A Gobal Search Procedure for Parameter Estimation in Neural Spatial Interaction Modelling

Fischer, Manfred M.; Hlavácková-Schindler, Katarina; Reismann, Martin

DOI:
10.57938/dfea3476-a8b0-4988-9857-640cc828120e

Published: 01/09/1998

Document Version
Publisher's PDF, also known as Version of record

WSG Diskussionspapiere

WSG 63/98

A Gobal Search Procedure for Parameter Estimation in Neural Spatial Interaction Modelling

Manfred M. Fischer, Katarina Hlavácková-Schindler and Martin Reismann

Institut für Wirtschafts- und Sozialgeographie

Wirtschaftsuniversität Wien

Department of Economic and Social Geography

Vienna University of Economics and Business Administration

WSG Discussion Papers

Abteilung für Theoretische und Angewandte Wirtschafts- und Sozialgeographie
Institut für Wirtschafts- und Sozialgeographie
Wirtschaftsuniversität Wien

Vorstand: o.Univ.-Prof. Dr. Manfred M. Fischer
A - 1090 Wien, Augasse 2-6, Tel. ++43-(0)1-31336-4836

Redaktion: Univ.-Ass. Dr. Petra Staufer

WSG 63/98

A Gobal Search Procedure for Parameter Estimation in Neural Spatial Interaction Modelling

Manfred M. Fischer, Katarina Hlavácková-Schindler and Martin Reismann

WSG-Discussion Paper 63

September 1998

Gedruckt mit Unterstützung
des Bundesministerium
für Wissenschaft und Verkehr
in Wien

WSG Discussion Papers are interim
reports presenting work in progress
and papers which have been submitted
for publication elsewhere.

ISBN 3 85037 075 5

Abstract

Parameter estimation is one of the central issues in neural spatial interaction modelling. Current practice is dominated by gradient based local minimization techniques. They find local minima efficiently and work best in unimodal minimization problems, but can get trapped in multimodal problems. Global search procedures provide an alternative optimization scheme that allows to escape from local minima. Differential evolution has been recently introduced as an efficient direct search method for optimizing real-valued multi-modal objective functions (Storn and Price 1997). The method is conceptually simple and attractive, but little is known about its behaviour in real world applications. This paper explores this method as an alternative to current practice for solving the parameter estimation task, and attempts to assess ist robustness, measured in terms of in-sample and out-of-sample performance. A benchmark comparison against backpropagation of conjugate gradients is based on Austrian interregional telecommunication traffic data.

1. Introduction

The development of spatial interaction models is one of the major intellectual achievements and, at the same time, perhaps the most useful contribution of spatial analysis to social science literature. Since the pioneering work of Wilson (1970) on entropy maximization, there have been surprisingly few innovations in the design of spatial interaction models. The competing destinations version of Fotheringham (1983), the use of genetic algorithms to breed new forms of spatial interaction models (Openshaw 1988, Diplock 1996, Turton, Openshaw and Diplock 1997, Fischer and Leung 1998) and the design of single hidden layer neural spatial interaction models (Openshaw 1993, Fischer and Gopal 1994) are the principal exceptions.

Neural spatial interaction models are termed neural in the sense that they are based on neural computational models, inspired by neuroscience. They are more closely related to spatial interaction models of the gravity type, and under commonly met conditions they can be understood as a special class of general feedforward neural network models

with a single hidden layer and sigmoidal transfer functions (Fischer 1998). This class of networks can provide approximation within an arbitrary precision (i.e. it has universal approximation property), as proven by Hornik et al. (1989).

Learning from examples, the problem for which neural networks were designed for to solve, is one of the most important research topics in artificial intelligence. A possible way to formalize learning from examples is to assume the existence of a function representing the set of examples and, thus, enabling to generalize. This can be called a function reconstruction from sparse data (or in mathematical terms, depending on the required precision, approximation or interpolation problem, respectively). Within this general framework, the main issues of interest are the representational power of a given network model and the procedures for obtaining the optimal network parameters. In this contribution, the second issue, network training (i.e. parameter estimation), will be addressed.

Many training (learning) methods find their roots in function minimization algorithms, which can be classified as local or global minimization algorithms. Local minimization algorithms, such as the gradient descent and the conjugate gradient methods (for more details, see Fischer and Staufer 1999), are fast, but usually converge to local minima. In contrast, global minimization algorithms have heuristic strategies to help escape from local minima.

Stochastic global search methods rely on probability to make decisions. The simplest probabilistic algorithm uses restarts to bring a search out of local minima when little improvememt can be made locally. More advanced random search methods rely on probability to indicate whether a search should ascend from a local minimum, for example, simulated annealing, when it accepts uphill movements. Other stochastic search methods rely on probability to decide which intermediate points to interpolate as new starting points, for example, random recombinations and mutations in evolutionary algorithms.

It is the objective of this paper to adopt the Differential Evolution Method (DEM), recently introduced by Storn and Price (1996, 1997), as a novel approach for parameter estimation and to assess its generalization performance in a benchmark comparison

against backpropagation of conjugate gradients in a neural spatial interaction modelling environment.

