Proceedings of the 2nd World Congress on Civil, Structural, and Environmental Engineering (CSEE'17)
Barcelona, Spain - April 2 - 4, 2017
Paper No. ICTE 105
ISSN: 2371-5294
DOI: 10.11159/icte17.105

Efficient Real-Time Allocation of Patrol Cars in Traffic Management

Hussain Al-Harthei[1], Viktor T. Toth[2], Atef Garib[3], Samy A. Mahmoud[2]
[1]Tatweer-Co,
Abu Dhabi UAE
[2]Department of Systems and Computer Engineering, Carleton University
Ottawa ON CANADA
[3]Ministry of the Interior and Abu Dhabi Police
Abu Dhabi UAE

Abstract - Efficient allocation and dispatch of emergency vehicles is a common problem encountered by law enforcement and public safety agencies. An example of such agencies is the traffic patrol department, which has the responsibility of regulating and managing the movement of road traffic as well as responding to road incidents in a timely manner. A brief analysis of the patrol car allocation problem is presented. The problem involves determining the minimum number of patrol cars required to maintain a required service standard and the optimum location of patrol cars to achieve the best response times. An algorithm for determining the optimal number of patrol cars and the prepositioning of the vehicles is presented. The algorithm combines an analytical model with a Monte Carlo simulation, keeping execution times manageable. Even when applied to large geographic areas, the computational requirements of the algorithm remain modest.

Keywords: optimization, geographic information systems, location-allocation, emergency service, traffic police, police patrol car

1. Introduction

The current font size is: 10.95pt

A common problem facing agencies tasked with dispatching vehicles for emergency response is the appropriate allocation of resources to ensure that the requisite number of emergency vehicles are available to respond to incidents within a predetermined response time limit. An example of such agencies is the traffic patrol department, which has the responsibility of regulating and managing the movement of road traffic and responding in a timely manner to accidents. Overallocation of patrol resources is a waste of personnel and equipment. On the other hand, underestimating the required patrol resources impedes the ability of the department to respond to many accidents in a timely manner.

As illustrated in this paper, it is possible to provide an algorithmic solution to this problem using the capabilities offered by geographic information systems. The analysis, however, begins with a practical and realistic formulation of the problem. Theoretically, the number of patrol cars required so that all simultaneous accidents can be attended to 100% of the time regardless of the number of such simultaneous accidents is bound to be very large indeed. However, the probability of a large number of independent simultaneous accidents taking place throughout a geographic region is very small. Thus it is more meaningful in practice to define the required patrol resources in accordance with a probabilistic formulation as follows:

- First, determine the required number of patrol cars so that in the overwhelming majority of cases involving simultaneous accidents at least one patrol car will be available to deal with each accident. The limit can be stated in specific terms. For example, it is reasonable to require that there should be no more than one in a hundred thousand incidents when a patrol car is not readily available for dispatch;
- Second, determine the required number of patrol cars such that when an accident is reported, a patrol car that is promptly dispatched should arrive at the accident site within a time duration that does not exceed a prescribed limit for a set percentage of the time.

The above two requirements are not potentially in conflict. Rather, they represent two separate problems. The first requirement does not prescribe the amount of time that a patrol car has available to arrive at an accident scene, only that a car must be available for dispatch to the accident site. This requirement can be met even if all the patrol cars are stationed at some remote site of the city. The second problem, however, requires pre-positioning patrol cars so that they can reach an accident site within a specified short period of time. In many practical situations, the geographic map, routes and traffic flow patterns of the zone under consideration are such that the number of patrol cars required in order to meet the second requirement, namely the travel time limit, will be higher than the number of cars that meet the first requirement of dealing with simultaneous accidents.

In the analysis presented in this paper, we solve these two problems separately by determining the number of patrol cars needed to meet each requirement independently. Obviously the larger of the two estimates will be adopted since it meets both requirements. As a first step, the type of data required in order to solve both problems must be determined.

2. Problem preparation, initial data

Our starting point is an electronic representation of the street-level map of the city or region, in which the area is covered by a suitable number of nodes representing the intersections.

This analysis does not depend on the level of granularity that is selected, but given the power and capacity of present-day computers, it is possible to make use of a highly granular intersections data set. Instead of dividing the city into larger zones, intersection-level granularity may be feasible, in which each node shall represent a major intersection. A large city may have several ten thousand intersections. However, in practice most large metropolitan areas can be reasonably represented by fewer than one thousand major intersections.

