Technical Report: Final Project

EECE 2560: Fundamentals of Engineering Algorithms

Smart City Resource Allocation

Matthew Garcia and Rostyslav Rozhok

Department of Electrical and Computer Engineering Northeastern University

garcia.matt@northeastern.edu

rozhok.r@northeastern.edu

November 16, 2024

Contents

1 Project Scope

The aim of this project is to design and develop a smart city resource allocation system, making use of greedy algorithms for optimization and graph algorithms to represent resource flow, in order to allocate resources to various sectors of a smart city.

The project's main objectives are to:

- Efficiently optimize resource allocation so usage is maximized and waste is minimized.
- Account for the dynamics presented within a city, such as a growing population and demand.
- Completely automate the process of allocating resources.

There is no exact definition of Smart City. Our project focuses on allocation of resources to the sectors that need them the most.

The key issues of any city:

- Energy management: optimization of use, conversion, transportation, and losses. One of the future practices will be the management of electric vehicles' batteries to take in the energy when there is surplus of energy and give off when there is shortage.
- Trash management. One of the solutions is diligent sorting of trash with further burning to get the energy, and reuse of remnants.
- Water purification. Chemicals in detergents, cleaning, fertilizers, pesticides, byproducts of production and heavy metals cause many illnesses. The policies and proper filters are of high priority for a smart city.
- Real estate development. New facilities and apartment complexes require infrastructure of energy, water ways, and waste connections and facilities which are usually not present in abundance.

While our project will focus on the quantifiable part of this, it is important to understand the issues behind the organization of a smart city.

2 Project Plan

2.1 Timeline

The overall timeline for this project will be divided into phases:

- Week 1 (October 7 - October 13): Define the scope of the project and how it will be implemented, allocate tasks to team members, create GitHub repository.
- Week 2 (October 14 - October 20): Do research on smart cities, create system design, start coding basic system functionalities.
- Week 3 (October 21 - October 27): Start technical report from research done prior week, start coding system design.

- Week 4 (October 28 - November 3): Work on algorithmic/system code, start coding frontend, continue the report.
- Week 5 (November 4 - November 10): Start user interface and integrate with system code, test code, continue the report.
- Week 6 and 7 (November 11 - November 17): Begin presentation, finalize testing and user interface, report, and presentation.
- Week 8 (November 2-Dec 48): Present project, submit report and code.

2.2 Milestones

Key milestones include:

- Project Scope and Plan (October 10).
- GitHub Repository Setup (October 10).
- System Code Completion (November 10).
- User Interface Completion (November 15).
- Final System Testing and Report Draft (November 17).
- Final Presentation and Report Submission (November 28).

3 Team Discussion Summary

3.1 Team Roles

- Matthew Garcia: System Development, GitHub Repository Management, Technical Documentation (Partially), System Testing.
- Rostyslav Rozhok: User Interface Development, System Development (Partially), Technical Documentation, System Testing.

3.2 Tools and Technologies

Team members need the following skills to complete the project:

- System Code: C++.
- User Interface: gtkmm, Qt, wxWidgets Libraries (Need to evaluate what works the best).
- Version Control: GitHub.
- Technical Report: Overleaf.

4 Skills and Tools Assessment

4.1 Skills Assessment

There are a few gaps in knowledge of using C++ libraries to display the GUI, greedy and graph algorithms, and the use of Overleaf. These skills can develop through attending class and completing assessments. The use of the correct C++ libraries can improve through attending workshops and practice.

4.2 Tools

We will use the following tools:

- C++ for system code.
- gtkmm, Qt, and/or wxWidgets for GUI.
- GitHub for version control.
- Overleaf for technical documentation.

4.3 Pseudocode and Complexity Analysis

Allocator

1. **ResourceAllocator(const string& file)**
   Initialisation of the allocator and call of loadResources to parse input.
2. **void loadResources(const string& file)**
   Parsing of input files and populating resource-specific vectors with data.
3. **void allocateResources()**
   Calling allocation methods for electricity, water, gas, waste.
4. **void allocateElectricity()**
   Sorting of electricity resources and adjusting availability by peak demand.
5. **void allocateWater() and void allocateGas()**
   Reduction of water and gas availability by a fixed amount for each district
6. **void allocateWasteManagement()**
   Deduction of waste management availability with adjustments in the districts.

Flow graph

1. **ResourceFlowGraph(int n)**
   Constructs a flow graph with specified node count and adjacency list.
2. **void addEdge**
   Addition of edge and reverse edge to support capacity and flow modeling.

4.4 DataCollection and Preprocessing

Input: A csv file is used as a data input. A graph configuration should be provided.

Output: Resource allocation statistics as console output, and a file for graph representation of resource flow.

5 Initial Setup Evidence

5.1 Project Repository

The project repository has been created in GitHub and is accessible to both team members. The repository can be found at https://github.com/mattgar417/Smart-City .

