Gotta (efficiently) catch them all: Pokémon GO meets Orienteering Problems

Eduardo Álvarez-Miranda*[1], Martin Luipersbeck†[2], and Markus Sinnl‡[2]

[1]*DIE, University of Talca, Curico, Chile.*
[2]*ISOR, University of Vienna, Vienna, Austria.*

Abstract

In this paper a new routing problem referred to as the *Generalized Clustered Orienteering Problem (GCOP)* is studied. The problem is motivated by the mobile phone game *Pokémon GO*, an augmented reality game for mobile devices holding a record-breaking reception: within the first month of its release, more than 100 million users have installed the game on their device. The game's immense popularity has spawned several side businesses, including taxi-tours visiting locations where the game can be played, as well as companies offering to play the game for users during times when they cannot. The GCOP applies – but is not limited to – these business cases.

Additional applications arise naturally in typical operative transportation problems which necessitate the availability of tours that are both time-effective and profitable. The studied problem combines aspects from several well-known routing problems, including the Traveling Salesman Problem (TSP), the Generalized TSP, and the Orienteering Problem.

In order to solve the GCOP to optimality, a polynomial-sized Mixed-Integer Linear Programming (MIP) formulation and an exponential-sized MIP formulation are presented. Based on these formulations, branch-and-bound (B&B) and branch-and-cut (B&C) algorithms are developed. The framework is further enhanced with valid inequalities, a lifting procedure for strengthening inequalities, as well as initialization and primal heuristics.

The computational performance of the proposed approaches is assessed in an extensive computational study, using real-world instances that combine crowd-sourced data associated with the Pokémon GO game with street maps of three European cities, as well as instances derived from the TSPLIB testbed.

1 Introduction and Motivation

The industry of mobile video games is a thriving sector, with revenues expected to be nearly USD 100 billion in 2016, surpassing revenues from more traditional gaming platforms such as consoles and PCs [33]. *Augmented reality* games are a growing subcategory in this area. In this type of game, the players are required to perform tasks in the real-world in order to gain points or move to the next game level. The most prominent example is Pokémon GO; however, many games of the same type exist [8]. Pokémon GO was released in July 2016 in several zones of the world, holding a record-breaking reception from the mobile phone users; within the first month of its release, more than 100 millions of users installed the game on their devices [see 3], leading to more than USD 250 millions in revenues [see 6]. Roughly speaking, the mobile device of a player (or *trainer*) is used to capture, battle, and train virtual creatures, known as *Pokémon*, who appear in the player's mobile in augmented reality, i.e., as if they were in the *same* real-world location as the player. Among the player's main goals is the collection of one member from each Pokémon species, as

*ealvarez@utalca.cl
†martin.luipersbeck@univie.ac.at
‡markus.sinnl@univie.ac.at

well as to catch as many Pokémon as possible. These tasks can be performed by wandering around a city, aiming to encounter and subsequently catch Pokémon.

The enormous success of the game has allowed the growth of many side business and has enhanced other industries as well; for instance, in cities such as Edinburgh, taxi drivers offer tourists trips not only to visit the main sights of the city, but also to the locations of valuable Pokémon [see,e.g., 5]; there are organized meet-ups to catch Pokémon together with a large group people [9]; moreover, companies have offered the service to play the game for their customers in order to assist them at enlarging their Pokémon collections [10]. Note that due to crowd-sourcing activities performed by the game's users, there exist several online maps providing real-world locations at which various Pokémon species have been sighted [see, e.g., 2].

In this paper, we show how to model the main task of the Pokémon GO game, i.e., traveling between locations with the aim of encountering Pokémon, which is also the core of the three real-world business applications mentioned above, as an optimization problem. The result is a routing problem that combines features of the Orienteering Problem (OP) [see 31, 45] and the Generalized (or Clustered) Traveling Salesman Problem (GTSP) [41]. This problem will be denoted as *Generalized Clustered Orienteering Problem (GCOP)*.

In the OP, we are given a root node and the goal is to find a tour which maximizes the (weighted) number of visited locations, while a budget-constraint for the length of the tour needs to be respected. In the GTSP, nodes are divided into clusters and the goal is to find a minimum distance tour that visits at least one element of each cluster.

The *Generalized Clustered Orienteering Problem (GCOP)* is defined as follows: Besides the typical traveling distances associated with the transportation links, we also have *prizes* or revenues associated with the nodes, i.e., a reward that is *collected* if a node is visited. Additionally, K node subsets (*clusters*) are given and we are given a budget B for the length of the tour. The optimization task addressed by the GCOP is to find an unrooted tour that maximizes the total collected prize while ensuring that (i) at least one node of each cluster is visited, and (ii) the total distance of the tour does not exceed the budget B.

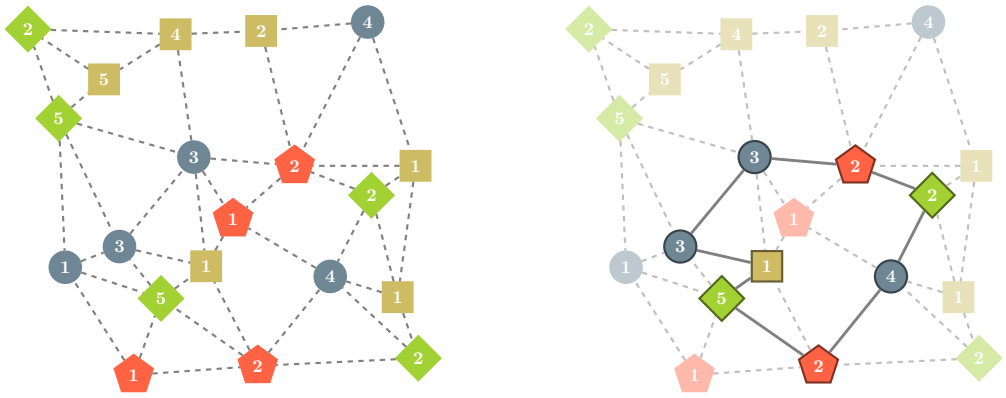
Note that considering the context of the addressed application, in this work we will focus on the case where all clusters are disjoint as in the GTSP, but do not require them to form a partition of V, i.e., there may be some nodes, which are not in any cluster. These nodes may be visited as intermediary nodes to shorten the length of the tour or to increase the revenue collected by the tour. However, in our problem definition we do not incorporate these assumptions, as all of the methods presented in this work are also valid in the more general context.

The GCOP is motivated by the following aspects of the Pokémon GO game: A player aims to complete his/her Pokémon collection, thus he/she likely wants to plan a Pokémon-catching tour in such a way that for some Pokémon species of his/her choosing, at least one location where this species is occurring, is visited, while also maximizing the total number of visited Pokémon locations. Moreover, a player is likely to be constrained with respect to the effort that can be spent for hunting Pokémon. Such a limitation might be associated with the amount of time that the user is willing to play, the autonomy of the device battery, or a maximum quota of data traffic associated with the user's mobile contract.

Finally, note that we do not impose a fixed root node (as is the case in the OP), since an important aspect of the game is that certain mechanisms only work if the player travels at walking speed. Thus, our goal is also to find some starting location in the city, which allows the most efficient tour with respect to collecting Pokémon. This point is further motivated by the fact that, sometimes, not a single person, but a group of people wants to walk together, so there is no clear location, e.g., the home of the player, which can be taken as root node. Note that the GCOP, however, allows for taking into account a fixed root node, since this can be implicitly imposed by defining a cluster of size one. It follows that the OP is a special case of the GCOP. As the OP is NP-hard, NP-hardness of the GCOP follows.

In Figure 1(a) an instance of the GCOP with disjoint clusters is shown; clusters are denoted by different nodes with particular color and shape, while prizes are indicated within each node. A solution for the problem is shown in Figure 1(b); as can be seen, nodes of each cluster are visited by a partial tour of the instance.

Aside from the motivating example given above, the GCOP is suitable in transportation settings where customers (retailers) can be divided into clusters or classes (e.g., shop locations, type of client, geographical



(a) Instance of the GCOP (b) Solution of the GCOP

Figure 1: An exemplary instance and a feasible solution for the GCOP. Prizes are given as numbers of the nodes; all nodes of a color/shape form a cluster. In this example, all the travel distances associated to the edges are assumed to be one, and the budget B for the length of the tour is eight.

area, end user stratum, etc.) and different revenues are associated to each of them. In such a setting, it seems natural to seek for distance-effective routes enabling the collection of as much revenue as possible while ensuring presence within each of the customer classes. Moreover, the GCOP might appear in applications where a manufacturer wants to distribute its production to a set of retailers that have not only one, but many shops or selling points (the set of locations of these shops corresponds to the clusters). The manufacturer assumes that if a single selling point of a retailer is served, then the retailer will thereafter distribute the product among the remaining shops. However, if the manufacturer transportation policy allows it, it is also desirable to visit other selling points of the same retailer; this could make his relation with such retailer more beneficial (as it saves the retailer transportation cost). In this sense, the prize of the selling point embodies, for instance, its relative importance with respect to the other shops of the same retailer.

Literature Review on Related Problems As said before, the GCOP combines the Orienteering Problem (OP), and the Generalized TSP (GTSP).

The first problem, the OP, was proposed in the early eighties [see 44, for one of the earliest references to the problem]. The problem is motivated by the family of sports with the same name [see 17, and the references therein]. Generally speaking, in the orienteering problem a competitor starts from an specific obligatory location (root) and tries to visit as many *stations* as possible while ensuring to return to the starting point within a given time bound. Each station has a certain prize so the goal of the competitor is to maximize the total collected prize. As pointed out in [45], the OP can be regarded as a combination between the Knapsack Problem and the TSP. Exact algorithms for the OP have been proposed, for instance, in [24, 28, 29]; likewise, heuristics schemes have been investigated in, e.g., [30, 40, 43]. The reader is referred to [31] and [45] for thorough reviews on the OP and its variants. Note that, aside of course from the cluster-requirements, a difference between the OP and the GCOP is that in the latter there is no predefined starting point; this adds an additional level of generality to the GCOP. Moreover, the GCOP differs from the Clustered OP, recently proposed in [11], since it is not required to visit *all* nodes of a cluster to gain its prize, and there is no fixed starting point.

The second problem, the GTSP, was first proposed back in the late sixties [41]. In the GTSP, nodes are partitioned into clusters and the goal is to find a minimum distance tour that visits at least one element from each cluster. Exact approaches have been proposed for the symmetric counterpart of the problem in [27] and [35], and also for the asymmetric counterpart [36]. Applications of the GTSP are discussed in [34].

Another related problem is the Prize-Collecting TSP, which was originally proposed by [13]; it corresponds to finding a tour that minimizes the sum of the travel costs and the non-collected prizes, while

visiting enough cities to collect a prescribed amount of the prizes (a *quota*). The author studied structural properties of the corresponding Mixed Integer Programming (MIP) polytope, and provided several families of facet-defining inequalities for it. From the algorithmic point of view, the problem has been approached by approximation algorithms [see, e.g., 12, 16], relaxations and bounding procedures [see 21, 26], heuristic schemes [see, e.g., 22], and exact algorithms [see 15].

Our Contribution and Paper Outline Although the GCOP shares common aspects with other problems, the presence of clusters and the lack of an obligatory starting node make the problem much more general than the previously proposed models. The main contribution of this paper is the formulation of the GCOP using two MIP models and the design of corresponding branch-and-bound (B&B) and branch-and-cut (B&C) algorithms based on these two MIP formulations. Concretely, we provide a polynomial-sized MIP formulation and an exponential-sized MIP formulation. The framework is enhanced with valid inequalities, and initialization and construction heuristics. The heuristics are employed in both algorithms.

