Dynamic VRPs: A Study of Scenarios

Philip Kilby
CSIRO Mathematical and Information Sciences,
GPO Box 664, Canberra ACT 2601,
Australia,

Patrick Prosser
Computing Science, University of Glasgow,
Glasgow G12 8QQ, Scotland

Paul Shaw
ILOG, BP 85,
9 rue de Verdun,
94253 Gentilly Cedex,
Paris, France

Abstract

We consider *dynamic* vehicle routing problems where new orders are received over time and must be incorporated into an evolving schedule. We identify the key features which make dynamic problems different to static problems. A sliding scale of dynamicity is introduced and tested.

Keywords: (831) Transportation, (970) Transportation: Vehicle routing (962) Transportation: Scheduling, (???) Algorithms: Dynamic update

1 Introduction

In the Vehicle Routing Problem, a fleet of vehicles must be routed to visit a set of customers at minimum cost. In the standard VRP all orders for all customers are known *a priori*. However, many routing problems in the real world include at least an element of dynamicity, where the schedule must change in response to new or altered requests. We wish to examine here some of the issues raised by dynamic problems.

In the type of problem we consider, some customer visits are specified *a priori*, and an initial schedule is created covering these visits. As the day progresses, new tasks (either pickups or deliveries) enter the system and must be incorporated into the evolving schedule.

In the model we consider, there is a "tentative schedule" which incorporates all currently known tasks. Each task is assigned to a driver, and an ordering of the tasks is specified. New tasks are added to the tentative schedule as new work arrives. There is a notional divide in the tentative schedule. Some of the tasks in the schedule have already been done, and some have been communicated to the drivers; these tasks are fixed in the schedule. The remaining tasks are free to be re-allocated to other drivers, or the order of tasks changed in order to improve the cost of the solution.

It is assumed that there is a communication system between the dispatcher and drivers. When the dispatcher tells the driver which visit(s) to serve next, this is a commitment which cannot then be changed.

Problems fitting this model appear frequently in industry. Most parcel delivery services and taxi services can be modelled in this way. Some travelling repairman services also fit this model. And although we call the work period a "day", the method presented here applies to longer or shorter work planning horizons. The model is therefore an important extension to the traditional Vehicle Routing Problem.

The model includes the standard vehicle routing constraints that the load on a vehicle must not exceed its capacity. Time windows on when the service at a customer may begin may also be present. We refer to the standard problem (where all information is known beforehand) as the *static* problem. For a description of the static problem and solution methods, please see the survey of routing problems by Ball *et al.* [**?**].

Dynamic problems have been studied only sporadically in the literature. A survey is given by Psaraftis [**?**], and we refer the reader to that work for a full review. In addition Bertsimas and Simchi-Levi [**?**] examine the worst and average-case behaviours for some dynamic routing problems.

In his paper, Psaraftis poses the question "How far from the optimum can a dynamic routing algorithm deviate by only using information currently available, and not the entire information (including future information) that could be available if the problem were static?". The answer to this question is: "It depends". The contribution of the current paper is to suggest a framework within which a sensible answer may be given to the question. We then use this framework to suggest some initial results.

The framework we suggest is based on the key features which differentiate a dynamic problem from a static problem, and relate to the business rules governing the operation of the dynamic system.

Indeed, it is one of the difficulties with studying dynamic problems that these busi-

[0] This work was performed while the authors were at the University of Strathclyde.

ness rules have the ability to completely change the face of a problem. These changes are much more radical than for equivalent static problems.

We suggest the following are the key features:

Pickup versus delivery In static problems, pure pickup problems (picking up goods at the customer) and pure delivery problems (delivering from a depot to the customer) can be modelled as equivalent. However, in dynamic problems it is physically impossible to add a delivery to a vehicle after it has left the depot. Pickups, on the other hand, can be added to a existing vehicle schedule at almost any time. Solutions for the two problems will look completely different.

The proportion of visits which are pickups compared to deliveries has an impact on the solution of the dynamic problem.

