Coordinating Response to Fatal Accidents

An Application of Many-to-One Matching

Eric Culver and Christina Ebben

Integer Programming, Spring 2019
Instructor: Dr. Steffen Borgwardt
May 9th, 2019

Department of Mathematical & Statistical Sciences
UNIVERSITY OF COLORADO DENVER

MATCHING PROBLEMS

What are matching problems?

- Given a graph, *G*, a “matching” is a pairing up of vertices, each pair connected by an edge. Usually, a matching of minimum cost or maximum cardinality is desired.

$$\sum_{e \in \delta(v)} x_e = 1 \quad \forall v \in V$$

$$\sum_{e \in E(S)} x_e \leq \frac{|S| - 1}{2} \; \forall S \subseteq V, S \text{ odd}$$

$$x_e \in \{0,1\} \quad \forall e \in E$$

Where $\delta(v)$ denotes the set of edges incident to node v for $\forall v \in V$

MATCHING PROBLEMS

Bipartite Matching

- Here, our graph is divided into two parts, *X* and *Y*. The "odd set inequalities" become redundant.
  - This is where most applications lie.

$$\sum_{e \in E(S)} x_e \leq \frac{|S| - 1}{2} \; \forall S \subseteq V, S \text{ odd}$$

MATCHING PROBLEMS

One-to-One Matching

- Standard bipartite matching problem, where each element of *X* is matched to exactly one element of *Y*, and every element of *Y* is matched to exactly one element of *X*.

$$\sum_{i \epsilon X} x_{ij} = 1 \; \forall j \in Y$$

$$\sum_{j \epsilon Y} x_{ij} = 1 \; \forall i \in X$$

$$x_{ij} \in \{0,1\} \quad \forall i \in X, j \in Y$$

MATCHING PROBLEMS

k-to-One Matching

- Each element of *X* is matched to exactly *k* elements of *Y*, and each element of *Y* is matched to exactly one element of *X*.

$$\sum_{i \epsilon X} x_{ij} = 1 \; \forall j \in Y$$

$$\sum_{j \epsilon Y} x_{ij} = k \; \forall i \in X$$

$$x_{ij} \in \{0,1\} \quad \forall i \in X, j \in Y$$

MATCHING PROBLEMS

Many-to-One Matching

- Each element of *X* is matched to at least one element of *Y*, and each element of *Y* is matched to exactly one element of *X*.

$$\sum_{i \epsilon X} x_{ij} = 1 \ \forall j \in Y$$

$$\sum_{j \epsilon Y} x_{ij} \geq 1 \ \forall i \in X$$

$$x_{ij} \in \{0,1\} \quad \forall i \in X, j \in Y$$

MATCHING PROBLEMS

Many-to-Many Matching

- Each element of *X* is matched to at least one element of *Y*, and each element of *Y* is matched to at least one element of *X*.

$$\sum_{i \epsilon X} x_{ij} \geq 1 \; \forall j \in Y$$

$$\sum_{j \epsilon Y} x_{ij} \geq 1 \; \forall i \in X$$

$$x_{ij} \in \{0,1\} \quad \forall i \in X, j \in Y$$

MATCHING PROBLEMS

Matching Problems are Integer Programs

- In particular, matching problems are binary programs.
- There are polynomial-time algorithms for one-to-one matchings on general graphs.
  - » However, the decision version of *k*-to-one bipartite matching is NP-Complete.

MANY-TO-ONE MATCHING APPLICATIONS

The College Admissions Problem

- D. Gale and L. S. Shapley
  - » *College Admissions and the Stability of Marriage*, 1962
- The College Admissions Problem
  - n student applicants
  - m colleges
    - college u_i will accept up to q_i students
- Many-to-One Matching Application
  - Note: Assignment Problem
    - » Assign n tasks to m machines each with a specified capacity

MANY-TO-ONE MATCHING APPLICATIONS

The College Admissions Problem

- Preferences
  - Student applicants rank the colleges in order of preference
  - Colleges rank the students in order of preference
- Note: For this to work, ignore ties in preference, the "Hungarian admission test" throw away rule, and "couples admission"

