4-2017

A dynamic programming approach for quickly estimating large network-based MEV models

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

Emma FREJINGER

Mogens FOSGEREAU

Fabian BASTIN

Transportation Research Part B 98 (2017) 179–197

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

A dynamic programming approach for quickly estimating large network-based MEV models

Tien Mai[a,*], Emma Frejinger[b], Mogens Fosgerau[c], Fabian Bastin[b]

[a] Department of Mathematical and Industrial Engineering, Polytechnique Montréal, Montréal, Québec, Canada
[b] Department of Computer Science and Operational Research, Université de Montréal and CIRRELT, Montréal, Québec, Canada
[c] Technical University of Denmark, Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 4 December 2015
Revised 21 October 2016
Accepted 21 December 2016
Available online 6 January 2017

Keywords:
Multivariate extreme value models
Dynamic programming
Discrete choice
Maximum likelihood estimation
Nested fixed point algorithm
Value iteration

ABSTRACT

We propose a way to estimate a family of static Multivariate Extreme Value (MEV) models with large choice sets in short computational time. The resulting model is also straightforward and fast to use for prediction. Following Daly and Bierlaire (2006), the correlation structure is defined by a rooted, directed graph where each node without successor is an alternative. We formulate a family of MEV models as dynamic discrete choice models on graphs of correlation structures and show that the dynamic models are consistent with MEV theory and generalize the network MEV model (Daly and Bierlaire, 2006). Moreover, we show that these models can be estimated quickly using the concept of network flows and the nested fixed point algorithm (Rust, 1987). The main reason for the short computational time is that the new formulation allows to benefit from existing efficient solution algorithms for sparse linear systems of equations.

We present numerical results based on simulated data with varying number of alternatives and nesting structures. We estimate large models, for example, a cross-nested model with 200 nests and 500,000 alternatives and 210 parameters that needs between 100–200 iterations to converge (4.3 h on an Intel(R) 3.2 GHz machine using a non-parallelized code). We also show that our approach allows to estimate a cross-nested logit model of 111 nests with a real data set of more than 100,000 observations in 14 h.

1. Introduction

Discrete choice models with large choice sets are of interest in various applications such as route and location choice analysis. A large number of alternatives can make the estimation and application of models impractical. In this context, sampling alternatives becomes appealing. McFadden (1978) shows how to obtain consistent estimates with sampled choice sets for the multinomial logit (MNL) model. More recently, Guevara and Ben-Akiva (2013) extend the work of McFadden (1978) for the Multivariate Extreme Value (MEV) model. Their approach has been used with a cross-nested logit route choice model (Lai and Bierlaire, 2015) and a mode destination choice model with simulated data (Daly et al., 2014). Sampling alternatives may however be difficult and time consuming in practice, both for estimating model parameters and for making predictions. In this paper we propose a novel approach based on dynamic discrete choice models that allows to estimate large-scale MEV models without sampling of alternatives. The approach is applicable when the correlation structure

* Corresponding author.
E-mail address: AnhTien.Mai@cirrelt.ca (T. Mai).

http://dx.doi.org/10.1016/j.trb.2016.12.017
0191-2615/

can be represented by a network (we hence call it network-based) and it is convenient to use for prediction since choice probabilities can be computed quickly.

Dynamic discrete choice models are in general more complex to estimate and to apply than static discrete choice models since they require dynamic programming problems to be solved in order to evaluate the log-likelihood function. Recently, Fosgerau et al. (2013a), Mai et al. (2015) and Mai (2016) showed that modeling path choices in transport networks as a sequence of link-choices using a dynamic discrete choice formulation is actually simpler to deal with than static, path-based, discrete choice models. This paper builds on a similar idea but in a different context. We propose a dynamic discrete choice approach that allows large network-based MEV models (McFadden, 1978; Daly and Bierlaire, 2006) to be estimated in short computational time.

Daly and Bierlaire (2006) show how to define choice probability generating functions (CPGF) for a certain kind of MEV models (Fosgerau et al., 2013b). These models are defined by a rooted, cycle-free graph with parameters associated with nodes and arcs. This graph is referred to as an MEV-network (also known as Generalized Extreme Value, GEV, network). This way of defining an MEV model is useful because we can simply verify that the graph is an MEV-network, then this is an easy way to prove that the CPGF satisfies the MEV properties. This paper complements and extends the work by Daly and Bierlaire (2006).

Based on ideas from route choice modeling we consider the graph as a transport network where the root is an origin and nodes representing alternatives are viewed as destinations. A unit flow is inserted at the origin and is transported via the network, such that the expected flows arriving at the destinations correspond to probabilities. The transition probabilities of a flow between two nodes are defined based on a dynamic discrete choice model, in a way similar to the route choice model proposed by Mai et al. (2015). We prove that the dynamic model is equivalent to a network MEV model (Daly and Bierlaire, 2006). Moreover, we prove that even if the graph contains cycles, the dynamic model is consistent with MEV theory.

The main challenge lies in the definition and the computation of the expected maximum utility from a node in the graph to the nodes representing the alternatives (called value functions) and in the computation of choice probabilities. The value functions are computed using a value iteration method, while the choice probabilities are computed as the expected flows from the root to destinations, which can be found as the solution to a system of linear equations. Moreover, efficient non-linear programming techniques require analytical derivatives of the log-likelihood function with respect to the unknown parameters. We show that also the derivatives of choice probabilities are solutions to systems of linear equations. We use the nested fixed point algorithm proposed by Rust (1987) to estimate the model by maximum likelihood. We present a discussion on computational complexity and provide some comparisons of computational times that clearly show the superiority of the proposed approach, compared to the standard approach which is based on recursive equations.

Even though the proposed dynamic model is formulated in a way similar to the recursive route choice model, the graphs in the present case are different from those of transport networks, as they have multiple destinations for each observation. Moreover, the link utilities are defined based on the structural parameters of the MEV model, and only the value function at the destinations represent the genetic utilities of the alternatives.

In summary, this paper makes two main contributions. First, we formulate a family of MEV static discrete choice models as dynamic discrete choice models on graphs of correlation structures and show that these models, under certain conditions, are consistent with MEV theory and generalize the Daly-Bierlaire network MEV model. Second, we show how these models can be estimated by maximum likelihood in short computational time. The estimation code is implemented in MATLAB and we share the code freely via GitHub as an open source project.[1]

The paper is structured as follows. Section 2 briefly presents the network MEV model and Section 3 presents the dynamic discrete choice approach for formulating static discrete choice models. We provide an illustrative example of a cross-nested logit model in Section 4. Furthermore, we show how to estimate the model by maximum likelihood in Section 5. Numerical results based on simulated and real data sets are presented in Section 6, and finally Section 7 concludes.

2. The network MEV model

Daly and Bierlaire (2006) propose the network MEV model as a general representation of a class of MEV models based on a network structure. They consider a rooted, connected and cycle-free graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ and $\mathcal{A}$ are the set of nodes and arcs, respectively. There is at most one arc for any given node pair. Each node without successors defines an alternative j in choice set C. The graph defines the correlation structure (and a simple example is the well-known tree of a nested logit model where each leaf is an alternative). A parameter $\mu_k > 0$ is associated with each node k and a parameter $\alpha_{ka} > 0$ with each arc (k, a). The CPGFs are defined as

$$G_k(y) = \begin{cases} y_k^{\mu_k} & \text{if} \quad k \in C \\ \sum_{a \in \mathcal{N}(k)} \alpha_{ka} G_a(y)^{\mu_k/\mu_a} & \text{if } k \in \mathcal{N} \backslash C, \end{cases} \tag{1}$$

where $y_k = e^{U_k}$ (U_k is the deterministic utility associated with the alternative represented by node k) and $\mathcal{N}(k)$ the set of successor nodes of node k. Daly and Bierlaire (2006) show that if $\mathcal{G}$ is cycle-free and $\mu_a \geq \mu_k$, $\forall a \in \mathcal{N}(k)$, then G_k, for a given node k, is a μ_k-MEV CPGF. That is, the choice probabilities generated by $G_k(y)$ are consistent with random

[1] https://github.com/maitien86/

utility maximization theory (McFadden, 1973). Daly and Bierlaire (2006) provide a recursive formula to compute the choice probability, namely,

$$P_k(j) = \begin{cases} \sum_{a\in\mathcal{N}(k)} P_a(j)\Omega_{ka} & \text{if } k \in \mathcal{N}\backslash C \\ 1 & \text{if } k = j \\ 0 & \text{if } k \in C, k \neq j \end{cases} \quad \forall j \in C, \tag{2}$$

where Ω_{ka} is

$$\Omega_{ka} = \frac{\alpha_{ka} G_a(y)^{\mu_k/\mu_a}}{G_k(y)}.$$

The probability of choosing alternative j given by $G_r(y)$ is $P_r(j)$.

3. Formulating the network MEV model using the dynamic discrete choice framework

In this section we consider the same definition of the graph $\mathcal{G}$ as in the previous section, that is, an MEV-network. We focus on formulating choice probabilities using the dynamic discrete choice framework and to explain the derivation we use a transport network analogy. More precisely, we view the root as an origin and the nodes representing alternatives as destinations. There is at least one path connecting the root to any destination. The objective is to define the choice probability of each destination/alternative $j \in C$, where C is assumed to be a finite set. These are obtained by describing the choice problem as a sequential node choice made by a traveler going from the origin (root) to a destination/alternative.

A key in our formulation is the fact that the probability of an alternative $j \in C$ can be defined in terms of incoming flow. We consider a unit outgoing flow from the origin to multiple destinations and assign the choice probability of each alternative as the expected incoming flow at the respective destination. We denote the incoming flow at node a as $F(a)$ and the probability of going from a to k as $P(a|k)$, $a, k \in \mathcal{N}$. The expected incoming flows is

$$F(a) = D(a) + \sum_{k\in\mathcal{N}} P(a|k)F(k), \quad \forall a \in \mathcal{N}, \tag{3}$$

where D is a vector with zero elements except for the origin which equals one. The choice probabilities $P(j), j \in C$ are

$$P(j) = F(j), \quad \forall j \in C. \tag{4}$$

It is easy to show that $\sum_{j\in C} P(j) = \sum_{j\in C} F(j) = 1$. Moreover, from (3), the probability of choosing $j \in C$ can be written equivalently as

$$P(j) = \sum_{[k_0,\ldots,k_J]\in\Phi(j)} \prod_{i=0}^{J-1} P(k_{i+1}|k_i), \quad j \in C, \tag{5}$$

where $\Phi(j)$ is the set of all paths connecting r and j. A path is defined by a sequence of nodes $k_0, k_1, \ldots, k_J$ such that $k_{i+1} \in \mathcal{N}(k_i)$, $\forall i = 0, \ldots, J-1$, where k_0 is the root r and k_J represents alternative j.

Both definitions of $P(j)$ are based on next node probabilities $P(a|k)$, $\forall k, a \in \mathcal{N}$. In the following we present how next node probabilities can be computed using the dynamic discrete choice framework and ideas in Mai et al. (2015). We associate a utility with each node $a \in \mathcal{N}(k)$ conditional on its predecessor k

$$u(a|k;\beta) = v(a|k;\beta) + \frac{1}{\mu_k}(\epsilon(a) - \gamma),$$

where $v(a|k;\beta)$ is a deterministic utility, β is a vector of parameters to be estimated, μ_k is a strictly positive scale parameter associated with node k, $\epsilon(a)$ is extreme value type I distributed and i.i.d. over $a \in \mathcal{N}(k)$. Euler's constant γ is subtracted to ensure that the random terms have zero mean.

The node probabilities depend on the nodes that are available downstream. This is captured by the expected maximum utility (or value functions) $V(k)$ from a node k to the destinations $j \in C$. We associate a deterministic utility U_j with each alternative $j \in C$ and define $V(j) = U_j$. We note that the node utilities play a different role than the value functions at the destinations. As we illustrate in Section 4, they include parameters related to the correlation structure while the value functions at the destinations represent the genetic deterministic utilities of the alternatives.

