Evaluation of Police Patrol Patterns

Stephen R. Sacks
University of Connecticut

Introduction

The Desktop Hypercube is an interactive tool designed to help planners improve police services without additional response units (or to maintain a constant level of service with fewer response units). It calculates a number of variables, including travel times, workloads, and preventive patrol frequencies, thus giving planners the information they need to achieve various goals, such as reduction of response times and balancing workloads among response units. Using this tool, planners can experiment with various patrol patterns to find those that best meet their Department's goals. Most importantly, it allows planners to see clearly the trade-offs that are unavoidable.

The focus of this paper is on those inevitable trade-offs. For example, users of The Desktop Hypercube find that those patrol patterns which are best in terms of average response time don't do as well as others in terms of workload balance, or those that are best in terms of achieving a uniform response time across different parts of the city don't do as well as others in terms of minimizing inter-district dispatches. There is, of course, no perfect solution for this problem: the facts of the situation force us to balance competing goals. What we can do is build into The Desktop Hypercube a mechanism for explicitly weighting the alternative objectives.

The Desktop Hypercube

Police services are in a fundamental way different from many other government services. A public works department, for example, is likely to send rubbish collection vehicles to every house in a town on a regular, planned route. Workers go street by street, knowing that, say, on Tuesdays they collect rubbish on the west side of town and on Fridays it will take all day to deal with the large apartment complexes in the north. On any given day, workers know where they are going and, with only slight variation, how long it will take to do their job. A police department, on the other hand, sends a car only to those homes and businesses that call for help (or are seen to need intervention by a patroling police car), and in a pattern that is far from regular and planned. Not only is where they go on a particular day unknown before their phone rings, but the total number of calls varies. Further, the number of calls varies (1) across the days of the week, (2) across the hours of the day, and

(3) from, say, 2 am on one Sunday to 2 am on another Sunday. The point here is that police departments have a more complicated planning task than do public works departments. If the situation were reversed, if there were random calls for rubbish collection, we might say "Do the best you can and it'll all be picked up eventually." But with citizen calls for help we're reluctant to say "We'll send a police car in a day or two." Instead, we have to buy enough cars and hire enough policemen to handle the peak load at all times. Given the high cost of patrol cars (more than a quarter of a million dollars per year, if we include salaries and benefits for manning the car round the clock, not to mention the cost of the car), cities and towns want to have no more than necessary to do the job. Because we can't be certain in advance how many cars we'll need at any time, we have to rely on queuing theory.

The Desktop Hypercube is a software implementation of queuing theory, but unlike the off-the-shelf methods described in standard Operations Research textbooks, it includes spatial concepts. That is, the examples that appear in textbooks (e.g., how long customers in a bank will have to wait or how long the line will be at the checkout line in a supermarket) assume that servers and calling units are in the same place; by contrast, The Desktop Hypercube focuses on the fact that both callers to a police department and the patrol units that will respond are widely separated and at locations that are, in advance, unknown. For example, standard formulae for the length of the queue and the expected wait in a multi-server system are

$$\mathbf{L_q} = \frac{(l/m)^{s+1}}{(s-1)!(s-l/m)^2}(\mathbf{P}_0) \qquad \text{and} \qquad W_q = L_q/l$$

where
l = mean arrival rate
m = mean service rate
s = number of servers
$\mathbf{P_n}$ = probability that there are n units in the system
$\mathbf{L_q}$ = expected number of calling units in the queue
$\mathbf{W_q}$ = expected time in queue

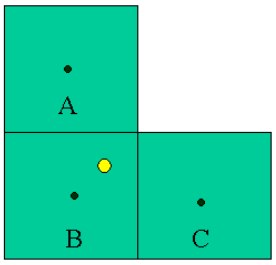
Usually, police data tell us the mean arrival rate l (i.e., calls for service) and the mean service rate m (how long it takes to complete a response), and the number of servers is the number of patrol cars. So if it weren't for the additional complication of location, we could calculate the expected number of callers who will have to wait and the expected amount of time they will have

to wait. But in police work location is extremely important: in order for an officer to "serve the caller," he or she must travel to the caller's location. Indeed, much of the mathematics that is at the heart of The Desktop Hypercube is concerned with calculating the expected location of callers and the expected location of patrol cars.

