Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-9 (2018) 434–439

On the Tracking of Dynamical Optimal Meeting Points

Elif Eser [*,**] Julien Monteil [**] Andrea Simonetto [**]

[*] Bilkent University, Ankara, Turkey (e-mail: elif.eser@bilkent.edu.tr)
[**] IBM Research, Ireland Lab (e-mail: julien.monteil@ie.ibm.com, andrea.simonetto@ibm.com)

Abstract: Finding optimal meeting points on road networks is becoming more and more relevant with the growth of ride-sharing services. Optimal meeting points serve as locations where multiple vehicles can drop their passengers, which will then be pooled in one single (high capacity) vehicle to reach their common final destination. Finding good meeting points is then key in ensuring low travel times to the chosen location and therefore high quality of service. Optimal meeting points are hardy *stationary*, since variations on traffic conditions, road events, or drivers predispositions to go slower/faster than predicted could shift optimality from one location to another one *continuously* in time. In this paper, we propose online algorithms to find and track optimal meeting points in such *dynamic* scenarios, as well as a system architecture to enable extensive simulations for any selected network of interest. Our algorithms are an extension of existing static algorithms. First we integrate realistic considerations such as the finite number of drop-off locations and the proximity radius constant to avoid constant rerouting of vehicles. Second we adapt those algorithms to the dynamic case, which requires to address the trade-off between computational time and optimality. With the aid of extensive numerical simulations, we illustrate and discuss the effectiveness of each algorithm under different scenarios: static networks, dynamic congested networks and dynamic congested network subject to dynamic events.

Keywords: Ride-sharing systems, Optimal Meeting Point, road traffic, traffic events.

1. INTRODUCTION

The design of smart, safe and sustainable mobility applications is essential to the present and future of transportation systems. Novel automotive control systems, see e.g. Pozzato et al. (2017), freeway traffic control systems, see e.g. Pasquale et al. (2017); Ferrara et al. (2016); Iordanidou et al. (2017), urban traffic control systems, see e.g. Haddad and Mirkin (2017); Donovan and Work (2017), and ride-sharing systems, see e.g. Alonso-Mora et al. (2017), are among the hot research topics for car-centered applications.

In particular, the explosion of shared mobility systems presents many opportunities to reduce the number of cars on the roads, while providing users with access to a certain quality of service and increasing the social aspect of mobility. One relevant application is the situation in which customers of a given ride-sharing service are told to meet at a specific location, the Optimal Meeting Point (OMP), where a high capacity vehicle will take them to a common route / destination. This application may be relevant to (i) optimise the pick-up location so as to minimise travel times across users, (ii) address the optimal placement problem of shared cars, (iii) minimise the number of cars used within a company. However, the cars evolve in a very dynamic environment and the ride-sharing service might not have access to extensive traffic state information. In this context, there is a need to adjust the OMP problem to its dynamic setting, despite the limited availability of data sources. Therefore, the obvious criteria to make such an application sustainable and smart are (i) to minimize the driving time or distance to reach the OMP across users, (ii) to avoid constant rerouting of users particularly when approaching the OMP. This will be the main focus on this paper.

OMP in the literature The OMP problem can be traced back to the Fermat-Weber problem which consists in finding the point that minimises the weighed Euclidean distances of many distinct points (Kuhn (1973)). Pierre de Fermat gave a geometric solution for 3 distinct points (Shmoys et al. (1997)), while Weiszfeld proposed an exact algorithm (Weiszfeld (1937)). An extension of this problem is the uncapacitated facility location problem (Cornuejols G. and A. (1990)), where the number of possible locations is finite and the costs are defined for general networks, instead of using the Euclidean distance. Besides, the set covering problem is a special case of the uncapacitated facility location problem (Shmoys et al. (1997)), and known to be NP-complete. Some inapproximability results are available in Slavík (1996). In fact, the OMP problem is the uncapacitated facility location problem with one single facility. Being aware of these negative results, the OMP problem was first studied for road networks in Yan et al. (2011). In their work, the authors essentially propose to compute an augmented convex hull before iterating to find the best solutions within that hull. The more recent work of Sun et al. (2013) makes use of the structure of spatial databases to speed up the exact OMP search process in road networks. While these works provide appealing solutions for the static OMP problem in road networks, here we shift the interest to the dynamic OMP problem.

Contributions of the paper In this paper, we work with the realistic assumption that no traffic information is provided to the ride-sharing system, however, the ride-sharing system may be aware of information on planned traffic events such as road closure, or even real-time events that could be made available to the service via an IoT platform. Our contributions are the following: (i) inspired from previous work we propose three static algorithms and tailor them to the dynamic case, in particular we introduce a threshold constant to avoid the re-routing of vehicles as they get close to the computed OMP, thereby considering the comfort of driving; (ii) we address the dynamic OMP problem by recomputing the proposed static OMP algorithms for fixed time intervals; (iii) we propose an architecture for the extensive testing of algorithms, based on open-source systems; (iv) in the case of traffic congestion we confirm that the dynamic OMP consistently contributes to reducing travel times for all users, in comparison with the static OMP; (v) in the case of traffic congestion with apriori knowledge of events, the travel time gain is even better; (vi) we conclude that future work is needed to improve the existing approximation algorithms for the OMP problem in road networks, which would further help the tracking of dynamical OMPs, and make ride-sharing applications even more sustainable.

Organisation of the paper The remainder of the paper is as follows. In Section 2 we formulate the dynamical OMP problem. In Section 3 we present the architecture of our system. Section 4 introduces the static and dynamic algorithms under consideration. In Section 5 we present a discussion of the simulation results. In Section 6 we conclude with a summary of our findings and a discussion over future research directions.

2. PROBLEM FORMULATION

Let n be the number of vehicles that request a meeting point. Let $x_i(t) \in \mathbb{R}^2$, $i \in \{1,\ldots,n\}$ be their location at time $t \geqslant 0$. Let $\mathcal{G} = (V,E)$ be the road network on which the vehicles and the meeting point live. The vertex set V represents all the possible locations that are reachable by the vehicles travelling through the edge set E. Each edge in E is weighted to model how easy is to transverse such an edge (typically highways have low weights, while country roads have high weights). We further denote with $\mathcal{P}$ the set of *parkable* locations in the graph $\mathcal{G}$.

We define the network distance over $\mathcal{G}$ as the distance that a vehicle would have to drive to go from point x to point y in the network, and we write $D(x,y,\mathcal{G})$. Similarly, we define the network time over $\mathcal{G}$ as the time a vehicle would have to take to go from point x to point y in the free-flow network, and we write $T(x,y,\mathcal{G})$.

We call the static optimal distance meeting point problem as the problem formulated as finding a point in the set $\mathcal{P} \subseteq V$, say O_D, such that the sum of the network distances from each $x_i(0)$ and O_D is minimized:

$$\mathsf{O}_D := \operatorname*{argmin}_{y\in\mathcal{P}} \sum_{i=1}^{n} D(x_i(0),y,\mathcal{G}). \tag{1}$$

Similarly, we call the static optimal time meeting point problem as the problem formulated as finding a point in the set $\mathcal{P} \subseteq V$, say O_T, such that the sum of the network times from each $x_i(0)$ and O_T is minimized:

$$\mathsf{O}_T := \operatorname*{argmin}_{y\in\mathcal{P}} \sum_{i=1}^{n} T(x_i(0),y,\mathcal{G}). \tag{2}$$

Naturally, both static optimal meeting points O_D and O_T are closely related to each other and in this paper we will drop the subscript and consider a generic optimal meeting point O.

Whenever some external events (congestion, route closures, diversions), or internal events (natural low/high speed predisposition) occur, then the static optimal meeting point, determined at time $t = 0$ is not optimal anymore. This is because some of the vehicles will not be at the location they were supposed to be if everything was going as predicted at time $t = 0$. In this context, it is reasonable to formulate a dynamic version. We formulated the dynamical optimal meeting point as follows. (Here formulated only for time, but equivalent for distances).

Dynamical optimal (time) meeting point problem. *Find and track a point in the set $\mathcal{P} \subseteq V$, say $\mathsf{O}(t)$, such that the sum of the network times from each $x_i(t)$ and $\mathsf{O}(t)$ is minimized for each time t:*

$$\mathsf{O}(t) := \operatorname*{argmin}_{y\in\mathcal{P}} \sum_{i=1}^{n} T(x_i(t),y,\mathcal{G}). \tag{3}$$

The Dynamical optimal (time) meeting point problem, DOMP for short, is a time-varying optimization problem [Cf. Simonetto and Dall'Anese (2017) for theoretical background]. What this means is that at each time t one has to find a new optimal meeting point $\mathsf{O}(t)$ based on the actual vehicles' locations $x_i(t)$. By sampling the DOMP problem (3) at discrete time instances t_k, $k = 0,1,\ldots$, we can arrive back at a static OMP that needs to be solved at each time instance,

$$\mathsf{O}(t_k) := \operatorname*{argmin}_{y\in\mathcal{P}} \sum_{i=1}^{n} T(x_i(t_k),y,\mathcal{G}). \tag{4}$$

The sequence of problems (4) (one for each t_k) determines the solution $\mathsf{O}(t)$ with arbitrary accuracy as long as the sampling period $h := t_k - t_{k-1}$ is chosen smaller and smaller.

Solving (4) at each time instance (and within the sampling period) is often not an option for a large number of vehicles. In particular, the problem (4) is non-convex and an approximate solution has to be sought (with some heuristics); this typically does not scale well with the number of vehicles.

In this paper, we propose three different algorithms to tackle the challenge to provide an approximate solution for (4) within the sampling period. The three algorithms leverage the dynamic nature of the problem at hand and restrict the search space $\mathcal{P}$ while the algorithms converge towards an approximate DOMP. The three algorithms trade-off computational complexity and quality of the approximation.

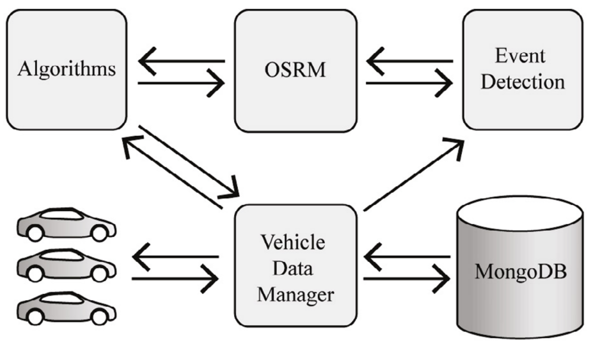

Fig. 1. Architecture of the developed system.

3. SYSTEM ARCHITECTURE

In this section we rapidly present the proposed architecture. Its aim is to enable the extensive testing in simulation of the algorithms for different network instances.

3.1 Generic architecture

We work with *vehicles* which are at different locations and desire to meet at an optimal point. These vehicles communicate with the *Vehicle Data Manager*, VDM, which is the main component of our architecture. VDM interacts with a spatial network database and the meeting point algorithms. In addition, we assume that event information (accident, road closures, etc.) will be reported to VDM and it would subsequently update the edge weights in E. For routing, we leverage *OSRM* (OSRM (2017)), and we construct contracted map files (which can be changed online by the algorithms in case of events). With the help of contracted routings, the algorithms avoid to choose the event-detected roads. Lastly, the characteristics of the network are stored in MongoDB (Chodorow (2013)) – that is a database – to facilitate geospatial query features such as finding nearest nodes to a location within a given radius. The types of queries are employed in different types of OMP algorithms, and their efficient implementation is key.

3.2 Visualisation of results

An additional component resides in the visualisation of the OMP solutions in time. We exploit Leaflet interactive maps with Folium library in Python. This visualization is a property of VDM, and it is not share with the vehicles. The only information that is provided to the vehicles is online GPS navigation.

4. ONLINE ALGORITHMS

In this section we describe three dynamic OMP algorithms, namely *Pruning*, *Greedy*, and *Convex* – starting with the most computational demanding and finishing with the least computational demanding. The main difference among the algorithms will be how they restrict the solution space $\mathcal{P}$ by making use of pruning, greedy, and convex search strategies. All three algorithms receive as input the current location of the vehicles $x_i(t_k)$ at time t_k and generate an approximate meeting point $\mathsf{O}(t_k)$. The sampling period is fixed and indicated as h.

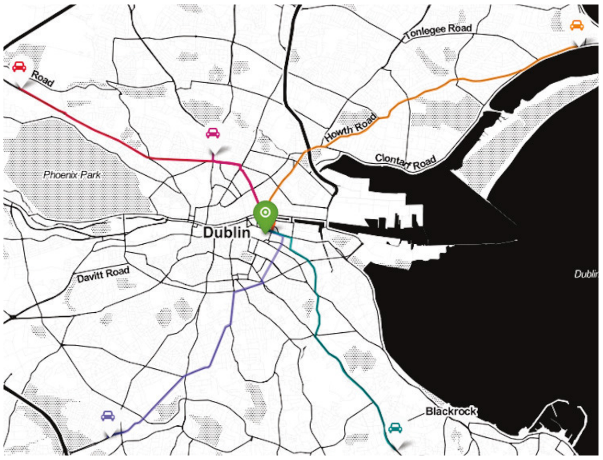

Fig. 2. An example of the visualization in Dublin with five vehicles.

4.1 Pruning Algorithm

The dynamic pruning algorithm exploits a fast-pruning approach to search for the most promising points in the set $\mathcal{P}$ and it is loosely inspired by the merged aggregate nearest neighbor (MANN) queries (Sun et al. (2013)).

In the algorithm, we first find the geolocational center (i.e., the Euclidean center of the convex hull) of the query set $\mathcal{Q}(t_0) := \{x_i(t_0)\}_{i=1}^n$, i.e.,

$$x_{\mathsf{C}}(t_0) = \frac{1}{n}\sum_{i=1}^{n} x_i(t_0), \tag{5}$$

and we project it onto the candidate location set $\mathcal{P}$. The projected point is assigned as the first approximate optimal meeting point $\hat{\mathsf{O}}_0$ to be improved with successive iterations.

We define a search radius $r(t_k)$ as the distance between $\hat{\mathsf{O}}_k$ and the closest vehicle, i.e.,

$$r(t_k) := \min_i \|x_i(t_k) - \hat{\mathsf{O}}_k\|_2, \quad \text{for all } k = 0, 1, \dots, \tag{6}$$

and we further define the ball $\mathbb{B}_{\delta,k} \subset \mathbb{R}^2$ as the ball centered at $\hat{\mathsf{O}}_k$ with radius $r(t_k)\delta$, with $\delta \in (0,1]$ to be chosen by the user to trade-off approximation quality and computational-time.

At time instance $k+1$ we restrict the search for the candidate location in the set $\mathcal{P}'_k := (\mathcal{P} \cap \mathbb{B}_{\delta,k})$. The candidate set is further reduced by the pruning approach described in Sun et al. (2013) (without loss of approximation quality) and the final resulting set is indicated as $\mathcal{P}''_{k+1} \subset \mathcal{P}'_k$.

The final algorithm is then the following.

Dynamic pruning algorithm

(1) Compute the initial approximation as

$$\hat{\mathsf{O}}_0 = \mathsf{Proj}_{\mathcal{P}}[x_{\mathsf{C}}(t_0)],$$

where $\mathsf{Proj}_{\mathcal{P}}$ indicates the projection operator;

(2) For all k from $k = 1$ till termination:
- Compute the search set $\mathcal{P}''_k$;
- Evaluate the cost function $\sum_{i=1}^n T(x_i(t_k), y, \mathcal{G})$ for the points y in $\mathcal{P}''_k$ until the sampling time h is reached;
- The point with minimum cost is the new approximate optimal meeting point $\hat{\mathsf{O}}_k$.

We notice that the pruning algorithm is an exhaustive search type of algorithm, whose search is terminated when the sampling time is reached. In the implementation, a heuristic queue system is put in place, so one evaluates first better "looking" points and subsequently worse "looking" ones. If the sampling time is sufficiently large, and assuming that the optimal meeting point is in $\mathcal{P}''_k$ (which is reasonable in practice[1]) the pruning algorithm will find it. In this sense, the pruning algorithm is an anytime optimal algorithm, and it will be our baseline.

4.2 Greedy Algorithm

We present here the dynamic greedy algorithm, closely related to the static greedy approach in (Yan et al. (2011)), but adapted here to parkable locations and dynamics.

The algorithm is initialized as the pruning one and $\mathcal{P}'_k$ is computed in the same way. We now define an auxiliary sequence $\{z^\ell_{\mathsf{C}}\}$ and a ball $\mathbb{B}_{d,\ell,k+1}$ centered in z^ℓ_{C} with radius d.

At each instance $k+1$ the algorithm sets $z^0_{\mathsf{C}} = \hat{\mathsf{O}}_k$ and search for better optimal meeting points in the set $\mathcal{P}'_k \cap \mathbb{B}_{d,0,k+1}$. If no better one is found, then we set $\hat{\mathsf{O}}_{k+1} = z^0_{\mathsf{C}}$, otherwise we set z^1_{C} as the new best and the search is continued in the new set $\mathcal{P}'_k \cap \mathbb{B}_{d,1,k+1}$, and so on. The greedy approach is tuned by the radius d. Small d's lead to small computational effort but lower quality solution than large d's.

The final algorithm is reported below.

Dynamic greedy algorithm

```
(1) Compute the initial approximation as
        $\hat{\mathsf{O}}_0 = \mathsf{Proj}_{\mathcal{P}}[x_{\mathsf{C}}(t_0)]$,
    where $\mathsf{Proj}_{\mathcal{P}}$ indicates the projection operator;
(2) For all $k$ from $k = 1$ till termination:
    • Set $z^0_{\mathsf{C}} = \hat{\mathsf{O}}_{k-1}$
    • For $\ell = 0$ till sampling period $h$ is reached
        · Compute the search set $\mathcal{P}''_\ell = \mathcal{P}'_{k-1} \cap \mathbb{B}_{d,\ell,k}$;
        · Evaluate the cost function $\sum_{i=1}^n T(x_i(t_k), y, \mathcal{G})$ for
          the points $y$ in $\mathcal{P}''_\ell$;
        · The point with minimum cost is the new $z^{\ell+1}_{\mathsf{C}}$;
        · If $z^{\ell+1}_{\mathsf{C}} = z^\ell_{\mathsf{C}}$ then stop, else continue.
    • Set $\hat{\mathsf{O}}_{k+1} = z^{\ell+1}_{\mathsf{C}}$
```

4.3 Convex Algorithm

The convex algorithm is the most computational light among the others. It is in fact only the first step of the pruning algorithm (or equivalently of the greedy algorithm). Specifically, we compute the Euclidean center of the convex hull of the queries as in (5) and we project it onto $\mathcal{P}'_k$.

Dynamic convex algorithm

```
(1) Compute the initial approximation as
        $\hat{\mathsf{O}}_0 = \mathsf{Proj}_{\mathcal{P}}[x_{\mathsf{C}}(t_0)]$,
    where $\mathsf{Proj}_{\mathcal{P}}$ indicates the projection operator;
(2) For all $k$ from $k = 1$ till termination:
    • Compute $\mathcal{P}'_{k-1}$;
    • Compute the new approximation as
        $\hat{\mathsf{O}}_k = \mathsf{Proj}_{\mathcal{P}'_{k-1}}[x_{\mathsf{C}}(t_k)]$.
```

[1] I.e., whenever the optimal point is in the convex hull of the queries, and the optimal points do not change significantly between subsequent time instances.

We notice that the quality of the approximate solution $\hat{\mathsf{O}}_k$ maybe quite distant from the optimal solution. As a matter of fact this convex algorithm is nothing else than the solution of the optimization problem,

$$\mathsf{O}(t_k) := \operatorname*{argmin}_{y \in \mathcal{P}'_{k-1}} \sum_{i=1}^n \|x_i(t_k) - y\|_2^2 \qquad (7)$$

where we have substituted the time distance T with the Euclidean distance. While this may be rather arbitrary in many situations, in regular urban environments this is not the heresy that it may seem. Consider for example a regular rectangular lattice, as one may find in many neighborhood of New York City, where each road is identical to any other road. In this case, the time distance is in fact the ℓ_1 norm. Adding diagonal segments would shift the ℓ_1 norm to a ℓ_2 norm. Obviously, real situations are more complex than this, and yet it seems rather reasonable that the time distance T could be close enough to a convex function, such as the Euclidean distance. We further elaborate on the quality of this approximation in the numerical experiments.

Finally, notice that the convex algorithm is nothing else than an online projection onto non-convex sets (extension of the usual POCS algorithms). This handle could be used to study its convergence (rate).

5. NUMERICAL RESULTS

5.1 Experimental Setup

For the experiment, we prepare 50 instances for three different problem sizes, i.e., number of vehicles: 5, 10, 20. A large number of vehicles may be justified when ride-sharing systems employ mini-buses as well as normal capacity vehicles. We randomly build the instances on the map of Dublin, Ireland. All experiments are conducted on a Windows 7 machine with 8 GB RAM size and 8-core processor. In all the simulations the threshold is set empirically to $\delta = 0.5$, while d of the greedy approach is set to $d = 1$ km. In the dynamic setup, the sampling period is set to $h = 30$ s. On average, the trip time is 15 minutes.

5.2 Static Case Comparison

First, we compare the performance of the three algorithms with respect to average travel time per vehicle in the static scenario. We use the pruning algorithm as a baseline (since in this case one can run it at optimality). The results are captured in Figure 3, where we indicate the loss of performance (i.e., gain in travel time) of the greedy and convex approaches w.r.t. the pruning one. As we notice, the loss can be significative – but not dramatically so, especially with large problem sizes. The gain in computational time is however very interesting for real implementations, giving an average of $[10.9, 24.1, 48.8]$ minutes for static pruning for increasing vehicle fleets, $[1.2, 2.2, 3.9]$ s for static greedy, and $[0.08, 0.13, 0.26]$ s for static convex.

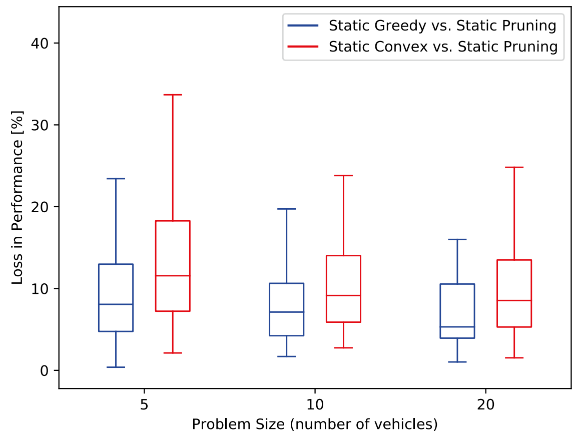

Fig. 3. Comparison of the algorithms in the static case.

5.3 Dynamic Case Setup

We proceed with the dynamic simulations. To simulate realistic variations from the a priori travel times (which assume no congestion), we introduce a simplified model of traffic congestion. For each network edge a cost function is introduced as per e.g. Krichene et al. (2014), assuming a triangular Fundamental Diagram (FD) of traffic flow. Each link is taken at a density state functional point of the FD, either in the free-flow part or in the congested part, which translates into a speed value and therefore a travel time for the link. The effect of varying demand with time is simulated by changing this functional point in a smooth fashion. The links heading towards the city center are assumed to be more congested than the others, and more so if close to the city center. We acknowledge that such a representation of traffic flow is very crude, but this may be refined with the use of a traffic micro-simulator, for instance.

5.4 Dynamic Algorithms Evaluation

We evaluate the dynamic algorithms both w.r.t. the static pruning one and w.r.t. the static version of each algorithm. The idea is to showcase both the gain in performance in terms of less average travel time in considering an approximate dynamical approach vs. (exact) static one, and to showcase how the different algorithms behave w.r.t. each other. Figure 4 depicts the behavior in terms of gain in performance (less average travel time) of the three dynamic algorithms compared to the static pruning one. The result suggests that a dynamic approach is helpful when dynamic traffic is taken into account. We see also that the gain is greater for a larger fleet. Finally, the dynamic pruning algorithm (even if terminated at the sampling period) yields the best results, while a dynamic convex approach is performing the worst, sometimes worse than the static pruning approach. On the computational complexity side, the dynamic pruning approach takes on average 20 s per iteration independently of the fleet size (at the beginning 30 s, since it is stopped at h, but successively it takes less due to a reduced search space), the greedy approach $1-3$ s depending on the fleet size, while the convex approach $0.05-0.2$ s.

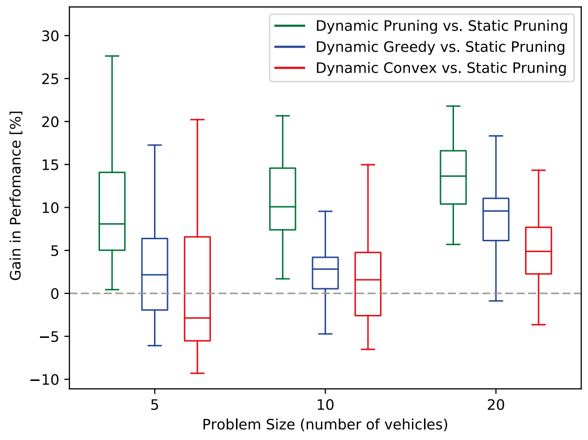

Fig. 4. Comparison of the dynamic vs. static pruning versions of the algorithms

In addition, Figure 5 show the difference of performance of the dynamic algorithms w.r.t. their static versions, suggesting that the dynamic versions perform consistently better than the static ones.

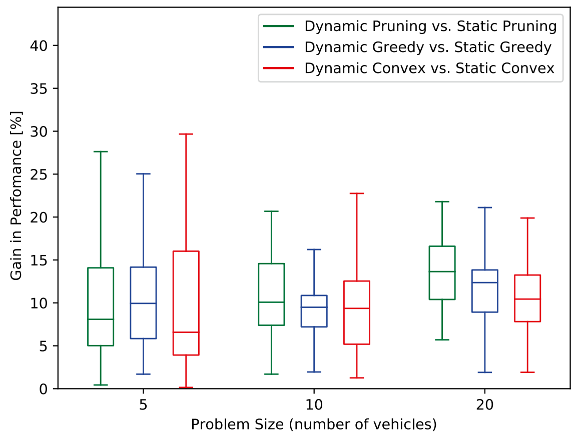

Fig. 5. Comparison of the dynamic vs. static versions of the algorithms

5.5 Events

In the final simulation setup, we consider important traffic events that would lead to a considerable change in the travel times for the vehicles. In particular, we consider two randomly located congestion events, which appears after 5 minutes from the starting of the simulations. The events lower the velocity of the vehicles on the selected edges (up to 10) to 1 km/h. The events affect the algorithms through a changing of the contract files of OSRM.

Figure 6 reports the results for the event case. As we see, the effect is more pronounced for small fleet sizes and generally in line with the conclusions we reached in the absence of events, even though the variance is higher than before, as expected.

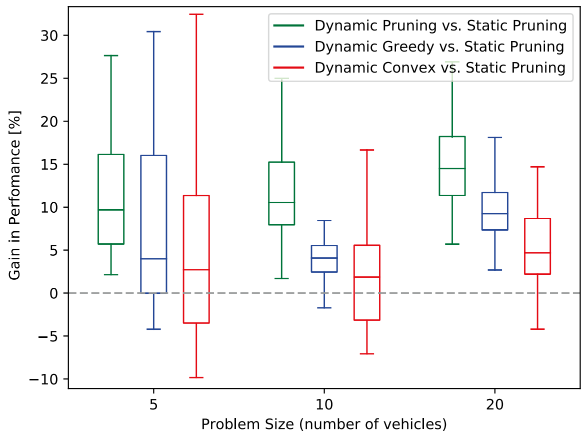

Fig. 6. Comparison of the dynamic vs. static pruning versions of the algorithms in case of events

6. CONCLUSIONS

We have presented the first computational study of the dynamical OMP problem. We believe that this is a very relevant problem for ride-sharing applications, as the total travel time of users should be minimised in a dynamical and optimal fashion, for improved sustainability and customers' satisfaction. In particular we have considered 3 algorithms inspired from recent work and we tailored them to the dynamic setting. Results show that the proposed algorithms consistently perform better in a dynamic setting, with travel time gains sometimes above 10%, even without knowledge on travel time information. In the case of traffic events, the positive effects are amplified. We should also note that the travel time gains grow with the number of vehicles considered for the OMP.

We hope that the promising results presented here will stimulate further research on the topic. In particular, on the simulation side, there is the need to test the algorithms with a more realistic model of traffic demand and traffic events, e.g. using a traffic simulator. On the algorithm side, more sophisticated approximations with competitive running times are needed. Indeed, in the static case, the greedy and convex algorithms suffer from a 10% loss when compared to the near-optimal (pruning) algorithm, and this effect is again observed in the dynamic case. It also remains to be seen how the dynamic algorithms with higher frequency updates or more timely frequency updates would perform.

REFERENCES

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., and Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences.*

Chodorow, K. (2013). *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage.* " O'Reilly Media, Inc.".

Cornuejols G., N.G.L. and A., W.L. (1990). The uncapacitated facility location problem. *Discrete Location Theory*, 119–171.

Donovan, B. and Work, D.B. (2017). Empirically quantifying city-scale transportation system resilience to extreme events. *Transportation Research Part C: Emerging Technologies*, 79, 333 – 346.

Ferrara, A., Sacone, S., Siri, S., Vivas, C., and Rubio, F.R. (2016). Switched observer-based ramp metering controllers for freeway systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 6777–6782.

Haddad, J. and Mirkin, B. (2017). Coordinated distributed adaptive perimeter control for large-scale urban road networks. *Transportation Research Part C: Emerging Technologies*, 77, 495 – 515.

Iordanidou, G.R., Papamichail, I., Roncoli, C., and Papageorgiou, M. (2017). Feedback-based integrated motorway traffic flow control with delay balancing. *IEEE Transactions on Intelligent Transportation Systems*, 18(9), 2319–2329.

Krichene, W., Reilly, J.D., Amin, S., and Bayen, A.M. (2014). Stackelberg routing on parallel networks with horizontal queues. *IEEE Transactions on Automatic Control*, 59(3), 714–727.

Kuhn, H.W. (1973). A note on Fermat's problem. *Mathematical programming*, 4(1), 98–107.

OSRM, P. (2017). URL `http://project-osrm.org/`.

Pasquale, C., Sacone, S., Siri, S., and De Schutter, B. (2017). A multi-class model-based control scheme for reducing congestion and emissions in freeway networks by combining ramp metering and route guidance. *Transportation Research Part C: Emerging Technologies*, 80, 384 – 408.

Pozzato, G., Hoffman, M.A., and Onori, S. (2017). Multichannel physics-based modeling and experimental validation of an uncoated gasoline particulate filter in clean operating conditions. In *2017 American Control Conference (ACC)*, 5392–5397.

Shmoys, D.B., Tardos, É., and Aardal, K. (1997). Approximation algorithms for facility location problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 265–274.

Simonetto, A. and Dall'Anese, E. (2017). Prediction-Correction Algorithms for Time-Varying Constrained Optimization. *IEEE Transactions on Signal Processing*, 65(20), 5481 – 5494.

Slavík, P. (1996). A tight analysis of the greedy algorithm for set cover. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 435–441.

Sun, W., Chen, C., Zheng, B., Chen, C., Zhu, L., Liu, W., and Huang, Y. (2013). Merged aggregate nearest neighbor query processing in road networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2243–2248.

Weiszfeld, E. (1937). Sur le point pour lequel la somme des distances de n points donns est minimum. *Thoku Mathematical Journal*, 43, 355–386.

Yan, D., Zhao, Z., and Ng, W. (2011). Efficient algorithms for finding optimal meeting point on road networks. *Proceedings of the VLDB Endowment*, 4(11).