A Heuristic Algorithm for Multi-Agent Vehicle Routing with Automated Negotiation

Dave de Jonge
IIIA-CSIC
Bellaterra, Catalonia, Spain
davedejonge@iiia.csic.es

Filippo Bistaffa
IIIA-CSIC
Bellaterra, Catalonia, Spain
filippo.bistaffa@iiia.csic.es

Jordi Levy
IIIA-CSIC
Bellaterra, Catalonia, Spain
levy@iiia.csic.es

ABSTRACT

We investigate a problem that lies at the intersection of three research areas, namely Automated Negotiation, Vehicle Routing, and Multi-Objective Optimization. Specifically, we investigate the scenario that multiple competing logistics companies aim to cooperate by delivering truck loads for one another, in order to improve efficiency and reduce the distance they drive. In order to do so, these companies need to find ways to exchange their truck loads such that each of them individually benefits. We present a new heuristic algorithm that, given one set of orders to deliver for each company, tries to find the set of all order-exchanges that are Pareto-optimal and individually rational. Furthermore, we present experiments based on real-world test data from two major logistics companies, which show that our algorithm is able to find hundreds of solutions in a matter of minutes.

KEYWORDS

Vehicle Routing Problem; Automated Negotiation; Multi-objective Optimization

ACM Reference Format:
Dave de Jonge, Filippo Bistaffa, and Jordi Levy. 2021. A Heuristic Algorithm for Multi-Agent Vehicle Routing with Automated Negotiation. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Logistics companies have very small profit margins and are therefore always looking for ways to improve their efficiency. It is not uncommon for such companies to have their trucks only half full when they are on their way to make their deliveries. Moreover, after completing those deliveries they often head back home completely empty. This is a clear waste of resources, not only economically, but also environmentally, as it causes unnecessary emissions of CO_2. For this reason, logistics providers are looking for collaborative solutions that allow them to share trucks with other logistics companies. This type of cooperation, in which multiple companies load their deliveries onto a shared truck, is known as *co-loading*. Finding the optimal co-loading opportunities that minimize the costs of the companies is a difficult problem, because there are many possible solutions, and for each of these solutions, calculating its cost savings amounts to solving a Vehicle Routing Problem (VRP).

This collaborative variant of the VRP has been studied before, but mainly as a *single-objective* optimization problem. That is, one tries to find the solution that minimizes the total cost of all companies involved, and then assumes the benefits will be fairly divided among them according to some pre-defined scheme. In this paper, on the other hand, we are looking at it from the point of view of Automated Negotiations. That is, we are assuming our algorithm only represents one of the companies, and only knows the exact cost function of that company, while it has to make *estimations* of the other companies' costs, because they are kept secret. The goal is to find the set of Pareto-optimal and individually rational solutions, which can then be proposed to the other companies according to some negotiation protocol and some negotiation strategy that aims to maximize the company's own profit.

Of course, even if the opponents' cost functions are only approximately known, one could still consider a single-objective approach, using a standard VRP-solver to find the solution that minimizes the total (estimated) costs of all companies. The problem with this approach is that it only yields one solution, and this solution may not be acceptable to the other companies, either because the estimations were not accurate enough, or because the returned solution is not individually rational. In contrast, our approach has the advantage that it can find a large set of potential proposals, which allows us to propose many alternatives in a negotiation process.

Carrying out this research we had the cooperation of two major logistics providers in the UK, namely Nestlé and Pladis. Although both these companies' primary activity is the production of *fast-moving consumer goods* (i.e. food, beverages and toiletries), they each have a large logistics department with large truck fleets that deliver several hundreds of loads throughout the UK every day. Their main operations consist in carrying products from their factories to their Distributions Centers (DC), and from their DCs to their customers, typically large supermarket chains.

The Vehicle Routing Problem [6] is a generalization of the well-known Traveling Salesman Problem, in which the goal is to find optimal routes for multiple vehicles visiting a set of locations. Many different versions and extensions of this problem have been defined in the literature, such as the *capacitated VRP* in which the vehicles are constrained by volume and/or maximum load weight, the *VRP with pickups and deliveries*, in which the loads have a specific pickup and delivery location, so if a vehicle passes a certain location to pick up a load it should also pass the delivery location of that load, and the *VRP with time windows*, in which the vehicles have to arrive at each location within a given time window. In this paper, we take all these constraints into account, so we are dealing with a *capacitated pickup and delivery problem with time windows* (CPDPTW). Our aim is to to create a system that can actually be used in real life by our industrial partners.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

We should remark that co-loading is only possible if the logistics companies are willing to disclose the locations of their customers to each other. Fortunately, our partners have indicated that this is not a problem for them (their customers are mainly supermarkets, so their locations are not really secret anyway).

2 RELATED WORK

2.1 Vehicle Routing Problems

The VRP was introduced by Dantzig and Ramser in 1959 [6], and is one of the most extensively studied combinatorial optimization problems. In 1964, Clarke and Wright proposed an effective greedy heuristic that improved on the Dantzig–Ramser approach [4]. Following these two seminal papers, hundreds of models and algorithms were proposed for the optimal and approximate solution of the different versions of the VRP. A classification scheme was given in [8]. The VRP has been covered extensively in the books by Toth & Vigo [33] and Golden et al. [14]. Laporte and Nobert [23] presented an extensive survey entirely devoted to exact methods for the VRP, and gave a complete and detailed analysis of the state of the art up to the late 1980s. Even the more specific topic of VRPs with pickups and deliveries and time windows has been studied extensively and comprehensive surveys on this topic have been published [9, 31].

2.2 Collaborative VRP

The collaborative VRP is a variant that involves multiple logistics operators. A recent survey of this topic was presented in [13]. This survey distinguishes between three methodologies: centralized collaborative planning, auction-based decentralized planning, and decentralized planning without auctions. We are interested in the last one. They identify 14 papers of this type, but only four of them deal with VRPs that include Time Windows, and Pickup-and-Delivery [5, 35–37]. Although these approaches are labeled as 'decentralized', this really only means 'not fully centralized' because, although the final decisions are made by the individual logistics companies, there is still a central system that does the search for potential solutions, based on the companies' cost models. Therefore, the collaborative VRP is still mostly a classical *single-objective* optimization problem. In [35], [36], and [37] the goal is to find a globally optimal solution that maximizes the total profit, while in [5] the central system calculates a price that fairly divides the benefits of collaboration among the two collaborating companies.

None of these solutions are feasible in our context, because our industrial partners have indicated that any form of sharing of information about their respective cost models is out of the question, even if it is only shared with a trusted central system.

2.3 Multi-Objective VRP

A large survey of VRPs with multiple objective functions has been conducted in [22]. However, as far as we can see, all papers discussed in this survey assume just a single logistics company with multiple objective functions. For example, a company may wish to strike a balance between minimizing the distance traveled and minimizing the probability of arriving too late, so they try to find all Pareto-optimal solutions w.r.t. to those two objectives. None of the papers in this survey cover the case that there are multiple companies.