The model can be considered as a dynamic programming problem with absorbing states $j \in C$, thus the value function $V(k)$ for $k \in \mathcal{N}\backslash C$ is recursively defined by Bellman's equation

$$V(k;\beta) = \mathbb{E}\left[\max_{a\in\mathcal{N}(k)} \left\{v(a|k;\beta) + V(a;\beta) + \frac{1}{\mu_k}(\epsilon(a) - \gamma)\right\}\right], \quad \forall k \in \mathcal{N}\backslash C,$$

or equivalently

$$\mu_k V(k;\beta) = \mathbb{E}\left[\max_{a\in\mathcal{N}(k)} \left\{\mu_k(v(a|k;\beta) + V(a;\beta)) + \epsilon(a)\right\}\right] - \gamma, \quad \forall k \in \mathcal{N}\backslash C, \tag{6}$$

which in this case is the logsum

$$\mu_k V(k;\beta) = \ln\Big(\sum_{a\in\mathcal{N}(k)} e^{\mu_k(v(a|k;\beta)+V(a;\beta))}\Big), \quad \forall k \in \mathcal{N}\backslash C. \tag{7}$$

In order to simplify the notation we omit from now on β from the value functions V and from the node-based utilities v. It follows from (6) that the probability of node a given node k is given by the MNL model,

$$P(k|a) = \delta(a|k)e^{\mu_k(V(a)+v(a|k)-V(k))}, \quad \forall k, a \in \mathcal{N}, \tag{8}$$

where $\delta(a|k)$ equals one if $a \in \mathcal{N}(k)$ and zero otherwise.

We define a vector Y of size $|\mathcal{N}|$ with entries

$$Y_k = e^{\mu_k V(k)}, \quad \forall k \in \mathcal{N}. \tag{9}$$

This allows us to write Bellman's equation (7) as a system of non-linear equations

$$Y_k = \begin{cases} \sum_{a\in\mathcal{N}(k)} e^{\mu_k v(a|k)} Y_a^{\mu_k/\mu_a} & \text{if } k \in \mathcal{N}\backslash C, \\ e^{U_k} & \text{if } k \in C. \end{cases} \tag{10}$$

Using (8), the choice probability of node a given k can be written as

$$P(a|k) = \delta(a|k)e^{\mu_k v(a|k)}\frac{Y_a^{\mu_k/\mu_a}}{Y_k}, \quad \forall k, a \in \mathcal{N}. \tag{11}$$

The computation of the choice probabilities using (5) and (8) requires the vector of value functions Y. The latter can be solved by value iteration as we present in Section 5.1.

Proposition 1 states that if the graph is an MEV-network (Daly and Bierlaire, 2006), then the proposed formulation is equivalent to the network MEV model. Hence, the choice probabilities of special cases of network MEV models, e.g. MNL, nested logit and cross-nested logit, can be computed using the proposed formulation. In this context it is interesting to note that any additive random utility model can be approximated by a cross-nested logit model (Fosgerau et al., 2013b).

Proposition 1. *If the graph $\mathcal{G}$ is a non-empty, cycle-free and $\mu_k \le \mu_a$, $v(a|k) = \frac{1}{\mu_k}\ln\alpha_{ka}$, $\forall k, a \in \mathcal{N}, a \in \mathcal{N}(k)$, then the resulting model is equivalent to the network MEV model i.e. $Y_k = G_k$, $\forall k \in \mathcal{N}$, and the choice probabilities given by (2) and (3) are identical.*

Proof. We first prove that $G_k(y) = Y_k$, $\forall k \in \mathcal{N}$. Indeed, under the hypotheses of Proposition 1 and according to (1) we have

$$Y_k = G_k, \quad \forall k \in C. \tag{12}$$

For $k \in \mathcal{N}\backslash C$, from (10) and using $\alpha_{ka} = e^{\mu_k v(a|k)}$,

$$Y_k = \sum_{a\in\mathcal{N}(k)} \alpha_{ka} Y_a^{\mu_k/\mu_a}. \tag{13}$$

The result then follows from (1), (12) and (13). Consequently, according to Daly and Bierlaire (2006) Y_k associated with a node $k \in \mathcal{N}$ is a $\mu_k - MEV$ CPGF. Furthermore, from (11) we obtain

$$\Omega_{ka} = P(a|k) \quad \forall k \in \mathcal{N}\backslash C, a \in \mathcal{N}(k).$$

This leads to the fact the probabilities given by (2) are identical to those given by (5). □

Theorem 1 extends the results of Daly and Bierlaire (2006) by showing that even if the graph contains cycles, the resulting model is consistent with MEV theory.

Theorem 1. *If the graph $\mathcal{G}$ is a non-empty and $\mu_k \le \mu_a$, $\forall k, a \in \mathcal{N}, a \in \mathcal{N}(k)$ and the Bellman's equation has a solution, then the model is an additive random utility MEV model with the CPGF $G(e^{U_i}, i \in C) = Y_r$, and Y_r is a $\mu_r - MEV$ CPGF.*

We provide the proof of this theorem in Appendix A.

4. Illustrative example

In this section we take the example of a cross-nested logit model to illustrate that the proposed formulation is equivalent to the CPGF given by Ben-Akiva and Bierlaire (1999)

$$G(y_1,\ldots,y_J) = \sum_{m\in\mathcal{M}}\left(\sum_{j\in C}\alpha_{jm}y_j^{\mu_m}\right)^{\mu/\mu_m}, \tag{14}$$

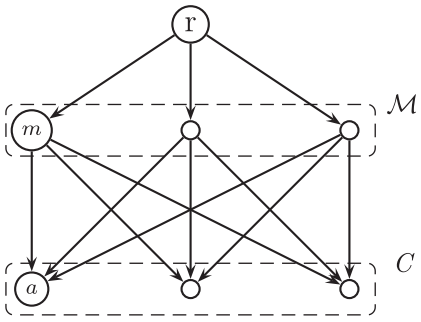

Fig. 1. A cross-nested structure.

where $\mathcal{M}$ is the set of nests, μ_m, μ and α_{jm} are the parameters of the model and $0 < \mu \leq \mu_m$, $\alpha_{jm} \geq 0$, $\forall j \in C, m \in \mathcal{M}$. The corresponding choice probability is

$$P(j) = \sum_{m\in\mathcal{M}} \frac{\left(\sum_{j'\in C} \alpha_{j'm} e^{\mu_m U_{j'}}\right)^{\mu/\mu_m}}{\sum_{m'\in\mathcal{M}} \left(\sum_{j'\in C} \alpha_{j'm'} e^{\mu_{m'} U_{j'}}\right)^{\mu/\mu_{m'}}} \frac{\alpha_{jm} e^{\mu_m U_j}}{\sum_{j'\in C} \alpha_{j'm} e^{\mu_m U_{j'}}}, \quad \forall j \in C. \tag{15}$$

In this case the graph consists of three layers: a root r, a set of nodes $\mathcal{M}$ representing the nests and a set of destination nodes C representing alternatives, as illustrated in Fig. 1.

There is an arc between the root and each nest and arcs from all nests to all alternatives. A parameter μ_m is associated with nodes $m \in \mathcal{M}$ and μ with the root ($\mu_r = \mu$). We define $v(a|m) = \frac{1}{\mu_m} \ln(\alpha_{am})$ $\forall m \in \mathcal{M}$, $a \in \mathcal{N}(m)$ where α_{am} are nesting parameters but define $v(m|r) = 0$, $\forall m \in \mathcal{M}$. According to (7) and (8) the probability of $a \in C$ for a given $m \in \mathcal{M}$ is

$$\begin{aligned} P(a|m) &= \frac{e^{\mu_m (U_a + v(a|m))}}{\sum_{a'\in\mathcal{N}(m)} e^{\mu_m (U_{a'} + v(a'|m))}} \\ &= \frac{\alpha_{am} e^{\mu_m U_a}}{\sum_{a'\in C} \alpha_{a'm} e^{\mu_m U_{a'}}}, \end{aligned} \tag{16}$$

and the nest probability is

$$P(m|r) = \frac{e^{\mu_r V(m)}}{\sum_{m'\in\mathcal{M}} e^{\mu_r V(m')}}. \tag{17}$$

Moreover, for each node $m \in \mathcal{M}$ the value function given by (7) is

$$V(m) = \frac{1}{\mu_m} \ln\left(\sum_{a\in\mathcal{N}(m)} e^{\mu_m (v(a|m) + V(a))}\right). \tag{18}$$

Using (5), (16), (17) and (18), the choice probability of choosing $a \in C$ is

$$\begin{aligned} P(a) &= \sum_{m\in\mathcal{M}} P(m|r) P(a|m) \\ &= \sum_{m\in\mathcal{M}} \frac{\left(\sum_{a'\in C} \alpha_{a'm} e^{\mu_m U_{a'}}\right)^{\mu_r/\mu_m}}{\sum_{m'\in\mathcal{M}} \left(\sum_{a'\in C} \alpha_{a'm'} e^{\mu_{m'} U_{a'}}\right)^{\mu_r/\mu_{m'}}} \frac{\alpha_{am} e^{\mu_m U_a}}{\sum_{a'\in C} \alpha_{a'm} e^{\mu_m U_{a'}}}, \end{aligned}$$

which is equivalent to the choice probability given by (15).

5. Maximum likelihood estimation

In this section we show how the model can be estimated using the nested fixed point algorithm proposed by Rust (1987). The algorithm combines an outer iterative non-linear optimization algorithm searching over the parameter space with an inner algorithm solving the value functions. The likelihood function and consequently the value functions are evaluated for each iteration of the non-linear optimization algorithm. The speed of the algorithm depends critically on how fast the value functions can be solved. We present a solution method that allows to quickly compute the value functions and the choice probabilities.

5.1. Solving the value functions

The value functions are the solution to a system of non-linear equations (10). We can solve this system using a simple value iteration method. To do so we introduce matrix notation and define a matrix M of size $|\mathcal{N}| \times |\mathcal{N}|$ and a vector b of

size $|\mathcal{N}|$, with entries

$$M_{ka} = \delta(a|k)e^{\mu_k v(a|k)}; \quad b_k = \begin{cases} e^{U_k} & \text{if } k \in C \\ 0 & \text{if } k \in \mathcal{N}\backslash C, \end{cases} \tag{19}$$

and a matrix $X(Y)$ of size $|\mathcal{N}| \times |\mathcal{N}|$, with entries $X(Y)_{ka} = Y_a^{\mu_k/\mu_a}$, $\forall k, a \in \mathcal{N}$. We can write (10) in as

$$Y_k = \sum_{a \in \mathcal{N}} M_{ka} Y_a^{\mu_k/\mu_a} + b_k, \quad \forall k \in \mathcal{N}, \tag{20}$$

or equivalently

$$Y = [M \circ X(Y)]e + b, \tag{21}$$

where $\circ$ is the element-by-element multiplication and e is a vector of size $|\mathcal{N}|$ with value one for all nodes. This equation can be solved by a value iteration. We start with an initial vector $Y^{(0)}$ and compute a new vector for each iteration i

$$Y^{(i+1)} \leftarrow [M \circ X(Y^{(i)})]e + b.$$

We iterate until a fixed point is found using $||Y^{(i+1)} - Y^{(i)}||^2 < \varsigma$ for a given threshold $\varsigma > 0$ as stopping criteria. It can be shown that if the Bellman equation has a solution, the value iteration method converges after a finite number of iterations (see for instance Rust, 1987; 1988). If the Bellman equation has no solution, then the sequence $\{Y^{(0)}, Y^{(1)} \ldots\}$ is unbounded. Mai et al. (2015) use value iteration with dynamic accuracy to compute the vector of the value functions in a real road network which contains cycles. Theorem 2 states that, in the case of network MEV models (cycle-free graphs) the value iteration method converges to a unique fixed point after a finite number of iterations. We observed that, in practice, only a few iterations are needed, for instance, it requires only two iterations for a MNL model and three iterations for a cross-nested logit model, independently of the number of nodes and the structure of the graph.

