Improved Algorithms for the Capacitated Team Orienteering Problem

Gianlorenzo D'Angelo
Gran Sasso Science Institute, L'Aquila, Italy

Mattia D'Emidio
Dept. of Information Engineering, Computer Science and Mathematics,
University of L'Aquila, L'Aquila, Italy

Esmaeil Delfaraz
Dept. of Information Engineering, Computer Science and Mathematics,
University of L'Aquila, L'Aquila, Italy

Gabriele Di Stefano
Dept. of Information Engineering, Computer Science and Mathematics,
University of L'Aquila, L'Aquila, Italy

Abstract

We study the Capacitated Team Orienteering Problem, where a fleet of vehicles with capacities have to meet customers with known demands and prizes for a single commodity. The objective is to maximize the total prize and to assign a sequence of customers to each vehicle while keeping the total distance traveled within a given budget and such that the total demand served by each vehicle does not exceed its capacity. The problem has been widely studied both from a theoretical and a practical point of view. The contribution of this paper is twofold: (1) We advance the theoretical knowledge on the problem by providing new approximation algorithms that achieve, under some natural assumption, improved approximation ratios compared to the current best algorithms; (2) We propose four efficient heuristics that outperform the current state-of-the-art practical methods in the sense that they compute solutions that collect nearly the same prize in a significantly smaller running time. We also experimentally test the scalability of the new heuristics, showing that their running time increases approximately linearly with the size of the input, allowing us to process large graphs which were not possible to analyze before.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis; Theory of computation → Design and analysis of algorithms

Keywords and phrases Vehicle Routing, Approximation algorithms, Algorithm Engineering

Digital Object Identifier 10.4230/OASIcs.ATMOS.2024.7

Funding This work was partially supported by: Project EMERGE: innovation agreement between Ministry of Economical Development, Abruzzo Region, Radiolabs, Elital, Leonardo, Telespazio, University of L'Aquila and Centre of EXcellence EX-Emerge, funded by Italian Government under CIPE n. 70/2017; Group for Scientific Computation of Istituto Nazionale di Alta Matematica (GNCS-INdAM); the European Union - NextGenerationEU under the Italian Ministry of University and Research National Innovation Ecosystem grant ECS00000041 - VITALITY - CUP: D13C21000430001; PNRR MUR project GAMING "Graph Algorithms and MinINg for Green agents" (PE0000013, CUP D13C24000430001); the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program RESTART), project MoVeOver/SCHEDULE ("Smart interseCtions witH connEcteD and aUtonomous vehicLEs", CUP J33C22002880001); ARS01_00540 - RASTA project, funded by the Italian Ministry of Research PNR 2015–2020. The first author acknowledges the support of the MUR (Italy) Department of Excellence 2023–2027.

24th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2024).
Editors: Paul C. Bouman and Spyros C. Kontogiannis; Article No. 7; pp. 7:1–7:17
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The *Capacitated Orienteering Problem* (C-OP) is an NP-hard combinatorial optimization problem belonging to the wide class of Vehicle Routing Problems (VRPs), which has received much attention in the literature on algorithms and operation research [2,6,8,13,16–19,22,25,26]. In C-OP, we are given a complete graph, with edge lengths, where each node represents a customer that is assigned a profit/prize and a demand/size. Given two nodes s and t the goal is to find a path from s to t that maximizes the total prize, and respects both a capacity constraint on the total size of the nodes on the s-t path and a budget constraint on the total length of the s-t path. C-OP is a natural generalization of two very well-known problems, namely the *Knapsack Problem* [27], which is a special case of C-OP where the length of all edges is zero, and the *Orienteering Problem* (OP) [9,10], which is a variant of C-OP where the size of all nodes is zero. A generalization of C-OP is the case in which the goal is to find s-t paths for a fleet of K homogeneous, capacitated vehicles that can be used to collect prizes. This problem is known as the *Capacitated Team Orienteering Problem* (C-TOP) and has been defined by Archetti et al. [2], originally in the flavor when $s = t$, i.e. when the aim is finding tours centered at a depot node, rather than paths.

From a theoretical viewpoint, the best known approximation algorithms for C-OP and C-TOP, that run in polynomial time, are due to Bock and Sanità, who achieved approximation factors of $(3 + \varepsilon)$, for any $\varepsilon > 0$, and 3.53, respectively [8]. From a practical perspective, for both C-OP and C-TOP, several heuristics without guarantees on the achieved quality of solution and exact algorithms with exponentially large worst-case running times have been introduced and experimentally evaluated with the aim of characterizing their practical effectiveness and applicability, i.e. evaluating the quality of the computed solutions and the running time necessary to achieve such solutions (see [1, 2, 6, 16, 17, 19, 22, 25, 26]). In all benchmark instances for the C-OP and C-TOP problems considered in such works, the prize of each node is fixed to be at least equal to half of its size. Specifically, the prize of each node v with size $r(v)$ is assigned to be equal to $\pi(v) = (h + 0.5)r(v)$, where h is a random number uniformly generated within interval $[0, 1]$ [2, 26]. This implies that, for two nodes u and v with $r(u) \geq r(v)$, we have $\pi(u) \geq \pi(v)/3$. Motivated by this observation, we consider problems C-OP and C-TOP under the natural assumption that choosing subsets of nodes with larger sizes results in achieving (almost) more prizes. In more detail, we assume that any subset S of nodes collects an overall prize that is at least equal to a multiplicative factor $\lambda \in (0, 1]$ times the prize collected by any subset of nodes whose sum of sizes is lower than the sum of the sizes of nodes in S (see Assumption 2.1 for a formal definition).

Our Contribution. The contribution of this paper is both theoretical and experimental. From the theoretical viewpoint, we improve over the state-of-the art by providing approximation algorithms, for C-OP and C-TOP, that guarantees an approximation ratio which, under particular assumptions, is smaller than the best approximation ratio known so far. In particular, we propose a $\max\{\alpha, \frac{2}{\lambda}\}$-approximation algorithm for C-OP and a $(1 - e^{-\frac{1}{\beta}})^{-1}$-approximation algorithm for C-TOP, where α is the approximation factor of an algorithm for OP, $\beta = \max\{\alpha, \frac{2}{\lambda}\}$, and λ is the parameter of Assumption 2.1. Observe that, the best known approximation algorithm for OP is that given by Chekuri et al. [9], which guarantees an approximation factor of $2+\varepsilon$, for any $\varepsilon > 0$. When $\lambda \in [\frac{2}{3}, 1]$, our algorithms for C-OP and C-TOP achieve approximation factors in the intervals $[2 + \varepsilon, 3]$ and $[2.55, 3.53)$, respectively, for any $\varepsilon > 0$. These improve over the long-standing results by Bock and Sanità [8] who achieved factors $3 + \varepsilon$ and 3.53 for C-OP and C-TOP, respectively, for any $\varepsilon > 0$.

Since our algorithms with theoretical guarantees have high computational complexity, we propose four efficient heuristic algorithms that do not give any proven guarantee on the quality of the computed solution but achieve good performance in practice. We experimentally evaluate our heuristics in benchmark instances from the literature and show that two of them produce solutions that are comparable to the best solutions known from the literature in terms of collected prize, but outperform all the state-of-the art algorithms in terms of running time. In particular, our heuristics require less than a second on the small instances (at most 100 nodes) and two orders of magnitude less time than other algorithms on large instances (at most 577 nodes), while achieving the same prize in most cases, a slightly worse prize in a few cases, and even a better prize in a few cases. To assess how the time performance of our heuristics scales with the input size, we also generated new instances with up to 15 500 nodes, starting from real-world road networks. Our experiments in these instances suggest that the running time of two of our algorithms tends to grow approximately linearly with the input size and highlight that, on the largest instance, such two algorithms take below one minute on average, whereas previous algorithms are not able to handle such large input graphs.

Related Work. Blum et al. [7] gave the first constant factor approximation algorithm for OP with approximation ratio of 4 when $s = t$ and showed that: (i) no polynomial-time approximation algorithm can achieve a factor better than $\frac{1481}{1480}$; (ii) OP is APX-hard. Bansal et al. [5] improved the bound of [7] by designing a 3-approximation algorithm for the case where $s = t$ while Chekuri et al. [9] proposed a $(2+\varepsilon)$-approximation algorithm that works for any positive constant ε. Friggstad and Swamy [15] designed, via LP-rounding, a 3-approximation algorithm when $s = t$. Paul et al. [23], gave a 2-approximation algorithms for OP when s and t are not given in advance. Finally, Chen and Har-Peled [10] gave a PTAS for the case where the points lie in a constant-dimensional Euclidean metric space.

A natural generalization of OP is the Team Orienteering Problem (TOP) where we are asked to find $K \geq 1$ paths from s to t that maximize the total prize, accumulated by all the K paths, and such that each path respects the budget B. Blum et al. [7] studied TOP under the name of *Multi-Path Orienteering problem* (M-OP) and showed that: (i) any α approximation for OP, when $s = t$, can be translated into a $1/(1-e^{-\alpha})$ approximation for M-OP; (ii) their algorithm for M-OP has a factor of $\alpha + 1$ when the starting point of each vehicle is arbitrary. Friggstad et al. [14] studied a variant of M-OP in the case where each vehicle needs to find a tour and each node has a cost. The goal is to find K tours so that the minimum total prize among all tours is maximized, i.e. to find $P' : \pi(P') = \max\min_P \pi(P)$. They called this problem *max-min* orienteering and showed that any α-approximation algorithm for OP results in an $(\alpha+2)$-approximation for *max-min* orienteering. Xu et al. [28] studied a variant of TOP in which the prize function is a special submodular function and showed the existence of a $1/(1-e^{-\alpha})$-approximation algorithm for such variant, where α is an approximation factor to OP. Finally, Xu et al. [29] focused on TOP when $s = t$, they call this variant the *monitoring reward maximization* problem and presented a 3-approximation algorithm. Clearly, C-OP is a generalization of OP in which we also consider node demands $r : V \to \mathbb{N}$ and a capacity bound C. Gupta et al. [18] showed that, given an α-approximation algorithm for OP, it is possible to derive a 2α-approximation algorithm for C-OP. By using the $(2+\varepsilon)$-approximation algorithm for OP [9], this leads to a $(4+\varepsilon)$-approximation for C-OP. Bock and Sanità [8] improved this result by giving a $(1+\alpha+\varepsilon)$-approximation algorithm for C-OP and by presenting a PTAS on trees and a PTAS on Euclidean metrics. Again, using the $(2+\varepsilon)$-approximation algorithm for OP, results in a $(3+\varepsilon)$-approximation for C-OP. For C-TOP, Bock and Sanità [8] designed a $(1-e^{\frac{1}{\beta}})$-approximation algorithm, where β is an approximation factor for C-OP. Using $\beta = 3+\varepsilon$ this leads to a 3.53-approximation algorithm for C-TOP.