By using instances based on crowd-sourced data for the Pokémon GO game in Vienna, Budapest and London, as well as instances generated from the TSPLIB, the state-of-the-art TSP testbed, we show the effectiveness of our proposed schemes to solve large-scale, real-world-based instances.

The paper is organized as follows. A formal definition of the problem and two MIP formulations are given in Section 2. Section 3 lists implementation details of our solution framework. In Section 4 we report computational results attained by the proposed algorithms on synthetic and real-world instances. Finally, conclusions and paths for future research are drawn in Section 5.

2 MIP Formulations for the GCOP

2.1 Problem Definition and Notation

Let $G = (V, E)$ be an undirected transportation network with node set V ($|V| = n$) and edge set E ($|E| = m$). Moreover, let $\mathcal{C} = \{C_1, \ldots, C_K\}$ denote a (not necessarily disjoint) collection of K node subsets $C_k \subset V$ referred to as *clusters*. The prize function $\mathbf{p} : V \to \mathbb{R}^n_{\geq 0}$ defines the prize collected if node $i \in V$ is visited. The distance function $\mathbf{d} : E \to \mathbb{R}^m_{\geq 0}$, associates a travel distance to each edge $e \in E$. The GCOP is defined as the problem of finding a *tour* $T = (E_T, V_T) \subset G$ such that (i) at least one element of each cluster is visited, (ii) its traveled distance does not exceed a given bound B (i.e., $\sum_{e \in E_T} d_e \leq B$), (iii) and the amount of collected prizes, $\sum_{i \in V_T} p_i$, is maximized.

For modeling the GCOP as MIP problem, the following notation is used. Let $\mathbf{y} \in \{0,1\}^n$ be a vector of binary variables such that $y_i = 1$ if node $i \in V$ is visited by the solution, and $y_i = 0$ otherwise; complementary, let $\mathbf{x} \in \{0,1\}^m$ be a vector of binary variables such that $x_e = 1$ if edge $e \in E$ is used in the solution, and $x_e = 0$ otherwise. A vector $(\mathbf{y}, \mathbf{x}) \in \{0,1\}^{n+m}$ defines a feasible solution if the following constraints are fulfilled:

$$\sum_{e \in \delta(i)} x_e = 2y_i, \qquad \forall\, i \in V \qquad \text{(DEGREE)}$$

$$\sum_{i \in C_k} y_i \geq 1, \qquad \forall\, k = \{1, \ldots, K\} \qquad \text{(CLUSTER)}$$

$$\sum_{e \in E} d_e x_e \leq B \qquad \text{(BUDGET)}$$

$$(\mathbf{y}, \mathbf{x}) \text{ contains no subtour.} \qquad \text{(NO-SUBTOUR)}$$

Constraints (DEGREE) ensure that the number of solution edges incident to a node i is exactly two ($\sum_{e \in \delta(i)} x_e = 2$) if the node is visited ($y_i = 1$), or zero ($\sum_{e \in \delta(i)} x_e = 0$) if not ($y_i = 0$). The condition that at least one node of every cluster must be visited is modeled by (CLUSTER). Constraint (BUDGET) ensures that the tour satisfies the given budget. Finally, (NO-SUBTOUR) states in a generic way that the

solution is not allowed to contain subtours. Hence, with these constraints the GCOP can be modeled as

$$\mathbf{p}^* = \max \left\{ \sum_{i \in V} p_i y_i \middle| \text{(DEGREE)-(NO-SUBTOUR)}, (\mathbf{y}, \mathbf{x}) \in \{0,1\}^{n+m} \right\}. \tag{MIP}$$

There are several MIP alternatives to model (NO-SUBTOUR). In this paper we will consider two modeling alternatives; one based on the so-called Miller-Tucker-Zemlin (MTZ) constraints [see 38], and one based on (generalized) subtour elimination constraints (GSECS) [see 37]. On the one hand, a MTZ-based formulation has polynomial size, i.e., it can be directly given to a state-of-the-art black-box MIP solver to tackle it, but usually gives rather poor bounds. On the other hand, a formulation using GSECs has an exponential number of constraints and thus requires on-the-fly cut-separation, i.e., branch-and-cut (B&C), but usually gives much better bounds compared to a MTZ formulation. In the remainder of this section, these two alternatives are studied.

Regardless of the chosen formulation, we observe that constraints (CLUSTER) can be lifted to

$$\sum_{i \in C_k} y_i \geq 1 + \sum_{e:\{i,j\} \in E|_{i,j \in C_k}} x_e, \ \forall\, k = \{1, \ldots, K\}. \tag{CLUSTER+}$$

The correctness of these constraints follows from this observation: if an edge, say $e : \{i, j\} \in E$, is part of the tour ($x_e = 1$) and both endpoints belong to the same C_k cluster ($i, j \in C_k$), then at least two nodes in C_k are visited ($\sum_{i \in C_k} y_i \geq 2$). These lifted constraints are a special case of the GSECs which are discussed in §(2.2).

Finally, due to the budget-type of constraint (BUDGET), the family of so-called cover inequalities are valid for the (MIP) formulation. Let C be a set of edges whose total distance exceeds the bound B (i.e., $\sum_{e \in C} d_e > B$); clearly, the inequality

$$\sum_{e \in C} x_e \leq |C| - 1, \tag{COVER}$$

must hold. A strengthened variant of inequalities (COVER) can be formulated based on the fact that any feasible solution forms a tour. Given a tour $T = (V_T, E_T)$ whose length exceeds B, the so-called *cycle-cover inequalities*, given by

$$\sum_{e \in E_T} x_e \leq \sum_{i \in V_T} y_i - 1, \tag{CYCLECOVER}$$

have been derived for the OP by [27]. (CYCLECOVER) excludes the cycle in E_T and is valid since T is infeasible due to the budget constraint. In the case of the OP, it is assumed that T contains the root node. Clearly, these inequalities are also valid for the GCOP if at least one cluster is a subset of V_T. This condition can be relaxed via a prize-based argument; let $P(S) = \sum_{i \in S} p_i$ be the amount of revenue contained in $S \subseteq V$, and LB the objective value of the current incumbent solution. If $P(V \setminus V_T) \leq LB$, then a feasible subtour of strictly better value than LB cannot exist in $V \setminus V_T$ alone. The consequence is that any improving solution contains at least one node from V_T. Note that (CYCLECOVER) may cut off feasible solutions, but only those with objective value lesser or equal to the incumbent's, which is acceptable in a B&C context.

2.2 A Polynomial-Sized MIP Formulation for the GCOP

We first give a *compact*, i.e., polynomial-sized model extending the well-known MTZ constraints [see 38] to our unrooted, prize-collecting setting. Note that MTZ constraints are often used for modeling the TSP and related problems. As said before, such a compact formulation allows the direct use of any off-the-shelf MIP solver without the need of implementing cutting plane separation procedures. This may be preferable if a practitioner wants to solve the GCOP without too much effort. However, note that formulations using MTZ

constraints are usually (much) weaker than formulations using GSECs, and thus are typically effective only on instances of limited size [see, e.g., 14].

The proposed MTZ-based formulation is an extended formulation, i.e., besides $\mathbf{y}$ and $\mathbf{x}$, additional variables are required. First, let us consider the bi-directed counterpart of G, $G_A = (V, A)$, where $A = \{(i,j),(j,i) \mid \forall e : \{i,j\} \in E\}$; likewise, for a given $S \subseteq V$, let $\delta^+(S) = \{(i,j) \in A | i \in S, j \in V \setminus S\}$ (*outgoing* arcs from S) and $\delta^-(S) = \{(i,j) \in A | i \in V \setminus S, j \in S\}$ (*incoming* arcs to S). Since the tour must visit every cluster, one can state that the tour shall start from an arbitrary cluster, say C_1; hence, let $\mathbf{z} \in \{0,1\}^{|C_1|}$ be a vector of binary variables such that $z_i = 1$ if node $i \in C_1$ is the starting node of the tour, and $z_i = 0$ otherwise. Moreover, let $\mathbf{u} \in \mathbb{Z}^n_{\geq 0}$ be a vector of integer variables such that u_i indicates the position in which node i is visited by the tour, starting at zero. Nodes not in the solution, i.e., with $y_i = 0$ will also get position zero. Since the underlying graph is now bidirected, let $\mathbf{t} \in \{0,1\}^{|A|}$ be a vector of binary variables so that $t_{ij} = 1$ if arc $(i,j) \in A$ is part of the tour, and $t_{ij} = 0$ otherwise. The tour structure of a pair $(\mathbf{y}, \mathbf{x})$ can be ensured by

$$
\begin{aligned}
(MTZ) \quad \sum_{(k,i) \in \delta^-(i)} t_{ki} &= y_i, && \forall i \in V && \text{(MTZ.1)}\\
\sum_{(i,j) \in \delta^+(i)} t_{ij} &= y_i, && \forall i \in V && \text{(MTZ.2)}\\
z_i &\leq y_i, && \forall i \in C_1 && \text{(MTZ.3)}\\
\sum_{i \in V} z_i &= 1 && && \text{(MTZ.4)}\\
u_i &\leq (n-1)(1 - z_i), && \forall i \in C_1 && \text{(MTZ.6)}\\
u_i &\leq n-1, && \forall i \in V && \text{(MTZ.7)}\\
u_i - u_j + n t_{ij} &\leq n-1, && \forall (i,j) \in A \,|_{j \notin C_1} && \text{(MTZ.8)}\\
u_i - u_j + n(t_{ij} - z_j) &\leq n-1, && \forall (i,j) \in A \,|_{j \in C_1} && \text{(MTZ.9)}\\
x_e &= t_{ij} + t_{ji}, && \forall e : \{i,j\} \in E. && \text{(MTZ.10)}
\end{aligned}
$$

Constraints (MTZ.1) and (MTZ.2) are the so-called in- and out-degree constraints, they ensure that if a node, say $i \in V$, is part of the tour ($y_i = 1$), then one incoming and one outgoing arc must be part of the tour. Constraints (MTZ.3) model the fact that if node $i \in C_1$ is selected as starting node of the tour ($z_i = 1$), then it must be part of the tour ($y_i = 1$). The fact that only one node can serve as starting point of the tour is embodied by constraint (MTZ.4). Constraints (MTZ.6) ensure that the node selected as starting node has position zero. The adapted MTZ constraints correspond to (MTZ.7)-(MTZ.9); they ensure that variables $\mathbf{t}$ and $\mathbf{y}$ build a tour (whose order is embodied by $\mathbf{u}$) that starts from the node selected as starting node. Finally, constraints (MTZ.10) translate variables from the arc ($\mathbf{t}$) to the edge space ($\mathbf{x}$).

Additional Valid Inequalities for (MTZ) In the following, lifted variants of some of the inequalities of formulation (MTZ) are given. Besides, additional valid inequalities which strengthen the formulation (i.e., improve the value of its linear programming (LP)-relaxation) or cut off symmetric solutions, are presented.

First, we present a simple-lifting for (MTZ.7) inequalities. Validity follows from the fact that a node i can only have a position greater than zero if the node i is in the solution; moreover, it can only be in position $n-1$ if an arc from i to the starting node (which must be in C_1) is in the solution.