2.4 VRP with Negotiations

As explained, many papers have been written that either involve multiple companies with one single shared objective function (the collaborative VRP) or a single company with multiple objective functions (the multi-objective VRP), but much less has been published about VRPs with multiple companies where each company has its own individual objective function. We are aware of only a few papers that do treat somewhat similar problems.

In [34] a case study is presented that explores one-to-many negotiations between one *4PL provider* and several *3PL providers*. A 3PL provider is a logistics company with its own truck fleet while a 4PL provider does not have a fleet, but receives large transport orders from shippers and then redistributes them among 3PL providers. A very similar scenario is treated in [29] and [30], except that they use auction mechanisms instead of negotiations. Two other papers that are closely related to our work, are [17] and [18] which describe an algorithm based on Branch & Bound for negotiations among competing package delivery companies about the exchange of packages. They, however, do not take into account time windows, or volume- and weight- constraints, and they are not using real-world data, but only artificial test cases. Somewhat less relevant to our work, but still related, is [3] which describes a system that carries out negotiation between a port terminal and logistics companies to negotiate time slots for arriving at the terminal.

3 AUTOMATED NEGOTIATION

The research field of Automated Negotiations deals with multi-agent systems in which each agent is purely self-interested, but in which the agents need to cooperate in order to find mutually beneficial solutions. The agents propose potential solutions to each other, which may then be either accepted or rejected. If a proposal is accepted by all agents involved, then each of them obtains the respective utility value it associates with that solution.

Although each agent is purely self-interested, the proposals it makes must also benefit the other agents, because otherwise they would never accept it. Therefore, each agent must strike a balance between maximizing its own utility, and providing enough utility to its opponents to make them accept the proposal. To do this, an agent typically starts by making very selfish proposals, but, as time passes, slowly concedes by making proposals that are less and less selfish. Such a strategy requires the agent to have a large set of potential proposals available, with varying degree of selfishness.

The aim of our work is to develop a negotiating agent that can be applied by a logistics company to negotiate co-loading opportunities with other logistics companies. However, in this paper we only focus on one component of such an agent, namely the search algorithm to find a set of potential proposals. This set of potential proposals can then be fed as the input to some negotiation strategy. The question how to implement such a negotiation strategy is beyond the scope of our work because many such algorithms have already been proposed and implemented [1, 11, 12, 38].

Search algorithms for automated negotiations have been studied, for example using simulated annealing [15, 25, 26], or genetic algorithms [19, 27]. However, these papers only looked at problems in which the utility of a single deal could be computed quickly. They did not involve the complexity of the VRP. Also, as mentioned before, [18] and [17] applied a Branch & Bound approach, but to a simpler and purely artificial scenario.

One important thing we should point out, is that we are assuming the companies only negotiate about which company will deliver which orders, and not about any form of financial compensation for the delivery of another company's orders. There are several reasons for this. Scientifically, price negotiations would make our scenario less interesting because the problem would just be a single-objective optimization problem again, with the goal of minimizing the sum of the costs of the companies. The companies would then only need to negotiate how to divide the joint financial gains. Such 1-dimensional negotiations are not very interesting compared to the state-of-the-art. A more practical reason, is that our partners have indicated that automated price negotiations are not acceptable in a true working system. They require prices to be fixed over a longer term, such as a whole year. Automated day-to-day price negotiations would yield an opaque pricing mechanism with possibly highly fluctuating prices, and this would be a serious problem for their bookkeeping.

So, any form of financial compensation should be fixed in advance, and cannot not be subject to automated negotiation. In this paper we will simply assume the financial compensation is zero, meaning that any company would only accept to make a delivery for another company if that other company returns the favor by making a delivery for the first one.[1]

Our negotiation domain is different from the more commonly studied domains in the automated negotiations literature, in the following two aspects:

(1) Although agents do not have exact knowledge about their opponents' utility functions, they can make reasonable estimations.
(2) Utility functions are expressed as a computationally complex problem (a VRP), so even with perfect knowledge an agent would still not be able to calculate utility values exactly. Instead, it has to resort to heuristic estimations.

Regarding the first point, most studies in automated negotiations assume the agents have absolutely no knowledge at all about their opponents' utility functions [1, 2]. Alternatively, in some work it is assumed that agents have perfect knowledge about each others' utility [21]. In our domain, however, the truth lies somewhere in the middle. The agents do not know each others' exact utility functions, but they can make reasonable estimations. After all, we do know that each company aims to minimize distance and time, and the distances between the locations are known. Furthermore, although each company may pay a somewhat different price for its fuel, the write-off of its vehicles, or the salaries of its drivers, those prices cannot be radically different among the companies.

One main example of a negotiation domain that has been studied extensively and that does also involve these two aspects, is the game of Diplomacy [10, 16, 20, 32]. However, this is a purely artificial game, while in this paper we are studying a real-world scenario.

4 DEFINITIONS

Formally, the problem we tackle in this paper is the following (the precise definitions of the concepts mentioned here are given in the rest of this section). Let $C_1, \ldots C_m$ denote a number of logistics companies. Then, *given a location graph* (L, R, d)*, a distance cost* $dc \in \mathbb{R}$ *a time cost* $tc \in \mathbb{R}$*, and, for each company* C_i *a set of orders* O_i*, a vehicle fleet* V_i *and an initial fleet schedule* fs_i*, find the set of order assignments that are both individually rational and Pareto-optimal with respect to the cost model* (dc, tc).

Here, $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{R}$ the set of real numbers, and $\mathcal{T}$ denotes some set of possible time stamps (e.g. Unix time stamps).

Definition 4.1. The **location graph** (L, R, d) is a weighted graph with vertices L, which we refer to as **locations**, edges R, which we refer to as **roads**, and a weight function $d : R \longrightarrow \mathbb{R}$.

This graph represents a set of possible locations where a logistics provider could pick up or drop off loads (i.e. the factories and distribution centers of the logistics companies, as well as the locations of their customers), and the roads between those locations. The number $d(r)$ represents the distance between two locations, in kilometers. We assume, without loss of generality, that the graph is complete and symmetric and that d satisfies the triangle inequality.

Customers place *orders* with the logistics companies. An order represents a certain number of pallets to be picked up and delivered within specified time windows and at specified locations.

Definition 4.2. An **order** is a tuple $(vol, w, l_{pu}, [t_1, t_2], l_{do}, [t_3, t_4])$, where: $vol \in \mathbb{N}$ is the **volume** of the load, measured as a number of pallets. $w \in \mathbb{R}$ is the **weight** of the load, measured in kilograms. $l_{pu} \in L$ is the **pick-up location**. $t_1 \in \mathcal{T}$ and $t_2 \in \mathcal{T}$ represent the earliest and latest time the load can be picked up, respectively. $l_{do} \in L$ is the **drop-off location**. $t_3 \in \mathcal{T}$ and $t_4 \in \mathcal{T}$ represent the earliest and latest time the load can be dropped off, respectively.