The nodes must be connected in the form of a directed graph, representing all possible one-way routes that connect each pair of nodes. This is necessary for two reasons. First, the routes may indeed be unidirectional, composed of one-way roads, multilevel road structures and expressway intersections. Second, the routes must be treated as asymmetrical since the time it takes to travel from node i to node j is not necessarily equal to the time it takes to travel from j to i under varying traffic conditions in each direction.

With each node i, we associate a probability p_i that an accident is presently occurring in the vicinity of that node. This information is assumed to be readily available from official reports of accident statistics (This part of the analysis could be further refined by taking into account statistical information about accident distribution, e.g., in the form of standard deviations). The probabilities p_i are functions of time, thus they may also be written as $p_i(t)$. However, we expect that the data set would be essentially sampled, hence $p_i(t)$ would be tabulated at different times of the day.

While p_i represents the probability of *exactly one* accident at node i, we must take into account the fact that multiple accidents may be occurring in the vicinity of a node at the same time. Under the reasonable assumption that multiple accidents in the vicinity of a given intersection occur independently, the probability of *exactly two* accidents will be p_i^2; *exactly three*, p_i^3; and so on. The probability of one or more accidents, in turn, is given by

$$P_i = \sum_{k=1}^{\infty} p_i^k = \frac{p_i}{1-p_i}. \tag{1}$$

From this, we can also tell that necessarily, $0 \le p_i \le 1/2$. When $p_i = 1/2$, $P_i = 1$. Furthermore, the probability of no accident occurring at node i is given by $1 - P_i$.

In addition to knowing the probability of accidents at each node, we also need to know the time it takes to travel between each ordered pair of nodes along the quickest route. We assume that the input data provides such times T_{ij}, to travel from node i to node j, for pairs of nodes that are directly connected (e.g., neighboring intersections; adjacent zones). As with the p_i, the values of T_{ij} may be functions of time, as travel times along various routes may vary throughout the hours or even minutes of the day, and may very well vary depending on the specific days of the week. To simplify the notation, the variables p_i and T_{ij} shall be used in the rest of this paper without explicitly indicating functional dependence on the time t. It shall be understood that any calculation presented here should be repeated as many times as needed, to achieve the required granularity in time (e.g., representing peak and off-peak hours).

3. The routing problem

We note that initially, the matrix T_{ij} may be sparse, containing travel times only between adjacent nodes. Therefore, the first task is to populate the matrix T_{ij} so that we know the minimum amount of time it takes to travel from any node i to any node j (at any time of the day), even when the nodes are not adjacent.

The literature of this so-called *Vehicle Routing Problem* is rich [1, 2, 3, 4, 5, 6, 7, 8]. Most of the literature deals with complex scenarios in which vehicles have different capabilities and carry varying loads, as they are required to visit several locations. The problem is often a *multi-objective optimization problem* [9, 10], also called *multicriteria optimization* [11], where multiple, and in some cases contradictory objectives must be optimally or suboptimally met. The computational complexity of these problems is such that an exhaustive search for the optimal solution is not feasible.

The vehicle routing problem considered here is comparatively simple. We seek to determine the path of least weight (shortest duration) from any node to any other node in our directed, weighted graph. This is actually a well-understood problem for which relatively simple algorithms are readily available [12]. A particular algorithm that can serve as a model implementation is provided in Figure 1 (a).

This algorithm is moderately efficient and can be applied even when the number of nodes is very large. The algorithm only needs to be run to compute the shortest travel times when traffic conditions change and the values T_{ij} need to be updated. The resulting values of T_{ij} can be saved in a database, readily available for use when needed, and only need to be changed if traffic information about the city changes and a new set of T_{ij} for directly connected nodes becomes available.