Theorem 2. *If the graph $\mathcal{G}$ is cycle-free, then there exists an integer number $K > 0$ such that $Y^{(i)} = Y^{(K)}$, $\forall i \geq K$.*

The proof of this theorem is given in Appendix B.

5.2. Choice probabilities

In Section 3 we present two equivalent definitions of choice probabilities: one based on all paths connecting the root with the destination/alternative and another based on incoming flow. Enumerating all paths can be expensive and in this section we show that the latter approach is fast since it amounts to solving a system of linear equations.

We can conveniently write (3) in matrix notation as a system of linear equations

$$(I - \mathcal{P}^T)F = D, \tag{22}$$

where I is the identity matrix and $\mathcal{P}$ is a matrix of size $|\mathcal{N}| \times |\mathcal{N}|$ with entries $\mathcal{P}_{ka} = P(a|k)$, $\forall k, a \in \mathcal{N}$. The probabilities of the alternatives in C are the corresponding entry flows

$$P(j) = F(j), \quad \forall j \in C. \tag{23}$$

$(I - \mathcal{P}^T)$ is invertible (Proposition 2), meaning that (22) has an unique solution. We can therefore compute the choice probabilities of the alternatives in C by just solving the system of linear equations in (22).

Proposition 2. *$(I - \mathcal{P}^T)$ is invertible.*

The proof is given in Appendix C.

5.3. Log-likelihood function and non-linear optimization

We assume that the utilities associated with alternatives, the deterministic utility associated with a pair of nodes $v(a|k)$, $a \in \mathcal{N}(k)$, and the scales of the model μ_k, $k \in \mathcal{N}$ are functions of parameters β to be estimated. They can, for example, be a sub-vector of β, constants or exponential functions of β as in Mai et al. (2015). The log-likelihood function is defined over observations $n = 1, \ldots, N$,

$$LL(\beta) = \sum_{n=1}^{N} \ln P(i_n|C_n), \tag{24}$$

where i_n is the chosen alternative, C_n the choice set of n and $P(i_n|C_n)$ is computed using (22).

In large-scale MEV models there can be many parameters associated with nodes μ_k, $k \in \mathcal{N}$, and node pair utilities $v(a|k)$. Not all of them are identifiable from data and normalizations are required. More precisely, as stated by Proposition 1 and Theorem 1, $\mu_a \geq \mu_k > 0$, $\forall a \in \mathcal{N}(k)$, need to be satisfied to ensure MEV consistency. Since only the ratios of μ_k, $\forall k \in \mathcal{N}$,

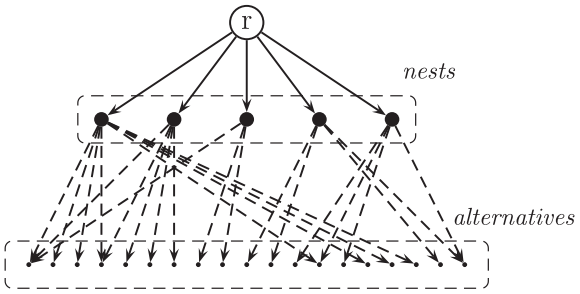

Fig. 2. A cross-nested structure.

matter in the model, we are free to fix $\mu_r = 1$. The maximum log-likelihood estimation problem is therefore formulated as a constrained non-linear optimization problem

$$\begin{aligned} \max_{\beta} \quad & LL(\beta) \\ \text{s.t.} \quad & \mu_a \geq \mu_k > 0, \quad \forall a \in \mathcal{N}(k) \\ & \mu_r = 1. \end{aligned}$$

We note that specific models may require additional constraints that are not written here. For instance, in the network MEV model, the expected values of the random terms depend on α_{ka}, $\forall k, a \in \mathcal{N},\ a \in \mathcal{N}(k)$, and may consequently vary over alternatives. It is therefore critical to define normalization constraints on the α parameters in order to have consistent expected utilities, Daly and Bierlaire (2006) propose such constraints. For the cross-nested logit model given by (14) and (15), the α parameters could be constrained to sum to one, i.e., $\sum_{m \in M} \alpha_{jm}^{\mu_m/\mu} = 1$ (Abbé et al., 2007; Papola, 2004).

Efficient non-linear optimization methods require the gradient of the log-likelihood function. We provide analytical derivatives in Appendix D and show that they are solutions to systems of linear equations, which makes their computation convenient. Moreover, we derive elasticities in Appendix E and show that they also can be obtained by solving systems of linear equations.

6. Application to large problems

In this section we report the performance of the proposed approach based on simulated and real data. InSections 6.1–6.3 we present results on different simulated data sets to illustrate the performance of our method. The objective is to evaluate the computational time required to estimate different MEV models with large choice sets. Simulated data serves this purpose well since we can choose the number of alternatives, observations and nesting structure. We also present results based on real data in Section 6.4 in order to illustrate how the proposed approach can be used to model a real choice problem and we analyze the computational time.

6.1. Experimental setup

For the purpose of this experiment, we use the same attribute values for all observations and we simulate a large number of observations so that the model parameters can be identified. This setup allows us to solve the value functions once each time we evaluate the log-likelihood function. This serves well the purpose of the numerical results since the computational time mainly stems from the evaluation of the value functions. When the value functions are specific to each observation or to different groups of observations, the code associated with the evaluation of the log-likelihood function can be parallelized to decrease the computational time. The most efficient way to do this depends on the application. We provide a more in-depth discussion on computation time in Section 6.3. In the following we present the experimental setup in more detail.

We use three different models (named CNL-1, CNL-2 and N-MEV) to generate observations. The first two models are cross-nested logit models (discussed in Section 4) and the corresponding MEV-network is illustrated in Fig. 2. The two models have different numbers of nests, CNL-1 has 5 and CNL-2 has 200. The third model (N-MEV) is a three-levels cross-nested logit model and the graph of this model is illustrated in Fig. 3. The first level has 5 nests and the second level has 50 nests, level one nests connect to all level two nests and level two nests connect to all alternatives. Note that the sizes of these graphs depend on the numbers of alternatives in the choice sets. In this experiment we create three choice sets of sizes 10,000, 100,000 and 500,000, denoted by D1, D2 and D3, respectively.

We associate 6 different attributes with each alternative. The attributes are generated uniformly in the interval [0, 5] using independent draws. The deterministic utility of each alternative is specified as a linear-in-parameters formulation of the six attributes. In order to generate observations, we manually choose parameter values in the interval $[-1, -2]$. Similarly, we choose the values of μ_k and $\alpha_{ka} = e^{\mu_k v(a|k)}$, $\forall k \in \mathcal{N} \backslash C,\ a \in \mathcal{N}(k)$, in the intervals the intervals [0.5, 1] and [0, 1], respectively. For each model we simulate samples of size 10^5, 10^6 and 2×10^6 from choice sets D1, D2 and D3. We show a summary of the setup in Table 1.

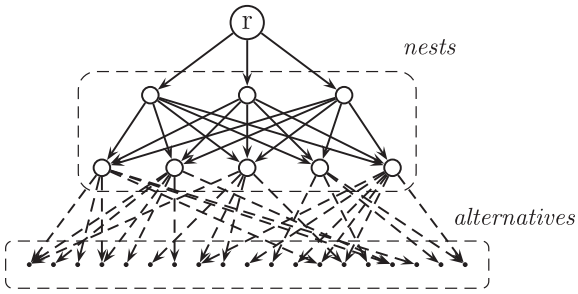

Fig. 3. A three-levels cross-nested structure.

Table 1
Models and simulated data sets.

Models	Nb. alternatives	# arcs in graphs	# Obs.
CNL-1 (5 nests)	10,000 (D1)	17,262	100,000
	100,000 (D2)	172,376	1,000,000
	500,000 (D3)	862,597	2,000,000
CNL-2 (200 nests)	10,000 (D1)	30,011	100,000
	100,000 (D2)	298,237	1,000,000
	500,000 (D3)	1,490,207	2,000,000
N-MEV (5–50 nests)	10,000 (D1)	56,972	100,000
	100,000 (D2)	566,751	1,000,000
	500,000 (D3)	2,831,668	2,000,000

We estimate all the parameters associated with alternative utilities and all the scale parameters μ_k, $\forall k \in \mathcal{N}\backslash C$. There are many parameters α_{ka}, $\forall k \in \mathcal{N}, a \in \mathcal{N}(k)$ and estimating all of them is impractical. For the sake of illustration, we estimate four of them (arbitrarily chosen) for the models CNL-1 and CNL-2 and fix the others at their true values. So model CNL-1 has 10 parameters to be estimated and 7 constraints, model CNL-2 has 211 parameters to be estimated and 202 constraints.

For the model N-MEV we estimate 56 scale parameters μ_k, $k \in \mathcal{N}\backslash C$ and all parameters α_{ka} $\forall a \in \mathcal{N}(k)$ and $k, a \in \mathcal{N}\backslash C$. The remaining parameters are fixed to their true values. Consequently, the N-MEV model has 305 constraints and 312 parameters to be estimated.

Algorithm 1: Log-likelihood(β).

begin
 # 1. Compute the value functions $Y = 0$;
do
$Y_{prev} = Y$;
$Y = [M \circ X(Y)].e + b$;
while $Y \neq Y_{prev}$;
 # 2. Compute the gradients of the value functions, nb_{pars} *is the number of parameters, see Appendix D*
for *i=1* **to** nb_{pars} **do**

$$\frac{\partial V}{\partial \beta_i} = (I - L)^{-1}(Re + h);$$

end;
 # 3. Compute the choice probabilities
$F = (I - \mathcal{P}^T)^{-1}D$;
 # 4. Compute the gradients of the choice probabilities, see Appendix D
for *i=1* **to** nb_{pars} **do**

$$\frac{\partial F}{\partial \beta_i} = (I - \mathcal{P}^T)^{-1}\frac{\partial \mathcal{P}^T}{\partial \beta_i}F;$$

end;
 # 5. Compute the log-likelihood function and its gradient

Table 2
Computational time (in seconds).

Models	Choice sets	Computational time		
		LL and gradient	Estimation	Elasticities
CNL-1	D1	0.14	4.61	0.04
	D3	1.75	301.55	0.48
	D5	7.84	1462.9	2.27
CNL-2	D1	1.44	84.62	0.06
	D2	18.29	1073.93	0.59
	D3	88.52	15,566.32	3.08
N-MEV	D1	2.74	715.6	0.12
	D2	38.47	11,464.06	1.14
	D3	162.52	52,168.92	5.52

The pseudo-code in Algorithm 1 is used to compute the log-likelihood function and its gradient based on the methods proposed in Section 5. We use the MATLAB *fmincon* interior point algorithm with BFGS Hessian approximation to solve the constrained optimization problems. For the large models (CNL-2 and N-MEV), the Hessian is approximated by the limited memory BFGS (L-BFGS) (Liu and Nocedal, 1989).

Our code is implemented in MATLAB 2015 and we use an Intel(R) 3.20 GHz machine with a x64-based processor. It is running in the Window 8 64-bit Operating System. The machine has a multi-core processor but we only use one processor to estimate the models as the code is not parallelized.

6.2. Computational time results

In Table 2, we report the computation time required to estimate the models for the different data sets. Given that there are many parameters, we do not report the parameter estimates but note that they are not significantly different from their true values according *t*-tests at the 0.05 significance level.

For the choice set D1 (10,000 alternatives) it takes around 5 s to estimate model CNL-1 and less than 2 min for estimating CNL-2. For the largest choices set (500,000 alternatives), the estimation requires approximately 20 min for model CNL-1 and around 4 h for model CNL-2. It is fast to compute elasticities, 5 s for the largest choice set with CNL-2. We note that for all the cross-nested logit models, the optimization algorithm needs around 100–300 iterations to converge.

We now turn our attention to the results for model N-MEV which is the most time consuming model to estimate. The non-linear optimization algorithm requires from 300 to 500 iterations to converge (the algorithm needs less than 50 iterations if the scale parameters μ_k are not estimated). The results show that we can estimate a large-scale three-levels network MEV model (N-MEV) with the largest choice set (500,000 alternatives) in approximately 14 h.

