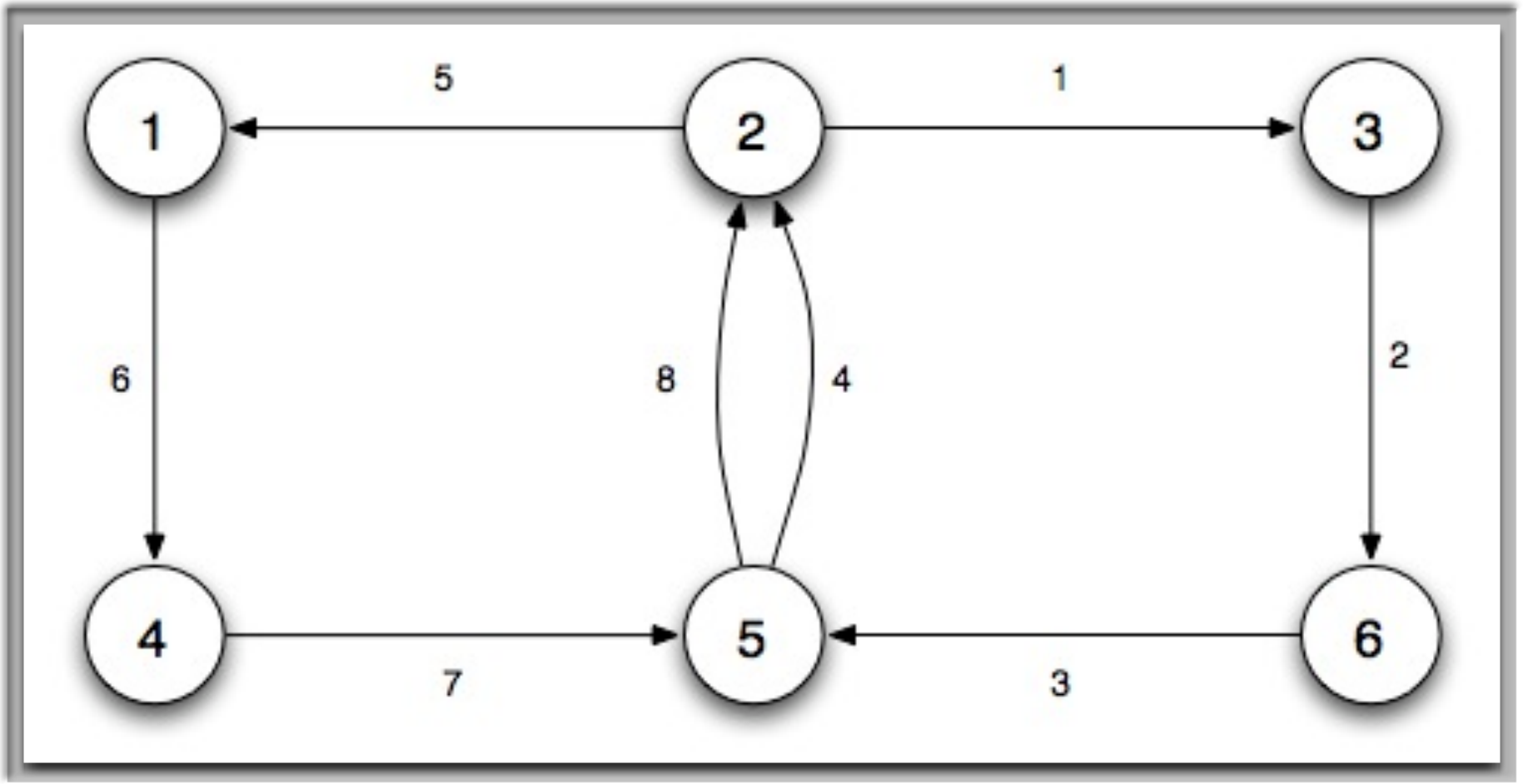
- We use a genetic algorithm (GA) that acts on a population of feasible schedules
- We use a simple and fast heuristic for the DRPP to produce a cycle whose length serves as a fitness function

GA Overview

- Chromosomes are schedules
- Begin with initial population of schedules
  - Constructed randomly
- Iterate
  - Breeding process - breed each schedule with a better 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
- Haphazard breeding, such as random street assignment swaps, destroys good solutions
- To construct a good breeding algorithm, we note that most Eulerian cycles can be decomposed into sub-cycles
- A good Eulerian cycle will have “good” sub-cycles and good sub-cycles will induce a good Eulerian cycle
- Our breeding algorithm tries to swap sub-cycles between schedules

