ABSTRACT

Emergency Medical Services (EMS) dispatchers are given the stressful task to redeploy EMS vehicles to maintain adequate coverage of a city. Most city dispatchers perform this task by looking at a physical or computer map and forming a decision based on their experience and intuition. This can result in inefficient redeployment of vehicles costing the city wasted resources (e.g. gas, vehicle maintenance) and manpower. The purpose of this research project is to create a tool to assist dispatchers in redeploying vehicles in a quick and efficient manner through algorithm analysis of total system state. An integrated software application was created to quickly provide the dispatcher with a recommendation, and in conjunction with their experience, a final redeployment decision would be made.

1 INTRODUCTION

Many cities use a third service model for managing Emergency Medical Services (EMS) vehicle deployments. Under such a system, vehicles are stationed to arbitrary posts around the city, independent of fire or police vehicles deployments (thus constituting a “third service” of the city). This model requires that vehicles be periodically redeployed in order to maintain adequate emergency coverage of the city. Currently, most cities decide how to implement redeployment by having an experienced dispatcher look at the current positions of the vehicles on a map and decide, through only his or her judgment and experience, which vehicles should be moved. This can result in unacceptably inefficient management of the EMS system. In one case, in the city of Toronto, Canada, a single vehicle drove several hundred kilometers in one shift, yet was only dispatched to two calls.

1.1 Problem Statement

For the EMS dispatcher who needs to redeploy vehicles after a call, the EMS Redeploy System is a computer program to assist the dispatcher through algorithm analysis of total system state; unlike current manual methods that require extensive experience, our automated product analyzes the entire system quickly and efficiently.

1.2 Project Goals

The goals for this project include the following:

Table 1: Goals

<table>
<thead>
<tr>
<th>Goal</th>
<th>Text/Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assist</td>
<td>User Interface (UI)</td>
</tr>
<tr>
<td>Dispatcher</td>
<td>• Text: Provide simple interface to dispatcher</td>
</tr>
<tr>
<td></td>
<td>• Metric: User study and sponsor feedback (rating)</td>
</tr>
<tr>
<td>Functional</td>
<td>• Text: Choose what vehicles to move and where</td>
</tr>
<tr>
<td></td>
<td>• Metric: Verification and User Study</td>
</tr>
<tr>
<td></td>
<td>(rating and seconds)</td>
</tr>
<tr>
<td>No Time</td>
<td>• Text: No increase in processing time</td>
</tr>
<tr>
<td>Increase</td>
<td>• Metric: User Study (seconds)</td>
</tr>
<tr>
<td>Compare</td>
<td>• Text: Compare the coverage of current to ideal</td>
</tr>
<tr>
<td>Coverage</td>
<td>• Metric: User Study and Sponsor</td>
</tr>
<tr>
<td>Quickly</td>
<td>Feedback (rating and seconds)</td>
</tr>
</tbody>
</table>

1.3 Scope

An integrated software application was created to quickly and efficiently analyze the system state and recommend redeployment. The system inputs are (1) a standard input file describing the specific city instance, (2) an ideal solutions file generated from historical data, and (3) a Geographic Information System (GIS) simulator file to provide the real-time capture of vehicles’ positions in the city. The system outputs are displayed on a Graphical User Interface (GUI) providing (1) overall coverage probabilities for current and ideal, (2) a comparison of the current vehicle locations and ideal solution, and (3) a redeployment recommendation (i.e. each vehicle’s current and final location as well as expected travel time).
2 INITIAL DEVELOPMENT

2.1 Research

To understand the EMS redeployment process, the group made a site visit to Tucson, AZ 911 Dispatch Communications Division to observe the real system [5]. Although the Tucson EMS system is a fire-based model (instead of third service), significant observations and information on the general EMS process was gained. Redeployment was generally accomplished with a physical or computer map at the dispatcher’s desk. However, fire based deployment models do not move emergency vehicles often. Through additional discussions with the Communications Coordinator, David Jones, it was said there is a 50% retention rate of dispatchers in the division. This high turnover rate is attributed to the high stress associated with the position as well as the unpopularity of shift work.

Weekly client meetings were held to gather and define a customer requirements set with Dr. Jeff Goldberg and Mr. Dave Lindberg [4]. Both mentors provided experience, historical data, and research papers in the project domain as background information in the research field of emergency vehicle management.

