SN Computer Science (2023) 4:157
https://doi.org/10.1007/s42979-022-01551-w

ORIGINAL RESEARCH

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

Waldy Joe[1] · Hoong Chuin Lau[1]

Received: 1 February 2022 / Accepted: 9 December 2022

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

Introduction

Business-to-business (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 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 [1]. 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

This article is part of the topical collection "Advances in Multi-Agent Systems Research: EUMAS 2021 Extended Selected Papers" guest edited by Davide Grossi, Ariel Rosenfeld and Nimrod Talmon.

✉ Hoong Chuin Lau
hclau@smu.edu.sg

Waldy Joe
waldy.joe.2018@phdcs.smu.edu.sg

[1] School of Computing and Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902, Singapore

Published online: 11 January 2023

not meaningful in solving 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) (see Fig. 1a and b).

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. [2] 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 more 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:

1. We define a new variant of VRP, ML-VRPLC and formulate the problem as an *n*-player strategic game.
2. 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.
3. We show experimentally that our approach outperforms a centralized approach in solving large-scale problem within an operationally realistic time limit of 1 hour while still providing enough incentives for LSPs to participate in a coordinated planning.

This paper is an extended version of a conference paper of the same title [3]. Besides improving the writeup, we provide extensive details on our proposed routing and scheduling heuristic for the best response computation step, as well as an example to illustrate our proposed approach. We also conduct a comprehensive set of experiments to further evaluate the robustness of our approach by varying the value of a certain key input parameter, testing on different problem sizes and incorporating plan deviations by some of the LSPs.

Related Works

VRP with Location Congestion

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 [4]. 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.

Lam and Van Hentenryck [4] 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 VRPPDTW [5] and combines it with a constraint programming subproblem to check the VRPPDTW solutions against the resource constraints. However, BPC

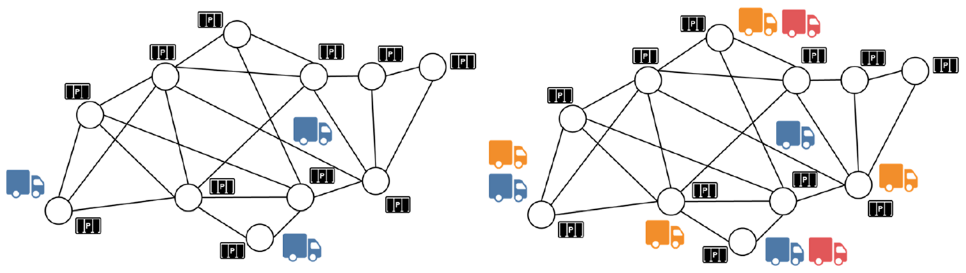


Fig. 1 Problem illustrations for single-LSP VRP with location congestion (VRPLC) and the multi-LSP version of the problem (ML-VRPLC)

(a) Similar to a typical VRP except that there are limited resources at each of the location.

(b) There exist multiple independent LSPs in the environment (represented in different colors). The limited resources at each location are shared among multiple LSPs.

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 order of hundreds per LSP and for our problem setting, a solution is expected within an hour due to operational requirement.

Song et al. [6] studied a similar problem involving docking congestion at shopping malls under travel time and service time uncertainty. They modeled that problem as a two-stage stochastic mixed integer program, developed an adaptive large neighborhood search algorithm that approximates the second stage recourse function using various sample sizes.

A direct application of the above works to ML-VRPLC assumes a fully centralized, collaborative planning approach which we discussed earlier that may not be practical nor meaningful under a multiple LSP context.

ML-VRPLC

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. Based on our understanding of our problem setting, approaches to ML-VRPLC should fall within Quadrant 3 (see Fig. 2) where the agents are non-collaborative but are still willing to cooperate to a certain degree. As ML-VRPLC is a new variant of VRP, there is no prior work done on this problem. Nevertheless, in the following subsections, we discuss existing works that are relevant to solving ML-VRPLC.

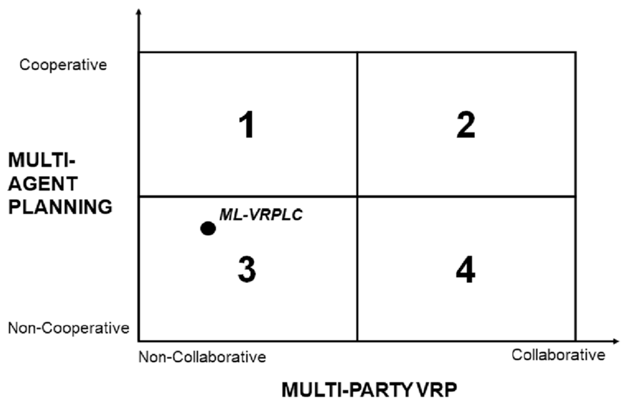


Fig. 2 ML-VRPLC as a multi-party VRP and multi-agent planning problem

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. Gansterer and Hartl [1] 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. [7, 8]) to sharing of resources such as sharing of vehicles or exchanging of requests through auction (for e.g. [9, 10]). 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.