Theorem 1. *Inequalities*

$$
u_i \leq (n-2) y_i + \sum_{(i,j) \in A|_{j \in C_1}} t_{ij}, \ \forall \in V \qquad \text{(MTZ.7+)}
$$

are valid for (MTZ).

The validity of the next constraints relies on the following observation: A node $i \in V \setminus C_1$ can only be the second node in the tour (i.e., $u_i = 1$), if it is visited from a node in C_1, since the starting node is a node in C_1.

Theorem 2. *Inequalities*

$$2y_i \leq u_i + \sum_{(j,i)\in A|_{j\in C_1}} x_{ji},\ \forall i \in V \setminus C_1 \tag{MTZ.11}$$

are valid for (MTZ).

Next, we present variants of inequalities (MTZ.8) and (MTZ.9)

Theorem 3. *Inequalities*

$$u_i - u_j + nt_{ij} + (n-2)t_{ji} \leq (n-1)y_i,\ \forall (i,j) \in A\ |_{j\notin C_1} \tag{MTZ.8+}$$

and

$$u_i - u_j + n(t_{ij} - z_j) + (n-2)t_{ji} \leq (n-1)y_i,\ \forall (i,j) \in A\ |_{j\in C_1} \tag{MTZ.9+}$$

are valid for (MTZ).

Proof. The validity of the introduction of the $(n-2)t_{ji}$-term follows from the observation that in case arc (j,i) is taken in the solution, node j must be on the position strictly before i in the tour. Observe that a similar lifting has been proposed for the original MTZ constraints for the TSP in [23]. Regarding the right-hand side of the inequalities, if $y_i = 0$, i.e., node i does not occur in the solution, per design it gets assigned $u_i = 0$ and clearly all arcs associated with it must be zero.

□

The following set of constraints, (ASYM), helps to get rid of symmetric solutions, i.e., different feasible vectors $(\mathbf{y}, \mathbf{x}, \mathbf{t}, \mathbf{z}, \mathbf{u})$ which correspond to the same solution when projected back to the original space $(\mathbf{y}, \mathbf{x})$. This situation can occur because different choices of the starting node can give the same tour in the original space, just with a different ordering $\mathbf{u}$, as we have $u_i = 0$, if $z_i = 1$, i.e., i is chosen as starting node. The following constraints,

$$y_i + \sum_{j\in V|_{j>i}} z_j \leq 1, \quad \forall i \in C_1, \tag{ASYM}$$

ensure that of all nodes visited in C_1, the one with the smallest index is chosen as root node. Finally, the following (directed) GSECs of size two,

$$x_{ij} + x_{ji} \leq y_i,\ \forall (i,j) \in A, \tag{GSEC-2}$$

are also valid, since they prevent subtours of size 2.

2.3 An Exponential-Sized MIP Formulation for the GCOP

We now present a different alternative for modeling the tour-structure of the solution; although exponentially large, this formulation enables the design of more effective algorithmic strategies. For a given subset of nodes $S \subset V$, let $\delta(S) = \{\{i,j\} \in E | i \in S, j \in V \setminus S\}$, i.e., $\delta(S)$ corresponds to the *cut-set* associated to the *cut* S. The constraints

$$\sum_{e\in\delta(S)} x_e \geq 2\,(y_i + y_j - 1)\,, \qquad \forall\ i \in S,\ j \in V \setminus S,\ S \subset V\ |_{2\leq|S|\leq n-1}, \tag{GSEC}$$

known as *generalized subtour elimination constraints (GSECs)* ensure that a pair $(\mathbf{y}, \mathbf{x})$ defines a tour on G. These constraints guarantee that the cut-set $\delta(S)$ separating two visited nodes (i from S and j from $V \setminus S$) must be crossed at least twice. We denote the formulation resulting from modeling the generic constraints (NO-SUBTOUR) by (GSEC) as (CUT). Note that these constraints are equivalent to those used by [27] for the generalized TSP. Constraints (GSEC) are exponential in number, meaning that the formulation (CUT)

cannot be solved directly even for small-sized instances. This calls for a more advanced algorithmic strategy. A natural option is the dynamic management of the GSECs in a B&C framework, in which these constraints are separated on-the-fly [see 15, 24, 27, 28, 29, 35, 36, for implementations of this idea for related problems]. The separation of these, and similar, inequalities corresponds to the main component of our algorithmic framework (see Section 3).

Lifting Inequalities (GSEC) For a given $S \subseteq V$, let $\mu(S) = |\{k \mid C_k \subseteq S,\ \forall k = \{1,\dots,K\}\}|$ (the number of clusters *contained* in S). Clearly, if $\mu(S) \neq 0$, at least one node in the set S must be in the solution. Thus, if for a cut S, S and $V \setminus S$ have $\mu(S) \neq 0$ and $\mu(V \setminus S) \neq 0$ a least a node in S and $V \setminus S$ must be visited in any feasible solution, which leads to the following lifted GSECs (see [27])

$$\sum_{e \in \delta(S)} x_e \geq 2,\ \forall S \subset V\ |_{2 \leq |S| \leq n-2}\ \text{ so that } \mu(S) \neq 0 \text{ and } \mu(V \setminus S) \neq 0. \tag{GSEC.1}$$

Similar arguments for $\mu(S) = 0$ and $\mu(V \setminus S) \neq 0$ lead to the lifted GSECs

$$\sum_{e \in \delta(S)} x_e \geq 2y_i,\ \forall i \in S,\ \forall S \subset V\ |_{2 \leq |S| \leq n-2}\ \text{ so that } \mu(S) = 0 \text{ and } \mu(V \setminus S) \neq 0. \tag{GSEC.2}$$

The separation of inequalities (GSEC), (GSEC.1) and (GSEC.2) is discussed in the next section. In contrast to the GTSP, for the GCOP, constraints (GSEC) and (GSEC.2) can be lifted by a prize-based argument similar to the one used for inequalities (CYCLECOVER) in §2.1. In particular, the condition $\mu(S) \neq 0, S \subset V$ can be relaxed to $P(V \setminus S) \leq LB$, with LB being the objective value of the incumbent solution, since at least one node in S must be in any solution better than the incumbent due to $P(V \setminus S) \leq LB$. Let $E(S) = \{e : \{i,j\} \in E \mid i,j \in S\}$. Following [27], in computations we use

$$\sum_{e \in E(S)} x_e \leq \sum_{i \in S \setminus \{i\}} y_i - y_j + 1,$$
$$\forall i \in S,\ \forall j \in V \setminus S,\ \forall S \subset V\ |_{2 \leq |S| \leq n-2}\ ,$$
$$\sum_{e \in E(S)} x_e \leq \sum_{i \in S} y_i - 1,\ \forall S \subset V\ |_{2 \leq |S| \leq n-2}\ \text{ so that } \mu(S) \neq 0 \text{ and } \mu(V \setminus S) \neq 0,$$
$$\sum_{e \in E(S)} x_e \leq \sum_{i \in S \setminus \{i\}} y_i,$$
$$\forall i \in S,\ \forall S \subset V\ |_{2 \leq |S| \leq n-2}\ \text{ so that } \mu(S) = 0 \text{ and } \mu(V \setminus S) \neq 0.$$

as equivalent forms of (GSEC), (GSEC.1) and (GSEC.2), respectively, which have less nonzero coefficients.

3 An Algorithmic Framework for the GCOP

We now provide the main elements of the algorithmic framework designed for the GCOP. First we describe the separation procedures devised for the identification of violated GSECs (GSEC),(GSEC.1), (GSEC.2) and cycle-cover inequalities (CYCLECOVER). (Note that separation of inequalities (COVER) has been left to the general-purpose cutting-plane generation procedure of the used MIP-solver.) Afterwards, we outline the primal heuristic and the initialization heuristic.

GSEC Separation Let $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ be the LP solution obtained at some node of the B&C tree, and $\tilde{G} = (V, E, \tilde{\mathbf{x}})$ an edge-capacitated graph, where the capacity c_e of every $e \in E$ corresponds to $\tilde{x}_e$. The GSECs (GSEC), (GSEC.1) and (GSEC.2) can be separated exactly by using max-flow computations on $\tilde{G}$ (and slightly modified versions of $\tilde{G}$) [see 27, for details]. However, in preliminary computations, exact separation (for non-integer solutions) turned out to be too time consuming. Thus, following [27], our implementation handles the separation of inequalities (GSEC), (GSEC.1) and (GSEC.2) via a simple scheme that

Data: An optimal LP solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, best incumbent objective value LB.
Result: A set of violated inequalities Q.

$(G, \tilde{G}) \leftarrow$ `constructFlowGraphs`$(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$
$s \leftarrow$ `chooseRandomElement`$(\arg\max_{i \in V} \tilde{y}_i)$
foreach $\varepsilon \in \{0.1, 0.01, 0.001\}$ **do**
  foreach $i \in V \setminus \{s\}, 2(y_s + y_i - 1) > \varepsilon$ **do**
    $(f^*, (S, V \setminus S), (S', V \setminus S')) \leftarrow$ `maxflow`$(s, i, \tilde{G})$
    if $f^* + \varepsilon < 2(y_s + y_i - 1)$ **then**
      $Q \leftarrow Q \cup$ `chooseGSEC`$(S, V \setminus S, S', V \setminus S', LB)$
      $c_{s,i} \leftarrow 2$
  foreach $k \in \{1, \ldots, K\} \setminus C(s)$ **do**
    $(f^*, (S, V \setminus S), (S', V \setminus S')) \leftarrow$ `maxflow`$(s, t, \tilde{G}_k)$
    if $f^* + \varepsilon < 2y_s$ **then**
      $Q \leftarrow Q \cup$ `chooseGSEC`$(S, V \setminus S, S', V \setminus S', LB)$
      $c_{s,t} \leftarrow 2$
  if $Q \neq \emptyset$ **then break**
if $Q = \emptyset$ **then**
  $Q \leftarrow Q \cup$ `separateCycleCover`$(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$

Algorithm 1: Separation procedure.

is exact under the mild condition that $\max\{\tilde{y}_i : i \in V\} = 1$, and heuristic otherwise. Note that for integer $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, this procedure is exact (and thus correctness of our approach is ensured), since at least one node must be in any feasible solution due to the cluster constraints (CLUSTER). It requires at most $n+m-2$ maximum flow computations, as in this case it can be argued that in order to identify a most violated inequality, it suffices to consider a node $s \in V$ with $\tilde{y}_s = 1$ as fixed source.

For this claim, first note that for each violated inequality (GSEC.1), $C_k \subseteq S$ and $C_{k'} \subseteq V \setminus S$, the inequality (GSEC.2), $C_k \subseteq V \setminus S$ and $i = s$, must have the same degree of violation. A similar argument can be applied for each inequality (GSEC.2), $C_k \subseteq V \setminus S$ and $i \neq s$. Two cases need to be distinguished. If $s \in S$, then an inequality (GSEC.2) with $i = s$ exists for the same S that is at least as violated. Otherwise if $s \in V \setminus S$, there exists an inequality (GSEC) with $j = s$ with the same degree of violation.

Algorithm 1 shows the implemented separation procedure. Notation $C(i)$ is used to denote the set of cluster indices where node $i \in V$ is contained in. In Step 1, capacitated graphs are constructed from the LP solution. Since the support is usually much smaller than the complete graph, it pays off to consider for separation only the induced subgraph. Next, the source s is chosen at random from $\arg\max_{i \in V} \tilde{y}_i$.

