# A robust algorithm for the simultaneous estimation of hierarchical logit models

M. Bierlaire

Report 95/3 February 24, 1995

**Abstract.**

Estimating simultaneous hierarchical logit models is conditional to the availability of suitable algorithms. Powerful mathematical programs are necessary to maximize the associated non-linear, non-convex, log-likelihood function. Even if classical methods (e.g. Newton-Raphson) can be adapted for relatively simple cases, the need of an efficient and robust algorithm is justified to enable practioners to consider a wider class of models. The purpose of this paper is to analyze and to adapt to this context methodologies available in the optimization literature. An algorithm is proposed based on two major concepts from non-linear programming : *a trust region method*, that ensures robustness and global convergence, and *a conjugate gradients iteration*, that can be used to solve the quadratic subproblems arising in the estimation process described in this paper. Numerical experiments are finally presented that indicate the power of the proposed algorithm and associate software.

**Transportation Research Group**
**Department of Mathematics,**
**Facultés Universitaires ND de la Paix, Namur, Belgium**
**E-mail : mb@math.fundp.ac.be**

**Keywords :** random utility models, non-linear optimization, hierarchical logit, discrete choice

# 1 Introduction

Modelling and predicting individual choice behaviour has been widely examined by researchers and practitioners these last two decades. The importance of such techniques has been proved in many fields : marketing, behavioural psychology, transportation studies, geography... In this context, random utility theory, and its corresponding models, is very popular (see, namely, Domencich and McFadden, 1975, Ben-Akiva and Lerman, 1985). If several models have been suggested (probit model, hysteresis model,...), a large majority of studies has been carried out using the logit model (see Ortúzar and Willumsen, 1990), namely because it analytically approximates the probit model, and is practically more tractable.

An extension of these models, suggested by Ben-Akiva (1973) and called the *hierarchical* or *nested* logit model, becomes popular in the literature in many fields : activity chains (Mannering, Marakami and Kim, 1994), stated preferences analysis (Bradley and Daly, 1994), housing choice analysis (Vandevyvere, 1994),...

Exploiting this methodology is conditional to the availability of suitable algorithms. If classical methods (Newton-Raphson being the most popular) can be applied to binomial and multinomial logit models, only a few algorithms have been suggested allowing hierarchical models to be estimated simultaneously (Daly, 1987). The difficulty arises mainly from the non-concavity of the associated log-likelihood function, powerful mathematical programs being thus necessary to maximize it.

Little is said in the literature on algorithmic specifications of such programs. Ben-Akiva and Lerman (1985) propose a sequantial estimation procedure; Daly (1987) suggests a modification of the Newton-Raphson procedure using a subsitute second derivatives matrix when non-concavity appears, but no additional detail is provided about algorithmic considerations; Vandevyvere (1994) uses the maximization functionality of the Mathematica package (Wolfram, 1988). Morikawa (1994) claims that *the simultaneous estimation procedure requires special (but not complicated) programming*, but without giving any detail on this procedure. While simple programming may be sufficient as long as the models to be estimated remain relatively simple, we however see the need of an efficient and robust algorithm to enable practioners to consider a wider class of models.

The purpose of this paper is to analyze and to adapt to this context methodologies available in the optimization literature. An algorithm is proposed based on two major concepts in non-linear programming : a *trust region method*, that ensures robustness and global convergence, and *conjugate gradients iterations*, that can be used to solve the optimization quadratic subproblems arising in the estimation process described in this paper.

After the description of the theoretical framework in Section 2, Section 3 will be devoted to the description of the optimization algorithm. The algorithm behaviour in analyzed on practical examples in Section 4 and a brief conclusion is finally presented in Section 5.

# 2 Theoretical framework

Discrete choice theory focus on the behaviour of an individual confronted to a choice within a finite number of alternatives. Each of these is associated with a quantity called *utility*, that mathematically express the individual's preference. This utility varies with the characteristics of the alternatives and of the individual himself. In the particular context of logit models, observable characteristics are assumed to influence linearly the utility, and the unobservable ones, together with other sources of uncertainty, are gathered in a random term, supposed to be Gumbel distributed (see Ben-Akiva and Lerman, 1985 for a complete description of these theoretical aspects).

Appreciated for its simplicity, the multinomial logit model presents some drawbacks, the necessity of the Independance from Irrelevant Alternatives (IIA) assumption being one of the most worrying in a practical context. An interesting method allowing popular logit models to be applied when the IIA assumption fails has been developed by Ben-Akiva (1973) and has been called *hierarchical logit model*.

## 2.1 The hierarchical logit model

In the context of hierarchical model theory (also called *nested* logit model by Ben-Akiva and Lerman, 1985, or *tree* logit model by Daly, 1987) alternatives sharing unobserved characteristics are gathered into subsets, associating a *expected maximum utility* to each of them. Before developing the probabilistic model itself, we define a convenient formal representation.

### 2.1.1 Tree definition

The hierarchical logit model can be easily represented using a directed tree, simply called a tree hereafter. A tree is a directed graph with more than one vertex that has no circuits. It must contains a vertex $\rho(T)$ (called the root of the tree) that has no predecessor, all vertices except $\rho(T)$ having exactly one predecessor.

The vertices of a tree are called here *nodes*. The predecessor of a node $\nu$ is called its *father* and is denoted by $\mathcal{F}(\nu)$. The set $\mathcal{C}(\nu)$ is called the set of *children* of $\nu$ and is defined by

$$\mathcal{C}(\nu) = \{\alpha \in T \mid \mathcal{F}(\alpha) = \nu\}, \tag{1}$$

where $T$ represents the tree.