Definition 4.3. A **vehicle** is a tuple (vol_{max}, w_{max}, s), where: $vol_{max} \in \mathbb{N}$ is the **volume** of the vehicle, i.e. the maximum number of pallets it can carry. $w_{max} \in \mathbb{R}$ is **maximum load weight** of the vehicle, measured in kilograms. s is the average **speed** we can realistically assume the vehicle to drive.

4.1 Schedules

We define the solutions of a VRP in terms of what we call *jobs*. A job represents a number of orders scheduled to be picked up and/or a number of orders scheduled to be delivered, by a single vehicle, at a single location, within a specific time window.

Definition 4.4. A **job** J is a tuple: $(l, O_{pu}, O_{do}, t_{ed}, t_{la})$ with: $l \in L$ some location. O_{pu} a (possibly empty) set of **orders to be picked up** at l, O_{do} a (possibly empty) set of **orders to be dropped off** at l, $t_{ed} \in \mathcal{T}$ the earliest possible departure time, and $t_{la} \in \mathcal{T}$ the latest possible arrival time, satisfying the following constraints:

- for each $o \in O_{pu}$ the pick-up location of o must be the location l of this job, and t_{ed} and t_{la} must be consistent with the pick-up time window of o, i.e $t_1 \leq t_{ed}$ and $t_{la} \leq t_2$.
- for each $o \in O_{do}$ the drop-off location of o must be the location l of this job, and t_{ed} and t_{la} must be consistent with the drop-off time window of o, i.e $t_3 \leq t_{ed}$ and $t_{la} \leq t_4$.

A *vehicle-schedule* represents the itinerary of a single vehicle.

Definition 4.5. A **vehicle schedule** is an ordered list: $(J_0, J_1, J_2, \ldots, J_n)$ where each J_i is a job, and $n \in \mathbb{N}$ can be any

[1]More complex deals are also possible, as long as each company involved in the deal benefits.

natural number. Any vehicle schedule must satisfy the following constraints (in the following, the sets of pick-up and drop-off orders of job J_i are denoted as $O_{pu,i}$ and $O_{do,i}$ respectively).

- The jobs are listed in chronological order:
  if $i < j$ then $t_{ed,i} < t_{ed,j}$ and $t_{la,i} < t_{la,j}$.
- Each order appearing in any of the jobs of the vehicle schedule has to be picked up and dropped off exactly once.
- Each order must be picked up *before* it can be dropped off:
  if $o \in O_{pu,i}$ and $o \in O_{do,j}$ then $i < j$.
- The location of J_0 is equal to the location of J_n, and is known as a **depot** (each company has one or more depots).

If o is an order, and vs is a vehicle schedule, we may write $o \in vs$ when we mean that o is picked-up and dropped off by vs. The set of all possible vehicle schedules is denoted VS.

Definition 4.6. A **fleet schedule** fs **for a set of vehicles** V **and a set of orders** O is a map that assigns every vehicle in V to some vehicle schedule vs such that every order $o \in O$ appears in exactly one of these vehicle schedules.

$$fs : V \longrightarrow VS \quad \text{such that} \quad \forall o \in O \;\; \exists! v \in V \;\; : \;\; o \in fs(v)$$

Furthermore, for each vehicle v the corresponding vehicle schedule $vs = fs(v)$ must satisfy:

- After each job of vs, the volume and weight of the orders loaded onto the vehicle must be below vol_{max} and w_{max}.
- The difference between the earliest departure times $t_{ed,i}$ and $t_{ed,i+1}$ of two consecutive jobs of vs must be consistent with the distance between the two locations and the speed s of the vehicle (and the same for the latest possible arrival times).

4.2 Cost Functions

For any vehicle schedule vs we calculate its **cost** $c(vs) \in \mathbb{R}$ as follows:

$$c(vs) \quad := \quad dc \cdot \sum_{i=1}^{n} d(r_i) \;+\; tc \cdot (t_{la,n} - t_{la,0}) \qquad (1)$$

where $dc \in \mathbb{R}$ is the **distance cost**[2] (in euros per kilometer), r_i the road between the locations of J_{i-1} and J_i of vs, $tc \in \mathbb{R}$ is the **time cost** (in euros per hour), $t_{la,n} \in \mathcal{T}$ is the latest possible arrival time of the last job J_n, and $t_{la,0} \in \mathcal{T}$ is the latest possible arrival time of the first job J_0 (which in this case should actually be interpreted as the latest possible *departure* time[3]).

The distance- and time costs dc and tc are together referred to as the **cost model**. In reality, each company would use a different cost model to calculate its own costs. However, since our algorithm represents only one company, and the cost models of the other companies are unknown, it always calculates the costs of any company using the same cost model (of the company it represents). Therefore, the calculated costs are just estimations of the true costs.

[2]Perhaps surprisingly, the distance cost does not depend on how much weight is loaded onto the vehicle. This may seem unrealistic, but this is how many real-world logistics companies do calculate their costs. Furthermore, to keep the discussion simple we here assume that dc does not depend on the vehicle. Our implementation, however, does allow dc to be different for each vehicle.

[3]Our current implementation does not yet take into account the time it takes to load or unload. Therefore, there is no real distinction between arrival- or departure- times. A vehicle departs immediately upon arrival. We will include service times later, but we do not expect this to have a significant impact on the details of our algorithm.

If fs is a fleet schedule for some set of vehicles V, then its **cost** $c(fs) \in \mathbb{R}$ is defined as the sum of the costs of all its vehicle schedules:

$$c(fs) := \sum_{v \in V} c(fs(v)) \qquad (2)$$

4.3 Assignments

Suppose there are m logistics companies $C_1, C_2, \dots C_m$. Each of these companies has a fleet of vehicles V_i and a set of orders O_i to fulfill. We say an order is **owned by** C_i if $o \in O_i$. However, any two companies C_i and C_j may agree with each other that some order o owned by C_i will be picked up and delivered by a vehicle of the other company C_j. In that case we say that an order is **assigned to** C_j.

Definition 4.7. An **order assignment** (or simply **assignment**) α for a set of orders O is a map that assigns each order in O to some company C_i: $\quad \alpha : O \longrightarrow \{C_1, C_2, \dots C_m\}$. We let $O_{\alpha,i}$ denote the set of orders assigned to C_i by α.

$$O_{\alpha,i} := \{o \in O \mid \alpha(o) = C_i\}$$

So, if O consists of all the orders owned by any of the companies and α is an assignment for O then we have $O = \bigcup_{i=1}^{m} O_i = \bigcup_{i=1}^{m} O_{\alpha,i}$.

The **initial assignment** $\overline{\alpha}$ is the assignment that simply assigns each order to the company that owns it, i.e. $\overline{\alpha}(o) = C_i$ iff $o \in O_i$. Therefore, we have $O_{\overline{\alpha},i} = O_i$.