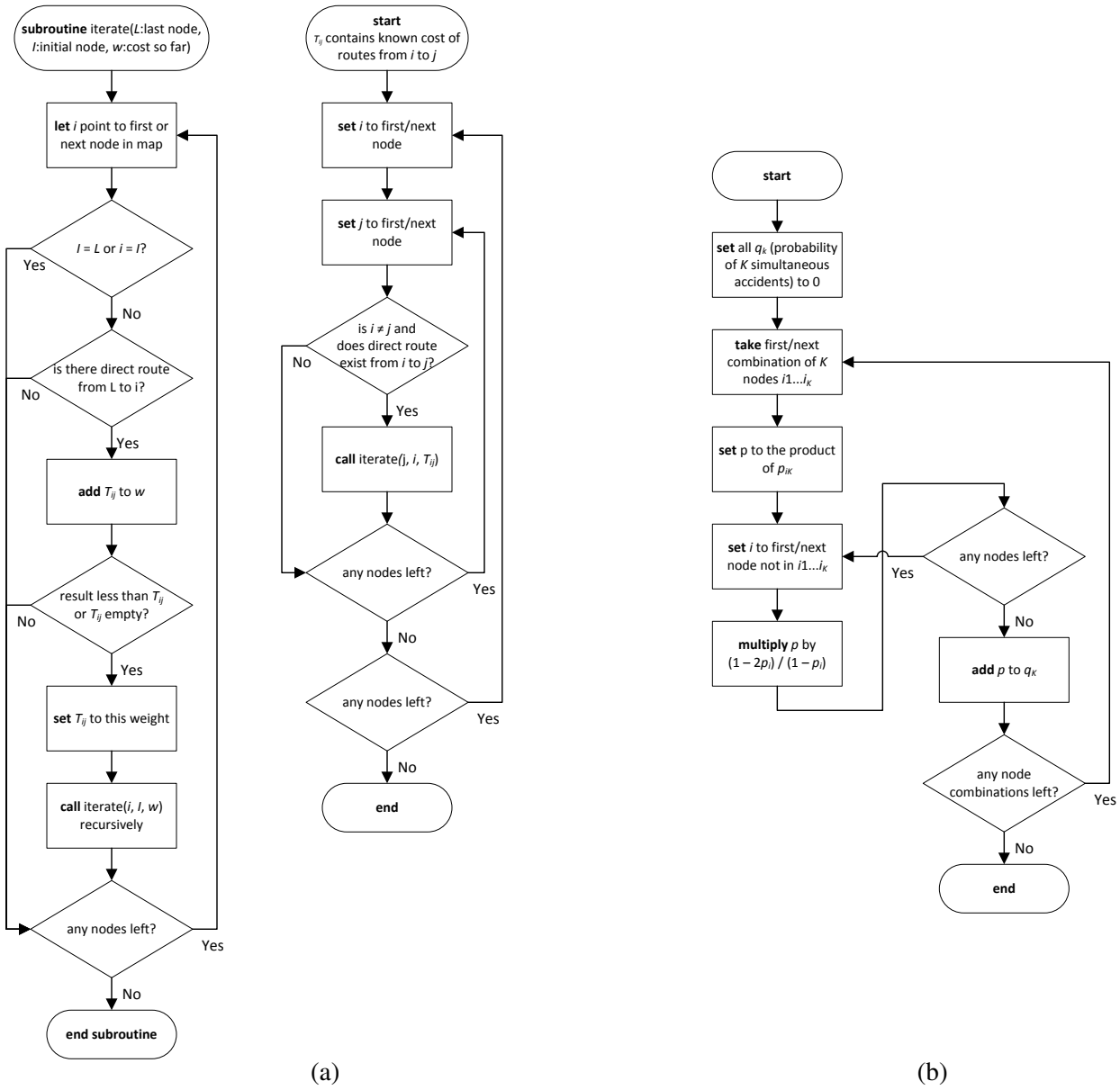

Fig. 1: (a) Algorithm to determine the shortest path between any pair of nodes in a directed, weighted graph. (b) Algorithm to determine the probability q_K of exactly K simultaneous accidents.

In summary, at this point we assume knowledge the probability p_i of a single accident taking place at node i; and we have an algorithm to compute T_{ij} for any ordered pair of nodes i, j, giving us the duration of travel from node i to node j.

4. Determining the minimum number of patrol cars

In this section, analysis is presented for computing the minimum number of patrol cars that are needed to ensure that the probability of having more accidents than patrol cars will be less than a prescribed value $p_{\min}$.

Conceptually, this can be done by taking every possible number K between 1 and ∞ representing the number of simultaneous accidents in the entire system, apply a partition function to distribute K accidents across all possible combinations of nodes, and then compute the product of the corresponding values of p_i to obtain the desired probability. The problem with this "brute force" approach is that the requisite number of calculations will be extremely large. Specifically, for N nodes and

K accidents, it is given by

$$C = \binom{N+K-1}{K}, \tag{2}$$

which, for large N and K, is approximately an exponential function of K. In other words, for large K (and in this context, $K = 100$ is already large), the computational cost of this "brute force" approach is prohibitive. Problems such as this one, which cannot be solved in a time that is a polynomial function of a characteristic size, are known as *NP*-hard problems.

