Cost of selfishness in the allocation of cities in the Multiple Travelling Salesmen Problem

Thierry Moyaux, Eric Marcon

Cost of selfishness in the allocation of cities in the Multiple Travelling Salesmen Problem

Thierry Moyaux[1] & Eric Marcon[2]
1. Univ Lyon, INSA-Lyon, DISP, EA 4570, F-69621, France.
Corresponding author.
2. Univ Lyon, INSA-Lyon, UJM-Saint Etienne, DISP, EA 4570, F-69621, France.

November 13, 2018

Abstract

The decision to centralise or decentralise human organisations requires quantified evidence but little is available in the literature. We provide such data in a variant of the Multiple Travelling Salesmen Problem (MTSP) in which we study how the allocation sub-problem may be decentralised among selfish selfmen. Our contributions are (i) this modification of the MTSP in order to include selfishness, (ii) the proposition of organisations to solve this modified MTSP, and (iii) the comparison of these organisations. Our 5 organisations may be summarised as follows: (i) OptDecentr is a pure Centralised Organisation (CO) in which a Central Authority (CA) finds the best solution which could be found by a Decentralised Organisation (DO), (ii) Cluster and (iii) Auction are CO/DO hybrids, and (iv) P2P and (v) CNP are pure DO. Sixth and seventh organisations are used as benchmarks: (vi) NoRealloc is a pure DO which ignores the allocation problem, and (vii) FullCentr is a pure CO which solves a different problem, *viz.*, the traditional MTSP. Comparing the efficiency of pairs of these mechanisms quantify the price of decentralising an organisation. In particular, our model of selfishness in OptDecentr makes the total route length 30% (respectively, 60%) longer with 5 (respectively, 9) salesmen than the traditional MTSP in FullCentr when the computation time is limited to 30 minutes. With this time limit, our results also seem to indicate that the level of coercion of the CA impacts more the total route length than the level of centralisation.

Keywords. Centralised Organisation (CO); Decentralised Organisation (DO); Selfishness; Multiple Travelling Salesmen Problem (MTSP).

1 Introduction

Everybody has an opinion about the centralisation of decision. We believe that this opinion depends too much on intuition rather than quantified evidence. In addition, several criteria exist to compare organisations. For

instance, Centralised Organisation (CO) is often said to find the optimum solution (or, at least, the best possible solution), while Decentralised Organisation (DO) is more reactive. Two incomparable metrics are opposed here, namely, quality of the solution found and computation speed. Like comparisons of capitalism and socialism (Ellman, 1989), numerical comparisons of CO and DO rely on comparisons of specific organisations instantiating these levels of centralisation. In other words, it is difficult to provide numerical evidence of the efficiency of CO and DO in general, hence particular mechanisms of these approaches are compared. In this article, we call "mechanism" an instance of an "organisation", and an "organisation" is an instance of an "approach". The possible "approaches" are pure CO and DO, and CO/DO hybrids. Let us illustrate these three terms: The CO/DO hybrid approach can be instantiated as an "organisation" with one or several (coercive) dispatchers, and/or (non-coercive) mediators such as an auctioneer, or peer-to-peer negotiation, etc. Next, the "organisation" with an auctioneer can be instantiated as a "mechanism" as either an English (multiple-shot first-price) or Vickrey (single-shot second-price) auction.

We think that the choice of a more or less centralised organisation is a question too often addressed by political discussions as a consequence of the lack of numerical evidence. This research question is important because the efficiency of organisations may be greatly improved by selecting the appropriate decision organisation. In fact, defenders of CO (respectively, DO) spends much effort in developing their mechanism, while they may obtain much better performances by introducing some features of DO (respectively, CO) — "If your only tool is a hammer then every problem looks like a nail", as the saying goes. Such a choice between various organisations requires numerical data about these organisations, such as the aforementioned quality of the solution and computation time. With such data, an organisation could be designed depending on its constraints; for instance, choosing the number of hierarchical levels and the appropriate organisation on each level depending on the time available to make a decision. In particular, we believe that our work will shed light on the organisation of the Physical Internet (Sarraj et al., 2014) since this project aims at connecting CO logistics networks in a DO way.

The literature provides little quantified comparisons of CO and DO, as noted in a review co-written by one of us (Moyaux and McBurney, 2012). In addition to the papers referenced in this review, we now outline others which compare CO and DO in applications to logistics. In this area, the scarcity of quantified comparisons has also been noted (Mes et al., 2007, p. 60). Similarly, Davidsson et al. (2005) regret the few quantitative comparisons of DO approaches with CO. One of the earliest such comparisons in logistics is a work by Fisher et al. (1995, 1996) who propose an extension of Smith (1980)'s Contract Net Protocol (CNP) in which task decomposition is decentralised in order to solve a static Vehicle Routing Problem with Time Windows. Their experiment shows that their protocol finds a total route length between 3% and 74% worse than the optimal and is thus comparable to heuristics from Operations Research. Then, they propose an improvement which reduces these figures

by about 12%. Since routing problems are NP-hard, the optimal route is often unknown and some studies compare the results of DO with those of approximate CO heuristics. In this context, Mes et al. (2007) report that their Vickrey auction (DO) performs as good as or even better than centralised heuristics to solve a dynamic Pickup and Delivery Problem with Time Windows (PDPTW). For the same problem, Máhr et al. (2010) also compare a Vickrey auction (DO) to the approximate solution found by CPLEX in a few 30-second intervals (CO). Their experimentation shows that the auction outperforms CPLEX when service time duration is highly uncertain. Next, van Lon and Holvoet (2015) compare a cheapest insertion heuristic (CO) with an implementation of Fisher et al. (1995)'s Contract Net Protocol (DO). Their results seem to indicate that DO outperforms CO. Glaschenko et al. (2009) have implemented a real-time multiagent scheduler for a taxi company in London, UK. The benefit of their tool is distributed between the drivers and the company, resulting in an increase of driver wages by 9%. Karmani et al. (2007) use a market-based approach to solve a capacitated version with multiple depot of the MTSP. Their experiment shows that their approach scales to thousands of cities and hundreds of salesmen with a total route length quite close to the optimum. Later, Kivelevitch et al. (2013) were inspired by this approach to propose a distributed meta-heuristic. The total length of the routes are within 1% of the optimal in 90% of the tested cases. More precisely, the median total length of the route is 1.7% higher than the optimum and 4.8% higher in the worst of the tested cases, but the median run time is 8 times faster than CPLEX. Nevertheless, this approach uses agents with no autonomy at all ($e.g.$, an agent can take one or all cities from another agent without permission), hence we think it is more related to parallel algorithms (Talbi, 2006) than to the making of decisions in human organisations with selfish agents as we investigate. Quite similarly, others solve logistic problems ($e.g.$, Travelling Salesman Problem (TSP) by Xie and Liu (2009) and Hanna and Cagan (2009)) with agents who, again, lack the human characteristics of selfishness and cannot hence be used to design organisations as well.

The goal of this article is to compare CO with DO by measuring the efficiency of several mechanisms on various levels of centralisation to solve a modified Multiple Travelling Salesmen Problem (MTSP) (MTSP is the Vehicle Routing Problem without capacities) such that the selfishness of the salesmen is taken into account. It is important to notice that our salesmen are selfishness because we want to investigate human organisations. This article has the following contributions:

1. *Conceptual contributions*: CO and DO are different by nature because, for example, the social welfare ($i.e.$, utility function of the group) in CO may have no trivial relationship with the individual utility functions in DO. Game Theory proposes models and tools to study CO and DO, $e.g.$, the Prisoner's Dilemma shows that DO may find a solution ($i.e.$, Nash equilibrium) which is Pareto dominated by the solution found by CO. Taking this game-theoretic background into account, we modify the MTSP by adding features representing the selfishness of the salesmen such that our modified MTSP

can be solved in more or less centralised organisations. Selfishness means that a salesman maximises his[1] own utility in a selfish manner without paying attention to the social welfare of the community of salesmen.

Subsection 2.2 introduces our MTSP constrained by 1-1 exchanges. More precisely, the salesmen "own" an initial endowment of cities, and they use a mechanism to modify this initial allocation of cities by exchanging one city against one city exclusively, which we refer to as "1-1 exchanges". This corresponds to our modified MTSP. Section 6 discusses other possibilities to obtain a pair of CO and DO versions of a problem.

2. *Technical contributions*: As said in the above literature review, no previous work compares more than two kinds of organisations to solve a same location and routing problem. Our organisations for the exchange of cities are inspired by the classes of coordination proposed in (Frayret, 2002, p. 131) as an extension to Mintzberg (1978). To be precise, this paper studies the following organisations which have various numbers of hierarchical levels, coercion levels and number of rounds of interactions, as summarised in Figure 1:

    1. OptDecentr makes the Central Authority (CA) find both (i) the best allocation of cities to salesmen with the constraint of 1-1 exchanges and (ii) for each salesman, the shortest path visiting all the cities allocated to this salesman – this is a CO mechanism which looks for the best solution which may be found by the following five DO mechanisms;
    2. Cluster makes the CA allocate groups of neighbouring cities to the salesmen with the constraint of 1-1 exchanges, then every salesman locally solves a TSP with his allocated cities;
    3. Auction is similar to Cluster except that the CA is less coercive since she plays the role of an auctioneer;
    4. CNP (Contract Net Protocol) has no CA and every salesman plays the role of an auctioneer;
    5. P2P also has no CA and relies on bilateral negotiations.

    Finally, two additional organisations are used as benchmarks:

    6. NoRealloc assumes no reallocations of the cities among the salesmen who only solve a TSP on their initial endowment;
    7. FullCentr is the same as OptDecentr without the constraint of 1-1 exchanges, that is FullCentr is the only mechanism that solves the traditional MTSP.

    Since our experimentation simulates more agents than the number of cores of the CPU in our computers, we perform only sequential simulations, then we infer what would have happened in real life with computations carried out in parallel.

    As said above, we call "mechanism" an instance of an "organisation". However, this article uses both terms interchangeably because

[1] We always use "him" for a salesman and "her" for the Central Authority (CA).

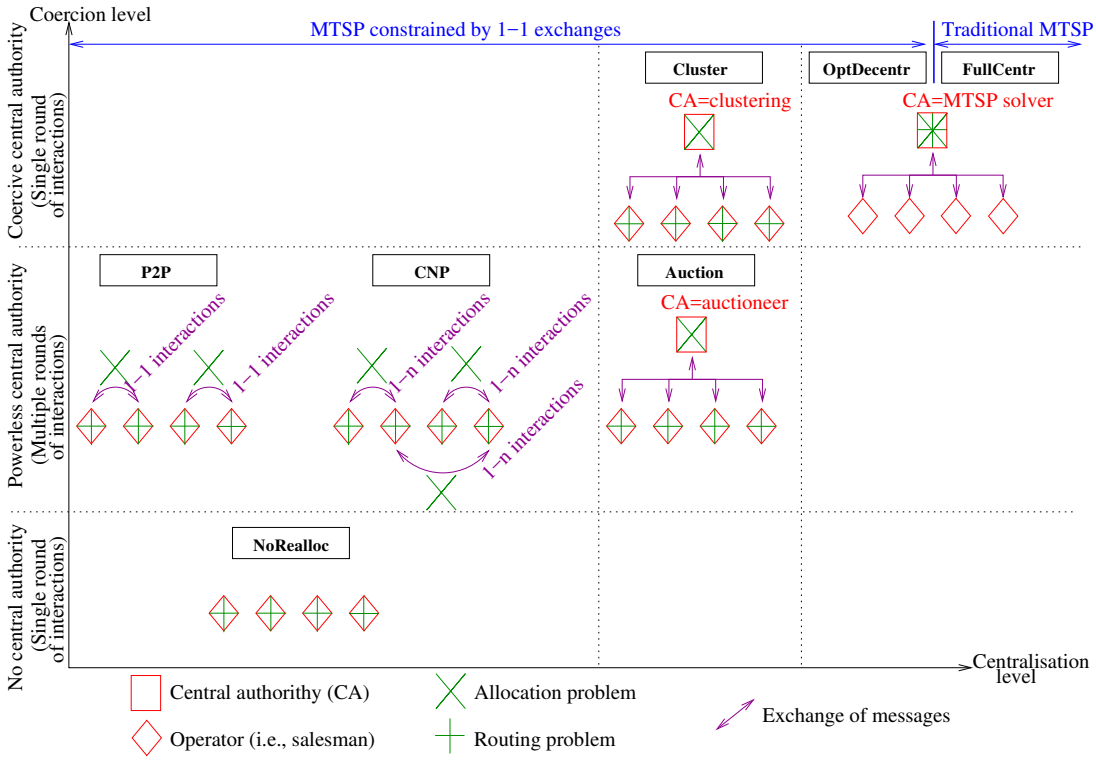


Figure 1: Overview of the seven organisations compared.

it deals with one mechanism per organisation only. In fact, each of the above organisations may be instantiated in mechanisms different from our descriptions in Section 3, as shall be discussed in Section 6. Finally, our mechanisms instantiating the above first five organisations are new, since they solve the MTSP constrained by 1-1 exchanges which we have never seen in the literature. On the contrary, benchmarks NoRealloc and FullCentr are not new as they solve a classic Mixed-Integer Linear Programming (MILP) formulation of TSP and MTSP.