The remainder of the paper is as follows. The next section provides a summarized description of single hidden layer neural spatial interaction models (Section 2). The parameter estimation problem is defined in Section 3 as a problem of minimizing the sum-of-squares error function along with a brief characterization of local search procedures that are generally used to solve this problem. The novel stochastic global minimization procedure, differential evolution, is outlined in some detail in Section 4. The testbed used for the evaluation of this parameter estimation procedure is Austrian interegional telecommunication traffic data (see Fischer and Gopal 1994), because this data set is known to pose a difficult problem to neural networks using gradient based learning due to multiple local minima. Section 5 reports on a set of experimental tests carried out to identify an optimal parameter setting and to assess the efficacy of the approach utilizing the Polak-Ribiere version of conjugate gradient error back-propagation as a benchmark. Section 6 summarizes the results achieved and outlines directions for future research.

2. The class of neural spatial interaction models under consideration

Suppose we are interested in approximating an N-dimensional spatial interaction function, where $\boldsymbol{F} : \boldsymbol{R}^N \rightarrow \boldsymbol{R}$ as N-dimensional Euclidean real space is the input space and $\boldsymbol{R}$ as 1-dimensional Euclidean real space is the output space. This function should estimate spatial interaction flows from regions of origin to regions of destination. (In practice, only bounded subsets of the spaces are considered.) The function $\boldsymbol{F}$ is not explicitly known, but given by a finite set of samples $S = \{(x^k, y^k), k = 1,..., K\}$, so that $\boldsymbol{F}(x^k) = y^k$, $k = 1,..., K$. The set S is the set of pairs of input and output vectors. The task is to find a continuous function which approximates (or interpolates) set S. In real world applications, K is a small number and the samples contain noise.

To approximate $\boldsymbol{F}$, we consider the class of neural spatial interaction models $\boldsymbol{\Omega}$ with one hidden layer, N input units, J hidden units and one output unit. $\boldsymbol{\Omega}$ consists of a composition of transfer functions so that the (single) output y of $\boldsymbol{\Omega}$ is:

$$y = \Omega(\mathbf{x}, \mathbf{w}) = \psi\left(\sum_{j=0}^{J} w_j \varphi_j \left(\sum_{n=0}^{N} w_{jn} x_n \right) \right), \tag{1}$$

Vector $\mathbf{x} = (x_1,..., x_N)$ is the input vector augmented with a bias signal x_0 which can be thought as being generated by a 'dummy unit' whose output is clamped at 1. The w_{jn}'s represent input to hidden connection weights and the w_j's hidden to output weights (including the biases). The symbol $\mathbf{w}$ is a convenient shorthand notation of the $d = (J (N + 1) + J + 1)$ - dimensional vector of all the w_{jn} and w_j network weights and biases (i.e. model parameters). $\varphi_j(.)$ and $\psi(.)$ are differentiable non-linear transfer functions of the hidden units $j = 1,..., J$ and the output unit, respectively. Following Fischer and Gopal (1994), we will consider only the case $N = 3$, i.e. the input space will be a closed interval of the three-dimensional Euclidean space $\boldsymbol{R}^3$. The three input units correspond to the independent variables of the classical unconstrained spatial interaction model of the gravity type. They represent measures of origin propulsiveness, destination attractiveness and spatial separation. The output unit corresponds to the dependent variable of the classical model and represents the spatial interaction flows from origin to destination.

One of the major issues in neural spatial interaction modelling includes the problem of selecting an appropriate member of model class $\boldsymbol{\Omega}$ in view of a particular real world application. This model specification problem includes both the choice of appropriate transfer functions φ_j and ψ and the determination of an adequate network topology of $\boldsymbol{\Omega}$ (i.e. the number of hidden units J). Clearly, the model choice problem and the parameter estimation problem, i.e. the determination of an optimal set of model parameters, are intertwined in the sense that if a good model specification can be found, the success of which depends on the particular problem, then the step of parameter estimation (also termed network training) may become easier to perform.

In this contribution we focus only on the parameter estimation problem. Without loss of generality we assume the transfer functions $\varphi_j(.) = \varphi(.) = \psi(.)$ for all $j = 1,..., J$, and

equal to the logistic function and, thus, consider the special class $\Omega_L(\mathbf{x}, \mathbf{w})$ of functions $\Omega(\mathbf{x}, \mathbf{w})$:

$$\Omega_L(\mathbf{x}, \mathbf{w}) = \left\{1 + \exp\left[- \sum_{j=0}^{J} w_j \left(1 + \exp\left(- \sum_{n=0}^{N} w_{jn} x_n \right) \right)^{-1} \right] \right\}^{-1} \tag{2}$$

The approximation of Ω_L then merely depends on the learning samples S, and the learning (training) algorithm that determines the parameter $\mathbf{w}$ from S and the number J of hidden units.

3. The parameter estimation problem and local minimization procedures

Without loss of generality, we assume the neural spatial interaction model to have a fixed topology, i.e. J is predetermined (in this study $J = 10$). Then, the goal of learning is to find suitable values $\mathbf{w}^*$ for the network weights of the model such that the underlying mapping $\boldsymbol{F} : \boldsymbol{R}^3 \rightarrow \boldsymbol{R}$ represented by the training set $S = \{(x^k, y^k), k = 1,..., K\}$, is approximated or learned, where k is the index of the training instance. y^k, $k = 1,..., K$ are scalars representing the desired network output (i.e. the spatial interaction flows) corresponding to x^k, $k = 1,..., K$. The process of determining optimal parameter values is called training or learning and can be formulated in terms of minimization of an appropriate error function (or cost function) E to measure the degree of approximation with respect to the actual setting of network weights. The most common error function is the squared-error function of the patterns over the finite set of training data, so that the parameter estimation problem may be defined as the following minimization problem:

$$\min_{\mathbf{w}} E(\mathbf{w}, S) = \min_{\mathbf{w}} \sum_{(x^k, y^k) \in S} (\Omega_L(x^k, \mathrm{w}) - y^k)^2, \tag{3}$$

where the minimization parameter is the weight vector $\mathbf{w}$ defining the search space. In this way, the problem of network trainingg has been formulated in term of the minimization of the error function E. This error function is a function of the adaptive

model parameters, i.e. network weights and biases. The derivatives of this function with respect to the model parameters can be obtained in a computationally efficient way using the backpropagation technique (see, e.g., Gopal and Fischer 1996; and Fischer and Staufer 1999, for more details on the equations of this technique). The minimization of continuous differentiable functions of many variables is a problem that has been widely studied, and many of the non-linear minimization algorithms available are directly applicable to the training of neural spatial interaction models as described in Section 2. The general scheme of these algorithms can be formulated as follows:

(i) Choose an initial vector **w** in parameter space and set $\tau = 1$;

(ii) Determine a search direction $\mathbf{d}(\tau)$ and a step size $\eta(\tau)$ so that

$$E(\mathbf{w}(\tau) + \eta(\tau)) < E(\mathbf{w}(\tau)), \tau = 1,2,...; \quad (4)$$

(iii) Update the parameter vector

$$\mathbf{w}(\tau + 1) = \mathbf{w}(\tau) + \eta(\tau)\mathbf{d}(\tau), \tau = 1,2,...; \quad (5)$$

(iv) If $dE(\mathbf{w})/d\mathbf{w} \neq 0$ then set $\tau = \tau + 1$ and go to (ii), else return $\mathbf{w}(\tau + 1)$ as the desired minimum.

In this study we refer to the Polak-Ribiere variant of the conjugate gradient procedure (Press et al. 1992), a mature local optimization method, which is used as a benchmark in Section 5. This algorithm computes the sequence of search directions as:

$$\mathbf{d}(\tau) = -\nabla E(\mathbf{w}(\tau)) + \beta(\tau)\mathbf{d}(\tau - 1), \tau = 1,2,...; \quad (6)$$

with

$$\mathbf{d}(0) = -\nabla E(\mathbf{w}(0)), \quad (7)$$

where $\beta(\tau)$ is a scalar parameter in the form

$$\beta(\tau) = \frac{[\nabla E(\mathbf{w}(\tau)) - \nabla E(\mathbf{w}(\tau - 1))]^T \nabla E(\mathbf{w}(\tau))}{\nabla E(\mathbf{w}(\tau - 1))^T \nabla E(\mathbf{w}(\tau - 1))} \text{ for } \tau = 1,2,... \quad (8)$$

$\mathbf{w}(\tau - 1)^T$ is the transpose of $\mathbf{w}(\tau - 1)$. The definition (8) of β guarantees that for the sequence of vectors $\mathbf{d}(\tau)$ the condition $\mathbf{d}(\tau - 1)^T \nabla E(\mathbf{w}(\tau)) = 0$ holds. The parameter $\eta = \eta(\tau)$ is chosen to minimize:

$$E(\mathbf{w}(\tau)+\eta(\tau)\mathbf{d}(\tau)),\ \tau=1,2,... \quad (9)$$

in the τ-th iteration. This gives the automatic procedure for setting the step length, once the search direction $\mathbf{d}(\tau)$ has been determined.

This and other local optimization methods tend to have difficulties, when the surface is flat (i.e. gradient is close to zero), when gradients are in a large range, and when the surface is very rugged. When gradients vary greatly, the search may progress too slowly, when the gradient is small and may overshoot where the gradient is large. When the error surface is rugged, a local search from a random starting point generally converges to a local minimum close to the initial point and a worse solution than the global minimum.

4. A new global search approach: The differential evolution method

Global search algorithms employ heuristics to allow to escape from local minima. These algorithms can be classified as probabilistic or deterministic. Of the few deterministic global minimization methods developed, most apply deterministic heuristics to bring search out of a local minimum. Other methods, like covering methods, recursively partition the search space into subspaces before searching. None of these methods operate well or provide adequate coverage when the search space is large as it is usually the case in neural spatial interaction modelling.

Probabilistic global minimization methods rely on probability to make decisions. The simplest probabilistic algorithm uses restarts to bring search out of a local minimum when little improvement can be made locally. More advanced methods rely on probability to indicate whether a search should ascend from a local minimum: simulated annealing, for example, when it accepts uphill movements. Other probabilistic algorithms rely on probability to decide which intermediate points to interpolate as new trial parameter vectors: random recombinations and mutations in evolutionary algorithms (see for example, Fischer and Leung 1998).

Central to global search procedures is a strategy that generates variations of the parameter vectors. Once a variation is generated, a decision has to be made whether or not to accept the newly derived trial parameter. Standard direct search methods (with few exceptions such as simulated annealing) utilize the greedy criterion to make the decision. Under this criterion, a new parameter vector is accepted if and only if it reduces the value of the error function. Although this decision process converges relatively fast, it has the risk of entrappment in a local minimum. Some stochastic search algorithms like genetic algorithms, and evolution strategies employ a multipoint search strategy, in order to escape from local minima.

