8-2022

Submodularity and local search approaches for maximum capture problems under generalized extreme value models

Tien Thanh DAM
Phenikaa University

Thuy Anh TA
Phenikaa University

Tien MAI
Singapore Management University, atmai@smu.edu.sg

Citation

DAM, Tien Thanh; TA, Thuy Anh; and MAI, Tien. Submodularity and local search approaches for maximum capture problems under generalized extreme value models. (2022). *European Journal of Operational Research*. 300, (3), 953-965.
Available at: https://ink.library.smu.edu.sg/sis_research/6239

Submodularity and Local Search Approaches for Maximum Capture Problems under Generalized Extreme Value Models

Tien Thanh Dam[1], Thuy Anh Ta[1], and Tien Mai[2]

[1] *ORLab, Faculty of Computer Science, Phenikaa University, Yen Nghia, Ha Dong, Hanoi, VietNam*
[2] *School of Computing and Information Systems, Singapore Management University, 80 Stamford Rd, Singapore 178902*

September 28, 2021

Abstract

We study the maximum capture problem in facility location under random utility models, i.e., the problem of seeking to locate new facilities in a competitive market such that the captured user demand is maximized, assuming that each customer chooses among all available facilities according to a random utility maximization model. We employ the generalized extreme value (GEV) family of discrete choice models and show that the objective function in this context is monotonic and submodular. This finding implies that a simple greedy heuristic can always guarantee a $(1 - 1/e)$ approximation solution. We further develop a new algorithm combining a greedy heuristic, a gradient-based local search, and an exchanging procedure to efficiently solve the problem. We conduct experiments using instances of different sizes and under different discrete choice models, and we show that our approach significantly outperforms prior approaches in terms of both returned objective value and CPU time. Our algorithm and theoretical findings can be applied to the maximum capture problems under various random utility models in the literature, including the popular multinomial logit, nested logit, cross nested logit, and mixed logit models.

Keywords: Facilities planning and design, maximum capture, random utility maximization, generalized extreme value, greedy heuristic.

1 Introduction

In the last decade, the facility location problem in a competitive market has received growing attention. In practice, modelling critical managerial decisions related to infrastructure planning, such as finding locations to locate new retail, service, or product facilities in a market, often lead

to facility location problems. The competitive facility location problem deals with a decision of selecting locations to open new facilities in a market to maximize the captured demand of users, where a set of incumbent competitors are already operating in order. There are two aspects that need to be considered in this problem, namely, the demand of customers and the competitors in the market. Customers are independent decision-makers and their choices among different facilities might be based on a given utility that they assign to each location. Such utilities might be a function of facility attributes/features, e.g., distances, prices, and transportation costs.

There are several ways to define and estimate customer demand (Berman et al., 2009). In this work, we focus on a probabilistic approach, i.e., customer demand is captured by a probability model that assigns choice probabilities to the facilities. The random utility maximization (RUM) framework (Horowitz, 1986, McFadden, 1973) is convenient and popular in the context. This framework is based on the assumption that each facility is associated with a random utility, which can be determined by the features/attributes of the facility. The RUM principle assumes that each customer selects a facility by maximizing his/her utilities. This way of modeling allows for predicting the probability that a customer selects a facility. The facility location problem then becomes the problem of locating new facilities in a competitive market to maximize an expected captured demand function, where customers select a facility (a new facility or one from the competitors) according to a RUM model. Thus, the problem is also called as the *maximum capture problem* (MCP).

To the best of our knowledge, existing related studies in the literature only employ the multinomial logit (MNL) or its mixed version (mixed logit model - MMNL) (Benati and Hansen, 2002, Haase and Müller, 2013, Hasse, 2009). It is well-known that the MNL retains the independence from irrelevant alternatives (IIA) property, which does not hold in many contexts (McFadden, 1981, McFadden and Train, 2000). On the other hand, the generalized extreme value (GEV) family provides flexible ways to relax the IIA property and to capture the correlation between alternative utilities (McFadden, 1981). However, under the GEV family, most of the important properties that have been used to develop solution methods for the MCP under the MNL and MMNL models do not hold or have not been proved to be true. More precisely, the objective function under the GEV family does not have a linear fractional structure, thus it is difficult to formulate the MCP into a mixed-integer linear program (MILP) as in prior work (Benati and Hansen, 2002, Zhang et al., 2012). Moreover, under the GEV family, the objective function of the continuous relaxation is not either concave or convex, making the outer-approximation methods (Ljubić and Moreno, 2018, Mai and Lodi, 2020) not applicable. Furthermore, since the structure of the objective function is driven by a GEV choice probability generating function, which may not have a closed-form and could be complicated, it is not clear whether the objective function is submodular or not. All the above remarks make the MCP under the GEV family challenging. We tackle this challenge in this paper.

Before presenting our contributions in detail, we note that, from now on, when saying a "GEV model", we refer to any choice model in the GEV family. Each GEV model can be determined by a choice probability generating function (CPGF) (Fosgerau et al., 2013) (see a detailed

definition in the next section).

Our contributions: In this paper, we formulate and solve the MCP under any GEV models. We leverage the properties of the CPGFs of GEV models (Daly and Bierlaire, 2006, McFadden, 1981) and show that the objective function in the context is monotonic increasing and submodular. These properties are already known for the MCP under the MNL model (Benati, 1996) and now we show that they also hold for any GEV models. The monotonicity and submodularity also imply that the MCP subjecting to a cardinality constraint, even though being *NP-hard*, always admits a $(1 - 1/e)$ approximation algorithm. In other words, a simple greedy heuristic always returns a solution whose value is at least $(1 - 1/e)$ (≈ 0.632) times the optimal values (Nemhauser et al., 1978).

To further enhance the greedy heuristic (GH), we develop a new algorithm that adds a gradient-based local search and exchanging procedures to the GH. While the latter is simply based on steps of exchanging a location in a set of chosen locations with one outside of the set to get a better objective value, the former is motivated by the fact that if we formulate the MCP as a binary program, then the objective function is differentiable and we can make use of gradient information to direct the search. The gradient-based local search is an iterative procedure in which at each iteration we solve a subproblem to (hopefully) find a better candidate solution, and we show that such a subproblem is solvable in polynomial-time. Our algorithm can be used to solve problems under any GEV models and under the MMNL model.

We conduct experiments using some datasets from the recent literature, including real-life large-scale instances from an *park-and-ride* location problem in New York City (Holguin-Veras et al., 2012). We compare our algorithm, named as GGX (stands for **G**reedy Heuristic, **G**radient-based Local Search, and **Ex**changing) with some state-of-the-art approaches from recent literature, i.e., the Branch & Cut method proposed by Ljubić and Moreno (2018) and outer-approximation algorithms (Bonami et al., 2011, Mai and Lodi, 2020). Experiments based on MNL, MMNL, and nested logit instances show that our algorithm remarkably outperforms the other approaches, in terms of both returned objective value and CPU time.

Literature review: The GEV family (McFadden, 1981) covers most of the discrete choice models in the demand modeling and operations research literature. Among existing GEV models, the MNL is the simplest and most popular one. It is also well-known that the MNL retains the IIA property, which implies that the ratio between the choice probabilities of two facilities will not change no matter what other facilities are available or what attributes that other facilities have. This property has been regarded as a limitation of the MNL model and should be relaxed in many applications (McFadden and Train, 2000). There are several GEV models that relax this property and provide flexible ways to model the correlation between choice alternatives. For example, the nested logit (Ben-Akiva, 1973), the cross-nested logit Vovsha and Bekhor (1998), the generalized nested logit (Wen and Koppelman, 2001), the paired combinatorial logit (Koppelman and Wen, 2000), the ordered generalized extreme value Small (1987), the specialized compound generalized extreme value models (Whelan et al., 2002) and network-

based GEV (Daly and Bierlaire, 2006, Mai et al., 2017) models. GEV models, in particular, the cross-nested and network GEV models, are fully flexible, in the sense that they can approximate any random utility maximization models (Fosgerau et al., 2013). Besides the GEV family, the MMNL is also an alternative to relax the IIA property. This model extends the MNL by assuming that the choice parameters are random. Similar to GEV models, the MMNL is also capble of approximating any random utilities choice model (McFadden and Train, 2000). However, the choice probabilities given by the MMNL model have no closed-form and often require simulation to approximate. Thus, the estimation and application of this model are expensive in many contexts.

In the context of the MCP, most existing studies focus on the MNL model due to its simplicity. Benati and Hansen (2002) seem the first to introduce the MCP under the MNL model. They propose three methods to compute upper bounds along with a branch-and-bound method to solve small instances. The first method is based on the concavity of the continuous relaxation of the objective function. They show the submodularity of the objective function and use this property to develop the second method. The third method is an equivalent mixed-integer linear program (MILP), which is based on the fact that the objective function has a linear fractional structure and can be linearized using additional variables. Benati and Hansen (2002) also introduced a simple variable neighborhood search (VNS) method to solve instances with more than 50 potential locations. Some alternative MILP models, afterward, have been proposed by Hasse (2009) and Zhang et al. (2012). Haase and Müller (2013) give an evaluation and comparison of these proposed MILP models and conclude that the MILP model from Hasse (2009) is the most efficient one. Freire et al. (2015) strengthen the MILP reformulation of Hasse (2009) by using some tighter coefficients in some inequalities and also propose a new branch-and-bound algorithm to deal with the problem. Lin and Tian (2021a,b) further improve the MILP approach using Benders decomposition. Ljubić and Moreno (2018) propose a branch-and-cut method that combines two types of cutting planes, namely, outer-approximation (OA) and submodular cuts. The first type of cuts relies on the fact that the objective function of the continuous relaxation of the problem is concave and differentiable and the second type is based on the submodularity and separability properties of the objective function. Their branch-and-cut method is an iterative procedure where cuts are generated for every demand point and a linear programming (LP) relaxation is solved at each iteration. Mai and Lodi (2020) propose a multicut outer-approximation algorithm that works in a cutting plane fashion by solving an MILP at every iteration. This algorithm generates cuts for groups of demand points instead of one cut for every demand point as in Ljubić and Moreno (2018) or one cut for all the demand points as in the classical outer-approximation scheme (Bonami et al., 2008, Duran and Grossmann, 1986). The branch-and-cut proposed by (Ljubić and Moreno, 2018) and multicut outer-approximation are regarded as two state-of-the-art approaches for the MCP under the MNL model. Note that in the context of the MCP, MNL and MMNL instances have similar structures. Thus, all the methods developed for the MNL model can be applied to MMNL instances. There are also a couple of studies investigating the MCP under the MMNL model (Haase and Müller, 2013, Hasse, 2009). These studies make use of MILP formulations,

which are expensive for large instances and generally outperformed by the branch-and-cut and outer-approximation approaches (Ljubić and Moreno, 2018, Mai and Lodi, 2020).