Specifically, given N nodes and p_i, P_i ($i = 1...N$) as defined in Eq. (1) above, the probability $P(0)$ of there being zero accidents is given by

$$P(0) = \prod_{i=1}^{N}(1 - P_i). \tag{3}$$

The probability of there being exactly one accident is given by

$$P(1) = \sum_{i=1}^{N} p_i \times \prod_{\substack{j=1 \\ j\neq i}}^{N}(1 - P_j) = P(0)\sum_{i=1}^{N}\frac{p_i}{1-P_i}, \tag{4}$$

and for exactly two accidents,

$$P(2) = \frac{1}{2}\sum_{i=1}^{N}\frac{p_i}{1-P_i}\left[P(1) + P(0)\frac{1-2P_i}{1-P_i}p_i\right], \tag{5}$$

with the formulae becoming increasingly complex for higher values of K in $P(K)$. A generic "brute force" algorithm for computing the probability of K simultaneous accidents, usable for small values of N and K, is given in Figure 1 (b).

Instead of using a computationally prohibitive, exhaustive analysis, we opted for a Monte-Carlo simulation. Using a suitably uniform random number generator in conjunction with the probabilities p_i, we generate a large number of accident scenarios, and in each case we count the number of simultaneous accidents. This is a very straightforward method, the computational complexity of which scales linearly with N, which is desirable in practice. Therefore, a very large number of simulations can be run in a short amount of time, making the Monte-Carlo method accurate. The actual number of Monte-Carlo simulations should be at least approximately an order of magnitude greater than $p_{\min}^{-1}$.

An algorithm for a model implementation is presented in Figure 2 (a). One element of this algorithm that may require a bit of additional explanation is the step that sets the value of k. Given a uniformly distributed random number $0 \leq p_{\text{rnd}} < 1$, we set the number of accidents k at a given node using the formula

$$k = 1 + \frac{\ln p_{\text{rnd}}}{\ln p_i}, \tag{6}$$

or, after rearranging and exponentiating,

$$p_i^{k-1} = p_{\text{rnd}}. \tag{7}$$

This expression allows us to map a uniformly distributed random number p_{rnd} to the probability distribution given by p^{k-1} that represents the probabilities of exactly k accidents occurring.

The outcome of this Monte-Carlo simulation is a number K, which is the number of accidents that will not be exceeded with a probability greater than $p_{\min}$. Consequently, K is the number of patrol cars needed to satisfy the first of our two principal requirements.

5. Patrol car pre-positioning

The second principal requirement concerns the timely dispatch of patrol cars. To meet this requirement, the patrol cars must be positioned in the city so that at least one vehicle will be sufficiently close to an accident site and, when dispatched, it will reach that site within the prescribed time limit.

Problems of this nature have been extensively studied in the literature. There are studies exploring the problem in a broad range of applications that include military operations [13], humanitarian logistics [14], and disaster recovery [15]. In the field of operations research, it falls under the class of problems known *Facility Location Problem* [16].