If V_i is the fleet of some company C_i and α some assignment, then $FS_{\alpha,i}$ denotes the set of all possible fleet schedules for fleet V_i and orders $O_{\alpha,i}$. Furthermore, we use $fs^*_{\alpha,i}$ to denote the optimal fleet schedule for company C_i under assignment α. That is:

$$fs^*_{\alpha,i} := \arg\min\{c(fs) \mid fs \in FS_{\alpha,i}\} \qquad (3)$$

and we use $c_i(\alpha)$ to denote the cost of that fleet schedule

$$c_i(\alpha) := c(fs^*_{\alpha,i}) \qquad (4)$$

We say an assignment α **dominates** another assignment α' iff for all $i \in \{1, \dots m\}$ $c_i(\alpha) \leq c_i(\alpha')$, and for at least one of these companies the inequality is strict. We say an assignment α is **Pareto-optimal** iff there is no α' that dominates α, and we say that α is **individually rational** iff it dominates $\overline{\alpha}$.

We should remark here that whenever we use terms like (Pareto-) optimal or 'individually rational', we actually mean (Pareto-)optimal or individually rational *with respect to the cost model* (dc, tc). After all, our algorithm calculates all costs for all companies using that cost model, even though in reality each company would calculate its own costs using a different cost model.

In the language of the automated negotiation literature, our problem is a negotiation domain, where the *agreement space* consists of all possible assignments α for the orders of all companies. The *utility functions* are the (negation of) the cost functions $c_i(\alpha)$ defined by Eq. (4), the *conflict outcome*, representing the case that no agreement is made, is the initial assignment $\overline{\alpha}$, and the *reservation values* are given by $c_i(\overline{\alpha})$.

Note that to calculate $c_i(\alpha)$ we need to find the optimal fleet schedule $fs^*_{\alpha,i}$ which amounts to solving a Vehicle Routing Problem.

5 ORDER PACKAGE HEURISTICS

In order to know which deals to propose, the negotiating agents have to evaluate the possible ways to exchange orders between companies, and find the best ones. If there are m companies and each company has X orders, then there are m^{mX} possible order assignments. For realistic cases this number is astronomical, because our industrial partners each typically have more than a hundred orders to deliver, every day. This means that our problem has two layers of complexity:

(1) There are many possible assignments: m^{mX}.
(2) Given a *single* assignment α, it is complex to calculate its cost $c_i(\alpha)$, because it involves solving a VRP (by Eq. (3)).

Typical (meta-)heuristic search algorithms like genetic algorithms and simulated annealing can solve the problem of the first layer of complexity, because they are able to find good solutions while only evaluating a small fraction of the entire search space. However, such algorithms typically may still require thousands of evaluations, so if each of these evaluations requires solving a VRP the overall algorithm will still be prohibitively slow. For this reason we needed to invent a new heuristic algorithm that can deal with the complexity at both levels. We call it the *Order Package Heuristics*.

The idea is that we first only look at what we call *one-to-one exchanges*, which are exchanges of orders in which one company gives a number of orders that were originally scheduled to be delivered by *the same vehicle* to another company, and that other company incorporates those orders into the schedule of *one* of its vehicles. So 'one-to-one' refers to the fact that the orders are moved from one vehicle schedule to one other vehicle schedule. After determining and evaluating the one-to-one exchanges we can then combine them into more general solutions. Furthermore, when we construct one-to-one exchanges we restrict ourselves to the exchange of sets of orders that correspond to a sequence of consecutive locations to be visited. We call such sets of orders *order packages*.

Our algorithm represents company C_1 and receives as input:

- A location graph (L, R, d).
- A set of orders O_i for each company C_i.
- A set of vehicles V_i for each company C_i.
- The cost model (dc, tc) of company C_1.
- For each company, an initial fleet schedule $fs_i \in FS_{\overline{\alpha},i}$.

The output of the algorithm is:

- A set of assignments $\{\alpha_1, \alpha_2, \dots\}$, which, in the ideal case, would be exactly the set of all Pareto-optimal assignments.

The initial fleet schedules fs_i should approximate the optimal initial schedules $fs^*_{\overline{\alpha},i}$ of each company (i.e. the optimal solution for each company if there was no collaboration at all). These can either be given to us by the other companies, or our agent can determine them by itself using a VRP-solving algorithm (although in that case they may not be the same as the ones actually used by the other companies).

In the rest of this section we give a detailed, step-by-step description of this heuristic.

5.1 Step 1: Find Compatible Order-Vehicle Pairs

Given the orders O_i and the the initial fleet schedule fs_i of each company, we start by determining for each order which vehicles of other companies could adjust their schedules to also pick up and drop off that order. If indeed it is possible for a vehicle v with schedule vs to make two detours to pick up and drop off o then we say that o and vs are compatible, or that o and v are compatible.

Definition 5.1. Let o be an order of one company C_i, let $vs = (J_0, J_1, \dots J_n)$ be a vehicle schedule of another company C_j, and let v be the vehicle scheduled to execute vs (i.e. $vs = fs_j(v)$). We say that o and vs are **compatible** if it is possible to insert two jobs J_{pu}, J_{do} anywhere into vs to obtain a new vehicle schedule

$$vs' = (J_0, \dots J'_k, J_{pu}, , J'_{k+1}, \dots J'_m, J_{do}, J'_{m+1}, \dots J_n)$$

that satisfies all relevant time- and capacity-constraints, where job J_{pu} is the pickup of order o, job J_{do} is the drop-off of order o, and where all other jobs of vs' are exactly the same as those in vs, except that for J_k, J_{k+1}, J_m, and J_{m+1} the latest arrival- and earliest departure times may be adjusted. We then also say that o and v form a **compatible order-vehicle pair**.

The operation of converting vs into vs' is essentially the same as what Li and Lim call the *PD-shift operator* [24].

Knowing all compatible order-vehicle pairs will allow us to prune a large part of the search space in Step 3, because we can discard all solutions involving orders and vehicles that are incompatible.

Time Complexity. If there are m companies and each company has X orders and for each company their initial fleet schedule involves Y vehicle schedules, then there are $mX \cdot (m-1)Y$ possible order-vehicle pairs. For each of these order-vehicle pairs we need to check whether the order and the vehicle schedule are compatible or not. This means we need to check whether the pickup and the drop off of the order can be inserted into the vehicle schedule. If the vehicle schedule has $n + 1$ different jobs then the pickup and the drop-off can both potentially be inserted in n different places, but since the drop off always needs to take place after the pickup, there are $\frac{1}{2}n \cdot (n-1)$ options to check. Furthermore, the value n can be estimated as $n \approx 2X/Y$ (if a company has X orders and Y vehicle schedules, then each vehicle schedule has on average X/Y orders to pick up and drop off, so it may need to visit $2X/Y$ locations). So, for each of the $mX \cdot (m-1)Y$ possible order-vehicle pairs we need to check whether it is compatible or not, which takes $\frac{1}{2} \cdot 2X/Y \cdot ((2X/Y) - 1)$ checks, which yields an overall time complexity of $(mX \cdot (m-1)Y) \cdot \frac{1}{2} \cdot 2X/Y \cdot ((2X/Y) - 1) = O(m^2X^3/Y)$.