To limit the number of inequalities added during each cutting plane iteration, we begin by only separating those cuts whose degree of violation exceeds a certain threshold $\epsilon \in \{0.1, 0.01, 0.001\}$. Initially setting $\epsilon = 0.1$, the threshold is lowered to the next smaller value only if no inequality with the given degree of violation exists.

The maximum-flow f^* is computed by procedure `maxflow` on $\tilde{G}$ between each pair $(s, j), j \in V \setminus \{s\}$. Clearly, all nodes j having $2(\tilde{y}_s + \tilde{y}_j - 1) < \varepsilon$ can be skipped. Our framework applies the preflow-push maximum-flow algorithm [18]. In addition to the maximum-flow f^*, the used implementation returns two associated minimum cuts, denoted by $(S, V \setminus S)$ and $(S', V \setminus S')$. If $f^* + \varepsilon < 2(\tilde{y}_s + \tilde{y}_j - 1)$, procedure `chooseGSEC` selects between (GSEC), (GSEC.1), and (GSEC.2) based on the contained clusters. It also takes into account the objective value LB of the incumbent solution to apply the prize-based lifting detailed in §2.3. If multiple cuts of the same type are identified, the one induced by the minimum cardinality subset S is added. A similar approach is followed for separation of cuts between s and each cluster that does not contain s. For each cluster $C_k \in \mathcal{C}$, the maximum flow is computed on a modified graph $\tilde{G}_k$ which contains an additional artificial node t as sink that is connected to each node in cluster C_k. Each time a violated cut is identified, the capacity of the edge connecting source and sink is set to 2, which prevents the same cut

from being separated multiple times. If no violated GSEC inequality has been identified, we try to identify violated inequalities (CYCLECOVER) as described in the next paragraph. The complexity of the total separation procedure (including separation of (CYCLECOVER)) is $O(n^4)$.

A special case of (GSEC.2) arises when $S = \{i, j\}$, in which the inequalities take the following form:

$$x_e \leq y_j \qquad \forall e \in \delta(j), j \in S.$$

In order to decrease the number of cutting plane iterations, the LP is initialized with a subset of these inequalities which are likely to be tight in an optimal solution. In our implementation, the inequalities associated to the five cheapest edges incident to each node are added at the beginning of the B&C procedure.

Separation of Inequalities (CYCLECOVER) Inequalities (CYCLECOVER) are separated heuristically as suggested in [28]. The procedure consists of two steps. First, a maximum spanning tree $T_{MST} \subset E$ is computed based on $\tilde{\mathbf{x}}$. Next, each edge $e \notin T_{MST}$ that induces a cycle when added to T_{MST} is considered. If the length of the cycle exceeds the budget, and if one of the two conditions detailed in §2.1 is satisfied, the inequality associated with the cycle is checked for violation.

Primal Heuristic The procedure consists of two steps: (i) *construction* and (ii) *refinement/repair*. In Step (i), given an LP solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, an attempt is made to construct a feasible tour on the subgraph induced by the node set $\tilde{V} = \{i \in V : \tilde{y}_i \geq 0.5\}$. Step (ii) is then executed on the whole graph.

For the construction step three complementary schemes have been devised:

1. **Farthest/nearest insertion.** Adaptations of the TSP construction heuristics to the GTSP have been proposed in [27]. Applying the same scheme to the GCOP is likely to yield a feasible solution as the set of visited nodes is minimal, but usually requires additional refinement as the amount of collected profit is low.

2. **Steiner-tree-based construction.** The given scheme generalizes the MST-heuristic for the TSP [39]. First, an extended auxiliary graph is constructed from G by adding for each cluster an artificial terminal node and connecting it to all its members. On the resulting graph a heuristic Steiner tree is computed using the *path-based construction heuristic* [42] and modified travel distances $\mathbf{d}'$ scaled by the current LP solution, i.e., $d'_e = d_e(1 - \tilde{x}_e), e \in E$. A tour is then defined by choosing a root node at random and performing a pre-order traversal of the tree. Clearly, the constructed tour visits all clusters and is likely to satisfy the budget constraint.

3. **TSP-based construction.** The procedure follows a more aggressive approach than the previous schemes by computing an optimal TSP tour visiting each node in $\tilde{V}$. For this task the state-of-the-art TSP solver Concorde [see 20] is employed, which finds an optimal tour on the considered benchmark instances and subgraphs within seconds. If $\tilde{V}$ is already a good approximation of the nodes visited in the optimal GCOP tour, this strategy is likely to yield a near-optimal solution. However, both the budget and cluster constraints may possibly be violated.

The shortcomings of the constructed tours are addressed in Step (ii), in which *insertion*, *deletion* and *swap* moves are applied exhaustively in order to attain a more profitable and (hopefully) feasible tour. Let $T = (V_T, E_T)$ be the (potentially infeasible) tour constructed in Step (i) and $D(V_T)$ its length. For each node $i \in V$, a score $\sigma_i = \min_{j \in V_T \setminus \{i\}} d_{ij}/p_i$ is computed. Then the following steps (a)-(e) are applied exhaustively (ties are broken arbitrarily):

(a) Insert $i^* = \arg\min_{i \in C_k} \sigma_i$ for each $C_k \cap V_T = \emptyset$.

(b) Insert $i^* = \arg\min_{i \in V \setminus V_T} \sigma_i$ while $D(V_T \cup \{i^*\}) \leq B$.

(c) Delete $i^* = \arg\max_{i \in V_T} \sigma_i$ while $D(V_T) > B$.

(d) Swap $i \in V_T$ and $j \notin V_T$ whenever $\sigma_i < \sigma_j$.

(e) Perform 2-opt and 3-opt.

The insertion of node $i \notin V_T$ is always performed via *cheapest insertion*, i.e., the resulting tour visits i between $j, k \in V_T$ such that $c_{ij} + c_{ik} - c_{jk}$ is minimal. The swap move simultaneously performs an insertion and deletion. Note that these moves alter the set V_T, but not the sequence in which nodes are visited. Although the collected profit is invariant with respect to the order of visits, finding a short tour may possibly enable further insertion or swap moves. For this purpose the *2-opt* and *3-opt* improvement heuristics for the TSP are applied to T.

Construction, and a subsequent refinement, is executed after each cutting plane iteration at the root node, and thereafter at every tenth (1000th for the MTZ formulation) node of the search tree. In preliminary experiments, neither construction scheme 1 nor 2 has been found to dominate the other, so in each call to the heuristic one of these two schemes is selected at random. Since construction scheme 3 has been found to be relatively time-consuming in some cases, it is only applied after every fifth cutting plane iteration. This decision is also supported by the fact that a near-optimal set $\tilde{V}$ is most likely only available during the final cutting plane iterations.

Initialization Heuristic In order to compute a high quality starting solution for initializing the B&C, ten iterations of the implemented construction and improvement heuristics are performed. In each iteration, perturbed distances $\tilde{d}_e, e \in E$, chosen uniformly from the interval $[d_e, 1.2\, d_e]$, are used for construction. Afterwards, *local branching* [25] is applied to the best solution found as an additional refinement procedure, in which the presented MIP framework is used to successively explore local neighborhoods defined by the current incumbent solution via a *local branching constraint*. Let $T = (V_T, E_T)$ be the current incumbent solution; the constraint

$$\sum_{i \notin V_T} y_i + \sum_{i \in V_T} (1 - y_i) \leq k, \tag{LB}$$

is added to the MIP model which restricts the search space to solutions that differ by at most k nodes from the incumbent. In our implementation, we initially set $k = 2$. If no improving solution is found for the resulting neighborhood within a time limit of 20 seconds, the neighborhood is successively enlarged in steps of two, up to $k = 6$. Each time an improving solution is found, the MIP solver is terminated and restarted with the neighborhood defined by the new solution and k reset to its initial size. The procedure is repeated until no improving solution can be found for $k = 6$ within the time limit. Note that solutions constructed by the primal heuristic during the B&C are still accepted even if they lie outside the current neighborhood.

Finally, the cutting planes separated in each local branching iteration are globally valid and are thus stored within a cut pool to be reused in each subsequent iteration. The cut pool is also used to initialize the final B&C.

4 Computational Results

The algorithmic scheme described in Section 3 has been implemented in C++ and compiled using GCC 4.9, CPLEX 12.6.3 has been used as a MIP solver, and OGDF [19] has been used for graph data structures. For solving TSP instances the release 03.12.19 of the Concorde TSP Solver [available in 20] has been used. All experiments have been performed single-threaded on an Intel Xeon CPU with 2.5 GHz. For each test run a time limit of one hour and memory limit of 8 GB has been set. Figures displaying maps have been rendered using Open Street Map [32] and the "Terrain" map tiles by Stamen Design [7]. Unless noted otherwise, relative optimality gaps listed in all tables are computed as $(UB - LB)/LB$ and given in percent. If a test run terminated due to the time or memory limit, the cell usually listing the running time contains *TL* or *ML* instead. For computing averages of running times, all runs that terminated due to reaching the memory limit before the time limit are counted using the full time limit.

Benchmark Instances Two sets of instances have been generated in order to assess the proposed framework's effectiveness:

- `TSPLIB`: The instances have been created based on an adaptation of the CLUSTERING procedure applied in [27], originally used to generate instances for the GTSP from the TSPLIB [4] testbed. Given an instance of the symmetric TSP, the procedure can be sketched as follows: First, $\lceil |V|/5 \rceil$ *centers* are chosen iteratively from V such that each newly selected center maximizes the distance to each already selected center. Next, each non-center is assigned to its closest center. We adapt this scheme by enforcing an additional lower bound of 2 on the cluster size in order to cover the most general case of the GCOP, in which no single starting point can be identified. In total 64 TSPLIB instances for the symmetric TSP with $|V| \leq 500$ have been selected for conversion to GCOP instances. For almost all of these instances the distance function is Euclidean.

  Note that for any given instance of the GCOP, the range of budget values B worth considering lies between the optimal objective values of the GTSP and TSP for said instance. For the GCOP, smaller values of B lead to infeasibility, while larger ones will not change the optimal objective value. We choose B for each instance as combination between the objective value of a feasible GTSP solution ($GTSP_{FEAS}$) visiting the node with the smallest index of each cluster, and the optimal TSP solution (TSP_{OPT}), i.e., $B = \alpha \cdot TSP_{OPT} + (1 - \alpha) \cdot GTSP_{FEAS}$, $\alpha \in \{0.25, 0.5, 0.75\}$. Two different schemes for assigning profits are considered which have been proposed in [28] for the OP: (i) uniform profits and (ii) pseudo-random profits between 1 and 100, computed as $p_i = 1 + 7213 \cdot i \mod 100, i \in V$.

- `Pokémon`: These instances have been derived from location data associated with the Pokémon GO game. The original data can be viewed online on the website *PokeMapper* [1], a global crowd-sourced map that allows its visitors to display and report locations at which certain Pokémon species have been sighted. We focus on the set of sightings present within the areas of three European cities: *Vienna*, *London* and *Budapest*. These cities have been selected based on the property that a large number of sightings was available within walking distance from the city center.