Commit policy At some stage in the execution of a schedule, the driver must be informed of the next destination or set of destinations. We call this "committing" to part of a schedule. Prior to this time, the schedule can change, but once a particular part of the schedule has been committed to, that part becomes fixed and unchangeable.

Exactly when this commitment is made is one of the fundamental questions of dynamic routing.

One policy is a "latest commitment" policy, where commitment is left until the last possible moment. The next place to visit is only communicated to the driver at the latest possible time that will allow the driver to reach that customer (and all subsequent customers in the current tentative schedule) by their deadline. In execution, this would require the central dispatcher to communicate with each driver, and inform them of their next task at *exactly* the time the driver must begin their journey to that task.

In some scenarios, it may be necessary to commit to a schedule ahead of the latest possible commit time. We use the term *commit horizon* to describe the extra "safety margin" during which the schedule is fixed. For example, a 10-minute commit horizon would force commitment to the next 10 minutes, fixing those tasks in the tentative schedule. The dispatcher then communicates with the driver some time in that 10 minutes.

The cost of this safety margin is the improvements that an on-line system might have made in the final 10 minutes. Less sophisticated communication systems usually require longer commit horizons.

Tasks known *a priori* In most cases, some tasks will be known beforehand. For instance in a delivery system some goods may arrive overnight, and be waiting when the crew starts working in the morning.

How much (what fraction) of the day's work that is known at the start of the day is one measure of the dynamicity of the problem.

Incorporation of improvements One of the main benefits of a dynamic system is the extra time available for a system to search for improvements to the current schedule. However, while the system is searching for improvements, new tasks may appear and have to be incorporated into the schedule. This leads to difficulties as the improvements may conflict with the new work.

- One method to handle potential conflicts is to abort the search whenever new tasks are added, then restart it from scratch. However, this method may severely degrade the efficiency of the improvement search.
- New tasks may be stored until the search finishes, then added en masse. This method has the drawback that the feasibility to incorporate the new tasks may not be known at the time of accepting the order.
- New tasks could be accepted and included in the tentative schedule. When the improvement process terminates only those improvements that do not conflict with the new work are incorporated - that is, incorporate only those improvements which are still feasible when the new work is included. This method could potentially be very difficult to implement.

The extended improvement process is one of the primary advantages of the dynamic system, so any trade-offs in the performance of the improvement process can impact on overall quality.

Time between notification and dispatch A request for service that arrives ten minutes before its deadline is much more difficult to handle than a request that allows several hours to incorporate in the schedule. We use as the measure the time between when a request arrives, and when a vehicle must be dispatched to serve it.

We do not explicitly consider this issue in the numerical tests in this paper. However, we would expect the extra time allowed for improvement would make a substantial difference to the quality of the resulting solution.

Use of historical information Historical information can be used to improve the performance of algorithms for solving *both* static and dynamic problems. Expected quantities and times of orders can be used to help design skeleton routes. However, we argue that this consideration is more important in a dynamic setting. For instance, having dispatched a small vehicle to an area at 9:00am, it is too late to decide you need a large vehicle at 10:00am. In the static case only the tentative solution needs to be updated; in the dynamic case, we have already committed to the important decision.

Again, in this paper we do not further investigate this issue.

Since each of these considerations have a sliding scale of possible values, each of which make sense in different business settings, it is very difficult to study a "typical" scenario. We have chosen to study the following scenario because it represents the dynamic case which is probably closest to the static Vehicle Routing Problem, and includes a (scalable) dynamic element. It is also a "real-world" problem in that it models a particular scenario that can be found in the real world.

Test Scenario

We look at pickup and delivery problems separately - that is, in a particular instance, either all the visits are pickups or all the visits are deliveries.

Common to both pickup and delivery problems, there is a single depot which has set opening hours. No vehicle can leave the depot before opening time, and all vehicles must return to the depot by the closing time.

The pickup scenario is as follows. There are a certain number of parcels left over from the previous day. These form the "tasks known *a priori*". Customers ring up to request a pickup. We designate the time of their call the "available time" of the visit. If the available time is before a particular cut-off time, then the visit is incorporated into the current day's schedule. Otherwise, it is picked up the next day (i.e. it becomes one of the *a priori* visits for the next day).