In the context of approximation algorithms for facility location, the literature has seen several algorithms having worse-case bounds. For instance, Shmoys et al. (1997) develop approximation algorithms for several classical facility location problems. Aboolian et al. (2007) consider the problem of simultaneously optimizing location and design decisions for a set of new facilities. They also propose a near-optimal solution approach, with adjustable error bounds, to solve some special cases along with two heuristic methods to handle large instances. Ageev et al. (2004), Ortiz Astorquiza et al. (2017) consider multilevel uncapacitated p-location problems and also make use of the submodularity property to develop greedy heuristics that return $(1 - 1/e)$ approximation solutions. More recently, Kung and Liao (2018) propose an approximation-relaxation-sorting-aggregation algorithm for the competitive facility location problem with network effects. This method consists of four major steps which first approximate the original objective function, then, relax the problem by decomposing the linear integer program into subproblems before constructing a feasible solution by a simple sorting procedure, and finally, finish by aggregating all solutions from all subproblems to generate a solution to the original problem. They also establish some worst-case performance guarantees for some special cases. More work on approximation algorithms for facility location can be found in Laporte et al. (2015). Note that our work seems to be the first attempt to develop approximation procedures with a performance guarantee for competitive facility location problems under GEV demand models where the objective functions are highly nonlinear.

Paper outline: The rest of paper is structured as follow. Section 2 briefly presents the GEV family focusing on the some essential properties of the CPGF, and the MCP under the GEV family. In Section 3, we investigate the monotonicity and submodularity of the MCP under the GEV family, and present our local search algorithm. Section 4 reports computational results. Finally, Section 5 concludes.

Notation: Boldface characters represent matrices (or vectors), and a_i denotes the i-th element of vector $\mathbf{a}$. We use $[m]$, for any $m \in \mathbb{N}$, to denote the set $\{1, \ldots, m\}$.

2 Generalized Extreme Value Models and the Maximum Capture Problem

In this section, we introduce some basic concepts and properties of the GEV family and formulate the MCP under GEV models.

2.1 Generalized Extreme Value Models

The Random Utility Maximization (RUM) framework (McFadden, 1978) is the most popular approach to model discrete choice behavior. Under the RUM principle, the decision-maker is assumed to associate a utility u_j with each alternative/option j in a given choice set S that contains all possible alternatives. The additive RUM (Fosgerau and Bierlaire, 2009, McFadden, 1978) assumes that each random utility is a sum of two parts $u_j = v_j + \epsilon_j$, where the term v_j is deterministic and can include values representing characteristics of the alternative and/or the decision-maker, and the random term ϵ_j is unknown to the analyst. There are several assumptions that have been made on the randoms terms, which leads to different types of discrete choice models in the literature, e.g., the MNL or nested logit models (McFadden, 1978, Train, 2003). The deterministic terms v_j often have a linear structure, i.e., $v_j = \beta^{\mathrm{T}}\alpha_j$, where $^{\mathrm{T}}$ is the transpose operator and β is a vector of parameters to be estimated from historical data of how people make decisions, and α_j is a vector of attributes of alternative j. The RUM principle then assumes that a decision is made by maximizing the random utilities, and the probability that an alternative j is selected can be computed as $P(u_j \geq u_k,\ \forall k \in S)$.

The GEV family covers most of the existing discrete choice models in the literature. This family of model is fully flexible, in the sense that it allows to construct various discrete choice models that are consistent with the RUM principle (McFadden, 1981). Assume that the choice set contains m alternative indexed as $\{1, \ldots, m\}$ and let $\mathbf{U} = \{v_1, \ldots, v_m\}$ be the vector of utilities . A GEV model can be determined by a choice probability generating function (CPGF) $G(\mathbf{Y})$ (Fosgerau et al., 2013, McFadden, 1981), where $\mathbf{Y}$ is a vector of size m with entries $Y_j = e^{v_j}$. Given $j_1, \ldots, j_k \in [m]$, let $\partial G_{j_1 \ldots j_k}$, be the mixed partial derivatives of G with respect to $Y_{j_1}, \ldots, Y_{j_k}$. It is well-known that the CPGF $G(\cdot)$ and the mixed partial derivatives have the the following properties (McFadden, 1978).

Remark 1 *A CPGF $G(\boldsymbol{Y})$ of a GEV model, has the following properties.*

(i) $G(\boldsymbol{Y}) \geq 0,\ \forall\, \boldsymbol{Y} \in \mathbb{R}^m_+$,

(ii) $G(\boldsymbol{Y})$ *is homogeneous of degree one, i.e.,* $G(\lambda\, \boldsymbol{Y}) = \lambda G(\boldsymbol{Y})$

(iii) $G(\boldsymbol{Y}) \to \infty$ *if* $Y_j \to \infty$

(iv) Given $j_1, \ldots, j_k \in [m]$ *distinct from each other,* $\partial G_{j_1,\ldots,j_k}(\boldsymbol{Y}) > 0$ *if* k *is odd, and* $\leq$ *if* k *is even*

(v) $G(\boldsymbol{Y}) = \sum_{j\in[m]} Y_j \partial G_j(\boldsymbol{Y})$

(vi) $\sum_{k\in[m]} Y_k \partial G_{jk}(\boldsymbol{Y}) = 0,\ \forall j \in [m]$.

Here we note that $(i) - (iv)$ are basic properties of a GEV generating function (McFadden, 1981), and Properties (v) and (vi) are direct results from the homogeneity property. We will

make use of these properties throughout the rest of the paper to explore the properties of the objective function of the MCP and derive solution algorithms for the MCP. Under a GEV model specified by a CPGF $G(\mathbf{Y})$, the choice probability of an alternative $j \in [m]$, conditional on $\mathbf{Y}$ and G, is given by

$$P(j|\mathbf{Y}, G) = \frac{Y_i \partial G_i(\mathbf{Y})}{G(\mathbf{Y})}.$$

The GEV framework allows for correlated utilities and one can build different CPGF to model different correlation patterns among random utilities. One can build a GEV model from a network of correlation structures, which provides a very flexible way to construct choice models that are able to capture complex relationships between alternatives (Daly and Bierlaire, 2006, Mai et al., 2017). In the following, we show some specific instances of the GEV family that are already popular in the demand modeling and operations research literature.

The MNL model: The MNL is one of the most widely-used discrete choice models in the literature. This model results from the assumption that the random terms ϵ_j, $j \in [m]$, are independent and identically distributed (i.i.d.) and follow the standard Gumbel distribution. The CPGF function has a simple form as $G(\mathbf{Y}) = \sum_{j\in[m]} Y_j$ and the choice probabilities have the fractional form below

$$P(j|\mathbf{Y}, G) = \frac{Y_j}{\sum_{j\in[m]} Y_j} = \frac{e^{v_j}}{\sum_{j\in[m]} e^{v_j}}. \tag{1}$$

It is well-known that the MNL model exhibits from the IIA property, which means that the choice probability of an alternative will not be affected by the attributes or the state of the other alternatives. However, in some situations, alternatives share unobserved attributes (i.e. random terms are correlated) and the IIA property does not hold.

The nested logit model: The nested logit model (Ben-Akiva, 1973) is one of the first attempts to relax the IIA property from the MNL model. In this GEV model, the choice set is partitioned into L nests, which are disjoint subsets of alternatives. Let denote by $n_1, \ldots, n_L$ the L nests. The corresponding CPGF can be written as

$$G(\mathbf{Y}) = \sum_{l\in L} \left(\sum_{j\in n_l} Y_j^{\mu_l} \right)^{1/\mu_l},$$

where $\mu_l \geq 1$, $l \in [L]$, are the parameters of the nested model. This model is based on the observation that, in many situations, some similar or closely related alternatives can be grouped into smaller subsets. It is easy to see that $G(\mathbf{Y})$ satisfies the six properties above and the choice probabilities can be computed as

$$P(j|\mathbf{Y}, G) = \frac{\left(\sum_{j'\in n_l} Y_{j'}^{\mu_l}\right)^{1/\mu_l}}{\sum_{l\in[L]} \left(\sum_{j'\in n_l} Y_{j'}^{\mu_l}\right)^{1/\mu_l}} \frac{Y_j^{\mu_l}}{\sum_{j'\in n_l} Y_{j'}^{\mu_l}}, \ \forall l \in [L], j \in n_l.$$

The cross-nested logit model (Ben-Akiva and Bierlaire, 1999) is an extension of the nested logit that allows the nests to share common alternatives. This model is known to be fully flexible, as it can approximate arbitrarily close any RUM models (Fosgerau et al., 2013). The network GEV model proposed in Daly and Bierlaire (2006) further generalizes the cross-nested model by proving a way to construct a GEV CPGF based on any rooted network of correlation structures.

Apart from the GEV family, the **MMNL model** (McFadden and Train, 2000) is also popular due to its flexibility in capturing utility correlation. In the MMNL model, the model parameters (and the utilities v_j) are assumed to be random, and the choice probabilities can be obtained by taking the expectations over random coefficients. Let $\mathbf{Y}^1, \ldots, \mathbf{Y}^K$ be K realizations sampled from the distribution of the random parameters, the choice probabilities can be approximated as

$$P(j|\mathbf{Y}^1, \ldots, Y^K, G) = \frac{1}{K} \sum_{k=1}^{K} \frac{Y_j^k}{\sum_{t\in[m]} Y_t^k}.$$

The MMNL model is highly preferred in practice due to its flexibility in modeling people demand. However, the estimation and application of this model in decision-making are well-known to be expensive and complicated, due to the fact that it requires simulation to approximate the choice probabilities.

2.2 The Maximum Capture Problem

We are interested in the situation that a "*newcomer*" firm wants to locate new facilities in a competitive market, i.e., there are already existing facilities from competitors that can serve customers. The firm may want to maximize the expected market share achieved by attracting customers to new facilities. To capture the customers' demand, we suppose that a customer selects a facility according to a RUM model. In this context, each customer would associate each facility with a random utility and we assume that the customer will choose a facility by maximizing his/her utilities. Accordingly, the firm aims at selecting a set of locations to locate new facilities to maximize the expected number of customers. In the following, we describe in detail the formulation of the MCP under GEV models.

We denote by $\mathcal{V} = [m]$ the set of possible locations. Let I be the set of geographical zones where customers are located and q_i is the number of customers in zone $i \in I$ and for customers at zone i, let v_{ij} be the corresponding deterministic utility of location $j \in [m]$. These utility values can be inferred by estimating the RUM model using historical data. The set I can be viewed as a set of customer types, e.g., customers that belong to different categories specified by, for instance, age or income. A GEV model for customers located at zone $i \in I$ is determined by a CPGF $G^i(\mathbf{Y}^i)$, where $\mathbf{Y}^i$ is a vector of size m with entries $Y_j^i = e^{v_{ij}}$.