- Stable Assignment
  - No student prefers a college over the one they have been admitted to AND no college prefers a student over the ones admitted
- Optimal Assignment
  - Stable AND every student is at least "well off" as if the assignment had been different
- "Philosophical Principle"

MANY-TO-ONE MATCHING APPLICATIONS

The College Admissions Problem

The Stable Marriage Problem

- Special case of the College Admissions Problem
  - Equal number of students and colleges
  - Each college u_i accepts up to 1 student

  $\Rightarrow n$ “boys” propose to n “girls”
- Becomes a One-to-One Matching Application
  - Assignment Problem
    - Assign n tasks to n people

- Consider Preferences
  - “Boys” propose in order of preference
  - “Girls” accept in order of preference
- Is there a stable assignment?

MANY-TO-ONE MATCHING APPLICATIONS

The College Admissions Problem

- Stable Assignments
  - » Gale and Shapley proved that stable marriage assignments exist

    ⇒ Stable college admission assignments exist
    - The proof "is entirely analogous to the proof given for the marriage problem"
- Optimal Assignments
  - » Optimal College Admission Assignments
    - "deferred acceptance"
- "The main proof is carried out in not in mathematical symbols but in ordinary English… Knowledge of calculus is not presupposed. In fact, one hardly needs to know how to count. Yet any mathematician will immediately recognize the argument as mathematical"

MANY-TO-ONE MATCHING APPLICATIONS

Many-to-One Matching Applications

- Economics
  - “who gets what” especially when “what” is “invisible”
- Dorm room assignments
- Determining the waitlist for organ transplants
  - “pairwise kidney exchange programs”
- Labor Markets
  - n people applying to m companies with a fixed number of openings
- Coordinating Response to Fatal Accidents

OUR APPLICATION

Coordinating Response to Fatal Accidents

Problem Outline

- Fatal car accidents require a response from both police and fire department

- Denver Police and Fire Departments have separate dispatch systems
  - Do not communicate
  - Responders are uninformed of the other entity’s likely response time

Objectives

- Construct a **Many-to-One matching** between police and fire stations
  - Generate a list of “Suggested Partnerships”
  - Aid responders by ensuring that they are informed of estimated arrival times of the other entity

OUR APPLICATION

Model

Where $P :=$ the set of police stations, $F :=$ the set of fire stations, $C :=$ the set of all car crashes

$$\textit{Minimize} \quad \sum_{i \in P, j \in F} c_{ij} x_{ij}$$

$$\textit{Subject To} \quad \sum_{i \in P} x_{ij} = 1 \quad \forall j \in F$$

$$\sum_{j \in F} x_{ij} \geq 1 \quad \forall i \in P$$

$$x_{ij} \in \{0,1\} \text{ for } \forall i \in P, \forall j \in F$$

$$\textit{Where} \quad c_{ij} = \sum_{c \in C} |dist(c,i) - dist(c,j)|$$

OUR APPLICATION

Data

- What we had
  - Idea + Model Formulation
  - Denver Open Data Catalog
    - Many data sets
    - Large data sets
    - Data sets with errors
  - Common Sense
    - What data will work and what will hurt our computers?

- What we needed
  - Usable data sets for:

    $P :=$ the set of police stations
    $F :=$ the set of fire stations
    $C :=$ the set of all car crashes

  - Data to give to Python and AMPL
    - Unique Identifiers
    - Latitude and Longitude

OUR APPLICATION

Data

Raw Data Sets (Denver Open Data Catalog)

- Police Stations
  - *Original: 29 data points*
  - Includes all police "places"
  - Addresses/PO Boxes given
- Fire Stations
  - *Original: 39 data points*
  - Addresses given
  - Station ID's given
- Traffic Accidents
  - *Original: 154,967 data points*
    - Five years of all reported traffic incidents
  - Latitude and Longitude given
    - Bonus!
  - Preassigned ID's given
    - Assigned by date/time for all reported traffic incidents

OUR APPLICATION

Data

Cleaning Up the Data

- Police and Fire Stations
  - » Only include deployable stations
    - i.e. exclude police bicycle impound and wildfire response stations
  - » PO Box → Physical Address
    - Generate Unique Identifiers