Finally, we note that if the alternative utilities are observation specific, then we need to compute the value functions and their gradients for each observation. In this case, the computational time increases proportionally to the number of observations if the code is not parallelized. For example, if the number of observations is 1000, then the computation of the log-likelihood and its gradients would take approximately 140 s for CNL-1 with 10,000 alternatives, 1750 s for CNL-1 with 100,000 alternatives, and more than 5 h for CNL-2 with 100,000 alternatives. For large number of observations it is obvious that the log-likelihood function and its gradient should be computed using several processors in parallel.

6.3. Comparisons of computational time

Based on the same experimental setting as in the previous section, the objective of this section is to provide a discussion of computational time of the dynamic programming approach as well as to present comparisons with the standard approach. The computations using the dynamic approach can be done as sparse matrix operations while the standard approach relies on recursive operations. The proposed matrix operation formulation allows us to benefit from existing efficient algorithms for solving large sparse systems of linear equations. This is the main reason for the reduction in computational time. In the following we provide more details. First, we provide an abstract discussion of computational time for the two approches in the case of observation specific utilities. This means that the value functions and their gradients need to be solved for each observation. Second, we present a comparison of computational time using the previously presented experiment.

Algorithm 1 shows the pseudo code to compute the log-likelihood and its gradient using matrix operations. All the matrices in Algorithm 1 have the size of $|\mathcal{N}| \times |\mathcal{N}|$. They are sparse and the number of non-zero elements is $|\mathcal{A}|$, i.e., the number of arcs in the graph. So the complexity of matrix operations is typically proportional to $|\mathcal{A}|$ (Gilbert et al., 1992). We note that, when the value functions are observation specific, given an observation, four steps #1, #2, #3, and #4 need to be performed to compute the choice probability as well as its gradients. In the following we discuss the computational complexity of those steps in more detail.

We denote $\phi(|\mathcal{A}|)$ the computational time for each iteration in the value iteration method and $\varphi(|\mathcal{A}|)$ the computational time for solving a system of linear equations $Ax = b$, where A is a matrix of size $|\mathcal{N}| \times |\mathcal{N}|$ with $|\mathcal{A}|$ non-zero elements. The

Algorithm 2: getYrecur(k) # *Compute the CPGF at node k.*

```
begin
    forall the a ∈ 𝒩(k) do
        if Y(a) has not been computed then
            if a ∈ C then
                Y(a) = e^{U_a};
            else
                Y(a) = getYrecur(a) ;
    return b_k + Σ_{a∈𝒩(k)} M_ka Y(a)^{μ_k/μ_a};
```

Table 3
Computational time comparisons (in seconds).

$\hat{\phi}(\lvert\mathcal{A}\rvert)$	$\hat{\varphi}(\lvert\mathcal{A}\rvert)$	$\hat{\tau}^1(\lvert\mathcal{A}\rvert), \hat{\tau}^2(\lvert\mathcal{A}\rvert)$
[0.28;0.32]	[0.035;0.045]	[4.5;5.5]

total computational time to evaluate a choice probability and its gradient using a single processor can be approximated by

$$\mathcal{CT}^M = nb_{iters} \times \phi(|\mathcal{A}|) + (2nb_{pars} + 1) \times \varphi(|\mathcal{A}|), \tag{25}$$

where nb_{iters} is the number of iterations required for the value iteration method.

The estimation of the network MEV model can also be carried out using the recursive functions proposed by Daly and Bierlaire (2006). In this case, a choice probability and its gradients can be computed based on the first 4 steps in Algorithm 1 but using recursive functions instead of matrix operations. More precisely, the CPGFs and their gradients can be computed using (1), and the choice probabilities using (2). Note that under the hypotheses of Proposition 1 we have $Y(a) = G_a$ and $\Omega_{ka} = P(a|k),\ \forall k, a \in \mathcal{N},\ a \in \mathcal{N}(k)$. Algorithm 2 shows a recursive implementation to compute $Y(k)$ (or G_k), for a node $k \in \mathcal{N}$. The computational time of a recursive implementation is typically proportional to $|\mathcal{A}|$. The gradients of the CPGFs, the choice probabilities as well as their gradients can be computed by a similar recursive implementation as in Algorithm 2. Finally, the computational time to compute an observation probability and its gradients using recursive functions can be approximated by

$$\mathcal{CT}^R = (nb_{pars} + 1) \times \tau^1(|\mathcal{A}|) + (nb_{pars} + 1) \times \tau^2(|\mathcal{A}|), \tag{26}$$

where $(nb_{pars} + 1) \times \tau^1(|\mathcal{A}|)$ is the time associated with the evaluation of the CPGFs and their gradients, and $(nb_{pars} + 1) \times \tau^2(|\mathcal{A}|)$ is the time associated with the choice probability and the respective gradients. We note that, for a given alternative $j \in C$, (2) needs to be evaluated in order to compute $P(j|C)$.

We now turn our attention to a comparison between the computational times given in (25) and (26). We start with a discussion of the four values $\tau^1(|\mathcal{A}|)$, $\tau^2(|\mathcal{A}|)$, $\phi(|\mathcal{A}|)$ and $\varphi(|\mathcal{A}|)$. When solving sparse systems of linear equations, iterative methods are often used. Many studies have focused on how to speed up large-scale sparse matrix operations, especially solving sparse systems of linear equations (e.g. Gilbert et al., 1992; Saad, 2003; Barrett et al., 1994). These methods produce an approximate solution after a finite number of iterations through a matrix-vector product or an abstract linear operator (Saad, 2003). This leads to the fact that $\varphi(|\mathcal{A}|)$ only involves product or linear operators. Since $\phi(|\mathcal{A}|)$ involves raising each element of a matrix by a real exponent, $\phi(|\mathcal{A}|)$, $\tau^1(|\mathcal{A}|)$ and $\tau^2(|\mathcal{A}|)$ are expected to be larger than $\varphi(|\mathcal{A}|)$.

We now compare $\mathcal{CT}^R$ with $\mathcal{CT}^M$. Since $\varphi(|\mathcal{A}|)$ is expected to be smaller than $\phi(|\mathcal{A}|)$, $\tau^1(|\mathcal{A}|)$, $\tau^2(|\mathcal{A}|)$, and $\phi(|\mathcal{A}|)$, $\tau^1(|\mathcal{A}|)$, $\tau^2(|\mathcal{A}|)$ may be not significantly different, we expect, for network MEV models with large number of alternatives, $\mathcal{CT}^M$ to be smaller than $\mathcal{CT}^R$.

We provide a numerical example using model CNL-2 with choice set D3 and report the ranges of $\hat{\phi}(|\mathcal{A}|)$, $\hat{\varphi}(|\mathcal{A}|)$, $\hat{\tau}^1(|\mathcal{A}|)$ and $\hat{\tau}^2(|\mathcal{A}|)$ in Table 3 (these are approximated values of $\phi(|\mathcal{A}|)$, $\varphi(|\mathcal{A}|)$, $\tau^1(|\mathcal{A}|)$, $\tau^2(|\mathcal{A}|)$ when performing Algorithms 1 and 2 in MATLAB). We note that $nb_{iter} = 3$ in Algorithm 1, and in order to achieve good performance, we compute the final sum in Algorithm 2 as a matrix operation instead of a for-loop. As expected, $\hat{\varphi}(|\mathcal{A}|) < \hat{\phi}(|\mathcal{A}|)$ and $\hat{\varphi}(|\mathcal{A}|) < \min\{\hat{\tau}^1(|\mathcal{A}|), \hat{\tau}^2(|\mathcal{A}|)\}$. Moreover, $\hat{\phi}(|\mathcal{A}|) < \min\{\hat{\tau}^1(|\mathcal{A}|), \hat{\tau}^2(|\mathcal{A}|)\}$. These results indicate that the matrix operations need less time than the recursive functions to solve systems of sparse linear equations. Consequently, the computational time for evaluating a choice probability and its gradient is also smaller, compared to the recursive approach. This is shown in Table 4 where we report the computational times for performing the first 4 steps of Algorithm 1. In the following we comment on each of these steps.

In Step # 1 we compute the value functions, relying on large sparse linear systems. In MATLAB, solving such systems is faster than computing the recursive functions as we can rely on matrix operations and efficient linear systems solvers. The two approaches are otherwise similar and both update values recursively. To illustrate this fact, we consider a cycle-free network. If we assign each node a rank corresponding to the number of the arcs of the longest path from this node to the

Table 4
Computational time (in seconds) comparisons of matrix and recursive operations.

Step	Matrix operations	Recursive functions
#1	1.21	4.55
#2	9.31	1117.8
#3	0.031	5.975
#4	8.90	1258.7

destinations, then we can show that (see Appendix B), during the value iteration method, the value function at the nodes of rank (0) (i.e. destinations) are updated to their fixed point values after the first iteration, and the value function at nodes of rank (1) are updated after the second iteration, and so forth. There are remarkable differences between matrix operations and recursive functions in Steps #2, #3 and #4. The differences in these steps are again due to the good performance of solving systems of linear equations. The differences between Steps #3 and #4 are mainly due to the large number of parameters to be estimated (the model CNL-2 has 211 parameters). If we estimate the model with 1000 observations, in order to evaluate the log-likelihood and its gradients, the Steps #3 and #4 take 31 s and 8900 s, respectively, if using matrix operations while they are 5975 and $\approx 10^6$ s, respectively, for the recursive functions. Hence, the matrix operation approach is still over a hundred times faster.

It is important to note that, if the value functions are not observation specific, the computational time using the recursive functions still increases proportionally with the number of observations while this number does not significantly affect the computational time using the matrix operations, as shown in the previous section. Hence, in real applications, if the observations can be aggregated into different groups such that the value functions are group specific, then we can solve the value functions once for each group and reduce the computational cost. This is also an advantage of the dynamic programming approach, compared to the recursive functions. We acknowledge that there may exist better implementations of the recursive functions that we are unaware of. Nevertheless, the results clearly show the potential of the dynamic programming approach.

6.4. Application to real data

In this section we illustrate the performance of Algorithm 1 on real data by estimating a cross-nested logit model. The data is confidential so we cannot provide a descriptive analysis, however it still serves our purpose well since the objective is to assess the performance in terms of computational time and to illustrate that the approach can be used on real data.

The data comes from a retail business and corresponds to records of product purchases. It was collected from week 35th to 52nd in 2013 in 229 stores across the U.S. and contains 134,320 observations. The choice set contains 374 alternatives and the main attribute is price. In addition, the alternatives are categorized according to their characteristics and these are captured by 111 constants. We specify a cross-nested structure grouping the alternatives according these characteristics which results in 111 nests and 1981 links in the network of correlation structure. We estimate the μ parameters and parameters associated with products' attributes, the α parameters are kept equal for each product. So there are 223 parameters to be estimated.

While the number of alternatives is considerably smaller than in the experimental setup, the choice sets are observation specific. More precisely, the data set contains 3565 different observed choice sets (subset of products that are available in a given store) and the numbers of products in the choice sets vary from 43 to 162. The value functions are choice set specific, meaning that in order to compute the LL function, the value functions needs to be solved 3565 times. We need 5.5 min to compute the log-likelihood functions and its gradients. The outer algorithm converges in 150 iterations, so in total we need approximately 14 h to estimate the model. We note that the recursive approach becomes too expensive for this data set because we need several hours to evaluate the log-likelihood function. The estimation would hence take several months, under the same experimental setting. Finally we note that the dynamic programming approach is fast for forecasting. We can compute the choice probabilities for more than 3000 choice sets in a few seconds.

7. Conclusion

In this paper we have introduced a novel approach to construct the network MEV model using the dynamic discrete choice framework. We have shown that under certain conditions the resulting models is consistent with McFadden's MEV theory and generalizes the network MEV model. We presented a new estimation method that is convenient to use for estimating network-based MEV models with large choice sets i.e. computing choice probabilities using the expected flows in the graph, estimating the model using the nested fixed point algorithm and computing the model derivatives and elasticities by solving systems of linear equations. The proposed formulation can be used to compute predicted probabilities in a fast and straightforward way.