Under a GEV model specified by a set of CPGF $G^i(\mathbf{Y}^i)$, $i \in I$, taking into consideration the

competitors, the choice probability of a location $j \in [m]$ is given as

$$P(j|\mathbf{Y}^i, G^i) = \frac{Y_j \partial G^i_j(\mathbf{Y}^i)}{1 + G^i(\mathbf{Y}^i)}.$$

Here, without loss of generality, we assume that the total utility of the competitor is 1 for the sake of simplicity, as if it is not the case, then we always can scale the utilities $\mathbf{Y}^i$ to get utilities of 1 for the competitors. More specifically, it is possible due to fact that, for any $\alpha > 0$,

$$\frac{Y_j \partial G^i_j(\mathbf{Y}^i)}{\alpha + G^i(\mathbf{Y}^i)} \overset{(a)}{=} \frac{\frac{Y_j}{\alpha} \partial G^i_j(\mathbf{Y}^i)}{1 + G^i(\mathbf{Y}^i/\alpha)} \overset{(b)}{=} \frac{\frac{Y_j}{\alpha} \partial G^i_j(\mathbf{Y}^i/\alpha)}{1 + G^i(\mathbf{Y}^i/\alpha)}$$

where (a) is due the homogeneity of $G^i(\cdot)$ ((ii) of Remark 1) and (b) is obtained by taking derivatives of the both sides of the equation $G^i(\alpha \mathbf{Y}^i) = \alpha G^i(\mathbf{Y}^i)$ w.r.t. Y^i_j

$$\alpha \partial G^i_j(\alpha \mathbf{y}^i) = \alpha \partial G^i_j(\mathbf{Y}^i), \text{ or } \partial G^i_j(\alpha \mathbf{y}^i) = \partial G^i_j(\mathbf{Y}^i), \text{ for any } \alpha > 0.$$

We are interested in the fact that the facilitates are located at a subset of locations $S \subset [m]$. Hence, the conditional choice probability can be written as

$$P(j|\mathbf{Y}^i, G^i, S) = \frac{Y^i_j \partial G^i_j(\mathbf{Y}^i|S)}{1 + G^i(\mathbf{Y}^i|S)}, \ \forall j \in S,$$

where the conditional CPGF $G^i(\mathbf{Y}^i|S)$ can be computed as $G^i(\mathbf{Y}^i|S) = G^i(\widetilde{\mathbf{Y}}^i)$, where $\widetilde{\mathbf{Y}}^i$ is a vector of size m with entries $\widetilde{Y}^i_j = Y^i_j$ if $j \in S$ and $\widetilde{Y}^i_j = 0$ otherwise. This can be interpreted as if a location j is not in S, then its corresponding utility becomes very small, i.e., $v_{ij} = -\infty$, then $Y^i_j = e^{v_{ij}} = 0$. The maximum capture problem under a GEV model specified by CPGFs $G^i(\mathbf{Y}^i)$, $i \in I$, can be stated as

$$\max_{S \in \mathcal{S}} \left\{ f^{\text{GEV}}(S) = \sum_{i \in I} q_i \sum_{j \in S} P(j|\mathbf{Y}^i, G^i, S) \right\}, \tag{2}$$

where $\mathcal{S}$ is the set of feasible solutions. Under a cardinality constraint $|S| \leq C$, $\mathcal{S}$ can be defined as $\mathcal{S} = \{\mathcal{S} \subset [m] | \ |S| \leq C\}$, for a given constant C such that $1 \leq C \leq m$. Note that the objective function can be further simplified as

$$\begin{aligned} f^{\text{GEV}}(S) &= \sum_{i \in I} q_i \frac{\sum_{j \in S} Y^i_j \partial G^i_j(\mathbf{Y}^i|S)}{1 + G^i(\mathbf{Y}^i|S)} \\ &\overset{(a)}{=} \sum_{i \in I} q_i - \sum_{i \in I} \frac{q_i}{1 + G^i(\mathbf{Y}^i|S)}, \end{aligned}$$

where (a) is due to Property (v) in Remark 1.

If the choice model is MNL, the objective function becomes

$$f^{\text{MNL}}(S) = \sum_{i \in I} q_i - \sum_{i \in I} \frac{q_i}{1 + \sum_{j \in m} Y_j^i},$$

and from previous studies, we know that $f^{\text{MNL}}(S)$ is submodular (Benati and Hansen, 2002). Thus, an approach based on sub-gradient and submodular cuts can be used (Ljubić and Moreno, 2018, Mai and Lodi, 2020) to efficiently solve the problem. In general, formulations based on GEV models would be much more complicated. For example, under the nested logit model, we can write the objective function as

$$f^{\text{GEV}}(S) = \sum_{i \in I} q_i - \sum_{i \in I} \frac{q_i}{1 + \sum_{l \in [L]} \left(\sum_{j \in n_l \cap S} (Y_j^i)^{\mu_l} \right)^{1/\mu_l}}.$$

Under a more general case, e.g., the network GEV model (Daly and Bierlaire, 2006, Mai et al., 2017), it is even not possible to write the objective function in a closed form.

It is important to note that, if we look at the objective function under a MMNL model

$$\begin{aligned} f^{\text{MMNL}}(S) &= \frac{1}{K} \sum_{k \in [K]} \left(\sum_{i \in I} q_i - \sum_{i \in I} \frac{q_i}{1 + \sum_{j \in m} Y_j^{i,k}} \right) \\ &= \sum_{k \in [K], i \in I} \frac{q_i}{K} - \sum_{i \in I, k \in [K]} \frac{q_i / K}{1 + \sum_{j \in m} Y_j^{i,k}}, \end{aligned}$$

where $\mathbf{Y}^{i,k}$, $k = 1, ..., K$, are K realization of the random utility vector $\mathbf{Y}^i$, then we see that this objective function can be viewed as one from the MNL-based MCP problem with $K \times |I|$ customer zones, in which there are q_i/K customers in zone (i,k)-th. So, all the results established for the MNL (and GEV in general) problem can also be used to solve the MMNL problem.

3 Maximum Capture Problem under GEV Models

In this section, we explore the MCP under GEV models. In particular, by leveraging the properties of the GEV CPGFs shown above, we show that the objective function in the context is monotonic and submodular. This generalizes some well-known results established for MNL-based problems in previous studies (Benati, 1996, Benati and Hansen, 2002). We then present our local search procedure to efficiently solve the problem.

3.1 Monotonicity and Submodularity

We show two key results of the paper, which indicate that the objective function under any GEV model is monotonic and submodular. As a result, we show that it is possible to obtain a

$(1-1/e)$ approximation solution using a simple greedy heuristic procedure.

To prove the results, we first formulate the MCP as a binary program. That is, given a subset $S \subset [m]$, let $\mathbf{x}^S$ be a binary vector of size m with entries $\mathbf{x}_j^S = 1$ if $j \in S$ and $\mathbf{x}_j^S = 0$ otherwise. We see that the conditional CPGF can be written as

$$G^i(\mathbf{Y}^i|S) = G^i(\mathbf{x}^S \circ \mathbf{Y}^i),$$

where $\circ$ is the element-by-element operator and $\mathbf{x} \circ \mathbf{Y}^i$ is vector of size m with entries $x_j^S Y_j^i$, $j = 1, \ldots, m$. We now can formulate (2) as

$$\max_{\mathbf{x} \in X} \left\{ f^{\text{GEV}}(\mathbf{x}) = \sum_{i \in I} q_i - \sum_{i \in I} \frac{q_i}{1 + G^i(\mathbf{x} \circ \mathbf{Y}^i)} \right\}, \tag{3}$$

where $X = \{\mathbf{x}^S \in \{0,1\}^m | \ \forall S \in \mathcal{S}\}$, i.e., the feasible set of binary solutions that corresponds to all the subsets in $\mathcal{S}$. It is worth noting that if the choice model is MNL (or MMNL), then the objective $f^{\text{GEV}}(\mathbf{x})$ is concave in $\mathbf{x}$ and the problem can be handled efficiently by an outer-approximation method (Bonami et al., 2011, Ljubić and Moreno, 2018, Mai and Lodi, 2020). It is however not the case under an arbitrary GEV model.

The following proposition tells us that the objective function is monotonic, which implies that adding more facilities always yields better objective values.

Proposition 1 (Monotonicity) *Adding more facilities always yields better objective values, i.e., $f^{\text{GEV}}(S \cup \{i\}) > f^{\text{GEV}}(S)$ for any $i \notin S$.*

Proof. To prove the claim, let $\mathbf{x} \in \{0,1\}^m$ be the binary vector representing set S. For any $j \in [m]$ such that $x_j = 0$, we need to prove $f^{\text{GEV}}(\mathbf{x} + \mathbf{e}^j) > f^{\text{GEV}}(\mathbf{x})$, where $\mathbf{e}^j$ is a vector of size m with zero entries except the j-th element that is equal to 1. To prove this, let us consider $G^i(\mathbf{x} \circ \mathbf{Y}^i)$. Taking the derivative of $G^i(\mathbf{x} \circ \mathbf{Y}^i)$ w.r.t. an x_j, $j \in [m]$, we have

$$\frac{\partial G^i(\mathbf{x} \circ \mathbf{Y}^i)}{\partial x_j} = Y_j^i \partial G_j^i(\mathbf{x} \circ \mathbf{Y}^i) \overset{(b)}{>} 0, \tag{4}$$

where (b) is due to Property (iv) of Remark 1. This implies that $G^i(\mathbf{x} \circ \mathbf{Y}^i)$ is (strictly) monotonic increasing in any x_j, $j \in [m]$. Thus

$$G^i((\mathbf{x} + \mathbf{e}^j) \circ \mathbf{Y}^i) > G^i(\mathbf{x} \circ \mathbf{Y}^i). \tag{5}$$

Together with the definition of $f^{\text{GEV}}(\mathbf{x})$ in (3), we have $f^{\text{GEV}}(\mathbf{x} + \mathbf{e}^j) > f^{\text{GEV}}(\mathbf{x})$ as desired. ■

Since $f^{\text{GEV}}(S)$ is monotonic, if we consider a cardinality constraint $|S| \leq C$ for a scalar $C \in \{1, \ldots, m\}$, then an optimal solution S^* always achieves the maximum cardinality, i.e., $|S^*| = C$. Thus, we can replace the cardinality constraint by an equality one, i.e., $|S| = C$. Note that

a similar claim has been validated for the MNL-based problems in prior work (Mai and Lodi, 2020).

The submodularity is well-known for the objective function under the MNL model (Benati, 1996, Benati and Hansen, 2002). The theorem below shows that it is also the case under any models in the GEV family.

