Theory of Stochastic Process
Vol.0 (00), no.0 ,2001, pp.00-00

THOMAS HELLSTRÖM and KENNETH HOLMSTRÖM[1]

GLOBAL OPTIMIZATION OF COSTLY NONCONVEX FUNCTIONS, WITH FINANCIAL APPLICATIONS[2]

The paper considers global optimization of costly objective functions, i.e. the problem of finding the global minimum when there are several local minima and each function value takes considerable CPU time to compute. Such problems often arise in industrial and financial applications, where a function value could be a result of a time-consuming computer simulation or optimization. Derivatives are most often hard to obtain, and the algorithms discussed make no use of such information.

Response surface methods are promising for global optimization of costly nonconvex objective functions. We discuss our implementation of an algorithm by Powell and Gutmann based on the use of radial basis functions (RBF). Another interesting response surface method is the Efficient Global Optimization (EGO) method by Jones et al. We have implemented these two methods, together with the DIRECT and constrained DIRECT method by Jones in the TOMLAB optimization environment (Holmström (1999)). We discuss the application of these global optimization methods for parameter estimation in trading algorithms and in models for time series prediction.

AMS 2000 subject classifications. 90C26,90C56,91B28.

Key words and phrases. Mathematical Programming, Nonlinear Programming, Matlab, Mathematical Software, Algorithms, Nonconvex Optimization, Global Optimization, Time Series Predictions, Financial time series.

[1]Invited speaker
[2]Invited talk

1. INTRODUCTION

The task of global optimization is to find the set of parameters x in the feasible region $\Omega \subset \mathbf{R^d}$ for which the objective function $f(x)$ obtains its smallest value. In other words, a point x^* is a *global optimizer* to $f(x)$ on Ω, if $f(x^*) \leq f(x)$ for all $x \in \Omega$. On the other hand, a point $\hat{x}$ is a *local optimizer* to $f(x)$, if $f(\hat{x}) \leq f(x)$ for all x in some neighborhood around $\hat{x}$. Obviously, when the objective function has several local minima, there could be solutions that are locally optimal but not globally optimal and standard local optimization techniques are likely to get stuck before the global minimum is reached. Therefore, some kind of global search is needed to find the global minimum with some reliability.

The global optimization page (http://solon.cma.univie.ac.at/~neum/glopt.html), maintained by Arnold Neumaier, contains many commented links to online information relevant to global optimization. There is also an online survey of different global optimization methods at http://www.cs.sandia.gov/opt/survey/. The emphasis in this paper is on problems with costly objective functions.

In our Applied Optimization and Modeling group at Mälardalen University we work in several applied areas with a need for global optimization techniques. One example is the problem of designing a passenger train, where the objective is to minimize the total mass with constraints on ride quality measures. The design parameters are the carbody mass and frequencies, the bogie frame mass and yaw damper attachment positions. In this problem, a single function value is computed by running an expensive (time-consuming) computer simulation. With a simplified model, one simulation takes more than three minutes and with an accurate model, about half an hour. In computational finance, we study the prediction of various kinds of quantities related to stock markets, like stock prices, stock volatility and ranking measures. These are noisy problems with several local minima. With the increasing use of high-frequency data, simulated trading and time series analysis of multiple data series results in costly global optimization problems as presented in Hellström and Holmström (1999), and further discussed in Section 4.

Previously we have made Matlab implementations of the DIRECT (Jones, Perttunen and Stuckman 1993), the new constrained DIRECT (Jones 2001), and the Efficient Global Optimization (EGO) (Jones, Schonlau, Welch 1998) algorithms. The implementations are part of the TOMLAB optimization environment, described in Holmström (1999a, 1999b, 1999c). The implementation of the DIRECT algorithm is further discussed and analyzed in Björkman and Holmström (1999). Recently Powell (1999) and Gutmann (1999) presented an algorithm of response surface type based on radial basis function approximation. The idea of the RBF algorithm is to use radial basis function interpolation to define a utility function (Powell 1999). The next point, where the original objective function should be evaluated, is determined

by optimizing on this utility function.