We will partition the nodes of $T$ into three categories : the root, the elemental nodes defined by

$$E(T) = \{\alpha \in T \mid \mathcal{C}(\alpha) = \emptyset\}, \tag{2}$$

and the structural nodes, defined by

$$S(T) = \{\alpha \in T, \alpha \neq \rho(T) \mid \mathcal{C}(\alpha) \neq \emptyset\}. \tag{3}$$

Obviously,

$$T = \rho(T) \cup E(T) \cup S(T) \tag{4}$$

### 2.1.2 Model characterization

We now characterize the representation of a hierarchical logit model. Assume that $\mathcal{N} = \{N_i; i = 1, \ldots, p\}$ is the set of available alternatives, partitioned into $p$ subsets. A tree $T$ represents a hierarchical logit model if and only if

1. $\rho(T)$ is associated with the set $\mathcal{N}$ of all alternatives,
2. each $\alpha \in E(T)$ is associated with an alternative $n$,
3. each $\beta \in S(T)$ is associated with a subset $N_i$ of alternatives,
4. $\mathcal{F}(\alpha) = \beta$ if and only if the set of alternatives[1] associated with $\alpha$ is included in the set of alternatives associated with $\beta$.

In Figure 1, the set $\mathcal{N}$ of all alternatives, symbolized by the root of the tree, is divided into alternative 1 and subset $N_2$. The latter is then divided into alternative 21 and subset $N_{22}$, and so on.

It is interesting to note that the multinomial logit model can be represented by a tree containing only a root and elemental nodes. From now on, the terms *alternative* and *node* will be interchangeabily used, according to the context.

[1] maybe containing only one elemental alternative.

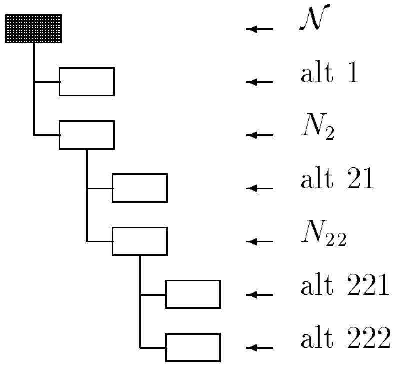


Figure 1: A tree structure

#### 2.1.3 Probabilistic model

We define the probability that individual $x$ chooses alternative $n \in N_i$ as

$$P_x(n) = P_x(N_i)P_x(n|N_i), \tag{5}$$

where $P_x(N_i)$ is the probability that $x$ chooses $N_i$, and $P_x(n|N_i)$ is the probability that $x$ chooses $n$ knowing that he has chosen $N_i$.

We assume below that the utility that individual $x$ associates with alternative $i$ is of the form

$$U_i^x = V_i^x + \varepsilon_i^x = \sum_j \beta_j c_{ij}^x + \varepsilon_i^x \tag{6}$$

where $V_i^x$ is the deterministic part of the utility, $\varepsilon_i^x$ is a Gumbel-distributed random variable, $\beta = (\beta_1, \ldots, \beta_j, \ldots)$ is a set of coefficients to be estimated, and $c_{ij}^x$ is the $j$th characteristic experimented by individual $x$ for alternative $i$.

Dropping subscript $x$ for the sake of clarity, the *expected maximum utility* (Ortúzar and Willumsen, 1990) associated with a subset $N_i$, called by some authors *inclusive value* (McFadden, 1978), or *accessibility* (Ben-Akiva and Lerman, 1979) of $N_i$, can be defined as

$$V_{N_i} = \theta_i \log \sum_{j \in N_i} e^{V_j}. \tag{7}$$

where $\theta_i$ is a positive coefficient to be estimated.

The logit formulation can then be applied for $P(N_i)$ and $P(n|N_i)$. We have

$$P(N_i) = e^{V_{N_i}} / \sum_{j=1}^{p} e^{V_{N_j}} \tag{8}$$

and

$$P(n|N_i) = e^{V_n} / \sum_{j \in N_i} e^{V_j} \tag{9}$$

More sophisticated models can be derived by splitting each $N_i$ into subsets, and applying the same methodology recursively. In the example of Figure 1, if $V_i (i = 1, 21, 221, 222)$ is the deterministic utility associated with elemental node $i$, we can compute the expected maximum utilities of the subsets associated with structural nodes :

$$\begin{aligned} V_{N_{22}} &= \theta_2 \log(e^{V_{221}} + e^{V_{222}}), \\ V_{N_2} &= \theta_1 \log(e^{V_{21}} + e^{V_{N_{22}}}). \end{aligned} \tag{10}$$

The probability of alternative 222, for example, can then be derived as followed :

$$P(222) = P(222|N_{22})P(N_{22}|N_2)P(N_2), \tag{11}$$

where

$$\begin{aligned} P(N_2) &= e^{V_{N_2}}/(e^{V_1} + e^{V_{N_2}}) \\ P(N_{22}|N_2) &= e^{V_{N_{22}}}/(e^{V_{21}} + e^{V_{N_{22}}}) \\ P(222|N_{22}) &= e^{V_{222}}/(e^{V_{221}} + e^{V_{222}}). \end{aligned} \tag{12}$$

## 2.2 The maximum likelihood method

Assuming we have

1. a tree structure corresponding to the characterization of Section 2.1.2,
2. a utility function, as defined by (6), associated with each elemental node,
3. a probabilistic logit model as defined in Section 2.1.3,
4. a sample of individuals, each of them associated with
   - a list of characteristics describing the individual himself,
   - a list of characteristics describing each alternative available for the individual,
   - the alternative that was actually chosen,

the name of the game is to estimate the set $\beta$ of coefficients of the utility functions (6) and the set $\theta$ of (so called) structural coefficients defined in the expected maximum utilities (7), that enable the model to reproduce the actual behaviour of individuals as well as possible.

If $n_x$ is the alternative actually chosen by individual $x$ in the sample, we call $P_x(\beta, \theta)$ the probability given by the model that $n_x$ is chosen. As all characteristics are supposed to be defined in the data set, $P_x(\beta, \theta)$ depends only on the unknown sets of coefficients $\beta$ and $\theta$.