A valuable introduction to dynamic model development was provided in A Dynamic Model and Parallel Tabu Search Heuristic for Real-time Ambulance Relocation (2000) by Gilbert Laporte, Michel Gendreau and Frederic Semet [3]. The paper focused on vehicle redeployment through maximizing emergency vehicles’ double coverage of city grids. The most important concepts discussed in this paper are the use of a dynamic, rather than scheduled, redeployment of emergency vehicles in a city and the constraints used to validate the system to the real world.

2.2 Requirements

A customer requirements set was collected from the site visit, additional research, and client meetings. This requirements set is listed in the complete project documentation available upon request. Next, system requirements were derived in four categories: mandatory, compatibility, performance, and code conventions. Key project requirements include the following.

2.2.1 Title: Real-time data consideration
2.2.1.1 Text: The system shall consider input of real-time data for vehicle location.
2.2.1.2 Rationale: This is the only methodology feasible for the project stakeholders, as solution validation is critical. If real-time data is not considered, the solution is not valid for the current system state.
2.2.1.3 Trace to: Customer Requirement (CuR) 3
2.2.1.4 Priority: High

2.2.2 Title: City Independence
2.2.2.1 Text: The system shall be designed without a specific city instance.
2.2.2.2 Rationale: This is important for reuse and sale of the product. The product must be capable of running on all possible cities given the correct inputs defined.
2.2.2.3 Trace to: CuR10.1
2.2.2.4 Priority: Low

2.2.3 Title: Automation.
2.2.3.1 Text: The system shall complete all calculations after a given start input from the user automatically (without human intervention).
2.2.3.2 Rationale: A main goal of the system is to analyze the entire system quickly and efficiently and to assist the dispatcher. This is best done through automation of computational tasks.
2.2.3.3 Trace to: CuR6
2.2.3.4 Priority: High

2.2.4 Title: Solution Presentation.
2.2.4.1 Text: The system shall calculate and present a solution within a Short Time Interval (STI).
2.2.4.2 Rationale: Time is a crucial element in system performance. With the use of real-time data (see 2.3.1.), it is necessary for the solutions to be calculated in a STI, otherwise the solution will become obsolete.
2.2.4.3 Trace to: CuR5
2.2.4.4 Priority: High
2.2.4.5 Metric:
   2.2.4.5.1 Lower Bound: 1 second
   2.2.4.5.2 Upper Bound: 1 minute

2.3 Functional Analysis

Upon completion of the above system requirements, a functional analysis was performed on the system in relation to the primary user, the Dispatcher, with consideration of all system stakeholders.
The above use case diagram (see Figure 1) was developed along with all corresponding narratives for each use case shown and descriptions of the actors, object, entity, and boundary class instances specified. This functional breakdown permitted the group to correctly represent the system and all necessary functions in a concise and communicative manner. As can be seen in the diagram above, the functions (use cases) are the focus of this analysis and placed in the center of the diagram.

3 ELABORATION

3.1 Mathematical Model Development

Upon completion of the initial system analysis, a mathematical model was derived and developed for the problem to be solved.

For any number of vehicles, an optimal deployment can be calculated by maximizing the expected coverage with respect to the vehicles’ positions. For instance, if there are ten vehicles available, it is possible to calculate the ten best locations in the city for the vehicles to be stationed. When another vehicle becomes available, the user would then move to the eleven vehicle solution. Because the population distribution of a city changes throughout the day, separate solutions must also be calculated for each time segment, usually taken as one hour increments. For this project, three time increments, 1:00am-8:00am, 8:00am-19:00pm, and 19:00pm-1:00am, were chosen. Therefore, if a city has \( v \) vehicles and \( m \) time segments, \( v \times m \) optimal deployments must be calculated in advance.

Once the appropriate ideal deployment is known, the system determines the travel time for each vehicle from its current location to all possible final/ideal posts. These travel times are either determined from historical data provided by the city, or later, dynamically estimated through a GIS software package. For \( v \) vehicles, this results in a square matrix of height and width \( v \), since the optimal deployment has exactly enough posts for the number of vehicles available.

Using this travel time matrix, the model (Equation 1) becomes an assignment problem, whereby the objective is to minimize the maximum travel time. By making a square matrix of binary decision variables with the same dimensions as the travel time matrix, and enforcing the constraint that each row and column of the decision matrix must have a sum of one, an off-the-shelf optimization package can be used to solve the assignment problem in a timely manner. This will provide the dispatcher with a list of recommended vehicle movements which, if followed, will arrange the vehicles in the pre-calculated ideal deployment for the number of available vehicles.

