Coordinating Multi-party Vehicle Routing with Location Congestion via Iterative Best Response

Waldy Joe and Hoong Chuin Lau(✉)

School of Computing and Information Systems, Singapore Management University, Singapore, Singapore
waldy.joe.2018@phdcs.smu.edu.sg, hclau@smu.edu.sg

Abstract. This work is motivated by a real-world problem of coordinating B2B pickup-delivery operations to shopping malls involving multiple non-collaborative Logistics Service Providers (LSPs) in a congested city where space is scarce. This problem can be categorized as a Vehicle Routing Problem with Pickup and Delivery, Time Windows and Location Congestion with multiple LSPs (or ML-VRPLC in short), and we propose a scalable, decentralized, coordinated planning approach via iterative best response. We formulate the problem as a strategic game where each LSP is a self-interested agent but is willing to participate in a coordinated planning as long as there are sufficient incentives. Through an iterative best response procedure, agents adjust their schedules until no further improvement can be obtained to the resulting joint schedule. We seek to find the best joint schedule which maximizes the minimum gain achieved by any one LSP, as LSPs are interested in how much benefit they can gain rather than achieving a system optimality. We compare our approach to a centralized planning approach and our experiment results show that our approach is more scalable and is able to achieve on average 10% more gain within an operationally realistic time limit.

Keywords: Vehicle routing problem · Multi-agent systems · Best response planning

1 Introduction

B2B pickup-delivery operations to and from commercial or retail locations involving multiple parties, commonly referred to as Logistics Service Providers (LSPs), more often than not cannot be done in silos. Resource constraints at these locations such as limited parking bays can cause congestion if each LSP adopts an uncoordinated, selfish planning. Thus, some form of coordination is

This research is supported by the National Research Foundation Singapore under its Corp Lab @ University scheme and Fujitsu Limited as part of the A*STAR-Fujitsu-SMU Urban Computing and Engineering Centre of Excellence.

needed to deconflict the schedules of these LSPs to minimize congestion thereby maximizing logistics efficiency. This research is motivated by a real-world problem of improving logistics efficiency in shopping malls involving multiple independent LSPs making B2B pickups and deliveries to these locations in small, congested cities where space is scarce.

Collaborative planning for vehicle routing is an active area of research and had been shown to improve efficiency, service level and sustainability [9]. However, collaborative planning assumes that various LSPs are willing to collaborate with each other by forming coalitions, exchanging of information and/or sharing of resources to achieve a common objective. This is different from our problem setting where LSPs are independent entities who can only make decision locally in response to other LSPs' decisions and they do not interact directly with each other to collaborate or make joint decision.

Ideally if we have one single agent who can control the routes and schedules of multiple LSPs with complete information and collaboration amongst the LSPs, we may achieve some form of system optimality. However, an unintended outcome is that some LSPs may suffer more loss than if they adopt their own planning independently. Moreover, such centralized approach is not scalable and not meaningful in solving the real-world problems, since LSPs may not always be willing to collaborate with one another.

To address the above concern, this paper proposes a scalable, decentralized, coordinated planning approach via iterative best response. The underlying problem can be seen as a Vehicle Routing Problem with Pickup and Delivery, Time Windows and Location Congestion with multiple LSPs (or ML-VRPLC in short).

More precisely, we formulate the problem as a strategic game where each LSP is a self-interested agent willing to participate in a coordinated planning (without collaborating directly with other LSPs) as long as there are sufficient incentives. [1] coined the term "loosely-coupled" agent to describe an agent which exhibits such characteristics. Through an iterative best response procedure, multiple agents adjust their schedules until no further improvement can be obtained to the resulting joint schedule. We seek to find the best joint schedule which maximizes the minimum gain achieved by any one LSP, since LSPs are interested in how much benefit they can gain rather than achieving a system optimality. To realize such gains, we propose to use maximum cost deviation from an ideal solution (a solution that assumes no other LSPs exist to compete for the limited resources) as the performance measure. It is clear that the minimum gain is equivalent to the cost deviation of the worst performing LSP from this ideal solution.

This paper makes the following contributions:

- We define a new variant of VRP, ML-VRPLC and formulate the problem as an n-player strategic game.
- We propose a scalable, decentralized, coordinated planning approach based on iterative best response consisting of a metaheuristic as route optimizer with a scheduler based on Constraint Programming (CP) model to solve a large-scale ML-VRPLC.

– We show experimentally that our approach outperforms a centralized approach in solving large-scale problem within a operationally realistic time limit of 1 h.

2 Related Works

VRPLC is essentially a variant of a classical VRP with Pickup and Delivery, and Time Windows (VRPPDTW) but with cumulative resource constraint at each location [13]. Resources can be in the form of parking bays, cargo storage spaces or special equipment such as forklifts. In VRPLC, there are temporal dependencies between routes and schedules that do not exist in classical VRPs. In classical VRPs, arrival times of vehicles are merely used to ensure time window feasibility. In VRPLC, changes to the time schedule of one route may affect the time schedule of another routes in the form of wait time or time window violation. Many existing approaches to VRP do not take into consideration this relationship between routes and schedules.