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 these 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 [11]. [2] introduced MA-STRIPS, a multi-agent planning model on which many cooperative MAP solvers are based on. [12] 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. [13] introduced the concept of planning games and proposed 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.

Lambert et al. [14] 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 [15]. This approach had been applied to various multi-agent optimization problems where resources are shared and limited such as dynamic traffic network routing [16], mobile units situation awareness problem [17], power management in sensor network [18] and multi-agent orienteering problem [19].

On a separate front, [20] 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, [21] applied a similar best-response planning approach to a real-world power management problem.

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 approach includes certain elements that are discussed above such as non-cooperative game theory and best-response planning.

There exist prior works that propose MAP approaches for problems with multiple self-interested agents similar to ours. One such work proposes an approach that combines mechanism design (specifically Vickrey-Clarke-Groves (VCG) mechanism) and a distributed planning approach (Multi-Agent A*) [22]. Both this work and ours share some common features such as an assumption that agents are willing to cooperate as long as there are sufficient incentives for them to do so and that agents update their plans in a iterative fashion based on observing the plans of other agents. However, the main key difference between these two approaches is that in [22], agents are able to communicate with other agents directly and each agent keeps track of its individual list of explored and unexplored states/plans. In contrast, for our approach, we assume a central agent that coordinates the message-passing function and keeps track of a central list of states/plans. This difference implies that the search process of our approach is more coordinated and structured while there is no clear structure or order that determines how often, when and which agents to communicate to in [22]. This difference alone results in both strengths and weaknesses for each approach. More coordinated approach can be more efficient as it avoids exploring the same solutions multiple times and incurs lower communication cost while, a more distributed approach is computationally less expensive and does not need to assume the presence of a trusted central agent. Thus, the suitability and the performance of these two approaches will be very much dependent on the specific problem settings and requirements.

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.

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.

Model Formulation

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. $u^1(s) = \cdots = u^n(s) = u(s)$. $s \in S_1 \times \ldots \times S_n$ is a finite set

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

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}

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 to visit) 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:

$s_i = \langle s_i.\text{routes}, s_i.\text{timeIntervals}\rangle$

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 follow 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 $\Gamma_{\text{ML-VRPLC}}$ since for every $i \in N$ and for every $s_{-i} \in S_{-i}$

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

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

Proof

□

Thus, $\Gamma_{\text{ML-VRPLC}}$ is a finite ordinal potential game and it possesses a pure-strategy equilibrium and has the finite improvement property (FIP) [23]. Having the FIP means that every path generated by a best response procedure in $\Gamma_{\text{ML-VRPLC}}$ 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

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

An equilibrium of $\Gamma_{\text{ML-VRPLC}}$ 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 $u^i(s^*) \le u^i(s_i, s^*_{-i})$ 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.

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

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

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

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 $\Gamma_{\text{ML-VRPLC}}$. 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 payoff of all players) but rather on how much benefit it can obtain by adopting a coordinated planning instead of planning independently.

Solution Approach

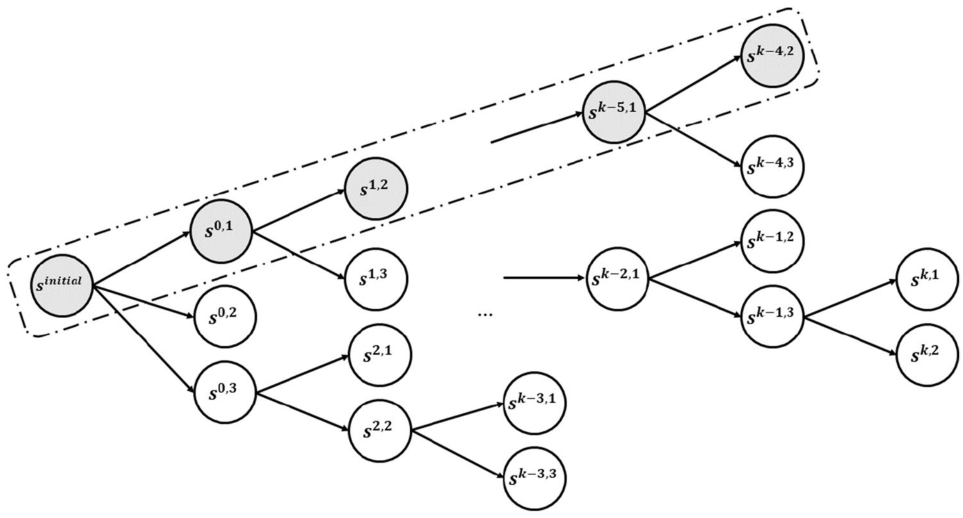
The key idea of our proposed approach is to improve a chosen joint schedule iteratively by computing the best responses of each player assuming the rest of the players adopt the chosen joint schedule until no improvement can be obtained to the resulting joint schedule or until a given time limit or maximum 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 its decision.

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.

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 ε-greedy sampling policy to allow for exploration of multiple improvements paths (see Fig. 3 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. Finally, we return the current best joint schedule as the final output.