These recommendations, as well as information on the change in expected coverage, are then presented to the dispatcher to assist in his or her decision. It is ultimately the dispatcher’s decision whether or not to use the system’s recommendations.

3.2 Alternatives

There were five architectural alternatives formed based on the requirements. The alternatives were: (1) Do Nothing, (2) Hand Calculations, (3) C++/GIS Suite All-in-One Implementation, (4) Java/Web App, and (5) Integrating Third Party Apps via Macro Interface. The Do Nothing alternative is used to confirm that the system is meeting the needs of the stakeholders by truly providing value over the existing system. Hand Calculation would require the performance of a limited set of probability calculations for the decision by hand each time. The C++/GIS Suite All-in-One Implementation is a software package that uses a GIS suite for current vehicle location as well as an off-the-shelf optimization package to model the city and return a decision. The Java/Web App is similar to the C++ alternative, just in the Java programming language. The Integrating Third Party Apps via Macro Interfaces is an integration of multiple third party applications with macro interfaces to return a decision.

The architectural decisions made in this tradeoff study were based on the research performed by the group and its sponsors as well as the skill sets and experience pools of the members. The numbers presented in Table 2 reflect utility values averaged from all group members. A tradeoff analysis was done to compare the alternatives with the following criteria: performance, cost, schedule, and risk. The Monotonic Increasing scoring function was used for this analysis and all score values were fitted to this distribution. The main criteria had the following sub criteria: Performance – (1) execution speed, (2) accuracy, and (3) ease of use. Cost – (1) development and (2) maintenance. Schedule – (1) development time, (2) maintenance frequency, and (3) maintenance duration. Risk – (1) failure rate. The most important criteria, which

\[
\begin{align*}
\text{min} & \quad \max_v \sum_p x_{v,p} t_{v,p} \\
\text{subject to:} & \\
\sum_v x_{v,p} &= 1 \quad \forall p \\
\sum_p x_{v,p} &= 1 \quad \forall v \\
\end{align*}
\]

\( t_{v,p} \) = Time for vehicle \( v \) to reach post \( p \).

\( x_{v,p} \) = Binary decision variable; 1 if vehicle \( v \) is being deployed to post \( p \).

Equation 1 Model
were weighted more than the others, were execution speed, accuracy, and failure rate.

Table 2: Tradeoff Analysis Results Summary

<table>
<thead>
<tr>
<th>Alternative</th>
<th>Final Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do Nothing</td>
<td>0.78</td>
</tr>
<tr>
<td>Hand Calculation</td>
<td>0.52</td>
</tr>
<tr>
<td>C++</td>
<td>0.88</td>
</tr>
<tr>
<td>Java</td>
<td>0.90</td>
</tr>
<tr>
<td>Macros</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Although the Java alternative scored the highest, the C++ alternative was the chosen because of quicker computation time, sponsor approval and team development familiarity.

4 IMPLEMENTATION

The system implementation chosen, as above, was C++ programming in Microsoft Visual Studio .NET 2003 employing the Frontline Systems Inc. Solver SDK for C++ [1]. Utilizing the Model-View-Controller design pattern, the group designed the GUI (View) component in the Visual Studio .NET suite, as well as the Controller and Model components. To design the software architecture before beginning the code, the team developed a UML class model, including a class diagram, in Sparx Systems’ Enterprise Architect software (see Figure 2) [2].

Figure 2 Class Diagram

The Controller class, in the middle of the above diagram, runs the entire program using the following general process:

1. Wait for user input.
2. Read user specified files to system model classes: Defines city. (see Appendices A1 and A2)
3. Wait for user input.
4. Read user specified GIS Simulator file: Defines current vehicle locations. (see Appendix A3)
5. Compute travel time matrix

6. Pass travel time matrix to Optimizer
7. Solve
8. Return and display results to user on GUI

By using the above schema, steps 4-8 are counted as the Calculation Step and 1-3 as the One-Time City Initialization. Therefore, in a deployment setting, only the final steps 4-8, which are all automated without human intervention, would be performed on a regular basis, whereas steps 1-3 would occur whenever the demand model or ideal solution set for the city changes. This separation allows simple measurement for requirement 2.3.4. Solution Presentation.