We assume that the city is divided into geographical "atoms," which may be a city block, a census block or block group, a square on a grid, or any other geographical location. An atom may be irregularly shaped and there may be dozens or hundreds of them in a city. The new version of this software is an extension for ArcView 3.2. It expects a shape file in which a map of the city consists of the atoms as polygon features.

The primary input to the system is historical data on the calls-for-service rate for each geographical atom. From these, The Desktop Hypercube calculates the expected locations of calls. It also needs to know the patrol patterns of the police cars, which the planner designs by using the mouse to mark atoms with different colors (the blue atoms are patrol district 1, the red atoms are patrol district 2, etc). From these, it calculates the expected locations of cars when they are not reponding to calls.

Calculating the expected location of each car is not difficult once patrol patterns are decided. The planner groups atoms into a patrol district for each car, and indicates whether the car divides its time uniformly among those atoms or in proportion to the atoms' calls-for-service history. For example, consider the very small district shown here, consisting of only three atoms. If this district's car spends 25% of its time in atom A, 50% in atom B, and 25% in atom C, then its expected location is the yellow dot which is located at the weighted average of the centroids (shown as black dots) of the three atoms.



Geographical Atoms
(centroids marked with black dot)

atom	coordinates of of centroid	patrol time
A	10, 30	15 min's
B	10, 10	30 min's
C	30, 10	15 min's

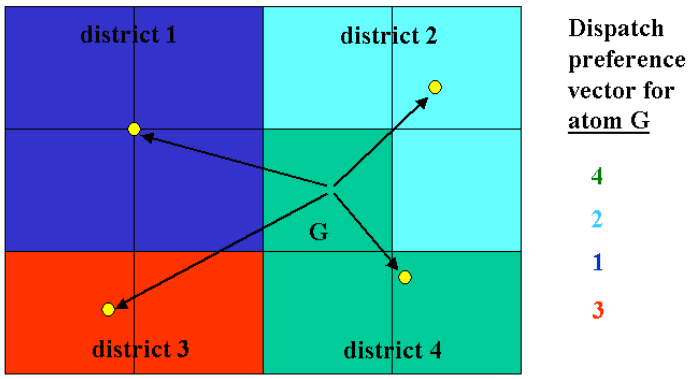
x coord.: (15*10 + 30*10 + 15*30)/60 = 15
y coord.: (15*30 + 30*10 + 15*10)/60 = 15

expected car location: 15, 15

Yellow dot marks expected location of patrol car.

In this way, it is possible to calculate the expected location of every car as it patrols its atoms. Then, for every atom, the expected distance is calculated from that atom to every car, so the cars can be ranked in that atom's "dispatch preference vector." Shown here is the dispatch preference vector for atom G, which ranks cars in order of how close their expected location is to the centroid of atom G: car 4 (its own car) is likely to be closest, car 2 is likely to be next (in case car 4 is busy), car 1 is next, and car 3 is likely to be dispatched to atom G only if the other three cars are busy.

Each atom's color indicates the district it belongs to.



Yellow dots mark expected locations of patrol cars.

Thus, once we have the probability of calls from each atom and the probable location of each car, it is possible to calculate the likelihoods that particular cars will serve particular calls and the probability that each car is busy.[1] With these results, it is possible to generate what we're ultimately interested in, several measures of how well the police are doing what the citizenry demand of them.

Performance Criteria

There are lots of different ways to judge the performance of a police department. Many of them are unaffected by the specifics of patrol patterns and hence are not considered here. Of those that are influenced by patrol district design, perhaps the most obvious is averge response time. But also very important are questions of how uniformly services are provided to the various parts of town. A police department that provides quick and appropriate help to citizens who live near the mayor but slower response in other areas are open to criticism by the town council.

---

[1] For the derivation of these probabilities in situations where location matters, see Larson, Richard C, "A Hypercube Queuing Model for Facility Location and Redistricting in Urban Emergency Services," Computers and Operations Research, 1(1) 1974, 67-95, and Larson, Richard C., "Approximating the Performance of Urban Emergency Service Systems," Operations Research, 23(5), (Sept-Oct 1975), 845-868.

