Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 47 (2020) 243–250

Transportation Research Procedia

www.elsevier.com/locate/procedia

22nd EURO Working Group on Transportation Meeting, EWGT 2019, 18-20 September 2019, Barcelona, Spain

Routing Drones in Smart Cities: a Biased-Randomized Algorithm for Solving the Team Orienteering Problem in Real Time

A. A. Juan[a,b], A. Freixes[a,b,*], J. Panadero[a,b], C. Serrat[c], A. Estrada-Moreno[d]

[a] *IN3 -– Computer Science Dept., Universitat Oberta de Catalunya, Av. Carl Friedrich Gauss 5, 08860 Castelldefels, Spain*
[b] *Euncet Business School, Ctra. Terrassa a Talamanca, 08225 Terrassa, Spain*
[c] *Dept. of Mathematics – IEMAE – EPSEB, Universitat Politècnica de Catalunya – BarcelonaTECH, 08028, Barcelona, Spain*
[d] *Dept. d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Av. Països Catalans 26, 43007 Tarragona, Spain*

Abstract

The concepts of unmanned aerial vehicles and self-driving vehicles are gaining relevance inside the smart city environment. This type of vehicles might use ultra-reliable telecommunication systems, Internet-based technologies, and navigation satellite services to decide about the routes they must follow to efficiently accomplish their mission and reach their destinations in due time. When working in teams of vehicles, there is a need to coordinate their routing operations. When some unexpected events occur in the city (e.g., after a traffic accident, a natural disaster, or a terrorist attack), coordination among vehicles might need to be done in real-time. Using the team orienteering problem as an illustrative case scenario, this paper analyzes how the combined use of extremely fast biased-randomized heuristics and parallel computing allows for 'agile' optimization of routing plans for drones and other autonomous vehicles.

Keywords: smart cities; unmanned aerial vehicles; team orienteering problem.

1. Introduction

Smart cities represent a multidisciplinary research field that is under a continuous updating process driven by urban, social, and technology evolution (Angelidou, 2015). Due to a generalized raise in population, modern cities also suffer from pollution and traffic-congestion problems that need to be efficiently addressed to guarantee their long-term sustainability. Different mathematical methods and computational algorithms can contribute to provide insight and automated decision making in the field of smart cities. For example, simulation, optimization, and searching algorithms (e.g., metaheuristics) can be used in processing the gathered data so that decision-making can be supported.

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: afreixes@euncet.es

2352-1465
10.1016/j.trpro.2020.03.095

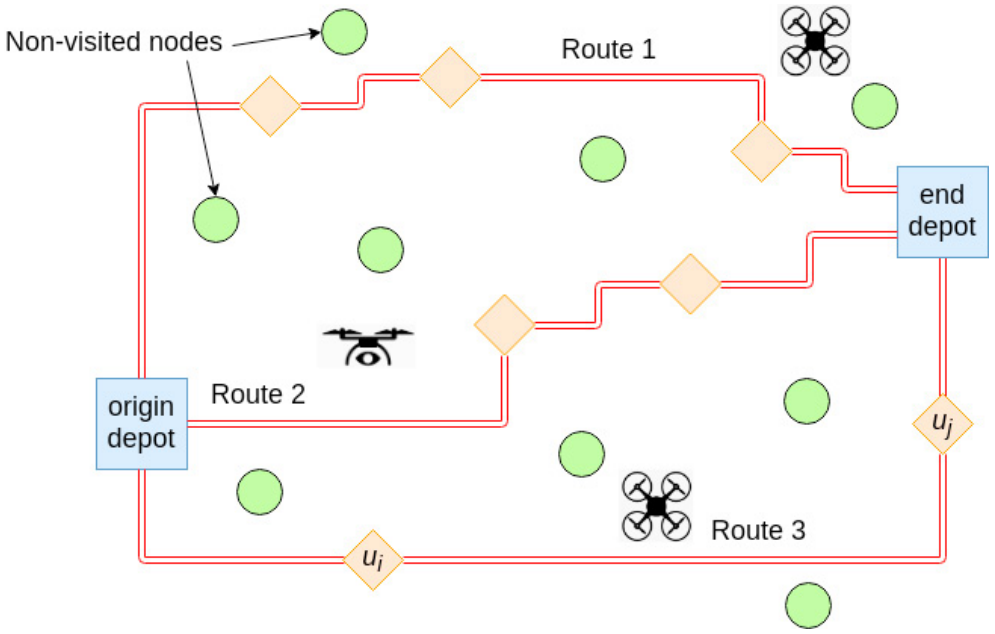

Fig. 1. The request for quotation process in logistics services.