Figure 3 GUI Design

5 VERIFICATION AND VALIDATION

System validation occurred through weekly client meetings with Dr. Goldberg, discussing historical data, and his wealth of experience with other similar projects.

Additionally, the group collaborated with Rashid Al-Jalahema, a Master of Science in Industrial Engineering student of the University of Arizona, as we finalized our model. Al-Jalahema also works with Dr. Goldberg on similar projects; however, much of his work is in different sections of the overall EMS system. For example, the ideal solutions provided to the system are from a program that he developed. Since this project works in conjunction with ours, it was important to communicate compatibility issues in code design and verifying all input files were in the CSV file format.

Finally, much of our visual design has been validated with meetings and demonstrations to faculty in University of Arizona Systems and Industrial Engineering department as well as members of industry known to Dr. Goldberg. This validation process was invaluable to the group in confirming the use of the system for real-world applications.

In addition to validation, numerous verification methods were employed for the project. The testing phase of the project included unit testing (throughout development as well as integrated unit testing), boundary constraints, visual/map inspection with deterministic data, and trial
cities. All these methods together have verified the systems is in compliance with the requirements presented above.

6 RESULTS

After writing an implementation of the model using Microsoft Visual C++ .NET and Frontline Systems Inc. Solver SDK for C++, the system was tested using various test cases. Several of these were small examples with 11 vehicles or less. The purpose of these tests was to confirm that the system could reliably handle simple cases, thus proving that the basic mechanics of the system functioned properly.

After initial debugging, the system was able to handle all small-scale test cases with less than 9 vehicles in under 30 seconds of calculation. When the system is run for a case with 10 or more vehicles the processing time increases exponentially. This is because the number of decision variables in the optimization problem increases by a power of 2 as the number of vehicles increases. A 10 vehicle problem took approximately 2 minutes to solve, and an 11 vehicle problem required about 9 minutes of processing time. The data for these tests was provided by the Region of Peel, Ontario, Canada.

Testing also confirmed that the system has high memory requirements. Because of the nature of the problem, the amount of data that must be stored in system memory grows exponentially with the number of vehicles. As a result of this, large problems will require several hundred megabytes of system memory. Testing showed that 1 gigabyte of system memory would be sufficient for most problems.

7 CONCLUSION

Due to the high processing time, the system is not suitable for use in solving problems with large numbers of vehicles. This will not be a problem since the system will likely only be used in situations where most of the vehicles in a city have been deployed. If a majority of a city’s EMS vehicles are currently available, there would not be a significant increase in coverage as a result of redeploying, and therefore no reason to assume the costs associated with redeployment.

The real value of the system is in assisting in the decision making process when most vehicles are not available. If a majority of a city’s vehicles are not available to respond to calls, the location of the remaining vehicles can be crucial to saving lives. The system is perfectly suited to this situation, as it can quickly and efficiently determine a globally optimum solution in a reasonable amount of time for problems involving a small number of available vehicles.

8 RECOMMENDATIONS

The basic functionality of the system was shown to be valid in small-scale testing, though there is room for improvement when solving problems with large numbers of vehicles. It is possible that different commercial solvers would have different ways of managing the optimization data (e.g. through the use of sparse-matrix arrays), thus avoiding the need for large amounts of system memory. In addition, solvers that utilized multi-threaded programming techniques would be able to take advantage of multi-core processors or distributed computing networks, further decreasing processing times. It is recommended that additional solvers be investigated for these reasons.

Finally, another possibility that the team would recommend is researching a distributed network of computers where the problem would be broken down into smaller, more manageable problems to be solved on separate computers and then combined into one final solution. This possibility stems from the fact that the minimax optimization problem will naturally lead to solutions where vehicles are moved only within their local area, thereby possibly allowing the problem to be broken down into sub sections and subsequently recombined. Such a system would likely reduce the processing time for overall solution, allowing large-scale problems to be solved in a reasonable amount of time.

In addition, the processing time of such a system would likely increase linearly with the number of vehicles, instead of exponentially as is the case with the current system. This is because increasing the size of the problem increases the number of sub-problems that must be solved without significantly increasing the size of the sub-problems. This recommendation could provide the added flexibility of a scalable system depending on the size of the city’s vehicle fleet, however further domain research is required.