[13] proposed a branch-and-price-and-check (BPC) approach to solve a single-LSP VRPLC. It is inspired by a branch-and-cut-and-price method for VRP-PDTW [20] and combines it with a constraint programming subproblem to check the VRPPDTW solutions against the resource constraints. However, BPC approach can only find feasible solutions for instances up to 150 pickup-delivery requests and proves optimality for up to 80 requests given a time limit of 2 h. Therefore, this approach is not scalable when applied directly to solve ML-VPRLC since pickup-delivery requests are usually in the region of hundreds per LSP and for our problem setting, solution is expected within a region of 1 h due to operational requirement. In addition, a direct application of BPC to ML-VRPLC assumes a fully centralized, collaborative planning approach which we have concluded earlier that it may not be practical and not meaningful.

ML-VRPLC can be considered as a problem belonging to an intersection between two main, well-studied research areas namely **Multi-Party VRP** and **Multi-Agent Planning** (MAP). Existing approaches to Multi-Party VRP and MAP can broadly be categorized based on the degrees of collaboration and cooperation respectively.

2.1 ML-VRPLC as a Multi-Party VRP

To solve VRPs involving multiple parties similar to ML-VRPLC, many existing works in the literature focus on collaborative planning approaches. [9] coined the term collaborative vehicle routing and it is a big area of research on its own. Collaborative vehicle routing can be classified into centralized and decentralized collaborative planning. The extent of collaboration ranges from forming of alliances or coalitions (for e.g. [6,11]) to sharing of resources such as sharing of vehicles or exchanging of requests through auction (for e.g. [7,23]). We have established earlier that existing works in this area are not directly applicable to our problem due to the non-collaborative nature of the LSPs.

2.2 ML-VRPLC as an MAP Problem

MAP is simply planning in an environment where there exist multiple agents with concurrent actions. Approaches to MAP can be further categorized into cooperative and non-cooperative domains although most MAP problems lie in between the two domains.

Cooperative Domain. Cooperative MAP involves agents that are not self-interested and are working together to form a joint plan for a common goal [22]. [1] introduced MA-STRIPS, a multi-agent planning model on which many cooperative MAP solvers are based on. [19] proposed a two-step approach consisting of centralized planner to produce local plan for each agent followed by solving a distributed constraint satisfaction problem to obtain a global plan. Meanwhile, [2] introduced the concept of planning games and propose two models namely coalition-planning games and auction-planning games. Those two models assume agents collaborate with each other through forming of coalitions or through an auction mechanism; similar to the approaches within the collaborative vehicle routing domain. In general, the approaches in this domain essentially assume cooperative agents working together to achieve a common goal.

Non-cooperative Domain. Planning in the context of multiple self-interested agents where agents do not fully cooperate or collaborate falls into the domain of non-cooperative game theory. MAP problem can be formulated as strategic game where agents interact with one another to increase their individual payoffs.

[15] proposed a sampled fictitious play algorithm as an optimization heuristic to solve large-scale optimization problems. Optimization problem can be formulated as a n-player game where every pure-strategy equilibrium of a game is a local optimum since no player can change its strategy to improve the objective function. Fictitious play is an iterative procedure in which at each step, players compute their best replies based on the assumption that other players' actions follow a probability distribution based on their past decisions [3]. This approach had been applied to various multi-agent optimization problems where resources are shared and limited such as dynamic traffic network routing [10], mobile units situation awareness problem [14], power management in sensor network [4] and multi-agent orienteering problem [5].

Meanwhile, [12] proposed a best-response planning method to scale up existing multi-agent planning algorithms. The authors used existing single-agent planning algorithm to compute best response of each agent to iteratively improve the initial solution derived from an MAP algorithm. It is scalable compared to applying the MAP algorithm directly to an MAP planning problem. However, the authors evaluated their proposed approach only on standard benchmark problems such as those found in the International Planning Competition (IPC) domains. On the other hand, [8] applied a similar best-response planning approach to a real-world power management problem.

2.3 ML-VRPLC as a Non-cooperative MAP Problem

Given that the LSPs in ML-VRPLC are considered as "loosely-coupled" agents, the approach to solve ML-VRPLC will be somewhere in between cooperative and non-cooperative domains of MAP, although it tends to lean more towards the non-cooperative domain since LSPs are still largely independent and self-interested. Our proposed approach includes certain elements that are discussed above such as non-cooperative game theory and best-response planning. Nevertheless, our work differs mainly from other existing works in that we apply techniques from other research fields (MAP and game theory) on a new variant of a well-studied optimization problem (VRP) with a real-world problem scale.

3 Problem Description

Multiple LSPs have to fulfill a list of pickup-delivery requests within a day. They have multiple vehicles which need to go to the pickup locations to load up the goods and deliver them to various commercial or retail locations such as warehouses and shopping malls. The vehicles need to return to their depot by a certain time and every request has a time window requirement. A wait time will be incurred if the vehicle arrives early and time violations if it serves the request late. In addition, every location has limited parking bays for loading and unloading, and a designated lunch hour break where no delivery is allowed. As such, further wait time and time window violations will be incurred if a vehicle arrives in a location where the parking bays are fully occupied or arrives during the designated lunch hour.