The delivery scenario is very similar. Parcels arrive at the depot throughout the day. The time of arrival at the depot is the "available time". Parcels that arrive before the cutoff time are included in the current day's schedule. Parcels arriving later are held over for delivery the next day.

Dispatching decisions cannot be pre-empted. That is, once the dispatcher commits to a decision (by telling the driver which visit to serve next) that decision cannot be changed.

This scenario would be a reasonable operating regime for a parcel delivery service.

In the next section we look at the solution methods adopted for this study. We then describe the data developed for studying dynamic problems. Finally, we present some results that compare the effect of varying some of the parameters discussed above.

2 Solution method

The methods used to solve the dynamic problem here are based on an "insert-and-improve" method. As each visit becomes available it is inserted in the best position. The schedule is continuously improved during the day.

The main advantage in solving a problem dynamically is that there is a lot of time available to find a good solution. In the small problems we are solving here (mostly around 100 visits) many improvements can be tested between the arrival of new orders. So whereas in a static case 10 minutes would seem like a long time to spend finding a solution to a 100 node problem, here we have all day so we might as well use it.

A significant disadvantage of dynamic routing is that many of the recent advances in routing techniques which rely on meta-heuristics ([**?**, **?**, **?**, **?**]) may not be applicable. Meta-heuristics allow an algorithm to find its way out of a local minimum, often by accepting a number of cost-increasing changes before returning to a (hopefully lower) local minimum. In the dynamic case, however, it would be inappropriate to accept any cost-increasing moves, lest the schedule be locked in to a high-cost solution by a commit at an inopportune time. Therefore Meta-heuristics can only be used if the search can proceed uninterrupted for some period while it finds a new local minimum. However, the problems with incorporation of improvements discussed in section 1 mean that it may be impossible to lock the solution while a meta-heuristic is run to that point. For these reasons, we do not use a meta-heuristic in this paper.

The method used to find a solution is to first find an initial schedule which incorporates all visits known *a-priori*. Each dynamic visit is then added to the schedule, as it becomes available, by inserting it in the legal position which increases the cost by the least amount.

2.1 Initial schedule

The initial solution is effectively a standard (static) VRP and can use any standard method. In this paper we use a method which (like Caseau [**?**]) inserts a number of visits and improves before continuing. This resembles the dynamic procedure quite strongly and has been shown to be an effective technique.

For the problems presented in section 5 we insert 10 visits at each iteration, and improve using a first accept local search incorporating 2-Opt [**?**], Or-opt [**?**], relocate [**?**], exchange [**?**] and cross [**?**].

We have found that the order of insertion has a large influence on the effectiveness of this and other insertion-based heuristics. We use insertion in the order given by a 2-optimal Travelling Salesman Tour around all visits. That is, we find a tour which ignores all constraints, but simply covers all visits with a single vehicle using travel distance as the objective function. This tour is then improved using 2-opt. The visits known *a-priori* are inserted into the initial plan in the order they appear in this travelling salesman tour. Just for clarity we repeat that the remaining dynamic visits are inserted in the order that they become available.

2.2 Improvement

In section 1 we discussed the problems of incorporating improvements into a schedule. In this paper, we have adopted a fairly simple scheme, based a on a time step. The improvement is allowed to run unhindered for one time-step. Any improvements that it has found during that time are then incorporated into the schedule. Any new work which has become available while the improvement process was running is added in at the least expensive position. Then, the tentative schedule up to the next time step is fixed so that nothing in that period can be altered by the improve process. The cycle then repeats with a new improvement phase.

For example, say the current time is 10:00am, and we are using a time-step of 5 minutes. We first of all incorporate all new work that became known since 9:55. Say the current schedule requires vehicle 1 to be dispatched to visit 50 at 10:02. Then that assignment is fixed, so that while the improvement is running in the next time step, vehicle 1 can be dispatched on time without the possibility of the schedule changing.