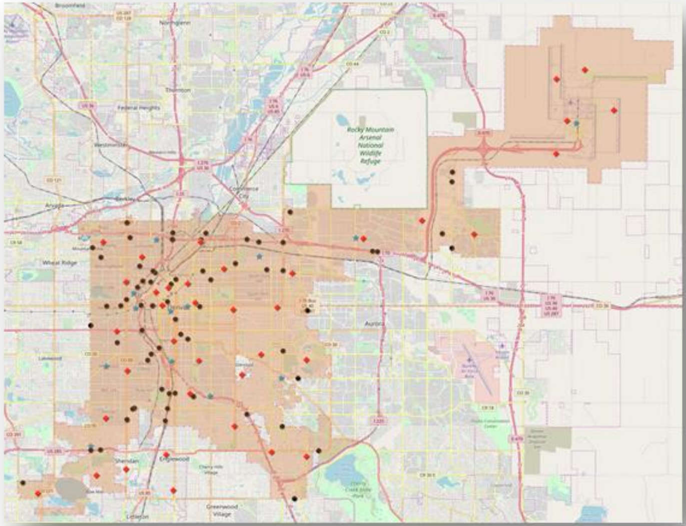
- Traffic Accidents
  - » Filter data to "Fatal Accidents"
    - » 154,967 → 330 data points
  - » Choose a usable time frame
    - Recall: We don't want to hurt our computers
    - 5 years → 15 Months
      - » 330 → 74 data points

OUR APPLICATION

Data

GIS: Filling in the Gaps

- Identify errors in the data
  - » Points in "Nowhere Island"?
    - Lat/Lon Missing
  - » Points in China?
    - Lat/Lon Inputted Wrong
- Generate latitude and longitude
  - » ARCGIS Geometry Toolbox



➢ Finalized Data Sets

- ★ $P :=$ the set of police stations
  14 Data Points
- ◆ $F :=$ the set of fire stations
  35 Data Points
- ● $C :=$ the set of all car crashes
  70 Data Points

OUR APPLICATION

Computer Code - AMPL

```
set FIRE; # fire stations
set POLICE; # police stations
set ACCIDENT; # potentially fatal accidents

param fx {FIRE}; # fire station locations x
param fy {FIRE}; # fire station locations y
param px {POLICE}; # police station locations x
param py {POLICE}; # police station locations y
param ax {ACCIDENT}; # potentially fatal accident
locations x
param ay {ACCIDENT}; # potentially fatal accident
locations y
param incompatibility {i in POLICE, j in FIRE} =
   sum {k in ACCIDENT} <<0; -1, 1>>(<<0; -1, 1>>(px[i] -
ax[k])
                  + <<0; -1, 1>>(py[i] - ay[k])
                  - <<0; -1, 1>>(fx[j] - ax[k])
                  - <<0; -1, 1>>(fy[j] - ay[k])) ; # how poorly i and
j would work together
```

```
var match {i in POLICE, j in FIRE} binary; # whether i
and j get matched

minimize Total_Incompatibility:
   sum {i in POLICE, j in FIRE} incompatibility[i,j] *
match[i,j];

subject to Fire_Match{j in FIRE}:
   sum {i in POLICE} match[i,j] == 1;

subject to Police_Match{i in POLICE}:
   sum {j in FIRE} match[i,j] >= 1;
```

OUR APPLICATION

Computer Code - Python

```
…
with open('fire_stations_lat_and_lon.csv', 'r') as fireFile:
fireCSV = csv.reader(fireFile, delimiter=',')
    for i, row in enumerate(fireCSV):
        if i is 0:
            for j in range(len(row)):
                if row[j] == 'FIRE_STATION_NUM':
                    namej = j
                if row[j] == 'REAL_LAT':
                    latj = j
                if row[j] == 'REAL_LON':
                    lonj = j
        else:
            fireDat.append([])
            fireDat[i-1].append(row[namej])
            fireDat[i-1].append(int(row[latj])*(10**(-6)))
            fireDat[i-1].append(int(row[lonj])*(10**(-6)))
```