3. *Numerical contributions*: Davidsson et al. (2007) compare four mechanisms on a same problem and we do not know of any work with more mechanisms. In comparison with this work, our work both consider more mechanisms and present many more experimental results. In fact, our results are robust since we compare the efficiency of the mechanisms when they solve the same instance among 130 for given numbers of salesmen and clients, and we present the fifth and ninth deciles obtained by 130 instances.

The outline of this paper is as follows. Section 2 presents our framework, $i.e.$, our MTSP constrained by 1-1 exchanges, next hypotheses about how to carry out a fair comparisons of the mechanisms. Section 3 details the seven mechanisms. Section 4 shows how the duration of the parallel

computations of up to 10 agents (9 salesmen + 1 Central Authority (CA)) is inferred from the sequential computations in our experimentation on one of our computers with a 4-core CPU. Section 5 presents the numerical results of this experimentation. Section 6 discusses these results, next how other variants of MTSP may be introduced taking the selfishness of the salesmen into account. Section 7 concludes.

2 Framework

This section first recalls the formulation of the traditional MTSP solved by Mechanism FullCentr, next our MTSP constrained by 1-1 exchanges of cities which is solved by the other six mechanisms. We assume that both problems have n cities to be visited by one of m salesmen, and d_{ij} is the Euclidean distance between Cities i and j. City $i = 0$ is the depot shared by all salesmen.

2.1 Traditional MTSP: FullCentr

The traditional MTSP in Equations 1-8 uses the 2-index decision variable x_{ij} such that $x_{ij} = 1$ only if one of the m salesmen goes from City i to City j. The objective function in Equation 1 minimises the total distance U^{trad} travelled by all the salesmen. Equation 2 (respectively, 5) ensures that m salesmen leave (respectively, enter) the depot. Similarly, Equation 3 (respectively, 4) ensures that a single salesman leaves (respectively, enters) every city. Equation 6 is a constraint eliminating sub-routes by the method of node potentials proposed by Miller et al. (1960) in which decision variable p_i counts the number of cities visited by a salesman before he visits City i.

$$\min U^{\text{trad}} = \quad \sum_{i=0}^{n-1} \sum_{j=0, j\neq i}^{n-1} d_{ij} x_{ij} \tag{1}$$

$$\sum_{j=1}^{n-1} x_{0j} = m \tag{2}$$

$$\sum_{j=0, j\neq i}^{n-1} x_{ij} = 1 \qquad 1 \leq i < n \tag{3}$$

$$\sum_{i=0, i\neq j}^{n-1} x_{ij} = 1 \qquad 1 \leq j < n \tag{4}$$

$$\sum_{i=1}^{n-1} x_{i0} = m \tag{5}$$

$$p_i - p_j + (n-1).x_{ij} \leq n-2 \qquad 1 \leq i \neq j < n \tag{6}$$

$$p_i \in \Re^+ \qquad 1 \leq i < n \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad 0 \leq i, j < n \tag{8}$$

2.2 Modified MTSP (constrained by 1-1 exchanges)

We now explain how we modify this traditional MTSP in order to include the individual selfishness of the salesmen such that a DO can also solve our modified problem. The following explanation calls U^{trad} the social welfare in the traditional MTSP ($i.e.$, the total route length in Eq. 1), and U^{mod} the social welfare and u_k^{mod} Salesman k's utility in the modified MTSP. We make the following assumptions:

1. Hyp. 1 – *Individual utilities measure the travelled distance*: We define u_k^{mod} as the distance travelled by Salesman k and $U^{\mathrm{mod}} = \sum_k u_k^{\mathrm{mod}}$ as the distance travelled by all. Of course, the advantage of such a hypothesis is that $U^{\mathrm{mod}} = U^{\mathrm{trad}}$.

   The drawback is that salesmen all want to get rid of their cities, but no one accepts cities from other salesmen. Consequently, such choices of u_k^{mod} and U^{mod} require the following two assumptions which operate together.

2. Hyp. 2 – *Salesmen "own" an initial endowment of cities*: Since Hyp. 1 makes our distance-minimiser salesmen all want to reduce the number of cities allocated to them, we cannot assume that they will "fight" in order to obtain cities from a shared pool. Instead, we assume that every Salesman k initially "owns" some cities, and he tries to change this initial allocation by exchanging cities whenever such an exchange reduces his individual distance u_k^{mod}.

3. Hyp. 3 – *Only 1-1 exchanges of cities are possible*: Let us study what happens when a Salesman k gives n cities to Salesman k' and receives n' cities from k' in a round of interaction. All exchanges with $n = n'$ are rational for both salesmen when they reduce both u_k^{mod} and $u_{k'}^{\mathrm{mod}}$, as shown in the next paragraph. This constraint of $n - n$ exchanges solves the above problem caused by selfishness which makes salesmen want to give but not receive cities. In this article, we only consider the case $n = n' = 1$; Cases $n = n' > 1$ are discussed in Section 6.

   Let us detail why $n < n'$ cannot be accepted by k' (the case $n > n'$ is similar). Without knowing the cities allocated to Salesman k' (such a list of clients is private in DO), Salesman k will have trouble convincing k' to accept more than $n' = n$ city in an interaction round. In fact, a consequence of the triangular inequality is that accepting n (respectively, $n+1$, $n+2$, etc.) cities while giving $n-1$ (respectively, n, $n+1$, etc.) increases more $u_{k'}^{\mathrm{mod}}$ than accepting the same number of cities as the number given (except when a city exactly lies on the optimal route – which is known by k' but ignored by k). As a result, even when k proposes to give n close cities in exchange for n' cities, k' would not be rational to accept $n' < n$.

The traditional MTSP is the problem faced, for example, by a home health care service in which all nurses are employees and have hence no individual utility function to optimise. In contrast, our modified MTSP corresponds to the problem faced by a private nurse association in which the nurses have an initial endowment of patients (the patients either make an appointment to their nurse or are provided by a physician) and the association is a place in which the nurses may exchange patients when this is mutually beneficial. In this modified problem, every private nurse has an individual utility function which they optimise.

2.3 **Hypothesis** Hyp. 4 about the computation time

In this article, we also make Assumption Hyp. 4 which states that we only take account of the computation time of the MILP solver, namely CPLEX. We choose to make this assumption as a solution to the problem of comparing various organisations implemented in various environments and languages. In fact, Hyp. 4 forbids the comparison of the duration of the operation of a mechanism which mostly uses CPLEX with the duration of another which simulates its interactions in AnyLogic[2] which runs in Java. Because of this hypothesis, we ignore the duration of anything not computed by CPLEX, such as:

- *Messages travel instantaneously*: DO implies interactions which are simulated by AnyLogic, not CPLEX. Hence, we ignore the travelling time of the messages exchanged.
- *Some techniques are not possible*: We cannot use multiagent techniques, such as BDI (Belief-Desire-Intention) architecture (Caballero et al., 2011) or reinforcement learning (Gunady et al., 2014), which may decrease the performance of our DO organisations. Similarly, Organisation Cluster cannot use the k-means algorithm, but a MILP formulation usable by CPLEX. Likewise, the salesmen locally solve a TSP in several of our organisations with CPLEX, but they cannot use Concorde[3] while it is often seen as the most efficient.

3 Allocation mechanisms

We now describe our six mechanisms/organisations to solve the MTSP constrained by 1-1 exchanges. Figure 1 shows an overview in which we can see, for example, that OptDecentr is more centralised than Cluster and Auction, and Cluster has a more coercive CA than Auction. This figure also points out that OptDecentr is the most centralised in the sense that the CA solves both the allocation and all routing problems while the other organisations let the salesmen locally solve a TSP on their allocated cities. Some organisations operate in a single round while others need more interactions. Finally, this figure recalls that FullCentr solves a problem different than the other mechanisms. Each subsequent subsection details a mechanism.

3.1 **Pure DO:** NoRealloc, P2P and CNP

We first present NoRealloc since its MILP formulation of TSP is both used in P2P, Cluster and Auction, and the base of the MILP formulation of the MTSP constrained by 1-1 exchanges in OptDecentr.

[2]The AnyLogic model and the outcomes of the experiments will be published on `www.github.com` after acceptation of this article for publication.

[3]`http://www.math.uwaterloo.ca/tsp/concorde/index.html`

3.1.1 NoRealloc

As said above, NoRealloc ignores the allocation problem and lets every salesman find the shortest route leaving the depot, visiting the N cities[4] allocated to him in the initial endowment and returning to the depot. There is a single round in which each of the m salesmen solves the TSP in Equations 9-14. This formulation uses the 2-index decision variable x_{ij} which equals one only if the considered salesman goes from City i to City j.

$$\begin{aligned}
\min \quad & \textstyle\sum_{i=0}^{N-1} \sum_{j=0, j\neq i}^{N-1} d_{ij} x_{ij} && && (9)\\
s.t. \quad & \textstyle\sum_{j=0, j\neq i}^{N-1} x_{ij} = 1 && 0 \leq i < N && (10)\\
& \textstyle\sum_{i=0, i\neq j}^{N-1} x_{ij} = 1 && 0 \leq j < N && (11)\\
& p_i - p_j + N.x_{ij} \leq N-1 && 1 \leq i \neq j < N && (12)\\
& p_i \in \Re^+ && 1 \leq i < N && (13)\\
& x_{ij} \in \{0,1\} && (i,j) \in \{0,1,\ldots N-1\}^2 && (14)
\end{aligned}$$

With this notation, Equation 9 is the same as Equation 1 except that it minimises the route length of a single salesman and n is thus replaced by N, Equation 10 (respectively, 11) is similar to Equations 2 and 3 (respectively, 4 and 5) except that it does not need to distinguish the case of the depot, and Equation 12 is the same constraint of sub-route elimination as Equation 6.