However, in smart cities there is also a need for developing 'agile' algorithms that can cope with large-scale systems in 'real-time' (a few seconds or even milliseconds in some applications).

Unmanned aerial vehicles or drones are known for autonomous operation and mobility. Though there are first studies available (Jensen, 2016), the usage of drones in smart cities is not fully explored yet. Until today they were mainly used for military activities, surveillance, security, precision agriculture, as well as goods and services deliveries (Menouar et al., 2017). However, there are still concerns on their effective implementation in smart cities.

With the use of reliable and intelligent transportation systems, it would be possible to replace human road support teams by a set of self-driving vehicles or even drones overflying highways to monitor possible traffic violations and accidents, or for providing specific information to other transport users. Further features of an intelligent transportation system might support multiple flying drones that could act as accident report agents, roadside units, speed cameras, police eyes, and dynamic traffic signals. The execution of these specific roles requires coordination of a group of vehicles when planning the routes that need to be covered by the team of drones.

The team orienteering problem (TOP), introduced by Chao et al. (1996), is a variant of the well-known vehicle routing problem (Caceres-Cruz et al., 2015) in which a set of vehicle tours are constructed in such in a way that (Figure 1): *(i)* the total collected reward received from visiting a subset of customers is maximized; and *(ii)* the length of each vehicle tour is restricted by a pre-specified limit. In the case of electric vehicles, such as drones, it is natural to consider this limit due to the battery duration. Being an extension of the vehicle routing problem, the TOP is also *NP-hard* problem. Recent methods used to solve the TOP have included particle swarm optimization (Dang et al., 2013), simulated annealing (Lin, 2013), genetic algorithms (Ferreira et al., 2014), a Pareto mimic algorithm (Ke et al., 2016), and branch-and-cut-and-price algorithms (Keshtkaran et al., 2016).

To the best of our knowledge, this is the first work addressing the TOP in the context of smart cities, where dynamic conditions must be considered and real-time responses are required. This allows for introducing the concept of 'agile' optimization in vehicle routing operations inside smart cities and discuss how biased-randomized heuristics can support this concept. Hence, the main contributions of this paper are: *(i)* a fast constructive heuristic to provide real-time solutions for the TOP; and *(ii)* the extension of the previous heuristic to a probabilistic algorithm via biased-randomization techniques, which when combined with parallel computing generates even better solutions also in real times.

The remaining of the paper is structured as follows. Section 2 provides some related work on the team orienteering problem that serves as a scenario for testing our concepts. In Section 3, we explain the main ideas behind the concept of 'agile' optimization. Section 4 describes the heuristic approach designed to solve the problem and how this can be transformed into a biased-randomized algorithm that can be easily parallelizable. Details on the parallelization stage are given in Section 5. Section 6 introduce the computational experiments that have been carried out to test the biased-randomized algorithm. Finally, Section 7 highlights the main contributions of our work as well as some future research lines.

2. Related Work on Orienteering Problems

The original orienteering problem was introduced by Golden et al. (1987). The problem is *NP-hard* and, as a result, the majority of solution methods have relied on heuristics. The simplest version of the problem considers a vehicle which chooses a set of nodes to visit during a specified time interval to maximize its reward. Gunawan et al. (2016) provides an excellent overview of the many variations that have been introduced since the problem was initially proposed. We focus here on the team orienteering problem, in which a team of m vehicles (unmanned aerial vehicles in our case) aims to maximize their combined reward from visiting a selection of nodes within a given time limit. The problem was first introduced in Chao et al. (1996), who extended their methodology from the single vehicle problem to consider multiple vehicles. They use a multilevel optimization approach with three levels: select which points to visit; assign points to each member of the team; and finally, determine the shortest path for each team member around the points they have been assigned. Such sequential approaches reduce the size of the overall problem but at the risk of removing the optimal solution from the resultant solution space.