The size of time-step has an effect on the solution in a similar way to the commit horizon. A long time-step means commitments are made further into the future, and so there is less scope for improvement. On the other hand a longer time step also means that the local search heuristic – or even a meta-heuristic – can be used more effectively.

In the tests run in section 5 the size of a time-step was chosen so that there were about 50 time-steps during the working day.

3 Data

Benchmark dynamic datasets have been created using the VRP benchmark datasets of Taillard [**?**] (13 files), Christophides [**?**] (7 files) and Fisher [**?**] (2 files). The numbers of visits varies between 50 and 385. Our objective was to design a set of data which were the dynamic equivalent of a set of VRP data. Therefore none of the datasets have time windows on visits. However, there is a deadline on when the vehicle may return to the depot.

We wish to test pickup and delivery systems separately, and so separate data-files were created for each case. The data for the pickup and the delivery case of each file differ only in the sign on the "DEMAND" field for each customer. A pickup is signified by a positive demand, a delivery by a negative demand.

We need to add three types of data to the basic VRP data in the original benchmarks. First an *available time* marking when the pickup or delivery first entered the system. We assume that nothing is known about the task until its available time. We have used a uniform random distribution to generate available times for each task. The times are spread throughout the working day (see below).

Second, a *duration* for each task is required. The duration was chosen so that approximately the same time would be spent performing tasks as travelling between the tasks.

Finally, a concept of the *working day* is required. We chose the working day to be eight times the maximum distance between pairs of visits. This fairly arbitrary figure was chosen as an estimate of travel times in a moderate sized city. However, the working day and durations must be chosen with complimentary values so that the expected time spent performing tasks plus travel time is roughly a working day.

The objective function (cost) for each problem is the total distance travelled (regardless of the number of vehicles required).

The data are available the APES web page:
`http://www.cs.strath.ac.uk/ãpes/apedata.html`
along with a description of the datafile format.

4 Scheduling Rules

All during the day, the system maintains a "tentative schedule" - the schedule as it would be implemented if everything were to be frozen. However the tentative schedule is always changing. New tasks are added in as the day progresses, and the system is also making changes as improvements are found by the continuous improvement process.

We tested only one scheduling rule of the many possibilities: a modified latest dispatch rule. The rule assumes the presence of time windows at customers and at the depot (signifying the working day). The system does not commit to a decision until the last possible moment. This means that the notional dispatcher does not tell the driver of his/her next task until the very moment that they must set off to perform the task. It only commits to the decision at this time. This allows the maximum opportunity for the system to identify better solutions.

In the case studied here where visits do not have time windows, the dispatch time is governed by the depot time window (*working day*). The vehicle must be dispatched in time to make its visits and return to depot before the depot closes.

A slightly different rule is used for the last visit - the return to depot. If the usual rule were used, as soon as a vehicle had performed the last visit on its current tentative schedule it would return home. However, that does not allow for more visits becoming available in its area. Therefore a different "return home" rule is used: don't return to depot until the end of the work day.

5 Computational Experiments

Tests were performed to investigate the effect of three of the parameters discussed in section 1: the differences between pickup and delivery scenarios, the number of tasks known *a priori*, and the length of the commit horizon.

To test the effect of changing the number of tasks known *a priori*, we use a sliding *cutoff time*. All tasks have an associated "available time". Any visits with an available time after the cutoff time are treated as if they arrived yesterday. As described in section 2, an initial schedule is found that serves all of these visits. The visits with arrival times before the cut-off are then introduced in order of their arrival time.

The cutoff time was expressed as a fraction of the working day in order to compare data with different working days.

We also test the effect of different lengths of commit horizon. For a horizon of n time units, all decision on dispatching vehicles and choosing the next visit were fixed n time units ahead of last possible moment. So a horizon of zero minutes means all decisions are committed to at the last possible moment - the most dynamic case. Again, the horizon is expressed as a fraction of the working day.

We have already mentioned the way time-steps are used in the simulation (section 2.2). The time step is chosen so that about 50 time-steps are needed to cover the working day. The improvement process is run during each time step. In the runs reported here, 30 seconds CPU time was used for improvement in each time step. Because - in general - it will not be possible to check all possible moves in the time available, a randomization process was used to select which moves to test during the improvement phase.