  Per city a complete graph has been generated in which nodes correspond to sighting locations and travel distances are set to the shortest path distances between locations based on the city's street network graph (as obtained from Open Street Map [32]). Note that in general, the resulting distance function does not satisfy the triangle inequality. In order to make the data suitable for a routing application, the location information displayed on the website had to be manually curated, as some sightings were located at coordinates inaccessible from the street network (e.g., within buildings). More specifically, all sighting locations have been mapped manually to suitably close nodes within the street network graphs. Finally, concerning the cluster sets necessary to define an instance of the GCOP, note that a natural clustering is already present within the data, since each sighting location is associated with a Pokémon type.

Table 1: Pokémon GO data sets.

			#locations per cluster			
city	*locations*	*clusters*	*min*	*avg*	*max*	TSP_{OPT}[km]
Budapest	468	100	1	5.09	29	75.134
London	437	102	1	4.28	35	104.920
Vienna	484	99	1	4.89	21	212.474

Table 1 summarizes the data obtained through this procedure for each city. Although the number of clusters and their average distribution may appear similar, the data differs significantly with respect to the spatial distribution of Pokémon sightings and the topological properties of the street network. Due to the same reasons as detailed for the `TSPLIB`, an additional post-processing step has been applied in which all clusters of size one have been eliminated. Nodes belonging to an eliminated cluster remain part of the instance, but their visit is optional. Column TSP_{OPT} lists the optimal travel distance in

kilometers for the TSP on the generated graphs. The magnitude of the computed distances makes apparent that visiting all clusters is likely to be infeasible for a Pokémon GO player traveling by foot.

Therefore, in our computational experiments we consider two scenarios: In the first we define budgets and profits as for the `TSPLIB` instances. In the second scenario, which aims to model a more realistic setting in terms of the Pokémon GO game, we consider variants of each city instance in which only a subset of all clusters is selected, while all remaining clusters are eliminated, i.e., visiting nodes in these clusters becomes optional. For the resulting instances, budget B has been chosen based on distances (in [km]) considered suitable for travel by foot or bike, i.e., $B \in \{10, 15, 20\}$. As for the `TSPLIB` instances, both the uniform and random profit scheme have been considered. However, due to the fact that in the game's setting the value of a Pokémon to a player is only determined by its species, random profits are assigned such that each node within a cluster is of equal profit.

4.1 Comparing (MTZ) and (CUT)

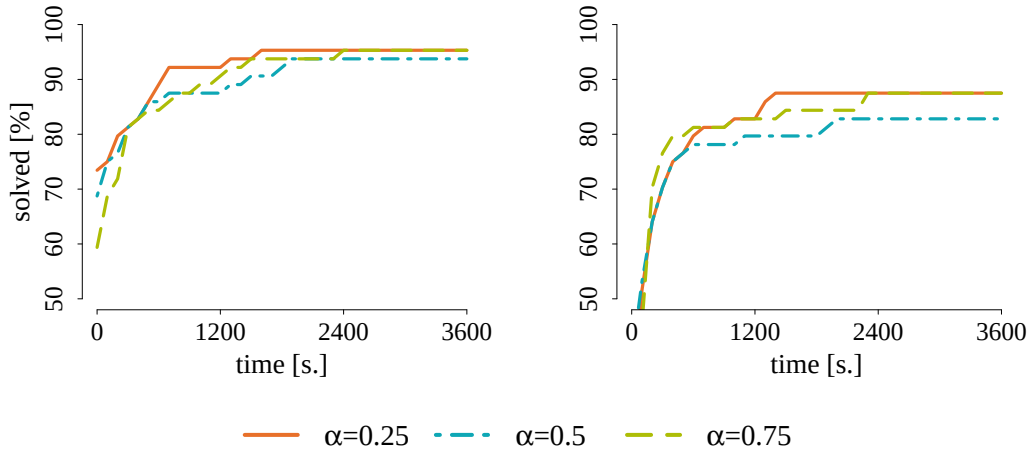
We begin by comparing the performance of formulations (MTZ) and (CUT) within the implemented algorithmic framework. Table 2 shows results on the `TSPLIB` instances having $|V| \leq 100$ for uniform and random profits. For this class of small-scale instances the performance is similar for each tested α, and thus only the results for $\alpha = 0.5$ are shown. Per formulation and instance, the number of B&B nodes (*#BBn.*), the relative optimality gap in percent (g[%]) and the running time until proven optimality (t[s.]) are reported.

Table 2: Performance comparison for formulations (MTZ) and (CUT) based on instances generated from the TSPLIB ($\alpha = 0.5$) having $|V| \leq 100$.

	uniform profits						***random profits***					
	(CUT)			(MTZ)			(CUT)			(MTZ)		
instance	*#BBn.*	g[%]	t[s.]	*#BBn.*	g[%]	t[s.]	*#BBn.*	g[%]	t[s.]	*#BBn.*	g[%]	t[s.]
burma14	0	0.00	1	0	0.00	1	0	0.00	2	0	0.00	1
ulysses16	0	0.00	1	0	0.00	1	0	0.00	6	0	0.00	1
gr17	0	0.00	1	10	0.00	1	0	0.00	1	154	0.00	1
gr21	0	0.00	1	0	0.00	1	0	0.00	7	0	0.00	1
ulysses22	0	0.00	1	0	0.00	1	0	0.00	3	0	0.00	2
gr24	0	0.00	2	0	0.00	1	7	0.00	51	65	0.00	3
fri26	0	0.00	1	0	0.00	1	0	0.00	26	233	0.00	15
bayg29	0	0.00	2	9	0.00	1	7	0.00	66	1964	0.00	22
bays29	0	0.00	2	0	0.00	1	0	0.00	12	80	0.00	3
dantzig42	0	0.00	1	0	0.00	9	0	0.00	43	2607	0.00	66
swiss42	0	0.00	1	7	0.00	13	49	0.00	69	1552	0.00	60
att48	0	0.00	1	13768	0.00	164	0	0.00	26	919012	0.32	*TL*
gr48	0	0.00	1	3	0.00	13	0	0.00	30	112771	0.00	680
hk48	0	0.00	1	133001	0.00	570	0	0.00	30	411933	1.01	*TL*
eil51	0	0.00	1	0	0.00	6	43	0.00	127	2077	0.00	64
berlin52	0	0.00	7	4112	0.00	97	0	0.00	19	17566	0.00	260
brazil58	0	0.00	2	142655	3.57	*ML*	0	0.00	9	354682	0.14	*ML*
st70	0	0.00	17	15707	0.00	167	162	0.00	219	92167	4.27	*ML*
eil76	0	0.00	2	0	0.00	25	0	0.00	42	3467	0.00	117
pr76	18563	1.56	*TL*	347543	6.25	*TL*	13725	0.00	1929	358909	5.85	*TL*
gr96	0	0.00	33	79851	1.10	*TL*	0	0.00	38	143113	0.87	*TL*
rat99	0	0.00	15	127959	1.23	*ML*	102	0.00	142	318955	0.77	*TL*
kroA100	0	0.00	11	207556	3.49	*TL*	25	0.00	213	234948	5.81	*TL*
kroB100	0	0.00	11	214600	5.88	*TL*	59	0.00	156	49763	5.69	*ML*
kroC100	0	0.00	4	215818	3.41	*TL*	0	0.00	166	40866	6.59	*ML*
kroD100	0	0.00	52	79659	5.88	*ML*	22	0.00	58	300417	3.42	*TL*
kroE100	0	0.00	129	184395	3.53	*TL*	178	0.00	219	51831	4.37	*ML*
(average)	688	0.06	145	65432	1.27	1240	533	0.00	137	126635	1.45	1648

The obtained results clearly indicate the strength of formulation (CUT), which solves to optimality almost all considered instances within seconds, with little or no branching necessary. The only exception forms instance *pr76*, which appears to be much harder than the other instances, despite its comparable size.

Figure 2: Performance comparison of (CUT) for different budget values (as specified by α) with uniform (left) and random profits (right). The percentage of instances solved within a certain time is shown.

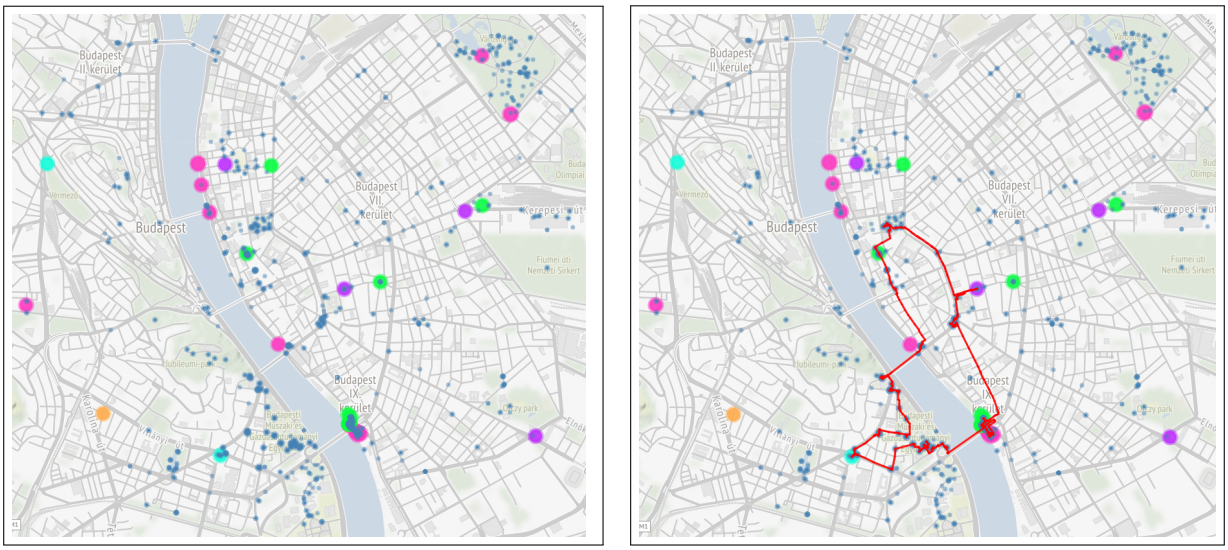


In contrast, formulation (MTZ) only manages to achieve results comparable to (CUT) on instances with up to 30 nodes. For larger instances, the framework fails to solve to optimality in most cases due hitting the memory or time limit, and the gap upon termination is still relatively large. We conclude that although model (MTZ) enables the quick enumeration of a vast number of B&B nodes, the provided bounds are too weak to be competitive with formulation (CUT). Formulation (MTZ) is thus excluded from all subsequent experiments on instances having $|V| > 100$.

4.2 Results on instance group `TSPLIB`

Tables 3–5 list results using formulation (CUT) for $\alpha \in \{0.25, 0.5, 0.75\}$ on instances from group `TSPLIB` for $|V| > 100$. The columns of each table report information in the following order: the name of the associated TSPLIB instance (*instance*), the budget computed from α (B), and the number of clusters ($|C|$). For uniform and random profits, the meaning of the shown columns is as follows: the best lower bound (LB, marked in bold if optimal), the number of separated inequalities (GSEC.1) and (GSEC.2) ($G1$, $G2$), the number of times an inequality could be strengthened due to the prize-based lifting (*Lift.*), and the number of separated inequalities (CYCLECOVER) (CC). The remaining columns carry the same meaning as for the previous table. The column (GSEC) has been omitted due to the fact that no inequalities of this type have been separated. This last fact is attributable mainly to the effective prize-based lifting procedure and to the careful selection of the GSEC-defining set S for each identified violated inequality. In general, instances with random profits appear to be computationally more difficult than those with uniform profits. Their structure also causes much more inequalities of type (CYCLECOVER) to be separated and the prize-based lifting is also more effective. Naturally, the lifting was most effective on the instances with $\alpha = 0.75$, as the revenue that can be collected is higher.