The objective of each LSP is to plan for a schedule that minimizes travel time, wait time and time window violations. Given that parking bays at every location are shared among the multiple LSPs, some sort of coordination is needed to deconflict their schedules to minimize congestion.

4 Model Formulation

4.1 ML-VRPLC as a Strategic Game

We formulate ML-VRPLC as an n-player game $\Gamma_{ML-VRPLC}$ with LSPs represented as players $i \in N$ having a finite set of strategies S_i and sharing the same payoff function i.e. $u^1(s) = ... = u^n(s) = u(s)$. $s \in S_1 \times .... \times S_n$ is a finite set since S_i is finite. Table 1 provides the set of notations and the corresponding descriptions used in the model.

Strategy. In this paper, we will use the terms 'strategy', 'solution' and 'schedule' interchangeably since a strategy of a player i.e. an LSP is represented in the form of a schedule. A schedule is a solution of a single-LSP VRPLC which consists of the routes (sequence of locations visited) of every vehicle and the corresponding time intervals (start and end service times) of every requests served by each vehicle. s_i is represented as the following tuple:

Table 1. Set of notations used in $\Gamma_{ML-VRPLC}$.

Notation	Description
N	A set of LSPs, $N \in \{1, 2, ..., n\}$
s_i	A schedule of LSP i, $i \in N, s_i \in S_i$
s	A joint schedule of all LSP, $s = (s_1, s_2, ..., s_n), s \in S$
s_{-i}	A joint schedule of all LSP except LSP i, $s_{-i} = (s_1, ..., s_{i-1}, s_{i+1}, ..., s_n)$
(s_i, s_{-i})	A joint schedule where LSP i follows a schedule s_i while the rest follows a joint schedule, s_{-i}
$u^i(s)$	Payoff of LSP i when all LSP follows a joint schedule, s
$B_i(s_{-i})$	Best response of LSP i when all other LSPs follow a joint schedule, s_{-i}

$$s_i = \langle s_i.routes, s_i.timeIntervals \rangle$$

Potential Function. We define a function, $P(s) = \sum_{i \in N} u^i(s)$ i.e. total weighted sum of travel times, wait times and time violations when all LSP follows a joint schedule s. In this paper, we define the payoff function, $u^i(s)$ as cost incurred (see Eq. (6) for the full definition). $P(s)$ is an *ordinal potential function* for $\Gamma_{ML-VRPLC}$ since for every $i \in N$ and for every $s_{-i} \in S_{-i}$

$$\begin{gathered} u^i(s_i, s_{-i}) - u^i(s'_i, s_{-i}) > 0 \text{ iff} \\ P(s_i, s_{-i}) - P(s'_i, s_{-i}) > 0 \text{ for every } s_i, s'_i \in S_i. \end{gathered} \tag{1}$$

Proof.

$$\begin{gathered} P(s_i, s_{-i}) - P(s'_i, s_{-i}) > 0 \\ \Rightarrow u^i(s_i, s_{-i}) + \sum_{j \in -i} u^j(s_{-i}) - \Big(u^i(s'_i, s_{-i}) + \sum_{j \in -i} u^j(s_{-i})\Big) > 0 \\ \Rightarrow u^i(s_i, s_{-i}) - u^i(s'_i, s_{-i}) > 0 \end{gathered}$$

Thus, $\Gamma_{ML-VRPLC}$ is a *finite ordinal potential game* and it possesses a pure-strategy equilibrium and has the Finite Improvement Property (FIP) [17]. Having the FIP means that every path generated by a best response procedure in $\Gamma_{ML-VRPLC}$ converges to an equilibrium. We are able to show conceptually and empirically that our approach converges into an equilibrium in the later sections.

Equilibrium and Local Optimality. $s' = (s'_i, s'_{-i})$ is an equilibrium if

$$u^i(s'_i, s'_{-i}) \leq u^i(s_i, s'_{-i}) \text{ for all } i \in N \text{ where } s_i \in B_i(s_{-i}). \tag{2}$$

An equilibrium of $\Gamma_{ML-VRPLC}$ is a local optimum since no player can improve its payoff/reduce its cost by changing its individual schedule. Conversely, every optimal solution, s^* of $\Gamma_{ML-VRPLC}$ is an equilibrium since $u^i(s^*) \leq u^i(s_i, s^*_{-i})$ for all $i \in N$ where $s_i \in B_i(s^*_{-i})$.

Objective Function. The objective of this problem is to minimize the maximum payoff deviation of any one LSP from an ideal solution.

$$min_{s \in S} f(s) \tag{3}$$

$$f(s) = max_{i \in N} Deviation_{LB}(s, i) \tag{4}$$

$$Deviation_{LB}(s, i) = \frac{u^i(s) - u^i(s^{ideal})}{u^i(s^{ideal})} \times 100\% \tag{5}$$

where s^{ideal} is defined as the joint schedule where all other LSPs do not exist to compete for parking bays. s^{ideal} is a Lower Bound (LB) solution since it is a solution of a relaxed $\Gamma_{ML-VRPLC}$. We are essentially trying to search for solutions where each LSP's payoff is as close as possible to its corresponding LB solution.

We do not define the objective function as $min_{s \in S} \sum_{i \in N} u^i(s)$ because in this game, the players are not concerned about the system optimality (total payoffs of all players) but rather on how much benefit it can obtain by adopting a coordinated planning instead of planning independently.