We have presented numerical experiments using simulated and real data. The results on simulated data indicate that we are able to quickly estimate the cross-nested logit and multi-levels network MEV models with large choice sets and large

number of observations. We also provided a discussion of the computational time of the dynamic programming approach compared to recursive operations. The key advantage of the proposed formulation is that computations to a large extent rely on solving sparse systems of linear equations which makes it possible to benefit from existing efficient solution methods. This is the main explanation for the short computational times. We also show that the model can be used to analyze real choices when choice sets vary over observations. In this case the estimation can be done in reasonable time and the model allows to compute predicted choice probabilities fast.

A sampling of alternatives approach can also be used to estimate models when choice sets are very large or even infinite. In this case, utilities need to be corrected in order to obtain consistent (but not efficient) parameter estimates. Such corrections are easy to compute for logit models (McFadden, 1978) but less so for other MEV models (Guevara and Ben-Akiva, 2013). While a sampling of alternatives approach represents an option for estimation, it is unclear how to use the same approach for prediction. The proposed formulation allows to do both in short computational time.

Finally we note that the estimation code is implemented in MATLAB and is available on GitHub.[2]

Acknowledgment

This research was funded by the National Sciences and Engineering Research Council of Canada, discovery grant 435678-2013. Mogens Fosgerau was funded by the Danish Innovation Fund. We have benefited from valuable discussions with Michel Bierlaire.

Appendix A. Proof of Theorem 1

This appendix presents a proof for Theorem 1. We first introduce some definitions.

Definition 1. Given a set of nodes $\mathcal{U} \subseteq \mathcal{N}$ and $k, a \in \mathcal{U}$, we denote $\Theta(k, a, \mathcal{U})$ be the set of paths of strictly positive lengths that connect k and a via nodes in $\mathcal{U}$, i.e., $\Theta(k, a, \mathcal{U}) = \{[k_0, \dots, k_J] \,|\, k_0 = k; k_J = a; k_i \in \mathcal{U}; k_{i+1} \in \mathcal{N}(k_i)\ \forall i = 0, \dots, J-1; J \geq 1\}$.

Note that if $k = a$ and $\Theta(k, a, \mathcal{U}) \neq \emptyset$, then $\Theta(k, a, \mathcal{U})$ contains all the cycles going from k and come back to k via nodes in $\mathcal{U}$.

Definition 2. Given a set of nodes $\mathcal{U} \subseteq \mathcal{N}$ and $k, a \in \mathcal{U}$, $\pi(k, a, \mathcal{U}) = 1$ if $\Theta(k, a, \mathcal{U}) \neq \emptyset$ or $k \equiv a$, and $\pi(k, a, \mathcal{U}) = 0$ otherwise.

Definition 3. Given a set $\mathcal{U} \subseteq \mathcal{N}$ and a node $k \in \mathcal{U}$, $\mathcal{K}(k, \mathcal{U})$ is the set of nodes that are connected to k via nodes in $\mathcal{U}$, i.e., $\mathcal{K}(k, \mathcal{U}) = \{a \,|\, \pi(k, a, \mathcal{U}) = 1\}$, $\forall k \in \mathcal{U}$.

Definition 4. Given a set $\mathcal{U} \subseteq \mathcal{N}$, we define $\mathcal{Y}(\mathcal{U})$ to be a system of non-linear equations which contains the following equations

$$Y_k = \begin{cases} \sum_{a \in \mathcal{N}(k) \cap \mathcal{U}} M_{ka} Y_a^{\mu_k/\mu_a} & \text{if } k \in \mathcal{U} \backslash C \\ e^{U_k} & \text{if } k \in \mathcal{U} \cap C \\ 0 & \text{if } k \notin C,\ \mathcal{N}(k) \cap \mathcal{U} = \emptyset, \end{cases} \tag{27}$$

where M_{ka}, $\forall a, k \in \mathcal{N}$, are defined in (19).

Lemma 1. *Under the hypotheses in Theorem 1, given $\mathcal{U} \subseteq \mathcal{N}$ and two nodes $a, k \in \mathcal{U}$, if $\pi(k, a, \mathcal{U}) = 1$ and $\pi(a, k, \mathcal{U}) = 1$ then $\mu_a = \mu_k$.*

Proof. The lemma is obviously verified, as $\mu_k \leq \mu_a$ if $\pi(k, a, \mathcal{U}) = 1$, $\forall k, a \in \mathcal{U}$. □

Lemma 2. *Under the hypotheses of Theorem 1 and given a set $\mathcal{U} \subseteq \mathcal{N}$. If $(Y_k, k \in \mathcal{U})$ is a vector solution to $\mathcal{Y}(\mathcal{U})$, then given a node $k_0 \in \mathcal{U} \backslash C$ and $\mathcal{N}(k_0) \cap \mathcal{U} \neq \emptyset$, there exist a set of vectors $\{Y^a, a \in \mathcal{N}(k_0) \cap \mathcal{U}\}$ and a scalar $\lambda > 0$ which is independent of Y_i, $\forall i \in C$, such that*

(i) *Y^a is a vector solution to system $\mathcal{Y}(\mathcal{K}(a, \mathcal{U}) \backslash \{k_0\})$, and*
(ii) *$Y_{k_0} = \lambda \sum_{a \in \mathcal{N}(k_0) \cap \mathcal{U}} M_{k_0 a} (Y_a^a)^{\mu_{k_0}/\mu_a}$,*

where Y_a^a is the entry in Y^a corresponding to node a, $\forall a \in \mathcal{N}(k_0) \cap \mathcal{U}$.

Proof. From (27), for given a node $k \in \mathcal{K}(k_0, \mathcal{U}) \backslash C$ and $\mathcal{N}(k) \cap \mathcal{U} \neq \emptyset$ we can write the recursive equation as

$$Y_k = \sum_{\substack{a \in \mathcal{N}(k) \cap \mathcal{U} \\ \pi(a, k_0, \mathcal{U}) = 0}} M_{ka} Y_a^{\mu_k/\mu_a} + \sum_{\substack{a \in \mathcal{N}(k) \cap \mathcal{U} \\ \pi(a, k_0, \mathcal{U}) = 1}} M_{ka} Y_a^{\mu_k/\mu_a}. \tag{28}$$

[2] https://github.com/maitien86/

We have $\pi(k_0, k, \mathcal{U}) = 1$, so for each $a \in \mathcal{N}(k)$ if $\pi(a, k_0, \mathcal{U}) = 1$ then $\pi(k, a, \mathcal{U}) = \pi(a, k, \mathcal{U}) = 1$. According to Lemma 1 we have $\mu_a = \mu_k$ and (28) can be written as

$$Y_k = \sum_{\substack{a \in \mathcal{N}(k) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=0}} M_{ka} Y_a^{\mu_k/\mu_a} + \sum_{\substack{a \in \mathcal{N}(k) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=1}} M_{ka} Y_a. \tag{29}$$

In a similar way for each Y_a such that $\pi(a, k_0, \mathcal{U}) = 1$, we can write Y_a as a sum of two parts as in (28). The first part involves nodes that are not connected to k_0 and the second involves nodes that are connected to k_0. Hence, the value of Y_k can be written as

$$Y_k = \sum_{\substack{\sigma \in \Theta(k,a,\mathcal{U}) \\ a \in \mathcal{K}(k,\mathcal{U}) \\ \pi(a,k_0,\mathcal{U})=0}} v(\sigma) Y_a^{\mu_k/\mu_a} + \sum_{\sigma \in \Theta(k,k_0,\mathcal{U})} v(\sigma) Y_{k_0} \tag{30}$$

where $v(\sigma) = \prod_{i=0}^{J-1} M_{h_i h_{i+1}}$ given a path $\sigma = \{h_0, \ldots, h_J\}$.

Now using (27) and then (29) and (30), we can write Y_{k_0} as

$$\begin{aligned} Y_{k_0} &= \sum_{a \in \mathcal{N}(k_0) \cap \mathcal{U}} M_{k_0 a} Y_a^{\mu_{k_0}/\mu_a} \\ &= \sum_{\substack{a \in \mathcal{N}(k_0) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=0}} M_{k_0 a} Y_a^{\mu_{k_0}/\mu_a} + \sum_{\substack{a \in \mathcal{N}(k_0) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=1}} M_{k_0 a} \left(\sum_{\substack{\sigma \in \Theta(a,c,\mathcal{U}) \\ c \in \mathcal{K}(a,\mathcal{U}) \\ \pi(c,k_0,\mathcal{U})=0}} v(\sigma) Y_c^{\mu_a/\mu_c} + \sum_{\sigma \in \Theta(a,k_0,\mathcal{U})} v(\sigma) Y_{k_0} \right) \\ &= \sum_{\substack{a \in \mathcal{N}(k_0) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=0}} M_{k_0 a} Y_a^{\mu_{k_0}/\mu_a} + \sum_{\substack{a \in \mathcal{N}(k_0) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=1}} M_{k_0 a} \sum_{\substack{\sigma \in \Theta(a,c,\mathcal{U}) \\ c \in \mathcal{K}(a,\mathcal{U}) \\ \pi(c,k_0,\mathcal{U})=0}} v(\sigma) Y_c^{\mu_a/\mu_c} + \sum_{\sigma \in \Theta(k_0,k_0,\mathcal{U})} v(\sigma) Y_{k_0}. \end{aligned}$$

This is equivalent to

$$\left(1 - \sum_{\sigma \in \Theta(k_0,k_0,\mathcal{U})} v(\sigma)\right) Y_{k_0} = \sum_{\substack{a \in \mathcal{N}(k_0) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=0}} M_{k_0 a} Y_a^{\mu_{k_0}/\mu_a} + \sum_{\substack{a \in \mathcal{N}(k_0,\mathcal{U}) \\ \pi(a,k_0,\mathcal{U})=1}} M_{k_0 a} \sum_{\substack{\sigma \in \Theta(a,c,\mathcal{U}) \\ c \in \mathcal{K}(a,\mathcal{U}) \\ \pi(c,k_0,\mathcal{U})=0}} v(\sigma) Y_c^{\mu_a/\mu_c}.$$

If we denote $\lambda = \frac{1}{1 - \sum_{\sigma \in \Theta(k_0,k_0,\mathcal{U})} v(\sigma)}$, then we get

$$Y_{k_0} = \lambda \left(\sum_{\substack{a \in \mathcal{N}(k_0) \cap \mathcal{U} \\ \pi(a,k_0,\mathcal{U})=0}} M_{k_0 a} Y_a^{\mu_{k_0}/\mu_a} + \sum_{\substack{a \in \mathcal{N}(k_0,\mathcal{U}) \\ \pi(a,k_0,\mathcal{U})=1}} M_{k_0 a} \sum_{\substack{\sigma \in \Theta(a,c,\mathcal{U}) \\ c \in \mathcal{K}(a,\mathcal{U}) \\ \pi(c,k_0,\mathcal{U})=0}} v(\sigma) Y_c^{\mu_a/\mu_c} \right). \tag{31}$$

It is obvious that λ is independent of Y_i, $\forall i \in C$, and according to (31) we have $\lambda > 0$, as Y_{k_0} and the term in the parentheses are positive. The lemma can be proved by considering each part in the sum in the parentheses in (31) as follows.