Concerning performance differences with respect to different values of α, Figure 2 summarizes the performance of formulation (CUT) on all instances from group `TSPLIB`. The percentage of instances solved within a certain time limit is shown for uniform and random profits. On average, uniform instances are clearly easier to solve than random profit ones. In addition, instances tend to be most difficult for $\alpha = 0.5$. This behavior is most noticeable for random profits, where almost 10% less instances can be solved to optimality within the given time limit.



(a) Instance Budapest

(b) Optimal tour for $|\mathcal{C}| = 5$, $B = 10$

Figure 3: Instance Budapest and an optimal tour ($|\mathcal{C}| = 5$, $B = 10$).

4.3 Results on instance group `Pokémon`

Table 6 reports results on the `Pokémon` instances where budgets and profits are defined as for the `TSPLIB` instances. The column meaning is equivalent to the one detailed at the beginning of the previous subsection. The results suggest performance similar to the largest `TSPLIB` instances. Note that the budgets considered in the instances associated to Table 6 are rather high, and may not be suitable for travel by foot or bike. Therefore Table 7 list results on smaller budgets and where only a random subset of all clusters has been selected. In addition, nodes within the same cluster have equal profit. Instances proven infeasible contain "-" in column *LB*. For each city clusters are chosen in a way such that the cluster collection of an instance with few clusters is a subset of the cluster collection of an instance with more clusters. If an instance is infeasible for a given number of clusters, then the next higher number of clusters is skipped.

Results suggest that the problem becomes much easier for this type of instances, as all but one instance could be solved to optimality for both uniform and random profits. However, note that the number of enumerated B&B nodes is in general much higher for random profits. In addition, a much higher number of inequalities (GSEC.2) is separated, mainly attributable due to the fact that many nodes are now purely optional, i.e., not contained in any cluster. Preliminary experiments have shown that proper cut management is crucial for solving these instances efficiently. This includes the use of separation thresholds and the selection of the minimum cardinality nodeset S.

Figure 3 shows a zoomed-out view of the real-world instance Budapest for five clusters, Figure 3(a), and a distance budget of ten kilometers. This distance can be traversed by foot in about two hours. Small blue circles denote nodes not associated to any cluster, while large colored circles denote locations associated to a specific cluster identified by the corresponding color. An optimal tour is shown as red line in Figure 3(b). Note that each of the five colors is visited at least once, and clusters of blue nodes are visited preferably. For the purpose of rendering the tour, the selected edges have been mapped back to the associated shortest paths in the street network. Note that due to the street map layout, circles and backtracking may appear in an optimal solution. The figure shows that the top-right corner is rich in profit, but nonetheless is not visited by the optimal tour due to providing only few nearby cluster nodes. Instead, the tour focuses on the lower part of the map, from which nodes from all clusters can be reached, and which still provides a reasonable amount of profit. However, long parts of the tour are traversed where no location is encountered for some time.

(a) Optimal tour for $|\mathcal{C}| = 5$, $B = 10$

(b) Optimal tour for $|\mathcal{C}| = 10$, $B = 15$

Figure 4: Zoomed-in view of instance Vienna and optimal tours for ($|\mathcal{C}| = 5$, $B = 10$) and ($|\mathcal{C}| = 10$, $B = 15$).

Figure 4 shows a zoomed-in view of instance Vienna for two configurations of clusters and budgets. Note that for the same city cluster collections have been generated in order to form subsets. Thus the images show how the instance changes when additional clusters are added. Figure 4(a) shows an extreme case in which the only nearby node from the orange cluster is far away. Figure 4(b) shows the same instance with an enlarged budget and five additional clusters. We see that with the larger budget, the optimal tour focuses on the profit-rich city center. The Donau channel is traversed in order to reach a node from the blue cluster. Figure 5 shows another zoomed-in view of the third instance, London. The view highlights the fact that an optimal solution might frequently require backtracking in the form of short side-trips for the purpose of collecting profit aside the main-tour.

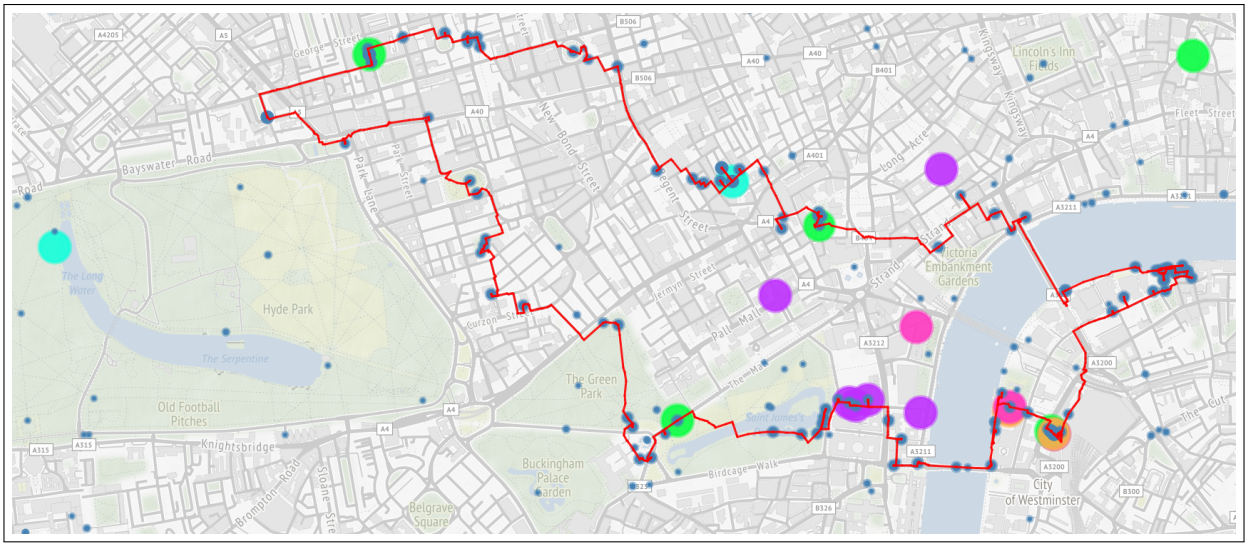


Figure 5: Zoomed-in view of instance London and an optimal tour for ($|\mathcal{C}| = 5$, $B = 15$).

Table 3: Results obtained using formulation (CUT) on instance group TSPLIB ($\alpha = 0.25$).

			uniform profits								***random profits***							
instance	B	$\|\mathcal{C}\|$	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]
eil101	405	22	**75**	27	30	0	0	0	0.00	9	**4095**	155	58	6	25	7	0.00	107
lin105	10634	22	**77**	62	81	0	0	0	0.00	19	**4196**	107	139	6	131	12	0.00	206
pr107	34262	23	**69**	23	12	0	0	0	0.00	3	**4213**	46	27	0	5	0	0.00	11
gr120	4656	25	**94**	19	29	0	0	0	0.00	3	**5290**	52	57	14	7	0	0.00	54
pr124	47480	26	**103**	58	58	0	0	0	0.00	11	**5334**	73	75	0	9	1	0.00	32
bier127	82049	27	**111**	71	48	3	1	0	0.00	25	**5888**	74	64	8	6	0	0.00	42
ch130	4298	27	**97**	44	90	0	0	0	0.00	39	**5311**	188	183	2	20	153	0.00	639
pr136	61852	29	**91**	246	193	0	14	0	0.00	83	**5322**	101	128	0	87	43	0.00	157
gr137	50773	29	**111**	40	26	0	0	0	0.00	6	**5889**	117	107	2	71	35	0.00	220
pr144	53431	30	**120**	101	90	0	0	0	0.00	66	**6380**	102	135	0	22	45	0.00	282
ch150	4337	31	**108**	48	69	0	1	0	0.00	17	**6096**	132	109	0	26	0	0.00	149
kroA150	17492	31	**107**	239	192	0	1	174	0.00	500	**5884**	409	233	3	71	189	0.00	310
kroB150	18212	31	**118**	37	53	1	0	0	0.00	18	**6411**	136	91	1	40	69	0.00	364
pr152	60537	32	**116**	233	215	1	0	360	0.00	316	**6220**	308	291	0	40	470	0.00	753
u159	30937	33	**121**	535	213	0	1	0	0.00	125	**6514**	381	174	0	37	50	0.00	477
si175	10058	36	**83**	225	274	0	0	0	0.00	27	**5605**	218	472	53	86	976	0.00	653
brg180	45525	37	**180**	0	0	0	0	0	0.00	6	**9030**	0	0	0	0	0	0.00	5
rat195	1485	40	**129**	155	58	0	0	0	0.00	57	**7278**	284	112	0	14	0	0.00	204
d198	12686	41	**189**	119	64	7	0	0	0.00	82	**9684**	280	58	28	15	0	0.00	418
kroA200	20470	41	**147**	618	415	0	0	1169	0.00	1383	8283	1024	815	3	0	5438	1.26	*TL*
kroB200	20245	41	**154**	130	129	0	3	0	0.00	37	**8140**	240	229	0	39	166	0.00	546
gr202	28498	42	**192**	122	78	10	0	21	0.00	631	**9848**	136	61	18	3	36	0.00	382
ts225	92219	46	176	755	396	2	0	1822	1.70	*TL*	**9359**	393	266	0	21	1511	0.00	1482
tsp225	2512	46	**155**	239	179	0	0	0	0.00	214	**8512**	169	215	0	15	0	0.00	204
pr226	69104	47	**218**	117	48	3	0	0	0.00	16	**11192**	167	57	13	0	0	0.00	70
gr229	82684	47	**184**	137	134	0	0	0	0.00	39	**9719**	584	248	10	26	66	0.00	391
gil262	1574	54	**193**	92	128	0	0	0	0.00	51	**10347**	235	237	0	0	32	0.00	474
pr264	36863	54	**184**	263	108	0	3	0	0.00	89	**10467**	230	133	0	344	1292	0.00	1394
a280	1688	57	**187**	1408	262	0	1	3	0.00	519	**10623**	1001	525	0	93	880	0.00	1370
pr299	33373	61	**213**	470	415	0	1	242	0.00	762	11773	1642	940	1	0	1652	0.90	*TL*
lin318	29029	65	**234**	373	396	0	2	41	0.00	485	12472	1805	654	0	20	1312	0.32	*TL*
rd400	9875	81	**284**	425	437	0	0	241	0.00	1664	15416	1210	904	1	2	654	1.26	*TL*
fl417	10105	85	**400**	363	221	1	0	31	0.00	740	20305	778	301	13	0	300	0.15	*TL*
gr431	107744	88	**381**	353	226	0	0	0	0.00	670	19860	591	263	111	0	483	0.03	*TL*
pr439	78628	89	**318**	266	304	0	0	0	0.00	297	**16885**	406	366	0	18	160	0.00	1011
pcb442	33447	90	301	2987	732	0	0	1	2.66	*TL*	17275	1518	551	0	0	322	1.59	*TL*
d493	25303	100	446	601	459	3	0	175	0.45	*TL*	23412	605	489	63	0	171	0.39	*TL*
(average)			175	324	185	1	1	116	0.13	535		430	264	10	35	447	0.16	1114

Table 4: Results obtained using formulation (CUT) on instance group TSPLIB ($\alpha = 0.5$).