5 Solution Approach

The key idea of our proposed approach is to improve a chosen joint schedule iteratively by computing the best responses of each player assuming the rest of the players adopt the chosen joint schedule until no improvement can be obtained to the resulting joint schedule or until a given time limit or maximum number of iterations has been reached. Our approach is decentralized in nature because each LSP is an independent agent which can compute its own route and schedule i.e. a central agent does not dictate how each player determine their decisions.

Given that we have established that our problem is a *potential game* and has an FIP, our approach will converge to an equilibrium which has been shown earlier to be equivalent to a local optimal solution. Therefore, our approach seeks to explore multiple local optimal solutions until the terminating conditions are met and returns the best one found so far.

5.1 Iterative Best Response Algorithm

Algorithm 1 describes how the iterative best response algorithm works. At each iteration (lines 3–22), a joint schedule is chosen from a sampling pool of previously obtained improved joint schedules or from the current best joint schedule (line 7). We implement an epsilon greedy sampling policy to allow for exploration

Algorithm 1: Iterative Best Response Algorithm to solve ML-VRPLC

```
Input  : Initial joint schedule $s^{initial}$, maximum iteration $K$, time limit $T$
Output: Best found joint schedule $s^{best}$
1  $s^{best} := s^{initial}$, $f_{min} := f(s^{initial})$, $k = 0$
2  Create a sampling pool of joint schedules, $\mathbf{H} = \{s^{initial}\}$
3  while $k < K$ and $runTime < T$ and $\boldsymbol{H} \neq \{\emptyset\}$ do
4      if $k = 0$ then
5          $s^k := s^{initial}$
6      else
7          With probability $\varepsilon$, $s^k \sim U(\mathbf{H})$ otherwise $s^k := s^{best}$
8      end
9      Remove $s^k$ from $\mathbf{H}$
10     Find new joint schedules $\{s^{k,1}, s^{k,2}, ..., s^{k,n}\}$ where
        $s^{k,i} = (s^k_i, s^k_{-i}), u^i(s^{k,i}) < u^i(s^k)$ and $s^k_i \in B_i(s^k_{-i})$
11     if $u^i(s^k) \leq u^i(s^{k,i})$ for all $i \in N$ then
12         $k+ = 1$
13         continue
14     end
15     if $min_{i \in N} f(s^{k,i}) \leq f_{min}$ then
16         $s^{best} := s^{k,i^*}$, $f_{min} := f(s^{k,i^*})$
17         put $\{s^{k,i}\}_{i \in N \setminus \{i^*\}}$ in $\mathbf{H}$
18     else
19         put $\{s^{k,i}\}_{i \in N}$ in $\mathbf{H}$
20     end
21     $k+ = 1$
22 end
23 return $s^{best}$
```

of multiple improvements paths (see Fig. 1 for an example of an improvement path) to search for best joint schedule. An improvement step consisting of $n - 1$ best response computations is applied to the chosen joint schedule to obtain new improved joint schedules (line 10). If no further improvement can be made to the sampled joint schedule, we proceed to the next iteration (lines 11–13). We update the current best joint schedule if any of the new joint schedules has a lower $f(s)$ value than f_{min} (lines 15–16). Otherwise, we place the new improved joint schedules into the sampling pool for further improvement steps in the subsequent iterations (lines 17,19). We repeat the process until termination conditions are met. Then, we return the current best joint schedule as the final output.

Initial Solution, Lower Bound and Upper Bound Solutions. The initial joint schedule can be initialized to any random, feasible joint schedule. However, in this paper, we use the uncoordinated joint schedule as the initial solution to be improved by iterative best response algorithm. To compute the initial joint schedule, $s^{initial}$, we first compute the best schedules for each LSP independently assuming no other LSPs exist to compete for the limited resources. This is akin to solving a single-LSP VRPLC. The resulting joint schedule is in fact s^{ideal} and is the LB solution to $\Gamma_{ML-VRPLC}$. Next, a scheduler consisting of a CP model that incorporates the resource capacity constraint at each location is solved for the combined routes of s^{ideal}. This forms an uncoordinated joint schedule, $s^{uncoord}$ which serves as an Upper Bound (UB) solution to $\Gamma_{ML-VRPLC}$ as any coordinated planning approaches should result in solutions that are better than an uncoordinated one. We use the LB and UB solutions in the experiments to evaluate the solution quality of our proposed approach.

Finite Improvement Paths and Convergence. Each improved joint schedule can be represented as a node in a directed tree. A series of nodes with parent-child relationship forms an improvement path as shown in Fig. 1 where $P(s^{k,i}) < P(s^{k-1,i'})$ for all $k \geq 1$ and $i, i' \in N$. Every improvement path is finite since S is a finite set. Every finite improvement path will converge to an equilibrium and every terminal point is a local optimum. However, since the best response is computed heuristically and there is no way to prove optimality, the resulting equilibrium is just an approximate. Nevertheless, we can show empirically in our experiments that our approach will converge to an approximated equilibrium solution after a certain number of iterations.