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



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 $\mathbf{H} \neq \emptyset$ 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 $s^{best}$
```

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 for each LSP. The resulting joint schedule is in fact s^{ideal} and is the LB solution to $\Gamma_{\text{ML-VRPLC}}$. 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 $\Gamma_{\text{ML-VRPLC}}$ as any coordinated planning approaches must 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. 3 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 (as discussed in "Best Response Computation", computing best response in our problem setting is equivalent to solving a mid-sized VRP with complex constraints which cannot be solved to optimality given the time limit provided), 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.

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

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^k_{-i}. Table 2 shows the notations used in this single-LSP VRPLC model.

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 [24] and ALNS is widely used to solve large-scale problem [25]. Algorithm 2 details the proposed best response computation consisting of ALNS and CP model.

```
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 $B_i(s^k_{-i})$
```

ALNS as Route Optimizer

The ALNS algorithm implemented in this paper is adapted from the vanilla version of ALNS proposed by [26] with differences in the choices of 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 detailed description of the CP model can be found in "CP Model as Scheduler".

As the words adaptive and large in ALNS imply, ALNS explores large search space for new solutions by adaptively

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

choosing remove and insert operators to remove a certain number of orders from existing solution and reinserting them back to other positions to form a new solution (lines 4–6). In our implementation, we use the following remove and insert operators.

Remove Operators

We define the following 5 remove operators and each operator will select $10 - 15\%$ of orders uniformly to be removed from the current solution.

1. *Random removal*: This operator randomly selects orders from the current solution. Random removal allows exploration of larger search space even though the probability of finding a better solution is low.
2. *Worst removal*: This operator selects orders that result in the maximum increase in payoff/cost if removed from the current solution.
3. *Spatio-temporal distance removal*: This operator selects orders which has the highest sum of spatio-temporal distances with their adjacent orders. This operator tries to relocate orders that are more likely to incur higher travel cost and time window violations. The definition of spatio-temporal distance implemented was first introduced in [27].
4. *Time-violation removal*: This operator selects orders that result in highest time window violation. Similar to spatio-temporal distance removal, this operator tries to relocate orders that are "out of position" with respect to their time window requirements.
5. *Shaw removal*: This operator was first introduced in [28]. The basic idea of Shaw removal is to select orders that are similar to each other. The intuition is that removing and reinserting orders that are similar to each other is easier and is more likely to create better solution. Removing and inserting orders that are vastly different from one another can be challenging and may result in unsuccessful reinsertions or poor insertion positions.

The above remove operators are selected to provide both exploitation and exploration in the search process. Randomization in the operator provides the exploration capability while removal operators like worst, spatio-temporal and time-violation are more exploitative in nature as they aim to improve the solution in a greedy manner. Shaw removal is essentially the opposite of worst removal and thus having both operators provides a diversification of search. Shaw removal select orders that are easier to remove and insert while worst removal select orders that are relatively harder to reinsert.

Insert Operators

Prior to any insertion operation, orders are selected for removal using the selected remove operations as described in previous section. The selected orders can be removed in two ways namely remove all or remove one by one. Remove all indicates that the selected orders for removal are removed together prior to insertion while remove one by one indicates that order is removed one at the time prior to insertion. Meanwhile, insertion is done sequentially which means that for remove all, all orders are removed and reinserted one by one while for remove one by one, each order is removed and reinserted one at a time.