The vast majority of methods to solve the TOP are heuristic-based. Heuristics and metaheuristics are generally fast and can provide near-optimal solutions. Using exact methods, optimal solutions have also been obtained for mid-sized problems with up to 100 vertices (Butt and Ryan, 1999). In Keshtkaran et al. (2016), the authors propose a branch-and-price algorithm and use dynamic programming to solve pricing problems. In El-Hajj et al. (2016), the authors apply a cutting planes approach, while Dang et al. (2013) employ a branch-and-cut approach. Campos et al. (2014) use a greedy randomized adaptive search procedure with path re-linking. Ke et al. (2016) provide a Pareto-mimic algorithm in which a Pareto front of non-dominated solutions are iteratively evolved. Their algorithm improved the best-known solutions to ten benchmark instances.

Many recent approaches have been using metaheuristics, including particle swarm optimization (Dang et al., 2013), simulated annealing (Lin, 2013), and genetic algorithms (Ferreira et al., 2014; Kobeaga et al., 2018). These approaches outperforms the results obtained by the previous heuristics, but the computational time to find a near-optimal solution increases. Finally, a stochastic version of the problem is considered in Panadero et al. (2017).

3. The 'Agile' Optimization Concept

'Agile' optimization algorithms represent a new paradigm in the design of optimization algorithms, which follows the following principles: *(i)* extremely fast execution, thus providing real-time decision support; *(ii)* easy to implement and run using parallelization techniques; *(iii)* flexibility to deal with different problems and variants; *(iv)* parameter-less, hence avoiding complex and time-costly fine-tuning processes; and *(v)* specifically designed to run iteratively every few seconds or minutes –hence allowing for high-frequency re-optimization– as new streams of data arrive in a dynamic and connected environment. This novel approach represents a breakthrough with respect to traditional optimization, simulation, and machine learning methods, which typically require long computation times –and, therefore, cannot deal with present and future transportation scenarios using unmanned and self-driving vehicles, which are characterized by their dynamism and uncertainty. Agile optimization works in an environment of dynamic (constantly changing) conditions, whereas traditional optimization tends to oversimplify these important aspects of the real world. Traditional optimization, simulation, and machine learning frameworks are limited when dealing with real-time coordination and optimization needs in current and future transportation applications in urban areas. This is especially the case when electric, unmanned, and connected / self-driving vehicles are considered.

Agile optimization is based on the hybridization of biased-randomized heuristics (Juan et al., 2013a) and parallel computing (Juan et al., 2013b). By using skewed probability distributions, biased randomization techniques introduce

```
1: sol ← generateDummySolution(Inputs)
2: savingsList ← computeSortedSanvingsList(Inputs, α)
3: while (savingsList is not empty) do
4:         arc ← selectNextArc(savingsList, β )
5:         iRoute ← getStartingRoute(arc)
6:         jRoute ← getClosingRoute(arc)
7:         newRoute ← mergeRoutes(iRoute, jRoute)
8:         travelTimeNewRoute ← calcRouteTravelTime(newRoute)
9:         isMergeValid ← validateMergeDrivingConstraints(travelTimeNewRoute, drivingRange)
10:        if (isMergeValid) then
11:                sol ← updateSolution(newRoute, iRoute, jRoute, sol)
12:        end if
13:        deleteEdgeFromSavingList(arc)
14: end while
15: sortRoutesByProfit(sol)
16: deleteRoutesByProfit(sol, maxVehicles)
17: return sol
```

Fig. 2. The proposed heuristic.

a special (non-uniform) randomization effect into a heuristic procedure –which can be defined as a set of common sense steps that can generate good or even high-quality solutions for complex decision-making problems. As a result, a deterministic heuristic –which is extremely fast in execution, even for large-scale optimization problems- is extended into a probabilistic algorithm without losing the logic behind the original heuristic. Thousands of these probabilistic algorithms can be run in parallel by using an affordable computing device, thus requiring the same wall-clock time as the original heuristic (typically in the order of milliseconds), but with the benefit that many alternative solutions –some of them outperforming in quality the one provided by the original heuristic– can be considered by the decision maker. Since their first appearance in the scientific literature, these biased-randomization techniques have been successfully tested in multiple scenarios regarding different transportation optimization problems (Dominguez et al., 2016; Martin et al., 2016; De Armas et al., 2017).