Further, speediness of response is not the only aspect of performance in which we seek uniformity. Balancing workloads across patrol cars is important to both the officers and the residents of their neighborhoods. Policemen are legitimately anxious to avoid patrol designs that cause some of them to respond to calls 50 minutes out of every hour while others are busy only 30 minutes of each hour. The other side of the same coin is that when a car is not responding to a call it is patrolling its beat, and preventive patrol is in itself important to the residents of any neighborhood.

In many cities, the concept of "community policing" is important. This phrase can carry many different meanings, but often it involves a belief that, as much as possible, calls should be serviced by an area's "own" officers. Hence, as with response times, police planners want to achieve uniform levels of interdistrict dispatches.

Recognizing that these are important measures of police performance, I have designed The Desktop Hypercube to report the following results each time a planner tries out a new set of patrol districts:

- region-wide average travel time
- response unit workloads
- response unit travel times
- district travel times
- atom travel times
- dispatches to other districts
- calls assigned to response units from other districts

Note that response unit travel times and district travel times are not the same thing: the former refers to the average travel time across all the trips that a particular car makes, whether to calls in its own district or to calls elsewhere in the city; the latter refers to the average across all the trips to atoms in a particular district, whether made by its "own" car or cars from other districts. One could imagine a situation in which car **A** makes a great many short trips, some to atoms in its own district and some to atoms in nearby districts, and yet many of the calls for service from atoms in district **A** may involve long trips by cars that come from distant districts.

Similarly, it is possible to design patrol patterns such that only a small proportion of car **X**'s calls take it out of its district and yet a large proportion of the calls originating in that district are served by response units from other districts. These are separate measures and both matter.

Not surprisingly, I find that patrol designs which achieve good results with respect to some of these measures don't do so well with respect to others. For that reason, in addition to each of these results, I also calculate an index[2] which allows the planner to see explicitly the trade-offs. The index synthesizes the seven measures bulleted above. For six of them (all except region-wide average travel time), we are interested in measuring dispersion; that is, we consider uniformity to be desirable. Consequently, for each criterion I calculate the sum across cars (or districts or atoms) of the square of the variable minus the mean of the variable. The lower is this statistic, the better is the performance of the system. If, for example, response unit workloads were all the same, then the sum of their squared deviations from their mean would be zero.

The performance index is a quadratic (a linear sum of squared terms), and hence, for each component, bigger deviations from the respective means will count more heavily. It is also unit-free: some of its components are by their nature percentages (e.g., workloads are measured as percentages of time, and interdistrict dispatches are measured as percentages of total dispatches), and the others (travel times) are converted to percentages by replacing them with their values as a percentage of their mean. For one of the measures, region-wide average travel time, the corresponding component of the index is the percentage deviation of the average from a user-supplied target. So it too is unit-free.

Since the real value of the index to the planner lies in the fact that it synthesizes multiple criteria, the weights used to sum the seven components are very important. Rather than presume to know how much any particular police department cares about each component, I use as weights whatever values the user chooses. The planner can enter any set of seven numbers (which will be normalized) to reflect the relative importance of the individual performance measures. Then, when a planner tries out several different patrol district designs, he'll get an overall performance measure for each design (as well as all of the more detailed specific data on workloads, response times, etc).

Each of the seven components of the index is a measure of something the user would like to minimize. In order to end up with a number that is positive, so that it can be thought of as measuring how good a pattern is, the final number is 10 minus the sum of the weighted components. Hence, an ideal patrol pattern will have an index of ten minus zero: all workloads, travel times, and interdistrict dispatches will be completely uniform

---

[2] My thanks to William Lott for his insights into the construction of this index.