We consider the first part of the sum in the parentheses in (31). Given a node $a_1 \in \mathcal{N}(k_0) \cap \mathcal{U}$ with $\pi(a_1, k_0, \mathcal{U}) = 0$, we have $k_0 \notin \mathcal{K}(a_1, \mathcal{U})$ and consequently $\mathcal{K}(a_1, \mathcal{U}) \backslash \{k_0\} \equiv \mathcal{K}(a_1, \mathcal{U})$. Now we will prove that $(Y_{a'}, a' \in \mathcal{K}(a_1, \mathcal{U}))$ is a vector solution to the system $\mathcal{Y}(\mathcal{K}(a_1, \mathcal{U}) \backslash \{k_0\})$. Under the hypotheses of the lemma and the definition in (27) we have that $(Y_{k'}, k' \in \mathcal{U})$ is a vector solution to $\mathcal{Y}(\mathcal{U})$. So given a node $a' \in \mathcal{K}(a_1, \mathcal{U}) \subset \mathcal{U}$ we have the following system of equations

$$Y_{a'} = \begin{cases} \sum_{c \in \mathcal{N}(a') \cap \mathcal{U}} M_{a'c} Y_c^{\mu_{a'}/\mu_c} & \text{if } a' \in \mathcal{U} \backslash C \\ e^{U_{a'}} & \text{if } a' \in \mathcal{U} \cap C \\ 0 & \text{if } a' \notin C, \mathcal{N}(a') \cap \mathcal{U} = \emptyset. \end{cases} \tag{32}$$

The following remarks are obviously verified: (i) $\mathcal{N}(a') \cap \mathcal{U} \equiv \mathcal{N}(a') \cap \mathcal{K}(a_1, \mathcal{U})$, (ii) $(\mathcal{U} \backslash C) \cap \mathcal{K}(a_1, \mathcal{U}) \equiv \mathcal{K}(a_1, \mathcal{U}) \backslash C$, (iii) $(\mathcal{U} \cap C) \cap \mathcal{K}(a_1, \mathcal{U}) \equiv \mathcal{K}(a_1, \mathcal{U}) \cap C$. From (i), (ii) and (iii), (32) can be written as

$$Y_{a'} = \begin{cases} \sum_{c \in \mathcal{N}(a') \cap \mathcal{K}(a_1,\mathcal{U})} M_{a'c} Y_c^{\mu_{a'}/\mu_c} & \text{if } a' \in \mathcal{K}(a_1, \mathcal{U}) \backslash C \\ e^{U_{a'}} & \text{if } a' \in \mathcal{K}(a_1, \mathcal{U}) \cap C \\ 0 & \text{if } a' \notin C, \mathcal{N}(a') \cap \mathcal{K}(a_1, \mathcal{U}) = \emptyset, \end{cases}$$

leading to the fact that $(Y_{a'}, a' \in \mathcal{K}(a_1, \mathcal{U}))$ is a vector solution to the system $\mathcal{Y}(\mathcal{K}(a_1, \mathcal{U}))$. Moreover, as mentioned earlier,

$\mathcal{K}(a_1,\mathcal{U})\backslash\{k_0\} \equiv \mathcal{K}(a_1,\mathcal{U})$, so we can write the first part of the sum in the parentheses in (31) as

$$\sum_{\substack{a_1\in\mathcal{N}(k_0)\cap\mathcal{U}\\ \pi(a_1,k_0,\mathcal{U})=0}} M_{k_0a_1}Y_a^{\mu_{k_0}/\mu_{a_1}} = \sum_{\substack{a_1\in\mathcal{N}(k_0)\cap\mathcal{U}\\ \pi(a_1,k_0,\mathcal{U})=0}} M_{k_0a_1}(Y_{a_1}^{a_1})^{\mu_{k_0}/\mu_{a_1}}, \tag{33}$$

where Y^{a_1} is a vector solution to $\mathcal{Y}(\mathcal{K}(a_1,\mathcal{U})\backslash\{k_0\})$.

We now consider the second part of the sum in the parentheses in (31). Given $a_2 \in \mathcal{N}(k_0)\cap\mathcal{U}$ such that $\pi(a_2,k_0,\mathcal{U})=1$, we define a vector $(\bar{Y}_c, c\in\mathcal{K}(a_2,\mathcal{U})\backslash\{k_0\})$ with entries

$$\bar{Y}_c = Y_c - \sum_{\sigma\in\Theta(c,k_0,\mathcal{U})} v(\sigma)Y_{k_0}, \quad \forall c\in\mathcal{K}(a_2,\mathcal{U})\backslash\{k_0\}. \tag{34}$$

Based on (30), for each $c\in\mathcal{K}(a_2,\mathcal{U})\backslash\{k_0\}$, $c\notin C$ and $\mathcal{N}(c)\cap\mathcal{U}\neq\emptyset$ we have

$$\begin{aligned}\bar{Y}_c &= \sum_{\substack{\sigma\in\Theta(c,d,\mathcal{U})\\ d\in\mathcal{K}(c,\mathcal{U})\\ \pi(d,k_0,\mathcal{U})=0}} v(\sigma)Y_d^{\mu_c/\mu_d}\\ &= \sum_{\substack{c'\in\mathcal{N}(c)\cap\mathcal{U}\\ \pi(c',k_0,\mathcal{U})=0}} M_{cc'}Y_{c'}^{\mu_c/\mu_{c'}} + \sum_{\substack{c'\in\mathcal{N}(c)\cap\mathcal{U}\\ \pi(c',k_0,\mathcal{U})=1\\ c'\neq k_0}} \left(M_{cc'} \sum_{\substack{\sigma\in\Theta(c',d,\mathcal{U})\\ d\in\mathcal{K}(c',\mathcal{U})\\ \pi(d,k_0,\mathcal{U})=0}} v(\sigma)Y_d^{\mu'_c/\mu_d} \right)\\ &= \sum_{\substack{c'\in\mathcal{N}(c)\cap\mathcal{U}\\ \pi(c',k_0,\mathcal{U})=0}} M_{cc'}Y_{c'}^{\mu_c/\mu_{c'}} + \sum_{\substack{c'\in\mathcal{N}(c)\cap\mathcal{U}\\ \pi(c',k_0,\mathcal{U})=1\\ c'\neq k_0}} M_{cc'}\bar{Y}_{c'}.\end{aligned} \tag{35}$$

Moreover, from the definition in (34) we note that $\bar{Y}_{c'} = Y_{c'}$ if $\pi(c',k_0,\mathcal{U})=0$, $\forall c'\in\mathcal{K}(a_2,\mathcal{U})$, so (35) can be written as

$$\bar{Y}_c = \sum_{\substack{c'\in\mathcal{N}(c)\cap\mathcal{U}\\ c'\neq k_0}} M_{cc'}(\bar{Y}_{c'})^{\mu_c/\mu_{c'}}, \quad \forall c\in\mathcal{K}(a_2,\mathcal{U})\backslash\{C\cup\{k_0\}\}, \mathcal{N}(c)\cap\mathcal{U}\neq\emptyset. \tag{36}$$

Moreover, it is obvious to verify that (iv) $(\mathcal{N}(c)\cap\mathcal{U})\backslash\{k_0\} \equiv \mathcal{N}(c)\cap\mathcal{K}(a_2,\mathcal{U})\backslash\{k_0\}$, (v) if $c\in C$ then $\bar{Y}_c = Y_c = e^{U_c}$ and (vi) if $\mathcal{N}(c)\cap\mathcal{U}\backslash\{k_0\}=\emptyset$ then $\bar{Y}_c = 0$. From (iv), (v), (vi), (36) and the definition in (27) we have that $\bar{Y}$ is a vector solution to $\mathcal{Y}(\mathcal{K}(a_2,\mathcal{U})\backslash\{k_0\})$. In summary, according to (35), for each $a_2\in\mathcal{N}(k_0)\cap\mathcal{U}$ and $\pi(a_2,k_0,\mathcal{U})=1$, there exists a vector $\bar{Y}$ which is a solution to $\mathcal{Y}(\mathcal{K}(a_2,\mathcal{U})\backslash\{k_0\})$ such that

$$\bar{Y}_{a_2} = \sum_{\substack{\sigma\in\Theta(a_2,c,\mathcal{U})\\ c\in\mathcal{K}(a_2,\mathcal{U})\\ \pi(c,k_0,\mathcal{U})=0}} v(\sigma)Y_c^{\mu_{a_2}/\mu_c}.$$

Consequently, there exits vectors Y^{a_2} which are solutions to $\mathcal{Y}(\mathcal{K}(a_2,\mathcal{U})\backslash\{k_0\})$, $\forall a_2\in\mathcal{N}(k_0)\cap\mathcal{U}, \pi(a_2,k_0,\mathcal{U})=1$, such that the second part of the sum in the parentheses in (31) can be written as

$$\sum_{\substack{a_2\in\mathcal{N}(k_0,\mathcal{U})\\ \pi(a,k_0,\mathcal{U})=1}} M_{k_0a_2} \sum_{\substack{\sigma\in\Theta(a_2,c,\mathcal{U})\\ c\in\mathcal{K}(a_2,\mathcal{U})\\ \pi(c,k_0,\mathcal{U})=0}} v(\sigma)Y_c^{\mu_{a_2}/\mu_c} = \sum_{\substack{a_2\in\mathcal{N}(k_0,\mathcal{U})\\ \pi(a_2,k_0,\mathcal{U})=1}} M_{k_0a_2}Y_{a_2}^{a_2}. \tag{37}$$

Finally, the lemma is proved using (31), (33) and (37). □

Lemma 3. *Under the hypotheses in Theorem 1, Y_r is a μ_r-MEV CPGF.*

Proof. We prove this lemma by using the results from Lemma 2. We first create a tree structure $\mathcal{T}$ where each node in $\mathcal{T}$ represents a pair $(k,\mathcal{U})$, where $k\in\mathcal{N}$ and $\mathcal{U}\subseteq\mathcal{N}$. We create nodes in $\mathcal{T}$ recursively as follows.

The root of $\mathcal{T}$ corresponds to $(r,\mathcal{N})$. For each created node $t(k,\mathcal{U})\in\mathcal{T}$, if $k\in C$ or $\mathcal{N}(k)\cap\mathcal{U}=\emptyset$ then we consider $t(k,\mathcal{U})\in\mathcal{T}$ as a leaf of $\mathcal{T}$, i.e., it has no successor. Otherwise, for each node $a\in\mathcal{N}(k)\cap\mathcal{U}$ we create a new successor node of t and associate a pair $(a,\mathcal{K}(a,\mathcal{U})\backslash\{k\})$ to the new node. We note that $\mathcal{N}(k)\cap\mathcal{U}=\emptyset$ occurs only when $|\mathcal{U}|=1$. Fig. 4 shows an example where the root r has two successors a_1, a_2 ($a_1,a_2\in\mathcal{N}(r)$). For each two nodes $p(k,\mathcal{U})$ and $q(k',\mathcal{U}')$ in $\mathcal{T}$ such that q is a successor node of p, by definition we have the fact that $|\mathcal{U}|>|\mathcal{U}'|$, so $\mathcal{T}$ does not contain any cycle. Moreover, since we stop creating nodes when $|\mathcal{U}|=1$, the number of nodes in $\mathcal{T}$ is finite. We denote $\mathcal{T}(p)$ be the set of successor nodes of p in $\mathcal{T}$. We recursively associate each note $t(k,\mathcal{U})\in\mathcal{T}$ with a value Z_t such that $Z_t = Y_k^k$, where Y^k is a vector solution to $\mathcal{Y}(\mathcal{U})$, as follows. The root r_t of $\mathcal{T}$ is associated with Y_r, i.e., $Z_{r_t} = Y_r$. Given a node $p(k,\mathcal{U})\in\mathcal{T}$ with the corresponding value Z_p. According to Lemma 2, there exist vectors Y^a, $\forall a\in\mathcal{N}(k)\cap\mathcal{U}$, such that

$$Z_p = \lambda \sum_{a\in\mathcal{N}(k)\cap\mathcal{U}} M_{ka}(Y_a^a)^{\mu_k/\mu_a}, \tag{38}$$

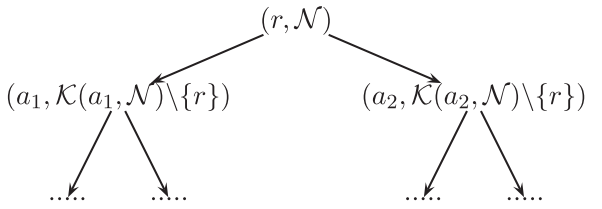

Fig. 4. Tree $\mathcal{T}$.