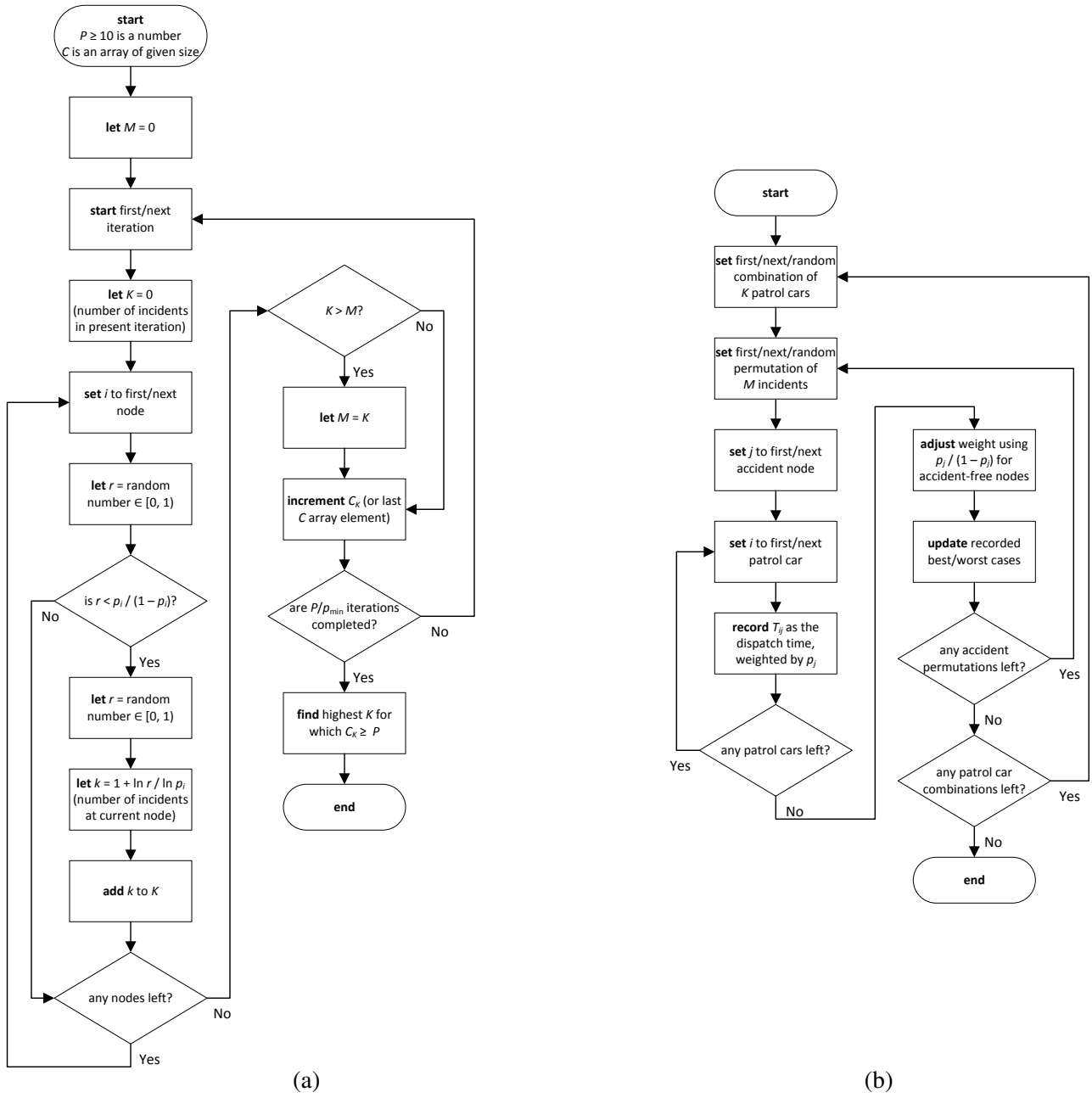

Fig. 2: (a) Algorithm to determine the maximum number of simultaneous incidents at a given probability. (b) Brute force method to enumerate all accident combinations.

The Facility Location Problem is also an *NP*-hard problem. This means that the use of any exhaustive, or brute force approach, such as that depicted in Figure 2 (b), is not a practically viable option. To illustrate this point, consider a network of roads with N nodes and M accidents. The total number of possible permutations of accident locations is given by $N!/M!$. This figure must be multiplied by the total number of combinations of K patrol car locations (with two or more patrol cars permitted to exist at the same location), yielding the following number of total permutations C:

$$C = \frac{N!}{M!}\binom{N+K-1}{K}. \tag{8}$$

As an example, the values of $N = 10000$, $K = 100$ and $M = 100$ will result in $C \simeq 5.35 \times 10^{35743}$, which is a figure