3.1.2 P2P

Mechanism P2P has several interaction rounds in which instances of TSP or a derivative of TSP are solved. The bottom of Figure 2 shows that P2P consists of two state charts which may run concurrently, namely, `P2P_host` and `P2P_guest` which replies to the former. Therefore, every salesman may take part in two interactions simultaneously, the first as guest and the second as host. Please remember that every salesman operates his own copy of the state charts in Figure 2. The names of the states and transitions in both state charts all start with `P2Pn_action` where `n` indicates their order of activation in a round and `action` summarises the action performed. P2P uses the variables at the top of Figure 2. Each of these variables is a pointer to a city or salesman, except $propCities[k][i]$ that records previous interactions as a matrix of Booleans which are true only when the considered salesman has already proposed City i to another Salesman k in order to prevent infinite loops. $allocatedCities$ (list of cities currently allocated to the considered salesman), $ownedCities$ (his initial endowment) and $route$ (similar to $allocatedCities$ but with the cities ordered according to the shortest route found by the mechanism in use, thus P2P in this subsection) also are variables but AnyLogic shows them with a different icon because they are collections of objects.

[4] N may be different between salesmen in the two variants of TSP shown in this article. We do not use N_k because N is a local variable for every salesman-agent k.

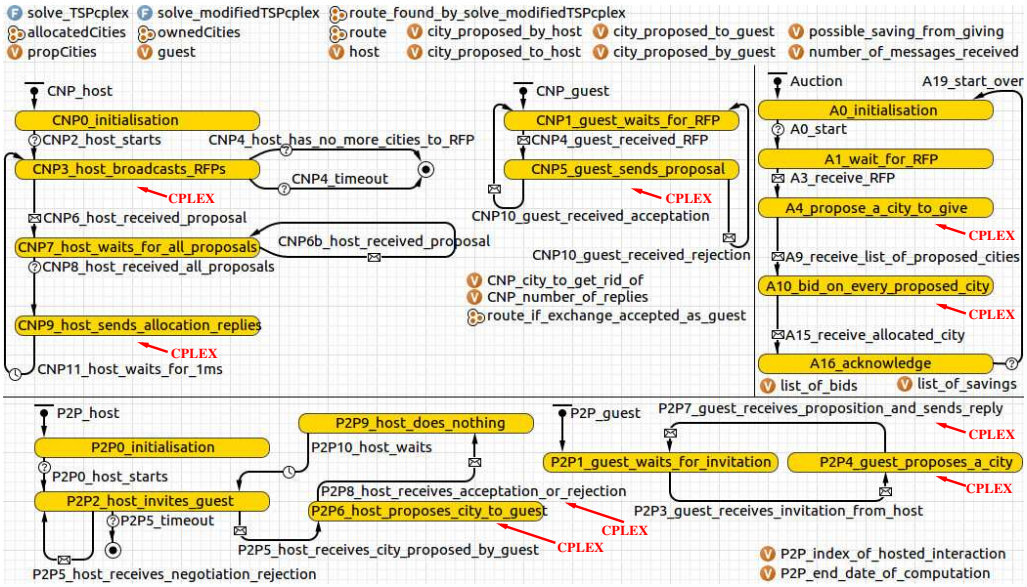


Figure 2: Implementation of the mechanisms in a salesman. ("CPLEX" arrows point to transitions and states whose duration are taken into account.)

We now detail the operation of P2P. State `P2P0_initialisation` mainly sets all entries in $propCities[k][i]$ to $false$. AnyLogic always executes the first state in all state charts, and transition `P2P0_host_starts` only fires when mechanism P2P is selected. The first salesman whose `P2P0_host_starts` fires first becomes the host in this round. (This state chart could generate more than one host at the same time, but we prevent this by setting AnyLogic to use only one thread, as detailed in Section 4.) Next, `P2P2_host_invites_guest` makes this host select a guest (*i.e.*, the salesman with the lowest number of $false$ in $propCities[k][i]$) and send him an invitation. The guest was waiting in state `P2P1_guest_waits_for_invitation` and this message fires his transition `P2P3_guest_receives_invitation_from_host`.

Then, `P2P4_guest_proposes_a_city` is the first call to CPLEX in this round of interaction. The guest solves the modified TSP in Equations 15-23 in order to propose a city to the host. The goal of this model is to find the city which should be removed from $allocatedCities$ such that the reduction of route length is maximised.[5] Equations 15-23 modify the formulation of TSP in Equations 9-14 by adding binary decision variable $kept$ such that $kept_i = 1$ for the $N - 1$ cities to be kept and $kept_i = 0$

[5]The problem in Equations 15-23 finds the city which causes the largest increase in the route length. Instead of this single round, we may solve this problem in $N - 1$ rounds in which the TSP in Equations 9-14 is solved with $N - 1$ cities (the i^{th} round without the i^{th} city), then the shortest of the $N - 1$ obtained route lengths indicates the city to be proposed. We have tested this method and seen that it takes more time than solving Equations 15-23.

for the city to be proposed to the host.[6]

$$\min \quad \sum_{i=0}^{N-1} \sum_{j=0, j\neq i}^{N-1} c_{ij} x_{ij} \tag{15}$$

$$s.t. \quad \sum_{j=0, j\neq i}^{N-1} x_{ij} = kept_i \qquad 0 \leq i < N \tag{16}$$

$$\sum_{i=0, i\neq j}^{N-1} x_{ij} = kept_j \qquad 0 \leq j < N \tag{17}$$

$$\sum_{i=1}^{N-1} kept_i = N - 2 \qquad 0 \leq i < N \tag{18}$$

$$p_i - p_j + N.x_{ij} \leq N - 1 \qquad 1 \leq i \neq j < N \tag{19}$$

$$kept_i = 1 \qquad i | \texttt{propCities}[host][i] = true \tag{20}$$

$$kept_i \in \{0, 1\} \qquad i \in \{1, \ldots N-1\} \tag{21}$$

$$p_i \in \Re^+ \qquad 1 \leq i < N \tag{22}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j) \in \{0, 1, \ldots N-1\}^2 \tag{23}$$

Consequently, Equations 15 and 19 are the same as Equations 9 and 12. Equations 16 and 17 are similar to Equations 10 and 11 except for the city i to be proposed which has $kept_i = 0$. Equation 18 checks that exactly one city will be proposed to the host. Equation 20 ensures that a city previously proposed and returned by the guest will not be proposed again.[7] If $propCities$ indicates that all cities have already been proposed, then the considered guest proposes `null`, which fires transition `P2P5_host_receives_negotiation_rejection`.

Otherwise, the city proposed by the guest fires transition `P2P5_host_receives_city_proposed_by_guest` and state `P2P6_host_proposes_city_to_guest` finds the city to be proposed by the host to the guest. This is the second call to CPLEX in this round of interaction. Again, CPLEX solves the above modified TSP in Equations 15-23 (with a small modification: `host` needs to be replaced by `guest` in Equation 20). The guest may not find a city reducing his route length and, hence, send a `null` reply to the guest. Otherwise, he memorises in $propCities$ not to keep proposing the city he has just proposed.

`P2P7_guest_receives_proposition_and_sends_reply` receives this proposed city. If the city proposed by the host is `null`, then the guest sends a message to confirm the failure of this round of negotiation. Otherwise, he uses a CPLEX to solve the traditional TSP in Equations 9-14 with all his cities, minus the one previously proposed to the host, plus the one which has just been proposed by the host. If the proposed exchange reduces his route length, then the guest sends an acceptation message and updates his route length by taking this exchange into account, otherwise he sends a `null`.

Finally, transition `P2P8_host_receives_acceptation_or_rejection` receives this reply. If the guest accepts the exchange, the host modifies his $allocatedCities$ then updates his $route$ by solving Equations 9-14 and

[6] When this problem is solved by a salesman in Mechanism Auction, the end of this sentence reads: "$\ldots kept_i = 0$ for the city to be proposed to the auctioneer".

[7] When Equations 15-23 are solved in Mechanism Auction, the previous sentence reads: "Equation 20 ensures that a city previously proposed and returned by the auctioneer will not be proposed again".

setting $propCities[city][guest] = false$ for all his $city$ in order to propose any city to this guest again.

P2P9_host_does_nothing and P2P10_host_waits make the host wait for one millisecond. This tiny pause is requested by AnyLogic in order to allow other salesmen to become host. Otherwise, Salesman 0 would keep the control in all subsequent rounds until he has proposed all his cities, next Salesman 1 would become host as long as he has not proposed all his cities, then Salesman 2, etc.

Finally, note that P2P does not loop forever because State P2P2_host_invites_guest does not send an invitation when $propCities[k][i] = true$ for all Salesmen k and all Cities i, which eventually causes every salesman except one to stop acting as host.

3.1.3 CNP

Like (Frey et al., 2003), our Mechanism CNP is inspired by the Contract Net Protocol. CNP is described by state charts CNP_host and CNP_guest in Figure 2. Similarly to P2P, the states and transitions have a name starting by CNPn_action where n is the order of activation of the considered element and action describes it. CNP runs by rounds in which a salesman (host) plays the role of an auctioneer who broadcasts a city to give and the other salesmen (guests) reply by proposing a city to be exchanged. Conversely to P2P, CNP has several guests.

The detail of this mechanism is as follows. CNP0_initialisation sets all the entries in $propCities[k][i]$ to $false$; This is performed by all salesmen because the first state in all state charts is always executed. All salesmen wait in state CNP1_guest_waits_for_RFP. Transition CNP2_host_starts in all salesmen may fire because its condition only checks that Mechanism CNP is selected; This condition fires first in one of the salesmen who becomes the host in this round. (Like P2P, CNP could have several hosts and we prevent this by allowing only one thread in AnyLogic, as detailed in Section 4.) The host does the first use of CPLEX in this round to select a city to give by solving the modified TSP in Equations 15-23; This city is sent in a Request For Proposals to all the guests in CNP3_host_broadcasts_RFPs. Every guest also uses CPLEX to solve the problem in Equations 15-23 in order to make a proposal in CNP5_guest_sends_proposal. When transitions CNP6_host_received_proposal and CNP6b_host_received_proposal have received all these proposals, the host selects the winner by calling CPLEX to test each proposed city in CNP9_host_sends_allocation_replies. More precisely, for each city submitted by the guests, the host solves the traditional TSP in Equations 9-14 with his allocated cities minus the city broadcast in the Request For Proposals (RFP) plus the city submitted by the considered guest. The guest sends an acceptation message to the guest who proposed the city reducing the most his route length. In this case, the guest updates his variables allocatedCities and route (route points to the same cities as allocatedCities but in the order minimising the route length). If no city causes such a reduction, then no acceptation is sent. Finally, the guest sends a rejection message to

the other guests. After the acceptation (respectively, rejection) message has been received by `CNP10_guest_received_acceptation` (respectively, `CNP10_guest_received_rejection`), another round may start.

After pure DO, we now turn our attention to pure CO.

