Online deployment of dynamic traffic assignment: architecture and run-time management

Y. Wen, R. Balakrishna, M. Ben-Akiva and S. Smith

Abstract: Key issues for the online deployment of dynamic traffic assignment systems are addressed, including model calibration, computational efficiency and system robustness to input data. A case study of the recent deployment of the DynaMIT route guidance system in Los Angeles is provided as an illustration of critical practical aspects in the deployment process. A preliminary evaluation of the system indicates online feasibility from a computational perspective and validates the system's ability to replicate off-line data. Online accuracy tests are promising. Ongoing research approaches to enhance system operation and performance are suggested.

1 Introduction

Dynamic traffic assignment (DTA) systems employ sophisticated modelling and simulation techniques to capture complex travel demand patterns, individual driver decisions, network supply phenomena and the many interactions between them in a realistic manner. Such approaches estimate time-varying origin–destination (OD) demand, allocate demand among alternative paths, simulate the movement of individual vehicles along these paths and evaluate network performance through the detailed modelling of traffic dynamics, congestion, queuing, spillback and delays.

Large-scale DTA systems such as DynaMIT (Dynamic Network Assignment for the Management of Information to Travellers) [1] and DYNASMART [2] have been developed to support a variety of online traffic operations applications including route guidance generation and the management of incidents and emergency evacuations in urban areas. These systems synthesise real-time surveillance data (such as sensor counts) with a historical database of expected network conditions and demand patterns to estimate and predict evolving congestion. The ability of such systems to accurately predict individual drivers' route and departure time choices is a key functionality and is expected to play an important role in modelling response to real-time traveller information and in determining prediction accuracy.

Until recently, real-world DTA applications have been limited to off-line evaluations. Typically, archived sensor data from sensors in the study area have been used to calibrate and validate the inputs and parameters used by the systems' constituent components [3–6]. While such tests have generally validated the modelling accuracy of the systems and their ability to replicate real-world conditions in archived datasets, they are limited in their evaluation of numerous online aspects, such as data interfaces, run-time efficiency and system management, that are crucial for successful online performance.

As part of a Federal Highway Administration project to implement incident detection and prediction-based traffic management capabilities in the city of Los Angeles [7], DynaMIT has been deployed at the Los Angeles Department of Transportation (LADOT) and is currently being tested in the South Park area of Los Angeles with real-time sensor data from LADOT's Automated Traffic Surveillance and Control (ATSAC) system. DynaMIT will fulfil two roles in this setup: (1) determining baseline, 'normal' traffic conditions for other systems to automatically detect non-recurring traffic congestion and incidents, and (2) predicting short-term traffic conditions, especially the severity and location of maximum impact zones due to confirmed incidents. Other potential applications include emergency/evacuation management and the generation of consistent, anticipatory route guidance to support an advanced traveller information system (ATIS). The situations outlined above rely on DynaMIT's ability to interface effectively with the surveillance system and meet real-time operating requirements on a daily basis. The focus of this article is the design, implementation and management of an online DynaMIT system that performs satisfactorily in real time.

The South Park area just south of downtown Los Angeles (Fig. 1) is a heavily travelled site, particularly with the regular hosting of special events (including conferences, conventions and sporting events) throughout the year. The high frequency of traffic incidents is thus a concern for the daily operation and management of the network and its traffic control systems. To prepare for deployment, the network was abstracted into a graph representation with 243 nodes

IEE Proceedings online no. 20055008

doi:10.1049/ip-its: 20055008

Paper received 8 December 2005

Y. Wen, R. Balakrishna and M. Ben-Akiva are in the Intelligent Transportation Systems Program, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Room 1-249, Cambridge, USA

S. Smith is with the Volpe National Transportation Systems Center, U.S. Department of Transportation, 55 Broadway, Cambridge, MA 02142, USA