Finally, it is fair to say that the number of vehicle schedules of a company should grow linearly with the number of orders, since each vehicle has a limited capacity. Therefore, within the big-O notation we can set X equal to Y, which means that Step 1 has a time complexity of $O(m^2X^2)$

5.2 Step 2: Determine All Order Packages

In the previous step we checked for each individual order whether it is possible to be delivered by some given other vehicle, but in general we want to know whether a *set* of orders can be exchanged from one vehicle (of one company) to another vehicle (of another company). However, since the number of such sets is exponential we only look at a particular type of order set, which we call an *order package*. An order package is a set of orders, originally scheduled in one vehicle schedule, such that if we remove them from the schedule the vehicle can skip a set of *consecutive* locations.

The idea behind this, is that if a few of the locations to be visited by a vehicle are close to each other, then we are most likely to achieve a significant distance reduction if all of those locations are skipped, and such closely clustered locations are likely to be visited consecutively in the original schedule.

If $\mathcal{J}$ is a set of jobs, then let $Ord(\mathcal{J})$ denote the set of all orders that are either picked up or dropped off in any of the jobs in $\mathcal{J}$.

Definition 5.2. Let $vs_d = (J_0, J_1, \dots J_n)$ be a vehicle schedule. An **order package** op **from** vs_d is a set of orders such that there exist two integers k, l with $0 < k < l < n$ for which

$$op = Ord(\{J_k, J_{k+1}, \dots J_l\})$$

The vehicle schedule vs_d is called the **donating vehicle schedule** of op. The vehicle v_d that was scheduled to execute vs_d (i.e. $vs_d = fs_i(v_d)$) is called the **donating vehicle**, and the company C_i that owns v_d and the orders of op is the **donating company**.

Step 2 consists in extracting all order packages from the vehicle schedules of the initial fleet schedules fs_i. For each of these order packages we then calculate the **cost savings** $sav(op)$ associated with it. That is, the difference between the cost $c(vs_d)$ of the original vehicle schedule vs_d minus the cost $c(vs'_d)$ of the new vehicle schedule vs'_d obtained by removing all pick-ups and drop-offs of the orders in op from vs_d.

$$sav(op) := c(vs_d) - c(vs'_d) \tag{5}$$

In order to calculate $c(vs'_d)$ we do not actually need to determine vs'_d itself. Instead, we only need to know its total time and distance (see Eq. (1)). To calculate the distance we simply take vs_d and remove the locations that are skipped. Calculating the new time cost is more difficult, so we simplify it by simply assuming the departure time $t_{la,0}$ and arrival time $t_{la,n}$ at the depot stay the same. In reality, of course, this may be overly pessimistic, so in general the true cost savings will be even better than the calculated ones.

Time Complexity. Given a vehicle schedule vs_d, each order package from vs_d is uniquely defined by the integers k and l, which can be any number between 1 and $n-1$. Therefore, for each vehicle schedule there are $\frac{(n-1)\cdot(n-2)}{2} = O(n^2)$ different order packages. As explained above, n can be estimated as $2X/Y$, so the number of order packages obtained from vs_d is $O(X^2/Y^2)$. Since we obtain the order packages from each vehicle schedule of each company we have to repeat this mY times, so there are $O(X^2/Y^2 \cdot mY) = O(mX^2/Y)$ order packages in total. Furthermore, calculating the cost savings means summing the distances of all n roads between the visited locations, and again using $n \approx 2X/Y$ the total time complexity of Step 2 is $O(mX^2/Y \cdot 2X/Y) = O(mX^3/Y^2)$. Arguing again that X can be set equal to Y, we can simplify this to $O(mX)$.

5.3 Step 3: Generate One-to-One Exchanges

In Step 3 we take all order packages from Step 2, and all vehicle schedules from the initial fleet schedules fs_i and combine them into *one-to-one order exchanges.*

Definition 5.3. A **one-to-one order exchange** or simply **one-to-one exchange** is a pair (op, vs_r) where op is an order package of one company, and vs_r is a vehicle schedule of another company. A one-to-one exchange is **feasible** if it is possible to find a single vehicle schedule vs'_r that delivers all orders of op as well as all orders of vs_r while satisfying all relevant time- and capacity constraints. The schedule vs_r is called the **receiving vehicle schedule**, while the vehicle v_r that was scheduled to execute vs_r (i.e. $fs_i(v_r) = vs_r$) is called the **receiving vehicle**, and the company C_i that owns v_r is the **receiving company**.

Determining whether a one-to-one exchange (op, vs_r) is feasible or not amounts to solving a VRP. For this, we use an existing VRP-solver from the OR-Tools library by Google [28]. Specifically, we take the set consisting of all orders from op and all orders from vs_r and then ask the VRP-solver to find a schedule for a single vehicle that delivers all those orders. If this is indeed possible, the solver will output a new vehicle schedule vs'_r. We then calculate the loss $loss(op, vs_r)$ for the receiving company, which is the difference between the cost $c(vs'_r)$ of this new schedule and the cost $c(vs_r)$ of the original schedule (both calculated with Eq. (1)).

$$loss(op, vs_r) = c(vs'_r) - c(vs_r) \tag{6}$$

However, calling the VRP-solver is computationally expensive, so before doing this we use the results from Step 1 to directly discard many one-to-one exchanges without calling the solver. Specifically, a pair (op, vs_r) is only considered if every order $o \in op$ is compatible (Def. 5.1) with vs_r. All other pairs (op, vs_r) are discarded.

We should note, however, that this procedure may discard many one-to-one exchanges that are actually feasible, because even if some orders of op are not compatible with vs_r it may still be possible to find some vehicle schedule that does deliver all orders. This is because 'compatible' only means that the order can be incorporated in the vehicle schedule with a few minor adjustments. It does not take into account that an entirely re-arranged vehicle schedule could still be found that does succeed in delivering all orders.

After we have obtained the set of feasible one-to-one exchanges, we can again discard many of them. Namely, those that do not yield any overall benefit because the loss for the receiving company is greater than the savings of the donating company, i.e. if $loss(op, vs_r) > sav(op)$.