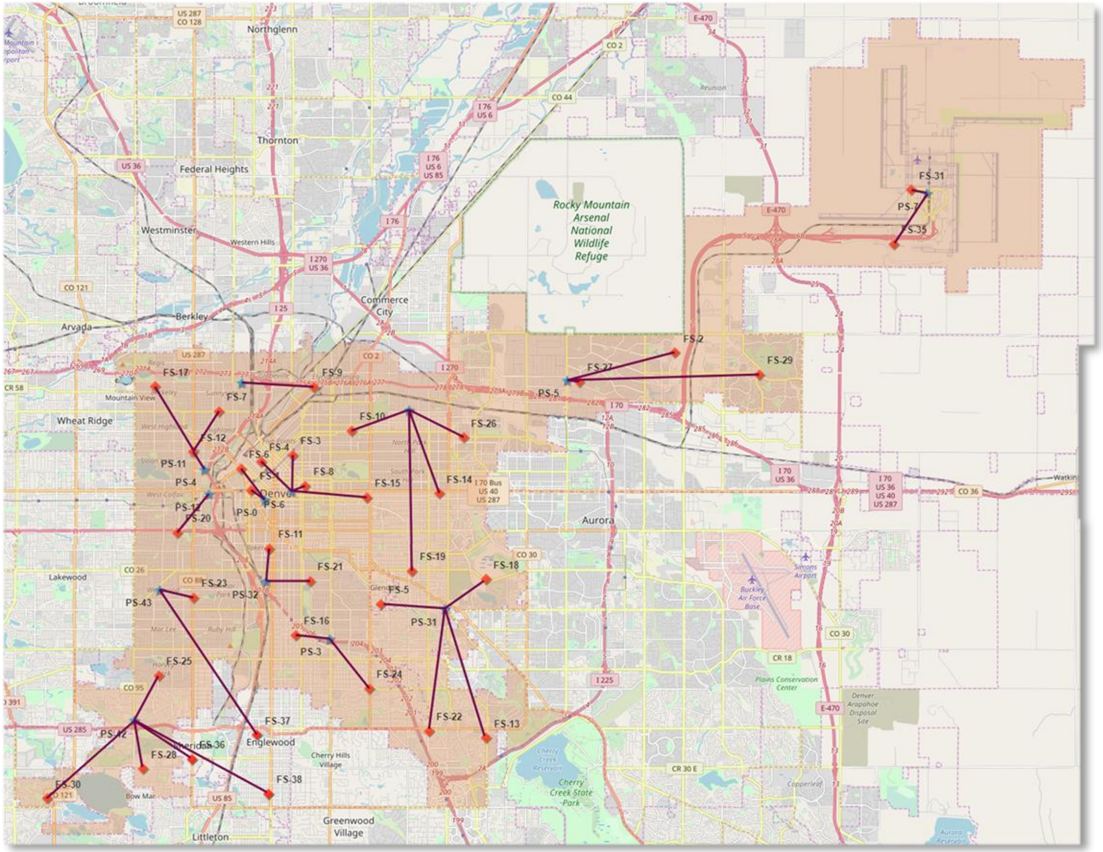
```
accDat = []

with open('Accidents_fatal.csv', 'r') as accFile:
    accCSV = csv.reader(accFile, delimiter=',')
    for i, row in enumerate(accCSV):
        if i is 0:
            for j in range(len(row)):
                if row[j] == 'CRASH_NUM':
                    namej = j
                if row[j] == 'REAL_LAT':
                    latj = j
                if row[j] == 'REAL_LON':
                    lonj = j
        else:
            accDat.append([])
            accDat[i-1].append(row[namej])
            accDat[i-1].append(int(row[latj])*(10**(-6)))
            accDat[i-1].append(int(row[lonj])*(10**(-6)))
…
```