The Differential Evolution Method (DEM), originally developed by Storn and Price (1996, 1997), is a global optimization algorithm that employs a structured, yet randomized parallel multipoint search strategy which is biased towards reinforcing search points at which the error function $E(\mathbf{w})$ being minimized has relatively low values. The DEM is similar to simulated annealing in that it employs a random (probabilistic) strategy. But one of the apparent distinguishing features of DEM is its effective implementation of parallel multipoint search. DEM maintains a collection of samples from the search space rather than a single point. This collection of samples is called *population* of trial solutions.

To start the stochastic multipoint search, an initial population P of, say M, d-dimensional parameter vectors $P(0) = \{\mathbf{w}_0(0),..., \mathbf{w}_{M-1}(0)\}$ is created. For the parameter estimation problem at hand $d = J\ (N + 1) + J + 1$. Usually this initial population is created randomly because it is not known a priori, where the globally optimal parameter is likely to be found in the parameter space. If such information is given, it may be used to bias the initial population towards the most promising regions of the search space by adding normally distributed random deviations to this a priori given solution candidate. From this initial population, subsequent populations $P(1)$, $P(2)$,..., $P(t)$,... will be computed by a scheme that generates new parameter vectors by adding the weighted difference of two vectors to a third. If the resulting vector yields a lower error function value than a predetermined population member, the newly generated vector will replace the vector which it was compared to, otherwise the old vector is retained. Similarly to evolution strategies, the greedy criterion is used in the iteration process and the probability distribution functions determining vector mutations are not a priori given.

The scheme for generating $P(t + 1)$ from $P(t)$ with $t \geq 0$ may be summarized by three major stages: the construction of $\mathbf{v}(t + 1)$-vectors from vector $\mathbf{w}(t)$-vectors (Stage 1), the construction of $\mathbf{u}(t + 1)$-vectors from $\mathbf{v}(t + 1)$ vectors and $\mathbf{w}(t)$-vectors (Stage 2), and the decision criterion whether or not the $\mathbf{u}(t + 1)$-vector should become members of the population (Stage 3) representing possible solutions of the parameter estimation problem under consideration at step $t + 1$ of the iteration process. The iteration process continues until some stopping criterion applies.

Stage 1: For each population member $\mathbf{w}_m(t)$, $m = 0, 1,..., M - 1$, a perturbed vector $\mathbf{v}_m(t + 1)$ is generated according to:

$$\mathbf{v}_m(t + 1) = \mathbf{w}_{best}(t) + \kappa(\mathbf{w}_{r1}(t) - \mathbf{w}_{r2}(t)) \tag{10}$$

with $r1$, $r2$ integers chosen randomly from $\{0,..., M - 1\}$ and mutually different. The integers are also different from the running index m. $\kappa \in (0, 2]$ is a real constant factor which controls the amplification of the differential variation $(\mathbf{w}_{r1}(t) - \mathbf{w}_{r2}(t))$. The parameter vector $\mathbf{w}_{best}(t)$ which is perturbed to yield $\mathbf{v}_m(t + 1)$ is the best parameter vector of population $P(t)$.

Stage 2: In order to increase the diversity of the new parameter vectors, some specific type of crossover may be introduced. We will use the crossover of the exponential type (Storn and Price 1996), yielding the vector:

$$\mathbf{u}_m(t + 1) = (u_{0m}(t + 1),\ldots, u_{(d\text{-}1)m}(t + 1)) \tag{11}$$

where

$$\mathbf{u}_{km}(t+1) = \begin{cases} \mathbf{v}_{km}(t+1) & \text{for k} = < i >_d, \ldots, < i + R - 1 >_d \\ \mathbf{w}_{km}(t) & \text{for all other } k \in [0, d-1] \end{cases} \tag{12}$$

is formed. The brackets $<>_d$ denote the modulo function with modulus d. In other words, a sequence of R coordinates of vector $\mathbf{u}(t + 1)$ is identical to the corresponding coordinates of vector $\mathbf{v}(t + 1)$, whereas the other coordinates of $\mathbf{u}(t + 1)$ are retained as the original values of $\mathbf{w}(t)$. The starting index i in (12) is a randomly chosen integer

from the interval [0, d - 1]. The integer R, which denotes the number of parameters that are going to be exchanged, is drawn from the interval [1, d] with the probability $Pr\ (R = \eta) = (\mathrm{CR})^{\eta}$, where $\eta > 0$ and $CR \in [0, 1]$ is the crossover probability and forms a control variable for the scheme. The random decisions for both i and R are made anew for each newly generated trial vector $\mathbf{u}_m(t + 1)$. It is worth noting that $CR = 1$ implies $\mathbf{u}_m(t + 1) = \mathbf{v}_m(t + 1)$.

Stage 3: The decision whether or not $\mathbf{u}_m(t + 1)$ should become a member of $P(t + 1)$, is based on the greedy criterion. If:

$$E(\mathbf{u}_m(t + 1)) < E(\mathbf{w}_m(t)) \tag{13}$$

then $\mathbf{w}_m(t)$ is replaced by $\mathbf{u}_m(t + 1)$ otherwise the old value $\mathbf{w}_m(t)$ is retained as $\mathbf{w}_m(t + 1)$.

When using non-linear optimization algorithms such as DEM, some choice must be made when to stop the training process. Possible choices are listed below:

(i) Stop after a fixed number of iterations. The problem with this approach is that it is difficult to know a priori how many iterations would be appropriate. But an approximate idea can be obtained from some preliminary tests.