2 Notation and Definitions

We are given an undirected complete graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, respectively. Let $l : E \to \mathbb{R}_{\geq 0}$ be a metric *length* function on edges, let $\pi : V \to \mathbb{R}_{\geq 0}$ be a *prize* function on the nodes, let $r : V \to \mathbb{R}_{\geq 0}$ be a *size* function on the nodes, and let $g : V \to \mathbb{R}_{\geq 0}$ be a *service time* function on the nodes. For any subgraph G' of G, we denote by $V(G')$ and $E(G')$ the set of nodes and edges in G', respectively. Given a subset $S \subseteq V$, $G[S]$ denotes the subgraph of G induced by S, i.e., $E(G[S]) = \{\{u, v\} \in E \mid u, v \in S\}$.

For an integer k, let $[k] := \{1, 2, \dots, k\}$. A *path* P_{uv} from node u to node v is a graph made of a sequence of distinct nodes $\{v_1 = u, \dots, v_k = v\}$ and a sequence of edges $\{v_i, v_{i+1}\}$, where $i \in [k-1]$. The cost of a path P_{uv} in G is the sum of the lengths of its edges and service times of its nodes, i.e., $\sum_{e \in E(P_{uv})} l(e) + \sum_{v \in V(P_{uv})} g(v)$. Given a path $P = (s, v_2, v_3, \dots, t)$ and a subset $S = \{v_{i_1}, v_{i_2} \dots, v_{i_k}\}$ of $k \geq 1$ nodes in $V(P) \setminus \{s, t\}$ with $i_j < i_{j+1}$ for $j \in [k-1]$, we call $P[S]$ the *subpath of P induced by S* which is the path made of nodes $\{s, t\} \cup S$ and edges $\{\{s, v_{i_1}\}, \{v_{i_k}, t\}\} \cup \{\{v_{i_j}, v_{i_{j+1}}\} : j \in [k-1]\}$.

In the Capacitated Orienteering Problem (C-OP), we are given two distinguished nodes $s, t \in V$, a cost budget $B \in \mathbb{R}_{\geq 0}$ and a capacity bound $C \in \mathbb{R}_{\geq 0}$ on the sizes, and the goal is to find a path P_{st} from s to t in G that maximizes the prize $\pi(P_{st}) = \sum_{v \in V(P_{st})} \pi(v)$ and that satisfies both $l(P_{st}) + g(P_{st}) = \sum_{e \in E(P_{st})} l(e) + \sum_{v \in V(P_{st})} g(v) \leq B$ and $r(P_{st}) = \sum_{v \in V(P_{st})} r(v) \leq C$. W.l.o.g. we assume that $r(v) \leq C$, for any $v \in V$, and that $r(s) = r(t) = 0$. The Capacitated Team Orienteering Problem (C-TOP) is a generalization of C-OP in which we are asked to find $K \geq 1$ vertex-disjoint paths that maximize the total collected prize and each path respects both the capacity and the budget constraints. Formally, in C-TOP, the goal is to find K paths $P^1_{st}, \dots, P^K_{st}$ from s to t that maximize $\sum_{k=1}^{K} \sum_{v \in P^k_{st}} \pi(v)$, and such that $l(P^k_{st}) + g(P^k_{st}) = \sum_{e \in E(P^k_{st})} l(e) + \sum_{v \in V(P^k_{st})} g(v) \leq B$ and $r(P^k_{st}) = \sum_{v \in V(P^k_{st})} r(v) \leq C$ for any $k \in [K]$.

In the remainder of the paper, we assume w.l.o.g. that the service times of s and t are equal to 0, that is $g(s) = g(t) = 0$. This implies that we can ignore the cost of service time by moving it to the edge length function. More formally, we redefine the length and service time functions as follows: the length is $l'(e) = l(e) + \frac{g(v)+g(u)}{2}$, for each edge $e = (u, v) \in E$, while the service time is $g'(v) = 0$ for each node $v \in V$. The cost of any path P_{st}, with the new functions, is therefore equal to $\sum_{e \in E(P_{st})} l'(e) = \sum_{e=(u,v) \in E(P_{st})} \left(l(e) + \frac{g(v)+g(u)}{2} \right) = \sum_{e \in E(P_{st})} l(e) + \sum_{v \in V(P_{st})} g(v)$. This implies that under this transformation: (1) the cost of any path is equal to the length of the path, and (2) clearly, the triangle inequality property is preserved. Thanks to this transformation, for the sake of simplicity and w.l.o.g., from now on we assume that any graph with node service times is converted to an equivalent graph with zero node service times. For the sake of readability, the obtained length l' will be denoted by l.

Given a subset of nodes $V' \subseteq V$, let $r(V') = \sum_{v \in V'} r(v)$ and $\pi(V') = \sum_{v \in V'} \pi(v)$. In all the benchmark instances that have been considered in the literature on C-OP, we observe that node size is positively correlated to node prize. In fact, in such real-world inspired instances, the prize of a node v is equal to $\pi(v) = (0.5 + h)r(v)$, where h is a random value in $[0, 1]$ (see [2, 25]). This implies that, for any two subsets of nodes $V_1, V_2 \subseteq V$ with $r(V_1) \geq r(V_2)$, we have $\pi(V_1) \geq \frac{1}{3}\pi(V_2)$, since $\pi(V_1) \geq \frac{1}{2}r(V_1)$ and $\pi(V_2) \leq \frac{3}{2}r(V_2) \leq \frac{3}{2}r(V_1)$. Indeed, in many practical applications we have that the prize of a subset of nodes increases as its size increases, that is for two subsets of nodes $V_1, V_2 \subseteq V$ with $r(V_1) \geq r(V_2)$, we have $\pi(V_1) \geq \lambda\pi(V_2)$, for some $\lambda \in (0, 1]$. Therefore, in this paper we consider C-OP and C-TOP under the following natural assumption.

Algorithm 1

Input: $I = \langle G = (V, E), s, t, \pi, l, B, r, C \rangle$.
Output: An $(s - t)$ path P_{st} s.t. $l(P_{st}) \leq B$ and $r(P_{st}) \leq C$.
Let $I' = \langle G = (V, E), s, t, \pi, l, B \rangle$ be an instance of OP;
Apply an A_{OP} to I'; let P_α be the returned solution;
if $r(P_\alpha) \leq C$ **then** $P_{st} \leftarrow P_\alpha$;
else // $r(P_\alpha) > C$
  Choose a subset of nodes $S \subseteq V(P_\alpha) \setminus \{s, t\}$ with $r(S) \geq C$ such that, for some $v \in S$, we have $r(S \setminus \{v\}) \leq C$;
  if $r(S) = C$ **then** $P_{st} \leftarrow P_\alpha[S]$;
  else // $r(S) > C$
    Let v be a node in S such that $r(S \setminus \{v\}) \leq C$;
    Partition S into two subsets $S_1 = S \setminus \{v\}$ and $S_2 = \{v\}$;
    Let $S' = \arg\max_{A \in \{S_1, S_2\}} \pi(A)$;
    $P_{st} \leftarrow P_\alpha[S']$;
return P_{st};

▶ **Assumption 2.1.** *Let $\lambda \in (0, 1]$ be a parameter to be fixed. For any two subsets of nodes $V_1, V_2 \subseteq V$ with $r(V_1) \geq r(V_2)$, we have $\pi(V_1) \geq \lambda\pi(V_2)$.*

Note that, Assumption 2.1 implies that selecting subsets of nodes with larger sizes results in collecting more prize, besides a multiplicative factor λ.

3 Approximation Algorithms with Theoretical Guarantees

In this section, we introduce some polynomial time algorithms for C-OP and C-TOP that guarantee bounded approximation ratios under Assumption 2.1. We first focus on C-OP under that assumption and introduce a polynomial time $\max\{\alpha, \frac{2}{\lambda}\}$-approximation algorithm where α is the approximation ratio guaranteed by an algorithm A_{OP} for OP that is used as a subroutine while $\lambda \in (0, 1]$ is the parameter of Assumption 2.1. Then, we show that this result implies, under some particular conditions, an improvement over the best known approximation ratio for C-OP. Finally, we show how to use this algorithm to approximate C-TOP.