For each set of $(\beta, \theta)$, we can compute the joint probability for the whole sample :

$$\mathcal{L}(\beta, \theta) = \prod_{x=1}^{X} P_x(\beta, \theta). \tag{13}$$

where $X$ is the sample size. This function is called the *likelihood funtion.*

The maximum likelihood method (see e.g. Stuart and Ord, 1991) consists in finding the set of coefficients that maximizes (13), that is in solving

$$\max_{\beta,\theta} \mathcal{L}(\beta,\theta) = \max_{\beta,\theta} \prod_{x=1}^{X} P_x(\beta,\theta). \tag{14}$$

If we define

$$\mathcal{L}'(\beta,\theta) = \log \mathcal{L}(\beta,\theta), \tag{15}$$

we have the equivalent mathematical program

$$\max_{\beta,\theta} \mathcal{L}'(\beta,\theta) = \max_{\beta,\theta} \sum_{x=1}^{X} \log P_x(\beta,\theta), \tag{16}$$

where $\mathcal{L}'(\beta,\theta)$ is known as the *log-likelihood function.*

This function is obviously non-linear. It can be shown (Ben-Akiva and Lerman, 1985) that, in the case of multinomial logit, $\mathcal{L}'(\beta,\theta) = \mathcal{L}'(\beta)$ is concave, and therefore rather easy to maximize. We can show that this nice property disappears with the introduction of the hierarchical model, a robust algorithm exploiting the non-concavity being thus necessary.

We have chosen the tree structure presented in Figure 1, where $V_1 = -10, V_{N_2} = \theta_1 \log(e^{V_{21}} + e^{V_{N_{22}}}), V_{21} = 10, V_{N_{22}} = \theta_2 \log(e^{V_{221}} + e^{V_{222}}), V_{221} = -10, V_{222} = \beta$. A sample of four individuals is considered, each of them being supposed to choose a different elemental alternative. The log-likelihood can then be written as

$$\mathcal{L}'(\beta,\theta_1,\theta_2) = \log(P_1) + \log(P_{21}) + \log(P_{221}) + \log(P_{222}) \tag{17}$$

where, using the hierarchical logit formulation,

$$\begin{array}{rcl}
P_1 &=& e^{V_1}/(e^{V_1} + e^{V_{N_2}}) \\
P_{N_2} &=& e^{V_{N_2}}/(e^{V_1} + e^{V_{N_2}}) \\
P_{21} &=& P_{N_2}\, e^{V_{21}}/(e^{V_{21}} + e^{V_{N_{22}}}) \\
P_{N_{22}} &=& P_{N_2}\, e^{V_{N_{22}}}/(e^{V_{21}} + e^{V_{N_{22}}}) \\
P_{221} &=& P_{N_{22}}\, e^{V_{221}}/(e^{V_{221}} + e^{V_{222}}) \\
P_{222} &=& P_{N_{22}}\, e^{V_{222}}/(e^{V_{221}} + e^{V_{222}})
\end{array}$$

Constraining $\theta_1 = 0.5$ and $\theta_2 = \beta$, we have

$$\mathcal{L}'(\beta) = \beta - 10 + 2\ (\beta - 1)\ \log(\phi(\beta)) - \frac{3}{2}\ \log(\tau(\beta)) - 4\ \log(e^{-10} + \sqrt{\tau(\beta)}), \tag{18}$$

where $\phi(\beta) = e^{-10} + e^{\beta}$ and $\tau(\beta) = e^{10} + \phi(\beta)^{\beta}$. Plotted on Figure 2, this particular part of the log-likelihood function is obviously non-concave when $\beta$ (and thus $\theta_2$) are between -1 and 1.

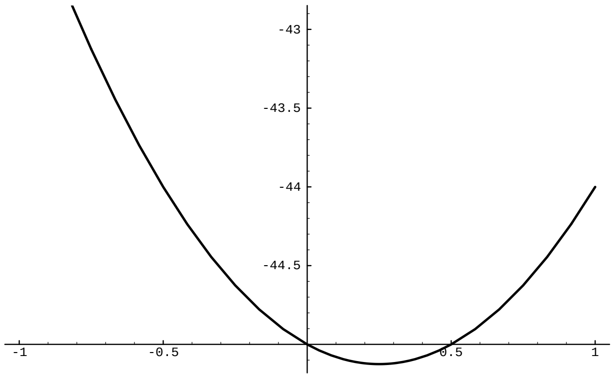


Figure 2: Plot of $\mathcal{L}'(\beta)$

# 3 Optimization algorithm

Even if the theoretical framework presented above has been known since 1973 (see Ben-Akiva, 1973 or, later, Williams, 1977), its practical use appeared only some years later (e.g. Sobel, 1980). The lack of a dedicated software, due to the complexity of the estimation process, has encouraged practitioners to restrict themselves to multinomial logit models, or to estimate nested models sequentially (Ben-Akiva and Lerman, 1985).

The first (as far as the author knows) software for estimating simultaneously all the coefficients of a hierarchical model with a maximum likelihood process has been proposed by Daly (1987). Using a so-called *hierarchical function*, Daly has been able to compute the log-likelihood function and its derivatives at relatively low additional cost. The optimization method is a modified Newton-Raphson procedure, replacing the true second derivatives matrix by a negative definite approximation when needed. Unfortunately, no additional detail is provided about algorithmic considerations.

If the Newton-Raphson works well with multinomial and relatively simple hierarchical models, a more sophisticated modification is needed in the non-concave context of hierarchical models, insuring robustness and efficiency. We now describe such an algorithm inspired from robust methods in non-linear programming and dedicated to the estimation of hierarchical logit models. The reader is namely referred to Powell (1970), Hestenes (1980), Toint (1981), Gill, Murray and Wright (1981), Steihaug (1983), Dennis and Schnabel (1983), Fletcher (1987), Conn, Gould and Toint (1991), Conn, Gould and Toint (1992a), Conn, Gould and Toint (1992b),... for further detail.

## 3.1 Minimization problem

In order to maintain consistency with the non-linear programming literature on which our algorithm is based (namely Toint, 1981 and Steihaug, 1983), we consider from here on the following **minimization** problem :

$$\min_{x \in \mathbb{R}^n} f(x). \tag{19}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is twice continuously differentiable and $x$ is the vector of $n$ unknowns. Obviously, (19) is equivalent with

$$\max_{(\beta,\theta) \in \mathbb{R}^n} \mathcal{L}'(\beta, \theta) \tag{20}$$

where $f = -\mathcal{L}'$ is the opposite of the log-likelihood function (15) and $x$ is the vector of $n$ unknowns composed of the $\beta$'s and the $\theta$'s. The vector of first derivatives, or gradient, evaluated at $x$ is denoted by $g(x)$, when $H(x)$ represents the second derivatives matrix, or Hessian matrix, evaluated at the same point.