Theorem 2 (Submodularity) *$f^{\text{GEV}}(S)$ is submodular.*

Proof. To prove the submodularity, we will show that for any set $A \subset B \subset [m]$ and for any $j \in [m] \backslash B$ we have

$$f^{\text{GEV}}(A \cup \{j\}) - f^{\text{GEV}}(A) \geq f^{\text{GEV}}(B \cup \{j\}) - f^{\text{GEV}}(B) \tag{6}$$

Let use denote each component of $f^{\text{GEV}}(S)$ as

$$g^i(S) = \frac{q_i}{1 + G^i(\mathbf{Y}^i|S)}, \forall i \in I$$

then (6) can be validated if we can prove

$$g^i(A \cup \{j\}) - g^i(A) \leq g^i(B \cup \{j\}) - g^i(B),$$

or equivalently,

$$\begin{aligned} &\frac{1}{1 + G^i((\mathbf{x}^A + \mathbf{e}^j) \circ \mathbf{Y}^i)} - \frac{1}{1 + G^i(\mathbf{x}^A \circ \mathbf{Y}^i)} \leq \frac{1}{1 + G^i((\mathbf{x}^A + \mathbf{e}^j) \circ \mathbf{Y}^i)} - \frac{1}{1 + G^i(\mathbf{x}^A \circ \mathbf{Y}^i)} \\ \Leftrightarrow\ &\frac{G^i((\mathbf{x}^A + \mathbf{e}^j) \circ \mathbf{Y}^i) - G^i((\mathbf{x}^A) \circ \mathbf{Y}^i)}{[1 + G^i((\mathbf{x}^A + \mathbf{e}^j) \circ \mathbf{Y}^i)][1 + G^i(\mathbf{x}^A \circ \mathbf{Y}^i)]} \geq \frac{G^i((\mathbf{x}^B + \mathbf{e}^j) \circ \mathbf{Y}^i) - G^i(\mathbf{x}^B \circ \mathbf{Y}^i)}{[1 + G^i((\mathbf{x}^B + \mathbf{e}^j) \circ \mathbf{Y}^i)][1 + G^i(\mathbf{x}^B \circ \mathbf{Y}^i)]} \end{aligned} \tag{7}$$

Now, for ease of notation, for any $\mathbf{x} \in \{0,1\}^m$ and $j \in [m]$ such that $x_j = 0$, let

$$\phi(\mathbf{x}) = G^i((\mathbf{x} + \mathbf{e}^j) \circ \mathbf{Y}^i) - G^i(\mathbf{x} \circ \mathbf{Y}^i)$$

For any $k \in [m]$ such that $x_k = 0$ and $k \neq j$ we take the partial derivative of $\phi(\mathbf{x})$ w.r.t. x_k and get

$$\frac{\partial \phi(\mathbf{x})}{\partial x_k} = Y^i_k \left(\partial G^i_k((\mathbf{x} + \mathbf{e}^j) \circ \mathbf{Y}^i) - \partial G^i_k(\mathbf{x} \circ \mathbf{Y}^i)\right) \tag{8}$$

Now, let define another function $\rho(\mathbf{x}) = \partial G^i_k(\mathbf{x} \circ \mathbf{Y}^i)$. Taking the partial derivative of $\rho(\mathbf{x})$ w.r.t. x_j we get

$$\frac{\partial \rho(\mathbf{x})}{\partial x_j} = Y^i_j \partial G^i_{kj}(\mathbf{x} \circ \mathbf{Y}^i),$$

and since $\partial G^i_{kj}(\mathbf{x} \circ \mathbf{Y}^i) \leq 0$ (Property (iv) of Remark 1), we have $\partial \rho(\mathbf{x})/\partial x_j \leq 0$. Thus, $\rho(\mathbf{x})$

is monotonic decreasing in x_j, which implies

$$\partial G_k^i((\mathbf{x}+\mathbf{e}^j)\circ\mathbf{Y}^i) - \partial G_k^i(\mathbf{x}\circ\mathbf{Y}^i) \leq 0. \tag{9}$$

Combine (8) and (9) we have $\partial\phi(\mathbf{x})/\partial x_k \leq 0$. Thus, $\phi(\mathbf{x})$ is monotonic decreasing in x_k, leading to the inequality

$$G^i((\mathbf{x}+\mathbf{e}^j)\circ\mathbf{Y}^i) - G^i(\mathbf{x}\circ\mathbf{Y}^i) \geq G^i((\mathbf{x}+\mathbf{e}^j+\mathbf{e}^k)\circ\mathbf{Y}^i) - G^i((\mathbf{x}+\mathbf{e}^k)\circ\mathbf{Y}^i).$$

Consequently, we have

$$G^i((\mathbf{x}^A+\mathbf{e}^j)\circ\mathbf{Y}^i) - G^i((\mathbf{x}^A)\circ\mathbf{Y}^i) \geq G^i((\mathbf{x}^B+\mathbf{e}^j)\circ\mathbf{Y}^i) - G^i((\mathbf{x}^B)\circ\mathbf{Y}^i), \tag{10}$$

for any $A \subset B \subset [m]$ and $j \notin B$. Moreover, using (5) from the proof of Proposition 1, since $A \subset B$, we have

$$G^i(\mathbf{x}^A\circ\mathbf{Y}^i) \leq G^i(\mathbf{x}^B\circ\mathbf{Y}^i)$$
$$G^i((\mathbf{x}^A+\mathbf{e}^j)\circ\mathbf{Y}^i) \leq G^i((\mathbf{x}^B+\mathbf{e}^j)\circ\mathbf{Y}^i).$$

Thus,

$$[1+G^i((\mathbf{x}^A+\mathbf{e}^j)\circ\mathbf{Y}^i)][1+G^i(\mathbf{x}^A\circ\mathbf{Y}^i)] \leq [1+G^i((\mathbf{x}^B+\mathbf{e}^j)\circ\mathbf{Y}^i)][1+G^i(\mathbf{x}^B\circ\mathbf{Y}^i)] \tag{11}$$

Combine (10) and (11) we obtain (7) and then (6) as desired.

■

Theorem 2 shows the submodularity of the MCP problem under any GEV models. Together with the fact that $f^{\text{GEV}}(S)$ is monotonic (Proposition 1), the results from (Nemhauser et al., 1978) imply that a simple greedy heuristic algorithm can secure a $(1-1/e)$ approximation solution, i.e., a greedy can return a solution $\overline{S}$ such that $f^{\text{GEV}}(\overline{S}) \geq (1-1/e)\max_{S\in\mathcal{S}} f^{\text{GEV}}(S)$. Such an algorithm can start from a null set and keep adding locations one at a time, taking at each step a location that increases the objective function the most, and stops when the maximum capacity $|S| = C$ is reached. We state this important result in the following corollary.

Corollary 3 (Performance guarantee for a greedy heuristic) *Under a cardinality constraint, a greedy heuristic algorithm can guarantee a* $(1-1/e)$ *approximation solution.*

3.2 Gradient-based Local Search

Due to the submodularity, a greedy heuristic can guarantee a $(1-1/e)$ approximation solution. In this section, we design a new local search procedure to further improve such a greedy solution. Our approach is motivated by the fact that the objective function $f^{\text{GEV}}(\mathbf{x})$ is differentiable, suggesting that we could use gradient information to direct the search. The general idea to

design an iterative procedure, where at each step we build a model function (linear or quadratic) to approximate the objective function using gradient and/or Hessian information. We then maximize the model function to find a new iterate. A key component of our approach is that the model function can be only an adequate representation of the objective function in a local neighbourhood of the current solution. Thus, we only maximize the model function within a restricted region. This approach is inspired by the trust-region method widely used in continuous optimization (Conn et al., 2000).

To start our exposition, let us define a model function based on Taylor series built around a solution candidate $\overline{\mathbf{x}}$

$$f^{\text{GEV}}(\mathbf{x}) \approx f^{\text{GEV}}(\overline{\mathbf{x}}) + \nabla f^{\text{GEV}}(\overline{\mathbf{x}})^{\text{T}}(\mathbf{x} - \overline{\mathbf{x}}) + \frac{1}{2}(\mathbf{x} - \overline{\mathbf{x}})^{\text{T}}\mathbf{B}(\mathbf{x} - \overline{\mathbf{x}}),$$

where $\mathbf{B}$ is the Hessian matrix or an approximation of it at $\overline{\mathbf{x}}$. In our context, the Hessian can be computed easily, but maximizing the model function will involve solving a binary quadratic maximization problem, which is expensive. Thus, we set $\mathbf{B}_k = 0$. In other words, we use a linear model function to approximate $f^{\text{GEV}}(\mathbf{x})$.

At each iteration, we need to solve the following sub-problem

$$\begin{aligned}
\max_{\mathbf{x}} \quad & \nabla f^{\text{GEV}}(\overline{\mathbf{x}})^{\text{T}}\mathbf{x} && \text{(P1)}\\
\text{subject to} \quad & \sum_{j\in[m]} x_j = C && (12)\\
& \sum_{j\in[m]} |x_j - \overline{x}_j| \le \Delta && (13)\\
& \mathbf{x} \in \{0,1\}^m
\end{aligned}$$

where (12) is the cardinality constraint, and (13) is to ensure that the new solution candidate is within a region of size Δ around $\overline{\mathbf{x}}$. Note that (13) can be linearized as

$$\sum_{j\in[m],\overline{x}_j=1} (1 - x_j) + \sum_{j\in[m],\overline{x}_j=0} x_j \le \Delta,$$

so as (P1) becomes a integer linear program, which can be handled by an MILP solver. In the following, we will look closely to (P1) and show that it can be solved to optimality in polynomial time.

Solving Subproblems. We further look into the subproblem of the gradient-based local search (P1) to design an efficient algorithm to solve it. To facilitate our exposition, we first note that the constraint $\sum_{j\in[m]} |\overline{x}_j - x_j| \le \Delta$ implies that there are at most $\Delta/2$ locations that either appears in S or in $\overline{S}$, but not in both, where $S, \overline{S}$ are the subsets representing $\mathbf{x}$ and $\overline{\mathbf{x}}$, respectively. For this reason, Δ should be integer and even, and the constraint $\sum_{j\in[m]} |\overline{x}_j - x_j| \le \Delta$ is equivalent to $|S \triangle \overline{S}| \le \Delta$, where $\triangle$ is the symmetric difference operator, i.e., $S \triangle \overline{S} = (S \backslash \overline{S}) \cup (\overline{S} \cup S)$.

We therefore can rewrite (P1) as

$$\max_{S\subset[m]} \quad \sum_{j\in S} d_j \tag{P2}$$

$$\text{subject to} \quad |S| = C \tag{14}$$

$$|S \triangle \overline{S}| \leq \Delta, \tag{15}$$

where $\overline{S} \subset [m]$ is the subset that corresponds to the binary vector $\overline{\mathbf{x}}$ and $d_j = \nabla f^{\text{GEV}}(\overline{x})_j$, $j \in [m]$. Under the cardinality constraint $|S| = C$, we see that $|S\backslash\overline{S}| = |\overline{S}\backslash S| = |S \triangle \overline{S}|/2$. The following proposition shows that d_j are non-negative, for all $j \in [m]$.

Proposition 2 *All the coefficients of the objective function of* (P2) *are non-negative.*

Proof. We prove the claim by showing that, for any $\mathbf{x} \in [0,1]^m$, $\nabla_{\mathbf{x}} f^{\text{GEV}}(\mathbf{x}) \geq 0$. Given $j \in [m]$, taking the derivative of $f^{\text{GEV}}(\mathbf{x})$ w.r.t. x_j we have

$$\begin{aligned}\frac{\partial f^{\text{GEV}}(\mathbf{x})}{\partial x_j} &= \sum_{i\in I} \frac{\partial G^i(\mathbf{x}\circ\mathbf{Y}^i)}{\partial x_j} \frac{q_i}{(1+G^i(\mathbf{x}\circ\mathbf{Y}^i))^2} \\ &= \sum_{i\in I} \frac{q_i Y_j^i \partial G_j^i(\mathbf{x}\circ\mathbf{Y}^i)}{(1+G^i(\mathbf{x}\circ\mathbf{Y}^i))^2} \geq 0 \end{aligned} \tag{16}$$

where (16) is due to the fact that $\partial G_j^i(\mathbf{x}\circ\mathbf{Y}^i) > 0$ (Property (iv) of Remark 1). We obtain the desired inequality. ■

In Algorithm 1 we describe our main steps to solve (P2). In Step 1, we find $\Delta/2$ smallest coefficients d_j in $\overline{S}$ and $\Delta/2$ largest coefficients d_j in $[m]\backslash\overline{S}$. This is motivated by the fact we only seek subsets generated by exchanging at most $\Delta/2$ elements in $\overline{S}$ with some outside $\overline{S}$. Thus, to obtain the best objective values, we should exchange elements of lowest coefficients in $\overline{S}$ with those of highest coefficients in $[m]\backslash\overline{S}$. In the second step, $\gamma(t)$ represents the best gain obtained by exchanging t elements, and in the third step we just select the best $\gamma(t)$ to get the best solution. Proposition 3 below shows that Algorithm 1 will efficiently return an optimal solution to (P2).