Time Complexity. The number of one-to-one exchanges equals the number of order packages times the number of vehicle schedules. The first has been calculated to be $O(mX^2/Y)$ and the second is mY, so the number of one-to-one exchanges is $O(m^2X^2)$. For each of these we need to call the VRP-solver. Although calling the VRP-solver is expensive in practice, the formal computational complexity of this step is actually $O(1)$. This is because we are here only using it to solve problem instances with a single vehicle, and the size of such instances is bounded by the capacity constraints of the vehicle. This means that the overall time complexity of Step 3 is $O(m^2X^2)$

5.4 Step 4: Combine One-to-One Exchanges into Full Exchanges

After Step 3 we are left with a set of feasible one-to-one exchanges. Each of these already represents an assignment, but many more assignments can be found if we combine them, so that multiple order packages can be exchanged and loaded onto multiple vehicles. Furthermore, if we do not assume any form of payment between the companies then a single one-to-one exchange would never be an acceptable deal, because the receiving company only loses money. But, if the overall savings of each one-to-one exchange is positive (i.e. $sav(op) > loss(op, vs_r)$) then we can combine them into bundles that are individually rational.

However, not every such bundle is feasible, because several one-to-one exchanges may contradict each other. For example, two different order packages, op_1 and op_2, may contain the same order o, and may appear in two different one-to-one exchanges (op_1, vs_1) and (op_2, vs_2) with different receiving schedules.

Definition 5.4. A **full order exchange** S is a set of one-to-one exchanges, i.e. $S = \{(op_1, vs_1), (op_2, vs_2), \ldots (op_k, vs_k)\}$, such that all order packages are mutually disjoint: $op_i \cap op_j = \emptyset$ for all $i, j \in 1 \ldots k$.

Again, determining the exact set of all full order exchanges is costly, so we simplify this by only looking for those sets S for which each vehicle either:

- only acts as receiving vehicle, in exactly one element of S, or
- only acts as donor vehicle, in one or more elements of S, or
- is not involved in any element of S.

This not only reduces the size of the set of possible solutions, but also has one other very big advantage: it means that for any company the total profit it makes from the deal can be calculated simply as the sum of all its savings minus the sum of all its losses for the elements of S.

The problem of finding the set of full order exchanges that satisfy these criteria and that are Pareto-optimal can now be modeled as a multi-objective optimization problem (MOOP), i.e. a constraint optimization problem with multiple objective functions, with the following features:

- **Variables:** The set of variables is the set of all vehicles (of all companies).
- **Values:** For each vehicle v the set of values that can be assigned to this variable is the set of one-to-one exchanges with v as the receiving vehicle, plus an extra value denoted *none*.
- **Hard constraints:** A vehicle cannot appear both as a receiving vehicle and as a donating vehicle in the same solution. Also, all order packages in S must be mutually disjoint.
- **Objective functions:** There is one objective function for each company, namely the sum of the company's savings over all one-to-one exchanges in the solution, minus the sum of its losses.

A solution to this MOOP is a variable assignment $\{ v_1 \mapsto (op_1, vs_1), v_2 \mapsto (op_2, vs_2), \ldots v_j \mapsto none, \ldots v_k \mapsto (op_k, vs_k) \}$, in which for each vs_i its receiving vehicle is v_i. This can then be converted to the full order exchange $\{(op_1, vs_1), (op_2, vs_2), \ldots (op_k, vs_k)\}$. The hard constraints imply that for every variable assignment $v_i \mapsto (op_i, vs_i)$ in the solution, the donating vehicle v_d of op_i cannot also appear as a receiving vehicle in the same solution, so for the variable v_d the solution must contain the variable assignment $v_d \mapsto none$.

To solve this MOOP we have implemented a multi-objective variant of And/Or Search. And/Or Search [7] is an exact search technique for constraint optimization problems that exploits the fact that not all variables depend on each other, which makes ordinary depth-first search unnecessarily inefficient. We have implemented a new variant of this technique, adapted to MOOPs. The main difference is that, rather than just returning one solution, or all solutions, it returns the set of Pareto-optimal solutions.

As a final step, every full exchange S returned by the algorithm should be converted to an assignment α, but this step trivial. All orders that appear in the order package of any one-to-one exchange in S should be assigned to receiving company of that one-to-one exchange, while all other orders are assigned to their owners.

Time Complexity. Since this step entails solving a MOOP its time complexity is exponential. Each variable of the MOOP corresponds to a vehicle so the time complexity is $O(e^{mY})$ with m the number of companies, and Y the number of vehicles used by each company in the initial solution.

5.5 Discussion

Step 4 of our algorithm still takes exponential time, so one may wonder what we have actually achieved. The point is that the problem to be solved in Step 4 is much simpler than the original problem. Firstly, because the preceding steps have greatly pruned the search space, and secondly because the new problem is an ordinary (multi-objective) constraint optimization problem, in which the objective functions are simple linear functions (the sum of the costs and the losses of the individual one-to-one exchanges). In other words, we have removed the second layer of complexity that we discussed at the beginning of this section.

In summary, our approach is fast for the following reasons: 1) We use the VRP-solver only to evaluate one-to-one exchanges rather than full exchanges, because one-to-one exchanges much smaller, and there are a lot less of them. 2) The number of one-to-one exchanges is reduced by discarding those that involve non-compatible order-vehicle pairs. 3) The number of one-to-one exchanges is further reduced by only considering those that exchange order packages rather than general sets of orders. 4) The number of one-to-one exchanges is reduced even further, by discarding those for which the loss is greater than the savings. 5) We only consider full exchanges in which vehicles can act either as donating vehicle or receiving vehicle, but not both, and in which a vehicle can only receive at most one order package. This has the advantage that the number of full exchanges is reduced and that the cost saving of a full solution can be calculated with a linear formula.

On the other hand, our approach has the disadvantage that it may be pruning the search space too strongly, because the constraints we are imposing on the one-to-one order exchanges and the full order exchanges may cause a number of good solutions to be discarded.

6 EXPERIMENTS

To evaluate our heuristics we have generated 10 test cases from real-world sample data provided by our industrial partners. In each of these test cases the two companies each had 100 orders to deliver on the same day. The total number of locations to be visited by either company varied among the test cases between 117 and 140. The average distance between any two locations of the graph varied between 189 km and 218 km and the diameter of each graph varied between 594 km and 680 km. The average volume of the orders was around 26 pallets. We assumed that each vehicle has a maximum

Table 1: **Number of solutions found by Order Package Search, and total cost reduction of Order Package Search and Single-objective Search.**

Test Case	#Assign.	#IR	Soc. Welf.	Single Obj.
A	535	133	3.99%	9.04%
B	208	53	3.16%	4.87%
C	59	12	2.29%	8.71%
D	229	111	6.90%	9.15%
E	616	305	4.47%	12.7%
F	318	111	4.29%	9.19%
G	105	53	2.85%	7.27%
H	399	53	3.12%	8.46%
I	400	149	7.81%	11.7%
J	325	67	3.77%	8.75%

capacity of 56 pallets or 25,000 kg, and that each company had access to an unlimited supply of vehicles because they can rent them from third parties whenever they do not have enough vehicles themselves.

The experiments were performed on a machine with a 12-core CPU, 3.70GHz and 32GB RAM. Our algorithm was implemented in Java. The results are displayed in Tables 1 and 2.