Our main algorithm, whose pseudo-code is summarized in Algorithm 1, takes as input an instance $I = \langle G = (V, E), s, t, \pi, l, B, r, C \rangle$ of C-OP. Starting from I, Algorithm 1 defines an OP instance I' with $I' = \langle G = (V, E), s, t, \pi, l, B \rangle$ and executes algorithm A_{OP} onto it. Let P_α be the solution returned by A_{OP} when applied to I'. An optimal solution $OPT_{I'}$ to instance I' of OP has value at least $\pi(OPT_{I'}) \geq \pi(OPT_I)$, where OPT_I is an optimal solution to the instance I of C-OP. It follows that, if $r(P_\alpha) \leq C$, then P_α is an α-approximation also for I. Therefore, if $r(P_\alpha) \leq C$, Algorithm 1 returns P_α as a solution. If $r(P_\alpha) > C$, Algorithm 1 chooses a subset of nodes $S \subseteq V(P_\alpha) \setminus \{s, t\}$ such that $r(S) \geq C$ and, for some $v \in S$, we have $r(S \setminus \{v\}) \leq C$. Now if $r(S) = C$, then Algorithm 1 returns $P_\alpha[S]$, the subpath of P_α induced by S, as a solution. Otherwise, it partitions S into two subsets of nodes $S_1 = S \setminus \{v\}$ and $S_2 = \{v\}$, where v is a node in S such that $r(S_1) \leq C$. Note that $r(v) \leq C$ and hence both S_1 and S_2 are feasible solutions for I. Finally, Algorithm 1 selects the set with the maximum prize between S_1 and S_2, denoted by $S' = \arg\max_{A \in \{S_1, S_2\}} \pi(A)$, and returns $P_\alpha[S']$ as a solution. In the next theorem, we show that Algorithm 1 guarantees a $\max\{\alpha, \frac{2}{\lambda}\}$-approximation algorithm for C-OP under Assumption 2.1.

▶ **Theorem 1.** *Algorithm 1 is a polynomial time* $\max\{\alpha, \frac{2}{\lambda}\}$*-approximation algorithm for* C-OP *under Assumption 2.1, where* α *denotes the approximation ratio for* OP *and* $\lambda \in (0, 1]$.

Proof. If $r(P_\alpha) \leq C$, then Algorithm 1 returns solution P_α. By the feasibility of P_α for instance I', we have that $l(P_\alpha) \leq B$ and hence P_α is feasible for instance I of C-OP. Moreover, $\pi(P_\alpha) \geq \frac{1}{\alpha}\pi(OPT_{I'}) \geq \frac{1}{\alpha}\pi(OPT_I)$, and hence P_α provides an α-approximation for I. If $r(P_\alpha) > C$, then Algorithm 1 selects a set $S \subseteq V(P_\alpha) \setminus \{s,t\}$ with $r(S) \geq C$ such that there exists a node $v \in S$ for which $r(S \setminus \{v\}) \leq C$. We distinguish between two cases.

1. $r(S) = C$. In this case, Algorithm 1 returns $P_\alpha[S]$ as a solution. Since $l(P_\alpha) \leq B$, then, by triangle inequality, we have $l(P_\alpha[S]) \leq B$. Moreover, $r(P_\alpha[S]) = r(S) = C$, as $r(s) = r(t) = 0$, and hence $P_\alpha[S]$ is feasible for I. By Assumption 2.1, it follows that $\pi(S) \geq \lambda\pi(OPT_I)$, since $r(S) = C$ and $r(OPT_I) \leq C$. Hence, $P_\alpha[S]$ is a $\frac{1}{\lambda}$-approximation for I.
2. $r(S) > C$. In this case, Algorithm 1 partitions S into two subsets $S_1 = S \setminus \{v\}$ and $S_2 = \{v\}$, with $r(S_1) \leq C$, selects the set with the maximum prize between S_1 and S_2, say S', and returns $P_\alpha[S']$ as solution. Since $l(P_\alpha) \leq B$, then, by triangle inequality, it follows that $l(P_\alpha[S']) \leq B$. Moreover, both $r(S_1)$ and $r(S_2)$ are upper bounded by C and hence $P_\alpha[S']$ is feasible for I. Regarding the approximation factor of $P_\alpha[S']$, we have $\pi(S') \geq \frac{1}{2}\pi(S) \geq \frac{\lambda}{2}\pi(OPT_I)$, where the first inequality holds as S' is the set with the maximum prize between two sets $S_1, S_2 \subseteq S$ with $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$, and the second inequality follows by Assumption 2.1, as $r(S) \geq C$ and $r(OPT_I) \leq C$. Therefore, $P_\alpha[S']$ is a $\frac{2}{\lambda}$-approximation for I. ◀

Theorem 1, along with the $(2+\varepsilon)$-approximation algorithm for OP given by Chekuri et al. [9] implies the following result.

▶ **Corollary 2.** *For any fixed* $\varepsilon > 0$*, Algorithm 1 is a* $\max\{2+\varepsilon, \frac{2}{\lambda}\}$*-approximation algorithm for* C-OP*, under Assumption 2.1 where* $\lambda \in (0, 1]$.

When $\lambda \geq \frac{2}{3}$ in Assumption 2.1, then $\frac{2}{\lambda} \leq 3$ and hence the above corollary implies that the approximation factor of Algorithm 1 is in the interval $[2+\varepsilon, 3]$, for any $\varepsilon \in (0, 1]$. This is an improvement on the approximation of C-OP under Assumption 2.1 over the factor $3+\varepsilon$ by Bock and Sanità [8].

▶ **Corollary 3.** *For any fixed* $\varepsilon \in (0, 1]$*, Algorithm 1 is a* β*-approximation algorithm with* $\beta \in [2+\varepsilon, 3]$ *for* C-OP*, under Assumption 2.1 where* $\lambda \in [\frac{2}{3}, 1]$.

Another interesting implication of Theorem 1 is that an α-approximation algorithm for OP results in an α-approximation for C-OP, under Assumption 2.1, when $\lambda \geq \frac{2}{\alpha}$. In particular, under this hypothesis, Algorithm 1 is a $(2+\varepsilon)$-approximation algorithm for C-OP, by using the result by Chekuri et al. [9].

▶ **Corollary 4.** *For any fixed* $\varepsilon > 0$*, Algorithm 1 is a* $(2+\varepsilon)$*-approximation algorithm for* C-OP*, under Assumption 1 where* $\lambda \geq \frac{2}{2+\varepsilon}$.

A β-approximation algorithm ALG for C-OP can be used as a black-box, to obtain a $(1 - e^{-\frac{1}{\beta}})^{-1}$-approximation algorithm for C-TOP [8], using the following greedy strategy (named GenStra):

1. For $i = 1$ to K do:
   a. Run the β-approximation algorithm for C-OP to obtain a path P_i on $G = (V, E)$.
   b. Remove all covered nodes $V(P_i)$ from G.
2. Return $P_1, \ldots, P_K$.

Using this result, we can generalize our result for C-OP to C-TOP under Assumption 2.1.

▶ **Theorem 5.** *Under Assumption 2.1, there exists a polynomial time* $(1-e^{-\frac{1}{\beta}})^{-1}$*-approximation algorithm for* C-TOP*, where* $\beta = \max\{2+\varepsilon, \frac{2}{\lambda}\}$ *for any fixed* $\varepsilon > 0$*.*

Proof. The theorem follows from Theorem 1 and the fact that any β-approximation algorithm for C-OP can be used as a subroutine in GenStra to achieve a $(1-e^{-\frac{1}{\beta}})^{-1}$-approximation factor for C-TOP [8]. ◀

Theorem 5 implies that when $\lambda \geq \frac{2}{3}$ in Assumption 2.1, one can achieve a ρ-approximation algorithm for C-TOP, where $\rho \in [(1-e^{-\frac{1}{2+\varepsilon}})^{-1}, (1-e^{-\frac{1}{3}})^{-1}]$, for any $\varepsilon \in (0,1)$, where $(1-e^{-\frac{1}{2}})^{-1} > 2.55$ and $(1-e^{-\frac{1}{3}})^{-1} < 3.53$. This is an improvement for C-TOP under Assumption 2.1 over the factor $(1-e^{-\frac{1}{3+\varepsilon}})^{-1}$, for $\varepsilon > 0$, by Bock and Sanità [8].

▶ **Corollary 6.** *For any fixed* $\varepsilon \in (0,1)$*, under Assumption 2.1 with* $\lambda > \frac{2}{3}$*,* C-TOP *admits a* ρ*-approximation algorithm, where* $\rho \in [2.55, 3.53)$*.*

4 Heuristic Algorithms

The running time of Algorithm 1 presented in Section 3 is dominated by the time required to run an approximation algorithm for OP at line 2. If we use the $(2+\varepsilon)$-approximation algorithm by Chekuri et al. [9] for this purpose, this step requires $O(n^{O(1/\varepsilon^2)})$ time, for any $\varepsilon > 0$. In this section, motivated by such high computational time, we design four efficient heuristic algorithms that have low computational time but do not guarantee any bound on the quality of the computed solution. In Section 5, we experimentally evaluate the proposed heuristics on relevant sets of instances of C-TOP, showing that they also produce high-quality solutions. In particular we show that our heuristics require small computational time and that the value of the computed solutions is comparable to that achieved by state-of-the-art methods. Both in this section and in Section 5, we assume that $s = t$ in C-OP and C-TOP, that is we need to find a tour instead of a path. We refer to node s as *depot*.

In what follows, we describe our heuristics for C-OP. Each algorithm ALG for C-OP can be generalized to be used for C-TOP by applying GenStra stated in Section 3, where we use ALG instead of a β-approximation algorithm for C-OP, and we set $s = t$.