(ii) Stop when the error function falls below some specified value. This criterion suffers from the problem that the a priori specified value may never be reached so a limit on iterations as in (i) is also required.

(iii) Stop when the relative change in error function falls below some a priori specified value. This may lead to premature termination if the error function decreases relatively slowly during some part of the training process.

(iv) Stop training when the error measured using an independent validation set starts to increase. This approach, called early stopping or cross-validation, may be used as part of a strategy to optimize the generalization performance of the network model (see Fischer and Gopal 1994 for details).

In practice, some combination of the above strategies may be employed as part of a largely empirical process of parameter estimation. We have chosen the first termination criterion in the experiments that will be described in the next section.

5. Performance test results

This section presents results of our experiments and the performance of the Differential Evolution Method to solve the parameter estimation problem (3) for the neural spatial interaction model (2) with three inputs, a single hidden layer with J = 10 hidden units and a single output unit. The output unit represents the intensity of telecommunication flows from one origin region to a destination region and the input units the three independent variables of the classical gravity model: the potential pool of telecommunication activities in the origin region, the potential draw of telecommunication activities in the destination region, and a factor representing the inhibiting effect of geographic separation from the origin to the destination region. The ultimate goal of this neural spatial interaction model is to exhibit good generalization performance, i.e. to make good predictions for new inputs. The spatial interaction modelling prediction accuracy (generalization measured in terms of out-of-sample performance) is generally more important than fast learning.

The model performance is measured in this study by the *average relative variance* $ARV(S)$ of a set S of patterns given by (Fischer and Gopal 1994):

$$\begin{aligned} ARV(S) &= \frac{\sum_{(x^k,y^k)\in S}(y^k - \Omega_L(x^k,\mathbf{w}))^2}{\sum_{(x^k,y^k)\in S}(y^k - \bar{y})^2} \\ &= \frac{1}{N_S}\frac{1}{\hat{\sigma}^2}\sum_{(x^k,y^k)\in S}(y^k - \Omega_L(x^k,\mathbf{w}))^2 \end{aligned} \tag{14}$$

where y^k denotes the target value and $\bar{y}$ the average over the K desired values in S. The averaging, i.e. division by N_S makes $ARV(S)$ independent of the size of the set S. Thus $ARV(S)$ provides a normalized mean squared error metric for assessing the in-sample and out-of-sample performance of trained neural spatial interaction models.

ARV(S) = 1 if the estimate is equivalent to the mean of the data (i.e, $\Omega_L(x^k, \mathbf{w}) = \bar{y}$). The division by the estimated variance $\hat{\sigma}_2$ of the data removes the dependence on the dynamic range of the data. In the following experiments, the ARV set {*ARV*1, *ARV*2} will refer to the average relative error corresponding to the {training set, test set}.

5.1. The data

The experiments were conducted using Austrian telecommunication flow data (see Fischer and Gopal 1994 for more details). The data set was constructed from three data sources: a (32, 32)-interregional telecommunication flow matrix, a (32, 32)-distance matrix, and gross regional products for the 32 telecommunication regions. It contains 992 4-tuples (x_1, x_2, x_3, y), where the first three components represent the input vector $\mathbf{x} = (x_1, x_2, x_3)$ and the last component the target output of the neural spatial interaction model, i.e. the telecommunication intensity from one region of origin to another region of destination. Input and target output data were preprocessed to logarithmically transformed data scaled into [0, 1]. The telecommunication data stem from network measurements of carried telecommunication traffic in Austria in 1991, in terms of erlang, which is defined as the number of phone calls (including facsimile transmission) multiplied by the average length of the call (transfer) divided by the duration of measurement. This data set was randomly divided into two separate subsets: about two thirds of the data were used for parameter estimation only, and one third as test set for assessing the generalization performance. There was no overlapping of the two sets of data. In comparison to Fischer and Gopal (1994), the data utilized was updated to take some measurement errors into account. Consequently the results of the experimental work described below cannot be directly compared to the previous results.

5.2. Experimental Work and Results

Before applying the Differential Evolution Method to the problem at hand, values for the DEM-parameters $\kappa \in (0, 2]$, $CR \in [0, 1]$ and M, the population size must be chosen. There is no way to *a priori* define useful combinations of values. In order to identify

good settings, extensive computational tests with different combinations of values have been performed. Since all simulations have similar computational complexity, iterations to converge to the optimal *ARV*2 value were used as a measure of learning time. Each experiment (i.e. a fixed combination of DEM parameter values) was repeated six times, the network model being initialized with a different set of random weights from [-0.3, 0.3] before each trial. To enable more accurate comparisons, the population of the parameter vectors was initialized with the same six sets of (d = 51) random weights for all experiments. All experiments were done on Pentium PC 400 Mhz under Linux 2.0 using the egcs-1.0.3 C-compiler. In all experiments the algorithm did run for 7000 generations.

Table 1 presents the results of a first series of experiments illustrating the effects of the κ-parameter (0.7, 0.8, 0.9, 1.0, 1.1) that controls the amplification of differential evolution. The crossover parameter was set to 1.0 and the cardinality M of the population to M = 1000. In-sample (out-of-sample) performance is measured in terms of *ARV*1 (*ARV*2). Some considerations are worth making. First, there is strong evidence of the robustness of the algorithm (measured in terms of standard deviation) both with respect to the choice of the DEM-parameter κ and to the choice of the initial population of the parameter vectors. Second, κ = 0.9 leads to the best result in terms of the average out-of-sample performance (*ARV*2 = 0.2280).