Whereas the proposed approach can outperform the solutions provided by traditional heuristics, it does not increment wall-clock times (something that would unfortunately happen if other approaches such as metaheuristics, exact methods, simulation, or even machine learning methods were employed instead). Hence, agile optimization represents a new perspective to that of traditional optimization, which typically assume a fixed duration time horizon in which conditions are non-dynamic. In contrast, agile optimization embraces the large scale and dynamism of the real world, with its constantly evolving environmental conditions (e.g., traffic, vehicles location, unexpected demands, disruptions, etc.). Offering real-time solutions in these dynamic and large-scale scenarios requires the new paradigm, especially when unmanned and self-driving electric vehicles are considered.

4. From a Heuristic to a Biased-randomized Algorithm

Our heuristic is inspired by the well-known savings heuristic for the vehicle routing problem (Clarke and Wright, 1964), but adapted to consider the particular characteristics of the TOP: *(i)* the origin and destination depots may be different; *(ii)* not all of the customers need to be visited; and *(iii)* the reward collected by visiting nodes must be considered during the construction of the routing plan. Figure 2 provides a high-level description of our constructive heuristic.

It starts by generating an initial dummy solution (line 1), in which one route per customer is considered so that for each customer $i \in A$, a vehicle departs from the origin depot (node 0), visits i, and then resumes its trip towards the destination depot (node $n + 1$). If any route in this dummy solution does not satisfy the driving-range constraint, the associated customer is discarded from the problem, since it cannot be reached with the current fleet of vehicles. Next, we compute the 'savings' associated with each edge connecting two different customers (line 2), i.e. the benefits

obtained by visiting both customers in the same route instead of using two distinct routes. In order to compute the savings associated with an edge, one has to consider both the travel time required to traverse that edge as well as the aggregated reward generated by visiting both customers. The savings associated with an edge ij, s'_{ij} are defined as $s'_{ij} = \alpha \cdot s_{ij} + (1 - \alpha) \cdot (u_i + u_j)$ to account for the trade-off between time-based savings $s_{ij} = t_{i(n+1)} + t_{0j} - t_{ij}$, and the aggregated reward, $u_i + u_j$, where $\alpha \in (0, 1)$ is a tuning parameter, which is dependent on the heterogeneity of customers in terms of rewards. The specific value of α needs to be empirically tuned, since it will depend on the heterogeneity of the customers in terms of rewards. Thus, in a scenario with high heterogeneity, α will be close to zero. On the contrary, α will be close to one for homogeneous scenarios. Notice that for each edge there are two associated savings, depending on the actual direction in which the edge is traversed. Thus, each edge generates two different arcs. After computing all savings, the list of arcs can be sorted from higher to lower savings. Then, a route-merging process, based on this sorted savings list, is started. In each iteration, the arc at the top of the sorted list is selected (line 4). This arc connects two routes, which are merged into a new route as far as this new route does not violate the driving-range constraint (line 9). Finally, the list of routes are sorted according to the total reward provided (line 15) to select as many routes from this list as possible taking into account the restricted number of vehicles in the fleet.

The previously described heuristic can be extended into a probabilistic algorithm as follows. The constructive heuristic is combined with biased-randomization techniques, which relax somewhat the greedy behavior of the heuristic (Grasas et al., 2017). In particular, biased-randomization techniques use skewed probability distributions to induce an 'oriented' (non-uniform) random behaviour. This process turns a deterministic heuristic into a randomized algorithm whilst preserving the logic behind the original greedy heuristic. In our case, this biased randomization process is introduced by employing a Geometric probability distribution with a parameter β ($0 < \beta < 1$) which controls the relative level of greediness present in the randomized behaviour of the algorithm. Following a tuning process, a good performance was observed for a β value of 0.3. Thus, for the numerical experiments this value was used.