Our heuristic algorithms for C-OP exploit a procedure, named DPROC, to produce solutions that respect the capacity constraints starting from a set of nodes $S \subseteq V$. Such procedure works as follows: first, it computes a subset of nodes $S' \subseteq S$ that maximizes the prize $\pi(S')$ and has size at most $r(S') \leq C$ by using the well-known dynamic programming for the Knapsack problem [27]. Then, it determines a tour T that includes the depot s and all nodes in S' using an approximation algorithm for the Traveling Salesman Problem (TSP). Specifically, for all algorithms we consider two versions of DPROC, which use either the 3/2- or 2-approximation for TSP [27], respectively, and, in Section 5, we will specify how the two versions are used in the experiments. Finally, DPROC returns T as output. We remark that the input graph is complete and metric. In the following, we denote the application of procedure DPROC with input S by DPROC(S). Now, for any two nodes u and v, let $w(u,v) = l(e) \cdot r(v)$, be the *weight* of edge $e = (u,v)$. Our heuristic algorithms for C-OP are as follows. In Section 5, we will extend each algorithm for C-OP to C-TOP by using procedure GenStra and, we call its extension with the same name for C-OP.

sqrB-ApxA (SBAA). This algorithm is inspired by the algorithms given by Kuo et al. [21] and by D'Angelo et al. [12] for the problem of finding a rooted out-tree in a directed graph that maximizes the sum of prizes associated to the nodes, subject to a budget constraint. Specifically, for each node $u \in V$, we compute a candidate set S_u and, at the end of the algorithm, we consider a set S_M that maximizes the prize, i.e. $S_M := \arg\max_{u \in V} \pi(S_u)$ and output $\text{DPROC}(S_M)$. In details, the candidate set S_u of a node $u \in V$ is computed as follows. We first sort all nodes $v \in V$ in non-increasing order of $\pi(v)/w(u,v)$ or $\pi(v)$. In Section 5 we will describe how the two sorting strategies are used in the experiments. Then, we consider two integers x and y and, for each pair $(x,y) \in \{0,1,2\} \times [50]$, we compute: (i) the set S_y^x containing the first $yB^{1-x/2}$ nodes in the ordering that have a distance at most $B^{x/2}$ from u; (ii) a tour $T_y^x = \text{DPROC}(S_y^x)$ and check if $l(T_y^x) \leq B$. Then, set S_u is selected as a set that produces a feasible tour in the previous step and maximizes the prize, i.e. $S_u := \arg\max\{\pi(S_y^x) : l(T_y^x) \leq B, (x,y) \in \{0,1,2\} \times [50]\}$. To improve the running time, we exploit the monotonicity of function π, iterate through the values of y from $y = 50$ to $y = 1$ and stop as soon as we find a feasible tour. The values for x and y have been chosen after a preliminary pilot experimental study on the algorithm's performance.

4-ApxA (4AA). This heuristic is based on the idea of Gupta et al. [18] who showed that, given an α-approximation algorithm for OP, it is possible to derive a 2α-approximation algorithm for C-OP. So, we use the best approximation algorithm for the unrooted version of OP in which there is no specified root node s that must be spanned, which is the 2-approximation algorithm proposed by Paul et al. [23]. In particular, given an instance $I = \langle G = (V,E), s, \pi, r, l, B, C\rangle$ of C-OP, we define an OP instance I' with $I' = \langle G = (V,E), s, \pi', l, B\rangle$ in which for any $v \in V$, $\pi'(v) = \pi(v) - \eta r(v)$, where $\eta \geq OPT/(2C)$ and OPT is an optimal solution to C-OP. As OPT is not known, we guess it through a binary search over the range $[\pi_{\min}, TP]$, where $\pi_{\min}$ be the minimum positive prize of a node and TP is the total prize of vertices. We know that $OPT \leq TP$. We estimate the value of OPT by guessing N possible values, where N is the smallest integer for which $\pi_{\min}2^{N-1} \geq TP$. For the instances considered in Section 5, we set η using this binary search. For each η, we compute the solution returned by the 2-approximation algorithm by Paul et al. [23] on the obtained instance I' and we let S_η be the nodes in this solution. By definition of OP, set S_η satisfies the budgeted constraint but it is not guaranteed to satisfy the capacity. Therefore, we compute $T_\eta = \text{DPROC}(S_\eta)$ to obtain a tour that satisfies the capacity constraint. Finally we output the tour T_M that maximizes the prize, i.e. $T_M := \arg\max\{\pi(T_\eta)\}$, where η is set based on the binary search to find OPT. Note that for any v, in case $\pi'(v) = \pi(v) - \eta r(v) < 1$, we set $\pi'(v) = 1$.

GreedyRandom-ApxA (GRA). This is a modification of the randomized algorithm proposed by Arora and Scherer [4]. The following randomized algorithm is repeated multiple times and the solution with best prize is selected (in the experiments we repeat for 10 times). We keep a solution S, initially equal to the empty set. For $3|V|$ times we repeat the following loop. We sample a node v uniformly at random and we check if $v \in S$. If so, we remove it from S. Otherwise, we add it to S. Then, we compute $T = \text{DPROC}(S)$ and check if $l(T) \leq B$ and $\pi(T) > \pi(S)$. In the affirmative case, we set $S := V(T)$ and repeat the loop.

Greedy-ApxA (GA). Like for SBAA, we compute a candidate set S_u, for each node $u \in V$, we select a set S_M that maximizes the prize, i.e. $S_M := \arg\max_{u \in V} \pi(S_u)$, and output $\text{DPROC}(S_M)$. For each node $u \in V$, the candidate set S_u is computed as follows. We first sort all nodes $v \in V$ in non-increasing order of $\pi(v)/w(u,v)$ or $\pi(v)$. In Section 5 we will specify the used sorting strategy. We initialize S_u as $S_u := \{u\}$. Then, we iterate over the nodes $v \in V \setminus \{u\}$, according to the sorting. At each iteration we check whether adding to S_u the next node v in the sorting induces a feasible solution with better prize of the current solution. Specifically, we compute $T = \text{DPROC}(S_u \cup \{v\})$ and check whether $l(T) \leq B$ and $\pi(T) > \pi(S_u)$. In the affirmative case, we set $S_u := V(T)$ and iterate to the next node in the ordering.

Note that `SBAA`, `4AA`, `GA` and GreedyRandom-ApxA are pseudo-poly algorithms as we use the well-known dynamic programming for the Knapsack problem. However, one can use the well-known FPTAS for the knapsack problem [27].

5 Experiments

In this section, we present and analyze the results of an extensive experimental evaluation on the performance of the heuristics proposed in Section 4. We design two experiments, named respectively COMPARISON and SCALABILITY, with the objective of answering to different experimental questions.

The aim of experiment COMPARISON (see Section 5.1), is comparing the performance of the four proposed heuristic algorithms against methods of the literature that are considered the state-of-the-art for C-TOP. Among them, based on the most recent experimental results on the problem (see [19]), we identify the most effective/competitive w.r.t. solution quality and running time, that is algorithms: `VNS`, `TSF`, `TSA` [2]; `BiFFf` and `BiFFs` [25]; `VSS-Tb` and `VSS-SA` [6]; `HALNS` [19]. We do not consider, instead, algorithms `ADEPT-RD` [22], `SA-ILS` [17], and `LNS/NLNS` [20] since they have been tested only on a subset of the benchmark instances and, in terms of performance, they are dominated by or comparable to `HALNS` [19]; Furthermore, `ILS` [16] provided the average results on each set instead of giving their result on each instance.

The aim of experiment SCALABILITY (see Section 5.2), is assessing the scalability properties of our newly introduced heuristics, i.e. to study how the performance of our heuristics change with the input size, and specifically if our algorithms can process larger instances than those that have been considered in past literature on the problem. For all experiments, we use implementations of the four heuristics of Section 4 we developed for the purpose. All our code is written in C++ (available at `https://shorturl.at/bMYNb`) and compiled with GCC 9.4.0 with optimization level $O3$; all our tests have been executed on a workstation equipped with an Intel© Xeon© processor, clocked at 2.30GHz, running Ubuntu Linux.

5.1 comparison Experiment

In this experiment, we test implementations of `SBAA`, `4AA`, `GRA`, and `GA` on two publicly available datasets of benchmark inputs for C-TOP, defined in [2] and [25], respectively, derived from instances of TSP and considered reference instances for assessing the performance of algorithms for C-TOP.

Input Data. The details of such datasets, which we call SMALL-CASE and LARGE-CASE inputs, respectively, are summarized in what follows:

small-case: this set contains 130 instances (divided into three subsets, named SC-1, SC-2 and SC-3 and having 10, 90 and 30 instances, resp.) defined in [2] by suitably manipulating the instances given in [11] for TSP. The number of nodes of graphs in this set is $n \in \{51, 76, 101, 121, 151, 200\}$; instances are generated by considering different combinations of fleet size $K \in \{2, 3, 4, 10, 15, 20\}$, budget $B \in \{50, 75, 100, 160, 200, 230, 720, 1040\}$ and capacity $C \in \{50, 75, 100, 140, 160, 200\}$.

large-case: this set contains 130 instances (divided in three subsets, named LC-1, LC-2, and LC-3 and having 10, 90, and 30 instances, resp.) developed in [25] by modifying the inputs to the Periodic Vehicle Routing Problem of [24]. The number of nodes in this set is $n \in \{337, 361, 385, 433, 481, 529, 505, 577\}$ while other parameters are $K \in \{6, 7, 8, 14, 15, 16, 18, 20, 21, 22, 24\}$, $B \in \{100, 200, 400, 660, 668, 675, 683, 705, 713, 720\}$ and $C \in \{75, 150, 200, 330, 335, 340, 345, 350, 360, 365, 375\}$.

Each of the above instances, in what follows, is identified by a unique string following the format *base*-n-K-C-B, where *base* is the name of the original TSP instance from either [11] or [24], while n is the number of nodes, K is the number of vehicles, C is the capacity and B is the budget. Note that, for both SMALL-CASE and LARGE-CASE instances, the prize of each node v having size $r(v)$ is assigned to be equal to $\pi(v) = (h + 0.5)r(v)$, where h is a random number uniformly generated within interval $[0, 1]$. This implies that for any instance having capacity C and number of vehicles K, the optimum for the instance is upper bounded by $(h + 0.5)KC \leq \frac{3KC}{2}$ in C-TOP.

Executed Tests. For all mentioned inputs, we run all four heuristics and measure both running time (column t, in seconds) and solution quality (i.e. achieved *prize*, column p). We then compare observed measures with the results obtained, on the same instances, by reference methods of the literature mentioned above, as summarized in Tables 1–4.

Table 1 Results of experiment COMPARISON for SMALL-CASE inputs, subset SC-1.