E-mail: wenyang@mit.edu

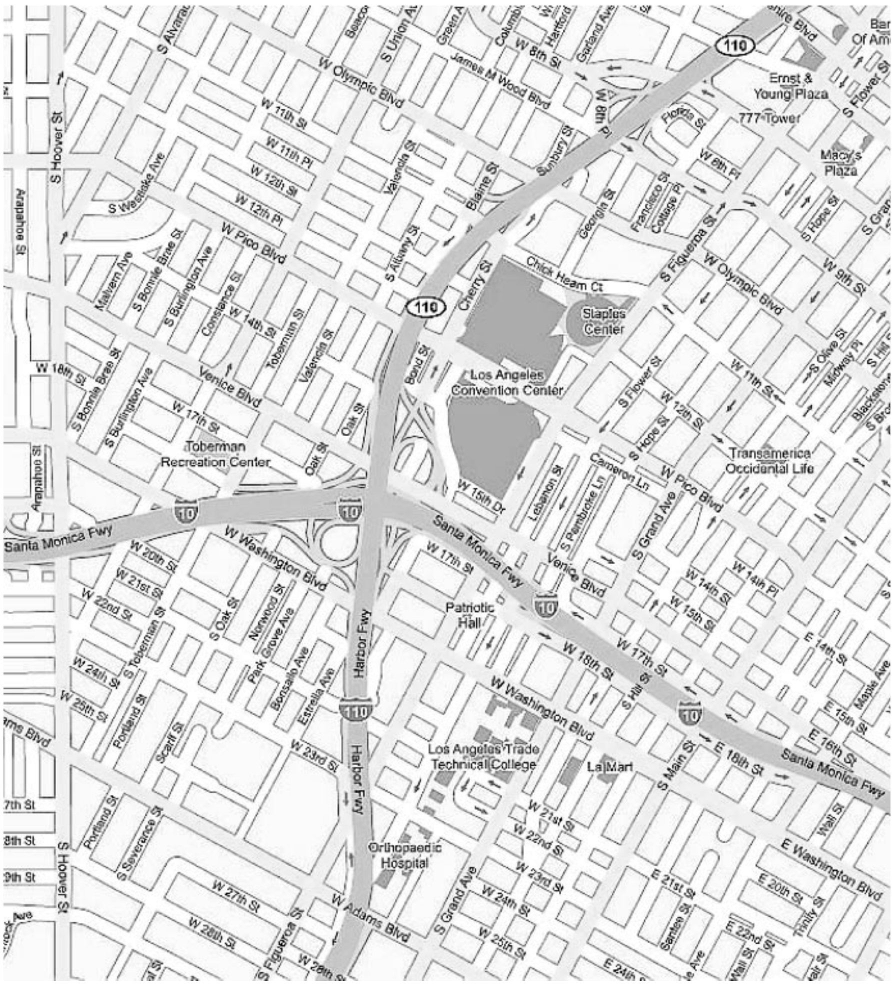

Fig. 1 *The study network (source: Google Maps)*

and 606 links, and DynaMIT's models were calibrated using archived sensor data from the month of September 2004. A historical database of mean OD flows and other model parameters was set up to cover weekdays, Saturdays and Sundays, and the system's state estimation and prediction capabilities were validated using sensor data set aside from the calibration dataset [8, 9]. DynaMIT has been running online using real-time surveillance data since September 2005.

In this article, we present an overview of the functional requirements and practical issues to be considered when planning for the online deployment of DTA systems at traffic management centres (TMCs). Our analysis draws on the ongoing deployment effort involving the DynaMIT route guidance system at the traffic control centre in Los Angeles. This example is used throughout the article to illustrate key design criteria, operating constraints and the run-time management functions that were necessary to successfully implement the online DynaMIT capability.

2 Dynamic traffic management systems

Real-time DTA systems are designed to be a part of dynamic traffic management systems (DTMS) that reside in a TMC. They will use surveillance data to estimate and predict the temporal evolution of traffic, allowing operators to examine evolving traffic conditions and thus facilitate proactive traffic management. Decisions can target supply-side modifications, such as diversions and changes to signal timing or ramp metering plans, or influence demand-side phenomena through route guidance. In this section, we provide a brief overview of the DynaMIT DTA system and its interactions with other components of a DTMS.

2.1 Overview of DynaMIT

DynaMIT is a DTA system designed to generate consistent, anticipatory route guidance. DynaMIT combines a microscopic demand simulator and a mesoscopic supply simulator to capture complex demand and supply processes and their interactions. Models of OD flows, pre-trip and en route driver decisions, traffic dynamics, queuing and spillback allow the system to estimate and predict network state in a realistic manner. DynaMIT is designed to prevent overreaction by ensuring that the guidance generated is consistent with the conditions that drivers are expected to experience. This is achieved through explicit modelling of drivers' reactions to information. The flexible simulation system can adapt to diverse ATIS requirements and is designed to handle a wide range of scenarios including incidents, special events, weather conditions, highway construction activities and fluctuations in demand.

Fig. 2 outlines DynaMIT's framework. DynaMIT integrates various sources of off-line and real-time traffic data, such as the network, historical traffic conditions and real-time traffic surveillance, to generate estimates and predictions of network state. The real-time information is provided by the surveillance and control systems on the network. The minimum real-time information required by DynaMIT is time-dependent link flows, incident characteristics (location, starting time, duration, severity) and traffic control strategies. DynaMIT operates in a rolling horizon framework.

Most existing surveillance systems are limited to vehicle detectors located at critical points in the network. The information provided by these traffic sensors must therefore be used to infer network-wide traffic conditions. The state estimation module performs the task of providing estimates of current network state in terms of OD flows, link flows, queues, speeds and densities. A detailed network representation coupled with traveller behaviour models allows DynaMIT to generate demand and network state estimates that are congruent, while utilising the most recent information available from the surveillance system.

DynaMIT's demand estimation is sensitive to the guidance generated and information provided to the users, through an explicit simulation of pre-trip departure time, mode and route choice decisions. The pre-trip demand simulator updates historical OD matrices by modelling the reaction of each individual to guidance information and aggregating individual trips to obtain updated OD matrices. These flows are further adjusted to reflect the current travel demand on the network, since actual OD flows could diverge from historical patterns due to capacity changes (such as road or lane closures and special events) and other day-to-day fluctuations. The OD estimation model (see [10]) uses updated historical OD flows, real-time surveillance measurements of actual link flows and estimates of assignment fractions (the mapping from OD flows to link flows based on route choice fractions and perceived travel times) to estimate OD flows for the current estimation interval in real time.