In Section 2 we describe the basic RBF algorithm and discuss some special features of the implementation. The other global optimization algorithms considered are briefly described in Section 3. In Section 4 the application of global optimization methods on financial problems are discussed.

2. THE RBF ALGORITHM

Our RBF algorithm is based on the ideas presented by Gutmann (1999), with some extensions and further development. The algorithm is implemented in the Matlab routine *rbfSolve* and described in more detail in Björkman and Holmström (2001). The RBF algorithm deals with box-bounded global optimization problems of the form

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s/t & -\infty < \quad x_L \quad \leq \quad x \quad \leq \quad x_U \quad < \infty, \end{array} \tag{1}$$

where $f(x) \in \mathbf{R}$ and $x, x_L, x_U \in \mathbf{R^d}$. We assume that no derivative information is available and that each function evaluation is very expensive. For example, the function value could be the result of a time-consuming experiment or computer simulation.

2.1. Description of the Algorithm. We now consider the question of choosing the next point where the objective function should be evaluated. The idea of the RBF algorithm is to use radial basis function interpolation and a measure of 'bumpiness' of a radial function, σ say. A target value f_n^* is chosen that is an estimate of the global minimum of f. For each $y \notin \{x_1, \ldots, x_n\}$ there exists a radial basis function s_y that satisfies the interpolation conditions

$$\begin{array}{rcl} s_y(x_i) & = & f(x_i), \quad i = 1, \ldots, n, \\ s_y(y) & = & f_n^*. \end{array} \tag{2}$$

The next point x_{n+1} is calculated as the value of y in the feasible region that minimizes $\sigma(s_y)$. It turns out that the function $y \mapsto \sigma(s_y)$ is much cheaper to compute than the original function.

Here, the radial basis function interpolant s_n has the form

$$s_n(x) = \sum_{i=1}^{n} \lambda_i \phi\left(\|x - x_i\|_2\right) + b^T x + a, \tag{3}$$

with $\lambda_1, \ldots, \lambda_n \in \mathbf{R}$, $b \in \mathbf{R^d}$, $a \in \mathbf{R}$ and ϕ is either cubic with $\phi(r) = r^3$ or the thin plate spline $\phi(r) = r^2 \log r$. Gutmann (1999) considers other choices of ϕ and of the additional polynomial, see the table below. Later in Gutmann (2000) he concludes that the situation in the multiquadric and Gaussian cases is disappointing.

RBF	$\phi(r) > 0$	$p(x)$
cubic	r^3	$a^T \cdot x + b$
thin plate spline	$r^2 \log r$	$a^T \cdot x$ + b
linear	r	b
multiquadric	$\sqrt{(r^2 + \gamma^2)}$	
Gaussian	$\exp(-\gamma r^2)$	

The unknown parameters λ_i, b and a are obtained as the solution of the system of linear equations

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}, \tag{4}$$

where Φ is the $n \times n$ matrix with $\Phi_{ij} = \phi\left(\|x_i - x_j\|_2\right)$ and

$$P = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ . & . \\ . & . \\ x_n^T & 1 \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ . \\ . \\ \lambda_n \end{pmatrix}, c = \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ b_d \\ a \end{pmatrix}, F = \begin{pmatrix} f(x_1) \\ f(x_2) \\ . \\ . \\ f(x_n) \end{pmatrix}. \tag{5}$$

s_y could be obtained accordingly, but there is no need to do that as one is only interested in $\sigma(s_y)$. Powell (1992) shows that if the rank of P is $d + 1$, then the matrix

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \tag{6}$$

is nonsingular and the linear system (4) has a unique solution.