3.2 CO with constraints of DO: OptDecentr

As previously said, OptDecentr is a CO organisation that mimics DO. In other words, CA uses CPLEX in order to find the optimal solution of our MTSP constrained by 1-1 exchanges. Please first notice that this constraint of 1-1 exchanges of cities makes all salesmen always keep the same number of cities as in their initial endowment. As a result, the decision variable must identify the salesmen in order to ensure that their number of allocated cities equals their number of cities owned in this endowment. Consequently, OptDecentr uses a MILP model with the 3-index decision variable x_{ijk} which equals one only if Salesmen k goes from City i to City j:

$$\min \quad U^{\text{mod}} = \sum_{k=1}^{m} u_k^{\text{mod}} \tag{24}$$

$$s.t. \quad u_k^{\text{mod}} = \sum_{i=0}^{n-1} \sum_{j=0, j\neq i}^{n-1} d_{ij} x_{ijk} \qquad 1 \leq k \leq m \tag{25}$$

$$\sum_{j=1}^{n-1} x_{0jk} = 1 \qquad 1 \leq k \leq m \tag{26}$$

$$\sum_{j=0, j\neq i}^{n-1} \sum_{k=1}^{m} x_{ijk} = 1 \qquad 1 \leq i < n \tag{27}$$

$$\sum_{i=0, i\neq j}^{n-1} \sum_{k=1}^{m} x_{ijk} = 1 \qquad 1 \leq j < n \tag{28}$$

$$\sum_{i=1}^{n-1} x_{i0k} = 1 \qquad 1 \leq k \leq m \tag{29}$$

$$p_i - p_j + n.x_{ijk} \leq n - 1 \qquad 1 \leq i, j < n, 1 \leq k \leq m \tag{30}$$

$$\sum_{i=0}^{n-1} \sum_{j=0, j\neq i}^{n-1} x_{ijk} = \sum_{j=0}^{n-1} o_{jk} \qquad 1 \leq k \leq m \tag{31}$$

$$(\sum_{l=0, l\neq j}^{n-1} x_{jlk}) - x_{ijk} \geq 0 \qquad 0 \leq i, j < n, 1 \leq k \leq m \tag{32}$$

$$p_i \in \Re^+ \qquad 0 \leq i < n \tag{33}$$

$$x_{ijk} \in \{0, 1\} \qquad 0 \leq i, j < n, 1 \leq k \leq m \tag{34}$$

In this model, Equations 24 and 25 minimise the total distance travelled by the community of salesmen. Equation 26 (respectively, 29) checks that all salesmen leave (respectively, enter) the depot. Equation 27 (respectively, 28) checks that all cities are left (respectively, entered) exactly once. Equation 30 is the same constraint of sub-route elimination as Equations 6, 12 and 19. Equation 32 ensures that the salesmen entering and leaving a city are the same. Equation 31 checks that the number of cities allocated to a salesman equals the number of cities he owns in his initial endowment. (The right-hand side in this equation is a constant as Parameter o_{jk} represents the initial endowment ("ownership") of cities, modelled by $o_{jk} = 1$ if City j is "owned" by Salesman k at the beginning of the experiment, and $o_{jk} = 0$ otherwise.)

3.3 CO/DO hybrids: Cluster and Auction

We now introduce our two hybrid organisations, $viz.$, Cluster and Auction. They are not pure DO since CA takes part in the allocation and not pure CO because CA lets the salesmen locally solve the TSP in Equations 9-14. Cluster is more coercive because the salesmen are supposed to let CA know about all their cities in order to solve the allocation problem, while Auction lets them free to never propose some of their cities if they want for some reason ($e.g.$, a city has an important client they want to keep or not disclose). Conversely to P2P and CNP which only perform bilateral exchanges, Cluster and Auction may involve more than two salesmen per exchange during a round, that is, Salesman s_1 may give a city to Salesman s_2, s_2 give to $s_3, \ldots$, and s_q give to s_1 for any $q \leq m$.

3.3.1 Cluster

Because of Hypothesis Hyp. 4 about the use of CPLEX to make all decisions, Mechanism Cluster cannot use whichever solving methods from the clustering literature but only those based on a MILP formulation. We use the model proposed in (Rao, 1971, Sec. 5):

$$\begin{array}{lcll} \min & D & & (35) \\ s.t. & D \geq d_{ij}(x_{ik} + x_{jk} - 1) & 1 \leq i < j < n, 0 \leq k < m & (36) \\ & \sum_{k=1}^{m} x_{ik} = 1 & 1 \leq i < n & (37) \\ & 1 + \sum_{j=1}^{n-1} x_{jk} = \sum_{j=1}^{n-1} o_{jk} & 0 \leq k < m & (38) \\ & x_{ik} \in \{0,1\}, D > 0 & 1 \leq i < n, 0 \leq k < m & (39) \end{array}$$

In this model, decision variable x_{ik} is a binary equal to one only when City i is allocated to Salesman/Cluster k. Equations 35 and 36 are the objective function which minimises the diameter of the cluster which has the largest diameter. More precisely, Equation 36 includes the linearisation of $D_k \geq d_{ij}x_{ik}x_{jk}$ which ensures that the diameter of Cluster k is at least the maximum distance between any two Cities i and j allocated to this cluster; Please notice that such a formulation defines circular clusters which may hence overlap. Like in any other allocation problem, Equation 37 checks that every city is affected to exactly one cluster/salesman. In addition to the original model by Rao (1971), we add Equation 38 in order to ensure that the size of the clusters(/salesmen) correspond to the number of cities provided by every (cluster/)salesman, that is, if Salesman k owns $\sum_j o_{jk}$ cities (again, o_{jk} is a constant which equals one only if Salesman k "owns" City j in his initial endowment), then a cluster with $\sum_j o_{jk}$ cities must exist in order to be allocated to k.

As previously said, Mechanism Cluster operates in a single round: CA first solves the problem in Equations 35-39 to allocate N cities to every salesman[8], then these salesmen locally solve the traditional TSP in Equations 9-14.

[8]Like in Footnote 4, N may not be that same for all salesmen since it is a local variable.

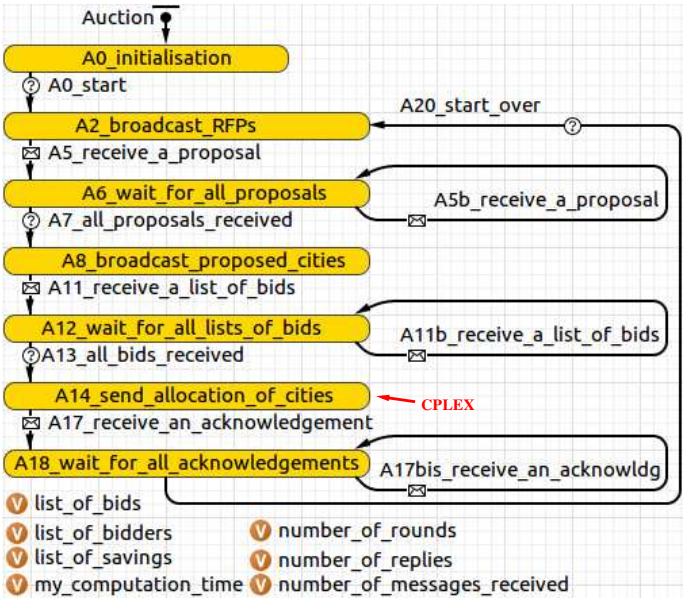


Figure 3: State chart of Auction in the Central Authority (CA).

3.3.2 Auction

Auction is an organisation in which CA is an auctioneer who is thus less coercive than in Cluster. Conversely to Cluster, Auction operates in several rounds. In each round, every salesman gives a city A to CA who either gives it back if no other salesman wants it, or gives a city B proposed by another salesman if exchanging A and B reduce the length travelled by both salesmen. Figure 3 and the top right corner of Figure 2 detail the states and transitions in this organisation. As above, their name has the form `An_action` where `n` helps the reader understand the order of their activation and `action` summarises their goal.

At the beginning of a round, all salesmen wait in `A1_wait_for_RFP` until CA sends them a Request For Proposals in `A2_broadcast_RFPs`. When a salesman receives this message, he uses CPLEX to look for a city to give in `A4_propose_a_city_to_give` by solving the modified TSP in Equations 15-23. When all salesmen have replied by sending either a city or `null` to CA, CA broadcasts this list of replies to all salesmen in `A8_broadcast_proposed_cities`. For each city in this list, every salesman uses CPLEX to compute a bid for it in `A10_bid_on_every_proposed_city`. A bid for a city is the additional distance travelled to visit it, or, more technically, as the differences between:

- the shortest length of the route visiting their remaining $N-1$ cities, that is, their allocated cities except the one proposed to the auctioneer in `A4_propose_a_city_to_give`. This is obtained by each salesman by solving the TSP in Equations 9-14 once.
- the shortest length of the route visiting their remaining $N-1$ cities

plus the city proposed by one of the other salesmen. This is obtained by each salesman by solving the TSP in Equations 9-14 for each city in the list of bids.

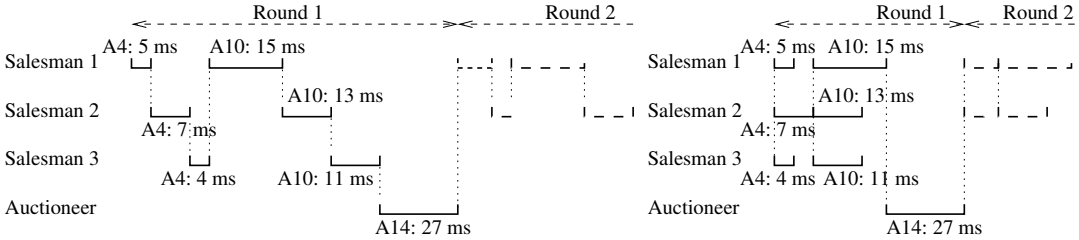
When all salesmen have returned their list of bids, CA uses CPLEX to solve an allocation problem in `A14_send_allocation_of_cities`. In order to describe this problem, let us call $\texttt{savings}[k][i]$ the bid of Salesman k for City i and binary decision variable x_{ki} equals 1 only if City i is allocated to Salesman k. For simplicity, we write m the number of salesmen who have not left the auction before the current round. The allocation model solved by CA is described by Equations 40-44. The objective in Equation 40 allocates the cities such that the total route length of all salesmen is minimum. The constraint in Equation 43 ensures that, in every auction round, every salesman does not increase his individual route length.

$$\begin{aligned}
\min \quad & \sum_{k=1}^{m} \sum_{i=0}^{m} \texttt{savings}[k][i].x_{ki} && && (40)\\
s.t. \quad & \sum_{i=0}^{m} x_{ki} = 1 && 0 \leq k < m && (41)\\
& \sum_{k=1}^{m} x_{ki} \leq 1 && 0 \leq i < m && (42)\\
& \sum_{i=0}^{m} \texttt{savings}[k][i].x_{ki} \leq \texttt{savings}[k][k] && 0 \leq k < m && (43)\\
& x_{ki} \in \{0,1\} && 0 \leq i,j \leq m && (44)
\end{aligned}$$

4 Real time spans deduced from sequential simulations

After the description of the compared mechanisms, we now present how the computation time span of every organisation is deduced from sequential experiments, that is, experiments running on a single thread of AnyLogic. In fact, our experimentation involves up to 10 agents (9 salesmen + 1 CA) while each of our computers has a CPU with 4 cores only. Thus, configuring AnyLogic to simulate this parallelism would require studying how AnyLogic schedules up to 4 agents in parallel, then deduce how 10 agents would have behaved in reality. Instead, we prefer to make AnyLogic carry out all computations sequentially, then infer how (pure and mixed) DO would occur concurrently in real life. The time span of the CPLEX computation (duration between the beginning of the first computation and the end of the last one) in an experiment is deduced as follows.

- OptDecentr and FullCentr: CO uses no parallelism and the computation time span is thus equal to the computation time recorded in our sequential experiments. More technically, this duration is the difference between the two `System.currentTimeMillis()` after and before `cplex.solve()`. This is also the Java code to measure the duration of all CPLEX calls in all our organisations below.
- Auction: Figure 4a illustrates how AnyLogic sequentially runs the operation of two salesmen, and Figure 4b how reality would look like with parallelism:
  - *AnyLogic*: Figure 4a shows that Salesman 1 calls CPLEX for 5 ms in State `A4_propose_a_city_to_give` (summarised as "A4:



a) Simulation: Sequential experimentations find that Round 1 lasts (5+7+4)+(15+13+11)+27=82ms.

b) Reality: Round 1 with parallel decisions would last max(5;7;4)+max(15;13;11)+27=49ms.

Figure 4: Difference between the computation time (a) in our experiments and (b) in real (b) of one round of Auction. ("Ax:" is the beginning of the the name in the state charts in Figures 2 and 3).