These are the six insert operators implemented:

1. *Remove all with greedy insertion*: All orders are removed prior to any insertion. Greedy insertion heuristics insert the order that returns the minimum increase in payoff/cost.

2. *Remove one by one with greedy insertion*: Similar to the first operator except that orders are removed and inserted one at a time.
3. *Remove all with greedy insertion with noise*: Similar to the first operator except that calculation of increase in payoff/cost includes a noise function.
4. *Remove one by one with greedy insertion with noise*: Similar to the second operator except that calculation of increase in payoff/cost includes a noise function.
5. *Remove all with regret-k insertion*: All orders are removed prior to any insertion. This operator inserts orders based on their regret values. Regret value is defined as the difference in the cost of inserting into its best route and its kth best route.
6. *Remove one by one with regret-k insertion*: Similar to the fifth operator except that orders are removed and inserted one at a time.

Similar to the remove operators, the above insert operators are selected to provide both exploitation and exploration in the search process. Randomization in the operator via noise function provides the exploration capability while greedy insertion is more exploitative in nature. In addition, regret insertion provides a certain degree of look ahead capability since greedy insertion is more likely to cause the solution to be stuck at local optima.

On top of the exploration capability provided by the remove and insert operators, a similar mechanism used in Simulated Annealing is applied in choosing poorer solutions to further enlarge the search space and also to escape local optima (line 14). For more detailed discussions of ALNS algorithm such as on the roulette wheel mechanism (line 4) and the adaptive weight adjustment for the operators (line 17), we refer our readers to [26].

CP Model as Scheduler

The following CP model is solved to obtain the updated time intervals of the newly-found routes and the corresponding payoff of the updated schedules at every ALNS iteration in line 7 of Algorithm 2. The payoff is computed as follow:

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

where

w_1, w_2, w_3 are predetermined set of weights,

$$\begin{aligned} & \text{waitTime}_{r,v} = \min\{0, (\text{start}(\text{timeInterval}_{r,v}) \\ & - \text{end}(\text{timeInterval}_{\text{prev}(r),v}) - d_{\text{prev(r),r}})\}, \\ & \text{timeViolation}_{r,v} = \min\{0, (end(\text{timeInterval}_{r,v}) - l_{r,v})\}, \\ & s_i.\text{timeIntervals} = \{\text{timeInterval}_{r,v}\}_{r \in R_v, v \in V} \end{aligned}$$

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

$$\begin{aligned} & \text{CUMULATIVE}(\{\text{timeInterval}_{r,v} : v \in V, \\ & r \in R_v \cap O_m\}, 1, C_{m,t}), \forall m \in M \end{aligned} \tag{7}$$

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

$$\begin{aligned} & \text{start}(\text{timeInterval}_{r,v}) \geq \text{end}(\text{timeInterval}_{\text{prev}(r),v}) \\ & \quad + d_{\text{prev}(r),r}, \forall r \in R_v, v \in V \end{aligned} \tag{9}$$

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

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

Solution Illustration

In this section, we provide a simple illustration on how our approach works using a simple toy example involving 2 LSPs and here, we assume each location has a capacity of one. Our approach begins by computing the initial solution, lower bound and upper bound solutions as explained in "Initial Solution, Lower Bound and Upper Bound". Each LSP plans independently assuming no other LSPs exist to compete for resource and the resulting joint schedule is as follows $s_1 = \langle [A, B, ...], [(5, 7), (9, 11), ...] \rangle$ and $s_2 = \langle [B, D, ...], [(4, 12), (15, 17), ...] \rangle$. This resulting joint schedule is in fact s^{ideal} which is a LB solution. However, it is observed that LSP 1 will need to wait at Location B because LSP 2 is still occupying Location B at time interval (9, 11). Thus, through a CP-based scheduler, this initial joint schedule is revised to a feasible solution such as $s_1 = \langle [A, B, ...], [(5, 7), \mathbf{(13,15)}, ...] \rangle$ while s_2 remains. Due to the waiting time incurred in Location B, the cost of this initial solution is higher. This is a $s^{uncoord}$ since there is no coordination involved and is also an UB solution.