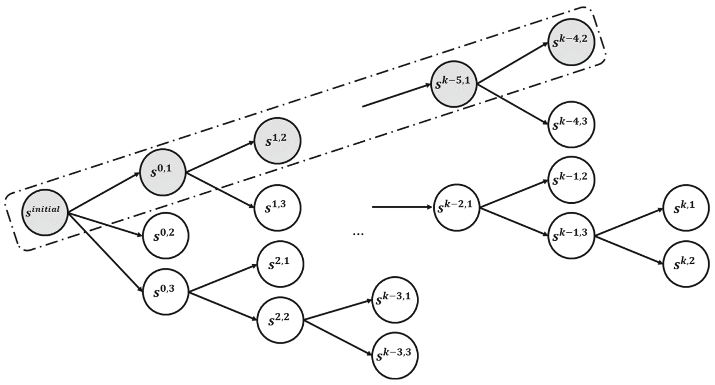


Fig. 1. One example of an improvement path assuming $n = 3$.

In short, our approach explore multiple improvement paths to search for joint schedule that return the best objective value, $f(s)$ with the lowest total payoffs, $P(s)$ as a secondary objective.

5.2 Best Response Computation

At every iteration, best response to a chosen joint schedule, s^k is computed for each LSP (line 10 of Algorithm 1). The best response computation of single LSP is equivalent to solving a single-LSP VRPLC where the resource constraint is determined by the resource utilization of each location by all other LSPs based on s_{-i}^k. Table 2 shows the notations used in this single-LSP VRPLC model.

Table 2. Set of notations used in the single-LSP VRPLC model.

Notation	Description
V	A set of vehicles
R	A set of all requests
M	A set of all locations
R_v	A set of requests served by vehicle v
O_m	A set of requests at location $m \in M$
$C_{m,t}$	Resource capacity at location m at time t
$e_{r,v}$	Lower time window of request r served by vehicle v
$l_{r,v}$	Upper time window of request r served by vehicle v
$prev(r)$	Previous request served prior to request r, $prev(r), r \in R_v$
$d_{x,y}$	Travel time from location of request x to location of request y
$timeInterval_{r,v}$	Time interval when request r in vehicle v is being served, consisting of start and end time
$\overline{T_0}, coolingRate$	Parameters for acceptance criteria in Simulated Annealing

We propose a heuristic consisting of Adaptive Large Neighbourhood Search (ALNS) as route optimizer and a scheduler based on a CP model to solve this single-LSP VRPLC. Heuristic is proposed as it is more scalable for a real-world problem setting. ALNS is used to search for better routes and the CP model based on the resulting routes is then solved to produce a schedule that meets the resource and time-related constraints. ALNS is chosen because it is probably the most effective metaheuristic for the VRPPDTW [16] and ALNS is widely used to solve large-scale problem [24]. Algorithm 2 details the proposed best response computation consisting of ALNS and CP model.

The ALNS algorithm implemented in this paper is adapted from the vanilla version of ALNS proposed by [21] with differences in the choices of the remove and insert operators and parameters used. However, the key difference in our ALNS implementation lies in line 7 of Algorithm 2. To compute the time intervals and the corresponding payoff of the updated solution, a CP model is solved. The payoff is computed as follow:

$$\begin{aligned} u^i(s_i) &= w_1 \times totalTravelTime(s_i.routes) \\ + minimize \sum_{v \in V} &\left\{ w_2 \times \sum_{r \in R_v} waitTime_{r,v} + w_3 \times \sum_{r \in R_v} timeViolation_{r,v} \right\} \end{aligned} \tag{6}$$

Algorithm 2: Best Response Computation

```
Input  : Chosen solution $s^k$, initial temperature $\overline{T_0}$, $coolingRate$
Output: $B_i(s^k_{-i})$
1  $s_i^{best} := s_i^k, s_i := s_i^{input}, \overline{T} = \overline{T_0}$
2  while termination criteria are not met do
3      $s_i' := s_i$
4      Select removal and insert operators via roulette wheel mechanism
5      Apply the selected removal operator to remove the requests from $s_i'.routes$
6      Apply the selected insert operator to insert the orders into $s_i'.routes$
7      Calculate the cost/payoff, $u^i(s_i', s^k_{-i})$ and update $s_i'.timeIntervals$
8      if $u^i(s_i', s^k_{-i}) < u^i(s_i^{best}, s^k_{-i})$ then
9          $s_i^{best} := s_i', s_i := s_i'$
10     else
11         if $u^i(s_i', s^k_{-i}) < u^i(s_i, s^k_{-i})$ then
12             $s_i := s_i'$
13         else
14             $s_i := s_i'$ with probability, $\min\{1, e^{(u^i(s_i, s^k_{-i}) - u(s_i', s^k_{-i}))/\overline{T}}\}$
15         end
16     end
17     Update the weights and scores of the operators accordingly
18     $\overline{T} := \overline{T} * coolingRate$
19 end
20 $B_i(s^k_{-i}) := s_i^{best}$
21 return $B_i(s^k_{-i})$
```

where

$$w_1, w_2, w_3 \text{are predetermined set of weights,}$$
$$waitTime_{r,v} = min\{0, (start(timeInterval_{r,v}) - end(timeInterval_{prev(r),v}) - d_{prev(r),r})\},$$
$$timeViolation_{r,v} = min\{0, (end(timeInterval_{r,v}) - l_{r,v})\},$$
$$s_i.timeIntervals = \{timeInterval_{r,v}\}_{r \in R_v, v \in V}$$