Table 2 shows how in-sample and out-of-sample performance changes depending on the crossover parameter *CR*. The κ-parameter was set at value κ = 0.9 and M = 1000. The best result (averaged over the six independent simulation runs) in terms of average in-sample and out-of-sample performance, was obtained with *CR* = 1.0 (0.2058 and 0.2280; respectively). The Differential Evolution algorithm generates stable performance over the six runs.

The third series of experiments (Table 3) involves variation of the cardinality M, with a range of M = 50, 100, 200, 400, and 1000. The *CR*-parameter was set at value *CR* = 1.0 and κ at 0.9. The results obtained are summarized in Table 3. Two observations are noteworthy here. First, larger populations of parameter vectors tend to provide better results, because a large population is more likely to contain representatives from a larger number of hyperplanes. Second, the computational cost of evolving large

Table 1. Performance of the differential evolution method: different parameter settings with $M = 1000$, $CR = 1.0$ and varying κ

Control Parameter κ	Trial	Iterations	ARV1		ARV2	
$\kappa = 0.7$	1	181	0.2137		0.2322	
	2	116	0.2195		0.2327	
	3	152	0.2080		0.2273	
	4	350	0.2106		0.2298	
	5	128	0.2122		0.2326	
	6	373	0.2042		0.2274	
	Average	217 ± 156	0.2114	(0.0047)	0.2304	(0.0023)
$\kappa = 0.8$	1	2018	0.1988		0.2297	
	2	813	0.2128		0.2327	
	3	705	0.2047		0.2328	
	4	543	0.2080		0.2323	
	5	5474	0.1912		0.2363	
	6	440	0.2128		0.2305	
	Average	1666 ± 3809	0.2047	(0.0077)	0.2307	(0.0023)
$\kappa = 0.9$	1	1218	0.2152		0.2341	
	2	456	0.2183		0.2301	
	3	4590	0.1841		0.2162	
	4	622	0.2149		0.2335	
	5	1371	0.2111		0.2244	
	6	3328	0.1912		0.2294	
	Average	1931 ± 2659	0.2058	(0.0132)	0.2280	(0.0062)
$\kappa = 1.0$	1	250	0.2426		0.2531	
	2	207	0.2305		0.2360	
	3	236	0.2297		0.2378	
	4	507	0.2255		0.2384	
	5	111	0.2459		0.2621	
	6	597	0.2246		0.2364	
	Average	318 ± 279	0.2331	(0.0082)	0.2440	(0.0100)
$\kappa = 1.1$	1	2678	0.2121		0.2312	
	2	6089	0.2068		0.2235	
	3	390	0.2294		0.2326	
	4	4243	0.2174		0.2311	
	5	884	0.2227		0.2317	
	6	4846	0.2146		0.2326	
	Average	3188 ± 2901	0.2172	(0.0073)	0.2304	(0.0032)

Average: Performance values represent the mean (standard deviation in brackets) of six simulations differing in the initial parameter values randomly chosen from [-0.3, 0.3];
Number of *Iterations* required to reach the parameter vector that provides the best out-of-sample performance;
ARV1: In-sample performance measured in terms of relative average variances;
ARV2: Out-of-sample performance measured in terms of relative average variances

populations does not seem to be rewarded by a comparable out-of-sample improvement of the solutions obtained, for $M > 200$. In fact, for $M = 200$ we obtained an average *ARV*1 of 0.2136 and an average *ARV*2 of 0.2276 over the six runs. For $M = 1000$ we obtained *ARV*1 = 0.2058 and *ARV*2 = 0.2280 averaged over the six trials.

The best settings obtained over six trial runs are presented in Table 4. For graphical presentation, *ARV*1 and *ARV*2 indices of the best solution obtained were plotted against learning time, measured in terms of the number of iterations (see Figure 1). Clearly, we

Table 2. Performance of the differential evolution method: different parameter settings with $M = 1000$, $\kappa = 0.9$ and varying CR

Control Parameter CR	Trial	Iterations	ARV1		ARV2	
$CR = 0.6$	1	4286	0.2260		0.2392	
	2	4589	0.2251		0.2377	
	3	896	0.2317		0.2361	
	4	1321	0.2336		0.2395	
	5	6701	0.2202		0.2443	
	6	2435	0.2324		0.2414	
	Average	3371 ± 3330	0.2282	(0.0048)	0.2397	(0.0026)
$CR = 0.7$	1	4310	0.2232		0.2340	
	2	6979	0.2215		0.2381	
	3	5748	0.2273		0.2391	
	4	5107	0.2190		0.2347	
	5	5879	0.2248		0.2397	
	6	1520	0.2291		0.2304	
	Average	4924 ± 3404	0.2241	(0.0034)	0.2360	(0.0033)
$CR = 0.8$	1	4087	0.2261		0.2413	
	2	2654	0.2240		0.2354	
	3	309	0.2409		0.2400	
	4	653	0.2315		0.2342	
	5	816	0.2314		0.2369	
	6	5072	0.2210		0.2329	
	Average	2265 ± 2807	0.2292	(0.0065)	0.2368	(0.0030)
$CR = 0.9$	1	1165	0.2310		0.2382	
	2	4270	0.2258		0.2354	
	3	182	0.2426		0.2473	
	4	3090	0.2245		0.2387	
	5	5316	0.2287		0.2358	
	6	159	0.2411		0.2344	
	Average	2364 ± 2952	0.2323	(0.0071)	0.2383	(0.0043)
$CR = 1.0$	1	1218	0.2152		0.2341	
	2	456	0.2183		0.2301	
	3	4590	0.1841		0.2162	
	4	622	0.2149		0.2335	
	5	1371	0.2111		0.2244	
	6	3328	0.1912		0.2294	
	Average	1931 ± 2659	0.2058	(0.0132)	0.2280	(0.0062)