5ms" in Figures 4a and 4b), and Salesman 2 and 3 spend 7 ms and 4 ms in this state. Next CA (auctioneer) receives the result of these computations and broadcasts it to all salesmen (not shown in Figures 4 because performed without CPLEX). Later, Salesman 1 computes for 15 ms, Salesman 2 for 13 ms and Salesman 3 for 11 ms in `A10_bid_on_every_proposed_city`. Finally, the auctioneer computes for 27 ms in `A14_send_allocation_of_cities`. The duration observed in AnyLogic is the sum of these durations, as shown in Figure 4a.

– *Reality*: In real life, all states with the same name can be computed in parallel, as shown in Figure 4b. It follows that the duration of the CPLEX computations in a state is the maximum of the durations of all salesmen in this state, $i.e.$, this round would take $\max(5;7;4)+\max(15;13;11)+\max(27)=49$ms.

- Cluster: The method for Cluster is the same as for Auction except that (i) there is a single round, and (ii) every salesman solves the TSP only once in this round. Shortly, the real-life duration is (i) the time spent by CA solving the clustering problem in Equations 35-39 plus (ii) the maximum of the time spent by the salesmen solving the TSP in Equations 9-14.
- NoRealloc: The method for NoRealloc is the same as Cluster without CA, that is, the real-life duration is the maximum of the time spent by the salesmen solving the TSP in Equations 9-14.
- P2P and CNP: The inference of the duration of interactions in pure DO is the most complicated because several interactions of various durations may take place concurrently. In contrast, the CA in Auction ensures that a new round starts only after the end of the previous one. On the contrary, no CA synchronises interactions in P2P and CNP because the state charts in Figure 2 allow every salesman to be host and guest at the same time in two concurrent interactions. We assume that interactions do not overlap, that is, an interaction is never stopped between its start and end. This complies with our observation of the operation of AnyLogic when only one thread is

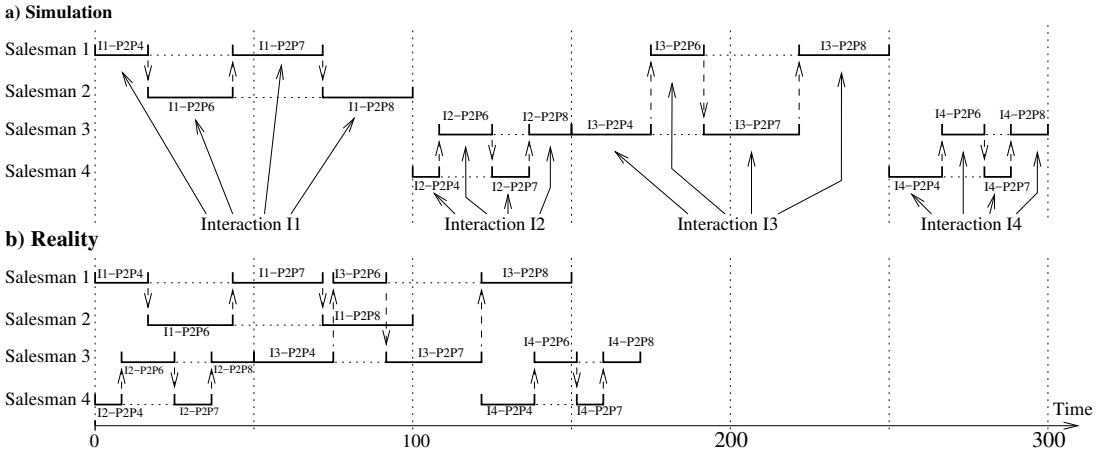


Figure 5: Difference between the computation time (a) in our experiments and (b) in real life in four rounds (called I1, I2, I3 and I4) of P2P interactions. ("Ix" refers to Interaction Ix and "-P2Py" to the names in the state charts in Figure 2).

used. In addition, using more threads would not change the total duration of interactions and would just make it more complicated to observe what we now explain. This explanation is provided for P2P because it is the most complex as several bilateral interactions may occur concurrently; CNP is slightly simpler because such overlaps are made impossible by the fact that an interaction always involves all salesmen.

In every round of a P2P interaction, we make AnyLogic record the (i) identities of the host and guest in this round, (ii) the starting time in AnyLogic of this round, and (iii) its duration ($i.e.$, sum of (1) the CPLEX computation time of the guest in `P2P4_guest_proposes_a_city`, (2) the time of the host in `P2P6_host_proposes_city_to_guest`, (3) the time of the guest in `P2P7_guest_receives_proposition_and_sends_reply` and (4) the time of host in `P2P8_host_receives_acceptation_or_rejection` – please notice that this is a summation because everything also happens sequentially in real life). Figure 5 illustrates how this information is used after the completion of the mechanism with an example of four interactions named I1, I2, I3 and I4:

– *AnyLogic*: Salesman 2 is the host of Interaction I1 (Figures 5a and 5b only show CPLEX computations, thus `P2P2_host_invites_guest` is not shown) and the first CPLEX computation is performed by his guest Salesman 2 in State `P2P4_guest_proposes_a_city`, which is represented by I1-P2P4 in Figure 5a. AnyLogic carries out the computations one after the other: I1-P2P4, then I1-P2P6, I1-P2P7 and I1-P2P8 for Interaction 1, next Interaction 2 with I2-P2P4, I2-P2P6, I2-P2P7 and I2-

P2P8, then Interaction 3, etc.

– *Reality*: Figure 5b shows that Interactions I1 and I2 will start at the same time in reality since they involve two different pairs of salesmen. Hence, the CPLEX calls in {I1-P2P4, I1-P2P6, I1-P2P7, I1-P2P8} would be in parallel with those in {I2-P2P4, I2-P2P6, I2-P2P7, I2-P2P8}. After that, I3 would start as soon as its host Salesman 1 is available, $i.e.$, just after I1-P2P7, then this host would wait for the reply of Salesman 2 at the end of I3-P2P3 because this guest would be taking part to I2. Similarly, I4 starts just after its host Salesman 3 finished I3-P2P7, which is not shown on Salesman 3 but with the effect on Salesman 4 who computes I4-P2P4.

Finally, both Figures 5a and 5b are incomplete because they show interactions without the initialisation of the salesmen who first solve the TSP in Equations 9-14 before $t = 0$, and the salesmen in Figures 5b would be ready for their first interaction at different times.

More technically, the computation time in P2P is calculated by updating a vector $clock_k$ after each computation of Salesman k. $clock_k$ represents the total computation time of Salesman k up to the considered interaction. We also call T_s the computation time of CPLEX in state s:

– *Duration of initialisation*: Every salesman k solves the TSP in Equations 9-14, which takes a duration $T_{\texttt{P2P0}_k}$. Since these computations are parallel, the duration of this initialisation is $\max_k(T_{\texttt{P2P0}_k})$. Hence, at the end of this initialisation, $clock_k = \max_k(T_{\texttt{P2P0}_k})$ for all Salesmen k.

– *For every round of bilateral interactions*:

  * An interaction starts as soon as its host and guest are both ready, thus an interaction starts at $\max(clock_{\textbf{host}}, clock_{\textbf{guest}})$. Hence, $clock_{\textbf{host}} = clock_{\textbf{guest}} = \max(clock_{\textbf{host}}, clock_{\textbf{guest}})$.
  * The end dates of the current round is $clock_{\textbf{guest}} = clock_{\textbf{host}} + T_{\texttt{P2P4\_guest\_proposes\_a\_city}} + T_{\texttt{P2P6\_host\_proposes\_city\_to\_guest}} + T_{\texttt{P2P7\_guest\_receives\_proposition\_and\_sends\_reply}}$ and $clock_{\textbf{host}} = clock_{\textbf{guest}} + T_{\texttt{P2P8\_host\_receives\_acceptation\_or\_rejection}}$.

– *Total computation time*: The searched duration of Mechanism P2P is $\max_k(clock_k)$ calculated when all interaction rounds are completed.

5 Numerical experimentation

Instead of performing Monte Carlo simulations to assess the mechanisms, we allow the reader to replicate our results by generating 130 instances by circular permutations on problem "CH130" in TSPLIB.[9] In order to describe these circular permutations, let us call (x_i, y_i) the coordinates of the i^{th} city in our simulation and (X_i, Y_i) the coordinates of the i^{th} city in TSPLIB. For a given number of cities n, our Instance zero uses the first

[9] `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ch130.tsp.gz`

n instances in TSPLIB such that City i has $x_i = X_i$ and $y_i = Y_i$ (cities $i \geq n$ in TSPLIB are ignored), $i.e.$, Salesman 1 is at $(334.5\ldots, 161.7\ldots)$, 2 at $(397.6\ldots, 262.8\ldots)$, 3 at $(503.8\ldots, 172.8\ldots)$, etc. Next, Instance ΔY uses $x_i = X_i$ and $y_i = Y_i + \Delta Y$, $i.e.$, for Instance 1, Salesman 1 is at $(334.5\ldots, 262.8\ldots)$, 2 at $(397.6\ldots, 172.8\ldots)$, 3 at $(503.8\ldots, 384.6\ldots)$, etc. (We noticed that Instance $\Delta Y = 122$ is often very long to solve for several mechanisms.)

This generation of instances allows us to present results in which the mechanisms work on the same instances. For example, the ratios in Figures 7, 8, 9, 10 are obtained by (i) setting n and m, (ii) comparing all mechanisms ($i.e.$, computing the ratios of the total route length of two mechanisms) on Instance $\Delta = 0$, (ii') repeating ii for Δ varying between 1 and 129, and (iii) writting in these figures the fifth and ninth deciles of these 130 ratios.

Our numerical experimentation have been performed on the 17 Personal Computers of a student laboratory in the department of Industrial Engineering at INSA-Lyon, Lyon, France. These computers are all identical and run Windows 7 professionnel SP1 64 bits on Intel Core i5-3470 CPU@3.20GHz with 8.00 Gb RAM. The softwares used were IBM ILOG CPLEX 12.6.3.0 and AnyLogic 7.3.2.

5.1 Results with a time limit of 24 hours

We consider two metrics to assess our mechanisms, $viz.$, total route length (also referred to as quality of the solution) and computation time. In this article, all graphs in a same figure use the same ranges on the y-axis. For each mechanism, Figure 6 shows the box-plots of the computation time of the 130 instances for various number of cities n when there are $m = 9$ salesmen. The main point to notice is the quick increase of the duration of OptDecentr from $n = 20$, which corresponds $(n-1)/m \approx 2.1$ cities per salesman ("minus one" prevents counting the depot). In this figure, the number of salesmen m is increased until one of the 130 instances cannot reach its end within 24 hours. As can be seen, this limit is set by OptDecentr which cannot find the optimal solution of at least one of the 130 instances for $m = 9$ salesmen and $n = 23$ cities. We do not show the equivalent of Figure 6 with the quality of the solution because it would be difficult to see a difference between the mechanisms.