6 Progress Review

6.1 Progress Update

In the past iteration we did the following:

1. A CSV file has been created in order to track the resources required for different sectors of the smart city
2. The code has been updated to read the csv files and use it to allocate resources rather than examples
3. The allocation algorithms have been fleshed out so that they now are able to allocate by the different resources (water, electricity, waste management)

4. The code has been separated into different modules, including source, header, and main files

In this iteration we

1. Fully updated the graph algorithm
2. Fully updated the resource allocator
3. Added some comments for explainability
4. Tested the code with the data being read

The plan for the next iteration is to finish the GUI, add more comments, test for more cases with different data and make changes as needed.

6.2 Issues Encountered

There were no issues we couldn't solve in this iteration.

7 Revised Project Plan

7.1 Updated Plan

Since we are presenting on the 4th, we are on schedule to make all of the adjustments we need.

8 Results and 9 Discussion

To be updated with further testing.

10 Conclusion

Designing and building Smart Cities is one of the most crucial objectives of humanity in the 21st century. Developments in this area can greatly enhance the living conditions, efficiency and security. Well designed interconnected systems can also allow for much work to be distributed to robots and automated machines.

Obtaining data and having knowledge of various needs of inhabitants and infrastructure upkeep necessities lead to more optimal design of control systems.

11 References

M. Zaman, M. Al Islam, A. Tantawy, C. J. Fung and S. Abdelwahed, "Adaptive Control for Smart Water Distribution Systems," *2021 IEEE International Smart Cities Conference (ISC2)*, Manchester, United Kingdom, 2021, pp. 1-6, doi: 10.1109/ISC253183.2021.9562812.

R. Chen, R. Wang, W. Zhao, H. Sun, X. Wu and J. Xie, "Cooperative benefit allocation for domestic electricity of smart communities based on coalitional game theory," *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, Chongqing, China, 2020, pp. 185-189, doi: 10.1109/ICICAS51530.2020.00045.

Bash, Cullen, et al. "IT for Sustainable Smart Cities: A Framework for Resource Management and a Call for Action." *The Bridge*, National Academy of Engineering, 17 Mar. 2023, https://www.nae.edu/290991/IT-for-Sustainable-Smart-Cities-A-Framework-for-Resource-Management-and-a-Call-for-Action.

Appendix A

main.cpp

```
#include <iostream>
#include "ResourceAllocator.h"
#include "ResourceFlowGraph.h"

using namespace std;

int main() {
  string dataFile = "resources_data.csv";

  ResourceAllocator allocator(dataFile);
  allocator.allocateResources();
  allocator.displayAllocation();

  ResourceFlowGraph flowGraph(5);
  flowGraph.addEdge(0, 1, 10);
  flowGraph.addEdge(1, 2, 5);

  cout << "Resource Allocation and Flow complete.\n";
  return 0;
  return 0;
}
```

ResourceAllocator.cpp

```
#include "ResourceAllocator.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <algorithm>

using namespace std;

ResourceAllocator::ResourceAllocator(const string& file) {
  loadResources(file);
}

void ResourceAllocator::loadResources(const string& file) {
  ifstream inFile(file);
  string line, type;
```

```
  while (getline(inFile, line)) {
      stringstream ss(line);
      string type, districtStr, costStr, importanceStr, availabilityStr, peakDemandStr, avgDemandStr,
waterQualityStr, gasPressureStr;

      getline(ss, type, '\t');
      getline(ss, districtStr, '\t');
      getline(ss, costStr, '\t');
      getline(ss, importanceStr, '\t');
      getline(ss, availabilityStr, '\t');
      getline(ss, peakDemandStr, '\t');
      getline(ss, avgDemandStr, '\t');
      getline(ss, waterQualityStr, '\t');
      getline(ss, gasPressureStr, '\t');

      int district = stoi(districtStr);
      double cost = stod(costStr);
      double importance = stod(importanceStr);
      double availability = stod(availabilityStr);

      if (type == "Electricity") {
          double peakDemand = stod(peakDemandStr);
          double avgDemand = stod(avgDemandStr);
          electricityResources.push_back({type, district, cost, importance, availability, peakDemand,
avgDemand});
      } else if (type == "Water") {
          double waterQuality = stod(waterQualityStr);
          waterResources.push_back({type, district, cost, importance, availability, waterQuality});
      } else if (type == "Gas") {
          double gasPressure = stod(gasPressureStr);
          gasResources.push_back({type, district, cost, importance, availability, gasPressure});
      } else if (type == "Waste management") {
          wasteManagementResources.push_back({type, district, cost, importance, availability});
      }
  }
}

void ResourceAllocator::allocateResources() {
  allocateElectricity();
```

```
    allocateWater();
    allocateGas();
    allocateWasteManagement();
}

void ResourceAllocator::allocateElectricity() {
    sort(electricityResources.begin(), electricityResources.end(), [](const Electricity& a, const
Electricity& b) {
        return a.peakDemandMW > b.peakDemandMW;
    });

    for (auto& e : electricityResources) {
        e.availability -= min(e.availability, e.peakDemandMW);
    }
}

void ResourceAllocator::allocateWater() {
    for (auto& w : waterResources) {
        w.availability -= 10;
    }
}

void ResourceAllocator::allocateGas() {
    for (auto& g : gasResources) {
        g.availability -= 50;
    }
}

void ResourceAllocator::allocateWasteManagement() {
    for (auto& wm : wasteManagementResources) {
        wm.availability -= 5;
    }
}

void ResourceAllocator::displayAllocation() const {
    cout << "Electricity Allocation Results:\n";
    for (const auto& e : electricityResources) {
        cout << "District " << e.district << ": Remaining Availability: " << e.availability << " MW\n";
    }
```

```
    cout << "\nWater Allocation Results:\n";
    for (const auto& w : waterResources) {
        cout << "District " << w.district << ": Remaining Availability: " << w.availability << " units\n";
    }

    cout << "\nGas Allocation Results:\n";
    for (const auto& g : gasResources) {
        cout << "District " << g.district << ": Remaining Availability: " << g.availability << " kPa\n";
    }

    cout << "\nWaste Management Allocation Results:\n";
    for (const auto& wm : wasteManagementResources) {
        cout << "District " << wm.district << ": Remaining Availability: " << wm.availability << " tons\n";
    }
}
```