Proposition 3 *Algorithm 1 returns an optimal solution to* (P1) *with complexity* $\mathcal{O}(m\Delta/2)$.

Proof. To prove the convergence, we let S be a feasible solution of (P2), i.e., $|S| = C$ and $|S \triangle \overline{S}| \leq \Delta$. We will prove that $\sum_{j\in S} d_j \leq \sum_{j\in S^*} d_j$, where S^* is the solution returned. from Algorithm 1. Let $t = |S \triangle \overline{S}|/2$, then we know that S can be obtained by exchanging t elements between S and $\overline{S}$. Let $\pi_1^1, \ldots, \pi_t^1$ be the indexes of t elements in $\overline{S}$ that are exchanged with t

Algorithm 1: Solving sub-problems

 # Step 1: Take smallest coefficients in $\overline{S}$ and largest coefficients in $[m]\backslash\overline{S}$

Choose $\sigma_1^1, \ldots, \sigma_{\Delta/2}^1 \in \overline{S}$ and $\sigma_1^2, \ldots, \sigma_{\Delta/2}^2 \in [m]\backslash\overline{S}$ such that

$$d_{\sigma_1^1} \leq d_{\sigma_2^1} \leq \ldots \leq d_{\sigma_{\Delta/2}^1} \leq \min_{j \in \overline{S}\backslash\{\sigma_1^1, \ldots, \sigma_{\Delta/2}^1\}} d_j$$

$$d_{\sigma_1^2} \geq d_{\sigma_2^2} \geq \ldots \geq d_{\sigma_{\Delta/2}^2} \geq \max_{j \in [m]\backslash\overline{S}\backslash\{\sigma_1^2, \ldots, \sigma_{\Delta/2}^2\}} d_j$$

 # Step 2: Select the best set for each local region size $|S \triangle \overline{S}| = 2t$, for $t = 1, \ldots, \Delta/2$

for $t = 1, \ldots, \Delta/2$ **do**

$$\gamma(t) = \sum_{h=1}^{t} \left(d_{\sigma_h^2} - d_{\sigma_h^1} \right)$$

Select $t^* = \text{argmax}_{t=1,\ldots,\Delta/2} \gamma(t)$

 # Step 3: Return the best solution within the local region $|S \triangle \overline{S}| \leq \Delta$

Return

$$S^* \leftarrow \overline{S} \cup \{\sigma_1^2, \ldots, \sigma_{t^*}^2\} \backslash \{\sigma_1^1, \ldots, \sigma_{t^*}^1\}$$

elements in t, indexed as $\pi_1^2, \ldots, \pi_t^2$. We have

$$\begin{aligned}
\sum_{j \in S} d_j &= \sum_{j \in \overline{S}} d_j - \sum_{h=1}^{t} d_{\pi_h^1} + \sum_{h=1}^{t} d_{\pi_h^2} \\
&\overset{(c)}{\leq} \sum_{j \in \overline{S}} d_j - \sum_{h=1}^{t} d_{\sigma_h^1} + \sum_{h=1}^{t} d_{\sigma_h^2} \\
&= \sum_{j \in \overline{S}} d_j + \gamma(t) \\
&\overset{(d)}{\leq} \sum_{j \in \overline{S}} d_j + \gamma(t^*) = \sum_{j \in S^*} d_j,
\end{aligned}$$

where (c) is due to the way we select σ_h^1 and σ_h^2, $h = 1, \ldots, \Delta/2$, and (d) is due to the way t^* is selected. This implies that S^* is an optimal solution to (P2), as desired.

For the complexity, we see that Step 1 would take $\mathcal{O}(\Delta/2|\overline{S}| + \Delta/2(m - |\overline{S}|)) = \mathcal{O}(m\Delta/2)$. Step 2 would require $\mathcal{O}(\Delta^2/4)$, which would be much smaller than $\mathcal{O}(m\Delta/2)$. Adding all together, the complexity of Algorithm 1 is $\mathcal{O}(m\Delta/2)$. ■

3.3 GGX Algorithm

Our main algorithm consists of three main phases. In the first phase (warm-up), we perform a greedy heuristic, which can be done by starting from the null set and adding locations one at a time, taking at each step the location that increases $f^{\text{GEV}}(\cdot)$ the most. This phase finishes when we reach the maximum capacity, i.e., $|S| = C$. After this phase, due to the submodularity, it is

guaranteed that the obtained solution yields at least a factor $(1-1/e)$ times the optimal value. In the second phase, we iteratively solve the sub-problem (P1) to seek better solutions. This phase ends when we cannot find any better solutions. In the last phase, we further enhance the solution obtained by performing a simple greedy local search based on exchanging some locations in a candidate solution S with some others from $[m]\backslash S$. We describe the three phases in detail in Algorithm 2.

Algorithm 2: GGX algorithm

*# 1: **G**reedy heuristics (warm up step)*
$S = \emptyset$
for $j = 1, \ldots, C$ **do**
  $j^* = \text{argmax}_{j \in [m]\backslash S} f^{\text{GEV}}(S \cup \{j\})$
  $S \leftarrow S \cup \{j^*\}$
*# 2: **G**radient-based local search*
$k = 0; S^0 = S$
do
  Solve (P1) based on a local region around $\mathbf{x}^{S^k}$ to get a new solution candidate $\overline{S}$
  if $f^{\text{GEV}}(\overline{S}) > f^{\text{GEV}}(S^k)$ **then**
    $S^{k+1} \leftarrow \overline{S}$
  else
    $S^{k+1} = S^k$
  $k \leftarrow k + 1$
until $S^k = S^{k-1}$;
*# 3: **Ex**changing phase*
do
  $(j^*, t^*) = \text{argmax}_{\substack{j \in S \\ t \in [m]\backslash S}} \left\{ f^{\text{GEV}}(S^k \cup \{t\} \backslash \{j\}) \right\}$
  $\overline{S} = S^k \cup \{k^*\} \backslash \{j^*\}$
  if $f^{\text{GEV}}(\overline{S}) > f^{\text{GEV}}(S^k)$ **then**
    $S^{k+1} \leftarrow \overline{S}$
  else
    $S^{k+1} = S^k$
  $k \leftarrow k + 1$
until $S^k = S^{k-1}$;
Return S^k.

In the context of assortment optimization under GEV models, Mai and Lodi (2019) also propose a gradient-based local search (named as Binary Trust Region - BiTR) procedure to solve the binary nonlinear formulation of the assortment optimization problem. There are major differences between Algorithm 2 and the one proposed in Mai and Lodi (2019). First, our algorithm starts with a greedy heuristic that guarantees a $(1-1/e)$ approximation solution, while there is no performance guarantee for the BiTR. Second, we explore the structure of the MCP under the GEV family, e.g., the coefficients of the sub-problem's objective function are non-negative and we only care about fixed-size subsets, to build a more efficient method to solve the sub-problems.

4 Numerical Experiments

In this section, we provide experimental results to compare our GGX algorithm with existing approaches. We use three datasets from recent literature (Ljubić and Moreno, 2018, Mai and Lodi, 2020) and generate instances under three popular discrete choice models, i.e., the MNL, MMNL, and nested logit models.

4.1 Experimental Settings

We will compare our algorithm with the standard greedy heuristic (GH - Step 1 of our GGX algorithm), the multicut, and singlecut outer-approximation algorithms (MOA and OA) (Mai and Lodi, 2020), and the Branch-and-Cut (BC) (Ljubić and Moreno, 2018). Note that for MNL and MMNL instances, it is possible to formulate the MCP as a MILP and solve it by an MILP solver (e.g. IBM's CPLEX). However, according to prior work (Ljubić and Moreno, 2018, Mai and Lodi, 2020), this approach is generally outperformed by the MOA and BC methods. Thus, we do not include the MILP solver in our experiments. In fact, for MNL and MMNL instances, the OA, MOA, and BC are exact algorithms, thus if they stop within the time budget, the solutions obtained are optimal. For nested logit instances, we only compare the GGX with GH, OA, MOA approaches, as BC is not designed to handle such instances. Since the objective function is highly non-concave in the context of the nested logit, all the approaches are heuristic, but only the GGX and GH can provide solutions with a performance guarantee.

To enhance the third step of GGX, we allow the algorithm to exchange one or two locations at a time. That is, in the last phase of GGX, we first iteratively exchange one location at a time until we cannot further improve the objective function, as described in Algorithm 2. We then iteratively exchange two locations at a time until getting no further improvements.

We use the following three datasets as benchmark instances and we refer the reader to Freire et al. (2015) for more detailed descriptions. These datasets have been also used in some recent MCP studies (Ljubić and Moreno, 2018, Mai and Lodi, 2020).

- **HM14**: The dataset includes 15 problems generated randomly in a plane, with $|I| \in \{50, 100, 200, 400, 800\}$ and $m \in \{25, 50, 100\}$.
- **ORlib**: The dataset includes 11 problems where there are four instances with $(|I|, m) = (50, 25)$, four instances with $(|I|, m) = (50, 50)$ and three instances with $(|I|, m) = (1000, 100)$.
- **PR-NYC** (or **NYC**): the dataset comes from a large-scale park-and-ride location problem in New York city with $|I| = 82341$ and $m = 59$. As reported in previous studies, these are the largest and most challenging instances.

We employ the same settings of parameters as in previous studies (Ljubić and Moreno, 2018, Mai and Lodi, 2020). The number of facilities that need to be opened C is varied from 2 to 10. The

deterministic part of the utility is defined as $v_{ij} = -\beta c_{ij}$ for a location $j \in M$ and $v_{ij'} = -\beta\alpha c_{ij'}$ for each competitor j', where c_{ij} is the distance between zone/client $i \in I$ and location $j \in [m]$, the parameter β is the sensitivity of customers about the perceived utilities and α represents the competitiveness of the competitors. These parameters are chosen as $\alpha = \{0.01, 0.1, 1\}$ and $\beta = \{1, 5, 10\}$ for datasets HM14 and ORlib, and $\alpha = \{0.5, 1, 2\}$ and $\beta = \{0.5, 1, 2\}$ for the NYC dataset. Therefore, for each discrete choice model chosen, each problem above has 81 different instances and the total numbers of instances for HM14, ORlib, NYC are 972, 891, 81, respectively.

The experiments are done on a PC with a processor AMD Ryzen 7-3700X CPU @ 3.80 GHz and 16 gigabytes of RAM. We use MATLAB 2020 to implement and run the algorithms, and we link to IBM ILOG-CPLEX 12.10 to solve MILPs under default settings. We also take the code used in Ljubić and Moreno (2018) to generate results for the MNL and MMNL instances with the BC approach.