A key input to the OD estimation model is the assignment matrix obtained from a traffic simulation model that simulates the actual traffic conditions in the network during the current estimation interval. The demand simulator and network state estimator may have to go through several iterations before converging to a consistent estimate of current network state. The output of this process is an estimate of the actual traffic

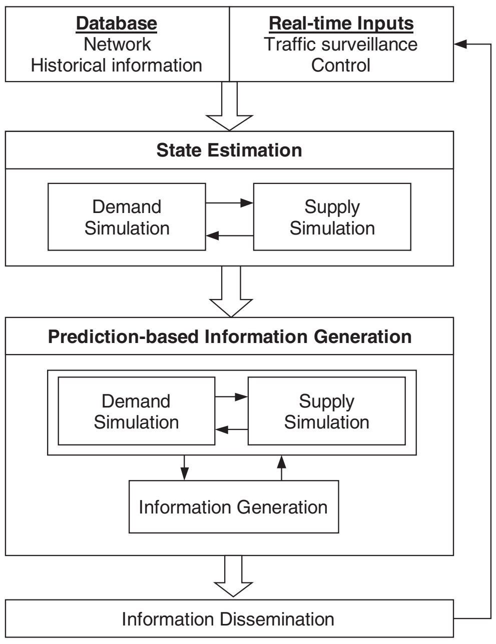

Fig. 2 *DynaMIT framework*

conditions on the network, including origins and destinations of vehicles, link flows, queues, speeds and densities.

The OD prediction model operates on the aggregated historical demand adjusted by the pre-trip demand simulator to provide the required estimates of future OD flows. The network state prediction module predicts future traffic conditions for a given control and guidance strategy, using current estimates of network state and predicted OD flows as inputs. It uses a traffic simulation model and driver en route behaviour model to predict the performance of the network for the prediction horizon. The traffic information and guidance generation function uses the predicted traffic conditions to generate information and guidance according to the various ATISs in place.

The DynaMIT guidance generation algorithm is designed to provide unbiased and consistent information. While unbiasedness guarantees that the information provided to travellers is based on the best available knowledge of current and anticipated network conditions, the consistency prevents overreaction by ensuring that DynaMIT's predictions of expected network conditions match what drivers would eventually experience on the network. Such a guidance strategy means that there is no better path a driver could have taken given the information provided (for a detailed treatment of the consistency problem, see [11]). An iterative process is employed in order to obtain guidance that satisfies these requirements. An iteration consists of a trial strategy, the state prediction under the trial strategy and the evaluation of the predicted state (for consistency).

DynaMIT explicitly models driver response to information. Pre-trip departure time and path choice, as well as en route path choice, are captured through discrete choice models [12]. Further, path choice decisions are based on the Path-Size Logit model [13, 14] to account for driver perceptions of overlapping paths. More information on DynaMIT can be found in [1, 15].

2.2 Integration of DynaMIT at the TMC

Clearly, an online DTA system cannot work without support from other sub-systems residing in the TMC. For example, the surveillance system (including the incident detection system) provides essential information about current network conditions. Setting up the required data communication interfaces is thus necessary for successful deployment.

Surveillance systems need to regularly report the data collected from sensors so that traffic estimation can be performed constantly using the rolling horizon scheme [15]. Incident and traffic control data must also be communicated to the DTA system so that the capacity of network elements can be updated. Other ITS sub-systems may also work closely with a DTA system. For example, predicted information (guidance) may be used by traffic control systems, ATISs and emergency management systems.

The Los Angeles TMC employs DynaMIT for state estimation and prediction. The aggregated sensor data required for OD estimation is obtained in real time from a data server at ATSAC. The data server interfaces with the surveillance system in the field to collate traffic volume and occupancy data every 15 s from arterial and freeway loop detectors. While the arterial sensors are managed by LADOT (through ATSAC), the freeway sensors are controlled by the California Department of Transportation (Caltrans). LADOT's data server retrieves the necessary freeway data through a City/State Link Server connecting ATSAC with Caltrans.

Incidents will be detected and reported by CLAIRE (an intelligent congestion and incident management system, see [16]), with the help of the Advanced Incident Detection Algorithm developed at LADOT.

DynaMIT output such as estimated and predicted network states and route guidance is also stored in the data server. CLAIRE and LADOT's Adaptive Traffic Control System can use this information to dynamically adjust signal timings to respond to evolving traffic demands.

3 Online deployment

Successful online deployment requires that real-time DTA systems have

- effective interfaces for communication with other systems,
- robust control logic in the face of various inputs and
- efficient run-time performance.

Each of these critical aspects is discussed in turn.

3.1 *Input/output interfaces*

Different systems running at a TMC are usually developed independently and may use different formats for input and output (I/O). Therefore a standard interface should be clearly defined and conversion between desired data formats should be provided.

Data interfaces can be implemented in different ways. Using text files, for example, is a simple but effective way to allow data sharing among systems without enforcing too much implementation overhead. Advantages of text files include

- ease of implementation,
- debugging convenience (the user can see the input/output data) and
- compact representation.

Disadvantages of text file interfaces include

- greater storage requirements (than binary representations) and
- the need for concurrency management (one process may attempt to read the file while another process is writing it).