ResourseAllocator.h

```
#ifndef RESOURCE_ALLOCATOR_H
#define RESOURCE_ALLOCATOR_H

#include <vector>
#include <string>

// Base struct for common attributes
struct Resource {
    std::string type;
    int district;
    double cost;
    double importance;
    double availability;
};

// Extended attributes for specific resource types
struct Electricity : public Resource {
    double peakDemandMW;
    double avgDemandMW;
};

struct Water : public Resource {
    double waterQualityPH;
```

```
};

struct Gas : public Resource {
    double gasPressureKPa;
};

struct WasteManagement : public Resource {};

// ResourceAllocator class
class ResourceAllocator {
private:
    std::vector<Electricity> electricityResources;
    std::vector<Water> waterResources;
    std::vector<Gas> gasResources;
    std::vector<WasteManagement> wasteManagementResources;

public:
    explicit ResourceAllocator(const std::string& file);
    void allocateResources();
    void displayAllocation() const;

private:
    void loadResources(const std::string& file);
    void allocateElectricity();
    void allocateWater();
    void allocateGas();
    void allocateWasteManagement();
};

#endif // RESOURCE_ALLOCATOR_H
```

```
#include "ResourceFlowGraph.h"

ResourceFlowGraph::ResourceFlowGraph(int n) : adjList(n) {}

void ResourceFlowGraph::addEdge(int from, int to, double capacity) {
    adjList[from].push_back({to, capacity});
    adjList[to].push_back({from, 0}); // Reverse edge
```

```
}
```

ResourceFlowGraph.cpp

```
#include "ResourceFlowGraph.h"

ResourceFlowGraph::ResourceFlowGraph(int n) : adjList(n) {}

void ResourceFlowGraph::addEdge(int from, int to, double capacity) {
  adjList[from].push_back({to, capacity});
  adjList[to].push_back({from, 0}); // Reverse edge
}
```

ResourceFlowGraph.cpp

```
#ifndef RESOURCE_FLOW_GRAPH_H
#define RESOURCE_FLOW_GRAPH_H

#include <vector>

struct Edge {
  int to;
  double capacity;
};

class ResourceFlowGraph {
private:
  std::vector<std::vector<Edge>> adjList;

public:
  explicit ResourceFlowGraph(int n);
  void addEdge(int from, int to, double capacity);
};

#endif // RESOURCE_FLOW_GRAPH_H
```

Appendix B

Csv file

resources_data.csv

Type	District	Cost	Importance	Availability	PeakDemand_MW	AverageDemand_MW	WaterQuality(pH)	GasPressure_kPa	GasUsage_MiliionCubicMeters	WaterUsage_MillionLiters
Electricity	1		0.4	20	30	20				
Electricity	2		0.3	70	50	35				
Electricity	3		0.5	200	400	220				
Electricity	4		0.3	300	210	130				
Electricity	5		0.6	190	350	200				
Water	1		0.5	30			7.4			10
Water	2		0.5	40			8			80
Water	3		0.4	500			7.8			56
Water	4		0.3	200			6.8			89
Water	5		0.5	15			6.56			210
Gas	1		0.3	100000				4	60000	
Gas	2		0.6	300000				3	200000	
Gas	3		0.5	100000				2	300000	
Gas	4		0.5	200000				5	450000	
Gas	5		0.4	1500000				11	1100000	
Waste management	1		0.3							
Waste management	2		0.5							
Waste management	3		0.3							
Waste management	4		0.6							
Waste management	5		0.5							

Output: