The Thirteenth International Symposium
on Combinatorial Search (SoCS 2020)

Cooperative Path Planning for Heterogeneous Agents

Keisuke Otaki, Satoshi Koide, Ayano Okoso, Tomoki Nishi
Toyota Central R&D Labs., Inc.
Bunkyo-ku 1-4-14, Tokyo, 112-0011, Japan
{otaki,koide,okoso,nishi}@mosk.tytlabs.co.jp

Abstract

Cooperation among vehicles is a promising concept for path-planning in transportation services. For instance, vehicle platooning on highways decreases fuel consumption because it reduces the air resistance, and this makes us construct paths sharing sub-paths in platooning. We study a model that permits heterogeneous cooperation and discuss a path-planning problem under the assumption that heterogeneous cooperation benefits the objective function. In this paper, we provide experimental evaluations and simple heuristic solvers to observe the behaviors of the optimization problem.

Introduction

We focus on path-planning problems for multiple vehicles, where vehicles cooperate with others by sharing sub-paths (i.e., sub-routes). The problem with one vehicle type, named *vehicle platooning problems* (VPPs), has been studied in the transportation domain (Larsson, Sennton, and Larson 2015). In previous work, the generalized problem of VPPs on heterogeneous vehicle types have been studied (Otaki et al. 2019b; 2019a) and named *cooperative path planning problems* (*CPP*). The heterogeneity can reflect different discount (or increment) effects. They reported theoretical and computational aspects of the problem using integer programming (IP) solvers. However, some metrics of experiments with IP solvers (e.g., gaps and computational times) were not precisely evaluated in the previous studies. Also, only small synthetic graphs were used in the experiments. We here experimentally evaluate the existing IP models using larger instances. To discuss the computational hardness of the problem, we also compare the results using an IP solver with those obtained by simple heuristic solvers.

The CPP Problem We follow the notations used in (Otaki et al. 2019a). Let $\mathcal{T}$ be a set of vehicle types and $\mathcal{C} \subseteq \mathcal{T} \times \mathcal{T}$ be a set of *binary* cooperation relation. For $(T_1, T_2) \in \mathcal{C}$, we say that vehicles of type T_1 are *parent* and those of type T_2 are *children* of the cooperation. Parameters $\eta^{(T_1 T_2)}$ are given to represent the benefit obtained from (T_1, T_2) cooperation for $(T_1, T_2) \in \mathcal{C}$. In addition, for an undirected weighted graph $G = (V, E, w)$, sets $\mathcal{R}^{(T)} = \{(o_t^{(T)}, d_t^{(T)}) \in V \times V \mid t \in [N_T]\}$ of travel requests are given for type $T \in \mathcal{T}$, where N_T is the number of type T vehicles and $[N_T] = \{1, \dots, N_T\}$. The CPP problem is to simultaneously compute a set $\mathcal{P}$ of paths from $o_t^{(T)}$ to $d_t^{(T)}$ for each vehicle $t \in [N_T]$ of type $T \in \mathcal{T}$ that minimizes the sum of travel cost $\mathbf{C}(\mathcal{P}) = \sum_{T \in \mathcal{T}, e \in E} w(e) g^{(T)}(e)$, where

$$
\begin{aligned}
g^{(T)}(e) &:= g^{\mathrm{p}(T)} + \sum_{\substack{T' \in \mathcal{T} \\ \text{if } (T,T') \in \mathcal{C}}} g^{\mathrm{c}(TT')}, \qquad (1)\\
g^{\mathrm{p}(T)}(e) &:= \#(\text{Type } T \text{ parent at } e),\\
g^{\mathrm{c}(TT')}(e) &:= \eta^{(TT')} \cdot \#(\text{Type } T' \text{ child of } (T,T') \text{ at } e).
\end{aligned}
$$

The value $g^{(T)}(e)$ represents the *discounted effect* of vehicles traveling along with e on G, where particularly the term $g^{\mathrm{c}(TT')}$ reflects the effect by cooperation with $\eta^{(TT')}$; if $\eta^{(TT')} < 1$, the effect benefits to reduce the objective function (as the fuel consumption is reduced in VPPs). If $\eta^{(TT')} = 1$, the cooperation can be interpreted as just following shortest paths. *Cooperation-wise capacity constraints* (the max. number of vehicles taking (T, T') in Eq.(1)) also imposed (see Def. 3 at (Otaki et al. 2019a)). The IP formulation to solve the problem above is provided in Appendix A. at (Otaki et al. 2019a).

Numerical Experiments

We extensively evaluate the IP formulation and simple heuristics based on shortest paths and graph matching on synthetic and real graphs obtained from OpenStreetMap. Experiments are conducted on a workstation with an Intel Xeon W-2145 CPU at 3.70GHz with 64GB of memory and Gurobi 8.1. Graphs `K`-d represent graphs obtained from Kyoto, Japan and those of `R`-d means synthetic graphs generated by (van de Hoef, Johansson, and Dimarogonas 2015)[1]. We only report results using clustered requests, which are generated as traveling from left to right on G.

Evaluation of IP formulation We evaluate CPP for single vehicle type, which is equivalent to *capacitated* VPP, in

[1]The center of map `K`-d is (lat, long)=(35.026244, 135.780877) and the value d is the distance to extract graphs. For synthetic graphs, d is set to be the number of vertices in G.

Table 1: The mean computational times on CPP with one vehicle type and $\eta^{(LL)} = 0.7$ on K-500.

N_T	Capacity $Q = 1$	2	4
10	12.2 (11.1)	7.9 (5.0)	3.8 (3.8)
15	103.3 (170.1)	71.6 (133.4)	63.1 (135.5)

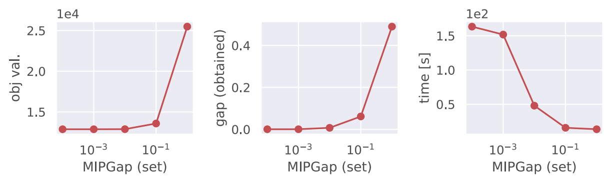

Figure 1: Results on clustered requests against `MIPGap` θ.

terms of the capacity constraints with parameter Q. Note that the capacity has not been discussed for VPP by Larsson et al. (Larsson, Sennton, and Larson 2015). We measure the mean computational time against Q and its standard deviation. Table 1 shows results on K-500 with $N_T \in \{10, 15\}$ and $Q \in \{1, 2, 4\}$ using 10 randomly generated clustered requests. The results indicate that small C values require more computational time. We conjecture that the difficulty is because they require to form a group of vehicles to minimize Eq. (1). Note that heterogeneous instances require more computational times (e.g., (Otaki et al. 2019b)). For example, we confirmed that the solver cannot find any solutions within 15 minutes if $d = 1000$, $Q = 4$, and $N_T = 20$.

The branch-and-bound algorithm returns a current solution when a relative difference between the computed upper-bound and lower-bound is less than `MIPGap` θ. Therefore the parameter affects the computation time and quality of returned solutions. To evaluate the effect of θ, we use the IP formulation built from the selected requests on K-500. We set $N_L = N_S = 10$, $Q^{(LL)} = 3$, $Q^{(LS)} = 3$ and $Q^{(SS)} = 2$ and vary $\theta \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$[2]. We then observe (1) objective values, (2) obtained gaps by Gurobi, and (3) computational times. Figure 1 illustrates the average values of the three on random 10 instances. We can confirm that for a relatively large gap (e.g., $10^{-1} = 10\%$ or $10^{-2} = 1\%$) the modern IP solver succeeds to find solutions efficiently (e.g., less than 10 seconds for clustered requests are enough to find solutions). We then conjecture that IP solvers with a *large gap parameter* θ can be used to find feasible solutions.

Evaluation of heuristic solvers We develop two simple heuristic solvers: **Greedy** and b-**Matching**. In the heuristics, we use the difference between Eq. (1) and the cost when following shortest paths as the score of benefit of taking cooperation to divide the problem into (disjoint) sub-problems. The **Greedy** heuristic makes the group of vehicles in a greedy manner of the score. The b-**Matching** heuristic selects groups using b-matching algorithms. We test them using clustered requests on three graphs (K-500, K-750, and R-300), wherein Gurobi could find solutions up to 10 minutes with $\theta = 10^{-4}$. Table 2 summarizes the results. Note that the results of the IP solver include instances for which the IP solver returns feasible solutions because it cannot find optimal ones. We confirmed that simple heuristics constantly construct the solution whose objective values are roughly 15-25% larger than the IP solver. These results are comparable to the setting of `MIPGap` $\in [0.15, 0.25]$[3]. By comparing results, simple heuristics are faster than Gurobi with large θ.

[2]Note that $Q^{(T_1 T_2)}$ is the capacity of cooperation $(T_1, T_2) \in \mathcal{C}$.

Table 2: IP solvers and heuristics are compared with respect to times (T). For heuristics, gap (G) means the ratio of scores based on the IP solver.

Label	IP solver	Heuristics			
		Greedy		b-**Matching**	
	T [s]	(G)	T [ms]	(G)	T [ms]
K-500	106	+1.38	5.78	+1.14	62.57
K-750	489	+1.38	8.16	+1.14	79.91
R-300	534	+1.47	3.49	+1.13	64.25

Conclusions and Future work

We studied the cooperation among heterogeneous vehicle types in path-planning and experimentally evaluate the difficulty of the problem in various aspects. Although the CPP problem is an abstract problem class from the viewpoint of mobility-as-a-service (MaaS) applications, this experimental study opens a new door on models that support cooperating heterogeneous vehicles with the travel costs discounted.

Our future work will include the developments of sophisticated solvers (e.g., anytime algorithms, taking time-synchronization into account) and efficient solvers that can find optimal solutions in large-scale problems.

References

Larsson, E.; Sennton, G.; and Larson, J. 2015. The vehicle platooning problem: Computational complexity and heuristics. *Transportation Research Part C: Emerging Technologies* 60:258–277.

Otaki, K.; Koide, S.; Keiichiro, H.; Okoso, A.; and Nishi, T. 2019a. Multi-agent path planning with heterogeneous cooperation. In *Proc. of ICTAI2019*, 93–100.

Otaki, K.; Koide, S.; Okoso, A.; and Nishi, T. 2019b. Cooperative routing with heterogeneous vehicles. In *Proc. of AAMAS2019*, 2150–2152.

van de Hoef, S.; Johansson, K. H.; and Dimarogonas, D. V. 2015. Coordinating truck platooning by clustering pairwise fuel-optimal plans. In *Proc. of ITSC2015*, 408–415.

[3]This parameter cannot be directly compared to the resulted gaps since `MIPGap` is the gap between the upper and lower bounds. However, the value θ is often related to the obtained gap as we see in Fig. 1 and it is applicable to observe the problem.