Files sizes can be easily estimated. If an ASCII encoding format is used, for instance, the size of a data file is determined by the total number of characters, including white spaces necessary to separate consecutive fields of data and other non-printable characters such as an end-of-line.

Consider a file to transfer sensor counts. The size of the file is proportional to the number of sensors multiplied by the number of characters of data per sensor, plus a small constant for the overhead. Normally it would not be difficult to keep the number of characters of data per sensor below 20 if each record for a single sensor includes only a sensor ID and the flow measured by the sensor. Consequently, the size of a sensor file for a time interval will be fairly small: considering a network with 1000 sensors, the size is $\sim$20 KB if the above assumption holds. This amount of data can be easily sent from one PC to another within 1 s in most of today's local area networks, which are often based on switched Ethernet or Wi-Fi technology running at speeds from 10 to 1000 Mb/s (megabits per second).

Table 1 shows the average sizes of different I/O data files used by DynaMIT and other systems at LADOT for communication per 5-min interval. We can see that file sizes are modest; they can be written and read quickly.

Using data files for communication among different systems can simplify their integration. As shown in Fig. 3, the data flow between a DTA system and other systems, all of which are in the application layer, can be implemented in the storage layer using shared file access.

Input data (such as sensor data and incidents) and output data (such as predicted flows and guidance) are stored as files on a shared disk accessible by both DynaMIT and other systems, all of which are connected by LADOT's intranet. SAMBA (http://www.samba.org/) is used to provide file service to allow different operating systems, including Linux and various versions of Microsoft Windows, to access a common hard disk. File locking mechanisms are employed to ensure that at any time each of the data files can be accessed by only one program.

As surveillance are not available for a brief period after midnight, shell scripts are developed to automatically restart DynaMIT every day (and also perform clean-up jobs to manage limited disk space and process output-archiving jobs regularly). The start-up script starts DynaMIT at $\sim$1 min after the data becomes available. This intentional delay helps to avoid concurrency issues. Moreover, missing input is handled elegantly, as described in Section 3.2.

3.2 *Input validation at run time*

A DTA system is expected to run 24 h a day online in real time. As previously stated, it needs data input from other systems. Sensor data for one or more time intervals may potentially be unavailable, however, due to reasons such as power failure or communications errors with the surveillance system. Assuming that all inputs are ready, and accepting everything as is, may thus result in meaningless output or even system crash. It is therefore necessary to validate the input data.

The first case is one in which real-time are not available for a time interval. In this case the system should continue to operate using estimates obtained from historical data and the latest information from previous intervals. For instance, in real-time OD flow estimation, the auto-regressive process can still be applied—on the estimated deviations up to the past interval—to update historical flows. This will capture the daily fluctuations and thus be better than using historical OD flow directly.

Table 1: I/O sizes in the Los Angeles project

Content	Type	Average size (KB per interval)
Traffic counts for all 182 sensors	Input	3
Incidents	Input	0.1 per incident
Estimated and predicted flows for all sensors	Output	1.6
Travel time guidance (606 links and a 45-min prediction horizon)	Output	250

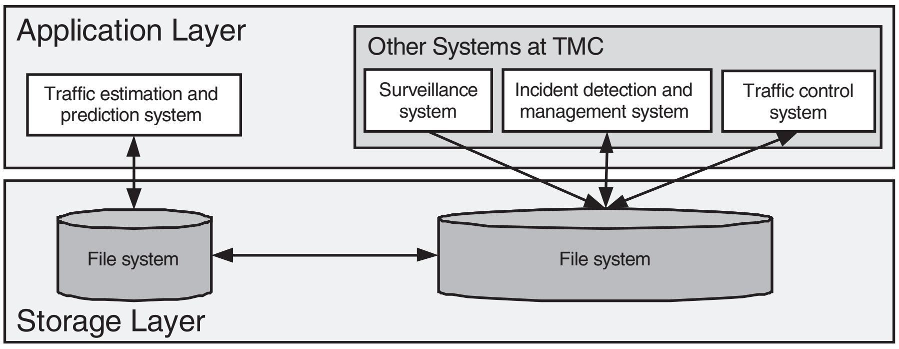

Fig. 3 *Components and data interface*

The second case is that an erroneous input for some interval is identified but reported only after the end of the estimation for that particular interval. Due to the sequential nature of state estimation (under a rolling horizon scheme), estimation error in an interval may affect subsequent intervals, and there is no safe way to guarantee the consistency without rolling back to the original interval and re-estimating the state. For example, if an incident that occurred at 9 a.m. is reported at 9.30 a.m., then the original state estimation (and thus prediction) after 9 a.m. is unlikely to reflect the reality, and we have to re-estimate the state at 9 a.m. while taking the incident into account. In order to effectively handle this situation, we can take a snapshot of the state at the end of each interval, and reload the snapshot in future if necessary.

In order to improve robustness while maintaining efficiency, input validation should be built into state-estimation's control logic. State estimation in general uses an iterative simulation of demand–supply interactions designed to reproduce real-time observations from the surveillance system [17]. As an example, Fig. 4 illustrates the extended workflow of state estimation with the support of input validation adopted by DynaMIT.