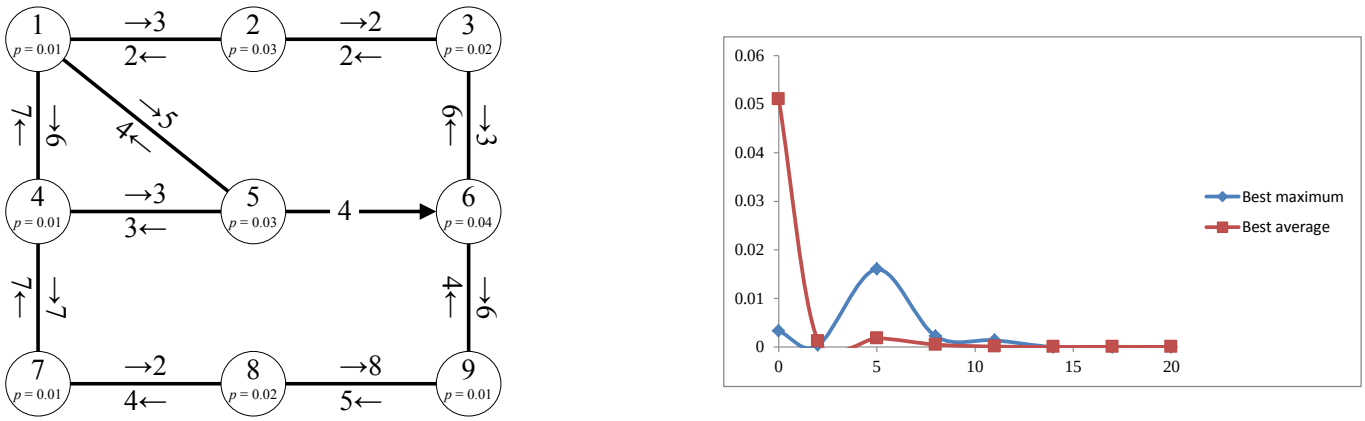

Fig. 3: A simple worked example, and corresponding weighted, binned distribution of response times for two combinations of patrol car locations.

far beyond even astronomical magnitudes. Even a more modest case, characterized by $N = 100$, $K = 10$ and $M = 10$, yields $C \simeq 1.09 \times 10^{165}$, thus the computational effort to do an exhaustive search remains prohibitive. This motivates the need to search for suboptimal solutions using simulations and heuristic algorithms. In very simple cases, the results of such algorithms can be compared to the results obtained by an exhaustive search. These comparisons were insightful in revealing the effectiveness and limitations of non-exhaustive methods.

As an example, we considered a network of 9 nodes. Both a Monte-Carlo simulation and a brute force direct calculation confirmed that at any given time, the probability of 5 or more simultaneous accidents is less than 1 in 100,000.

The algorithm iterates through patrol car distributions (all distributions in brute force mode, or a random subset of distributions in Monte-Carlo mode). For each patrol car scenario, the algorithm iterates through accident scenarios (again, either exhaustively in brute force mode, or using a random subset in Monte-Carlo mode.)

For each patrol car distribution, the algorithm records two values:

- The *average response time* is defined as the travel time of a dispatched patrol car to reach an accident site, averaged over all accident scenarios for the given patrol car distribution;
- The *maximum response time* is defined as the longest time it takes to dispatch a patrol car to any accident site, calculated over all accident scenarios for the given patrol car distribution.

The algorithm then ranks patrol car distributions according to these two values. When the run is completed, four patrol car distributions are displayed: the distributions with the best (shortest) and worst (longest) average response time, and the distributions with the best and worst maximum response time.

When we perform an exhaustive enumeration of all scenarios with 4 or fewer accidents for the 9-node example, we find that the two rankings are quite different: scenarios with good average response times produce mediocre results when characterized by the maximum response time and vice versa.

Specifically, the worst (longest) maximum response time (20 time units) is obtained if all patrol cars are placed at remote, not well-connected nodes. The same scenario also yields the worst average response time (14 time units).

Conversely, the best (shortest) maximum response time (12 time units) is obtained when all police cars are placed at a central, well-connected node; the corresponding average response time is also much improved (5.5 time units.)

The most diverse distribution of patrol cars occurred when the best (shortest) average response time (3.5 time units) was sought. Although in this case, the maximum response time (16 time units) is high, it should be noted that scenarios that correspond to this maximum response time occur very infrequently.

In order to analyze this aspect, we binned the result by response time. The outcome (see Figure 3) proved to be interesting. Even though the "best average" scenario yielded a mediocre maximum response time, the frequency or probability of this occurrence is very small. Meanwhile, the "best average" scenario significantly outperforms the "best maximum" scenario in nearly all bins.

This suggests that, although ultimately the choice between improving averages vs. improving maximum response times is a policy decision, a preferred distribution of patrol cars should aim to optimize the average response time at the cost of allowing longer maximum response times, as the latter will occur very infrequently.

Unfortunately, as indicated above, an exhaustive search is not feasible when real-life problems are analyzed with as many as $10^3 \ldots 10^5$ nodes and hundreds or more patrol cars. Instead, it is necessary to adopt an approach that is probabilistic in nature while yielding suboptimal solutions that are reasonably close to the theoretical, optimal solution most of the time.

Such a solution may begin with a heuristic algorithm to find a suboptimal set of K patrol car locations. As an initial, trial solution, obvious locations may be chosen: for instance, the patrol cars may be located at or near police stations, positioned in administrative districts, or near locations that are known for frequent accidents. Vehicles may even be assigned randomly at or near specific, visible sites.