In Table 1 the first column shows the identifier of each test case. The second column shows for each test case how many assignments were returned by our algorithm. The third column displays how many of them were individually rational. To give an idea of the quality of the returned solutions, we picked for each test case the returned solution with the highest social welfare (i.e. the lowest sum of costs) and calculated how much this solution reduced the joint costs with respect to the initial, non-collaborative solution. This is displayed in the fourth column. In order to compare this with the single-objective approach discussed in the introduction we also displayed the cost reduction of the solution found by the single-objective approach (obtained with the same VRP-solver as we used for Step 3) in the last column.

Our main observation from Table 1, is that the results display high variance among the test cases. For some test cases we find many more solutions than for other test cases. The socially optimal solutions reduce the total costs of the two companies between 2% and 8%. We also notice that in most cases the single-objective search is much better at finding a socially optimal solution, but of course such a search only returns one solution, while our approach yields dozens, or even hundreds of alternatives which can be proposed.

In Table 2 we display the average time it took to execute Steps 3 and 4 of our algorithm, as well as the average time for the single-objective search, for comparison. The time it took to run Steps 1 and 2 of our algorithm was negligible (typically less than 100 ms.), so they are omitted. All values are averaged over 3 repetitions of the experiment and are displayed together with their standard errors.

Again, we see very high variance among the test cases, especially for Step 4. Note that the times in this column are sometimes indicated in milliseconds, and sometimes in seconds. Step 3 took between 94 and 224 seconds, while for step 4 it took between 10 milliseconds and 324 seconds.

Table 2: Run times of Steps 3 and 4 of the Order Package Search, compared with Single-Objective Search.

Test Case	Step 3	Step 4	Single Obj.
A	139 ± 8 sec.	150 ± 26 **sec**.	159 ± 3 sec.
B	138 ± 1 sec.	43 ± 3 ms.	105 ± 1 sec.
C	107 ± 1 sec.	10 ± 4 ms.	114 ± 2 sec.
D	94 ± 1 sec.	780 ± 82 ms.	178 ± 1 sec.
E	175 ± 1 sec.	290 ± 25 ms.	141 ± 1 sec.
F	124 ± 1 sec.	664 ± 221 ms.	135 ± 1 sec.
G	117 ± 1 sec.	36 ± 1 ms.	121 ± 1 sec.
H	209 ± 1 sec.	10 ± 1 **sec**.	98 ± 1 sec.
I	224 ± 1 sec.	324 ± 95 **sec**.	120 ± 1 sec.
J	184 ± 1 sec.	363 ± 3 ms.	106 ± 2 sec.

The reason for these differences, is that the effectiveness of And/Or Search highly depends on the structure of the problem instance. If all variables in the instance depend on each other, then And/Or search is no more effective then a depth-first search. On the other hand, if all variables are completely independent from each other it can solve the problem in linear time. Therefore, small variations between instances can yield vary large variations in run time. Furthermore, the effectiveness of And/Or search also depends heavily on the order in which the variables are evaluated. To find the optimal variable ordering, we use non-deterministic heuristic, so this may sometimes yield less effective orderings. This explains, for example, the high standard error in Test Case I.

7 CONCLUSIONS

We have presented a heuristic algorithm for a problem that, to the best of our knowledge, has never been studied before. Namely, a collaborative VRP without any form of trusted central system and in which the agents do not know each others' cost functions, but are able to estimate them. The goal is, for one agent, to find a large set of potential proposals for the exchange of orders, so that they can be used as the input for a negotiation algorithm. These proposals should ideally be Pareto-optimal and individually rational. We have compared our approach with a single-objective approach and conclude that the two approaches are roughly equally fast. The single-objective approach returns a solution of higher quality, but has the disadvantage that it only yields one solution, so if this solution gets rejected our Order Package approach can be used to find many alternative solutions that can be proposed according to some negotiation strategy.

ACKNOWLEDGMENTS

This work was supported by project LOGISTAR funded by the E.U. Horizon 2020 Research and Innovation Programme, Grant Agreement 769142, by project CI-SUSTAIN funded by the Spanish Ministry of Science and Innovation (PID2019-104156GB-I00), and by a Juan de la Cierva research grant from the Spanish Ministry of Science and Innovation (IJC2018-036443-I).

REFERENCES

[1] Tim Baarslag, Reyhan Aydoğan, Koen V. Hindriks, Katsuhide Fuijita, Takayuki Ito, and Catholijn M. Jonker. 2015. The Automated Negotiating Agents Competition,

2010-2015. *AI Magazine* 36, 4 (12/2015 2015), 115–118. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2609

[2] Tim Baarslag, Koen Hindriks, Catholijn M. Jonker, Sarit Kraus, and Raz Lin. 2010. The First Automated Negotiating Agents Competition (ANAC 2010). In *New Trends in Agent-based Complex Automated Negotiations, Series of Studies in Computational Intelligence*, Takayuki Ito, Minjie Zhang, Valentin Robu, Shaheen Fatima, and Tokuro Matsuo (Eds.). Springer-Verlag.

[3] Csaba Attila Boer, Alexander Verbraeck, Arjen de Waal, Bas van Eck, Jerry Seager, and TBA Nederland ILLYAN. 2003. Distributed e-services for road container transport simulation. In *Proceedings 15th European Simulation Symposium*. 541–550.

[4] Geoff Clarke and John W Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research* 12, 4 (1964), 568–581.

[5] Sascha Dahl and Ulrich Derigs. 2011. Cooperative planning in express carrier networks — An empirical study on the effectiveness of a real-time Decision Support System. *Decision Support Systems* 51, 3 (2011), 620 – 626. https://doi.org/10.1016/j.dss.2011.02.018

[6] George B Dantzig and John H Ramser. 1959. The truck dispatching problem. *Management science* 6, 1 (1959), 80–91.

[7] Rina Dechter and Robert Mateescu. 2007. AND/OR search spaces for graphical models. *Artificial Intelligence* 171, 2–3 (2007), 73 – 106. https://doi.org/10.1016/j.artint.2006.11.003

[8] Martin Desrochers, Jan Karel Lenstra, and Martin WP Savelsbergh. 1990. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research* 46, 3 (1990), 322–332.

[9] Yvan Dumas, Jacques Desrosiers, and Francois Soumis. 1991. The pickup and delivery problem with time windows. *European journal of operational research* 54, 1 (1991), 7–22.

[10] Angela Fabregues and Carles Sierra. 2011. DipGame: a challenging negotiation testbed. *Engineering Applications of Artificial Intelligence* (2011).

[11] Peyman Faratin, Carles Sierra, and Nicholas R. Jennings. 1998. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems* 24, 3-4 (1998), 159 – 182. https://doi.org/10.1016/S0921-8890(98)00029-3 Multi-Agent Rationality.

[12] Peyman Faratin, Carles Sierra, and Nicholas R. Jennings. 2000. Using Similarity criteria to make negotiation trade-offs. In *International Conference on Multi-Agent Systems, ICMAS'00*. 119–126.