9 FUTURE WORK

The information from this entire project will be turned over to Dr. Goldberg on May 2, 2006. He will utilize the EMS Redeploy system as a basis for future projects in the design space of EMS systems. Should this product be incorporated into a real system, it will have direct connection with that system’s GIS, Computer Aided Dispatch (CAD) and Automatic Vehicle Location (AVL) packages. Within an integrated software package, the EMS Redeploy system would automatically run in conjunction to a real time information feed.
Acknowledgments

We would like to thank Dr. Jeff Goldberg, Mr. David Lindberg of Rural/Metro Corp. and Mr. Rashid Al-Jalahema of the Department of Systems and Industrial Engineering at the University of Arizona for their help and support in the development and completion of the project. We would also like to thank David Jones, the Communications Coordinator from the City of Tucson for taking the time to introduce us to the EMS system at Tucson 911 Dispatch Communications Division.

Appendix

A1 Input File Format

<table>
<thead>
<tr>
<th>Zones</th>
<th>Posts</th>
<th>Call Types</th>
<th>Total Time</th>
<th>Average Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>803</td>
<td>92</td>
<td>1</td>
<td>17884800</td>
<td>4613</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1055.66</td>
<td>4613</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1055.66</td>
<td>4613</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1055.66</td>
<td>4613</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1055.77</td>
<td>4613</td>
</tr>
</tbody>
</table>

This is an example of the first five lines of a standard input file as generated from a the historical call database of a city EMS department. The first line contains (L to R): # of zones, # of posts, # of call types, total time in seconds. The second set of lines contains (L to R): zone, call type, number of calls, average service time and an extra piece of data.

A2 Ideal Solution File Format

<table>
<thead>
<tr>
<th>Available Vehicles</th>
<th>Posts</th>
<th>Zone</th>
<th>Call Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.245761</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.416673</td>
<td>31</td>
<td>91</td>
</tr>
<tr>
<td>3</td>
<td>0.458367</td>
<td>31</td>
<td>91 42</td>
</tr>
<tr>
<td>4</td>
<td>0.498125</td>
<td>31</td>
<td>91 42 2</td>
</tr>
</tbody>
</table>

This is an example of the first four lines of an ideal solution file as generated from a the above standard input file. The format is (L to R): # of available vehicles in solution, followed by the posts where those vehicles are located.

A3 GIS Simulator File Format

<table>
<thead>
<tr>
<th>Available Vehicles</th>
<th>Zones</th>
<th>Posts</th>
<th>Customer Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>271</td>
<td>593</td>
<td></td>
</tr>
<tr>
<td>228</td>
<td>122</td>
<td>686</td>
<td></td>
</tr>
</tbody>
</table>

This is an example of a GIS Simulator file as manually generated to simulate the interaction with a GIS compatible CAD/AVL system. The format is (L to R): # of available vehicles, followed by the zones where those vehicles are located.

References


Abbreviation and Acronyms

AVL: Automatic Vehicle Location
CAD: Computer Aided Dispatch
CuR: Customer Requirement
EMS: Emergency Medical Services
Fire-Based: A model used for Emergency Medical Services that combines the dispatch of Medical, Police and Fire systems (also see Third Service).
GIS: Graphical Information System
STI: Short Time Interval
Third Service: A model used for Emergency Medical Services that separates Medical from Police and Fire based distribution systems (also see Fire Based).

Author Biographies

Tim Belshe is a systems engineering senior at the University of Arizona with an emphasis in numerical methods and optimization. He has worked as an independent contractor on several research projects involving integration of neural networks with optimization software. He will be graduating in May 2006 and will be working with ECIII, LLC at the US Army Yuma Proving Grounds. Tim can be contacted via email at <tbelshe@gmail.com>.

Jeff Goodhue is a systems engineering senior at the University of Arizona with an emphasis on software systems development. He has software design and development experience from Lockheed Martin, NASA Space Grant Research and IBM. He will be graduating in May 2006. Jeff can be contacted via email at <Jeffrey.Goodhue@gmail.com>.

Jessica Yeh is a systems engineering senior at the University of Arizona with an emphasis in engineering management. She has database administration and project engineering experience from Qwest Communications and Lockheed Martin Integrated Systems and Solutions. She will be graduating in May 2006. Jessica can be contacted via email at <jjcyeh@gmail.com>. 