At the beginning of each estimation interval, DynaMIT will always perform the first iteration of OD estimation as shown in Fig. 4. An auto-regressive process will be applied in the 'Update historical data' step so that the outcome of the first iteration will capture daily fluctuations reflected in the latest information from past intervals. The surveillance are now expected, and DynaMIT pauses to wait for the data. It counts the time it has been waiting, and if data are still unavailable at the end of the 'grace period' (e.g. 90 s), DynaMIT will stop waiting and use results obtained from the first iteration to continue.

Note that this approach is both efficient and flexible: DynaMIT pauses only at the point where it cannot continue without real-time data, and the length of the 'grace period' can be customised by users.

Even under normal conditions when surveillance data are available, it is still possible to have some knowledge of the quality of surveillance data. It will therefore be appropriate to adjust the estimation in real time. For example, if some sensors are reported to be malfunctioning, it is meaningless to match the estimated OD flows to observed counts from those sensors. Instead, DynaMIT can reduce the weights for equations corresponding to those sensors so that they are in effect ignored by the generalised least squares (GLS) OD estimation procedure. In the extreme case when all sensors are malfunctioning, weights for all sensors would be reduced to such a small level that GLS will return nothing but the updated historical OD flows. This is consistent with the case when no real-time data are available, as previously described.

3.3 Computational efficiency

Real-time DTA systems can generate predictive route guidance (such as link travel times and congestion levels) that when provided to drivers can result in better decisions and improved system performances [17]. Balakrishna *et al.* [18] pointed out that the benefits of dynamic route guidance, however, might be reduced if drivers cannot receive the guidance promptly. Therefore, it is critical for such systems to run faster than real time. This leads to strict requirements on computational efficiency.

In contrast to the richness of literature on the topic of 'real-time' transportation systems, few have been tested and proved to be successful for complex and heavily loaded transportation networks from the real world. This is not accidental, as methodologies developed for small networks with clean data may not work effectively for large-scale problems. While the choice of algorithms remains dominant in this case, implementation issues become more important: poor choice of hardware platforms, data structures, or even programming languages may significantly degrade performance. We discuss some of these issues in this section.

3.3.1 Sparse algorithms for OD estimation: DTA systems use real-time surveillance data to estimate a time-dependent OD matrix. Previous experience and profiling studies have identified OD estimation as one of the major computational bottlenecks of DTA systems.

OD estimation can be formulated as a fixed-point problem [19]. In the real-time context, Kalman-filter-based estimators [17, 20] and least squares estimators [21] are widely used to generate OD estimates. In these algorithms, the time complexity is usually $O(n^3)$, where n is the number of unknown variables, determined by the sum of the total number of OD pairs and the number of sensors. Hence the estimation of OD matrices is computationally intensive and can easily become a bottleneck as the number of unknowns increases to a significant level (as is typically the case in many non-trivial transportation networks).

The Kalman filter algorithm, based on an assignment matrix that maps OD flows to link counts, has been widely proposed for OD estimation. We refer readers to [12] and [17] for a detailed description of the formulation and solution algorithm. The approach works well

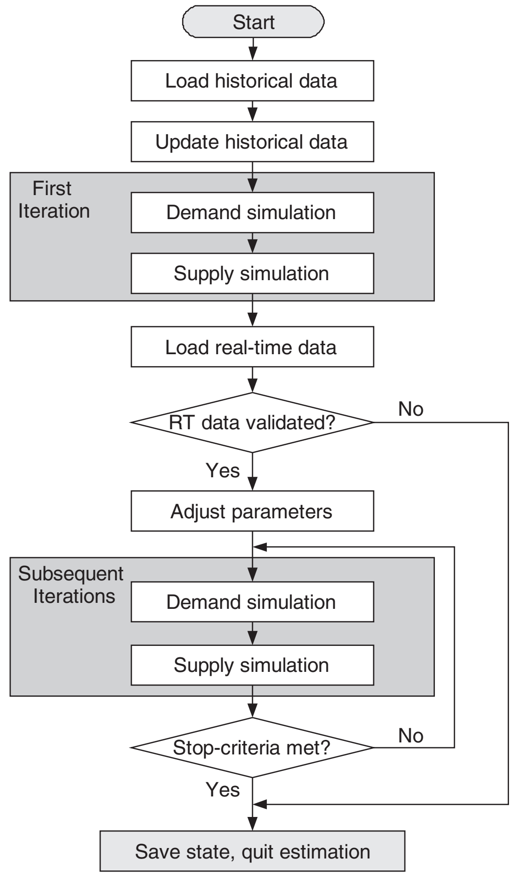

Fig. 4 *State estimation with built-in input validation logic*

for small networks where the numbers of OD pairs and the links with sensors are relatively few. As these numbers increase, however, the run-time performance decreases drastically because of its main disadvantage: the inability to exploit the sparsity of the assignment matrices.

In general, assignment matrices consist of only a small fraction of non-zero elements. This is mainly because the flow for any specific OD pair might be relevant only for a limited number of links. Unfortunately, the Kalman filter approach requires not only matrix multiplications but also inversions. This inevitably causes large amounts of fill-in and thus fails to take advantage of the sparse nature of the problem. The resulting linear algebra computations are intensive, limiting use in large-scale problems.