One specific algorithm utilizes a see-saw approach [17, 18] in which, as the first step, a coverage area is associated with each patrol car location. As the next step, the location of the patrol car within the coverage area is optimized. This process can be repeated iteratively until no further improvement is achieved.

Other approaches are also possible. For instance, a Monte-Carlo simulation can be used to find a good (albeit suboptimal) placement of patrol cars; subsequently, individual patrol car positions can be adjusted by moving them to adjacent nodes if the move yields an improvement.

It should also be mentioned that in real-life scenarios, pre-positioning may be constrained. Not all nodes may be suitable locations for pre-positioned patrol cars. A traffic authority may not wish to pre-position some or all of its vehicles, preferring instead to position them at suitably located stations or depots. There may be a preference to have vehicles on patrol, moving along specific routes instead of waiting for calls at stationary locations. All these policy decisions, along with the choice of criteria used to measure patrol car availability, will have an impact on the choice of algorithm and its efficiency.

6. Patrol car dispatch

The availability of the values T_{ij} makes the dispatch of patrol cars rather simple. When an accident occurs at node j, the only action needed is to find the location i where a free patrol car is present and for which T_{ij} is minimal.

A more sophisticated approach to patrol car dispatch would take into account real-time patrol car location and traffic information, compute the best route for each available patrol car to the accident site, and select the patrol car that can reach the site in the shortest amount of time. The methods and information needed to perform this function are readily available in the form of GPS routing and mapping software and performing the necessary calculations in real time is a simple exercise.

A more sophisticated algorithm may take into account strategic considerations, for instance not dispatching a nearby patrol car and leaving an area with no patrol car in the vicinity, when another patrol car with only a slightly longer response time can be dispatched from a more well-covered location. Such an algorithm may, for instance, rely on an *ad hoc* Monte-Carlo simulation performed using the real-time distribution of patrol cars in combination with known accident frequencies.

7. Algorithm summary

The full algorithm that determines the minimum requisite number of patrol cars and offers an optimal set of locations for patrol car pre-positioning can be summarized as follows:

1. Create T_{ij} and populate it with travel time information for all direct routes;
2. Populate p_i with accident probabilities at all nodes i;
3. Use a routing algorithm such as that shown in Figure 1 (a) to find routes for all other combinations of (i, j), and populate all missing values of T_{ij};
4. Run a Monte-Carlo simulation, such as that shown in Figure 2 (a), to determine the maximum number K of accidents that can occur with a probability $p_{\min}$ or greater;
5. Use an approximate algorithm like that described in Section 5 to determine an optimal distribution of K patrol cars.
6. If no satisfactory distribution is obtained, increase K.
7. Repeat the above for different times of day, as required.

Memory requirements to implement this algorithm can be significant if the patrol car coverage area is divided into a large number of nodes. For instance, with 10,000 nodes (the island of Manhattan contains approximately 12,000 intersections [19]), the array T_{ij} will contain 10^8 real numbers. While such an array is large, it can still fit easily into the operating memory of even a handheld computing device.

The execution time of the algorithm in Figure 1 (a) is a polynomial function of the number of nodes, and therefore the algorithm can be run efficiently even when the number of nodes is large. The Monte-Carlo algorithm of Figure 2 (a) also executes in polynomial time; in fact the execution time of this algorithm will be proportional to the number of nodes and inversely proportional to $p_{\min}$. The execution time required for patrol car pre-positioning depends on the specific algorithm chosen, and it is essentially a linear function of the number of nodes or number of patrol cars.

8. Conclusions

In this paper, a brief analysis of the patrol car allocation problem was presented. It was established that the problem consists of two parts. The first part is about determining the minimum number of patrol cars required to maintain a required service standard. The second part involves the determination of the optimum initial positioning of patrol cars to achieve the best response times.

A set of algorithms was proposed, based on solutions from the available literature but adapted to the specific requirements of patrol car allocation. Simple model implementations of the proposed algorithms were presented and tested using a test case with a small number of intersections. The algorithms, however, can be and have been extended and applied to large areas that include hundreds of major road intersections and complex traffic networks.

The analysis presented in this paper demonstrates that computing the shortest travel times between any two intersections can be done efficiently. Therefore, predicting patrol car response times, or finding the nearest patrol car and the shortest route for it to reach an accident site are computationally simple procedures.