4.2 Multinomial Logit - MNL

We take MNL instances from previous work (Ljubić and Moreno, 2018, Mai and Lodi, 2020) and report numerical results in Table 1 below, where each instance is given a CPU time budget of 600 seconds. Each row of the table corresponds to 81 instances and we indicate the largest number of instances solved with the best objective values in bold. We use the same settings as in Mai and Lodi (2020). We do not show the CPU times for GH as it runs very fast. The GH finishes 26/27 problems in less than 0.01 seconds and it just needs about 0.5 seconds to finish all the instances of the largest dataset (i.e. the NYC one). On the other hand, solutions obtained by GH are relatively good, in the sense that the percentage gaps between the objective values yielded by those solutions and the best objective values vary only from 0 to 2.94%. In terms of number of instances with the best objective values, GGX performs the best as it gives the largest numbers best objective values in 26/27 problems. Moreover, GGX solves 81/81 instances with the best objective values in 25/27 problems. On the other hand, GGX only requires short CPU times to finish (the average CPU times are always less than 1.5 seconds except for the NYC instances). Furthermore, when comparing GGX with the OA, MOA, and BC approaches, the average CPU times required by GGX are about 78 times lower than OA, 28 times lower than MOA, and 12 times lower than BC. For small instances with $|I| \leq 100$, BC achieves good performance. It provides the best objective values for all 81 instances of each problem with the lowest CPU times. However, for larger problem instances ($|I| > 100$), BC becomes more expensive, especially for the three large problems in the ORlib dataset with $(|I|, |M|) = (800, 100)$ and for the NYC instances (the average CPU times are always more than 90 seconds). The MOA has the best performance for the NYC dataset, as it only requires 2.32 seconds to return the best objective values for all the 81 instances. In general, among all the approaches considered, GGX achieves the best performance for the MNL instances.

Instance	$\|I\|$	m	# instances with best objective					Average CPU time (s)			
			GGX	GH	OA	MOA	BC	GGX	OA	MOA	BC
HM14	50	25	**81**	**81**	**81**	**81**	**81**	0.14	19.15	0.12	0.01
HM14	50	50	**81**	**81**	73	**81**	**81**	0,15	109.45	0.15	0.01
HM14	50	100	79	79	58	**81**	**81**	0.24	188.91	0.32	0.05
HM14	100	25	**81**	80	73	**81**	**81**	0.14	138.30	0.18	0.01
HM14	100	50	**81**	77	69	**81**	**81**	0.15	170.80	0.28	0.03
HM14	100	100	**81**	77	59	**81**	**81**	0.26	187.43	0.60	0.13
HM14	200	25	**81**	**81**	72	**81**	**81**	0.14	146.79	0.34	0.02
HM14	200	50	**81**	80	64	**81**	80	0.16	189.46	0.80	0.06
HM14	200	100	**81**	77	59	**81**	**81**	0.33	235.27	15.99	0.29
HM14	400	25	**81**	73	71	**81**	80	0.14	116.44	0,62	0.04
HM14	400	50	**81**	78	60	80	**81**	0.18	200.32	12.13	0.13
HM14	400	100	**81**	73	58	76	**81**	0.49	291.38	99.45	0.65
HM14	800	25	**81**	76	59	**81**	**81**	0.14	160.71	2.27	0.11
HM14	800	50	**81**	62	59	64	75	0.23	251.8	160.47	0.48
HM14	800	100	**80**	72	55	58	76	0.94	363.54	234.67	14.29
ORlib	50	25	**81**	**81**	**81**	**81**	**81**	0.14	0.21	0.20	0.01
ORlib	50	25	**81**	**81**	**81**	**81**	**81**	0.14	0.24	0.25	0.01
ORlib	50	25	**81**	**81**	**81**	**81**	**81**	0.14	0.19	0.23	0.01
ORlib	50	25	**81**	80	**81**	**81**	**81**	0.14	0.31	0.23	0.01
ORlib	50	50	**81**	74	**81**	**81**	**81**	0.15	0.37	0.31	0.02
ORlib	50	50	**81**	**81**	**81**	**81**	**81**	0.15	0.32	0.35	0.02
ORlib	50	50	**81**	80	**81**	**81**	**81**	0.15	0.38	0.34	0.02
ORlib	50	50	**81**	**81**	**81**	**81**	**81**	0.15	0.36	0.34	0.02
ORlib	1000	100	**81**	58	**81**	**81**	**81**	1.31	141.10	226.68	114.74
ORlib	1000	100	**81**	61	**81**	**81**	**81**	1.23	113.27	220.45	94.72
ORlib	1000	100	**81**	53	**81**	**81**	**81**	1.42	145.22	232.68	149.88
NYC	82341	59	**81**	72	77	**81**	80	33.87	164.01	2.32	161.71
Average			80.89	75.19	71.78	79.30	80.48				

Table 1: Numerical results for MNL instances, grouped by the problem name (81 instances per row).

To further assess the quality of the solutions obtained by GGX, we run the MOA (the best exact method for MNL instances) with a time limit of one hour. With this time budget, the MOA is able to return optimal solutions for all the instances. This also indicates that the GGX, even though requires only less than 1 second for most of the instances, is able to return optimal solutions for 2185/2187 instances and there are only 2 instances from datasets HM14 with $|I| = 50$ and $m = 100$ that it does not give optimal solutions. In these cases, the percentage gaps between the objective values given by the GGX and the optimal values are only less than 0.7%.

4.3 Mixed Logit - MMNL

In this section, we report numerical results for MMNL instances. To generate such instances, we assume that each utility v_{ij}, $i \in I, j \in [m]\}$, contains a random error component that follows a normal distribution of zero mean. We also assume that the variance of the random number is proportional to the distance c_{ij}. More precisely, each v_{ij} associated with customer zone (or client) $i \in I$ and location $j \in [m]$ is defined as $v_{ij} = -\theta c_{ij} + c_{ij}\tau_{ij}/3$, where τ_{ij} is a

standard normal random number. We also keep the utilities associated with the competitors deterministic. For each problem, we approximate the objective function by the Monte Carlo method. To do so, we choose a sample size $N = 100$ for the HM14 and ORlib datasets and $N = 10$ for the NYC one. For the latter, we only choose small N because the NYC problem is already large even under the MNL model. As mentioned, we consider and solve these MMNL instances as extended MNL ones, in which the number of customer zones is 5000, 5000 and 823,410 for instances from the HM14, ORlib, and NYC datasets, respectively. We also give a CPU time budget of 600 seconds for all instances.

In Table 2, for each problem, we report the number of instances with the best objective values and the average CPU times over 81 instances for five approaches, i.e., GGX, GH, OA, MOA, and BC. We indicate in bold the largest numbers of instances solved with the best objective values. The results clearly show that GGX generally outperforms the other approaches. More precisely, GGX manages to return the best objective values for all instances considered (i.e. 2187/2187 instances). Moreover, GH also performs very well, in the sense that the percentage gaps between the objective values given by GH and the best objective values only vary from 0% to 2.92%. In terms of CPU time, GH is still the fastest approach when it just requires less than 2 seconds to solve every instance except the NYC ones, which take only about 6.5 seconds in average. The GGX approach, even though being slower than the GH, but is still much faster than the others. We also observe that the OA, MOA, and BC approaches need much more time to solve MMNL instances, as compared to solving the MNL instances. The average CPU times required by these three approaches are more than 250 seconds. Overall, GGX dominates GH in terms of returned objective value and outperforms OA, MOA, and BC in terms of both returned objective value and CPU time.

Since the best objective values might be not the optimal ones, we further expand the time budget to see whether the GGX can return optimal solutions. More precisely, we run the BC (the best exact method) with a time budget of one hour and obtain optimal solutions for 1808/2187 instances. Interestingly, GGX also returns optimal solutions for all these instances, noting that GGX always terminates in a matter of seconds.

The fact that MMNL instances are more expensive to solve than MNL ones suggests that one can approximate MMNL instances by replacing the sample utilities by their mean values and treat these approximate instances as MNL instances. This would indeed reduce the computing cost but would lead to degradation in solution quality. To further explore this degradation, we compare the objective values obtained by solutions given by solving MNL approximate instances and the optimal values of the MMNL instances. To this end, for each MMNL instance, we solve its MNL-instance counterpart and obtain a solution S^{MNL}. The performance gap between this solution and the best solution (denoted as S^{MMNL}) obtained by directly solving the MMNL instance is computed as

$$\frac{f^{\text{MMNL}}(S^{\text{MMNL}}) - f^{\text{MMNL}}(S^{\text{MNL}})}{f^{\text{MMNL}}(S^{\text{MMNL}})},$$

where $f^{\text{MMNL}}(\cdot)$ is the objective function of the MCP under MMNL. We report these performance

Problem	$\|I\|$	m	# instances with best objective values					Average CPU time (s)				
			GGX	GH	OA	MOA	BC	GGX	GH	OA	MOA	BC
HM14	50	25	**81**	**81**	66	58	**81**	0.24	0.01	236.36	189.23	1.66
HM14	50	50	**81**	**81**	46	54	**81**	0.83	0.02	312.73	232.16	6.45
HM14	50	100	**81**	**81**	41	29	**81**	3.49	0.05	390.60	425.57	40.78
HM14	100	25	**81**	75	53	55	**81**	0.29	0.01	279.68	212.33	11.37
HM14	100	50	**81**	**81**	47	31	**81**	1.11	0.03	356.28	430.85	36.57
HM14	100	100	**81**	75	41	22	78	4.76	0.10	528.64	519.79	133.67
HM14	200	25	**81**	**81**	54	33	**81**	0.35	0.02	317.28	375.46	47.32
HM14	200	50	**81**	77	39	27	78	1.51	0.05	402.01	464.52	126.53
HM14	200	100	**81**	77	34	28	77	6.75	0.23	456.24	328.89	174.75
HM14	400	25	**81**	77	52	32	77	0.44	0.03	336.85	404.77	138.9
HM14	400	50	**81**	74	40	25	74	2.03	0.13	472.38	515.90	237.10
HM14	400	100	**81**	63	29	15	63	9.34	0.67	569.4	570.10	325.16
HM14	800	25	**81**	66	52	29	68	0.60	0.07	345.27	396.63	229.83
HM14	800	50	**81**	73	39	27	74	3.06	0.34	406.18	467.89	341.22
HM14	800	100	**81**	72	32	18	72	14.29	1.25	562.63	565.37	421.74
ORlib	50	25	**81**	70	57	54	77	0.24	0.01	388.27	485.02	276.30
ORlib	50	25	**81**	72	57	54	80	0.24	0.01	388.19	497.24	268.27
ORlib	50	25	**81**	66	50	51	78	0.24	0.01	365.41	457.82	218.40
ORlib	50	25	**81**	67	56	51	74	0.25	0.01	369.85	484.69	268.10
ORlib	50	50	**81**	**81**	33	38	**81**	0.84	0.02	459.78	619.15	425.71
ORlib	50	50	**81**	80	33	38	80	0.84	0.02	452.13	620.70	422.28
ORlib	50	50	**81**	**81**	33	35	**81**	0.83	0.02	459.68	643.83	435.96
ORlib	50	50	**81**	**81**	33	40	**81**	0.84	0.02	458.49	652.69	429.13
ORlib	1000	100	**81**	**81**	14	14	**81**	16.50	1.52	677.73	677.73	600.00
ORlib	1000	100	**81**	**81**	44	14	**81**	16.46	1.52	600.91	691.05	600.00
ORlib	1000	100	**81**	**81**	38	13	**81**	16.47	1.52	600.91	637.31	600.00
NYC	82341	59	**81**	72	75	68	72	114.42	6.50	178.18	135.24	381.48
Average			**81**	75.81	44.00	35.30	77.56					

Table 2: Numerical results for MMNL instances, grouped by the problem name (81 instances per row).

gaps in Table 3, where each row of the table corresponds to the average gap of MMNL instances with C varying from 2 to 10. The results show that the degradation is significant and the performance gaps can go up to quite high values (e.g. 14.48% for HM14 instances with $|I| = 800$ and $m = 100$). Note that while MMNL instances are expensive for OA, MOA and BC approaches, our local search algorithms (GGX and GH) can always solve them in less than 17 seconds, except for the NYC dataset.