Average: Performance values represent the mean (standard deviation in brackets) of six simulations differing in the initial parameter values randomly chosen from [-0.3, 0.3];
Number of *Iterations* required to reach the parameter vector that provides the best out-of-sample performance;
ARV1: In-sample performance measured in terms of relative average variances;
ARV2: Out-of-sample performance measured in terms of relative average variances

would like to stop when the *ARV*2 curve arrives at its minimum (i.e. after 4590 iterations), even though (*ARV*1 and) *ARV*2 decrease only very slowly after 3000 iterations.

For comparison purposes, we implemented the Polak-Ribiere version of conjugate gradient error backpropagation as a benchmark. The runs were made using the batch mode of operation and letting the algorithm run six times. The results of the benchmark comparison are summarized in Table 5.

Table 3. Performance of the differential evolution method: different parameter settings with $\kappa = 0.9$, $CR = 1.0$ and varying M

Population Size	Trial	Iterations	ARV1		ARV2	
$M = 50$	1	2900	0.2291		0.2324	
	2	2320	0.2264		0.2390	
	3	200	0.2292		0.2446	
	4	960	0.2250		0.2363	
	5	5880	0.2244		0.2325	
	6	980	0.2229		0.2421	
	Average	2207 ± 3673	0.2262	(0.0023)	0.2378	(0.0046)
$M = 100$	1	3060	0.2191		0.2357	
	2	510	0.2205		0.2257	
	3	1920	0.2206		0.2351	
	4	440	0.2246		0.2349	
	5	280	0.2348		0.2340	
	6	5490	0.2167		0.2379	
	Average	1950 ± 3540	0.2227	(0.0059)	0.2339	(0.0038)
$M = 200$	1	3890	0.2068		0.2231	
	2	760	0.2182		0.2307	
	3	6200	0.2087		0.2276	
	4	320	0.2256		0.2287	
	5	4950	0.2128		0.2295	
	6	2235	0.2095		0.2258	
	Average	3059 ± 3141	0.2136	(0.0065)	0.2276	(0.0025)
$M = 400$	1	1233	0.2128		0.2332	
	2	3302	0.2070		0.2265	
	3	326	0.2217		0.2317	
	4	3825	0.2088		0.2237	
	5	1790	0.2128		0.2277	
	6	3707	0.2052		0.2249	
	Average	2364 ± 2038	0.2114	(0.0054)	0.2279	(0.0035)
$M = 1000$	1	1218	0.2152		0.2341	
	2	456	0.2183		0.2301	
	3	4590	0.1841		0.2162	
	4	622	0.2149		0.2335	
	5	1371	0.2111		0.2244	
	6	3328	0.1912		0.2294	
	Average	1931 ± 2659	0.2058	(0.0132)	0.2280	(0.0062)

Average: Performance values represent the mean (standard deviation in brackets) of six simulations differing in the initial parameter values randomly chosen from [-0.3, 0.3];
Number of *Iterations* required to reach the parameter vector that provides the best out-of-sample performance;
ARV1: In-sample performance measured in terms of relative average variances;
ARV2: Out-of-sample performance measured in terms of relative average variances

If in-sample and out-of-sample performance is more important than fast learning, then DEM exhibits superiority. As can be seen by comparing the *ARV*-values, DEM leads to a statistically higher prediction performance in average. The average generalization performance, measured in terms of *ARV*2, is 0.2276 (DEM) and 0.2385 (backpropagation of conjugate gradients). DEM is rather stable over the different trials. If, however, the goal is to minimize learning time and a sacrifice in generalization accuracy is acceptable, then conjugate gradient error backpropagation is the method of choice. The benchmark procedure outperforms by far the DEM in terms of execution time.

Table 4. Best DEM-parameter settings

DEM-Parameter	Value
M	200
κ	0.9
CR	1.0

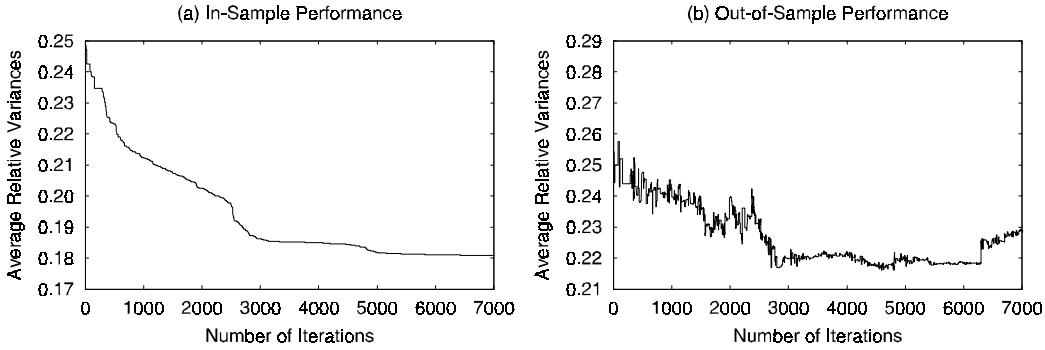

Fig. 1. Performance of the differential evolution algorithm with the best parameter setting: M = 1000, κ = 0.9 and CR = 1.0 (run 3). **(a)** In-Sample Performance; **(b)** Out-of-Sample Performance