Instead, we show Figures 7 and 8 which show ratios of quality (compared to OptDecentr in Figure 7 and to FullCentr in Figure 8) for $m = 5$ (left) and $m = 9$ salesmen (right). More precisely, Figure 7 compares the total route length found by the mechanisms solving the MTSP constrained by 1-1 exchanges and this comparison is carried out against the optimal value found by OptDecentr. As said above, for given instance ΔY and values of m and n, the route length found by a mechanism is divided by the route length found by OptDecentr for each of the 130 instances; Both top graphs in 7 show the ninth decile of these 130 ratios and both bottom graphs show their median. Figure 8 is computed the same way, but the base of comparison is FullCentr instead of OptDecentr. In other words, Figure 7 compares mechanisms solving the same problem, while Figure 8

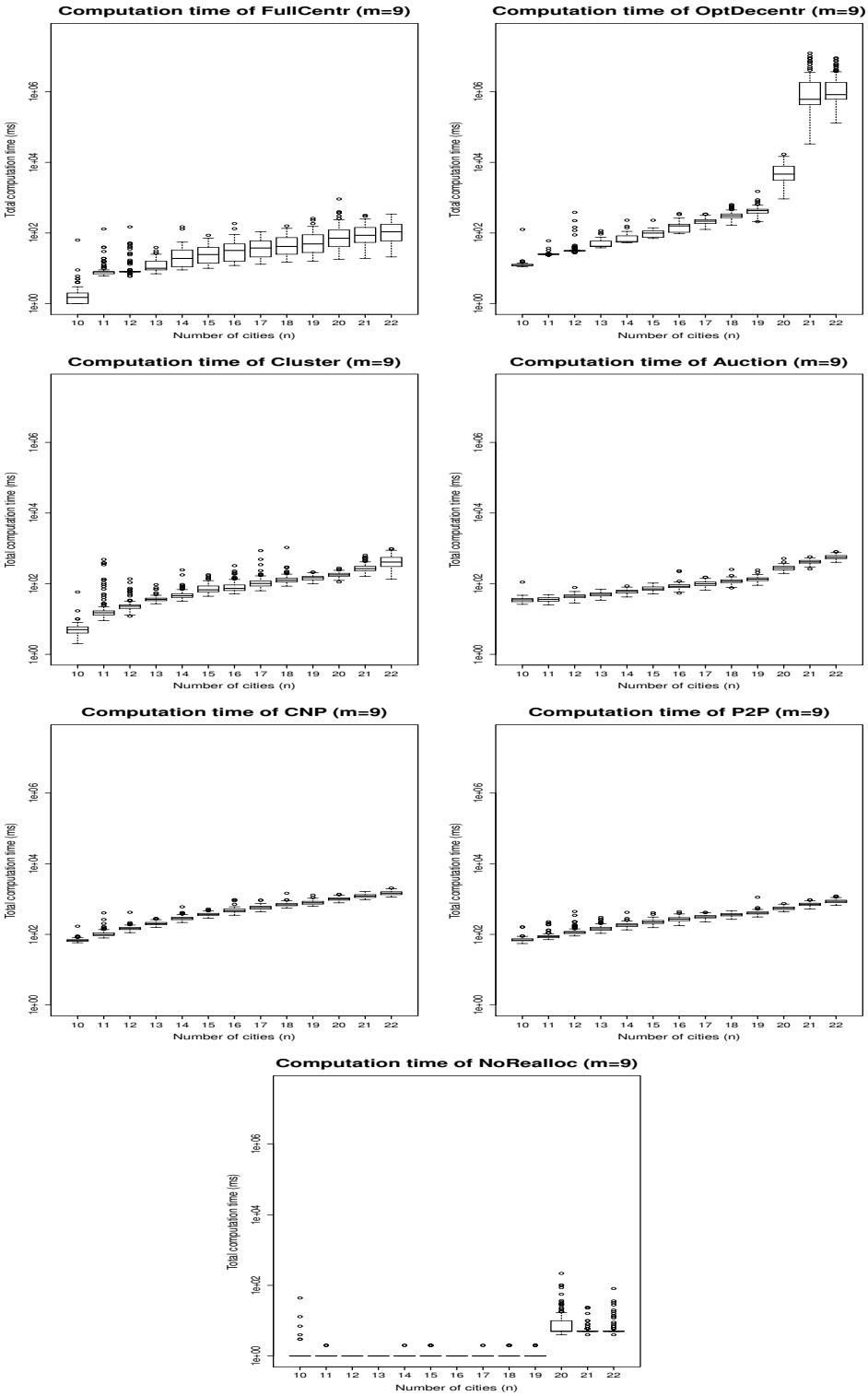


Figure 6: Box-plots of the computation time of the 130 instances for $m = 9$ salesmen and various number of cities n.

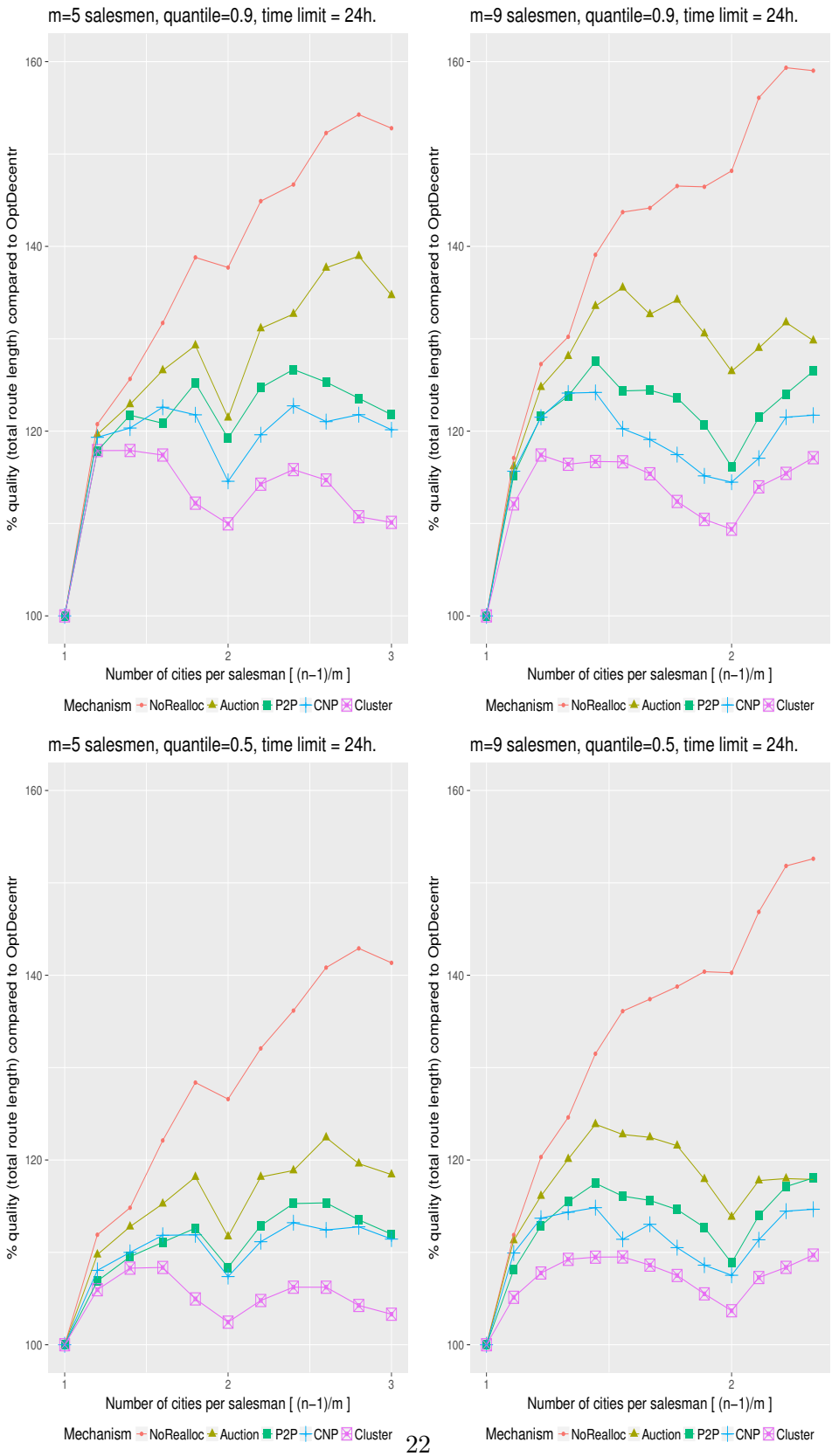


Figure 7: Ratios of quality compared to OptDecentr for $m = 5$ (left) and $m = 9$ (right) salesmen with no time limit.

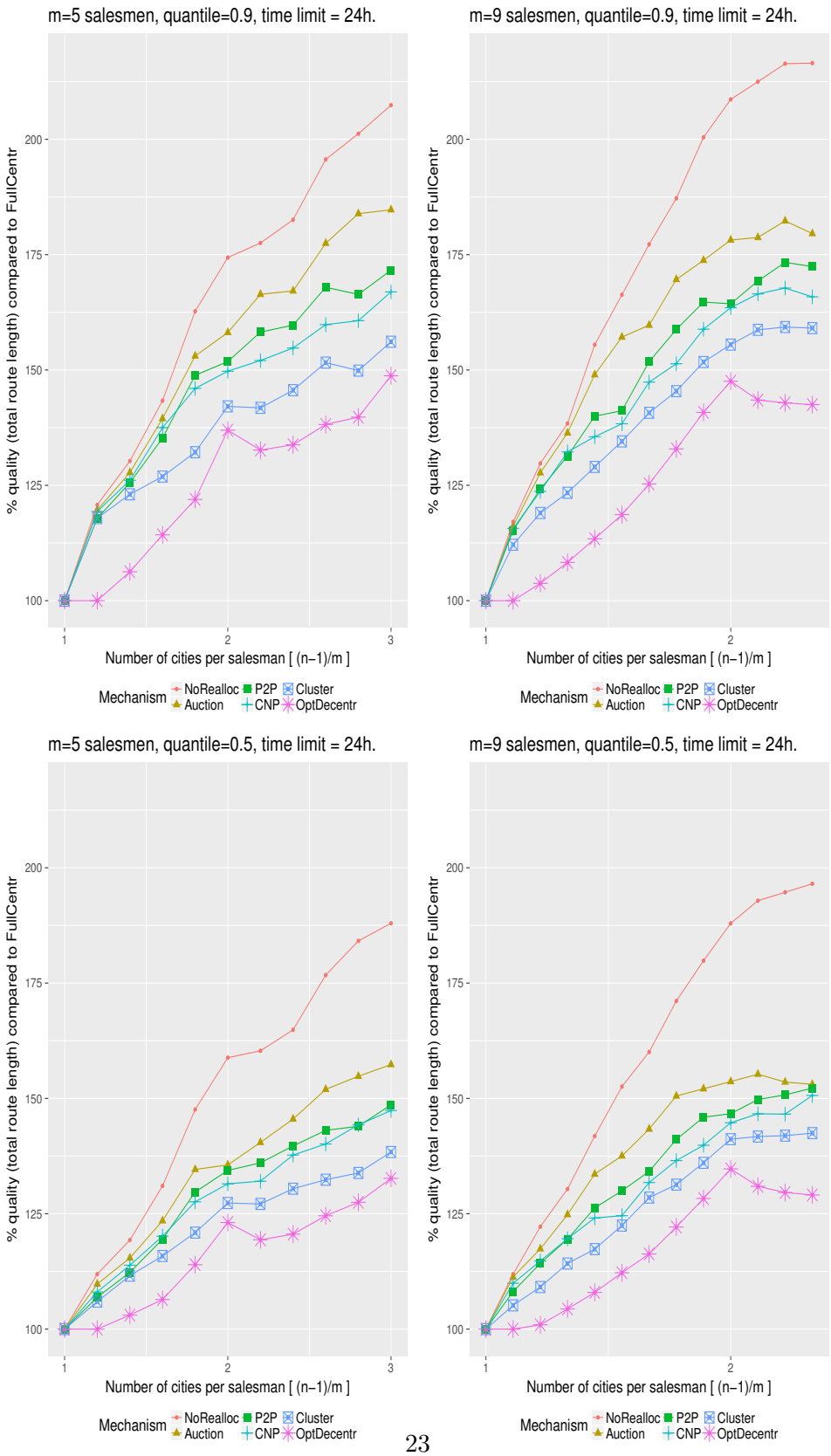


Figure 8: Ratios of quality compared to FullCentr for $m = 5$ (left) and $m = 9$ (right) salesmen with no time limit.

compares DO mechanisms to the CO mechanism.