## 3.2 Framework of the algorithm

The algorithm we present here belongs to the class of iterative methods. Such methods are typically divided into three parts : initialization, iterations and stopping criteria. These steps are now described in our particular context.

**Initialization** : This phase defines a set of starting values for $x$. It is commonly called *iteration 0* and, therefore, the starting point is denoted by $x^0$. Even if, theoretically, any value can be used as a starting point, when no relevant value is available, it is advised to start with all $\beta$'s set to zero, and all $\theta$'s set to one. Indeed, these values correspond to the most naïve model where all alternatives have the same probability. A bad choice of values could lead to a starting model worst than this simple one.

**Iterations** : Given a set of values $x^k$, this phase finds another set $x^{k+1}$ that decreases the value of the objective function. We will elaborate on this step in the next section.

**Stopping criteria** : Iterations are performed until one of the following conditions is satisfied, where constants *itermax*, $\varepsilon_g$ and $\varepsilon_s$ are supposed to be given.

- The maximum number of iterations *itermax* is reached.
- The relative gradient magnitude is sufficiently close to 0, that is

$$\max_{1 \leq i \leq n} \left| \frac{\max(|x_i^k|, 1) \cdot g_i(x^k)}{\max(|f(x^k)|, 1)} \right| \leq \varepsilon_g \tag{21}$$

  where $n$ is the number of coefficients, $x_i^k$ is the $i$th coefficient at iteration $k$, $f(x^k)$ is the objective function evaluated at $x^k$, and $g_i(x^k)$ the $i$th first derivative of this function.

- The step between two successive iterates becomes too small, that is

$$\max_{1\le i\le n} \frac{|x_i^{k+1} - x_i^k|}{\max(|x_i^{k+1}|, 1)} \le \varepsilon_s. \tag{22}$$

Default values for $\varepsilon_g$ and $\varepsilon_s$ are drawn from Dennis and Schnabel (1983). If $\varepsilon_m$ is the smallest significant value with respect to 1 (called the *machine epsilon*), for a particular processor, it is suggested that $\varepsilon_g = (\varepsilon_m)^{\frac{1}{3}}$ and $\varepsilon_s = (\varepsilon_m)^{\frac{2}{3}}$.

## 3.3 Iterations

We now describe how iterate $x^{k+1}$ is obtained from the current iterate $x^k$, the gradient of $x^k$ ($g(x^k)$) and the Hessian $H(x^k)$.

**Step 1** Compute $x_{\text{min}}$ by (approximately) minimizing the quadratic model

$$m(x) = g(x^k)^T x + \frac{1}{2} x^T H(x^k) x \tag{23}$$

subject to

$$\|x\|_p \le \delta_k, \tag{24}$$

where $\|\cdot\|_p$ is any Hölder norm, $\delta_k$ is the radius of the trust region (see Section 3.4.1) at iteration $k$ and $T$ denotes the transpose operator. Solving (23)-(24) is performed using a modified conjugate gradients algorithm (CG), developed in Section 3.4.2. A candidate for the next iterate is then computed :

$$\tilde{x}^k = x^k + x_{\text{min}}. \tag{25}$$

**Step 2** Decide if the candidate $\tilde{x}^k$ defined by (25) is satisfactory, and set $x^{k+1}$ and $\delta_{k+1}$ accordingly. The quality of $\tilde{x}^k$ is measured by the ratio $\rho_k$ between the predicted and actual decreases in the model and in the objective function, respectively, obtained at $\tilde{x}^k$ :

$$\rho_k = \frac{f(x^k) - f(\tilde{x}^k)}{m(x^k) - m(\tilde{x}^k)}. \tag{26}$$

Given any suitably chosen $\varepsilon_1$, $\varepsilon_2$, $\alpha$, $\gamma$ such that $0 < \varepsilon_1 < \varepsilon_2 < 1$ and $0 < \alpha < 1 < \gamma$, we have the following possibilities.

- If $\rho_k \le \varepsilon_1$, than $\tilde{x}^k$ is not satisfactory. The candidate is then rejected and the trust region reduced.

$$\delta_{k+1} = \alpha\delta_k, x^{k+1} = x^k. \tag{27}$$

- If $\varepsilon_1 < \rho_k < \varepsilon_2$, than $\tilde{x}^k$ is satisfactory. The candidate is accepted and the trust region remains unchanged.

$$\delta_{k+1} = \delta_k, x^{k+1} = \tilde{x}^k. \tag{28}$$

- If $\rho_k \geq \varepsilon_2$, than $\tilde{x}^k$ is very satisfactory. The candidate is accepted and the trust region increased.

$$\delta_{k+1} = \gamma\delta_k, x^{k+1} = \tilde{x}^k. \tag{29}$$

Next section will elaborate on particular features of this algorithms.

## 3.4 Algorithmic features

We describe here three algorithmic features adapting the general method presented in Sections 3.2 and 3.3 to the special case of problem (16). After some comments on the general philosophy of truist region methods, the CG method is modified to handle non-positive definite quadratic models. A preconditioning method is then described, in order to improve the practical behaviour of the CG method. Due to the high computational cost of evaluating the Hessian matrix of (16), a method is then presented that avoids this computation.

### 3.4.1 Trust region methods

First introduced by Powell (1970), the trust region method is one of the most popular modifications of Newton method that ensure global convergence (together with the line searches technics). The idea is to define a region around the current iterate where the quadratic model (23) is supposed to fit properly the objective function. The constrained minimization problem (23)-(24) has to be solved at every iteration, at least approximately. The trust region size must be suitably updated so as to (1) produce relatively small steps when far from the solution, ensuring robustness, and (2) keep local quadratic convergence property of Newton method. The proof of the global convergence can be found in many references (see Theorem 5.1.1 in Fletcher (1987), for instance).