Here we note that under the MNL and MMNL models, the MILP equivalent formulations can be used with an MILP solver (e.g. CPLEX or GUROBI) to exactly solve the MCP. Even-though from prior work, we know that this approach is much slower than other exact methods (MOA or BC), we try to feed the MILP solver with a good initial solution obtained from other methods, aiming to explore whether we can enhance the MILP approach by a good warm-up. We do this by giving the MILP solver (CPLEX) the solutions obtained by the GGX (the best method). We then observed that the MILP solver is either not able to stop within 600 seconds or not able to run for some large-size instances. In general, the MILP approach is not able to return any better solutions than the initial solutions we give to it.

Problem	$\|I\|$	m	Gap (%)
HM14	50	25	6.96
HM14	50	50	3.56
HM14	50	100	2.71
HM14	100	25	4.50
HM14	100	50	2.28
HM14	100	100	5.33
HM14	200	25	5.65
HM14	200	50	3.37
HM14	200	100	8.49
HM14	400	25	1.52
HM14	400	50	3.08
HM14	400	100	5.85
HM14	800	25	4.59
HM14	800	50	3.80
HM14	800	100	14.84
ORlib	50	25	2.52
ORlib	50	25	2.52
ORlib	50	25	2.52
ORlib	50	25	2.52
ORlib	50	50	5.76
ORlib	50	50	5.86
ORlib	50	50	5.76
ORlib	50	50	5.76
ORlib	1000	100	0.01
ORlib	1000	100	0.01
ORlib	1000	100	0.06
NYC	82341	59	0.35
Average			4.08

Table 3: Percentage gaps between objective values given by MNL approximate solutions the best objective values found by solving MMNL instances.

4.4 Nested Logit Model

This section reports numerical results for nested logit instances. We perform a comparison between 4 approaches, namely, the GGX, GH, OA, and MOA algorithms. We do not include the BC approach in this experiment, as it is not designed to handle nested logit instances. For the OA and MOA approaches, since it is quite straightforward to generate outer-approximation cuts using gradient information, we apply these algorithms to solve the nested logit instances to see how they perform. Note that, in the context, the objective function is no-longer concave, thus OA and MOA become heuristic with no performance guarantee, to the best of our knowledge. To generate nested logit instances, we build a customer nested logit model by partitioning the set of locations into $L = 5$ different and disjoint groups of equal size. In particular, the NYC dataset has 59 locations ($m = 59$), so for this problem we partition the locations into four groups with 12 locations and one with 11 locations. We choose the nested logit parameters as $\mu = (1.1, 1.2, 1.3, 1.4, 1.5)$, noting that more nests and/or other nested logit parameters can be chosen. Our selections here are just to illustrate the performance of different algorithms in handling GEV instances. We also give a time budget of 600 seconds for all the algorithms.

Table 4 reports comparison results of the four approaches. Each row of the table corresponds to 81 solved instances and we also indicate the largest numbers of instances solved with the best objective values in bold. The results clearly show that GGX outperforms the other approaches in terms of the number of instances solved with the best objective values. More precisely, GGX gives the best objective values for all problem instances while GH only performs the best for 9/27 problems. In terms of CPU time, GGX is not very fast. In particular, for the NYC instances, the average CPU time is about 355.36 seconds and is much larger than the average CPU times required by GH, OA, and MOA. The reason is that the objective function in this context is quite expensive to evaluate, as compared to the cases of the MMNL and MNL models, and the *exchanging* procedure of the GGX (Phase 3) requires calculating the objective function several times to find a pair of locations to swap. The GH is still very fast and the returned objective values are pretty close to the best values given by GGX. The percentage gaps between the objective values obtained from GH and the best objective values only vary from 0 to 3.32%. The OA and MOA approaches, even though run very fast, but give bad solutions most of the time. This can be explained by the fact that the objective function under a nested logit model is highly non-concave, thus a subgradient cut (or an outer-approximation cut) could potentially remove good solutions during the cutting-plane procedure.

Problem	$\|I\|$	m	# instances with best objective				Average CPU time (s)			
			GGX	GH	OA	MOA	GGX	GH	OA	MOA
HM14	50	25	**81**	**81**	24	20	0.35	0.01	0.15	0.09
HM14	50	50	**81**	**81**	2	0	1.94	0.03	2.83	0.12
HM14	50	100	**81**	**81**	27	27	13.08	0.10	104.39	0.19
HM14	100	25	**81**	80	18	15	0.43	0.01	0.66	0.16
HM14	100	50	**81**	75	3	0	2.44	0.04	0.39	0.16
HM14	100	100	**81**	76	0	1	16.04	0.11	83.31	0.27
HM14	200	25	**81**	**81**	2	0	0.57	0.02	0.80	0.18
HM14	200	50	**81**	80	1	0	3.41	0.19	1.85	0.23
HM14	200	100	**81**	**81**	0	0	20.76	0.15	111.70	0.46
HM14	400	25	**81**	72	24	20	0.85	0.16	0.75	0.18
HM14	400	50	**81**	77	5	0	5.37	0.21	1.83	0.35
HM14	400	100	**81**	72	0	0	32.02	0.08	167.55	1.15
HM14	800	25	**81**	77	0	0	0.93	0.04	0.89	0.31
HM14	800	50	**81**	66	0	0	6.38	0.10	2.14	0.61
HM14	800	100	**81**	69	22	18	41.99	0.30	299.35	5.64
ORlib	50	25	**81**	**81**	1	0	0.35	0.01	0.09	0.10
ORlib	50	25	**81**	**81**	1	0	0.35	0.01	0.10	0.10
ORlib	50	25	**81**	**81**	0	0	0.35	0.01	0.09	0.10
ORlib	50	25	**81**	78	0	0	0.35	0.01	0.09	0.09
ORlib	50	50	**81**	74	0	0	1.96	0.03	0.10	0.11
ORlib	50	50	**81**	**81**	0	0	1.96	0.03	0.11	0.11
ORlib	50	50	**81**	78	0	0	1.97	0.03	0.11	0.11
ORlib	50	50	**81**	80	0	0	1.96	0.03	0.10	0.11
ORlib	1000	100	**81**	60	0	0	46.17	0.37	1.02	66.50
ORlib	1000	100	**81**	54	0	0	46.23	0.37	10.91	66.24
ORlib	1000	100	**81**	50	0	0	46.18	0.37	8.38	56.44
NYC	82341	59	**81**	72	5	4	355.36	6.89	0.85	0.65
Average			**81**	74.78	5.00	3.89				

Table 4: Numerical results for nested-logit instances, grouped by the problem name (81 instances per row).

We look more closely into the NYC problem (the largest problem) to see how the algorithms work. In table 5, we report comparison results for the NYC instances in detail. Each row of the table corresponds to 9 instances with a value of C, varying from 2 to 10. GGX performs the best in terms of objective value, as it gives the best objective values for all the instances while GH only gives 6/9 best objective values for $C \in \{2, 3, 4\}$. The numbers of instances with the best objective values given by OA and MOA are very low. They both have 4 instances with the best objective values when $C = 2$ and the OA has one more instance with the best objective value when $C = 5$. This clearly shows that OA and MOA are outperformed by GGX and GH. On the other hand, in terms of CPU time, GGX is much more expensive than the other approaches. The average CPU times required by GGX is about 52 times, 418 times, and 547 times higher than those required by GH, OA, and MOA approaches, respectively. In summary, for these large instances, GH performs much better than OA and MOA, and GGX manages to significantly improve the objective values returned by GH.

C	# instances with best objective values				Average CPU time (s)			
	GGX	GH	OA	MOA	GGX	GH	OA	MOA
2	**9**	6	4	4	81.90	2.28	1.03	0.74
3	**9**	6	0	0	234.77	3.53	1.24	0.83
4	**9**	6	0	0	454.85	4.75	1.26	0.94
5	**9**	**9**	1	0	551.34	5.74	1.13	0.67
6	**9**	**9**	0	0	556.38	6.94	1.14	0.60
7	**9**	**9**	0	0	541.49	8.04	0.52	0.54
8	**9**	**9**	0	0	538.21	9.19	0.49	0.52
9	**9**	**9**	0	0	537.17	10.27	0.48	0.51
10	**9**	**9**	0	0	541.68	11.25	0.41	0.52
Average	**9**	8	0.56	0.44				

Table 5: Comparison results for NYC instances, grouped by C, 9 instances per row.

The MCP under the nested logit model has the advantage that the demand model is more flexible to capture the correlation between customers' utilities, but the objective function is highly non-concave, making it difficult to be solved exactly. On the other hand, the MCP under MNL is much easier to be solved to optimality due to the concavity of the objective function. Thus, similar to the experiments with MMNL instances, we also try to simplify nested instances and solve them by MNL solvers and explore the degradation in solution quality. To this end, for each nested instance, we simply set all the nested parameter μ to one, which will convert the nested instance into an MNL one. We then solve this MNL instance by the best exact method (i.e, MOA) and compute the performance gap as

$$\frac{f^{\text{GEV}}(S^{\text{GEV}}) - f^{\text{GEV}}(S^{\text{MNL}})}{f^{\text{GEV}}(S^{\text{GEV}})},$$

where S^{MNL} is the solution obtained by solving the MNL instance and S^{GEV} is the best solution found by solving the nested instance. To broader demonstrate the degradation, we compute the gaps with different settings as follows. We use 2 sets of nested logit parameter $\mu^1 = (1.1, 1.2, 1.3, 1.4, 1.5)$ and $\mu^2 = (1.8, 3.6, 1.0, 2.2, 4.1)$ (the latter is significantly higher than

the former) to see how these parameters affect the gaps. Moreover, the utility values in our benchmark instances are either too small or too large (e.g, from -30 to 0 in the HM14 dataset and from -150 to 450 in the NYC dataset), making the $\mathbf{Y}$ values (exponential functions of the utilities) extremely small or extremely large. This further makes GEV instances very similar to MNL instances in terms of the objective function. Thus, to better demonstrate the differences, we scale the utilities as $v'_{ij} = v_{ij}/20$. We then report performance gaps with the following three settings. The first setting, denoted as Gap 1, refers to the use of the nested parameters $\mu^1 = (1.1, 1.2, 1.3, 1.4, 1.5)$ and the original utilities. For the second setting (Gap 2) we use the set $\mu^1 = (1.1, 1.2, 1.3, 1.4, 1.5)$ and the scaled utilities, and the third setting (Gap 3) corresponds to the set $\mu^2 = (1.8, 3.6, 1.0, 2.2, 4.1)$ with the scaled utilities.