We think that the most interesting points are the presence of plateaus. Some mechanisms seem to reach a plateau in Figure 7: Cluster seems to be about 5% worse, CNP about 11% worse, P2P about 12% and Auction about 20% worse than OptDecentr with the median for both $m = 5$ and $m = 9$, and the ninth decile also seems to stabilise on slightly higher figures. This indicates that these decentralised mechanisms do not explore the entire search space of the possible 1-1 exchanges since the CA in DO is sometimes able to find a better allocation which satisfies the selfishness of the salesmen. Unfortunately, there does not seem to be such plateaus in Figure 8, because either they do not exist or n has not been increased enough to reach them. We believe that the latter explanation is true. In order to check that we are right, the next subsection reduces the time limit and keeps increasing n even when the computation time reaches this limit.

5.2 Results with a time limit of 30 minutes

In order to obtain figures with much larger numbers of cities n, we limit the computation time to 30 minutes. (Hence, we only show the ratios of total route lengths since the equivalent of Figure 6 would only show that this time limit is respected.) In both Figures 7 and 8, the real-life duration of the operation of the mechanisms inferred from sequential simulations, as explained in Section 4, was computer after the end of the simulations (offline). In order to obtain more data, we have modified our AnyLogic models such that this computation is done throughout the simulation (online) by updating `get_Main().remainingComputationTime` which is shared by all salesmen, and added `cplex.setParam(IloCplex.Param.TimeLimit, get_Main().remainingComputationTime/1000)` in order to make CPLEX stop before or at the time limit. In Figures 9 and 10, this time limit is set to 30 minutes in parallel simulations. That is, DO mechanisms may run much longer sequential simulations in AnyLogic, but they cannot last more than 30 minutes when we infer what they would last in reality.

As a consequence, OptDecentr and FullCentr find the optimal solution of their respective version of MTSP in Figures 7 and 8 because we stopped the experimentation as soon as the time limit is reached. Hence, it was not possible to have a point below 100%. On the contrary, Figure 9 shows points below 100% for high values of n when the simulation stops before CPLEX has found the optimal solution of OptDecentr. (Figure 10 has no points below 100% because FullCentr is much quicker to optimiser.)

The most salient points to observe in Figures 9 and 10 are as follows:

- *The cost of selfishness is about 30% higher for $m = 5$ and 60% higher for $m = 9$ than the cost in the traditional MTSP*: In order to see this, we compare FullCentr (MTSP without our constraint modelling selfishness) and OptDecentr (benchmark showing the best results that Cluster, CNP, P2P and Auction could find):
  - *For the instances of small size* ($i.e.$, less than 6 cities per salesman for $m = 5$ in the lower left graph in Figure 10, and less

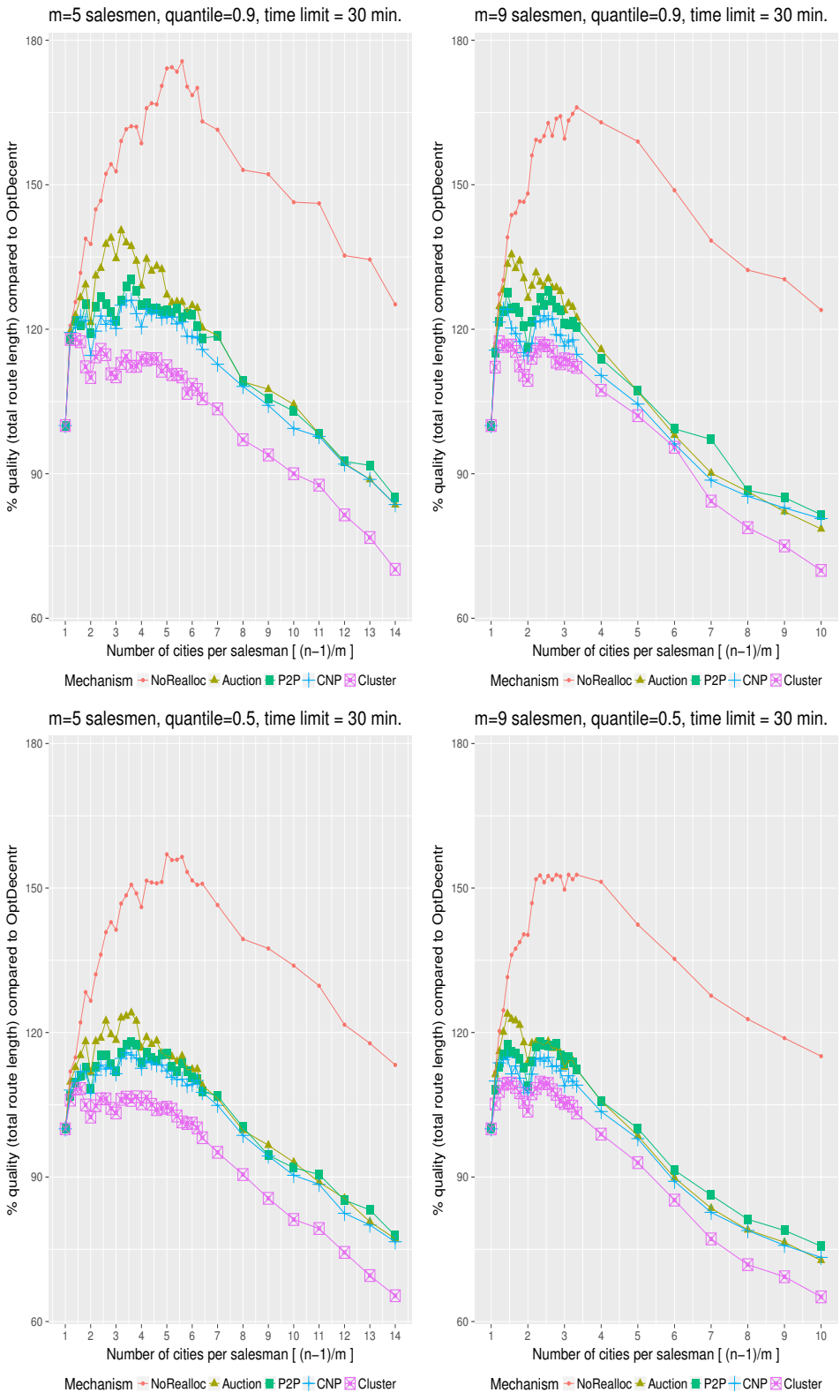


Figure 9: Ratios of quality compared to OptDecentr for $m = 5$ (left) and $m = 9$ (right) salesmen when the time limit is 30 minutes.

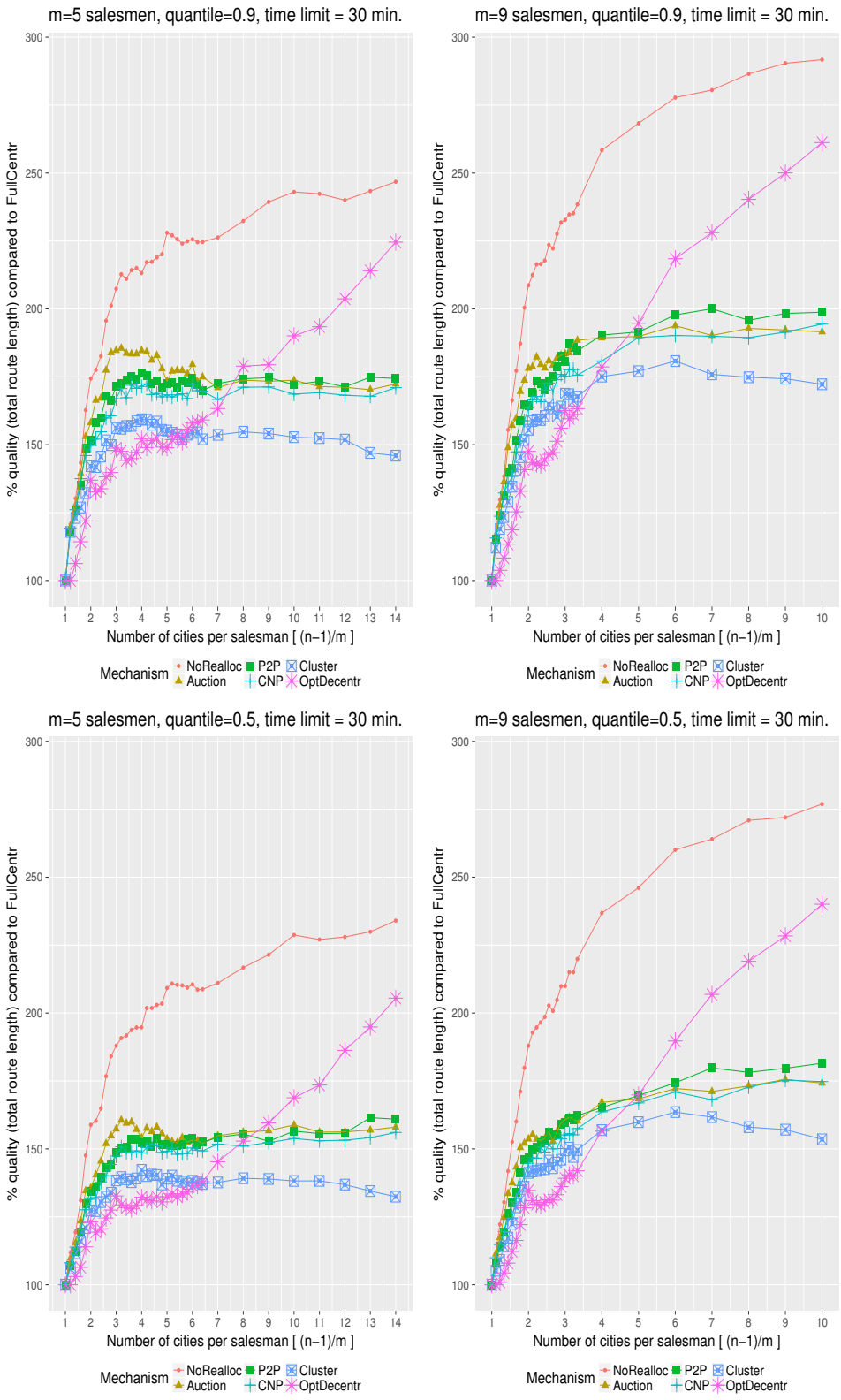


Figure 10: Ratios of quality compared to FullCentr for $m = 5$ (left) and $m = 9$ (right) salesmen when the time limit is 30 minutes.

than 4 cities per salesman for $m = 9$ in lower right), OptDecentr finds the best DO solution because the time limit of 30 minutes does not stop this mechanism too early.

– *For larger instances*, OptDecentr does not have enough time to find the optimum. Therefore, we look at the quality of Cluster instead, because it is the best DO mechanism. In other words, we use Cluster in order to infer the quality that OptDecentr would obtain without the time limit for such larger instances. In fact, OptDecentr would find the optimal solution of our modified MTSP if there were no time limit, hence we assume that the quality of the solution found by Cluster is an upper bound of a perfect DO mechanism.

Since the median route length of Auction reaches a plateau at about 130% (respectively, 160%) the length of FullCentr for $m = 5$ (respectively, $m = 9$), we conclude that these figures are upper bounds of the median route length of OptDecentr. As a conclusion, the modelling of selfishness in our modified MTSP increases the total route length by 30% (respectively, 60%) for $m = 5$ (respectively, $m = 9$) in comparison with the traditional MTSP.