5. Parallelization Issues

The intrinsic characteristics of biased-randomized algorithms, explained in the previous section, make them a good candidate for parallelization. Typically, biased-randomized heuristics are wrapped in a multi-start framework, which is a sequential and iterative algorithm. Instead of using this sequential algorithm, we can run multiple instances of the biased-randomized heuristic in a parallel way, increasing the speedup to obtain a solution in real-time. In the context of unmanned aerial vehicles, which have to take decisions in a few seconds or even in milliseconds, distributed computing is not a suitable approach to be considered due to the latencies of the network. Thus, we must use on-chip parallelism programming models, as could be multi-threading or multi-processing approaches. In this sense, graphical process units (GPUs) offer massively parallel computation, providing an extraordinary performance and relative energy efficiency in comparison to traditional processors. A GPU consists of hundreds of smaller low-energy cores, which are divided in groups named streaming multiprocessors, each of which contains a fixed set of independent processing cores. Due to the benefits of this massively architecture, GPUs has become an enabling technology for embedded real-time systems, as could be the unmanned aerial vehicles or the autonomous vehicles (Hussain and Zeadally, 2018), to tackle general purpose computations in a efficient and fast way.

6. Computational Experiments

The previous algorithm was implemented as a Java application and run over a standard computer. For a set of 34 benchmark instances provided in Chao et al. (1996), Table 1 shows detailed information regarding the best-known solution (BKS), as provided in Ke et al. (2016), and our 'real-time' solutions, including: *(i)* the savings-based heuristic explained above; and *(ii)* the biased-randomized version of that heuristic (BRA), which can be executed in parallel using multiple threads. Notice that the average gaps are 2.18% and 0.87%, respectively. These gaps are quite small considering that, while the heuristics can be run in just 0.13 seconds on the average, the BKS have been obtained after an average time which is close to 2 minutes of computing.

Figure 3 shows the percentage gaps between: *(i)* the heuristic and the BKS; and *(ii)* the biased-randomized algorithm and the BKS. All in all, our biased-randomized algorithm is able to clearly outperform the solutions provided by

Table 1. Obtained results for a set of classical instances for the team orienteering problem.

Instance	BKS	Heuristic	BRA	Time (s)	Gap H - BKS	Gap BRA - BKS
p1.4.j	75	70	75	0.07	6.67	0.00
p1.4.k	100	95	100	0.13	5.00	0.00
p1.4.l	120	120	120	0.20	0.00	0.00
p1.4.p	175	165	175	0.12	5.71	0.00
p2.2.b	120	120	120	0.12	0.00	0.00
p2.2.e	190	190	190	0.06	0.00	0.00
p2.2.f	200	200	200	0.08	0.00	0.00
p2.2.g	200	200	200	0.11	0.00	0.00
p2.3.i	200	200	200	0.06	0.00	0.00
p2.3.j	200	200	200	0.20	0.00	0.00
p2.3.k	200	200	200	0.04	0.00	0.00
p2.4.d	70	70	70	0.24	0.00	0.00
p2.4.e	70	70	70	0.17	0.00	0.00
p2.4.f	105	105	105	0.10	0.00	0.00
p2.4.g	105	105	105	0.15	0.00	0.00
p2.4.h	120	120	120	0.23	0.00	0.00
p2.4.i	120	120	120	0.24	0.00	0.00
p2.4.j	120	120	120	0.14	0.00	0.00
p2.4.k	180	180	180	0.16	0.00	0.00
p3.2.a	90	90	90	0.22	0.00	0.00
p3.2.b	150	150	150	0.09	0.00	0.00
p3.2.c	180	170	180	0.12	5.56	0.00
p3.2.f	300	290	290	0.08	3.33	3.45
p3.2.i	460	440	440	0.23	4.35	4.55
p3.3.n	570	550	550	0.10	3.51	3.64
p3.3.o	590	580	580	0.22	1.69	1.72
p3.3.r	710	690	710	0.20	2.82	0.00
p3.4.k	350	350	350	0.08	0.00	0.00
p3.4.m	390	360	380	0.01	7.69	2.63
p3.4.n	440	430	430	0.09	2.27	2.33
p3.4.p	560	510	530	0.07	8.93	5.66
p3.4.t	670	670	670	0.10	0.00	0.00
p3.4.k	350	320	350	0.13	8.57	0.00
p3.4.l	380	350	360	0.17	7.89	5.56
Averages	*260.59*	*252.94*	*256.76*	*0.13*	*2.18*	*0.87*

the heuristic and provide, in milliseconds, results that are below a 1% gap with respect to the near-optimal solutions provided in the literature.