Bierlaire and Crittin [22] showed that a modified version of the LSQR algorithm, originally proposed by Paige and Saunders [23] for solving sparse least squares problems, can significantly reduce the computational burden and fit well into the real-time context. Crittin [24] also proposed an 'inverse' version of the generalised secant method (called iGSM) for solving large-scale systems of nonlinear equations. These algorithms demonstrate great promise in real-time applications, especially since they exploit the special structure of the problem without reducing accuracy.

3.3.2 Implementation issues: DynaMIT's OD estimation module was recently switched from a Kalman-filter-based algorithm to a sparse constrained least squares algorithm. This brings a substantial reduction in run time, as detailed in subsequent sections. The new implementation also takes into account system bottlenecks introduced by the hardware of a PC to further optimise the performance.

Profiling studies have shown that the major delay is most likely to occur in the memory sub-system of a PC. Simulations and matrix computations require the processing of a huge amount of data, which has to be loaded into memory and transferred to the CPU. Due to the memory paging technique adopted by most modern operating systems, the data might be temporarily stored in the virtual memory such as a file on the hard disk. This adversely affects performance, since disk I/O is much slower than accessing memory. Moreover, even if data are available in main memory, data transmission between main memory and the CPU may still be a system bottleneck, for main memory is likely to work at a slower speed than the CPU. The processor may thus waste many clock cycles waiting for the memory to load the required data.

Installing more memory and using a powerful CPU with more cache (high-speed memory placed between the processor and the main memory and designed to speed up memory access) will certainly help to achieve better performance. However, such an option has limits. It is also important to optimise the system performance from a software design perspective.

Lebeck and Wood [25] showed that programs exhibiting greater locality in memory references would have a higher cache hit ratio and thus better system performance. In general, a program can exhibit two kinds of locality: 'temporal locality', which means the data used by the program will be used again in the near future, and 'spatial locality', which means subsequently accessed data items are located close together in the memory [26]. Compacted data structures, dedicated algorithms, optimising compilers and system supports are possible approaches to improve locality. These topics are out of the scope of this article. Nonetheless, the following rules of thumb were considered during the new implementation of DynaMIT:

- reduce the amount of data that needs to be processed,
- avoid frequently allocating and de-allocating memory and
- use highly efficient algorithms and libraries whenever possible.

3.3.3 Evaluation of DynaMIT's efficiency: DynaMIT has been deployed on a PC with a 2.5 GHz Pentium IV CPU, 256 KB L2 Cache and 512 MB memory. During our online evaluation of the Los Angeles network, horizons comprised of 5-min estimation and 30-min prediction intervals took $\sim$1 min for two iterations of estimation and two iterations of prediction. On average, adding one more iteration for estimation requires 10–15 s extra, including overhead. For the OD estimation, which had been a major bottleneck before, a constrained least squares algorithm using sparse assignment matrix representation is adopted. According to our profiling studies, it can perform one iteration of OD estimation within $\sim$6 s on the current machine. This result clearly indicates that DynaMIT is capable of running in real time under the current configuration.

In order to show the benefit of using sparse-matrix-based algorithms, we have also performed an off-line test using archived real data for different implementations of the GLS algorithm for OD estimation.

A precalibrated Los Angeles network (with 3908 unknown OD pairs and 182 sensors representing South Park) was used. Measurements were recorded at different times of day, such as 3.00–4.00 a.m. and 6.00–6.15 a.m. OD flows were aggregated into 15-min intervals. Two different implementations were chosen, and each was tested for all selected time intervals ten times to avoid potential stochastic effects. This test was performed on a PC with a 1.9 GHz Pentium IV CPU, 256 KB L2 Cache and 512 MB memory.

The averaged results are shown in Table 2. Both implementations use *Lsqlin*, a function in Matlab's optimisation toolbox for solving constrained least squares problems. It will automatically choose the sparse algorithm if the input data contain sparse matrices. Note that the first implementation is similar to what DynaMIT uses online. From the results in Table 2, we see that in the presence of sparse matrices, the first implementation takes approximately 1/40 of the time needed by the second one, thus clearly demonstrating the advantage of utilising sparsity.

Notice that the number of equations in the least squares formulation for the off-line test is $3908 + 182 = 4090$, which is approximately three times the number for the online evaluation case ($1129 + 182 = 1311$). Comparing the time for OD estimation between the off-line and online cases, a similar ratio (15:6) has been observed. Strictly speaking, the run times in the two cases cannot be compared directly: the off-line test was performed on a different PC that is slightly slower; there is also implementation overhead in the online case. However, this result still implies that if sparsity is exploited, the complexity of the OD estimation algorithm is much better than $O(n^3)$.

Table 2: Comparison of the run time of different algorithms for OD estimation

No.	Platform	Use sparsity	Averaged run time (s)
1	Matlab 7.0 (R14)	Yes	15
2	Matlab 7.0 (R14)	No	600

4 Preliminary evaluation of accuracy

Fig. 5 illustrates the accuracy of the off-line DynaMIT calibration performed with archived data from September 2004. The comparisons of DynaMIT's simulated sensor counts with those obtained from the field during the morning peak period indicate a high degree of fit in explaining traffic conditions in the archived dataset. For a more detailed discussion of the off-line calibration and evaluation methodology, see [8].