Through a iterative best response procedure, our approach tries to improve this initial solution. For instance, at a given iteration, we first assume s_2 to remain and compute the best response of LSP 1 by using the proposed heuristic (see "Best Response Computation"). At the same time, we compute the best response of LSP 2 assuming s_1 remains. Each of the resulting new joint schedule is kept if its objective value is better than the f_{min} (see line 15–17 of Algorithm 1) or is better than the solution at the start of this iteration (lines 18–19). Assuming that only 1 resulting joint schedule, $s_1 = \langle [A, F, ..., B, ...], [(5, 7), (10, 13), ..., \mathbf{(20,22)}, ...] \rangle$ and $s_2 = \langle [B, D, ...], [(4, 12), (15, 17), ...] \rangle$ returns an improved objective value, we set this schedule as the current best solution, s^{best} and place it in the sampling pool H. This resulting joint schedule is an improved solution because a congestion is avoided at Location B as LSP 1 visits Location B at a later time interval, (20, 22).

At the next iteration, since the sampling pool consists only 1 joint schedule and it is also the current best solution, another round of best response computations is done on this schedule. Assuming that the resulting new joint schedules do not return improved objective values, this current best solution is then returned as the local optimal solution which is also an approximate equilibrium solution since no one player can improve its own payoff. This example illustrates a terminating condition where the sampling pool is empty and no improvement can be made to the current best solution. Another instance of terminating condition will be when the computation time reaches a pre-determined time limit.

In this simple illustration, we simulate 1 iteration of iterative best response procedure. In practice, there would be multiple iterations and each iteration will explore multiple improvement paths since the sampling pool would contain multiple improved joint schedules.

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 (line 10 of Algorithm 1). In other words, explorations of multiple improvement paths as shown in Fig. 3 can be done concurrently. 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.

Experiments

The objective of the experiment is twofold. First, we would like to empirically verify whether our approach converges to an equilibrium for our problem setting and second, to evaluate the solution quality produced by our proposed approach. For a more comprehensive evaluation of our proposed approach, we look into the following aspects in our experiments:

1. *Exploration vs. Exploitation*: We investigate the impact of implementing ε-greedy sampling policy to the solution quality.
2. *Our Approach vs. Centralized*: We compare our proposed decentralized solution approach with 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.
3. *Sensitivity Analysis*: We also investigate the impact of plan deviations by any of the LSPs to solution quality.

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, $T = 60$ mins and $\varepsilon = 0.3$. The implementation codes are written 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 32GB RAM.

Benchmark Algorithm

As there is no existing work that solves the non-collaborative ML-VRPLC, we choose a centralized variant of our proposed non-collaborative planning approach as a benchmark algorithm. It is centralized in that 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 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

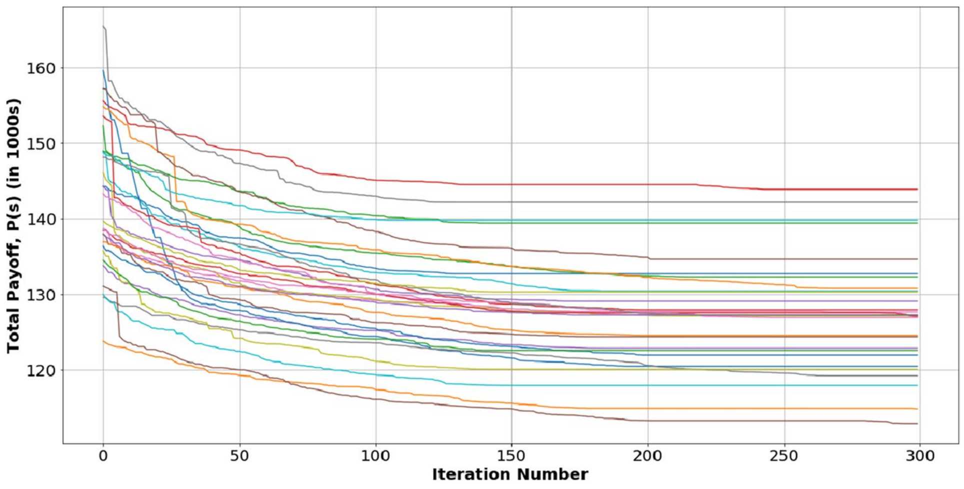


Fig. 4 The total payoffs converge for all 30 test instances. Each coloured line represents the result of one test instance

Table 3 The impact of ε to the solution quality in terms of the objective function, *f*(*s*) and total payoff, *P*(*s*) across 30 test instances

Performance measure		$\varepsilon = 0$	$\varepsilon = 0.3$	$\varepsilon = 0.5$
Max payoff	Q1	19.3%	17.1%	17.0%
deviation from LB	Q2	21.3%	21.1%	20.5%
f(*s*)	Q3	25.0%	24.2%	26.2%
	Avg	**31.8%**	**21.1%**	**26.5%**
Total payoff (in 1000s)	Q1	124.0	122.6	122.1
P(*s*)	Q2	126.5	127.2	128.4
	Q3	130.6	130.7	132.6
	Avg	**128.2**	**127.5**	**128.1**

Bold values are summarized values—more precisely, they show the Avg (average) values by averaging over the values above

Heuristics adapted from [29]. 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 approaches more comprehensively. The other performance measures introduced are as follows:

1. *Maximum payoff deviation from an uncoordinated solution*. $g(s)$ 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 thus, the lower the value, the higher the improvement gained from the UB solution.

$$g(s) = \max_{i\in N}\text{Deviation}_{UB}(s,i) \tag{11}$$

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

2. *Average payoff deviation from an ideal solution*. Unlike *f*(*s*) which measures the payoff deviation of the worst performing LSP, $f'(s)$ measures the average payoff deviation across all LSPs with respect to the ideal solution. The lower the value of $f'(s)$, the closer the solution is to the LB solution on average.

$$f'(s) = \frac{1}{n} \times \sum_{i\in N} \text{Deviation}_{LB}(s,i) \tag{13}$$