The second term of Eq. (6) is the objective function of the CP model with $\{timeInterval_{r,v}\}_{r \in R_v, v \in V}$ as the primary decision variables of the model. The key constraints of the CP model are as follow:

$$CUMULATIVE(\{timeInterval_{r,v} : v \in V, r \in R_v \cap O_m\}, 1, C_{m,t}), \forall m \in M \tag{7}$$

$$noOverlap(\{timeInterval_{r,v} : r \in R_v\}), \forall v \in V \tag{8}$$

$$start(timeInterval_{r,v}) \geq end(timeInterval_{prev(r),v}) + d_{prev(r),r}, \forall r \in R_v, v \in V \tag{9}$$

$$start(timeInterval_{r,v}) \geq e_{r,v}, \forall r \in R_v, v \in V \tag{10}$$

Constraint (7) is used to model the resource capacity constraint at each location at a given time t where $start(timeInterval_{r,v}) \leq t \leq end(timeInterval_{r,v})$ and

$C_{m,t}$ is determined by the resource utilization of all other LSPs based on s^k_{-i}. Constraint (8) ensures that the time intervals of requests within a route do not overlap. Constraints (9) and (10) ensure that the start time of a request must at least be later than the end time of the previous request plus the corresponding travel time and it should not start before its lower time window. Other constraints relating to operational requirements such as no delivery within lunch hours, operating hours of the locations and vehicles are omitted to simplify the discussion as it is fairly straightforward to incorporate these constraints.

Scalability and Flexibility. Our approach is scalable because the best response computations for every LSP can be done in parallel since they are independent of each other. Our approach is also flexible as it also allows any other forms of solution approach to single-LSP VRPLC to be used to compute the best response.

6 Experiments

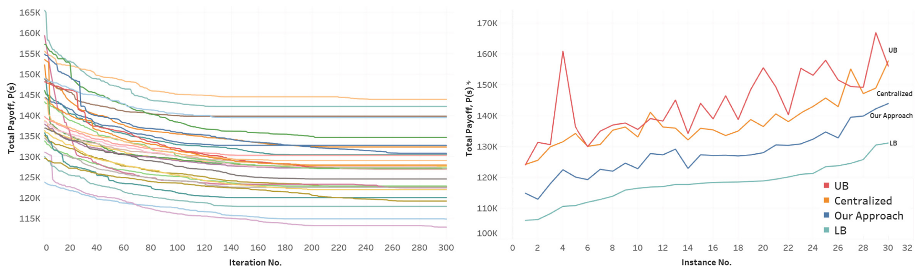
The objective of the experiment is twofold. Firstly, we would like to empirically verify whether our approach converges to an equilibrium for our problem setting and secondly, to evaluate the solution quality produced by our decentralized approach against a centralized approach with respect to s^{ideal} (LB) and $s^{uncoord}$ (UB). Intuitively, our approach should return solutions with lower payoff/cost than UB solution and within a reasonable deviation from LB solution.

6.1 Experimental Setup

We synthetically generate 30 test instances to simulate a month's worth of pickup-delivery requests for 20 LSPs. These instances are generated based on existing datasets of our trials with several local LSPs. Each test instances consists of 100 requests per LSP and each LSP has 10 vehicles. To simulate congestion at the delivery locations, we narrow down the delivery locations to 15 unique shopping malls with maximum capacity of 4 parking bays per location. Our approach is implemented with K set at 300 with $T = 60$ min.

The solution approach is implemented in Java while CP Optimizer ver. 12.8 is used to solve the CP model. The experiments are run on a server with the following configurations: CentOS 8 with 24 CPU Cores and 32 GB RAM.

Benchmark Algorithm. We chose a centralized, non-collaborative planning approach as a benchmark algorithm. It is centralized since all LSPs are treated as one single LSP and the central agent makes the routing and scheduling decision on behalf of the LSPs. It is non-collaborative as no exchange of requests or sharing of vehicles are allowed i.e. each vehicle can only serve requests from the LSP they belong to. We use a heuristic approach combining ALNS and CP



(a) The total payoffs converge for all 30 test instances. Each coloured line represents the result of one test instance.

(b) Our proposed approach outperforms the centralized approach and its solutions are well within the LB and UB solutions.

Fig. 2. Convergence plot and total payoffs across 30 test instances.

model similar to the one used to compute best response to solve this single-LSP VRPLC. The initial solution is constructed via randomized Clarke-Wright Savings Heuristics adapted from [18]. The algorithm is run for 1 h and 2 h for each test instance.