For our preliminary analysis of online accuracy, we use surveillance data and DynaMIT's output collected on September 29, 2005 (Thursday), and November 14, 2005 (Monday). For every sensor that reports traffic counts on the selected days, we compare estimated and predicted flows against those actually measured on the network. While DynaMIT can also report estimated and predicted speeds and densities for all links in the network, we use only flow data for comparison, since

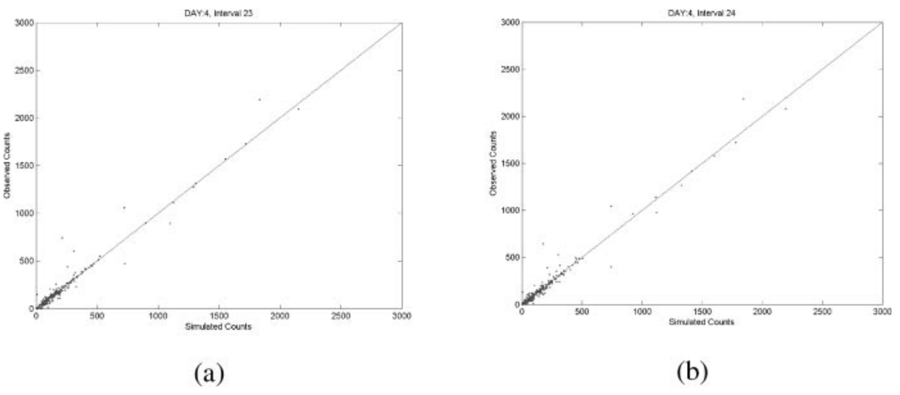

Fig. 5 *Off-line calibration: fit to counts*
a 8.30–8.45 a.m.
b 8.45-9.00 a.m. [8]

actual measurements of speed and density are not available yet.

Our preliminary studies have shown promising results: estimated counts from DynaMIT generally stay close to the actual counts all through the day; predicted counts also follow the same trends and most of them are reasonably accurate. This indicates that estimates reported by DynaMIT through its traffic simulations are likely to reflect real traffic conditions, while the predictions are likely to anticipate the evolution of traffic. Note that DynaMIT used only two iterations of OD estimation and two iterations of prediction while generating its output. Increasing these quantities is expected to enhance accuracy and is the subject of ongoing research. Further, recalibrating DynaMIT with the latest data from 2005 will also result in better online performance, since the current calibrated model inputs reflect traffic conditions observed in 2004. Our preliminary evaluation also illustrates that one-step prediction (obtained just one interval ahead of real time) is in general better than two-step or three-step prediction. This coincides with our a priori belief that the accuracy of prediction will deteriorate over the length of the horizon.

We observed that the counts at most sensors were consistently underestimated during some periods of time for the two given days. This might be due to the use of biased model parameters. For example, DynaMIT's network capacity inputs could be lower than those in the field. Variations in other factors, such as weather and road surface conditions, may also cause traffic conditions to differ significantly from the calibrated average values.

Theoretically, we can use more recent information such as data collected in the previous month to recalibrate DynaMIT's input parameters. However, for real-time DTA systems, off-line calibration is unlikely to make the system sensitive enough to the variability of traffic conditions on different days. Online calibration methods that adjust previously calibrated model parameters in real time to fit current conditions [27] may therefore be appropriate.

5 Conclusion

Successful deployment of online DTA systems requires a calibrated network for accurate predictions, a simple and effective interface with other systems that can elegantly handle erroneous input, and an efficient implementation and hardware configuration to run in real time.

DynaMIT has demonstrated its ability to work online since its recent deployment at the TMC at LADOT. Preliminary studies have shown promising results in terms of computational efficiency and prediction accuracy. We are currently performing a more detailed evaluation. Future studies may focus on the integration of online calibration, which will allow DTA systems to adjust model parameters in real time to adapt to changes in weather conditions, signal timing plans and other factors.

6 Acknowledgments

We would like to thank the Federal Highway Administration and the Los Angeles Department of Transportation for their financial and logistical support towards the DynaMIT project in Los Angeles. The views expressed in this report are those of the authors. The United States government does not endorse products or services. Trade names appear herein only because they are considered essential to the objective of this article.

7 References

1 Ben-Akiva, M., Bierlaire, M., Koutsopoulos, H.N., and Mishalani, R.: 'Real-time simulation of traffic demand-supply interactions within DynaMIT', in Gendreau, M., and Marcotte, P. (Eds): *'Transportation and network analysis: current trends. Miscellenea in honor of Michael Florian'* (Kluwer Academic Publishers, 2002), pp. 19–36

2 Ben-Akiva, M., Bierlaire, M., Burton, D., Koutsopoulos, H.N., and Mishalani, R.: 'Network state estimation and prediction for real-time transportation management applications', *Netw. Spat. Econ.*, 2001, **1**, (3/4), pp. 293–318

3 Mahmassani, H.S.: 'Dynamic network traffic assignment and simulation methodology for advanced system management applications'. Proc. 81st Annual Meeting of the Transportation Research Board, Washington, DC, 2002, CDROM