3. *Average payoff deviation from an uncoordinated solution*. Similar to $f'(s)$, $g'(s)$ measures the average payoff deviation across all LSPs but with respect to the UB solution. However, like Eq. (11), a negative $g'(s)$ value indicates reduction in cost which translates to the improvement gained from the UB solution.

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

We include results in terms of average and percentiles for a more extensive evaluation since the approaches being evaluated are heuristics and contain a certain degree of stochasticity. However, not all of the performance measures are being used in every experiment. At different parts of the experiments, we select only those which are relevant.

Experimental Results

Convergence

Figure 4 shows that the total payoff of all players converges after 200 iterations on average for all test instances. This supports our earlier deduction that $\Gamma_{\text{ML-VRPLC}}$ possesses an FIP and our proposed algorithm explores multiple

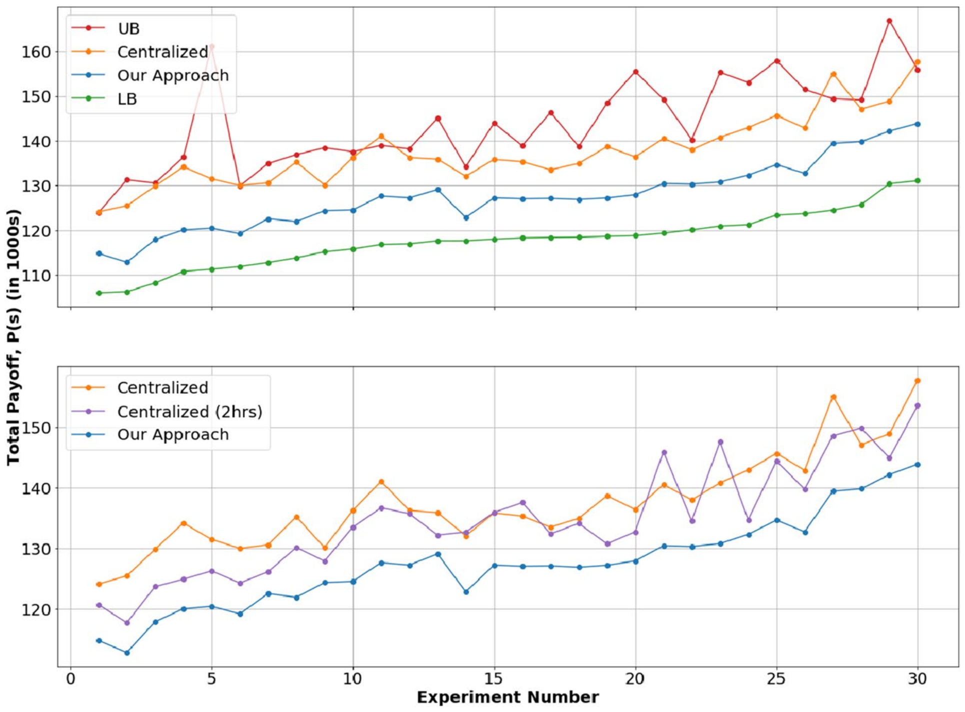


Fig. 5 Our proposed approach outperforms the centralized approach (even when the run-time is doubled) and its solutions are well within the LB and UB solutions in terms of total payoff

Table 4 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	17.1%	28.0%	23.9%
deviation from LB	Q2	21.1%	30.7%	28.8%
f(*s*)	Q3	24.2%	36.5%	33.5%
	Avg	**21.1%**	**35.0%**	**32.0%**
Max payoff	Q1	−3.1%	10.5%	6.8%
deviation from UB	Q2	−2.1%	13.7%	9.2%
g(*s*)	Q3	−1.1%	16.4%	15.4%
	Avg	**−2.7%**	**13.7%**	**13.1%**
Avg payoff	Q1	7.5%	15.5%	12.5%
deviation from LB	Q2	8.6%	17.2%	14.1%
$f'(s)$	Q3	9.4%	18.7%	17.1%
	Avg	**8.6%**	**17.4%**	**14.9%**
Avg payoff	Q1	−11.8%	−5.2%	−7.4%
deviation from UB	Q2	−9.9%	−2.4%	−4.6%
$g'(s)$	Q3	−8.2%	−0.6%	−1.6%
	Avg	**−10.4%**	**−3.0%**	**−5.1%**

Bold values are summarized values—more precisely, they show the Avg (average) values by averaging over the values above

improvement path that will converge to an approximated equilibrium. Meanwhile, the average run-time for 200 iterations is around 1 h.

Exploration vs. Exploitation

We investigate the impact of the value of ε to the solution quality in terms of the objective value, *f*(*s*) and the total payoff, *P*(*s*) as secondary objective function. As mentioned earlier, the value of ε determines the probability of exploring "poorer" improvement path at every best response iteration. $\varepsilon = 0$ implies full exploitation and no exploration, meaning

Table 5 The impact of plan deviations by 10%, 30% and 50% of the LSPs on the solution quality across 30 test instances

Performance measure		10% of LSPs deviate	30% of LSPs deviate	50% of LSPs deviate
Max payoff	Q1	12.5%	13.7%	16.3%
deviation from	Q2	18.7%	19.7%	21.4%
generated plan	Q3	27.7%	25.1%	47.2%
	Avg	**28.5%**	**35.7%**	**39.8%**
Avg payoff	Q1	4.3%	5.3%	6.1%
deviation from	Q2	5.4%	5.8%	8.0%
generated plan	Q3	6.3%	7.3%	10.3%
	Avg	**5.7%**	**7.2%**	**8.4%**

Bold values are summarized values—more precisely, they show the Avg (average) values by averaging over the values above

that each best response procedure is done to improve only the current best solution. In this experiment, we run our solution approach with 3 different ε values (0, 0.3 and 0.5) against the 30 test instances.

As shown in Table 3, although the impact on the total payoff does not seem to be significant, our approach with $\varepsilon = 0.3$ returns solutions where the payoff of worst performing LSP is within 21.1% on average compared to 26.5% and 31.8% when $\varepsilon = 0.5$ and 0 respectively. Based on our experiment, our choice of $\varepsilon = 0.3$ provides a balance between exploration (high ε value) and exploitation (low ε value) and produces better quality solutions consistently across the 30 test instances.

Our Approach vs. Centralized

As shown in Fig. 5, 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 payoff are lower than the centralized approach and are well within the UB and LB solutions in all 30 test instances.

Table 4 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. 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 21.1% from the LB solution and at least gain about 2.7% improvement over the uncoordinated solution. Meanwhile, even with doubling of the runtime, the centralized approach can only manage to ensure that the payoff of the worst performing LSP is within 32.0% from the LB solution while incurring a 13.1% additional cost as compared to an uncoordinated planning.

On average, across all LSPs, our approach return solutions that are well within 8.6% deviation from the LB solution and improve the payoff of the LSPs by an average of 10.4% from an uncoordinated planning approach. This is contrasted with the centralized approach which can only manage to return solutions that are within 14.9% of LB solution on average and an improvement of about 5.1% from the UB solution even when the run-time is doubled.