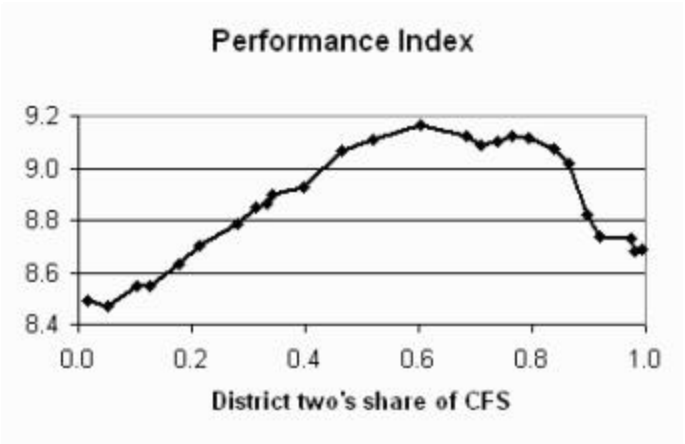
and region-wide travel time will be exactly on-target. For other patterns, the higher the index the better the design.

Experimental Results

This section describes a number of experiments I ran using the performance index of The Desktop Hypercube. The atom definitions are Hartford, CT census block groups and the CFS data are fabricated. In some experiments, I assigned the 106 block groups to 10 patrol districts, and in other experiments I grouped them into 5 or 3 districts. The Desktop Hypercube allows the user to scale the CFS data to any percentage of system capacity. Thus, the CFS data can be viewed as the relative frequencies of calls for service from the atoms. All the experiments were run three times, with the data scaled to 40%, 50%, or 60% of capacity. In each experiment, I started with one of the patrol districts very small (encompassing perhaps only 3 or 4 of the 106 atoms) and gradually increased the size of that district, keeping track of the performance index and of its components as the district grew. There are a number of interesting observations that came out of those experiments.

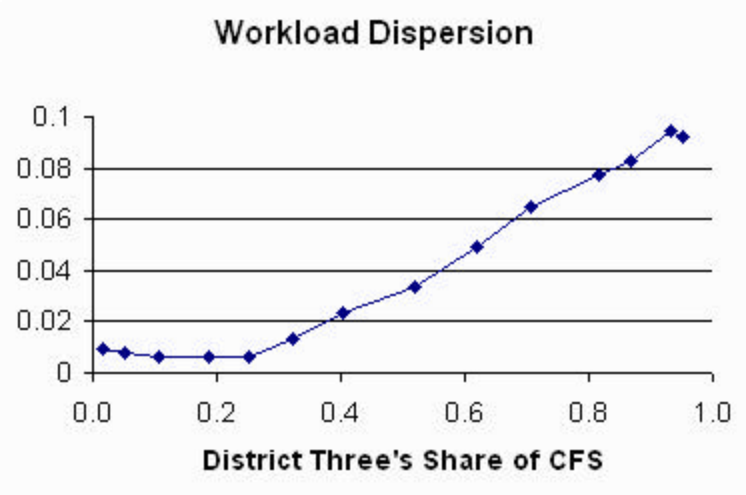
In many cases, changing the patrol pattern results in surprisingly modest changes in the performance index, but these mask more substantial, but offsetting, changes in the components of the index. This makes clear both the advantage and the danger of using an index: its purpose is to allow the user to balance offsetting strengths and weaknesses of a patrol pattern, but to rely on the index alone would be to ignore possibly important differences between patterns. For these experiments, all the components are given the same weights; very likely, each police department will want to use a less uniform set of weights.

In all of my experiments, the performance index at first improves and then worsens as one of the districts grows in size from very small to very large (finally including nearly the entire city). The graph shown here results from a three-district pattern run with CFS at 40% of system capacity. It is surprising how large one district has to get before the performance index turns down; in this case the index continues to improve until that district encompasses 60% of all calls, and doesn't really worsen until it has more than 80% of the city's calls. With 5 districts, the performance index peaks when the growing district is between 40% and 50%, and with 10 districts the peak of the

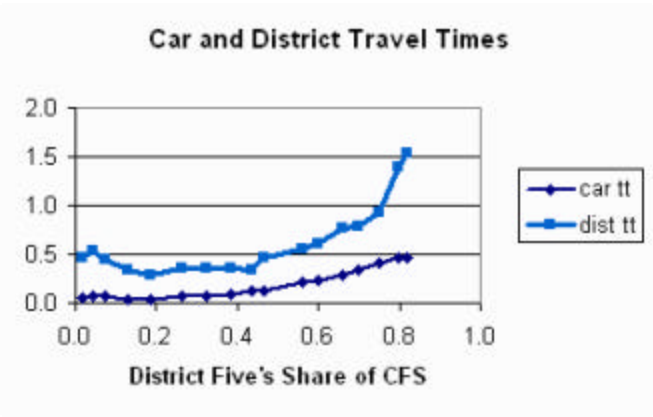


performance index occurs when the expanding district has 20% of the city's calls. Apparently, uniform size (e.g., 10% of CFS for each of ten districts) does not maximize the performance index.