### 3.4.2 Conjugate gradients

The conjugate gradients algorithm, which solves the unconstrained problem (23) when the second derivatives matrix $H(x^k)$ is positive definite, can be described as follow. Subscript $l$ is used to denote the conjugate gradients iteration number, $y^l$ represents the current iterate and $d^l$ the search direction.

**Initialization** $y_i^0 = 0, d_i^0 = -g_i(x^k)$, for $i = 1, \ldots, n$

**Loop** While

$$\|d^l\|_p >= \varepsilon_{\mathrm{CG}}\|d^0\|_p \tag{30}$$

**Minimize along $d^l$** : find $s^l$ solution of the univariate minimization problem

$$\min_s m(y^l + sd^l) \tag{31}$$

**Update** $y^{l+1} = y^l + s^l d^l$, and choose a direction $d^{l+1}$ such that

$$d^{l+1} H(x^k) d^i = 0 \quad i = 0, \ldots, l. \tag{32}$$

Such a vector $d^{l+1}$ is said to be *conjugate* to $d^0, \ldots, d^l$ with respect to the matrix $H(x^k)$.

**End loop** Terminates the iteration by updating index $l$.

Note that (31) is easy to compute because $m$ is quadratic. Theoretically, this algorithm converges in $n$ iterations. Unfortunately, experiments show that the number of iterations is usually higher, especially when the conditioning of the matrix $H(x^k)$ is ill. This problem is treated in Section 3.4.3.

In order to take possible non-convexity and constraint (24) into account, Toint (1981) and Steihaug (1983) suggest to run the previous algorithm until one of the three following conditions is verified.

1. The unconstrained solution is reached, that is

$$\|d^l\|_p < \varepsilon_{\mathrm{CG}}\|d^0\|_p. \tag{33}$$

   Candidate $x_{\min}$ is then defined as $y^l$.

2. The next iterate is out of the trust region, that is

$$\|y^{l+1}\|_p > \delta_k. \tag{34}$$

   In this case, a step $\alpha$ is then computed such that

$$\|y^l + \alpha d^l\|_p = \delta_k, \tag{35}$$

   and $x_{\min}$ is defined as $y^l + \alpha d^l$.

3. $d^l$ is a direction of negative curvature, that is

$$\langle d^l, H(x_k) d^l \rangle \leq 0 \tag{36}$$

   where $\langle .,. \rangle$ denotes the standard inner product on $\mathbb{R}^n$. A step $\alpha$ verifying (35) is then computed, and $x_{\min}$ is defined as $y^l + \alpha d^l$.

We stress out here that (36) not only exploit explicitly non-convexity, but also insures the robustness of the algorithm. Indeed, even if the Hessian matrix is not invertible (i.e. there exist a $d^l$ such that $\langle d^l, H(x_k)d^l\rangle \simeq 0$), the algorithm just goes on. This is illustrated in Section 4.3.

We finally emphasize on stopping condition (30), by noting that only a *relative* reduction of the initial direction magnitude is required. This practice, directly derived from large scale optimization (e.g. Toint, 1981 and Steihaug, 1983), obviates additional computation when the algorithm is far from convergence, and improve the precision close to the solution.

### 3.4.3 Preconditioner

The conjugate gradients method theoretically requires $n$ iterations to solve a $n \times n$ positive-definite system of linear equations. Unfortunately, it is not the case in practice. One of the main causes of the method slackening is an ill-conditioning of the Hessian matrix.

The condition number of a matrix $H$ is defined by Golub and Van Loan (1983) as

$$\kappa(H) = \|H\|_p \|H^{-1}\|_p, \tag{37}$$

where $\|\cdot\|_p$ can be any matrix $p$-norm. If $\kappa(H)$ is large, the matrix is ill-conditioned. This situation corresponds to a high sensitivity of the system solution to small perturbations in the data.

In order to improve the method speed, we solve an *equivalent* system of equations with, we hope, a better-conditioned matrix. The method is based on the following observations. In the ideal case where the quadratic model is convex (that is $H(x^k)$ positive definite), the solution of the unconstrained problem (23) is also the solution of the linear system

$$H(x^k)x = -g(x^k). \tag{38}$$

As $H(x^k)$ is symmetric, the system (38) can be solved using a Cholesky factorization $LL^T$ of the Hessian matrix. We refer the reader to Golub and Van Loan (1983) for a description of the Cholesky factorization. In such a context, this *direct* method gives the solution and the conjugate gradients iteration is useless.

As it was already stated, non-convexity can arise. The idea is therefore to use the Cholesky factorization adapted to non-positive definite matrices proposed by Schnabel and Eskow (1991) as a preconditioner for the conjugate gradients iteration. This method computes a Cholesky factorization $LL^T$ of $H(x^k) + D$, where $D$ is 0 if $H(x^k)$ is positive definite, and $D$ is a non-negative diagonal matrix such that $H(x^k) + D$ is safely positive definite otherwise. The equivalent system mentioned above is then

$$L^{-1}H(x^k)L^{-1}y = -L^{-1}g(x^k), \tag{39}$$

with $y = Lx$, and usually has an improved condition number.

### 3.4.4 Approximation of the Hessian matrix

Complex models with large data sets result in a high cost for the computation of the second derivatives matrix, or Hessian matrix $H(x^k)$. To avoid spending too much time in such computations, it is usually cheaper to approximate this matrix using first order information.

The class of method using this philosophy is commonly called *Quasi-Newton methods*. Among most popular technics, differing by the matrix update strategy, we can cite Powell-symmetric-Broyden (PSB), Broyden-Fletcher-Goldfarb-Shanno (BFGS), Davidon-Fletcher-Powell (DFP) or symmetric-rank-one (SR1). We refer the reader to Chapter 9 of Dennis and Schnabel (1983) for a detailed development. Symmetric rank-one is prefered here, due to the fact that this strategy does not necessarily produce a positive definite approximation (preference for symmetric rank-one is debated in a theoretical context by Conn, Gould and Toint, 1991 and practically by Conn, Gould and Toint, 1992b).