where Y^a is a vector solution to system $\mathcal{Y}(\mathcal{K}(a,\mathcal{U})\backslash\{k\})$. For each successor node $q(a,\mathcal{K}(a,\mathcal{U})\backslash\{k\})$ of node p we set $Z_q = Y_a^a$ and associate arc (p, q) a non-negative scalar $\eta_{pq} = \lambda M_{ka}$. Note that if $p(k,\mathcal{U})$ is a leaf, i.e. $k \in C$ or $|\mathcal{U}| = 1$, then $Z_p = e^{U_k}$ (if $k \in C$) or $Z_p = 0$ (if $k \notin C$ and $|\mathcal{U}| = 1$). Moreover, we also associate each node $p(k,\mathcal{U})$ with a positive scalar $\xi_p = \mu_k$. Hence, each node $p \in \mathcal{T}$ is associated with a positive value Z_p and a scalar $\xi_p > 0$. Based on (38) the value of Z_p for each node $p(k,\mathcal{U})$ can be defined recursively as

$$Z_p = \begin{cases} \sum_{q\in\mathcal{T}(p)} \eta_{pq}(Z_q)^{\xi_p/\xi_q} & \text{if } p \text{ is not a leaf} \\ e^{U_k} & \text{if } k \in C \\ 0 & \text{if } |\mathcal{U}| = 1 \text{ and } k \notin C. \end{cases} \tag{39}$$

By simply removing nodes p with zero values ($Z_p = 0$) from tree $\mathcal{T}$, (39) can be written as

$$Z_p = \begin{cases} \sum_{q\in\mathcal{T}(p)} \eta_{pq}(Z_q)^{\xi_p/\xi_q} & \text{if } p \text{ is not a leaf} \\ e^{U_k} & \text{otherwise.} \end{cases} \tag{40}$$

(40) is equivalent to the definition of the CPGFs given by (1). Accordingly, Z_{r_t} is a ξ_{r_t}-MEV CPGF, or equivalently Y_r is a μ_r-MEV CPGF. □

Corollary 1. *Under the hypotheses in Theorem 1, Y_k, $\forall k \in \mathcal{N}\backslash C$, is a μ_k – MEV CPGF function.*

Proof. It makes no difference for the statement that the graph may contain cycles since we can consider any node $k \in \mathcal{N}\backslash C$ as a root and apply the proof in Lemma 3 to the sub-graph with nodes $\mathcal{K}(k,\mathcal{N})$ to prove that Y_k is a μ_k-MEV CPGF. □

Lemma 4. *If $G(y) = Y_r$ is a μ_r-MEV CPGF then the choice probability given by the generating function $G(y_i, i \in C) = Y_r$ is*

$$P(i) = \sum_{[k_l,\dots,k_0=i]\in\Phi(i)} \prod_{i=0}^{l-1} P(k_i|k_{i+1}), \quad \forall i \in C.$$

Proof. We derive the probabilities given by an MEV model with the CPGF $G(y) = G(y_i, i \in C) = Y_r$. Each alternative i is associated with the utility $U_i + \epsilon_i$, where vector ϵ is MEV distributed with the CPGF $G(y)$. Since Y_r is a $\mu_r - MEV$ CPGF function, the choice probability is (McFadden, 1978)

$$\begin{aligned} P(i) &= \frac{y_i \frac{\partial G}{\partial y_i}(y)}{\mu_r G(y)} \\ &= \frac{y_i}{\mu_r Y_r}\frac{\partial Y_r}{\partial y_i}, \quad i \in C. \end{aligned}$$

From (10), the partial derivative of Y_k with respect to y_i, $i \in C$ is

$$\frac{\partial Y_k}{\partial y_i} = \sum_{a\in\mathcal{N}(k)} e^{\mu_k v(a|k)} \frac{\mu_k}{\mu_a} Y_a^{\mu_k/\mu_a} \frac{\partial Y_a}{Y_a \partial y_i}, \quad k \in \mathcal{N}\backslash C. \tag{41}$$

Denote $S_k^i = \frac{y_i}{\mu_k Y_k}\frac{\partial Y_k}{\partial y_i}$, $i \in C, k \in \mathcal{N}$. Based on (41), we obtain recursive formulas for S_k^i as

$$S_k^i = \sum_{a\in\mathcal{N}(k)} e^{\mu_k v(a|k)} \frac{Y_a^{\mu_k/\mu_a}}{Y_k} S_a^i = \sum_{a\in\mathcal{N}(k)} P(a|k) S_a^i, \quad \forall k \in \mathcal{N}\backslash C.$$

Note that

$$S_i^i = \frac{y_i}{\mu_i Y_i}\frac{\partial Y_i}{\partial y_i} = \frac{y_i}{\mu_i y_i^{\mu_i}} \mu_i y_i^{\mu_i - 1} = 1,$$

and

$$S_j^i = \frac{y_i}{\mu_i Y_j}\frac{\partial Y_j}{\partial y_i} = 0, \quad \forall j \in C, j \neq i,$$

so the choice probability given by the MEV model is

$$P(i) = S_r^i = \sum_{[k_l,\dots,k_0=i]\in\Phi(i)} \prod_{i=0}^{l-1} P(k_i|k_{i+1}), \quad \forall i \in C.$$

□

Since the choice probability given by Lemma 4 is identical to (5), Theorem 1 is completely proved.

Appendix B. Proof of Theorem 2

We provide a detailed proof for Theorem 2 in this appendix. We first introduce a lemma. For a node $k \in \mathcal{N}$ we define $l(k)$ be the number of arcs of the longest path (defined by the number of links) from k to the destinations. We also call $l(k)$ the level of node k. We then denote $\mathcal{L}(t)$ be the set of nodes of level t: $\mathcal{L}(t) = \{k \in \mathcal{N} | \, l(k) = t\}$. Since the graph is cycle-free, the level of each node is finite. Moreover, the root has the highest level in the graph. We have the following lemma

Lemma 5. *Given $t > 0$, $t \in \mathbb{Z}$, we have $Y_k^{(i)} = Y_k^{(t+1)}$, $\forall i \geq t+1$ and $\forall k \in \cup_{j=1}^{t} \mathcal{L}(j)$.*

Proof. We prove this lemma by induction. For $t = 1$, $\mathcal{L}(t)$ is the set of nodes that connect directly to the destination. So $\forall i \geq 1$ and $\forall k \in \mathcal{L}(1)$, according to (20) we have

$$Y_k^{(i+1)} = \sum_{a\in C} M_{ka} (Y_a^{(i)})^{\mu_k/\mu_a}. \tag{42}$$

Moreover, from (19) and (20) we have $Y_a^{(i)} = e^{U_a}$ $\forall a \in C$, $i \geq 1$. Substituting into (42), we obtain

$$Y_k^{(i+1)} = \sum_{a\in C} M_{ka} (e^{U_a})^{\mu_k/\mu_a} = Y_k^{(2)}, \quad \forall i \geq 1, k \in \mathcal{L}(1),$$

meaning that the result is true for $t = 1$. Now we assume that the result is true for $t \geq 1$

$$Y_k^{(i)} = Y_k^{(t+1)}, \quad \forall i \geq t+1, k \in \cup_{j=1}^{t} \mathcal{L}(j), \tag{43}$$

leading to the fact that

$$Y_k^{(i)} = Y_k^{(t+2)}, \quad \forall i \geq t+2, k \in \cup_{j=1}^{t} \mathcal{L}(j). \tag{44}$$

Given a node $k \in \mathcal{L}(t+1)$, we note that for each node $a \in \mathcal{N}(k)$, $a \in \cup_{j=1}^{t} \mathcal{L}(j)$ (otherwise the level of k would be greater than $t+1$). From (20) we have

$$Y_k^{(i+1)} = \sum_{\substack{a\in\mathcal{N}(k) \\ a\in\cup_{j=1}^{t}\mathcal{L}(j)}} M_{ka} (Y_a^{(i)})^{\mu_k/\mu_a} + b_k, \quad \forall i \geq t+1. \tag{45}$$

Then according to (43), (45) can be written as

$$Y_k^{(i+1)} = \sum_{\substack{a\in\mathcal{N}(k) \\ a\in\cup_{j=1}^{t}\mathcal{L}(j)}} M_{ka} (Y_a^{(t+1)})^{\mu_k/\mu_a} + b_k = Y_k^{(t+2)}, \quad \forall i \geq t+1.$$

This means that $Y_k^{(i)} = Y^{(t+2)}$, $\forall i \geq t+2$, $\forall k \in \mathcal{L}(t+1)$, and from (44) we have

$$Y_k^{(i)} = Y_k^{(t+2)}, \quad \forall i \geq t+2, \quad \forall k \in \cup_{j=1}^{t+1} \mathcal{L}(j).$$

This validates the result for $t+1$, as required. □

Now if we choose $K = l(r) + 1$ ($l(r)$ is the level of the root), based on Lemma 5 we have the following result

$$Y_k^{(i)} = Y_k^{(K)}, \quad \forall i \geq K, \quad \forall k \in \cup_{j=1}^{l(r)} \mathcal{L}(j),$$

and note that $\cup_{j=1}^{l(r)} \mathcal{L}(j) \equiv \mathcal{N}$. Hence, Theorem 2 is proved.

Appendix C. Proof of Proposition 2

We consider matrix $\mathcal{P}$ and note that it can be written in the following canonical form

$$\mathcal{P} = \left(\begin{array}{c|c} \mathbf{Q} & \mathbf{R} \\ \hline \mathbf{0}_1 & \mathbf{0}_2 \end{array} \right),$$

where $\mathbf{Q}$ is a $(|\mathcal{N}| - |C|)$-by-$(|\mathcal{N}| - |C|)$ matrix corresponding to the transient states, $\mathbf{R}$ is a $(|\mathcal{N}| - |C|)$-by-$|C|$ matrix, $\mathbf{0}_1$ is a $|C|$-by-$(|\mathcal{N}| - |C|)$ zero matrix, and $\mathbf{0}_2$ is a $|C|$-by-$|C|$ zero matrix. It is clear that Q is a sub-stochastic matrix and has no recurrent class, leading to the following well-known result

$$\lim_{n\to\infty} \mathbf{Q}^n = 0. \tag{46}$$

Moreover, it is easy to show that $\mathcal{P}^n$ has the following form

$$\mathcal{P}^n = \left(\begin{array}{c|c} \mathbf{Q}^n & \mathbf{Q}^{n-1}R \\ \hline \mathbf{0}_1 & \mathbf{0}_2 \end{array}\right).$$

So according to (46) we have that $\lim_{n\to\infty} \mathcal{P}^n = 0$. This implies that the modulus of the eigenvalues of $\mathcal{P}$ lie within the unit disc, and consequently, $(I - \mathcal{P})^T = I - \mathcal{P}^T$ is invertible.