			uniform profits								***random profits***							
instance	B	$\|\mathcal{C}\|$	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]
eil101	480	22	**87**	20	13	0	0	0	0.00	3	**4630**	88	31	1	11	0	0.00	227
lin105	11882	22	**91**	43	39	1	0	0	0.00	4	**4785**	67	58	2	42	3	0.00	145
pr107	37609	23	**83**	62	22	0	0	0	0.00	16	**4827**	57	16	0	6	0	0.00	37
gr120	5418	25	**106**	16	21	0	0	0	0.00	5	**5681**	24	30	3	6	0	0.00	29
pr124	51330	26	**112**	49	32	1	0	0	0.00	12	**5802**	65	51	1	19	2	0.00	272
bier127	94127	27	**121**	36	24	5	0	0	0.00	21	**6284**	84	33	22	0	0	0.00	88
ch130	4902	27	**112**	34	47	0	0	0	0.00	19	**5943**	216	171	6	36	358	0.00	458
pr136	73492	29	**110**	41	39	0	4	0	0.00	37	**6180**	81	54	1	26	23	0.00	146
gr137	57133	29	**124**	33	23	0	0	0	0.00	31	**6489**	84	51	12	5	0	0.00	153
pr144	55133	30	**129**	82	103	0	0	0	0.00	85	**6727**	136	132	0	31	991	0.00	697
ch150	5067	31	**126**	66	73	0	0	0	0.00	26	**6824**	115	123	2	7	250	0.00	428
kroA150	20503	31	**126**	137	118	0	0	0	0.00	71	**6816**	277	165	6	17	176	0.00	537
kroB150	20851	31	**131**	71	79	2	0	0	0.00	57	**7014**	165	120	1	2	74	0.00	193
pr152	64919	32	**132**	190	148	0	1	111	0.00	328	**6948**	163	169	6	371	2545	0.00	1102
u159	34651	33	**137**	205	92	0	1	0	0.00	65	**7259**	346	124	6	44	75	0.00	318
si175	13841	36	**116**	593	411	0	0	251	0.00	705	**7336**	118	84	4	99	226	0.00	442
brg180	31000	37	**180**	0	0	0	0	0	0.00	6	**9030**	0	0	0	0	0	0.00	6
rat195	1764	40	**153**	273	66	0	4	0	0.00	289	**8477**	48	26	0	6	0	0.00	56
d198	13717	41	**197**	93	13	30	0	0	0.00	46	**9901**	127	3	42	0	0	0.00	109
kroA200	23436	41	169	424	295	1	0	2470	0.59	*TL*	9297	491	359	5	0	4365	0.43	*TL*
kroB200	23309	41	**173**	124	120	1	0	0	0.00	187	**9168**	134	121	0	20	34	0.00	390
gr202	32385	42	**198**	65	36	4	0	0	0.00	28	**10043**	93	31	25	0	0	0.00	96
ts225	103694	46	197	583	221	0	0	1815	0.51	*TL*	10312	513	235	0	5	2526	0.34	*TL*
tsp225	2981	46	**184**	209	126	0	0	0	0.00	185	**9896**	142	154	5	33	64	0.00	337
pr226	72859	47	**222**	181	19	56	0	0	0.00	15	**11316**	210	12	70	2	3	0.00	326
gr229	99990	47	**212**	61	54	0	0	0	0.00	64	**10853**	216	98	5	2	43	0.00	281
gil262	1842	54	**221**	104	136	0	0	3	0.00	168	**11722**	992	367	2	45	1144	0.00	2048
pr264	40954	54	**225**	186	116	0	0	0	0.00	390	**12007**	124	53	4	7	0	0.00	271
a280	1985	57	**224**	325	158	1	0	11	0.00	344	12301	1105	364	3	9	1825	0.52	*TL*
pr299	38312	61	**248**	421	439	0	0	559	0.00	1841	13485	731	477	1	0	1198	0.85	*TL*
lin318	33362	65	**274**	249	201	0	2	31	0.00	564	14312	692	426	3	19	968	0.31	*TL*
rd400	11677	81	**336**	269	256	0	1	121	0.00	1307	17910	791	494	2	0	431	1.28	*TL*
fl417	10690	85	**409**	367	167	3	0	0	0.00	413	20660	624	256	45	0	272	0.11	*TL*
gr431	128967	88	**410**	208	144	44	0	5	0.00	528	20968	579	189	51	0	311	0.05	*TL*
pr439	88158	89	**375**	388	213	0	2	120	0.00	1568	19776	678	290	30	77	406	0.03	*TL*
pcb442	39224	90	**366**	596	365	0	5	190	0.00	1912	20149	834	313	2	6	340	0.22	*TL*
d493	28536	100	478	260	242	15	0	161	0.21	*TL*	24495	413	265	81	0	155	0.15	*TL*
(average)			197	191	126	4	1	158	0.04	598		314	161	12	26	508	0.12	1319

Table 5: Results obtained using formulation (CUT) on instance group TSPLIB ($\alpha = 0.75$).

			uniform profits								***random profits***							
instance	B	$\|\mathcal{C}\|$	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]
eil101	554	22	**95**	7	4	1	0	0	0.00	2	**4998**	124	5	18	30	6	0.00	127
lin105	13130	22	**99**	20	16	4	0	0	0.00	2	**5145**	35	29	7	41	10	0.00	127
pr107	40956	23	**95**	49	38	0	0	0	0.00	34	**5266**	70	36	19	34	0	0.00	296
gr120	6180	25	**115**	25	26	2	0	0	0.00	18	**5904**	152	16	30	24	15	0.00	239
pr124	55180	26	**119**	72	38	1	0	0	0.00	116	**6107**	80	21	26	31	3	0.00	209
bier127	106204	27	**125**	74	0	30	0	0	0.00	43	**6387**	69	3	35	1	0	0.00	33
ch130	5506	27	**122**	26	23	0	0	0	0.00	56	**6372**	98	52	12	27	91	0.00	281
pr136	85132	29	**125**	45	35	2	0	0	0.00	105	**6670**	97	11	23	99	0	0.00	436
gr137	63493	29	**132**	57	16	6	0	0	0.00	171	**6814**	142	2	45	34	16	0.00	288
pr144	56835	30	**136**	59	101	0	0	14	0.00	303	**7046**	113	106	15	12	138	0.00	302
ch150	5797	31	**139**	57	59	0	0	3	0.00	315	**7302**	96	46	18	17	0	0.00	221
kroA150	23513	31	**140**	43	40	0	0	0	0.00	47	**7417**	170	47	35	7	149	0.00	399
kroB150	23490	31	**142**	48	48	4	0	0	0.00	197	**7463**	168	20	62	4	36	0.00	316
pr152	69300	32	**144**	79	71	0	0	0	0.00	206	**7440**	118	67	7	0	0	0.00	289
u159	38365	33	**151**	38	40	0	0	0	0.00	10	**7767**	565	116	91	23	5556	0.00	2395
si175	17624	36	**147**	114	109	1	0	0	0.00	115	**8407**	36	26	3	8	0	0.00	110
brg180	16475	37	**180**	0	0	0	0	0	0.00	5	**9030**	0	0	0	0	0	0.00	3
rat195	2043	40	**176**	195	52	0	0	0	0.00	320	**9374**	152	43	6	4	9	0.00	277
d198	14748	41	**197**	79	0	32	0	0	0.00	13	**9901**	121	1	51	0	0	0.00	55
kroA200	26402	41	**187**	186	167	1	0	736	0.00	1532	**9890**	278	78	57	9	3618	0.00	2309
kroB200	26373	41	**188**	69	74	0	0	0	0.00	412	**9827**	129	75	8	20	64	0.00	429
gr202	36272	42	**200**	117	0	63	0	0	0.00	74	**10132**	140	0	51	0	0	0.00	249
ts225	115168	46	212	726	329	0	0	1700	1.42	*TL*	11061	491	161	10	5	2419	0.13	*TL*
tsp225	3450	46	**207**	147	76	0	0	0	0.00	302	**10845**	380	132	28	58	809	0.00	1038
pr226	76614	47	**224**	166	0	68	0	0	0.00	18	**11404**	236	0	92	4	0	0.00	243
gr229	117296	47	**222**	194	83	29	0	0	0.00	364	**11407**	178	75	22	1	18	0.00	251
gil262	2110	54	**244**	114	116	0	0	6	0.00	324	**12705**	331	163	4	0	192	0.00	604
pr264	45044	54	**254**	226	110	2	1	260	0.00	1035	**12979**	168	54	10	8	25	0.00	292
a280	2282	57	**256**	275	108	0	0	725	0.00	1328	**13567**	177	93	9	7	57	0.00	354
pr299	43251	61	276	306	270	0	0	931	0.72	*TL*	14637	474	202	22	0	943	0.36	*TL*
lin318	37695	65	**299**	237	139	1	0	630	0.00	2412	**15498**	315	183	4	33	340	0.00	1582
rd400	13479	81	**374**	120	140	0	0	0	0.00	249	19661	391	212	9	0	353	0.22	*TL*
fl417	11275	85	**414**	410	142	64	0	0	0.00	578	20936	655	110	184	0	270	0.17	*TL*
gr431	150190	88	**423**	245	132	30	0	11	0.00	729	21542	560	127	130	0	261	0.03	*TL*
pr439	97687	89	**417**	414	198	1	0	66	0.00	1287	21536	404	219	24	14	311	0.33	*TL*
pcb442	45001	90	**414**	324	168	0	1	11	0.00	804	21841	502	141	35	7	237	0.01	*TL*
d493	31769	100	487	399	167	162	0	168	1.03	*TL*	24803	447	44	243	0	168	0.06	*TL*
(average)				156	85	14	0	142	0.09	657		234	73	39	15	436	0.04	1150

Table 6: Results obtained using formulation (CUT) on instance group `Pokémon` with budgets and profits defined as for instance group `TSPLIB`.

			uniform profits								***random profits***						
instance	B[km]	$\|\mathcal{C}\|$	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]
$\alpha = 0.25$																		
Budapest	47.523	72	**393**	190	892	13	0	115	0.00	1804	20565	513	1202	106	16	161	0.01	*TL*
London	60.924	75	340	253	1101	12	0	250	1.18	*TL*	17931	475	1246	25	0	222	0.66	*TL*
Vienna	115.927	72	**417**	52	621	5	0	0	0.00	752	21404	166	1101	55	0	148	0.42	*TL*
(average)				165	871	10	0	122	0.39	2052		385	1183	62	5	177	0.36	*TL*
$\alpha = 0.50$																		
Budapest	56.727	72	**430**	130	679	33	0	0	0.00	651	22249	232	808	96	5	172	0.20	*TL*
London	75.589	75	**386**	120	791	35	0	90	0.00	1808	20110	402	896	128	13	266	0.09	*TL*
Vienna	148.109	72	**453**	123	579	67	0	85	0.00	2269	23264	206	616	96	0	191	0.27	*TL*
(average)				124	683	45	0	58	0.00	1576		280	773	107	6	210	0.19	*TL*
$\alpha = 0.75$																		
Budapest	65.930	72	**455**	121	417	60	0	0	0.00	476	23248	311	461	169	0	169	0.06	*TL*
London	90.254	75	**418**	87	454	50	0	30	0.00	848	21571	275	431	154	0	252	0.05	*TL*
Vienna	180.291	72	475	232	369	194	0	129	0.21	*TL*	24092	270	360	192	0	130	0.24	*TL*
(average)				147	413	101	0	53	0.07	1641		285	417	172	0	184	0.12	*TL*