If we denote

$$s^k = x^{k+1} - x^k \tag{40}$$

and

$$y^k = g(x^{k+1}) - g(x^k), \tag{41}$$

the SR1 formula is given by

$$\tilde{H}^{k+1} = \tilde{H}^k + \frac{(y^k - \tilde{H}^k s^k)(y^k - \tilde{H}^k s^k)^T}{(y^k - \tilde{H}^k s^k)^T s^k}. \tag{42}$$

where $\tilde{H}^k$ is the approximation of the Hessian matrix at iteration $k$.

Experiments have shown that (see Section 4), for large models, even if the number of iterations increases, the amount of time saved is worth using the approximated matrix.

# 4 Numerical experiments

We now illustrate the main features of the algorithm through numerical experiments. Real case studies were preferred to artificial data so as to avoid possible bias due to the latter.

The first model (called "parking model" in the following) is derived from a stated preferences survey on parking choice in Namur (Belgium) that has been conducted by a student in the context of her masters thesis on congestion and parking policy (Deputter, 1994). A total of 160 individuals working in Namur city center were interviewed. Selecting only error free information resulted in a total of 1386 observations. The aim of the study was to analyze the impact of parking cost, search time, walk time, travel time, individual's characteristics (sex, social level, profession, incomes,...) on parking choice. Details on on-street/off-street parking selection has been asked only when legal parking in the city center was selected. The algorithm was used to estimate a model with 14 elemental and 1 structural coefficients, with the tree structure represented if Figure 3. The alternatives are : legal parking in the centre, illegal parking in the centre, use of public transportation and parking out of the city centre. If legal parking is chosen, it can be either on-street or off-street.

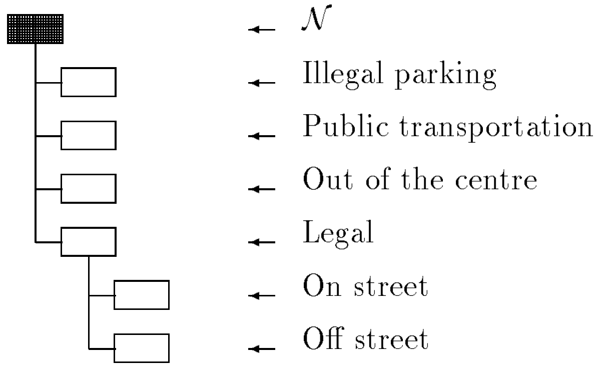


Figure 3: Tree structure of the parking model

The second model (called "shopping model" in the following) is derived from a stated preferences survey conducted by the Urban Planning Group of the University of Eindhoven (Arentze, Borgers, Dellaert, Timmermans and Bierlaire, 1995). The aim was to analyze the behaviour of individuals performing multi-purpose multi-stop shopping trips. Three shopping facilities (supermarket, drug store and clothing shop) were proposed in three different locations, and individuals were asked to describe the trip chain they would perform. The algorithm was used to estimate a model with 12 elemental and 3 structural coefficients, with the complex tree structure containing 33 elemental nodes and 10 structural ones, represented in Figure 4. A total of 1179 observations has been used.

## 4.1 Results

The algorithm has been applied both to the parking and the shopping models. We illustrate the impact of the particular algorithmic features discussed above by using different variants. The algorithm has been tested with and without the preconditioner described in Section 3.4.3, and both with the exact hessian matrix and a Symmetric Rank One approximation (see Section 3.4.4), yielding to four variants.

All tests have been computed on a 486DX (33 Mhz) PC computer using the HieLoW package (see for example Bierlaire, 1994 and Bierlaire and Vandevyvere, 1995) running in the Windows environment. Main results are gathered in Table 1 for the shopping model and Table 2 for the parking model. For each variant of the algorithm, the following information is reported : the total number of iterations (Iter.), the total time (Time), the average

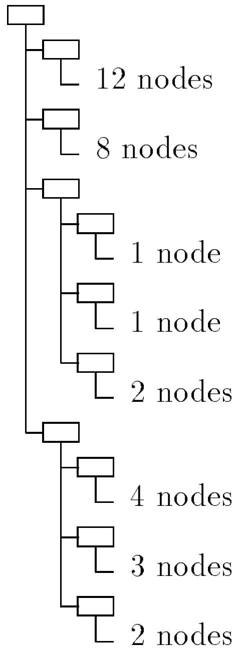


Figure 4: Tree structure of the shopping model

time per iteration (Time/Iter.), the total number of conjugate gradients iterations (CG), the number of iterations where the conjugate gradients stopped because the boundary of the trust region was reached (condition (34), coded TRUST in the table), because the convergence criterion has been reached (condition (33), coded CVGCE), because a non-strictly positive curvature direction has been found (condition (36), coded -CURV).

We now analyze different features of the described algorithm through the numerical results.

## 4.2 Trust region

The trust region acts as a handrail, a barricade, avoiding the next iterate falling too far away when the information used to compute it is unreliable. The "TRUST" line in Table 1 and 2 is explicit on the role of the trust region, but we prefer analyzing it on a particular example. Considering the algorithm with preconditioner and exact hessian applied on the parking model, this role is illustrated on Figure 5, where the magnitude of the step and the trust region size are reported on a logarithmic scale for each iteration. The trust region enables larger and larger steps (e.g. iterations 7 to 18) while the quadratic model provides good information, but restricts them otherwise (e.g. iterations 19 and 20). Note also that, when convergence occurs, the trust region becomes larger than the step length. The constrained problem (23)-(24) is then equivalent to the associated unconstrained one,

| | With prec. Rank One | Without prec. Rank One | With prec. Exact Hessian | Without prec. Exact Hessian |
|---|---|---|---|---|
| Iter. | 107 | 85 | 26 | 37 |
| Time | 1h 13m 31s | 0h 58m 29s | 2h 13m 49s | 3h 05m 15s |
| Time/Iter. | 41s | 41s | 5m 09s | 5m 00s |
| CG | 176 | 476 | 46 | 148 |
| TRUST | 64 (59.8%) | 56 (65.9%) | 20 (76.9%) | 32 (86.5%) |
| CVGCE | 22 (20.6%) | 14 (16.5%) | 4 (15.4%) | 4 (10.8%) |
| -CURV | 21 (19.6%) | 15 (17.6%) | 2 (7.7%) | 1 (2.7%) |

Table 1: Results for the shopping model

| | With prec. Rank One | Without prec. Rank One | With prec. Exact Hessian | Without prec. Exact Hessian |
|---|---|---|---|---|
| Iter. | 99 | 104 | 32 | 65 |
| Time | 24m 02s | 25m 14s | 42m 41s | 1h 26m 05s |
| Time/Iter. | 15s | 15s | 1m 20s | 1m 19s |
| CG | 251 | 783 | 135 | 1995 |
| TRUST | 60 (60.6%) | 65 (62.5%) | 24 (75.0%) | 57 (87.6%) |
| CVGCE | 19 (19.2%) | 20 (19.2%) | 5 (15.6%) | 4 (6.2%) |
| -CURV | 20 (20.2%) | 19 (18.3%) | 3 (9.4%) | 4 (6.2%) |

Table 2: Results for the parking model

ensuring local quadratic convergence of Newton's method.