Finding the minimum number of patrol cars required to ensure that sufficient number of cars are available to reach and deals with accident sites is an *NP*-hard problem that cannot be solved practically using "brute force" or exhaustive algorithms. Fortunately, a Monte-Carlo simulation appears entirely adequate for this purpose, and it can be performed using modest computational resources.

With a given number of patrol cars available, the biggest gain in response time efficiency can be achieved by pre-positioning patrol cars in specific locations. However, the pre-positioning approach may be constrained by policy and practical requirements. In general, an exhaustive search for the best combination of patrol car positions is also an *NP*-hard problem. Hence, it is a candidate for a solution using a combination of heuristic algorithms and Monte-Carlo simulation techniques. This last area can benefit the most from further research towards establishing the best optimization criteria to match policy objectives, and finding the most efficient approximate optimization algorithm that corresponds to the selected criteria.

References

[1] Z. Jianming, "Supply Allocation and Vehicle Routing Problem with Multiple Depots in Large-Scale Emergencies," in *Emergency Management*, B. Eksioglu, Ed. InTech, 2012.

[2] P. Kalina and J. Vokrinek, "Algorithm for Vehicle Routing Problem with Time Windows Based on Agent Negotiation."

[3] C. Koc and I. Karaoglan, "A Branch and Cut Algorithm for the Vehicle Routing Problem with Multiple Use of Vehicles," in *Proceedings of the 41st International Conference on Computers & Industrial Engineering*.

[4] R. V. Nikolaev, N. A. Nechval, and V. F. Strelchonok, "Model of Vehicle Allocation Problem with Uncertain Demands," *Transport and Telecommunication*, vol. 4, no. 2, 2003.

[5] W. B. Powell, "An Operational Planning Model for the Dynamic Vehicle Allocation Problem with Uncertain Demands," *Transpn. Res.-B*, vol. 21B, no. 3, pp. 217–232, 1987.

[6] W. B. Powell, J. A. Shapiro, and H. P. Simao, "An Adaptive Dynamic Programming Algorithm for the Heterogeneous Resource Allocation Problem," Department of Operations Research and Financial Engineering, Princeton University, Tech. Rep. CL-00-06, Nov. 2000.

[7] S. Rathinam, R. Sengupta, and S. Darbha, "A Resource Allocation Algorithm for Multi-Vehicle Systems with Non-Holonomic Constraints."

[8] S. L. Smith, "Task Allocation and Vehicle Routing in Dynamic Environments," Ph.D. dissertation, Mechanical Engineering, University of California, Santa Barbara, Sep. 2009.

[9] K. Bringmann, T. Friedrich, F. Neumann, and W. Markus, "Approximation-Guided Evolutionary Multi-Objective Optimization," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011, pp. 1198–1203.

[10] H. C. Lau, L. Agussurja, S.-F. Cheng, and P. Jin Tan, "A Multi-Objective Memetic Algorithm for Vehicle Resource Allocation in Sustainable Transportation Planning," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*.

[11] M. Ehrgott, *Multicriteria Optimization*. Springer, 2005.

[12] J. Nievergelt, J. C. Farrar, and E. M. Reingold, *Computer Approaches to Mathematical Problems*. Prentice-Hall, 1974.

[13] D. P. Johnstone, R. R. Hill, and J. T. Moore, "Mathematically Modeling Munitions Prepositioning and Movement," *Mathematical and Computer Modeling*, vol. 39, pp. 759–772, 2004.

[14] Özdamar, Linet and Ertem Alp, "Selected models, solutions and case studies in humanitarian logistics," *in preparation*.

[15] A. Pico, Y. Hui, and R. Tan, "Heuristics for Solving Problem of Evacuating Non-ambulatory People in a Short-notice Disaster," Dec. 2012.

[16] S. Li, "A 1.488 Approximation Algorithm for the Uncapacitated Facility Location Problem," Sep. 2011, Preprint submitted to Information and Computation.

[17] G. Jándy, *Rendszerelemzés és operáció-kutatás (Systems Analysis and Operations Research—in Hungarian)*. Műszaki Könyvkiadó, Budapest, 1980.

[18] F. E. Maranzana, "On the Location of Supply Points to Minimize Transport Costs," *Operational Research Quarterly*, no. 3, 1964, Cited in [17].

[19] `http://newyorkinplainsight.blogspot.ca/2010/04/how-many-street-corners-are-there-on_19.html`.