We observe that the worst performing LSP in centralized approach consistently returns g values that are positive (see Table 4) 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.

Sensitivity Analysis

Our approach assumes that every LSP follow the generated coordinated schedules. However, in real-world settings, there

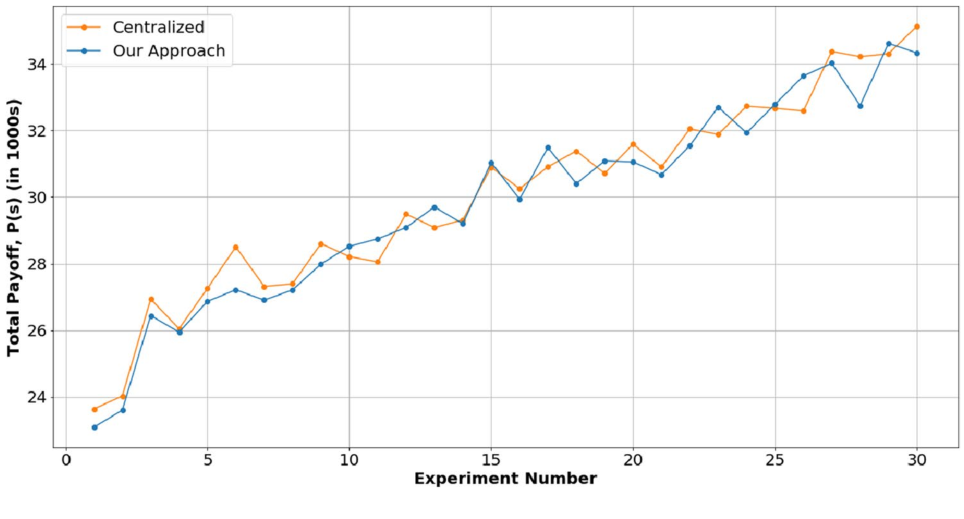


Fig. 6 Our proposed approach and the centralized approach produce comparable solutions in terms of total payoff across 30 test instances (5 LSPs). Similar to Fig. 5, we intentionally present the results as a line chart and sort the test instances based on increasing total payoff of the ideal solution for ease of visualization

are possibilities that some LSPs deviate from the generated plans. In this experiment, we investigate the impact of such plan deviations by any of the LSPs on the solution quality. To simplify the discussion, we assume scenarios where 10%, 30% and 50% of LSPs deviate from the generated schedules and follow their own planned schedules. We assume that LSPs are rational agents which mean that their own schedules are computed with the objective of minimizing their own total payoff/cost.

We generate and run another 30 test instances for this experiment. To evaluate the impact on the solution quality of the resulting schedules, we use the maximum and average payoff deviations from the generated plan as the performance measures. As shown in Table 5, a slight plan deviation caused by only 10% of the LSPs result in the worst performing LSP suffering a loss of almost 30%. Due to the stochastic nature of our approach, the eventual worst performing LSP may not be the same every time the algorithm is run. This may create sufficient deterrence for LSPs from deviating from the generated schedules. On the other hand, even with half of the LSPs not following the generated plans, the average payoff deviation is kept within 10%. This shows that our proposed approach is able to produce solutions that are robust against plan deviations albeit with respect to the average performances of all the LSPs.

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 find 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 approach.

To further verify the performance of the centralized approach and its lack of scalability, we run another set of experiments involving 5 LSPs with 100 pickup-delivery per LSP and each LSP having 10 vehicles. To simulate congestion at the delivery locations, we set the maximum capacity at 2 parking bays per location. Figure 6 shows that both our proposed approach and the centralized approach produce comparable solutions in terms of total payoff across 30 test instances given the same time budget. To further substantiate this claim, we conduct the following paired t test:

$H_0 : \mu_d = 0$

$H_1 : \mu_d \neq 0$ where μ_d refers to the mean difference between the total payoffs of our proposed approach and the centralized approach. In this test, we assume 95% confidence level. The test returns a p-value of 0.79 which is greater than $\alpha = 0.05$. Thus, there is no significant evidence to reject the null hypothesis and as such we can conclude statistically that the total payoff of our proposed approach and centralized approach are comparable. We have shown earlier that the performance gap between the two approaches widens when the problem scale gets larger (see "Our Approach vs. Centralized"). Therefore, the centralized approach indeed performs well only with smaller scale problems.

Although further experimentation may be needed to evaluate the robustness of our approach against various problem scenarios such as varying the number of LSPs, vehicles, requests and pickup-delivery locations, we have deliberately chosen to conduct sufficiently varied types of experiments to show that even though there will be LSPs who gain more and others who will gain less, our approach is able to ensure that there are enough incentives (and deterrence too) for LSPs to adopt and follow this coordinated planning as compared to them performing their own selfish, independent planning.

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 deterministic, which may not be the case in real-world setting. Furthermore, 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. Even though we performed a sensitivity analysis to describe the impact of plan deviations to our solution, it is interesting to further investigate and enhance our approach to take into consideration sources of uncertainty in the environment. One such extension is to take into account stochastic service time i.e. loading and unloading time. In this paper, we assume constant service time.