### 4.3 Negative curvature

The stopping condition (36) for CG iterations, detecting non-convexity in the quadratic model, appears to play a twofold role in the behaviour of the algorithm. Indeed, the candidate produced in this case by the CG iterations can be either accepted, and non-convexity is then taken explicitly into account to determine the next iterate, or rejected. This rejection, followed by a reduction of the trust region size, insures the robustness of the algorithm when the model doesn't fit correctly the objective function or when singularity in the quadratic model is detected.

If the number of rejections seems larger, especially when approximated Hessian is used (see column NO of Table 3, the number of instances where non-convexity is exploited is

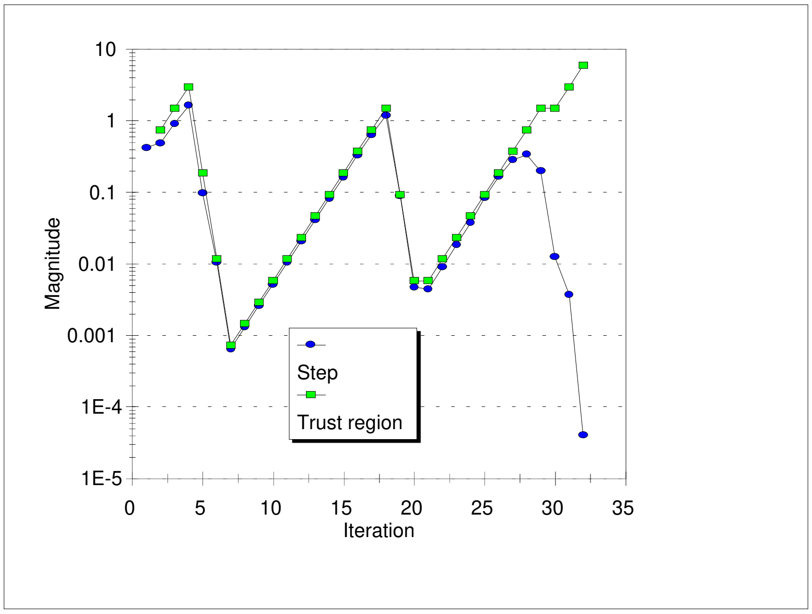


Figure 5: Step magnitude and trust region size

not negligible (column OK of Table 3). Note that column -CURV of Table 3 gathers the same information as the last line in Table 1 and 2.

## 4.4 Preconditioner

The preconditioner is aimed at simplifying the minimization of (23)-(24). This is well illustrated by the important decrease of the total number of conjugate gradients iterations in the presence of the preconditioner (see Tables 1 and 2). Unfortunately, an improvement of the global algorithm is not always guaranteed, as illustrated by the shopping model with Rank One update, where 22 additional iterations are needed with the preconditioner, although a 63% reduction of conjugate gradients iterations is observed. In all other cases, the algorithm efficiency is significatively improved by the use of a preconditioner.

## 4.5 Approximate second derivatives matrix

It is first noticed that using an approximation of the Hessian matrix implies an important increase in the number of iterations (from 60 up to 311% in our examples). However, computational time per iteration is reduced by 86% (shopping model) and 81.5% (parking model). Globally, a substantial gain is thus obtained by using the approximated matrix. Figure 6 illustrates cumulative running time of both strategies. Even if 70 additional iterations would be required, the algorithm using Rank One update would still be competitive from an execution time point of view.

| Model | Hessian | Precond. | -CURV | OK | NO |
|---|---|---|---|---|---|
| Shopping | Exact | Yes | 2 | 1 | 1 |
| | | No | 1 | 0 | 1 |
| | Approx. | Yes | 21 | 3 | 18 |
| | | No | 15 | 2 | 13 |
| Parking | Exact | Yes | 3 | 2 | 1 |
| | | No | 4 | 2 | 2 |
| | Approx. | Yes | 20 | 10 | 10 |
| | | No | 19 | 9 | 10 |

Table 3: Number of accepted and rejected candidates in the presence of negative curvature

# 5 Conclusion

We have exploited powerful and up-to-date technics in non-linear programming and adapted them to the problem of simultaneous estimation of hierarchical logit models. An optimization algorithm that is based on a quasi-Newton technic combined with trust region strategy, ensuring robustness and global convergence, and a modified preconditioned conjugate gradients iterations, exploiting non-concavity.

The method has been applied to two numerical examples derived from real exercices, illustrating its major features, and clarifying the effect of some algorithmic options. The algorithm described in this paper has been included in the HieLoW package (Bierlaire, 1994 and Bierlaire and Vandevyvere, 1995).

# 6 Acknowledgement

The author is indebted to Ph. Toint and A. Sartenaer for their advices and support about optimization theory and algorithmic features. Data used for tests has been made available thanks to B. Dellaert, C. Deputter, S. Gayda (Stratec) and Y. VandeVyvere. The financial support granted to the author by the AGIR program of the Région Wallonne is finally gratefully acknowledged.

# References

Arentze, T., Borgers, A., Dellaert, B., Timmermans, H. and Bierlaire, M. (1995). A multi-purpose multi-stop logit model of consumer shopping centre choice, *Technical report*, Urban Planning Group, Eindhoven University of Technology, The Netherlands.