- *(De-)centralisation seems to have a negligible impact on the quality of the solution:* CNP, Auction and P2P have very close results in Figure 10 when n is large, while the second of these mechanisms has a CA, $i.e.$, an auctioneer. The next bullet point indicates that this may due to the fact that this CA is not coercive.
- *Coercion seems to improve the quality of the solution:* Both Figures 9 and 10 suggest that the more coercive the CA in a mechanism, the better the quality of the solution found by this mechanism:
  - *Ranking of mechanisms by quality of the solution*: If we write $>$ for "is more efficient than", then we can see in Figures 9 and 10 that FullCentr $>$ OptDecentr $>$ Cluster $>$ Auction.
  - *Ranking of mechanisms by coercion level of their CA*: The same order applies to the coercion level of the CA in these four mechanisms:
    1. FullCentr has a more coercive CA than OptDecentr because she ignores the selfishness of the salesmen.
    2. The CA in OptDecentr is more coercive in Cluster because she controls both allocation and routing.
    3. The CA in Cluster requires to know of all cities while, as said in the previous bullet point, she is just a mediator in Auction.

Shortly, Figures 9 and 10 for high value of n suggest the following ranking (from most to least efficient): FullCentr $>$ OptDecentr $>$ Cluster $>$ (Auction $\approx$ CNP $\approx$ P2P) $>$ NoRealloc.

6 Discussion

This discussion puts into perspective this article with regard to the general question of the price of the selfishness in DO. This presentation is carried out according to our contributions:

6.1 Conceptual contributions

This article focuses on one way to include DO features in human organisations – namely, the selfishness of the salesmen – into the traditional MTSP addressed by CO. However, please notice that other pairs of CO/DO problems are possible. These other possibilities may be summarised as follows:

- *Relax Assumption* Hyp. 3: As aforementioned, Hyp. 3 may be relaxed by replacing the constraint of 1-1 exchanges by n-n exchanges. Clearly, the cases with $n > 1$ may find better exchanges but a combinatorial number of possible exchanges would have to be considered by salesmen.
- *Modify the objective function in MTSP*: As shown in Subsection 2.2, we choose to keep the same social welfare as the objective function in the traditional MTSP, which eventually result in adding the constraint of 1-1 exchanges with an initial endowment of cities.

  Instead of such a modification of the constraints, we could have modified the objective function. For example, we could have i) given a value v_{ik} to each City i for each Salesman k (for instance, Salesman 1 earns v_{11}=€3 for visiting City 1 and v_{21}=€4 for City 2 while Salesman 2 receives v_{12}=€6 and v_{22}=€5 respectively) and ii) modified the objective function to make a trade-off between the distance travelled and the value earned for visiting cities (for example, this distance is transformed into a cost of fuel and this cost is subtracted from the money earned in the cities). With such a utility function, the salesmen would agree to increase the number of their cities when their value is high enough.

  Other examples of modifications rely on the fact that our modified MTSP relies on the utilitarian social welfare ($i.e.$, sum of individual utilities), but other mappings of individual utilities to a social welfare have been proposed ($e.g.$, maximum, minimum or product of individual utilities).
- *Derive a CO problem from a DO one*: Instead of adding the selfishness of the decision makers to a CO problem, other work in the literature do the contrary. For instance, Sallez et al. (2010) compare the dynamic allocation and routing in a real flexible manufacturing system managed by a DO mechanism to a CO benchmark which does not take all the constraints into account because of the combinatorial explosion.
- *Change who make decisions*: We assume that salesmen fight for cities, but the opposite is also possible. It is even possible that both salesmen and cities make decisions.

Finally, we observe that CO can estimate the performance of DO. In fact, OptDecentr implements CO in order to find the best solution of our modified MTSP, and this solution is a benchmark for Cluster, CNP, P2P and Auction.

6.2 Technical contributions

Our main technical contribution is the implementation of various mechanisms solving our modified MTSP. It is interesting to notice that we have implemented one mechanism per organisation, but other mechanisms are possible for all the organisations in Figure 1. For example, Cluster uses the MILP formulation proposed by Rao (1971), but others exist.[10]

We have chosen to measure the computation time of CPLEX only. On the one hand, we first started to program our mechanisms with a MILP solver written in Java, namely ojAlgo v40 (`http://www.ojalgo.org` in order to have only pure Java in AnyLogic. We stopped this and turned to CPLEX because it is a recognised benchmark while we know little about the performance of ojAlgo. On the other hand, we also first thought about using various tools and environments (Concorde, k-means, etc.), next use benchmarks to compare their computation times. Unfortunately, no recognised benchmarks exist and contradicting information may even be found, such as "C runs faster than Java because it operates on a lower level" and "Java runs faster than C because its virtual machine adapts the program to the computer".

6.3 Numerical contributions

Besides the data shown in Section 5, especially the bullet points in Subsection 5.2, our results are robust because each point in Figures 7, 8, 9 and 10 represent a decile calculated on 130 instances. Of course, our conclusions at the end of Subsection 5.2 may turn out to be wrong with other instances, as well as with other hypotheses about how to modify MTSP in order to take the selfishness of the salesmen into account.

7 Conclusion

This article compares more or less centralised organisations in order to quantify the cost of having selfish decision makers. For that purpose, our first contribution is to introduce a same problem with features for both Centralised Organisation (CO) and Decentralised Organisation (DO). We address this issue by constraining the Multiple Travelling Salesmen Problem (MTSP) with both 1-1 exchanges of cities an an initial endowment. Our second contribution is to be the first article comparing five decision organisations to solve a same joint allocation and routing problem, while

[10] We have also tested Mechanism Cluster with the formulation by Sağlam et al. (2006) to which we added our constraint for 1-1 exchanges. Our experimentation (not presented in this article) shows similar performance than the formulation by Rao (1971) described in Paragraph 3.3.1, even though the obtained clusters sometimes differ.

the few other similar comparisons only address two organisations. Our third contribution is the quantification of the cost of decentralising the making of decisions. We think that the most interesting result is the fact that DO ($i.e.$, our modified MTSP) has a median total route length which reaches a plateau ≈30% (respectively, ≈60%) longer than CO ($i.e.$, the traditional MTSP) when there are five (respectively, nine) salesmen and many cities. This stabilisation was hoped but unpredictable without experimentation. We also notice that the coercion level of CA seems to impact much more on the quality of the solution than the level of centralisation of a mechanism.

As future work, we plan to study how our model of selfishness impacts on the efficiency of the mechanisms. For that purpose, we will use the model of preferences detailed in the discussion section in which the salesmen make a trade-off between the value of cities and the distance travelled, and then adapt our mechanisms to this other variant of MTSP.

Acknowledgement

We would like to thank the department of Industrial Engineering of INSA-Lyon, France for the use of the computers in one of its student laboratories in order to obtain all the results in this article.

References

Caballero, A., Botía, J., and Gómez-Skarmeta, A. (2011). Using cognitive agents in social simulations. *Engin. Appl. of Artif. Intel.*, 24(7):1098–1109.

Davidsson, P., Henesey, L., Ramstedt, L., Törnquist, J., and Wernstedt, F. (2005). Agent-based approaches to transport logistics. *Transportation Research - Part C Emerging Technologies*, 13(4):255–271.

Davidsson, P., Persson, J. A., and Holmgren, J. (2007). On the integration of agent-based and mathematical optimization techniques. In Carbonell, J. G. and Siekmann, J., editors, *Proc. 1st KES Int. Symposium, Agent and Multi-Agent Systems: Technologies and Applications, Lecture Notes in Computer Science 4496 (Springer)*, pages 1–10, Wroclaw, Poland.

Ellman, M. (1989). *Socialist Planning, 2d edition*. Cambridge University Press.

Fisher, K., Müller, J. P., Pischel, M., and Schier, D. (1995). A model for cooperative transportation scheduling. In *Proc. 1st Int. Conf. MultiAgent Systems*, San Francisco, CA, USA.

Fisher, K., Müller, J. P., Pischel, M., and Schier, D. (1996). Cooperative transportation scheduling: An application domain for DAI. *J. of Applied Artificial Intelligence*, 10(1):1–33.

Frayret, J.-M. (2002). *A conceptual framework to operate collaborative manufacturing networks*. Ph. D. thesis, Univ. Laval, Quebec City, PQ, Canada.

Frey, D., Nimis, J., Wörn, H., and Lockemann, P. (2003). Benchmarking and robust multi-agent-based production planning and control. *Engineering Applications of Artificial Intelligence*, 16(4):307–302.

Glaschenko, A., Ivaschenko, A., Rzevski, G., and Skobelev, P. (2009). Multi-agent real time scheduling system for taxi companies. In *Proc. 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Budapest, Hungary.

Gunady, M. K., Walid, Gomaaa, and Takeuchi, I. (2014). Aggregate Reinforcement Learning for multi-agent territory division: The Hide-and-Seek game. *Engin. Applic. of Artif. Intel.*, 34:122–136.

Hanna, L. and Cagan, J. (2009). Evolutionary multi-agent systems: An adaptative and dynamic approach to optimization. *J. of Mechanical Design*, 131:011010–8–011010–1.

Karmani, R. K., Latvala, T., and Agha, G. (2007). On scaling multi-agent task reallocation using market-based approach. In *Proc. 1st Int. Conf. on Self-Adaptive and Self-Organizing Systems (SASO'07)*, Boston, MA, USA.

Kivelevitch, E., Cohen, K., and Kumar, M. (2013). A market-based solution to the multiple traveling salesmen problem. *J. Intel. & Robotic Sys.*, 72:21–40.

Máhr, T., Srour, J., de Weerdt, M., and Zuidwijk, R. (2010). Can agents measure up? A comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research - Part C*, 18:99–119.

Mes, M., van der Heijden, M., and van Harten, A. (2007). Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *Eur. J. of Operational Research*, 181(0):59–75.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Interger programming formulation of traveling salesmen problems. *J. Association of Computing Machinery*, 7:326–9.

Mintzberg, H. (1978). *The Structuring of Organizations: A Synthesis of the Research*. Englewood Cliffs, N.J.: Prentice-Hall.

Moyaux, T. and McBurney, P. (2012). Centralised vs. market-based and decentralised decision-making: A review. *Cybernetics & Systems*, 43(7):567–622.

Rao, M. R. (1971). Cluster analysis and mathematical programming. *J. of the American Statistical Association*, 66(335):622–626.

Sallez, Y., Berger, T., Raileanu, S., Chaabane, S., and Trentesaux, D. (2010). Semi-heterarchical control of FMS: From theory to application. *Eng. Applic. Artif. Intell.*, 23:1314–1326.

Sarraj, R., Ballot, E., Pan, S., Hakimi, D., and Montreuil, B. (2014). Interconnected logistic networks and protocols: Simulation-based efficiency assessment. *Int. J. Production Research*, 52(11):3185–3208.

Sağlam, B., Salman, F. S., Sayn, S., and Türkay, M. (2006). A mixed-integer programming approach to the clustering problem with an application in customer segrmentation. *Eur. J. of Operational Research*, 173:866–879.

Smith, R. G. (1980). The contract net protocol: High level communication and control in distributed problem solver. *IEEE Trans. on Computers*, C-29(12):1104–1113.

Talbi, E.-G. (2006). *Parallel Combinatorial Optimization*. Wiley.

van Lon, R. R. S. and Holvoet, T. (2015). Towards systematic evaluation of multi-agent systems in large scale and dynamic logistics. In *Proc. 18th Inter. Conf. Principles and Practice of Multi-Agent Systems (PRIMA 2015)*, Bertinoro, Italy.

Xie, X.-F. and Liu, J. (2009). Multiagent optimization system for solving the traveling saleman problem (TSP). *IEEE Trans. Syst., Man, and Cybernetics – Part B: Cybernetics*, 39(2):489–502.