OUR APPLICATION

Results

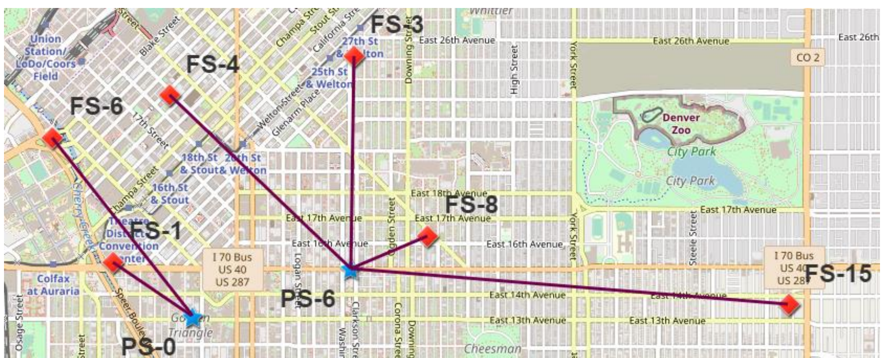
(PS-0,FS-1)
(PS-0,FS-6)
(PS-1,FS-9)
(PS-11,FS-7)
(PS-11,FS-17)
(PS-12,FS-20)
(PS-2,FS-26)
(PS-2,FS-14)
(PS-2,FS-19)
(PS-2,FS-10)
(PS-3,FS-24)
(PS-3,FS-16)
(PS-31,FS-22)
(PS-31,FS-13)
(PS-31,FS-18)
(PS-31,FS-5)
(PS-32,FS-21)
(PS-32,FS-11)
(PS-4,FS-12)
(PS-42,FS-38)
(PS-42,FS-36)
(PS-42,FS-25)
(PS-42,FS-28)
(PS-42,FS-30)
(PS-43,FS-37)
(PS-43,FS-23)
(PS-5,FS-29)
(PS-5,FS-2)
(PS-5,FS-27)
(PS-6,FS-15)
(PS-6,FS-8)
(PS-6,FS-3)
(PS-6,FS-4)
(PS-7,FS-31)
(PS-7,FS-35)



OUR APPLICATION

Policy Recommendations

- Implementation
  - » Input the set of “Suggested Partnerships” into the separate **dispatch** systems



» When a potentially fatal accident happens, the system “pings” the dispatcher with the partnership

» Dispatcher relays the information to responders

OUR APPLICATION

Looking Ahead

- Things to Change
  - Capacities
    - Static and Event Dependent
  - Dispatcher Needs
    - Human Application
  - Inverse?
    - When do we NOT need a partnership
      - Accountability and resource conservation
- More In-Depth Analysis
  - Patrol Routs
    - GPS Real Time Locators
  - Drive Time
  - Spatial Statistics
    - Clustering and weighting accident data
  - Stability
    - Compare results for different timeframes

REFERENCES

- *ArcGIS Desktop*. Redlands, CA: ESRI, 2010. Computer software.
- City and County of Denver, Denver Police Department / Data Analysis Unit. *Denver Open Data Catalog: Police Stations*. City and County of Denver, 2018. Data File.
- City and County of Denver, Denver Police Department / Data Analysis Unit. *Denver Open Data Catalog: Traffic Accidents*. City and County of Denver, 2019. Data File.
- Conforti, Michele, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. N.p.: Springer, 2014. Print.
- Denver Fire Department. *Denver Open Data Catalog: Fire Stations.* City and County of Denver, 2018. Data File.
- Ebben, Christina. " Poster_Map_Matchings." *ArcGIS*, ESRI, 2019, https://www.arcgis.com/home/item.html?id=acb6d4d47b53473bb2cfc158061364db.
- ESRI. *Arcgis.com*. N.p., 2018. Web.
- Fourer, Robert, David M. Gay, and Brian W. Kernighan. *AMPL*. Vers. 5.2. Belmont, Calif.: Duxbury Press, 2013. Computer software.
- Gale, D., and L. S. Shapley. "College Admissions and the Stability of Marriage." *The American Mathematical Monthly* 69.1 (1962): 9. Print.
- Niederle, Muriel, Alvin E. Roth, and Tayfun Sönmez. "Matching." *The New Palgrave Dictionary of Economics*. 2nd ed. N.p.: Palgrave Macmillan, 2008. Print.