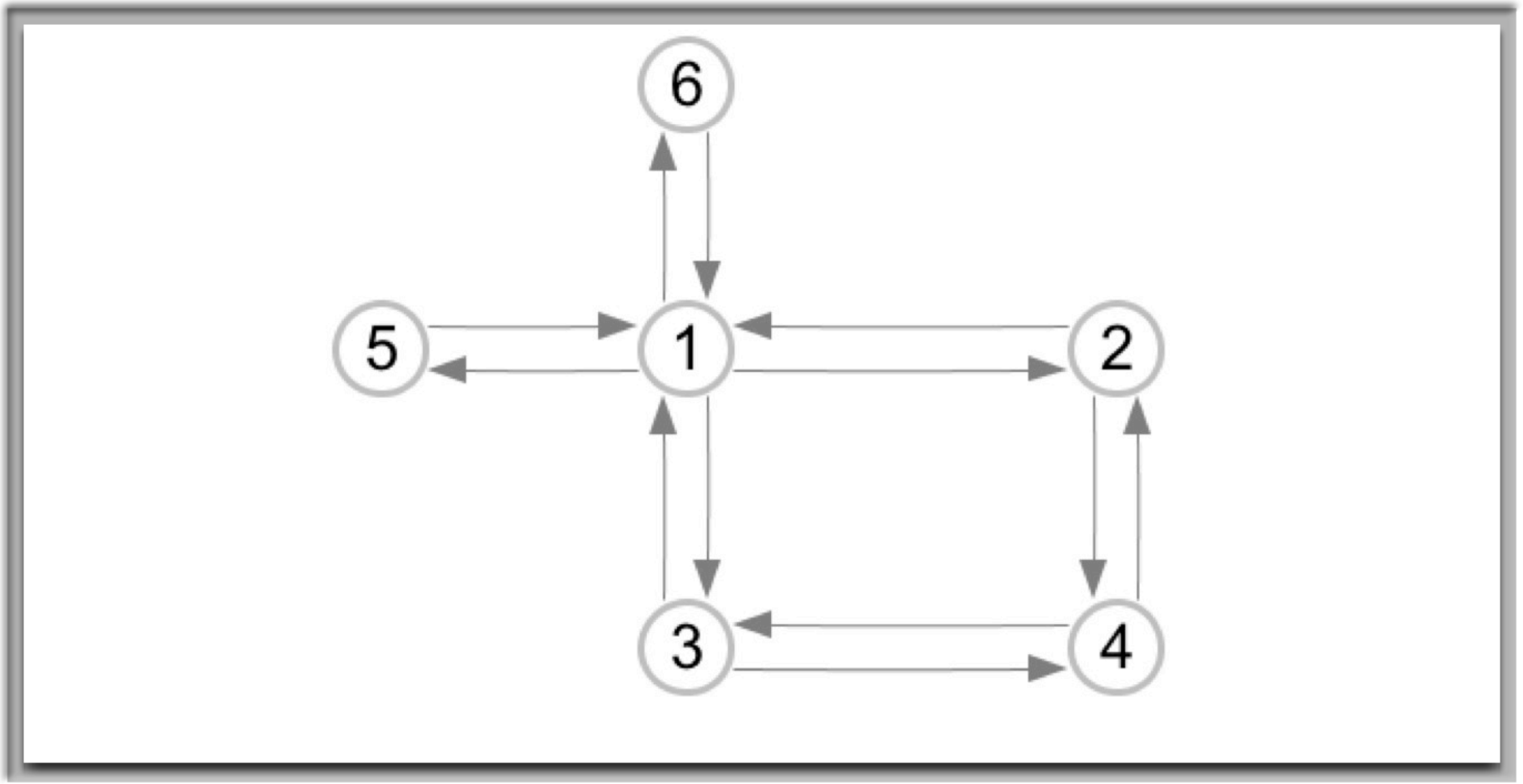


Cycle Example

Figure-8 is decomposed into 2 cycles

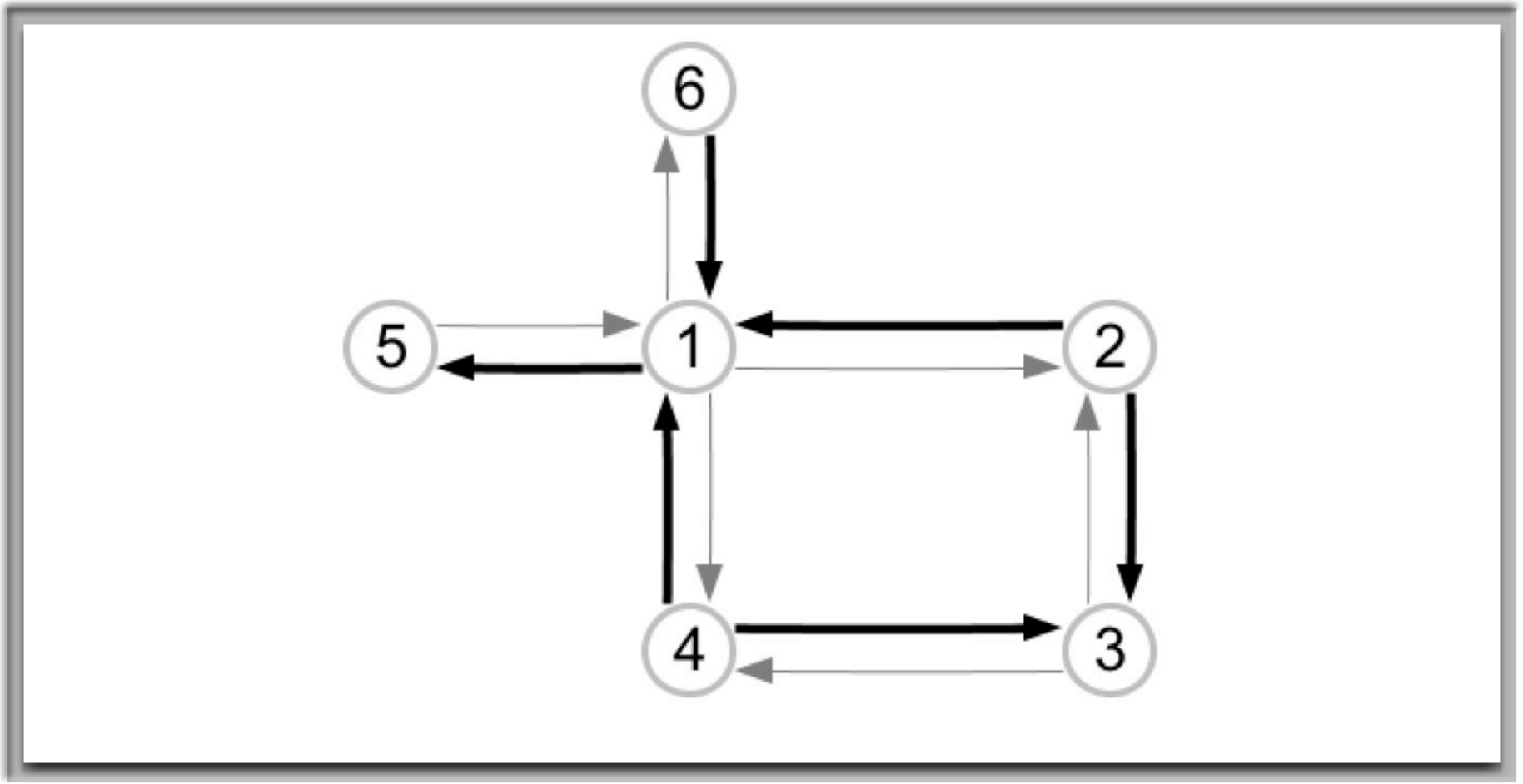
Breeding Process

- A schedule does not have sub-cycles itself; the Eulerian cycle that it induces does
- We make the reasonable assumption that a good schedule allows for good cycles
- Our breeding algorithm determines a cycle allowed by the first schedule on a random day and adjusts the second schedule to allow the same cycle on the same day