Table 7: Results obtained using formulation (CUT) on `Pokémon` instances with budgets, profits and clusters chosen according to the real-world problem setting. Proven infeasibility is denoted by '-' in column LB (lower bound), while optimality is denoted bold.

		uniform profits								***random profits (per species)***							
B[km]	$\|\mathcal{C}\|$	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]	*LB*	*G1*	*G2*	*Lift.*	*CC*	*BBn.*	g[%]	t[s.]
Budapest																	
10	5	**113**	218	2707	0	0	12	0.00	679	**6752**	216	2546	2	0	592	0.00	617
	10	**75**	382	1717	0	0	0	0.00	218	**4229**	439	1762	0	8	19	0.00	329
	15	**-**	382	1494	0	0	0	-	154	**-**	449	1641	0	0	0	-	212
15	5	**167**	131	2308	4	0	20	0.00	743	**9801**	190	2489	5	0	660	0.00	880
	10	**156**	329	2031	0	1	21	0.00	583	**9039**	411	1789	1	27	325	0.00	623
	15	**147**	426	2084	0	0	0	0.00	462	**8346**	488	2260	0	42	137	0.00	522
	20	**147**	387	2153	0	0	0	0.00	521	**8279**	428	2018	1	101	390	0.00	944
20	5	**213**	132	2539	25	0	105	0.00	1149	12257	241	3790	162	0	960	0.30	*TL*
	10	**207**	335	1953	2	0	29	0.00	1164	**11888**	387	1944	3	4	204	0.00	907
	15	**202**	354	2124	0	0	13	0.00	1608	**11425**	548	2332	0	3	151	0.00	949
	20	**202**	418	2152	0	0	0	0.00	1392	**11358**	568	2392	0	33	1125	0.00	2803
London																	
10	5	**96**	90	1768	2	0	12	0.00	313	**5374**	97	2040	6	0	108	0.00	399
	10	**85**	235	1756	0	0	27	0.00	354	**4741**	298	1931	0	0	82	0.00	386
	15	**74**	340	1861	0	0	0	0.00	242	**4031**	358	1976	0	0	33	0.00	404
	20	**-**	301	1700	0	0	0	-	156	**-**	332	1828	0	0	0	-	286
15	5	**137**	25	1079	3	0	0	0.00	85	**7508**	42	1599	2	0	50	0.00	211
	10	**137**	54	1185	1	0	0	0.00	161	**7488**	85	1516	0	0	20	0.00	224
	15	**133**	80	1240	0	0	0	0.00	166	**7209**	103	1274	0	0	27	0.00	175
	20	**119**	187	1847	0	4	31	0.00	335	**6768**	145	1432	0	0	37	0.00	317
	25	**119**	197	1622	0	0	11	0.00	207	**6756**	169	1562	0	0	139	0.00	372
20	5	**167**	72	1541	5	0	51	0.00	268	**9288**	26	2244	19	0	296	0.00	550
	10	**167**	76	1775	3	0	12	0.00	358	**9265**	63	1598	1	0	30	0.00	312
	15	**166**	91	1428	0	0	27	0.00	328	**9265**	233	2087	3	12	104	0.00	452
	20	**160**	102	1464	0	1	11	0.00	227	**8893**	127	1860	1	1	221	0.00	447
Vienna																	
10	5	**52**	277	2025	0	0	34	0.00	668	**2878**	345	2488	1	0	1292	0.00	920
	10	**-**	254	1218	0	0	0	-	89	**-**	395	1390	0	0	0	-	130
15	5	**101**	167	1885	1	0	0	0.00	250	**5394**	297	3045	11	0	1009	0.00	1220
	10	**90**	342	1703	0	3	11	0.00	348	**4976**	408	2079	0	12	46	0.00	388
	15	**-**	208	1044	0	0	0	-	72	**-**	276	1189	0	0	0	-	119
20	5	**132**	144	2129	5	0	36	0.00	446	**7218**	227	3177	30	75	673	0.00	1015
	10	**131**	279	1551	0	0	0	0.00	310	**6984**	434	2285	6	94	676	0.00	804
	15	**-**	362	1812	0	0	0	-	318	**-**	479	2048	0	2	0	-	534
(average)			234	1788	1	0	14	0.00	495		306	2067	7	16	322	0.01	754

5 Conclusions and Future Work

The Pokémon GO game is a prime example of *augmented reality games*, in which players have to perform tasks in the real world in order to progress in the game. The enormous success of the game has allowed the growth of many side business and has enhanced other industries as well; for instance, in cities such as Edinburgh, taxi drivers offer tourists trips not only to visit the main sights of the city, but also to the locations of valuable Pokémon [see,e.g., 5]; and companies have offered the service to play the game for their customers in order to assist them at enlarging their Pokémon collections [10].

In this paper, we introduce the Generalized Clustered Orienteering Problem (GCOP), which is motivated by the Pokémon GO game. The proposed problem is a generalization of the Orienteering Problem and also possesses elements of the Generalized Travelling Salesman Problem. Aside from the motivating example of the Pokémon GO game, it also has applications in more traditional transportation settings.

We present an algorithmic framework for the exact solution of the GCOP. The framework is based on Mixed-Integer Programming (MIP) models for the problem. Two MIP formulations, one based on the Miller-Tucker-Zemlin (MTZ) constraints, and the other based on generalized subtour elimination constraints (GSECs) are devised. The framework is enhanced by valid inequalities, lifting of inequalities, an initialization heuristic and a primal heuristic.

In order to evaluate the performance of our proposed framework, an extensive computational study is presented using instances of the well-known `TSPLIB` data-set, as well as real-life instances based on crowd-sourced data from the Pokémon GO game for the cities of Vienna, Budapest and London. The results show that our approach allows to solve real-life instances to optimality within reasonable time.

For further work, it could be interesting to extend the prize-collecting aspect of the problem also to the clusters, i.e., visiting a cluster is optional, but yields a profit. Moreover, considering a stochastic or robust version of the problem appears promising as the real-world data may be uncertain due to its crowd-sourced nature.

Acknowledgments

E. Álvarez-Miranda acknowledges the support of the Chilean Council of Scientific and Technological Research, CONICYT, through the grant FONDECYT N.11140060 and through the Complex Engineering Systems Institute (ICM-FIC:P-05-004-F, CONICYT:FB0816). The research of M. Sinnl was supported by the Austrian Research Fund (FWF, Project P 26755-N19). M. Luipersbeck acknowledges the support of the University of Vienna through the uni:docs fellowship programme.

References

References

[1] PokeMapper. `http://www.pokemapper.co`, Accessed: 2016-07-27.

[2] Pokémon GO Official Website. `http://www.pokemongo.com/`, Accessed: 2016-08-03.

[3] Pokémon GO Wikipedia Article. `https://en.wikipedia.org/wiki/Pok%C3%A9mon_Go`, Accessed: 2016-08-03.

[4] Library of sample instances for the TSP (and related problems). `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`, Accessed: 2016-08-03.

[5] Head to Edinburgh if youd like to try a Pokémon Go taxi tour (Lonely Planet). `http://www.lonelyplanet.com/news/2016/07/27/try-pokemon-go-taxi-tour/`, Accessed: 2016-08-12.

[6] Pokémon Go crosses USD 250 millions in revenues since launch. `http://www.ft.com/cms/s/0/2dd63522-5fdf-11e6-ae3f-77baadeb1c93.html`, Accessed: 2016-08-13.

[7] Stamen design. http://www.stamen.com, Accessed: 2016-08-13.

[8] 10 Best Augmented Reality Games Like Pokemon GO. http://www.mobipicker.com/games-like-pokemon-go/, Accessed: 2016-08-13.

[9] Austria's first PokeWalk. http://www.pokewalk.at/, Accessed: 2016-08-16.

[10] PokeWalk. http://www.pokewalk.com/, Accessed: 2016-08-16.

[11] E. Angelelli, C. Archetti, and M. Vindigni. The clustered orienteering problem. *European Journal of Operational Research*, 238(2):404–414, 2014.

[12] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *SIAM Journal on Computing*, 28(1):254–262, 1998.

[13] E. Balas. The prize-collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[14] T. Bektas and L. Gouveia. Requiem for the millertuckerzemlin subtour elimination constraints? *European Journal of Operational Research*, 236(3):820–832, 2014.

[15] J. Bérubé, M. Gendrau, and J. Potvin. A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem. *Networks*, 54:56–67, 2009.

[16] D. Bienstock, M. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Progamming*, 59(1-3):413–420, 1993.

[17] I. Chao, B. Golden, and E. Wasil. Theory and methodology the team orienteering problem. *European Journal of Operational Research*, 88:464–474, 1996.

[18] B. Cherkassky and A. Goldberg. On implementing push-relabel method for the maximum flow problem. In E. Balas and J. Clausen, editors, *Proceedings of IPCO IV*, volume 920 of *LNCS*, pages 157–171. Springer, 1995.

[19] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (OGDF). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*. CRC Press, 2014. Chapter 17.

[20] W. Cook. Concorde TSP Solver Website. http://www.math.uwaterloo.ca/tsp/concorde/index.html.

[21] M. Dell'Amico, F. Maffioli, and P. Väbrand. On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research*, 2:297–308, 1995.

[22] M. Dell'Amico, F. Maffioli, and A. Sciomachen. A lagrangian heuristic for the prize collecting travelling salesman problem. *Annals of Operations Research*, 81:289–306, 1998.

[23] Martin Desrochers and Gilbert Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.

[24] D. Feillet, P. Dejax, and M. Gendreau. Travelling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

[25] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1–3):23–47, 2003.

[26] M. Fischetti and P. Toth. An additive approach for the optimal solution of the prize-collecting traveling salesman problem. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 319–343. North Holland, 1988.

[27] M. Fischetti, J. Salazar-González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

[28] M. Fischetti, J. Salazar-González, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148, 1999.

[29] M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective travelling salesman problem. *Networks*, 32:263–273, 1998.

[30] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.

[31] A. Gunawan, H. Chuin Lau, and P. Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *To appear in European Journal of Operational Research*, 2016.

[32] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[33] A. Kharpal. Mobile game revenue to pass console, PC for first time. `http://www.cnbc.com/2016/04/22/mobile-game-revenue-to-pass-console-pc-for-first-time.html`, Accessed: 2016-08-13.

[34] A. Laporte, G. Asef-Vaziri and C. Sriskandarajah. Some applications of the generalized travelling salesman problem. *The Journal of the Operational Research Society*, 47(12):1461–1467, 1996.

[35] G. Laporte and Y. Nobert. Generalized travelling salesman problem through n sets of nodes: an integer programming approach. *INFOR*, 21:61–75, 1983.

[36] G. Laporte, H. Mercure, and Y. Nobert. Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics*, 18(2):185–197, 1987.

[37] E. Lawler, J. Lenstra, H. Rinnoy Kan, and D. Shmoys, editors. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, 1st edition, 1985.

[38] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960.

[39] Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581, 1977.

[40] M. Schilde, K. Doerner, R. Hartl, and G. Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3:179–201, 2009.

[41] S. Srivastava, S. Kumar, R. Garg, and P. Sen. Generalized travelling salesman problem through n sets of nodes. *CORS Journal*, 7:97–101, 1969.

[42] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[43] M. Tasgetiren. A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic and Social Research*, 4(2):1–26, 2001.

[44] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, 1984.

[45] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.