Instance	GRA			SBAA			GA			4AA			VNS [2]			TSF [2]			TSA [2]			BiFFf [25]			BiFFs [25]			VSS-Tb [6]			VSS-SA [6]			HALNS [19]		
	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g
03-101-15-200-200	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	904	362	35.84	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	2	0.00	**1409**	< 1	0.00
06-51-10-160-200	**761**	< 1	0.00	**761**	< 1	0.00	**761**	< 1	0.00	191	73	74.90	**761**	< 1	0.00	**761**	< 1	0.00	**761**	< 1	0.00	**761**	< 1	0.00	**761**	< 1	0.00	**761**	< 1	0.00	**761**	< 1	0.00	**761**	< 1	0.00
07-76-20-140-160	**1327**	< 1	0.00	**1327**	< 1	0.00	**1327**	< 1	0.00	1238	146	6.70	**1327**	< 1	0.00	**1327**	< 1	0.00	**1327**	< 1	0.00	**1327**	< 1	0.00	**1327**	< 1	0.00	**1327**	< 1	0.00	**1327**	1	0.00	**1327**	< 1	0.00
08-101-15-200-230	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	916	391	34.98	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	< 1	0.00	**1409**	1	0.00	**1409**	< 1	0.00
09-151-10-200-200	2063	< 1	0.09	2064	< 1	0.04	2057	< 1	0.38	1586	1255	0.00	2064	3600	0.00	2061	163	0.00	2062	127	0.00	**2065**	2	0.00	**2065**	2	0.00	**2065**	39	0.00	**2065**	120	0.00	**2065**	1	0.00
10-200-20-200-200	**3048**	< 1	0.00	**3048**	< 1	0.00	**3048**	< 1	0.00	2828	4218	7.21	**3048**	< 1	0.00	**3048**	< 1	0.00	**3048**	< 1	0.00	**3048**	< 1	0.00	**3048**	< 1	0.00	**3048**	< 1	0.00	**3048**	11	0.00	**3048**	< 1	0.00
13-121-15-200-720	**1287**	< 1	0.00	**1287**	< 1	0.00	**1287**	< 1	0.00	**1287**	417	0.00	**1287**	< 1	0.00	**1287**	< 1	0.00	**1287**	< 1	0.00	**1287**	< 1	0.00	**1287**	< 1	0.00	**1287**	< 1	0.00	**1287**	2	0.00	**1287**	< 1	0.00
14-101-10-200-1040	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	< 1	0.00	**1710**	3	0.00	**1710**	< 1	0.00
15-151-15-200-200	**2159**	< 1	0.00	**2159**	< 1	0.00	**2159**	< 1	0.00	**2159**	1450	0.00	**2159**	< 1	0.00	**2159**	< 1	0.00	**2159**	< 1	0.00	**2159**	< 1	0.00	**2159**	< 1	0.00	**2159**	< 1	0.00	**2159**	7	0.00	**2159**	< 1	0.00
16-200-15-200-200	2965	< 1	0.13	2965	< 1	0.13	2966	< 1	0.10	2941	3829	0.94	2968	3600	0.03	2965	270	0.13	2967	377	0.06	-	-	-	-	-	-	**2969**	61	0.00	**2969**	254	0.00	**2969**	76	0.00

Observe that, for subsets of inputs SC-2, SC-3, LC-2 and LC-3, which have a large number of instances, we report a meaningful selection of the results of our tests, while full data will appear in a longer version of the paper. Besides running time and prize, for each algorithm A and for each instance, we report the gap g_A between the solution Sol_A computed by A and the best known solution for the instance, obtained by any of the algorithm in the set X of considered algorithms, i.e. $g_A = \frac{BKS - Sol_A}{BKS} \cdot 100$, where $BKS = \max_{A' \in X} Sol_{A'}$. Algorithms achieving the maximum solution quality, for each instance, are highlighted in bold. For the sake of fairness, we remark that all the considered algorithms from the literature have a randomized nature and have been evaluated by following a measurement strategy commonly referred to as *Time-To-Best*, which consists of: (i) running a given algorithm 10 times; (ii) selecting the run that performs best in terms of solution quality (prize); (iii) reporting solution quality and running time only of such run of the algorithm (see [19] and references therein). While this measurement strategy is reasonable when one compares only randomized solutions, it appears to be not well suited to be applied in comparisons that include deterministic algorithms, such as ours GA, SBAA or 4AA, which output the same solution even if they are executed multiple times. Indeed, a more empirically appropriate assessment strategy would require to measure, for randomized approaches, the sum of the running times of the all executions, since that represents the actual time the algorithm have to run to obtain the best solution, and compare such time with that of deterministic algorithms. Therefore, running times reported for algorithms from the literature might likely be underestimations of the actual average running time.

Note that, procedure DPROC, used by all our heuristics, considers different possibilities for computing a tour on a subset of the nodes, namely the 3/2- and 2-approximation algorithms for TSP [27]. Moreover, heuristics SBAA and GA use two different node sorting strategies, based on the prize or on the ratio between prize and weight. After a preliminary experimental

Table 2 Excerpt of the results of experiment COMPARISON for subsets SC-2 (top) and SC-3 (bottom).

Subset sc-2

Instance	GRA			SBAA			GA			4AA			VNS [2]			TSF [2]			TSA [2]			BiFFf [25]			BiFFs [25]			VSS-Tb [6]			VSS-SA [6]			HALNS [19]		
	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g
03-101-4-100-100	510	< 1	4.13	523	< 1	1.69	516	< 1	3.00	501	257	5.82	529	963	0.56	531	317	0.18	529	357	0.56	531	7	0.18	**532**	42	0.00	**532**	27	0.00	**532**	12	0.00	**532**	21	0.00
06-51-4-100-100	450	< 1	5.49	470	< 1	2.48	472	< 1	2.07	388	21	19.50	481	135	0.20	**482**	25	0.00	**482**	26	0.00	**482**	< 1	0.00	**482**	< 1	0.00	**482**	< 1	0.00	**482**	2	0.00	**482**	2	0.00
07-76-4-100-100	510	< 1	1.72	510	< 1	2.11	514	< 1	1.34	451	107	13.43	**521**	342	0.00	**521**	25	0.00	514	26	1.34	518	< 1	0.57	**521**	< 1	0.00	**521**	< 1	0.00	**521**	2	0.00	**521**	2	0.00
08-101-4-100-100	514	< 1	2.06	523	< 1	1.69	516	< 1	3.00	501	107	5.82	529	963	0.56	531	317	0.18	529	357	0.56	531	7	0.18	**532**	41	0.00	**532**	29	0.00	**532**	20	0.00	**532**	31	0.00
09-151-4-100-100	532	< 1	1.64	542	< 1	0.73	539	< 1	1.26	506	1074	7.32	545	2934	0.18	539	924	1.28	536	959	1.83	545	51	0.18	**546**	38	0.00	**546**	53	0.00	**546**	31	0.00	**546**	31	0.00
10-200-4-100-100	544	< 1	1.80	548	< 1	0.90	550	< 1	0.54	522	2969	5.60	548	3600	0.90	549	2077	0.72	550	3232	0.54	**553**	183	0.00	**553**	243	0.00	**553**	11	0.00	**553**	40	0.00	**553**	43	0.00
13-121-4-100-100	415	< 1	1.19	415	< 1	0.95	417	< 1	0.47	383	23	8.59	**419**	179	0.00	**419**	24	0.00	**419**	48	0.00	**419**	< 1	0.00	**419**	< 1	0.00	**419**	< 1	0.00	**419**	1	0.00	**419**	1	0.00
14-101-4-100-100	511	< 1	3.04	522	< 1	0.57	521	< 1	0.76	488	212	7.04	**525**	670	0.00	523	210	0.38	**525**	292	0.00	**525**	< 1	0.00	**525**	< 1	0.00	**525**	5	0.00	**525**	1	0.00	**525**	5	0.00
15-151-4-100-100	542	< 1	1.27	545	< 1	0.72	544	< 1	0.91	518	1077	5.64	548	2828	0.18	**549**	1252	0.00	545	1015	0.72	548	10	0.18	**549**	206	0.00	**549**	66	0.00	**549**	49	0.00	**549**	87	0.00
16-200-4-100-100	553	< 1	0.53	555	< 1	0.53	554	< 1	0.71	538	3009	3.58	554	3600	0.71	554	2124	0.71	553	3559	0.89	556	3	0.35	**558**	67	0.00	**558**	5	0.00	**558**	79	0.00	**558**	36	0.00

Subset sc-3

Instance	GRA			SBAA			GA			4AA			VNS [2]			TSF [2]			TSA [2]			BiFFf [25]			BiFFs [25]			VSS-Tb [6]			VSS-SA [6]			HALNS [19]		
	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g
03-101-4-200-200	936	< 1	1.47	938	< 1	1.26	939	< 1	1.15	914	927	3.78	**950**	961	0.00	946	110	0.42	947	42	0.31	**950**	< 1	0.00	**950**	< 1	0.00	**950**	16	0.00	**950**	11	0.00	**950**	10	0.00
06-51-4-160-200	681	< 1	0.29	682	< 1	0.14	680	< 1	0.43	648	82	5.12	**683**	53	0.00	**683**	5	0.00	682	4	0.14	**683**	< 1	0.00	**683**	< 1	0.00	**683**	< 1	0.00	**683**	1	0.00	**683**	1	0.00
07-76-4-140-160	705	< 1	0.28	705	< 1	0.28	704	< 1	0.42	686	376	2.97	**707**	296	0.00	**707**	44	0.00	702	39	0.70	**707**	2	0.00	**707**	5	0.00	**707**	2	0.00	**707**	1	0.00	**707**	< 1	0.00
08-101-4-200-230	947	< 1	0.31	949	< 1	0.10	946	< 1	0.42	924	916	2.73	**950**	726	0.00	949	89	0.10	949	38	0.10	**950**	1	0.00	**950**	10	0.00	**950**	8	0.00	**950**	8	0.00	**950**	11	0.00
09-151-4-200-200	1029	< 1	0.38	1031	< 1	0.19	1024	4	0.87	1008	3552	2.42	**1033**	2903	0.00	**1033**	480	0.00	1029	254	0.38	**1033**	2	0.00	**1033**	2	0.00	**1033**	44	0.00	**1033**	11	0.00	**1033**	16	0.00
10-200-4-200-200	**1064**	< 1	0.00	**1064**	< 1	0.00	1062	10	0.18	1060	8854	0.37	**1064**	3600	0.00	**1064**	1260	0.00	1063	789	0.09	**1064**	1	0.00	**1064**	< 1	0.00	**1064**	15	0.00	**1064**	9	0.00	**950**	11	0.00
13-121-4-200-720	**908**	< 1	0.00	**908**	< 1	0.00	**908**	< 1	0.00	**908**	1474	0.00	**908**	954	0.00	907	76	0.11	906	40	0.22	**908**	< 1	0.00	**908**	< 1	0.00	**908**	217	0.00	**908**	27	0.00	**908**	6	0.00
14-101-4-200-1040	**975**	< 1	0.00	**975**	< 1	0.00	**975**	< 1	0.00	**978**	< 1	0.00	**975**	483	0.00	**975**	56	0.00	975	38	0.00	**975**	1	0.00	**975**	1	0.00	**975**	< 1	0.00	**975**	1	0.00	**975**	< 1	0.00
15-151-4-200-200	1024	< 1	0.67	1027	< 1	0.38	1016	5	1.45	1010	3645	2.03	**1031**	2832	0.00	1019	618	1.16	1030	276	0.09	**1031**	3	0.00	**1031**	3	0.00	**1031**	262	0.00	**1031**	58	0.00	**1031**	7	0.00
16-200-4-200-200	1071	< 1	0.18	1071	< 1	0.18	1068	11	0.46	1062	9171	1.02	**1073**	3600	0.00	1072	1263	0.09	1071	897	0.18	**1073**	1	0.00	**1073**	1	0.00	**1073**	40	0.00	**1073**	15	0.00	**1073**	1	0.00