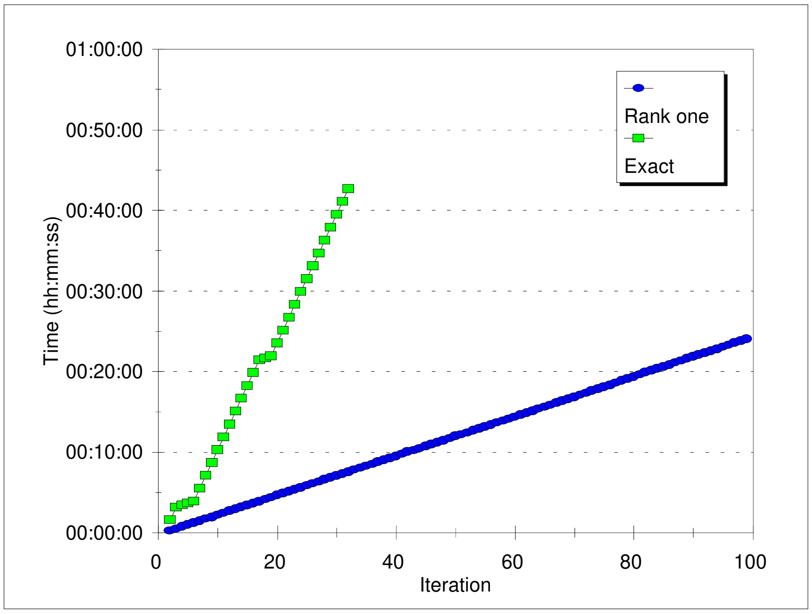


Figure 6: Comparison of computational time between exact and approximated Hessian


Ben-Akiva, M. and Lerman, S. R. (1979). Disaggregate travel demand and mobility choice models and measures of accessibility, *in* D. Hensher and P. Stopher (eds), *Behavioral Travel Modeling*, Croom Helm, London.

Ben-Akiva, M. E. (1973). *Structure of passenger travel demand models*, PhD thesis, Department of Civil Engineering, MIT, Cambridge, Ma.

Ben-Akiva, M. E. and Lerman, S. R. (1985). *Discrete Choice Analysis: Theory and Application to Travel Demand*, MIT Press, Cambridge, USA.

Bierlaire, M. (1994). HieLoW : un logiciel d'estimation de modèles logit emboîtés, *Les cahiers du MET* **2**: 29–43.

Bierlaire, M. and Vandevyvere, Y. (1995). *HieLoW: the interactive user's guide*, Transportation Research Group - FUNDP, Namur.

Bradley, M. and Daly, A. (1994). Use of the logit scaling approach to test for rank-order and fatigue effects in stated preferences data, *Transportation* **21**: 167–184.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1991). Convergence of quasi-Newton matrices generated by the symmetric rank one update, *Mathematical Programming* **50**(2): 177–196.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1992a). LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*, number 17 in *Springer Series in Computational Mathematics*, Springer Verlag, Heidelberg, Berlin, New York.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1992b). Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization, *Technical Report 92/16*, Department of Mathematics, FUNDP, Namur, Belgium.

Daly, A. (1987). Estimating "tree" logit models, *Transportation Research B* **21**(4): 251–268.

Dennis, J. E. and Schnabel, R. B. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall, Englewood Cliffs, USA.

Deputter, C. (1994). *La congestion et la politique de parking à namur*, Master's thesis, FUNDP - Facultés des Sciences Economiques et Sociales.

Domencich, T. and McFadden, D. (1975). *Urban Travel Demand: A Behavioural Analysis*, North-Holland, Amsterdam.

Fletcher, R. (1987). *Practical Methods of Optimization*, second edn, J. Wiley and Sons, Chichester.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press, London and New York.

Golub, G. H. and Van Loan, C. F. (1983). *Matrix computations*, North Oxford Academic, Oxford, UK.

Hestenes, M. R. (1980). *Conjugate Direction Methods in Optimization*, Springer Verlag, New York.

Mannering, F., Marakami, E. and Kim, S.-G. (1994). Temporal stability of traveler's activity choice and home-stay duration: some empirical evidence, *Transportation* **21**(4): 371–392.

McFadden, D. (1978). Modelling the choice of residential location, *in* A. K. et al. (ed.), *Spatial interaction theory and residential location*, North-Holland, Amsterdam, pp. 75–96.

Morikawa, T. (1994). Correcting state dependence and serial correlation in the RP/SP combined estimation method, *Transportation* **21**(2): 153–165.

Ortúzar, J. D. and Willumsen, L. (1990). *Modelling Transport*, J. Wiley and Sons, Chichester (England).

Powell, M. J. D. (1970). A new algorithm for unconstrained optimization, *in* J. B. Rosen, O. L. Mangasarian and K. Ritter (eds), *Nonlinear Programming*, Academic Press, New York.

Schnabel, R. B. and Eskow, E. (1991). A new modified Cholesky factorization, *SIAM Journal on Scientific and Statistical Computing* **11**: 1136–1158.

Sobel, K. L. (1980). Travel demand forecasting with the nested multinomial logit model, *59th Annual meeting of the Transpn. Res. Board*, Washington D.C.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization, *SIAM Journal on Numerical Analysis* **20**(3): 626–637.

Stuart, A. and Ord, J. K. (1991). *Classical Inference and relationship*, Vol. 2 of *Kendall's Advanced Theory of Statistics*, fifth edn, Edward Arnold, London-Melbourne-Auckland.

Toint, P. L. (1981). Towards an efficient sparsity exploiting Newton method for minimization, *in* I. S. Duff (ed.), *Sparse Matrices and Their Uses*, Academic Press, London, pp. 57–88.

Vandevyvere, Y. (1994). *Complex spatial decision-making and stated preferences decompositional modeling: residential choice in Louvain-la-Neuve, Belgium*, PhD thesis, Université Catholique de Louvain.

Williams, H. (1977). On the formation of travel demand models and economic measures of user benefit, *Environment and Planning* **9A**: 285–344.

Wolfram, S. (1988). *Mathematica, a system for doing mathematics by computer*, Addison-Wesley Publishing Company.