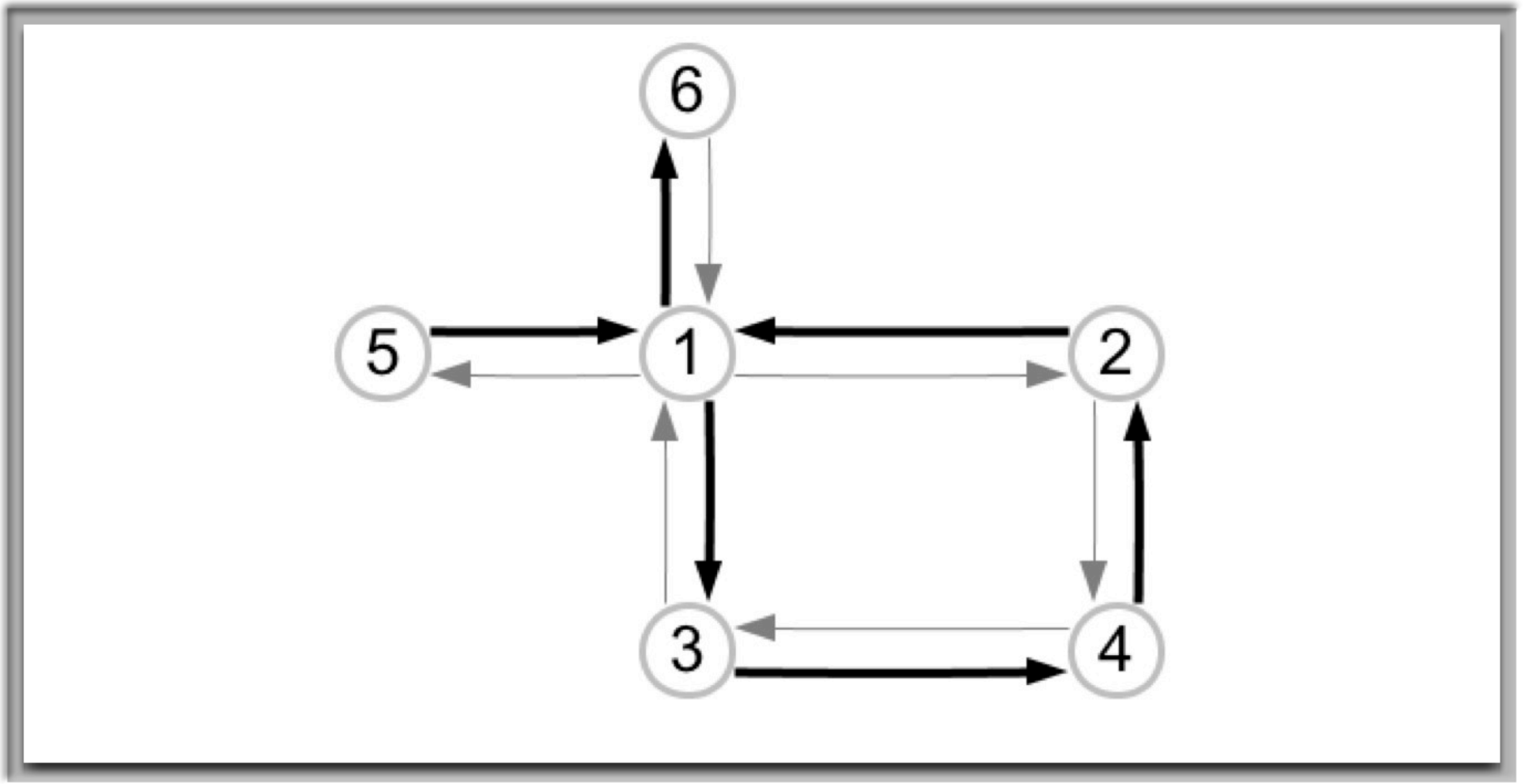
Breeding Example

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



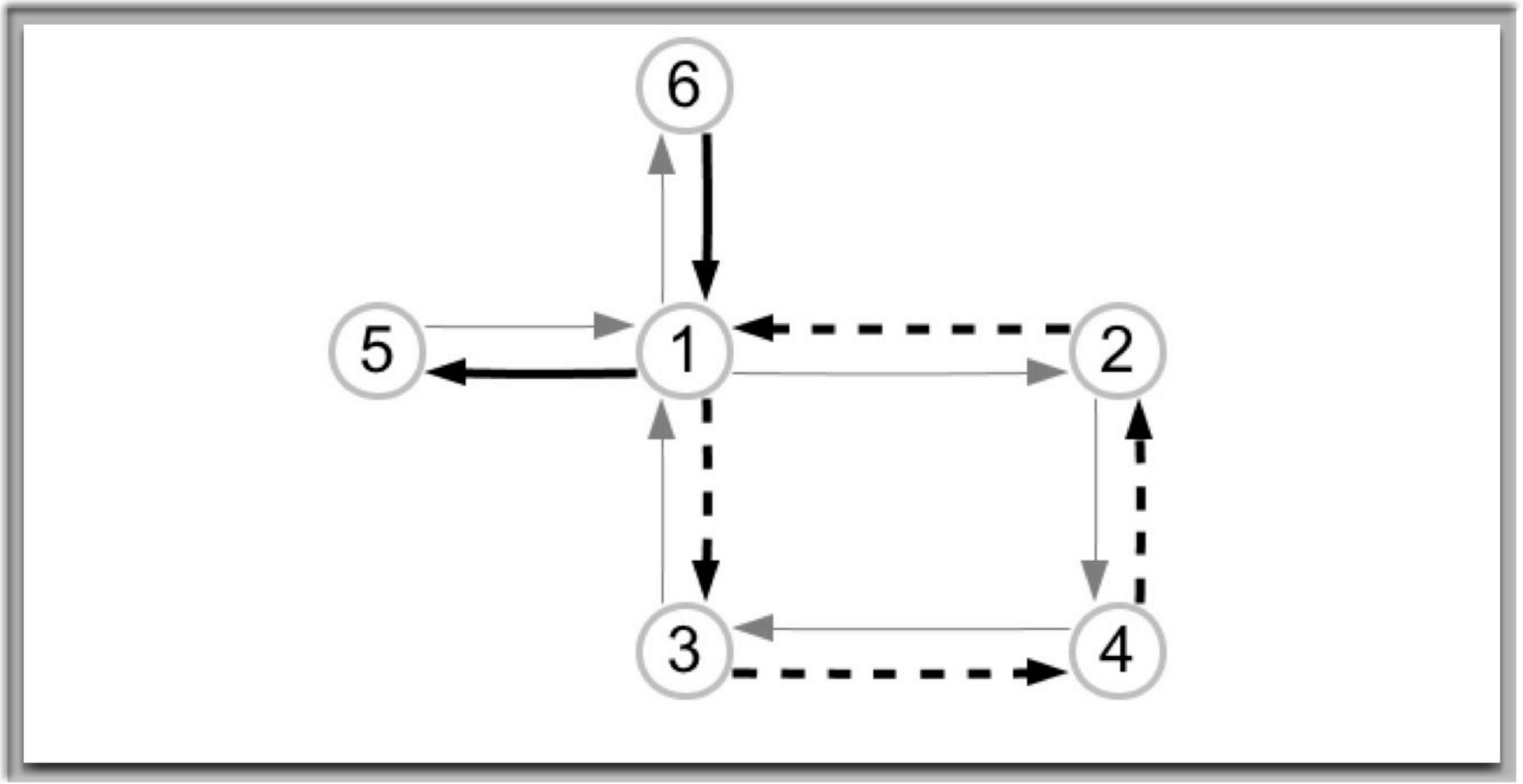
Breeding Example

Schedule 1 on even days - Note induced deadhead



Breeding Example

Schedule 2 on even days - Note good cycle 1, 3, 4, 2, 1



Breeding Example

The good structure of schedule 2 is imparted on schedule 1

Results - Problem 1

- Compare GA against a CPLEX implementation of an integer program (IP) of the problem as well as a naive local search procedure (LS)
  - Local search changes one element of the schedule at a time and keeps improvements
- All run on a 1.83 GHz Intel Core 2 Duo Processor
- Two types of test instances: randomly generated to emulate a city and 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)

Results - Problem 1

- CPLEX on 40 small instances (25 and 50 nodes)
  - CPLEX hits its time limit of 7,200 seconds on 25 instances
  - Fails to obtain a feasible solution to 2 instances
  - Produces optimal solution to 15 instances

Results - Problem 1

- 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

Results - Problem 1

- CPLEX running times on small instances
  - Range from 0.743 seconds to 6580 seconds (and 7200 seconds)
  - Average of 548 seconds, not including instances where CPLEX hits its time limit
- GA: 5.4 seconds on average
- LS: 1.2 seconds on average