study, we found out that on instances SC-1, SC-2, and SC-3, on average the algorithms based on prize ordering performs worse in terms of collected prize than those based on prize-over-weight ordering. Therefore, for these instances we use the prize-over-weight ordering and both the 3/2- and 2-approximation algorithms for TSP. We then select the solution with the highest prize between these two and report the sum of the running times of both approaches. Similarly, for instances LC-1, LC-2 and LC-3, our preliminary experiments show that algorithms based on the 2-approximation algorithm for TSP perform worse than those based on the 3/2-approximation and hence, in these instances, we use only this latter and both prize and prize-over-weight orders. We then select the solution with the highest prize and report the sum of the running times of both approaches. Finally, for `SBAA` we fix parameter y, determining an upper bound on how many nodes can be assigned to each vehicle, to 20, whenever the number of vehicles is large enough so that the nodes of graphs can be divided among vehicles.

Analysis. Our data lead to two main general conclusions: first, the newly introduced algorithms are competitive with existing ones in terms of solution quality. In fact, they achieve, in many cases, best known solutions (i.e. have zero gap), and solutions with good quality, with gaps that are in the order of few percentage points, one or two tens in the worst cases, in the remaining cases. Second, `SBAA`, `GA` and `GRA` are significantly faster than methods known in the literature, requiring running times that are up to orders of magnitude smaller to achieve solutions that have comparable quality (with prizes equal or very close to the best ones and corresponding small gaps). The only exception to this behavior is algorithm `4AA`, whose running time does not scale well with the graph size, due to using the 2-approximation algorithm of [23]. For this reason, we omit from the comparison the results of algorithm `4AA` for larger instances, i.e. LARGE-CASE. In more details, we observe that:

Table 3 Results of experiment COMPARISON for LARGE-CASE inputs, subset LC-1.

Instance	GRA			SBAA			GA			BiFFf [25]			BiFFs [25]			VSS-Tb [6]			VSS-SA [6]			HALNS [19]		
	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g
01-337-14-345-720	3836	30	8.05	4139	1	0.79	4014	51	3.78	**4172**	17	0.00	**4172**	17	0.00	**4172**	1	0.37	**4172**	2	0.00	**4172**	1	0.00
02-385-16-350-713	4434	43	7.31	4753	< 1	0.64	4605	77	3.74	**4784**	25	0.00	**4784**	25	0.00	**4784**	1	0.37	**4784**	3	0.00	**4784**	2	0.00
03-433-18-330-675	4911	59	5.57	5187	< 1	0.26	4949	103	4.84	**5201**	33	0.00	**5201**	32	0.00	**5201**	2	0.37	**5201**	3	0.00	**5201**	4	0.00
04-481-20-335-713	5463	85	6.35	**5834**	2	0.00	5697	155	2.24	5828	48	0.10	5828	47	0.10	5828	1	0.10	5828	3	0.10	5828	3	0.10
05-529-22-340-705	5892	127	8.59	**6446**	5	1.95	6211	215	3.63	6445	101	0.01	6445	98	0.01	6445	3	0.01	6445	5	0.01	6445	3	0.01
06-577-24-365-683	6709	164	5.26	**7082**	5	0.00	6937	266	1.89	7071	94	0.15	7071	93	0.15	7071	1	0.15	7071	2	0.15	7071	6	0.15
07-361-15-335-668	3877	41	10.97	4134	< 1	5.07	4122	56	5.35	**4355**	24	0.00	**4355**	23	0.00	**4355**	1	0.37	**4355**	3	0.00	**4355**	1	0.00
08-433-18-350-675	4706	74	9.39	5133	2	1.17	4965	122	4.40	**5194**	51	0.00	**5194**	49	0.00	**5194**	2	0.37	**5194**	4	0.00	**5194**	1	0.00
09-505-21-360-660	5267	118	14.81	5841	5	5.53	5953	176	3.71	**6183**	103	0.00	**6183**	104	0.00	**6183**	3	0.37	**6183**	22	0.00	**6183**	5	0.00
10-577-24-375-675	6601	166	8.88	**7245**	8	0.00	7132	289	1.47	7239	144	0.08	7239	144	0.08	7239	4	0.08	7239	7	0.08	7239	6	0.08

- for SMALL-CASE instances, algorithms `GRA`, `SBAA` and `GA` outperform all other approaches in terms of running time by completing their execution always in less than 1 second; at the same time they compute best solutions in all cases with few exceptions where the gap is below 5%; in more details, for subset SC-1, algorithms `VSS-Tb`, `VSS-SA` and `HALNS` have running times up to 254 seconds (with zero gap) while `GRA` and `SBAA`, `GA` take less than 1 second (with gaps below 0.39%); for subset SC-2, similarly, `VSS-Tb`, `VSS-SA` and `HALNS` have running times up to 100 seconds (with zero gap) while our simple algorithms `SBAA` and `GRA` take always less than 1 second (with gaps below 5% and 7%, resp.); instead, `GA` has running time below 4 seconds (while exhibiting gaps below 5%); finally, for subset SC-3, `VSS-Tb`, `VSS-SA` and `HALNS` have running times up to 300 seconds, while `SBAA` and `GRA` run always for less than 1 second and their gaps are below 1.78% and 3.42%, resp.; `GA` has running time up to 11 seconds with gap below 2.00%; Note that for subsets SC-1, SC-2 and SC-3, `4AA` has gap mostly below 15.00% with high running time.
- for LARGE-CASE instances, algorithm `SBAA` outperforms the literature in 4 out of 10 instances of subset LC-1, in terms of quality of solution, while having running time at most 8 seconds; in the remaining 6 instances of subset LC-1, method `SBAA` is competitive w.r.t. the state-of-the-art, in terms of quality of solution, while achieving a gap that is always below 5.40%; for subset LC-2, algorithms `VSS-Tb`, `VSS-SA` and `HALNS` have large running times (up to 3900 seconds) while `SBAA` and `GRA` are the best performing in terms of time, with executions lasting at most 32 seconds (which is at least two orders of magnitude faster than `VSS-Tb`, `VSS-SA` and `HALNS`); on top of that, the gap obtained by `SBAA` and `GRA` remain below 15% and 20% respectively in most cases, and the gap of `GA` is mostly below 15% (with running time up to 132 seconds); finally, for subset LC-3, algorithms `VSS-Tb`, `VSS-SA` and `HALNS` have huge running times (up to 16000 and 1700 seconds, resp., for `VSS-Tb` and `VSS-SA`, while `HALNS` runs for up to 7000 seconds) while `SBAA` and `GRA` run for at most 56 seconds (meaning that `SBAA` is at least two orders of magnitude faster than `VSS-Tb`, `VSS-SA` and `HALNS`) with gap mostly below 8%; similarly, `GRA` has running time up to 56 seconds with the gap mostly below 15%, and `GA` takes up to 437 seconds to yield gaps that are mostly below 5%; to summarize, the results for LARGE-CASE inputs suggest that our very simple algorithms outperform the literature by far in terms of time (at least an order of magnitude) while having a good gap, and hence they can be considered more practical.

Table 4 Excerpt of the results of experiment COMPARISON for subsets LC-2 (top) and LC-3 (bottom).