Table 5. Benchmark comparison of the differential evolution method (M = 200, κ = 0.9, CR = 1.0) with backpropagation of conjugate gradients

Method	Trial	ARV1		ARV2	
Differential	1	0.2068		0.2231	
Evolution Method	2	0.2182		0.2307	
M = 200	3	0.2087		0.2276	
κ = 0.9	4	0.2256		0.2287	
CR = 1.0	5	0.2128		0.2295	
	6	0.2095		0.2258	
	Average	0.2136	(0.0065)	0.2276	(0.0025)
Backpropagation	1	0.2148		0.2355	
of conjugate	2	0.2282		0.2385	
gradients	3	0.2062		0.2256	
(Polak-Ribiere	4	0.2152		0.2357	
method)	5	0.2140		0.2372	
	6	0.2423		0.2584	
	Average	0.2201	(0.0118)	0.2385	(0.0098)

Average: Performance values represent the mean (standard deviation in brackets) of six simulations differing in the initial parameter values randomly chosen from [-0.3, 0.3];
ARV1: In-sample performance measured in terms of relative average variances;
ARV2: Out-of-sample performance measured in terms of relative average variances

6. Conclusions and outlook

This paper presents a global search approach, called Differential Evolution Method, suitable for minimizing non-linear and (non-)differentiable functions, such as the squared error function (see equation (3)). Little is known about the behaviour of this global search procedure in real world applications.

The method permits search over the whole parameter space, thus, providing the possibility of escaping from local minima. The algorithm employs a very simple and straightforward strategy. Indeed, the main search procedure can be written in less than 30 lines of C-code. It is also very easy to use as it needs only a few control parameters that can be chosen from well defined numerical intervals. As shown in the simulation studies, DEM successfully solves the parameter estimation problem of neural spatial interaction models. A benchmark comparison against backpropagation of conjugate gradients illustrates that it slightly outperforms the benchmark in terms of in-sample and out-of-sample performance, but at a very high price of computational costs.

The computational resources required include processing power to conduct M separate searches, since DEM is based on a multipoint search strategy. Our implementation was done on a serial platform even though multipoint search strategies are inherently parallelizable. Considerable computational resources may be required if the problem at hand has a high dimensionality (in our case $d = 51$).

The issue of accuracy has ramifications with respect to a priori knowledge of the response surface. If a correct neural spatial interaction model structure is assumed, DEM, in general, tend to be slower than conventional local optimization schemes such as the conjugate gradient procedure. This results from the inefficiency of not using information about the gradient of the error function although gradient methods could be incorporated in parallel with differential evolution. There is potential for further development of this novel strategy in this direction.

Acknowledgement

The authors gratefully acknowledge the grant no. P12681-INF provided by the Fonds zur Förderung der wissenschaftlichen Forschung (FWF).

References

Diplock GJ (1996) Building new spatial interaction models using genetic programming and a supercomputer. Paper presented at the First International Conference on GeoComputation, Leeds, September 17-19, 1996 (available from the School of Geography, University of Leeds, Leeds LS29JT, UK)

Fischer, MM (1998) Computational neural networks: A new paradigm for spatial analysis. *Environment and Planning A* 30 (10): 1873-1891

Fischer MM, Leung Y (1998) A genetic-algorithm based evolutionary computational neural network for modelling spatial interaction data. *The Annals of Regional Science* 32 (3): 437-458

Fischer MM, Gopal S (1994) Artificial neural networks: A new approach to modelling interregional telecommunication flows. *Journal of Regional Science* 34 (4): 503-527

Fischer MM, Staufer P (1999) Optimization in an error backpropagation neural network environment with a performance test on a spectral pattern classification problem. *Geographical Analysis* 31 (in press)

Fotheringham AS (1983) A new set of spatial interaction models: The theory of competing destinations. *Environment and Planning A* 15: 15-36

Goldberg D (1989) *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Reading, MA

Gopal S, Fischer MM (1996) Learning in single hidden-layer feedforward network models: Back-propagation in a spatial interaction modelling context. *Geographical Analysis* 28 (1): 38-55

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Networks* 2: 359-366

Openshaw S (1988) Building an automated modelling system to explore a universe of spatial interaction models. *Geographical Analysis* 20 (1): 31-46

Openshaw S (1993) Modelling spatial interaction using a neural net, In Fischer MM, Nijkamp P (eds) *Geographical information systems, spatial modelling, and policy evaluation*, pp 147-164. Springer, Berlin Heidelberg New York

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) *Numerical recipes in C. The art of scientific computing.* Cambridge University Press, Cambridge

Storn R, Price K (1996) Minimizing the real functions of the ICEC'96 contest by differential evolution. *IEEE Conference on Evolutionary Computation*, Nagoya, pp 842-844

Storn R, Price K (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11: 341-359

Turton I, Openshaw S, Diplock GJ (1997) A genetic programming approach to building new spatial model relevant to GIS. In Kemp Z (ed) *Innovations in GIS 4*, pp 89-102. Taylor and Francis, London

Wilson AG (1970) *Entropy in urban and regional modelling.* Pion, London