7. Conclusions and Future Work

In this paper, the concept of 'agile' optimization has been introduced, and the need for developing new algorithms that can provide good-quality solutions in extremely short computing times has been discussed, specially in the context of unmanned aerial vehicles and self-driving vehicles. The paper proposes a fast heuristic algorithm for the team orienteering problem. This heuristic is then transformed into a probabilistic algorithm by employing biased-randomization techniques. Biased-randomization algorithms are extremely powerful and easy to be run in parallel, thus providing higher-quality solutions than heuristics without increasing the associated wall-clock times. Our numer-

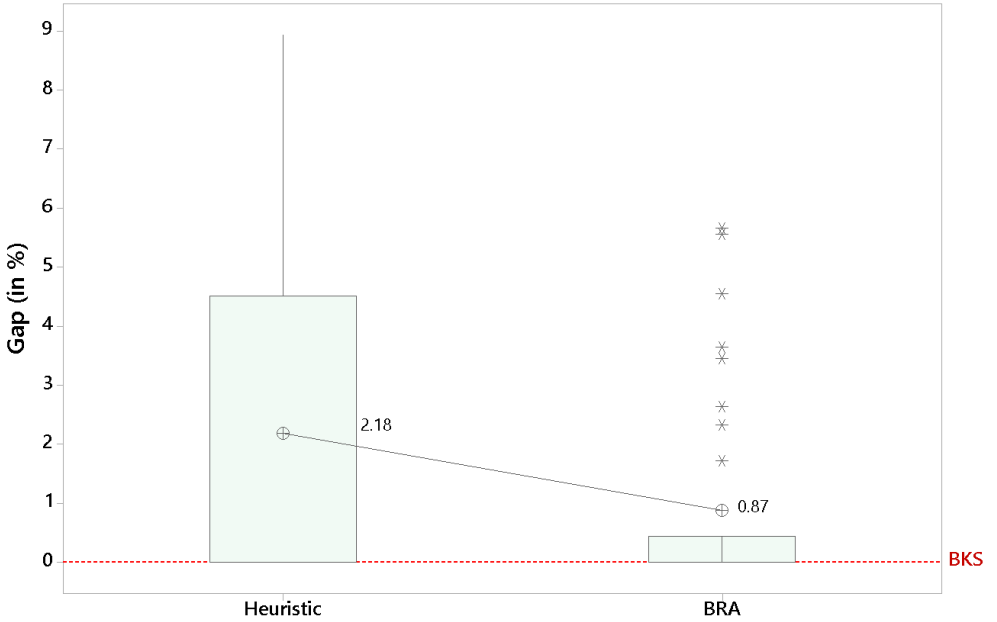

Fig. 3. Gaps with respect to the best-known solution using 'real-time' algorithms.

ical experiments make use of a well-known set of benchmarks for the team orienteering problem to show that it is possible to generate high-quality solutions in just a few milliseconds.

One possible extension of our research could be to consider a dynamic environment in which some of the inputs (e.g., rewards or travel times) are changing over time. This would force the routing plans to be adapted accordingly, i.e.: at each instant, all future decisions can be re-evaluated and modified as new input values are provided by sensors, in-route vehicles, video-cameras, etc.

Acknowledgements

This work has been partially supported by the SmartTransLog@EU Erasmus+ Consortium (2018-1-ES01-KA103-049767), and the Catalan Agency for Management of University and Research Grants (2017-DI-066).

References

Angelidou, M., 2015. Smart cities: A conjuncture of four forces. Cities 47, 95–106.

Butt, S.E., Ryan, D.M., 1999. An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computers & Operations Research 26, 427–441.

Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., Juan, A.A., 2015. Rich vehicle routing problem: Survey. ACM Computing Surveys (CSUR) 47, 32.

Campos, V., Martí, R., Sánchez-Oro, J., Duarte, A., 2014. Grasp with path relinking for the orienteering problem. Journal of the Operational Research Society 65, 1800–1813.

Chao, I.M., Golden, B.L., Wasil, E.A., 1996. A fast and effective heuristic for the orienteering problem. European journal of operational research 88, 475–489.

Clarke, G., Wright, J.W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations research 12, 568–581.

Dang, D.C., Guibadj, R.N., Moukrim, A., 2013. An effective pso-inspired algorithm for the team orienteering problem. European Journal of Operational Research 229, 332–344.

De Armas, J., Juan, A.A., Marquès, J.M., Pedroso, J.P., 2017. Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic. Journal of the Operational Research Society 68, 1161–1176.

Dominguez, O., Guimarans, D., Juan, A.A., de la Nuez, I., 2016. A biased-randomised large neighbourhood search for the two-dimensional vehicle routing problem with backhauls. European Journal of Operational Research 255, 442–462.

El-Hajj, R., Dang, D.C., Moukrim, A., 2016. Solving the team orienteering problem with cutting planes. Computers & Operations Research 74, 21–30.
Ferreira, J., Quintas, A., Oliveira, J.A., Pereira, G.A., Dias, L., 2014. Solving the team orienteering problem: developing a solution tool using a genetic algorithm approach, in: Soft Computing in Industrial Applications. Springer, pp. 365–375.
Golden, B.L., Levy, L., Vohra, R., 1987. The orienteering problem. Naval Research Logistics (NRL) 34, 307–318.
Grasas, A., Juan, A.A., Faulin, J., de Armas, J., Ramalhinho, H., 2017. Biased randomization of heuristics using skewed probability distributions: a survey and some applications. Computers & Industrial Engineering 110, 216–228.
Gunawan, A., Lau, H.C., Vansteenwegen, P., 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research 255, 315–332.
Hussain, R., Zeadally, S., 2018. Autonomous cars: Research results, issues and future challenges. IEEE Communication Surveys & Tutorials doi:10.1109/COMST.2018.2869360.
Jensen, O.B., 2016. Drone city–power, design and aerial mobility in the age of "smart cities". Geographica Helvetica 71, 67–75.
Juan, A.A., Faulin, J., Ferrer, A., Lourenço, H.R., Barrios, B., 2013a. Mirha: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. Top 21, 109–132.
Juan, A.A., Faulin, J., Jorba, J., Caceres, J., Marquès, J.M., 2013b. Using parallel & distributed computing for real-time solving of vehicle routing problems with stochastic demands. Annals of Operations Research 207, 43–65.
Ke, L., Zhai, L., Li, J., Chan, F.T., 2016. Pareto mimic algorithm: An approach to the team orienteering problem. Omega 61, 155–166.
Keshtkaran, M., Ziarati, K., Bettinelli, A., Vigo, D., 2016. Enhanced exact solution methods for the team orienteering problem. International Journal of Production Research 54, 591–601.
Kobeaga, G., Merino, M., Lozano, J.A., 2018. An efficient evolutionary algorithm for the orienteering problem. Computers & Operations Research 90, 42–59.
Lin, S.W., 2013. Solving the team orienteering problem using effective multi-start simulated annealing. Applied Soft Computing 13, 1064–1073.
Martin, S., Ouelhadj, D., Beullens, P., Ozcan, E., Juan, A.A., Burke, E.K., 2016. A multi-agent based cooperative approach to scheduling and routing. European Journal of Operational Research 254, 169–178.
Menouar, H., Guvenc, I., Akkaya, K., Uluagac, A.S., Kadri, A., Tuncer, A., 2017. Uav-enabled intelligent transportation systems for the smart city: Applications and challenges. IEEE Communications Magazine 55, 22–28.
Panadero, J., de Armas, J., Currie, C.S., Juan, A.A., 2017. A simheuristic approach for the stochastic team orienteering problem, in: 2017 Winter Simulation Conference (WSC), IEEE. pp. 3208–3217.