Subset lc-2																								
Instance	GRA			SBAA			GA			BiFFf [25]			BiFFs [25]			VSS-Tb [6]			VSS-SA [6]			HALNS [19]		
	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g
81-337-8-200-400	1718	16	15.82	1911	16	6.36	1976	50	3.57	2032	155	0.44	2032	1236	0.44	2039	1286	0.10	**2041**	763	0.00	2040	1167	0.05
81-337-8-200-400	1757	8	13.91	1911	16	6.36	1976	50	3.57	2032	155	0.44	2032	1236	0.44	2039	1286	0.10	**2041**	763	0.00	2040	1167	0.05
82-385-8-200-400	1830	8	11.50	1850	15	10.54	2022	55	2.22	2064	1216	0.19	2065	5641	0.15	2066	2121	0.10	**2068**	1078	0.00	2066	1983	0.10
83-433-8-200-400	1939	16	7.57	2024	10	3.52	2058	77	1.90	2096	574	0.10	2097	3173	0.05	2097	2724	0.05	**2098**	1649	0.00	2098	1562	0.00
84-481-8-200-400	1876	16	11.96	2014	21	5.49	2075	106	2.62	2127	112	0.19	2127	103	0.19	2130	3317	0.05	**2131**	1627	0.00	2130	1116	0.05
85-529-8-200-400	1708	20	21.03	2023	24	6.47	2109	111	2.77	2154	1038	0.32	2155	7914	0.37	2162	3764	0.05	**2163**	2252	0.00	2161	1339	0.09
86-577-8-200-400	1991	24	9.66	2116	19	3.99	2171	132	1.54	2204	7385	0.05	2204	1658	0.05	**2205**	6426	0.00	**2204**	2289	0.05	**2205**	2078	0.00
87-361-8-200-400	1804	8	12.59	1939	16	6.05	1978	55	4.16	2063	1964	0.05	2063	3762	0.05	2063	889	0.05	**2064**	886	0.00	**2064**	1329	0.00
88-433-8-200-400	1800	16	13.21	1814	22	12.53	2011	90	3.03	2068	683	0.29	2070	3589	0.19	2072	1974	0.10	**2074**	2000	0.00	2072	784	0.10
89-505-8-200-400	1718	16	18.88	1942	32	8.30	2051	126	3.03	2115	11022	0.14	2115	17380	0.14	2116	2693	0.09	**2118**	2515	0.00	2115	2849	0.14
90-577-8-200-400	1961	24	9.75	2077	24	4.41	2139	129	1.56	2168	2168	0.23	2172	15678	0.05	2171	3326	0.09	**2173**	3197	0.00	**2173**	2901	0.00

Subset lc-3																								
Instance	GRA			SBAA			GA			BiFFf [25]			BiFFs [25]			VSS-Tb [6]			VSS-SA [6]			HALNS [19]		
	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g	p	t	g
21-337-8-345-720	2858	19	9.52	2933	26	7.15	2975	135	5.82	**3159**	927	0.00	**3159**	1745	0.00	3158	1462	0.03	**3159**	1576	0.00	**3159**	824	0.00
22-385-8-350-713	2911	24	11.54	3033	26	7.83	3183	161	3.28	3290	144	0.03	**3291**	583	0.00	**3291**	2850	0.03	**3291**	3018	0.00	**3291**	1401	0.00
23-433-8-330-675	2970	26	6.89	3058	17	4.13	3143	188	1.47	**3190**	455	0.00	**3190**	461	0.00	**3190**	4288	0.00	**3190**	3143	0.00	**3190**	616	0.00
24-481-8-335-713	3038	34	7.63	3144	29	4.40	3218	273	2.15	**3289**	451	0.00	**3289**	426	0.00	**3289**	283	0.00	**3289**	314	0.00	**3289**	297	0.00
25-529-8-340-705	3078	41	10.36	3258	31	5.12	3368	295	1.92	3432	1953	0.06	**3434**	7418	0.00	**3434**	4145	0.00	**3434**	4396	0.00	**3434**	5964	0.00
26-577-8-365-683	3385	51	9.70	3538	33	3.70	3674	406	2.00	**3749**	997	0.00	**3749**	1007	0.00	3748	14090	0.03	3748	9342	0.03	**3749**	1430	0.00
27-361-8-335-668	2738	24	12.15	3916	22	6.44	2947	131	5.45	3116	187	0.03	3116	189	0.03	**3117**	3090	0.00	**3117**	2248	0.00	**3117**	2046	0.00
28-433-8-350-675	2828	33	14.38	3106	40	5.96	3166	235	4.14	3301	1834	0.06	3302	3455	0.03	**3303**	5691	0.00	**3303**	5120	0.00	**3303**	3583	0.00
29-505-8-360-660	2794	45	20.82	3083	55	12.63	3303	373	6.40	3510	2282	0.54	3525	14499	0.11	3528	7370	0.03	**3529**	11119	0.00	3526	7134	0.09
30-577-8-375-675	3233	56	14.53	3518	48	7.00	3698	437	2.24	3781	2784	0.05	**3783**	11663	0.00	3781	10154	0.05	**3783**	9865	0.00	**3783**	7147	0.00

5.2 scalability Experiment

Here we evaluate how the running times of SBAA, GRA and GA change as the input size increases.

Input Data. We generate input instances whose size is far larger than that of any of the available benchmark inputs, with up to 15 500 nodes, by manipulating instance BRUSSELS2, used by Arnold et al. [3], for a version of the capacitated vehicle routing problem where, given a graph with edge lengths and a set of vehicles with limited capacity, the goal is to cover all the nodes with minimum total length and in such a way that each vehicle respects the capacity constraint. We sample uniformly at random 10 subgraphs from BRUSSELS2, each having from 500 nodes to 15 500 nodes with steps of 1 000 nodes. For each subgraph, we consider $K \in \{2, 4, \dots, 10\}$ and $B \in \{200, 400, 600\}$ while the capacity is fixed to $C = 200$. Note that in the original instance, BRUSSELS2, the capacity of each vehicle is set to 150. Similarly to the other benchmark instances, the prize is set to $\pi(v) = (h + 0.5)r(v)$, for each node v with size $r(v)$, with h randomly uniformly chosen within $[0, 1]$.

Executed Tests. For each subgraph and combination of K and B, we ran heuristics `SBAA`, `GRA`, and `GA`, and measure achieved prize and running time. We omit heuristic `4AA` from this part of the study since its running time is observed to be high even for not so large input combinations (see Section 5.1). The results of this experiment are summarized in Figures 1– 3: for each heuristic and for each considered value of B, we report measured solution quality and running time, averaged over all K.

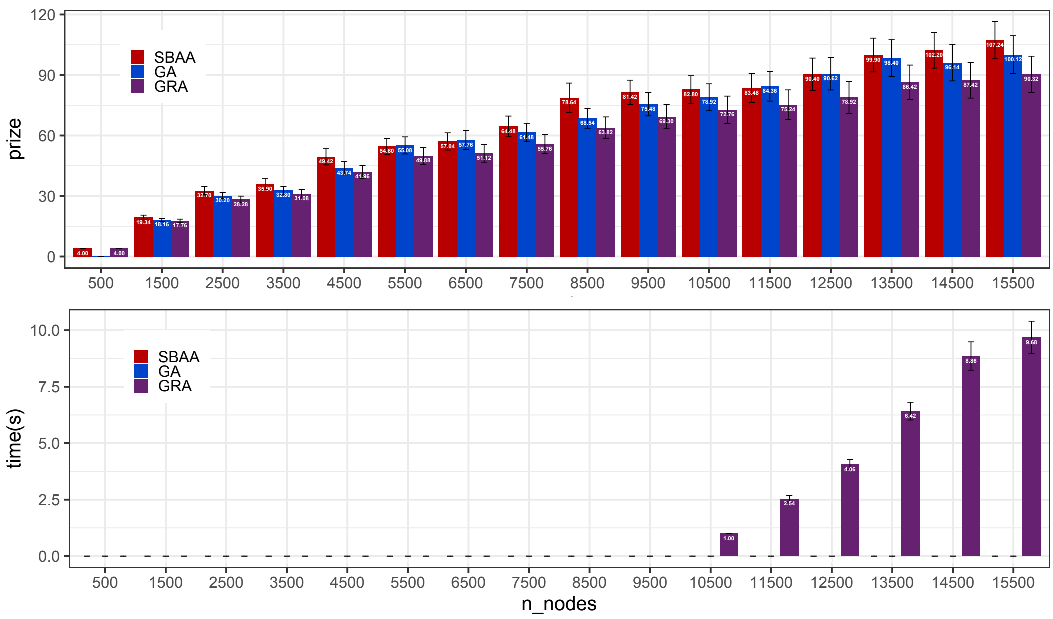

Figure 1 Results of the SCALABILITY experiment for $B = 200$.

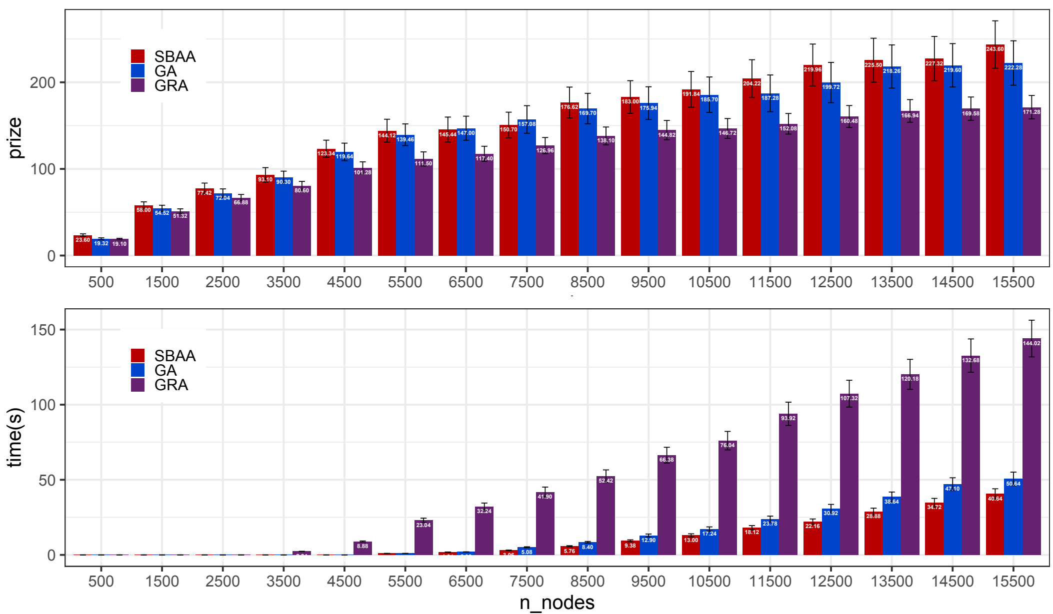

Figure 2 Results of the SCALABILITY experiment for $B = 400$.