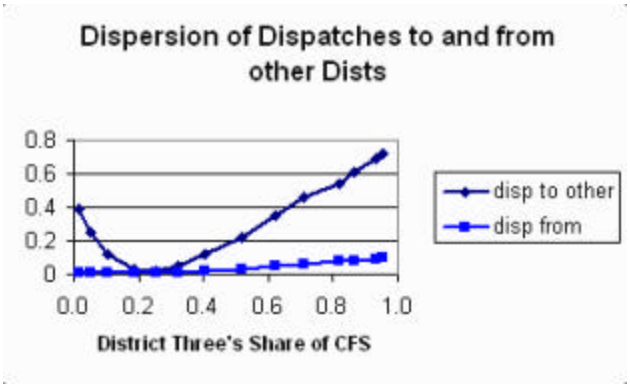
Comparison of the results of experiments which are the same except for the level of CFS shows that the closer the system is to saturation, the less sensitive the performance index is to changes in patrol district design. This is intuitively obvious with respect to workload dispersion: if all cars are responding to calls nearly 100% of each hour, there can't be much difference in workloads. This holds true whether the number of patrol districts is three, five, or ten. In all cases, workload imbalance increases as one of the districts increases in size. That is evident in this graph of workload imbalance in an experiment with five districts.



In general, dispersion of district travel times is greater than dispersion of car travel times and the former is more sensitive to the changing size of the growing patrol district. There is a clear U-shape to the graph of dispersion of district travel time, whereas the dispersion of car travel times is in many cases fairly flat. For example, in an experiment with ten districts, the dispersion of the travel times of patrol cars remains below 0.5 even as one of the districts exceeds 80% of all CFS, but dispersion of district times becomes higher than 1.5.

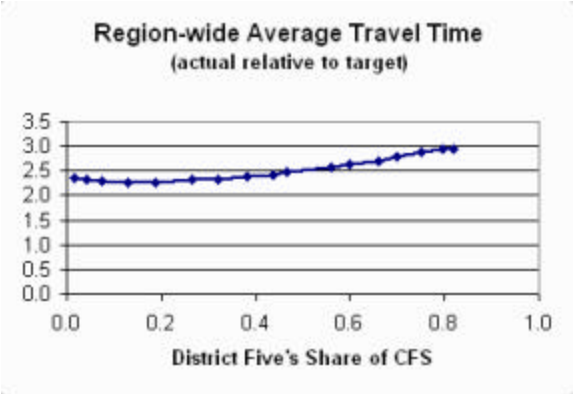


There is a similar asymetry in the data on inter-district dispatches: the dispersion of dispatches of patrol cars **to** other districts is quite responsive to changes in the sizes of the districts while the dispersion of dispatches **from** other districts is pretty flat, despite the growing size of one of the districts. In this example from an experiment with



five districts there seems to be no correlation between dispersion of the two kinds of dispatches.

Many police departments will choose to put a relatively high weight on region-wide average travel time. The actual value depends, of course, on travel speed, distances, number of cars, etc., but patrol patterns matter, too. This graph, based on experiments with ten districts, shows that the ratio of average travel time to target travel time varies from less than 2.5 to nearly 3.0 as one of the districts gobbles up the others. It should be noted that all of these experiments were done with the "Preference for own car" switch turned on (this assumes that a district's own car will respond to a call if it's free, even if another car is free and closer. This graph would look different if that switch were turned off.



Conclusion

The Desktop Hypercube is tool that assists police planners in designing patrol districts. With it, the planner can calculate several characteristics of a proposed pattern of patrols, as well as an index of statistics that are important to the department. This index synthesizes disparate measures of performance such as the uniformity of workloads and travel times across districts and the average of travel times. By selecting weights that reflect a police department's concerns, the planner can make informed choices in the face of inevitable trade-offs.