Table 6 reports the percentage gaps under these three settings, where each row corresponds to the average gap of nested instances with C varying from 2 to 10. The *Gap 1* column reports the gaps with the original nested parameters and original utility functions and we see that the gaps are relatively small, especially for HM14 dataset, which indicates that solving the MNL approximate instances would provide good solutions. For the second setting with scaled utilities, the gaps are more significant for all the instances, which verifies our statement about the issue of having too small or too large utilities. In the third setting where we increase some nested parameters and the nested instances are more different from the MNL counterparts, the gaps reported in the last column are remarkably higher and can go up to 9.32%. The average gap increases from 1.29% to 4.38%, as compared to the second setting, and there are only 2 problems whose gaps are less than 2%. The results also imply that if the nested parameters are close to 1, nested instances would be close to their MNL counterparts and one can solve these MNL instances to obtain good solutions for the nested problem.

5 Conclusion

In this paper, we have studied the maximum capture problem in facility location where customer behavior is captured by any GEV model. By leveraging the properties of the GEV generating function, we have shown that the objective function is monotonic and submodular, implying that a simple greedy heuristic can always give a solution whose value is at least $(1 - 1/e)$ times the optimal value. We have further developed an algorithm based on a greedy heuristic, a gradient-based local search, and an exchanging procedure to solve the problem under any GEV model and the MMNL model. We have tested and compared our algorithm with some state-of-the-art algorithms using MNL, MMNL, and nested logit instances and our numerical experiments clearly demonstrate the advantages of our approach, in terms of both returned objective value and CPU time. Our theoretical findings and algorithm can be applied to the maximum capture problem under any GEV model, including the popular MNL model and other complex GEV models in the literature.

Problem	$\|I\|$	[m]	Gap 1(%)	Gap 2(%)	Gap 3(%)
HM14	50	25	0.00	1.60	7.14
HM14	50	50	0.00	3.41	3.71
HM14	50	100	0.02	1.22	4.39
HM14	100	25	0.00	2.24	6.09
HM14	100	50	0.01	1.57	2.12
HM14	100	100	0.00	0.90	2.99
HM14	200	25	0.14	1.94	9.30
HM14	200	50	0.00	0.64	3.09
HM14	200	100	0.20	1.03	5.73
HM14	400	25	0.00	0.73	3.23
HM14	400	50	0.00	1.45	4.19
HM14	400	100	0.01	2.64	4.51
HM14	800	25	0.60	1.10	7.15
HM14	800	50	0.03	1.49	4.97
HM14	800	100	1.30	2.59	4.62
ORlib	50	25	1.92	0.88	2.99
ORlib	50	25	1.92	0.88	2.98
ORlib	50	25	1.93	0.88	2.99
ORlib	50	25	1.93	0.88	2.99
ORlib	50	50	2.17	1.13	4.16
ORlib	50	50	2.18	1.13	4.16
ORlib	50	50	2.18	1.13	4.16
ORlib	50	50	2.18	1.13	4.16
ORlib	1000	100	0.35	0.48	4.07
ORlib	1000	100	0.74	0.79	1.32
ORlib	1000	100	0.28	0.18	1.83
NYC	82341	59	0.02	0.73	9.32
Average			0.74	1.29	4.38

Table 6: Percentage gaps between objective values given by MNL approximate solutions and the best objective values obtained by solving nested instances.

In this work, we only focus on decisions involving location selections, noting that facility set-up costs would be also important to consider when building new facilities. In the context of random utilities, such decisions can be included in the utility function as $v_{ij} = a_{ij}x_j + b_{ij}$, where x_j is the set-up cost of facility j, a_{ij} is a parameter representing the sensitivity of customer i with respect to the cost x_j, and b_{ij} is another term that affects customers' utilities. In this setting, the joint facility and set-up cost planning problem would be formulated as an optimization problem with binary and continuous variables, which would be changeling to handle and would be interesting for future work. Other future directions would be to formulate and solve a maximum capture problem in the situation that the choice parameters are not known with certainty, or to consider a combination of facility location and security planning under the MNL/MMNL or any GEV models.

Acknowledgments

We thank the Editor and two anonymous referees whose comments substantially helped us improve the previous version of the paper.

References

Aboolian, R., Berman, O., and Krass, D. Competitive facility location and design problem. *European Journal of Operational Research*, 182:40–62, 02 2007.

Ageev, A., Ye, Y., and Zhang, J. Improved combinatorial approximation algorithms for the k-level facility location problem. *SIAM Journal on Discrete Mathematics*, 18(1):207–217, 2004.

Ben-Akiva, M. *The structure of travel demand models.* PhD thesis, MIT, 1973.

Ben-Akiva, M. and Bierlaire, M. Discrete choice methods and their applications to short term travel decisions. Chapter for the Transportation Science Handbook, Preliminary Draft, MIT, March 1999.

Benati, S. Submodularity in competitive location problems. *Ricerca Operativa*, 1996.

Benati, S. and Hansen, P. The maximum capture problem with random utilities: Problem formulation and algorithms. *European Journal of Operational Research*, 143:518–530, 12 2002. doi: 10.1016/S0377-2217(01)00340-X.

Berman, O., Drezner, T., Drezner, Z., and Krass, D. Modeling competitive facility location problems: New approaches and results. In *Decision Technologies and Applications*, pages 156–181. INFORMS, 09 2009. ISBN 978-1-877640-24-7.

Bonami, P., Biegler, L. T., Conn, A. R., Cornuéjols, G., Grossmann, I. E., Laird, C. D., Lee, J., Lodi, A., Margot, F., Sawaya, N., et al. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.

Bonami, P., Lee, J., Leyffer, S., and Wächter, A. More branch-and-bound experiments in convex nonlinear integer programming. *Preprint ANL/MCS-P1949-0911, Argonne National Laboratory, Mathematics and Computer Science Division*, 2011.

Conn, A. R., Gould, N. I., and Toint, P. L. *Trust region methods.* SIAM, 2000.

Daly, A. and Bierlaire, M. A general and operational representation of generalised extreme value models. *Transportation Research Part B*, 40(4):285 – 305, 2006.

Duran, M. A. and Grossmann, I. E. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, 36(3):307–339, 1986.

Fosgerau, M. and Bierlaire, M. Discrete choice models with multiplicative error terms. *Transportation Research Part B*, 43(5):494–505, 2009.

Fosgerau, M., McFadden, D., and Bierlaire, M. Choice probability generating functions. *Journal of Choice Modelling*, 8:1–18, 2013. ISSN 1755-5345.

Freire, A., Moreno, E., and Yushimito, W. A branch-and-bound algorithm for the maximum capture problem with random utilities. *European Journal of Operational Research*, 252, 12 2015.

Haase, K. and Müller, S. A comparison of linear reformulations for multinomial logit choice probabilities in facility location models. *European Journal of Operational Research*, 232, 08 2013.

Hasse, K. *Discrete location planning.* Institute of Transport and Logistics Studies, 2009.

Holguin-Veras, J., Reilly, J., Aros-Vera, F., et al. New york city park and ride study. Technical report, University Transportation Research Center, 2012.

Horowitz, J. Discrete choice analysis: Theory and application to travel demand, by M. Ben-Akiva and S.R. Lerman. *Transportation Science*, 20(4):290–291, 1986. ISSN 0041-1655.

Koppelman, F. and Wen, C.-H. The paired combinatorial logit model: properties, estimation and application. *Transportation Research Part B*, 34:75–89, 2000.

Kung, L.-C. and Liao, W. An approximation algorithm for a competitive facility location problem with network effects. *European Journal of Operational Research*, 267:176–186, 2018.

Laporte, G., Nickel, S., and da Gama, F. S. Introduction to location science. In *Location science*, pages 1–18. Springer, 2015.

Lin, Y. H. and Tian, Q. Branch-and-cut approach based on generalized benders decomposition for facility location with limited choice rule. *European Journal of Operational Research*, 293 (1):109–119, 2021a.

Lin, Y. H. and Tian, Q. Generalized benders decomposition for competitive facility location with concave demand and zone-specialized variable attractiveness. *Computers & Operations Research*, 130:105236, 2021b.

Ljubić, I. and Moreno, E. Outer approximation and submodular cuts for maximum capture facility location problems with random utilities. *European Journal of Operational Research*, 266(1):46–56, 2018.

Mai, T. and Lodi, A. An algorithm for assortment optimization under parametric discrete choice models. *Available at SSRN 3370776*, 2019.

Mai, T. and Lodi, A. A multicut outer-approximation approach for competitive facility location under random utilities. *European Journal of Operational Research*, 284, 01 2020.

Mai, T., Frejinger, E., Fosgerau, M., and Bastin, F. A dynamic programming approach for quickly estimating large network-based mev models. *Transportation Research Part B*, 98: 179–197, 2017.

McFadden, D. Conditional logit analysis of qualitative choice behaviour. In Zarembka, P., editor, *Frontiers in Econometrics*, pages 105–142. Academic Press New York, New York, NY, USA, 1973.

McFadden, D. Modelling the choice of residential location. In Karlqvist, A., Lundqvist, L., Snickars, F., and Weibull, J., editors, *Spatial Interaction Theory and Residential Location*, pages 75–96. North-Holland, Amsterdam, 1978.

McFadden, D. Econometric models of probabilistic choice. In Manski, C. and McFadden, D., editors, *Structural Analysis of Discrete Data with Econometric Applications*, chapter 5, pages 198–272. MIT Press, 1981.

McFadden, D. and Train, K. Mixed MNL models for discrete response. *Journal of applied Econometrics*, pages 447–470, 2000.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Ortiz Astorquiza, C., Contreras, I., and Laporte, G. Formulations and approximation algorithms for multilevel uncapacitated facility location. *INFORMS Journal on Computing*, 29:767 – 779, 09 2017.

Shmoys, D. B., Tardos, É., and Aardal, K. Approximation algorithms for facility location problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274, 1997.

Small, K. A. A discrete choice model for ordered alternatives. *Econometrica*, 55(2):409–424, 1987.

Train, K. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2003.

Vovsha, P. and Bekhor, S. Link-nested logit model of route choice Overcoming route overlapping problem. *Transportation Research Record*, 1645:133–142, 1998.

Wen, C.-H. and Koppelman, F. S. The generalized nested logit model. *Transportation Research Part B: Methodological*, 35(7):627–641, 2001.

Whelan, G., Batley, R., Fowkes, T., and Daly, A. Flexible models for analyzing route and departure time choice. *Publication of: Association for European Transport*, 2002.

Zhang, Y., Berman, O., and Verter, V. The impact of client choice on preventive healthcare facility network design. *OR Spectrum*, 34, 04 2012.