Appendix D. Derivatives of the log-likelihood function

In this appendix, we derive the derivatives of the log-likelihood function defined in (24). The gradient of the choice probability $P(i_n|C_n)$ can be obtained by taking the Jacobian of vector F which can be derived based on (22). The Jacobian of F with respect to parameter β_j is

$$\frac{\partial F}{\partial \beta_j} = (I - \mathcal{P}^T)^{-1} \frac{\partial \mathcal{P}^T}{\partial \beta_j} F. \tag{47}$$

Hence it requires the first derivative of each element of matrix $\mathcal{P}$ with respect to parameter β_j. Note that

$$\mathcal{P}_{ka} = P(a|k) = M_{ka} \frac{Y_a^{\phi_{ka}}}{Y_k}, \quad \forall k, a \in \mathcal{N}.$$

We take the derivative of a given P_{ka} and obtain

$$\begin{aligned} \frac{\partial \mathcal{P}_{ka}}{\partial \beta_j} = & \frac{\partial M_{ka}}{\partial \beta_j} \frac{Y_a^{\phi_{ka}}}{Y_k} - M_{ka} \frac{Y_a^{\phi_{ka}}}{Y_k^2} \frac{\partial Y_k}{\partial \beta_j} \\ & + M_{ka} \frac{Y_a^{\phi_{ka}}}{Y_k} \left(\frac{\partial \phi_{ka}}{\partial \beta_j} \ln Y_a + \frac{\phi_{ka}}{Y_a} \frac{\partial Y_a}{\partial \beta_j} \right), \end{aligned}$$

involving the derivative of Y_k, $\forall k \in \mathcal{N}$. We take the derivative of a given value Y_k, $k \in \mathcal{N}$ as defined by (20) and obtain

$$\frac{\partial Y_k}{\partial \beta_j} = \sum_{a\in\mathcal{N}} \left(\frac{\partial M_{ka}}{\partial \beta_j} Y_a^{\phi_{ka}} + M_{ka} Y_a^{\phi_{ka}} \left(\frac{\partial \phi_{ka}}{\partial \beta_j} \ln Y_a + \frac{\phi_{ka}}{Y_a} \frac{\partial Y_a}{\partial \beta_j} \right) \right) + \frac{\partial b_k}{\partial \beta_j}. \tag{48}$$

We introduce two matrices S and H of size $|\mathcal{N}| \times |\mathcal{N}|$ with entries

$$\begin{cases} S_{ka} = \dfrac{\partial M_{ka}}{\partial \beta_j} Y_a^{\phi_{ka}} + M_{ka} Y_a^{\phi_{ka}} \dfrac{\partial \phi_{ka}}{\partial \beta_j} \ln Y_a \\ H_{ka} = M_{ka} Y_a^{\phi_{ka}} \dfrac{\phi_{ka}}{Y_a} \end{cases} \quad \forall k, a \in \mathcal{N}.$$

So (48) becomes

$$\frac{\partial Y_k}{\partial \beta_j} = \frac{\partial b_k}{\partial \beta_j} + \sum_{a\in\mathcal{N}(k)} \left(S_{ka} + H_{ka} \frac{\partial Y_a}{\partial \beta_j} \right), \quad \forall k \in \mathcal{N}.$$

This allows us to define the Jacobian of vector Y as a system of linear equations

$$\frac{\partial Y}{\partial \beta_j} = Se + H \frac{\partial Y}{\partial \beta_j} + \frac{\partial b}{\partial \beta_j}$$

or equivalently this is an equivalence, not an implication

$$\Rightarrow \frac{\partial Y}{\partial \beta_j} = (I - H)^{-1} \left(Se + \frac{\partial b}{\partial \beta_j} \right). \tag{49}$$

As suggested by Mai et al. (2015) we can derive the Jacobian of V instead of Y to avoid numerical issues. Note that $Y_k = e^{\mu_k V(k)}$, the gradient of Y_k with respect to β_j, can be written as

$$\frac{\partial Y_k}{\partial \beta_j} = \frac{\partial V(k)}{\partial \beta_j} \mu_k Y_k - \frac{\partial \mu_k}{\partial \beta_j} V(k) Y_k, \quad \forall k \in \mathcal{N}. \tag{50}$$

Using (48) and (50) we obtain

$$\frac{\partial V(k)}{\partial \beta_j} = \sum_{a\in\mathcal{N}} R_{ka} + \sum_{a\in\mathcal{N}} L_{ka}\frac{\partial V(a)}{\partial \beta_j} + h_k, \quad \forall k \in \mathcal{N},$$

where

$$R_{ka} = \frac{1}{\mu_k}\frac{\partial M_{ka}}{\partial \beta_j}\frac{Y_a^{\phi_{ka}}}{Y_k} + \frac{1}{\mu_k}M_{ka}Y_a^{\phi_{ka}}\frac{\partial \phi_{ka}}{Y_k \partial \beta_j}\ln Y_a - \mu_k M_{ka}\frac{V(a)Y_a^{\phi_{ka}}}{Y_k}\frac{\partial \mu_a}{\partial \beta_j},$$

$$L_{ka} = M_{ka}\frac{Y_a^{\phi_{ka}}}{Y_k},$$

$$h_k = \frac{1}{\mu_k Y_k}\frac{\partial b_k}{\partial \beta_j} + \mu_k V(k)\frac{\partial \mu_k}{\partial \beta_j}.$$

Let R, L be two matrices and h be a vector of size $|\mathcal{N}| \times |\mathcal{N}|$, $|\mathcal{N}| \times |\mathcal{N}|$, $|\mathcal{N}|$, with entries R_{ka}, L_{ka} and h_k, $\forall k, a \in \mathcal{N}$, respectively. The Jacobian of the vector of value functions can be written as a linear system

$$\frac{\partial V}{\partial \beta_j} = (I - L)^{-1}(Re + h). \tag{51}$$

Although (51) and (49) are equivalent, there are numerical considerations. On the one hand, each element of matrix L is defined as $L_{ka} = M_{ka}\frac{Y_a^{\phi_{ka}}}{Y_k}$ and according to (20) we have $Y_k > M_{ka}Y_a^{\phi_{ka}} > 0$, leading to the fact that the elements of L vary in (0,1). On the other hand, each element of H is $H_{ka} = M_{ka}Y_a^{\phi_{ka}-1}\phi_{ka}$ and varies in $(0, \infty)$. So the elements of matrix L are closer in value, compared to matrix H, meaning that using (51) to compute the gradient of LL function is better than (49) for numerical reasons. Note that Mai et al. (2015) has a similar conclusion when comparing two formulas of the derivative of the value functions in route choice applications.

We note that the derivative of each element of matrix M with respect to parameter β_j is

$$\frac{\partial M_{ka}}{\partial \beta_j} = M_{ka}\left(\frac{\mu_k \partial \nu(a|k)}{\partial \beta_j} + \nu(a|k)\frac{\partial \mu_k}{\partial \beta_j}\right), \quad \forall k, a \in \mathcal{N}.$$

In summary, the derivatives of the model have complicated form but can be quickly computed for large-scale problems using the linear systems in (47) and (51). The model derivatives allow us to use classic Hessian approximations such as BHHH and BFGS (see for instance Berndt et al., 1974; Nocedal and Wright, 2006) to maximize the log-likelihood function. We note that Eberwein and Ham (2008) also derive the analytical derivatives for a general dynamic discrete choice model. These formulas are however different with the ones derived in this section due to the different definitions of the log-likelihood functions.

Appendix E. Elasticities

The elasticity of demand for alternative i with respect to an attribute x_j of alternative j is

$$e_{i,x_j} = \frac{\partial P(i)}{\partial x_j}\frac{x_j}{P(i)} = \frac{\partial P(i)}{\partial U_j}\frac{\partial U_j}{\partial x_j}\frac{x_j}{P(i)}, i, j \in C.$$

If the utility U_j is linear in x, $\frac{\partial U_j}{\partial x_j}$ is a constant. We now analyze the model structure in terms of the sensitivity of demand to changes in the utility of alternatives $\frac{\partial P(i)}{\partial U_j}$. Similarly to the previous section we derive formulas for the elasticity of demand so that they can be computed quickly.

Note that $\frac{\partial P(i)}{\partial U_j} = \frac{\partial F_i}{\partial U_j}$ and the Jacobian of vector F with respect to U_j can be derived using (22)

$$\frac{\partial F}{\partial U_j} = (I - \mathcal{P}^T)^{-1}\frac{\partial \mathcal{P}^T}{\partial U_j}F, \tag{52}$$

Using (11), the derivative of an element $\mathcal{P}_{ka}$, $k, a \in \mathcal{N}$ with respect to U_j is

$$\frac{\partial \mathcal{P}_{ka}}{\partial U_j} = \mathcal{P}_{ka}\left(\frac{\phi_{ka}}{Y_a}\frac{\partial Y_a}{\partial U_j} - \frac{\partial Y_k}{Y_k \partial U_j}\right),$$

where $\phi_{ka} = \mu_k/\mu_a$. Hence it requires the first derivatives of Y_k, $\forall k \in \mathcal{N}$, with respect to U_j. Taking the derivative of (20) with respect to U_j, we obtain

$$\frac{\partial Y_k}{\partial U_j} = \sum_{a\in\mathcal{N}} \phi_{ka}M_{ka}Y_a^{\phi_{ka}-1}\frac{\partial Y_a}{\partial U_j} + \frac{\partial b_k}{\partial U_j}, \quad \forall k \in \mathcal{N}.$$

And we note that

$$\frac{\partial b_k}{\partial U_j} = \begin{cases} 0 & \text{if } k \neq j \\ Y_j/\mu_j & \text{if } k = j \end{cases}.$$

So if we denote a matrix $T(|\mathcal{N}| \times |\mathcal{N}|)$ with entries $T_{ka} = \phi_{ka} M_{ka} Y_a^{\phi_{ka}-1}$, $\forall k, a \in \mathcal{N}$, then the Jacobian of vector Y can be written as system of linear equations

$$\frac{\partial Y}{\partial U_j} = (I - T)^{-1} d, \tag{53}$$

where d is a vector of size $|\mathcal{N}|$ with zero values for all nodes except for node j that equals Y_j/μ_j. Therefore, the elasticity of demand for alternative i with respect to an attribute x_j can be computed by solving the linear systems (52) and (53).

References

Abbé, E., Bierlaire, M., Toledo, T., 2007. Normalization and correlation of cross-nested logit models. Transp. Res. Part B 41 (7), 795–808.

Barrett, R., Berry, M.W., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., Van der Vorst, H., 1994. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM.

Ben-Akiva, M., Bierlaire, M., 1999. Discrete choice methods and their applications to short-term travel decisions. In: Hall, R. (Ed.), Handbook of Transportation Science. Kluwer, pp. 5–34.

Berndt, E.K., Hall, B.H., Hall, R.E., Hausman, J.A., 1974. Estimation and inference in nonlinear structural models. Ann. Econ. Social Meas. 3/4, 653–665.

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

Daly, A., Hess, S., Dekker, T., 2014. Practical solutions for sampling alternatives in large-scale models. Transp. Res. Rec. (2429) 148–156.

Eberwein, C., Ham, J.C., 2008. Obtaining analytic derivatives for a popular discrete-choice dynamic programming model. Econ. Lett. 101 (3), 168–171.

Fosgerau, M., Frejinger, E., Karlstrom, A., 2013a. A link based network route choice model with unrestricted choice set. Transp. Res. Part B 56 (1), 70–80.

Fosgerau, M., McFadden, M., Bierlaire, M., 2013b. Choice probability generating functions. J. Choice Modell. 8 (0), 1–18.

Gilbert, J.R., Moler, C., Schreiber, R., 1992. Sparse matrices in MATLAB: design and implementation. SIAM J. Matrix Anal. Appl. 13 (1), 333–356.

Guevara, C.A., Ben-Akiva, M.E., 2013. Sampling of alternatives in multivariate extreme value (MEV) models. Transp. Res. Part B 48 (1), 31–52.

Lai, X., Bierlaire, M., 2015. Specification of the cross nested logit model with sampling of alternatives for route choice models. Transp. Res. Part B 80, 220–234.

Liu, D.C., Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. Math. Program. 45 (1–3), 503–528.

Mai, T., 2016. A method of integrating correlation structures for a generalized recursive route choice model. Transp. Res. Part B 93 (1), 146–161.

Mai, T., Fosgerau, M., Frejinger, E., 2015. A nested recursive logit model for route choice analysis. Transp. Res. Part B 75 (1), 100–112.

McFadden, D., 1973. Conditional logit analysis of qualitative choice behaviour. In: Zarembka, P. (Ed.), Frontiers in Econmetrics. Academic Press New York, pp. 105–142.

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

Nocedal, J., Wright, S.J., 2006. Numerical Optimization, 2nd ed. Springer, New York, NY, USA.

Papola, A., 2004. Some developments on the cross-nested logit model. Transp. Res. Part B 38 (9), 833–851.

Rust, J., 1987. Optimal replacement of GMC bus engines: an empirical model of harold zurcher. Econometrica 55 (5), 999–1033.

Rust, J., 1988. Maximum likelihood estimation of discrete control processes. SIAM J. Control Optim. 26 (5), 1006–1024.

Saad, Y., 2003. Iterative Methods for Sparse Linear Systems. SIAM.