5.1 Results

The results are presented in figures 1 and 2. In both figures, the actual data is shown as a background to the "overall" line which shows the averaged data. Both figures also have the same Y axis: the percentage rise in objective compared to the "static" case at the 0 point of the X axis.

Figure 1 presents the results for varying the number of tasks known *a priori*. The horizontal axis can be thought of as the fraction of the day that the company continues to accept orders for the same day. Thus the 0 point is a completely static scenario - all order are known beforehand. At the 0.5 point, half the orders are known from the previous day, and fresh orders are taken only for the first half of the day.

First, comparing the pickup and delivery graphs, it is clear that the cost of deliveries grows quicker with dynamicity than the equivalent pickup scenario. For the scenarios examined here it is around twice as expensive to deliver as to pickup. We conjecture that this is primarily because of the reduced opportunity for rerouting. In a pickup scenario, the dispatcher (human or computer) is free to change the tentative schedule of each vehicle right up until the time it is sent to its next pickup point. The schedule can thus be changed freely in response to new orders. In a delivery scenario, the goods are already loaded onto the vehicle when it leaves the depot. Therefore, only the delivery order can change once the vehicle has left the depot - visits cannot be moved between vehicles.

The most costly delivery scenarios tended to be those with early dispatch times - those that are most affected by inability to change the tentative schedule after the vehicle is dispatched.

Figure 2 shows the effect of varying the length of the commit horizon. The X axis shows the length of the commit horizon, expressed as a fraction of the working day. The cutoff time is fixed at 0.6 for all these problems, that is 40% of the work is known *a-priori* and 60% arrives dynamically.

The first thing to note for both graphs is that the maximum commit horizon shown is one quarter of a day: longer horizons very quickly become unworkable. You can still see many of the background data lines stopping. This indicates that for higher values of the commit horizon, not all tasks could be performed with the available fleet.

Comparing the graphs for pickup and delivery we see that there is less difference between costs when varying this parameter: at 0.25 of a day the cost of deliveries has risen by 45% compared with 30% for pickups.

The cost of deliveries is rising almost linearly - for every 0.1 part of a day "safety margin" ($\frac{3}{4}$ hour in an 8-hour day) the costs rise by about 17%.

The pickup case is similar, with a rise of approximately 10% for every 0.1 of a day additional commit time.

6 Conclusions

The starting point of this paper is the question: How far will a solution to a dynamic vehicle routing problem lie from the solution to the equivalent static solution. We conclude that the answer lies in the business rules governing the operating mode of the dynamic system.

We have attempted to define the key features which will influence the size of this gap. The features suggested are: pickup versus delivery; commit policy; tasks known *a priori*; rules for incorporation of improvements; and time between task notification and dispatch of a vehicle. We have also identified use of historical information as a significant issue.

We have introduced two measures of the dynamicity of a problem, both of which can be adjusted on a sliding scale: the proportion of the visits known *a-priori*, and the *commit horizon*. We have performed a series of tests to examine the effect of these two parameters and seen that these two parameters affect the delivery and pickup scenarios in quite different ways.

We have then examined the effect of three of the key features by choosing two scenarios for further study - a pickup scenario and a delivery scenario. Using base data from a number of classic vehicle routing benchmarks, we have defined a set of dynamic problems, which represent a small but interesting step away from the static vehicle routing problem.

There are two main observations from these test runs. In the dynamic case deliveries are much more expensive than pickups. And second, long commit horizons appear to be unworkable.

This work can be seen as a beginning. There are many questions left to investigate. Some of the more interesting ones are:

- what is the effect of other dispatching criteria (commit policies) used in business today?
- how can historical information on customer orders be used to inform a dynamic model of expected demand?
- how can we use a meta-heuristic when the data is changing, such as in this dynamic case?
- what other parameters affect the cost of dynamic routing, and to what degree?