Performance Measures. On top of $f(s)$, we introduce other performance measures to evaluate the two approaches. The other performance measures introduced are as follow:

Maximum Payoff Deviation from an Uncoordinated Solution. $f'(s)$ measures the payoff deviation of the worst performing LSP from the payoff if it follows a schedule based on an uncoordinated planning. A negative deviation value indicates reduction in cost and the lower the value, the higher the improvement gained from the UB solution.

$$f'(s) = max_{i \in N} Deviation_{UB}(s, i) \tag{11}$$

$$Deviation_{UB}(s, i) = \frac{u^i(s) - u^i(s^{uncoor})}{u^i(s^{uncoor})} \times 100\% \tag{12}$$

Average Payoff Deviation from an Ideal Solution. The lower the value, the closer the solution is to the LB solution.

$$g(s) = \frac{1}{n} \times \sum_{i \in N} Deviation_{LB}(s, i) \tag{13}$$

Average Payoff Deviation from an Uncoordinated Solution. Similar to Eq. (11), a negative deviation value indicates reduction in cost.

$$g'(s) = \frac{1}{n} \times \sum_{i \in N} Deviation_{UB}(s, i) \tag{14}$$

6.2 Experimental Results

Convergence. Figure 2a shows that the total payoffs of all players converged after 200 iterations on average for all test instances. This supports our earlier deduction that $\Gamma_{ML-VRPLC}$ possesses an FIP and our proposed algorithm explores multiple improvement path that will converge to an approximated equilibrium. Meanwhile, the average run-time for 200 iterations is around 1 h.

Our Approach vs. Centralized. As shown in Fig. 2b, we intentionally present the results as a line chart and sort the test instances based on increasing total payoff of the ideal solution to better illustrate that our approach returns solutions whose total payoffs are lower than the centralized approach and are well within the UB and LB solutions in all 30 test instances.

Table 3 shows that our approach outperforms the centralized approach on every performance measure even when the run-time for the centralized approach is increased to 2 h. We include results in terms of average and percentiles for a more extensive comparison. In terms of the performance of the worst LSP, our approach is able to ensure that on average, the payoff of the worst performing LSP is still within about 20.7% from the LB solution and at least gain about 2.6% improvement over uncoordinated solution. Meanwhile, even with doubling of the run-time, the centralized approach can only manage to ensure that the payoff of the worst performing LSP is within 31.6% from the LB solution while incurring a 12.9% additional cost as compared to an uncoordinated planning.

On average, across all LSPs, our approach return solutions that are well within 8.3% deviation from the LB solution and improve the payoff of the LSPs by an average of 11.2% from an uncoordinated planning approach. This is contrasted with the centralized approach which can only manage to return solutions that are within 14.4% of LB solution on average and an improvement of about 6.1% from the UB solution even when the run-time is doubled.

We observe that the worst performing LSP in centralized approach consistently returns f' values that are positive (see Table 3) which indicates that the solution for the worst performing LSP is even worse than that of an uncoordinated planning approach. This is because the centralized approach only concerns about the system optimality and not on the performance of each individual LSP. This reiterates our point that a centralized approach may result in some LSPs performing worse than if they are to plan independently.

Experiment Discussion. The experiments show that our proposed decentralized approach outperforms a centralized approach given the available run-time limit of 1 h in all 30 test instances and in all 4 performance measures. Furthermore, we also found that the centralized approach is computationally more expensive and therefore not as scalable as our decentralized approach as it needs longer run-time (>2 h) to return solutions that are at least comparable to our

Table 3. Our approach outperforms the centralized approach on every performance measures across 30 test instances.

Performance measure		Our approach	Centralized (1 h)	Centralized (2 h)
Max payoff	Q1	16.7%	26.3%	24.0%
Deviation from LB	Q2	21.1%	30.7%	27.5%
$f(s)$	Q3	24.0%	36.5%	32.7%
	Avg	**20.7%**	**34.2%**	**31.6%**
Max payoff	Q1	−3.0%	9.9%	6.8%
Deviation from UB	Q2	−1.9%	13.4%	10.1%
$f'(s)$	Q3	−1.1%	16.4%	15.4%
	Avg	**−2.6%**	**12.9%**	**12.9%**
Avg payoff	Q1	5.1%	12.4%	10.3%
Deviation from LB	Q2	8.1%	16.9%	14.2%
$g(s)$	Q3	11.6%	21.5%	18.3%
	Avg	**8.3%**	**16.9%**	**14.4%**
Avg payoff	Q1	−12.4%	−7.1%	−9.5%
Deviation from UB	Q2	−9.1%	1.9%	4.0%
$g'(s)$	Q3	−6.3%	2.9%	0.5%
	Avg	**−11.2%**	**−4.1%**	**−6.1%**

approach. To verify the lack of scalability of the centralized approach, we run another set of experiments with 5 LSPs and find that it indeed performs well with smaller scale problems. Overall, even though there will be LSPs who gain more and others who will gain less, based on our experiments, our approach ensures that there are enough incentives for LSPs to adopt this coordinated planning as compared to them performing their own selfish, independent planning.

7 Conclusion and Future Works

The key idea proposed in this paper is a scalable, decentralized, coordinated planning approach that can be tailored to large-scale optimization problems involving multiple "loosely coupled" entities competing for shared resources. Our proposed iterative best response algorithm decomposes a multi-agent problem into multiple single-agent problems allowing existing single-agent planning algorithms to be applied to a smaller problem.