Results - Problem 1

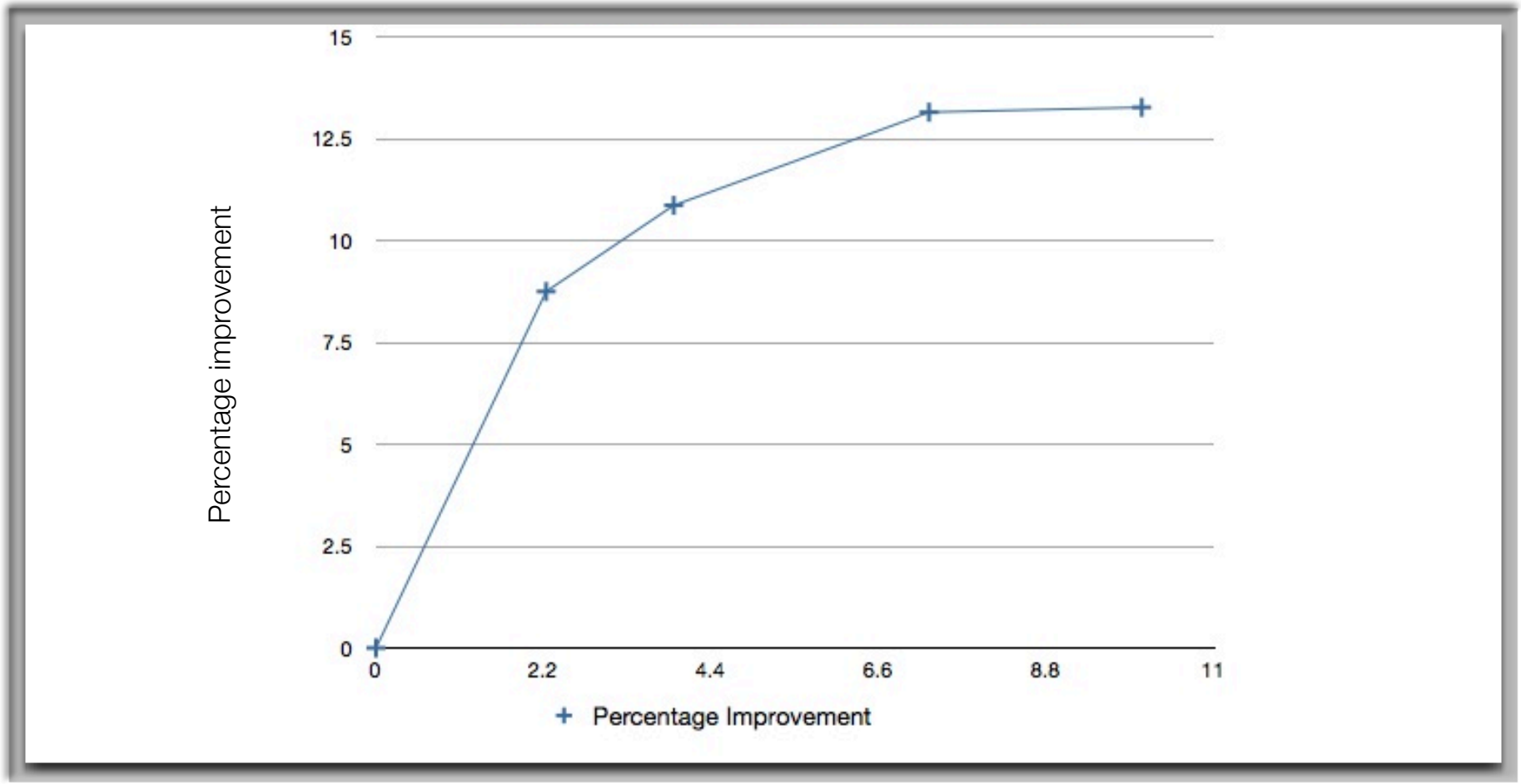
- CPLEX on 20 large instances (100 and 225 nodes)
  - Hits time limit of 7200 seconds on 19 instances
  - Fails to obtain a feasible solution to 18 instances
  - CPLEX is not a viable way of solving large instances