Acknowledgment

The production of this paper was supported by the GreenTrip project, a research and development undertaking partially funded by the ESPRIT Programme of the Commission of the European Union as project number 20603. The partners in this project are Pirelli (I), ILOG (F), SINTEF (N), Tollpost-Globe (N), and the University of Strathclyde (UK).

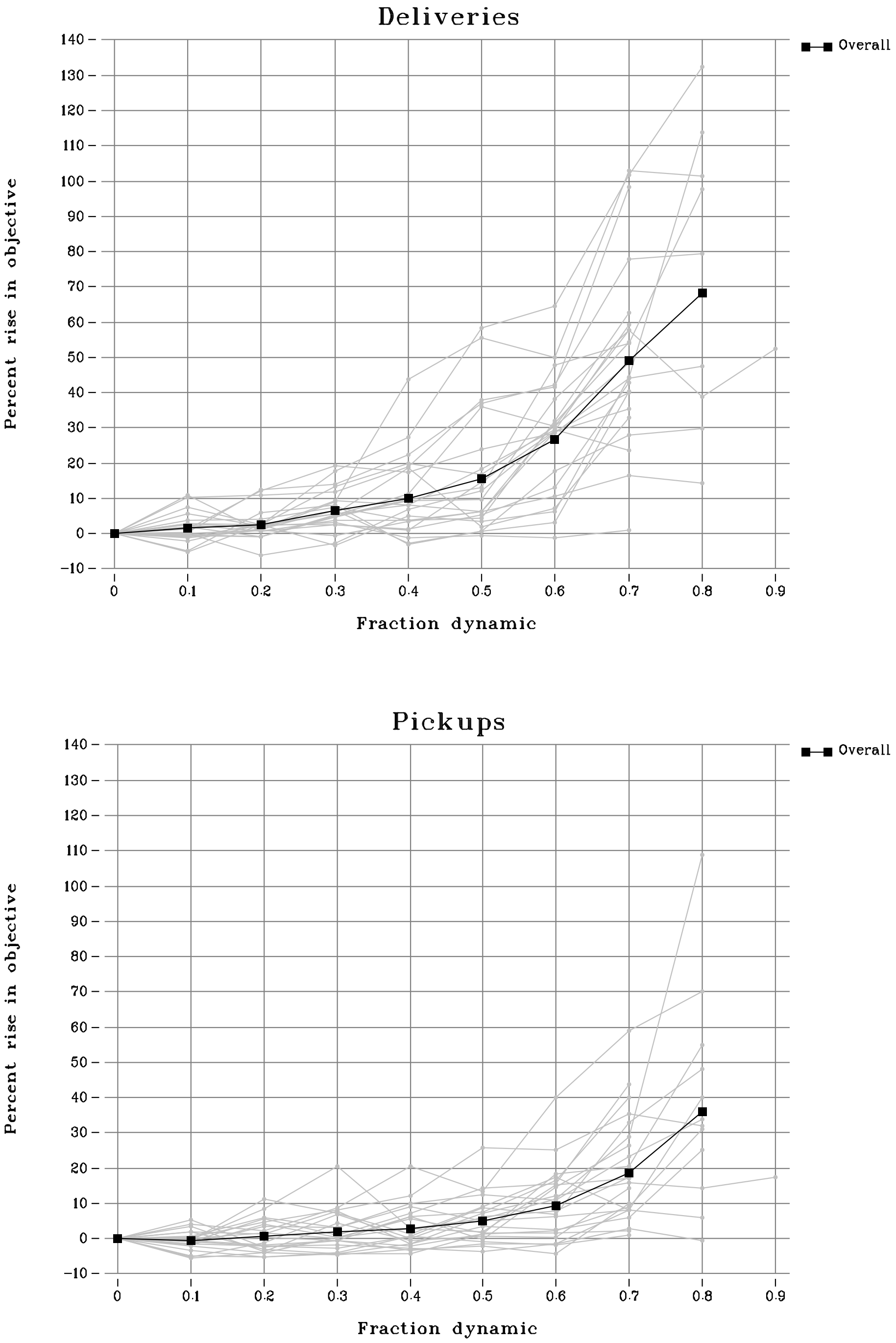

Figure 1: Varying dynamicity

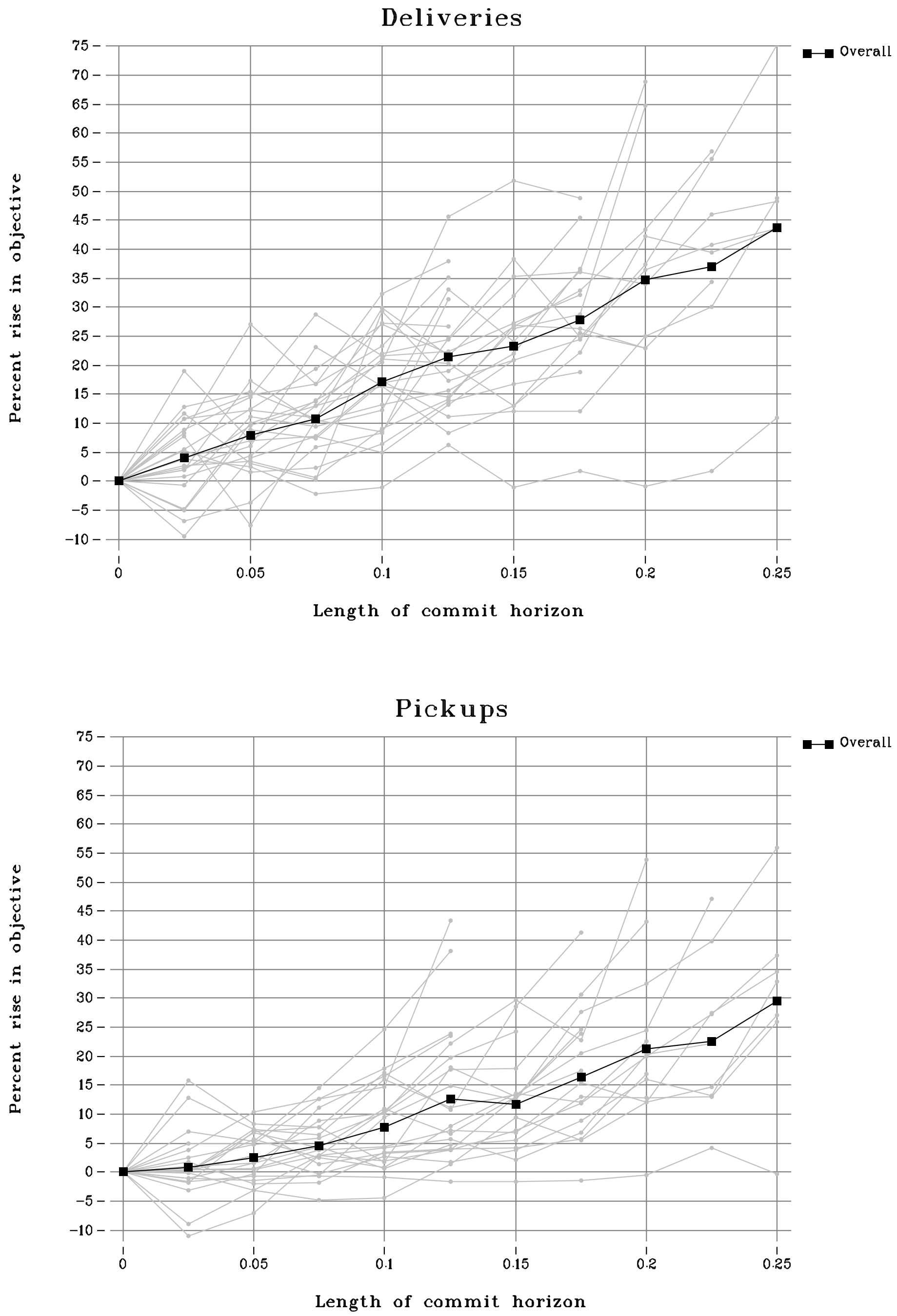

Figure 2: Varying commit horizon (as a fraction of the working day)