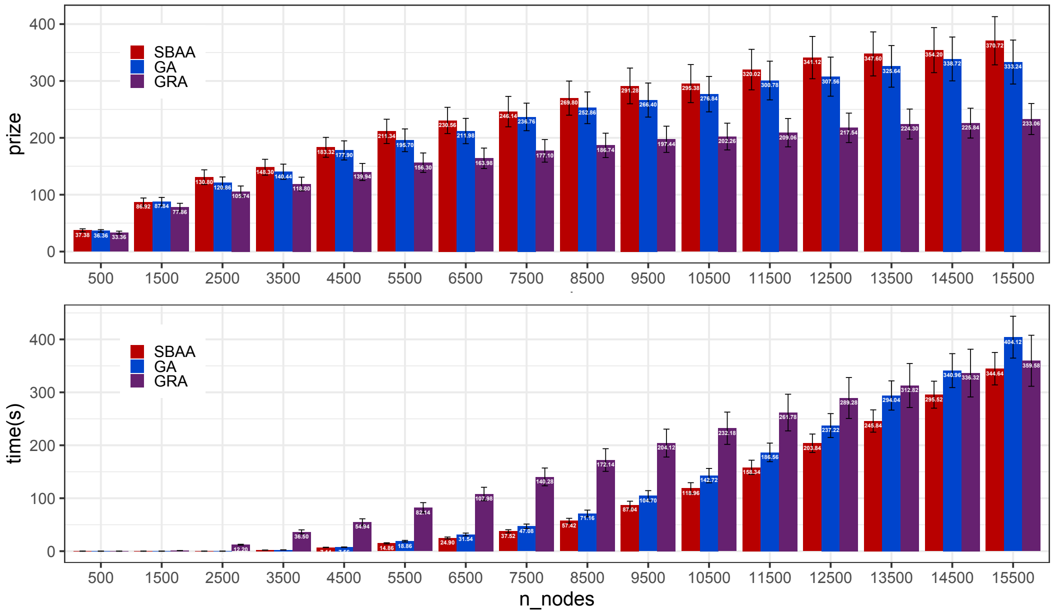

Figure 3 Results of the SCALABILITY experiment for $B = 600$.

Analysis. Our experimental data highlight the following general behavior. When $B = 200$ (see Figure 1), `GA` and `SBAA` are extremely fast, with running times smaller than 1 second even when the number of nodes n approaches 15 500; the running time of `GRA` is higher than the first two heuristics and grows faster as n increases, but remains below 10 seconds even for the largest case of $n = 15\,500$. The latter value is far below the average running times of algorithms tested in Section 5.1 for smaller graphs. Moreover, despite the low running time, `GA` and `SBAA` on average outperform `GRA` also w.r.t. achieved prize. When the budget is increased to 400 (see Figure 2) the observed trends are similar, with the average running time of `GA` and `SBAA` being below 1 second until n is less than 6 500 and remaining below 50 seconds. `GA` and `SBAA` outperform `GRA` w.r.t. both execution time and prize while `SBAA` outperforms `GA`, by small factors, w.r.t. both execution time and prize. Finally, when B is further increased to 600 (Figure 3), the trend in terms of solution quality appears not to be affected while the running time of all considered heuristics significantly increases, with the difference between `SBAA`, `GA` and `GRA` that seems to decrease as n approaches the largest value of 15 000. In general, our experiments suggest that the running time of `SBAA` and `GA` tends to grow approximately linearly with the input size and highlight that, on the the largest instance, `SBAA` and `GA` take below one minute on average, whereas previous algorithms are not able to handle such large input graphs. Note that however, we use the dynamic programming for the knapsack problem in both `SBAA` and `GA`, the capacity C in our instances is less than the number of nodes.

References

1 Claudia Archetti, Nicola Bianchessi, and Maria Grazia Speranza. Optimal solutions for routing problems with profits. *Discret. Appl. Math.*, 161(4-5):547–557, 2013.

2 Claudia Archetti, Dominique Feillet, Alain Hertz, and Maria Grazia Speranza. The capacitated team orienteering and profitable tour problems. *J. Oper. Res. Soc.*, 60(6):831–842, 2009.

3 Florian Arnold, Michel Gendreau, and Kenneth Sörensen. Efficiently solving very large-scale routing problems. *Comput. Oper. Res.*, 107:32–42, 2019.

4 Sankalp Arora and Sebastian A. Scherer. Randomized algorithm for informative path planning with budget constraints. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 4997–5004. IEEE, 2017.

5 Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 166–174. ACM, 2004.

6 Asma Ben-Said, Racha El-Hajj, and Aziz Moukrim. A variable space search heuristic for the capacitated team orienteering problem. *J. Heuristics*, 25(2):273–303, 2019.

7 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM J. Comput.*, 37(2):653–670, 2007.

8 Adrian Bock and Laura Sanità. The capacitated orienteering problem. *Discret. Appl. Math.*, 195:31–42, 2015.

9 Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms*, 8(3):23:1–23:27, 2012.

10 Ke Chen and Sariel Har-Peled. The euclidean orienteering problem revisited. *SIAM J. Comput.*, 38(1):385–397, 2008.

11 Nicos Christofides. The vehicle routing problem. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, 10(V1):55–70, 1976.

12 Gianlorenzo D'Angelo, Esmaeil Delfaraz, and Hugo Gilbert. Budgeted out-tree maximization with submodular prizes. In Sang Won Bae and Heejin Park, editors, *33rd International Symposium on Algorithms and Computation, ISAAC 2022, December 19-21, 2022, Seoul, Korea*, volume 248 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

13 Mattia D'Emidio, Esmaeil Delfaraz, Gabriele Di Stefano, Giannantonio Frittella, and Edgardo Vittoria. Route planning algorithms for fleets of connected vehicles: State of the art, implementation, and deployment. *Applied Sciences*, 14(7), 2024. `doi:10.3390/app14072884`.

14 Zachary Friggstad, Sreenivas Gollapudi, Kostas Kollias, Tamás Sarlós, Chaitanya Swamy, and Andrew Tomkins. Orienteering algorithms for generating travel itineraries. In Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek, editors, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 180–188. ACM, 2018.

15 Zachary Friggstad and Chaitanya Swamy. Compact, provably-good lps for orienteering and regret-bounded vehicle routing. In Friedrich Eisenbrand and Jochen Könemann, editors, *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, volume 10328 of *Lecture Notes in Computer Science*, pages 199–211. Springer, 2017.

16 Aldy Gunawan, Kien Ming Ng, Vincent F Yu, Gordy Adiprasetyo, and Hoong Chuin Lau. The capacitated team orienteering problem. *Proceedings of the 9th International Conference on Industrial Engineering and Operations ManagementBangkok, Thailand, March 5-7, 2019*, pages 1630–1638, 2019.

17 Aldy Gunawan, Jiahui Zhu, and Kien Ming NG. The capacitated team orienteering problem: a hybrid simulated annealing and iterated local search approach. In *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT*, volume 2, 2021.

18 Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Running errands in time: Approximation algorithms for stochastic orienteering. *Math. Oper. Res.*, 40(1):56–79, 2015.

19 Farouk Hammami. An efficient hybrid adaptive large neighborhood search method for the capacitated team orienteering problem. *Expert Systems with Applications*, 249:123561, 2024.

20 André Hottung and Kevin Tierney. Neural large neighborhood search for routing problems. *Artif. Intell.*, 313:103786, 2022.

21 Tung-Wei Kuo, Kate Ching-Ju Lin, and Ming-Jer Tsai. Maximizing submodular set function with connectivity constraint: Theory and application to networks. *IEEE/ACM Trans. Netw.*, 23(2):533–546, 2015.

22 Zhixing Luo, Brenda Cheang, Andrew Lim, and Wenbin Zhu. An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem. *Eur. J. Oper. Res.*, 229(3):673–682, 2013.

23 Alice Paul, Daniel Freund, Aaron M. Ferber, David B. Shmoys, and David P. Williamson. Budgeted prize-collecting traveling salesman and minimum spanning tree problems. *Math. Oper. Res.*, 45(2):576–590, 2020.

24 Sandro Pirkwieser and Günther R. Raidl. Multilevel variable neighborhood search for periodic routing problems. In Peter I. Cowling and Peter Merz, editors, *Evolutionary Computation in Combinatorial Optimization, 10th European Conference, EvoCOP 2010, Istanbul, Turkey, April 7-9, 2010. Proceedings*, volume 6022 of *Lecture Notes in Computer Science*, pages 226–238. Springer, 2010.

25 Christos D. Tarantilis, Foteini Stavropoulou, and Panagiotis P. Repoussis. The capacitated team orienteering problem: A bi-level filter-and-fan method. *Eur. J. Oper. Res.*, 224(1):65–78, 2013.

26 Dimitra Trachanatzi, Manousos Rigakis, Andromachi Taxidou, Magdalene Marinaki, Yannis Marinakis, and Nikolaos F. Matsatsinis. A novel solution encoding in the differential evolution algorithm for optimizing tourist trip design problems. In Nikolaos F. Matsatsinis, Yannis Marinakis, and Panos M. Pardalos, editors, *Learning and Intelligent Optimization - 13th International Conference, LION 13, Chania, Crete, Greece, May 27-31, 2019, Revised Selected Papers*, volume 11968 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2019.

27 Vijay V Vazirani. *Approximation algorithms*, volume 1. Springer, 2001.

28 Wenzheng Xu, Weifa Liang, Zichuan Xu, Jian Peng, Dezhong Peng, Tang Liu, Xiaohua Jia, and Sajal K. Das. Approximation algorithms for the generalized team orienteering problem and its applications. *IEEE/ACM Trans. Netw.*, 29(1):176–189, 2021.

29 Wenzheng Xu, Chengxi Wang, Hongbin Xie, Weifa Liang, Haipeng Dai, Zichuan Xu, Ziming Wang, Bing Guo, and Sajal K Das. Reward maximization for disaster zone monitoring with heterogeneous uavs. *IEEE/ACM Transactions on Networking*, 2023.