4 Doan, D.L., Ziliaskopoulos, A., and Mahmassani, H.: 'Online monitoring system for real-time traffic management applications', *Transp. Res. Rec.*, 1999, **1678**, pp. 142–149

5 Kunde, K.: 'Calibration of mesoscopic traffic simulation models'. Master's thesis, Massachusetts Institute of Technology, 2002

6 Chen, Y.-S., Giessen, T., Hendriks, H., and Mahmassani, H.S.: 'Calibrating and validating a large-scale dynamic traffic assignment model under European context'. Proc. 83rd Annual Meeting of the Transportation Research Board, Washington, DC, 2004

7 Balakrishna, R., Koutsopoulos, H.N., and Ben-Akiva, M.: 'Calibration and validation of dynamic traffic assignment systems'. In Mahmassani, H.S. (Ed.): Proc. 16th Int. Symp. Transportation and Traffic Theory, Maryland, July 2005, pp. 407–426

8 Ghaman, R.S., Lieu, H.C., and Scemama G.: 'Implementation of CLAIRE traffic estimation and prediction system, and adaptive traffic control in the US'. Proc. Eleventh Int. Conf. Road Transport Information and Control, London, March 2002, (IEE Conference Proceeding), pp. 95–99

9 Gupta, A.: 'Observability in estimating origin-destination flows for dynamic traffic assignment'. Master's thesis, Massachusetts Institute of Technology, 2005

10 Balakrishna, R.: 'Off-line calibration of dynamic traffic assignment models, PhD thesis (draft), Massachusetts Institute of Technology, 2005

11 Ashok, K., and Ben-Akiva, M.: Alternative approaches for real-time estimation and prediction of time-dependent origin-destination flows, *Transp. Sci.*, 2000, **34**, (1), pp. 21–36

12 Bottom, J., Ben-Akiva, M.E., Bierlaire, M., and Chabini, I.: 'Generation of consistent anticipatory route guidance'. Presented at TRISTAN III Symp., San Juan, 1998

13 Antoniou, C., Ben-Akiva, M., Bierlaire, M., and Mishalani, R.: 'Demand simulation for dynamic traffic assignment'. Proc. 8th IFAC Symp. Transportation Systems, July 1997, pp. 652–656

14 Ben-Akiva, M., and Bierlaire, M.: 'Discrete choice models with applications to departure time and route choice', In Hall, R. (Ed.): *'Handbook of Transportation Science'* (Kluwer, 2003, 2nd edn.)

15 Ramming, S.M.: 'Network knowledge and route choice'. PhD thesis, Massachusetts Institute of Technology, 2002

16 Ben-Akiva, M., Bierlaire, M., Bottom, J., Koutsopoulos, H.N., and Mishalani, R. G.: 'Development of a route guidance generation system for real-time application'. Proc. 8th Int. Federation of Automatic Control Symp. Transportation Systems, IFAC, Chania, Greece, June 1997, pp. 433–438

17 Scemama, G.: 'CLAIRE: an independent, AI-based supervisor for congestion management', *Traffic Eng. Control*, 1995, **36**, (11), pp. 604–612

18 Balakrishna, R., Ben-Akiva, M., and Koutsopoulos, H.N.: 'Simulation-based evaluation of DynaMIT's route guidance and its impact on travel times'. Proc. 10th World Conf. Transport Research, Istanbul, Turkey, June 2004

19 Cascetta, E., and Postorino, M.: 'Fixed point approaches to the estimation of O/D matrices using traffic counts on congested networks', *Transp. Sci.*, 2001, **35**, (2), pp. 134–147

20 Ashok, K., and Ben-Akiva, M.: 'Dynamic Origin-Destination matrix estimation and prediction for real-time traffic management systems'. In Daganzo, C.F. (Ed.): Transportation and Traffic

Theory: Proc. 12th Int. Symp. Theory of Traffic Flow and Transportation, Berkeley, CA, 1993 (Elsevier Science Publishing Company Inc.), pp. 465–484

21 Cascetta, E., Inaudi, D., and Marquis, G.: ‘Dynamic estimators of origin-destination matrices using traffic counts’, *Transp. Sci.*, 1993, **27**, (4), pp. 363–373

22 Bierlaire, M. and Crittin, F.: ‘An efficient algorithm for real-time estimation and prediction of dynamic OD tables’, *Oper. Res.*, 2004, **52**, (1), pp. 116–127

23 Paige, C.C., and Saunders, M.A.: ‘LSQR: an algorithm for sparse linear equations and sparse least squares’, *ACM Trans. Math. Softw.*, 1982, **8**, (1), pp. 43–71

24 Crittin, F.: ‘New algorithmic methods for real-time transportation problems’. PhD thesis, Swiss Federal Institute of Technology, 2003

25 Lebeck, A., and Wood, D.: ‘Cache profiling and the spec benchmarks: a case study’, *Computer*, 1994, **27**, (10), 15–26

26 LaMarca, A.G.: ‘Caches and algorithms’ PhD thesisUniversity of Washington, 1996

27 Antoniou, C.: ‘Online calibration for dynamic traffic assignment’. PhD thesis, Massachusetts Institute of Technology, 2004