Even though we assume that the best response algorithms and the payoff functions of each LSP (or agent) are identical, our approach can be extended to problems where each LSP adopts different best response algorithm and payoff function. The best response computation algorithm is akin to a black-box which can be replaced with any solution algorithm to solve single-LSP VRPLC (or single-agent version of the problem). Moreover, even with non-identical payoff functions, the inequality condition in Eq. (1) will still be valid and therefore our approach will still converge to an approximated equilibrium.

One key limitation of our approach is that we assume the environment is static which may not be the case in real-world setting. We assume that every LSP in the system is cooperative in the sense that it participates and adheres to the coordinated planning without any possibility of plan deviation such as dropping out of the system or making changes to their pickup-delivery requests. It is interesting to investigate and enhance our approach to take into consideration uncertainty in the environment and evaluate its robustness in a dynamic environment, as well as to extend it to domains beyond logistics.

Another interesting direction for future work will be to go beyond the empirical study that we did in this paper by further defining and analyzing the theoretical bounds of our approach to n-player game $\Gamma_{ML-VRPLC}$ in terms of the classical notions of Price of Stability (PoS) and Price of Anarchy (PoA).

References

1. Brafman, R.I., Domshlak, C.: From one to many: planning for loosely coupled multi-agent systems. In: Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling, pp. 28–35 (2008)
2. Brafman, R.I., Domshlak, C., Engel, Y., Tennenholtz, M.: Planning games. In: IJCAI, pp. 73–78. Citeseer (2009)
3. Brown, G.W.: Iterative solution of games by fictitious play. Activity Anal. Prod. Alloc. **13**(1), 374–376 (1951)
4. Campos-Nañez, E., Garcia, A., Li, C.: A game-theoretic approach to efficient power management in sensor networks. Oper. Res. **56**(3), 552–561 (2008)
5. Chen, C., Cheng, S.F., Lau, H.C.: Multi-agent orienteering problem with time-dependent capacity constraints. Web Intell. Agent Syst. Int. J. **12**(4), 347–358 (2014)
6. Cuervo, D.P., Vanovermeire, C., Sörensen, K.: Determining collaborative profits in coalitions formed by two partners with varying characteristics. Transp. Res. Part C: Emerg. Technol. **70**, 171–184 (2016)
7. Dai, B., Chen, H.: A multi-agent and auction-based framework and approach for carrier collaboration. Logist. Res. **3**(2–3), 101–120 (2011). https://doi.org/10.1007/s12159-011-0046-9
8. De Nijs, F., Spaan, M.T., de Weerdt, M.M.: Best-response planning of thermostatically controlled loads under power constraints. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
9. Gansterer, M., Hartl, R.F.: Collaborative vehicle routing: a survey. Eur. J. Oper. Res. **268**(1), 1–12 (2018)
10. Garcia, A., Reaume, D., Smith, R.L.: Fictitious play for finding system optimal routings in dynamic traffic networks. Transp. Res. Part B: Methodol. **34**(2), 147–156 (2000)
11. Guajardo, M., Rönnqvist, M., Flisberg, P., Frisk, M.: Collaborative transportation with overlapping coalitions. Eur. J. Oper. Res. **271**(1), 238–249 (2018)
12. Jonsson, A., Rovatsos, M.: Scaling up multiagent planning: a best-response approach. In: Twenty-First International Conference on Automated Planning and Scheduling (2011)

13. Lam, E., Hentenryck, P.V.: A branch-and-price-and-check model for the vehicle routing problem with location congestion. Constraints **21**(3), 394–412 (2016). https://doi.org/10.1007/s10601-016-9241-2
14. Lambert, T.J., Wang, H.: Fictitious play approach to a mobile unit situation awareness problem. Technical report, Univ. Michigan (2003)
15. Lambert Iii, T.J., Epelman, M.A., Smith, R.L.: A fictitious play approach to large-scale optimization. Oper. Res. **53**(3), 477–489 (2005)
16. Li, Y., Chen, H., Prins, C.: Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. Eur. J. Oper. Res. **252**(1), 27–38 (2016)
17. Monderer, D., Shapley, L.S.: Potential games. Games Econ. Behav. **14**(1), 124–143 (1996)
18. Nazari, M., Oroojlooy, A., Takáč, M., Snyder, L.V.: Reinforcement learning for solving the vehicle routing problem. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 9861–9871 (2018)
19. Nissim, R., Brafman, R.I., Domshlak, C.: A general, fully distributed multi-agent planning algorithm. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 1323–1330 (2010)
20. Ropke, S., Cordeau, J.F.: Branch and cut and price for the pickup and delivery problem with time windows. Transp. Sci. **43**(3), 267–286 (2009)
21. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006)
22. Torreño, A., Onaindia, E., Komenda, A., Štolba, M.: Cooperative multi-agent planning: a survey. ACM Comput. Surv. (CSUR) **50**(6), 1–32 (2017)
23. Wang, X., Kopfer, H.: Collaborative transportation planning of less-than-truckload freight. OR Spectr. **36**(2), 357–380 (2013). https://doi.org/10.1007/s00291-013-0331-x
24. Wang, Y., Lei, L., Zhang, D., Lee, L.H.: Towards delivery-as-a-service: effective neighborhood search strategies for integrated delivery optimization of e-commerce and static O2O parcels. Transp. Res. Part B: Methodol. **139**, 38–63 (2020)