Results - Problem 1

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

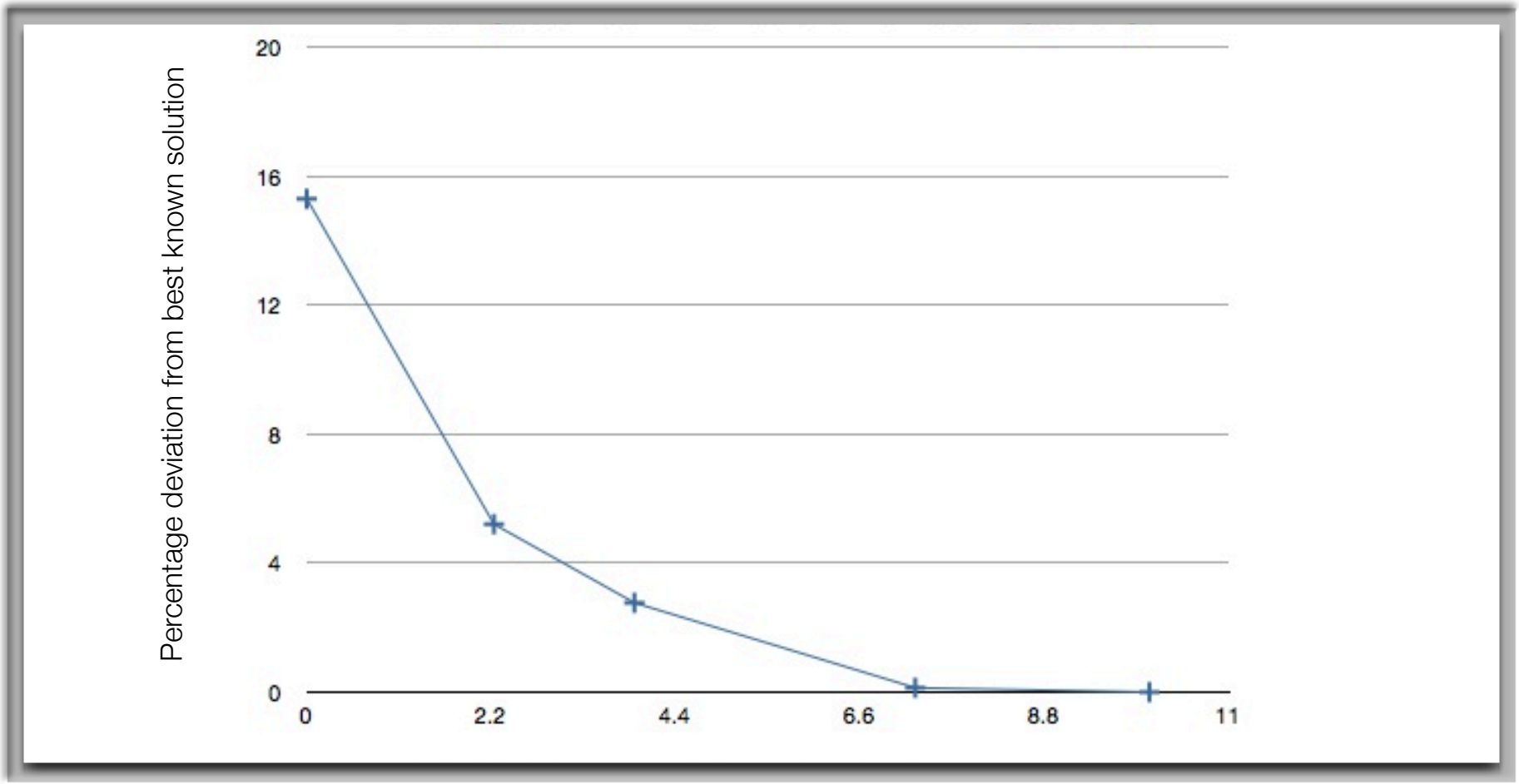
Results - Problem 2

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



Problem 2

Percentage improvement vs. percentage of allowed street sign changes



Problem 2

Percentage deviation from Problem 1 vs. percentage of allowed street sign changes

Results Problem 2

- Convergence of the fitness function in Problem 2 to best-known obtained from Problem 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 to begin with an initial solution and modify it

Conclusions

- Described two new street scheduling and sweeping problems encountered in Washington, DC
- Constructed a novel genetic algorithm
  - Accurate
  - Fast running times
- Interesting managerial results in Problem 2