[13] Margaretha Gansterer and Richard F Hartl. 2018. Collaborative vehicle routing: a survey. *European Journal of Operational Research* 268, 1 (2018), 1–12.

[14] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. 2008. *The vehicle routing problem: latest advances and new challenges*. Vol. 43. Springer Science & Business Media.

[15] Takayuki Ito, Mark Klein, and Hiromitsu Hattori. 2008. A multi-issue negotiation protocol among agents with nonlinear utility functions. *Multiagent Grid Syst.* 4 (January 2008), 67–83. Issue 1. http://dl.acm.org/citation.cfm?id=1378675.1378678

[16] Dave de Jonge, Tim Baarslag, Reyhan Aydoğan, Catholijn Jonker, Katsuhide Fujita, and Takayuki Ito. 2019. The Challenge of Negotiation in the Game of Diplomacy. In *Agreement Technologies, 6th International Conference, AT 2018, Bergen, Norway, December 6-7, 2018, Revised Selected Papers (Lecture Notes in Computer Science)*, Marin Lujak (Ed.), Vol. 11327. Springer International Publishing, Cham, 100–114. https://doi.org/10.1007/978-3-030-17294-7_8

[17] Dave de Jonge and Carles Sierra. 2012. Automated Negotiation for Package Delivery. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 IEEE Sixth International Conference on*. 83–88. https://doi.org/10.1109/SASOW.2012.23

[18] Dave de Jonge and Carles Sierra. 2015. NB3: a Multilateral Negotiation Algorithm for Large, Non-linear Agreement Spaces with Limited Time. *Autonomous Agents and Multi-Agent Systems* 29, 5 (2015), 896–942. https://doi.org/10.1007/s10458-014-9271-3

[19] Dave de Jonge and Carles Sierra. 2016. GANGSTER: an Automated Negotiator Applying Genetic Algorithms. In *Recent Advances in Agent-based Complex Automated Negotiation*, Naoki Fukuta, Takayuki Ito, Minjie Zhang, Katsuhide Fujita, and Valentin Robu (Eds.). Springer International Publishing, 225–234. http://www.iiia.csic.es/~davedejonge/homepage/files/articles/Gangster.pdf

[20] Dave de Jonge and Carles Sierra. 2017. D-Brane: a Diplomacy Playing Agent for Automated Negotiations Research. *Applied Intelligence* 47, 1 (2017), 158–177. https://doi.org/10.1007/s10489-017-0919-y

[21] Dave de Jonge and Dongmo Zhang. 2020. Strategic negotiations for extensive-form games. *Autonomous Agents and Multi-Agent Systems* 34, 1 (Apr 2020). https://doi.org/10.1007/s10458-019-09424-y

[22] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. 2008. Multi-objective vehicle routing problems. *European journal of operational research* 189, 2 (2008), 293–309.

[23] Gilbert Laporte and Yves Nobert. 1987. Exact algorithms for the vehicle routing problem. In *North-Holland Mathematics Studies*. Vol. 132. Elsevier, 147–184.

[24] Haibing Li and Andrew Lim. 2003. A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools* 12, 02 (2003), 173–186.

[25] Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, Juan R. Velasco, and Enrique de la Hoz. 2009. Effective bidding and deal identification for negotiations in highly nonlinear scenarios. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2* (Budapest, Hungary) *(AAMAS '09)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1057–1064. http://dl.acm.org/citation.cfm?id=1558109.1558160

[26] Makoto Niimi and Takayuki Ito. 2016. AgentM. In *Recent Advances in Agent-based Complex Automated Negotiation*, Naoki Fukuta, Takayuki Ito, Minjie Zhang, Katsuhide Fujita, and Valentin Robu (Eds.). Springer International Publishing, 235–240.

[27] Li Pan, Xudong Luo, Xiangxu Meng, Chunyan Miao, Minghua He, and Xingchen Guo. 2013. A Two-Stage Win-Win Multiattribute Negotiation Model: Optimization and then Concession. *Computational Intelligence* 29, 4 (2013), 577–626. https://doi.org/10.1111/j.1467-8640.2012.00434.x

[28] Laurent Perron and Vincent Furnon. [n.d.]. *OR-Tools*. Google. https://developers.google.com/optimization/

[29] Valentin Robu, Han Noot, Han La Poutré, and Willem-Jan van Schijndel. 2008. An Interactive Platform for Auction-based Allocation of Loads in Transportation Logistics. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track* (Estoril, Portugal) *(AAMAS '08)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 3–10. http://dl.acm.org/citation.cfm?id=1402795.1402797

[30] Valentin Robu, Han Noot, Han La Poutré, and Willem-Jan van Schijndel. 2011. A Multi-agent Platform for Auction-based Allocation of Loads in Transportation Logistics. *Expert Syst. Appl.* 38, 4 (April 2011), 3483–3491. https://doi.org/10.1016/j.eswa.2010.08.136

[31] Martin WP Savelsbergh and Marc Sol. 1995. The general pickup and delivery problem. *Transportation science* 29, 1 (1995), 17–29.

[32] Alexios Theodoridis and Georgios Chalkiadakis. 2020. Monte Carlo Tree Search for the Game of Diplomacy. In *SETN 2020: 11th Hellenic Conference on Artificial Intelligence, Athens, Greece, September 2-4, 2020*, Constantine D. Spyropoulos, Iraklis Varlamis, Ion Androutsopoulos, and Prodromos Malakasiotis (Eds.). ACM, 16–25. https://dl.acm.org/doi/10.1145/3411408.3411413

[33] Paolo Toth and Daniele Vigo. 2002. *The Vehicle Routing Problem*. SIAM monographs on discrete mathematics and applications, Vol. 9. SIAM. https://doi.org/10.1137/1.9780898718515

[34] Sander van der Putten, Valentin Robu, Han La Poutré, Annemiek Jorritsma, and Margo Gal. 2006. Automating Supply Chain Negotiations Using Autonomous Agents: A Case Study in Transportation Logistics. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems* (Hakodate, Japan) *(AAMAS '06)*. ACM, New York, NY, USA, 1506–1513. https://doi.org/10.1145/1160633.1160926

[35] Xin Wang and Herbert Kopfer. 2014. Collaborative transportation planning of less-than-truckload freight. *OR spectrum* 36, 2 (2014), 357–380.

[36] Xin Wang and Herbert Kopfer. 2015. Rolling horizon planning for a dynamic collaborative routing problem with full-truckload pickup and delivery requests. *Flexible Services and Manufacturing Journal* 27, 4 (2015), 509–533.

[37] Xin Wang, Herbert Kopfer, and Michel Gendreau. 2014. Operational transportation planning of freight forwarding companies in horizontal coalitions. *European Journal of Operational Research* 237, 3 (2014), 1133–1141.

[38] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. 2011. Using Gaussian Processes to Optimise Concession in Complex Negotiations against Unknown Opponents. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, Toby Walsh (Ed.). IJCAI/AAAI, 432–438. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-080