However, in practice, loading and unloading time may vary and may cause further congestion and disruptions to the planned schedules. Thus, incorporating stochastic service time in the planning approach would result in a more robust and improved solution [30].

Other than to evaluate the robustness of our approach in a stochastic environment, it is also interesting to evaluate the applicability of our approach in other problem domains beyond logistics. Another possible 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 $\Gamma_{\text{ML-VRPLC}}$ in terms of the classical notions of price of stability (PoS) and price of anarchy (PoA).

Declarations

Conflict of Interest This research is funded 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.

References

1. Gansterer M, Hartl RF. Collaborative vehicle routing: a survey. Eur J Oper Res. 2018;268(1):1–12.
2. Brafman RI, 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, 2008;28–35.
3. Joe W, Lau HC. Coordinating multi-party vehicle routing with location congestion via iterative best response. In: Multi-Agent Systems: 18th European Conference, EUMAS 2021, Virtual Event, June 28–29, 2021, Revised Selected Papers, 2021; 72. Springer Nature.
4. Lam E, Van Hentenryck P. A branch-and-price-and-check model for the vehicle routing problem with location congestion. Constraints. 2016;21(3):394–412.
5. Ropke S, Cordeau J-F. Branch and cut and price for the pickup and delivery problem with time windows. Transp Sci. 2009;43(3):267–86.
6. Song R, Lau HC, Luo X, Zhao L. Coordinated delivery to shopping malls with limited docking capacity. Transp Sci. 2022;52:501–27.
7. Cuervo DP, Vanovermeire C, Sörensen K. Determining collaborative profits in coalitions formed by two partners with varying characteristics. Transp Res Part C: Emerg Technol. 2016;70:171–84.
8. Guajardo M, Rönnqvist M, Flisberg P, Frisk M. Collaborative transportation with overlapping coalitions. Eur J Oper Res. 2018;271(1):238–49.
9. Dai B, Chen H. A multi-agent and auction-based framework and approach for carrier collaboration. Logist Res. 2011;3(2–3):101–20.
10. Wang X, Kopfer H. Collaborative transportation planning of less-than-truckload freight. OR Spectr. 2014;36(2):357–80.
11. Torreño A, Onaindia E, Komenda A, Štolba M. Cooperative multi-agent planning: a survey. ACM Comput Surv (CSUR). 2017;50(6):1–32.
12. Nissim R, Brafman RI, Domshlak C. A general, fully distributed multi-agent planning algorithm. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: 2010;1:1323–1330.
13. Brafman RI, Domshlak C, Engel Y, Tennenholtz M. Planning games. In: IJCAI, 2009;73–78. Citeseer.
14. Lambert Iii TJ, Epelman MA, Smith RL. A fictitious play approach to large-scale optimization. Oper Res. 2005;53(3):477–89.
15. Brown GW. Iterative solution of games by fictitious play. Act Anal Prod Alloc. 1951;13(1):374–6.
16. Garcia A, Reaume D, Smith RL. Fictitious play for finding system optimal routings in dynamic traffic networks. Transp Res Part B: Methodol. 2000;34(2):147–56.
17. Lambert TJ, Wang H. Fictitious play approach to a mobile unit situation awareness problem. Univ. Michigan, Tech. Rep 2003.
18. Campos-Nañez E, Garcia A, Li C. A game-theoretic approach to efficient power management in sensor networks. Oper Res. 2008;56(3):552–61.
19. Chen C, Cheng S-F, Lau HC. Multi-agent orienteering problem with time-dependent capacity constraints. Web Intell Agent Syst. 2014;12(4):347–58.
20. Jonsson A, Rovatsos M. Scaling up multiagent planning: a best-response approach. In: Twenty-First International Conference on Automated Planning and Scheduling 2011.
21. De Nijs F, Spaan MT, de Weerdt MM. Best-response planning of thermostatically controlled loads under power constraints. In: Twenty-Ninth AAAI Conference on Artificial Intelligence 2015.
22. Nissim R, Brafman RI. Cost-optimal planning by self-interested agents. In: Twenty-Seventh AAAI Conference on Artificial Intelligence 2013.
23. Monderer D, Shapley LS. Potential games. Games Econ Behav. 1996;14(1):124–43.
24. 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. 2016;252(1):27–38.
25. Wang Y, Lei L, Zhang D, Lee LH. 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. 2020;139:38–63.
26. Ropke S, Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci. 2006;40(4):455–72.
27. Qi M, Lin W-H, Li N, Miao L. A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows. Transp Res Part E: Logis Transp Rev. 2012;48(1):248–57.
28. Shaw P. A new local search algorithm providing high quality solutions to vehicle routing problems. Glasgow: APES Group: Dept of Computer Science, University of Strathclyde; 1997.
29. Nazari M, Oroojlooy A, Takáč M, Snyder LV. Reinforcement learning for solving the vehicle routing problem. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018;9861–9871.
30. de Weerdt MM, Stein S, Gerding EH, Robu V, Jennings NR. Intention-aware routing of electric vehicles. IEEE Trans Intell Transp Syst. 2015;17(5):1472–82.