σ is defined in Gutmann (2000). For s_n in (3) it is

$$\sigma(s_n) = \sum_{i=1}^{n} \lambda_i s_n(x_i). \tag{7}$$

Further, it is shown that $\sigma(s_y)$ is

$$\sigma(s_y) = \sigma(s_n) + \mu_n(y) \left[s_n(y) - f_n^*\right]^2, \quad y \notin \{x_1, \ldots, x_n\}. \tag{8}$$

Thus minimizing $\sigma(s_y)$ subject to constraints is equivalent to minimizing g_n defined as

$$g_n(y) = \mu_n(y) \left[s_n(y) - f_n^* \right]^2, \quad y \in \Omega \setminus \{x_1, \ldots, x_n\}, \tag{9}$$

where $\mu_n(y)$ is the coefficient corresponding to y of the Lagrangian function L that satisfies $L(x_i) = 0$, $i = 1, \ldots, n$ and $L(y) = 1$. It can be computed as follows. Φ is extended to

$$\Phi_y = \begin{pmatrix} \Phi & \phi_y \\ \phi_y^T & 0 \end{pmatrix}, \tag{10}$$

where $(\phi_y)_i = \phi(\|y - x_i\|_2)$, $i = 1, \ldots, n$, and P is extended to

$$P_y = \begin{pmatrix} P \\ y^T & 1 \end{pmatrix}. \tag{11}$$

Then $\mu_n(y)$ is the $(n+1)$-th component of $v \in \mathbf{R^{n+d+2}}$ that solves the system

$$\begin{pmatrix} \Phi_y & P_y \\ P_y^T & 0 \end{pmatrix} v = \begin{pmatrix} 0_n \\ 1 \\ 0_{d+1} \end{pmatrix}. \tag{12}$$

We use the notation 0_n and 0_{d+1} for column vectors with all entries equal to zero and with dimension n and $(d+1)$, respectively. The computation of $\mu_n(y)$ is done for many different y when minimizing $g_n(y)$. This requires $O(n^3)$ operations if not exploiting the structure of Φ_y and P_y. Hence it does not make sense to solve the full system each time. A better alternative is to factorize the interpolation matrix and update the factorization for each y. An algorithm that requires $O(n^2)$ operations is described in Björkman and Holmström (2001).

When there are large differences between function values, the interpolant has a tendency to oscillate strongly. It might also happen $\min s_n(y)$ is much lower than the best known function value, which leads to a choice of f_n^* that overemphasizes global search. To handle these problems, large function values are in each iteration replaced by the median of all computed function values.

Note that μ_n and g_n are not defined at $x_1, \ldots, x_n$ and

$$\lim_{y \to x_i} \mu_n(y) = \infty, \quad i = 1, \ldots, n. \tag{13}$$

This will cause problems when μ_n is evaluated at a point close to one of the known points. The function $h_n(x)$ defined by

$$h_n(x) \;=\; \begin{cases} \frac{1}{g_n(x)}, & x \notin \{x_1, \ldots, x_n\} \\ 0, & x \in \{x_1, \ldots, x_n\} \end{cases} \tag{14}$$

is differentiable everywhere on Ω, and is thus a better choice as objective function. Instead of minimizing $g_n(y)$ in (9) one may minimize $-h_n(y)$. In our implementation we instead minimize $-\log(h_n(y))$. By this we avoid a flat minimum and numerial trouble when $h_n(y)$ is very small.

2.2. The Choice of f_n^*. For the value of f_n^* it should hold that

$$f_n^* \in \left[-\infty, \min_{y\in\Omega} s_n(y)\right]. \tag{15}$$

The case $f_n^* = \min_{y\in\Omega} s_n(y)$ is only admissible if $\min_{y\in\Omega} s_n(y) < s_n(x_i)$, $i = 1, \ldots, n$. There are two special cases for the choice of f_n^*. In the case when $f_n^* = \min_{y\in\Omega} s_n(y)$, then minimizing (9) is equivalent to

$$\min_{y\in\Omega} s_n(y). \tag{16}$$

In the case when $f_n^* = -\infty$, then minimizing (9) is equivalent to

$$\min_{y\in\Omega\setminus\{x_1,\ldots,x_n\}} \mu_n(y). \tag{17}$$

So how should f_n^* be chosen? If $f_n^* = -\infty$, then the algorithm will choose the new point in an unexplored region, which is good from a global search point of view, but the objective function will not be exploited at all. If $f_n^* = \min_{y\in\Omega} s_n(y)$, the algorithm will show good local behaviour, but the global minimum might be missed. Therefore, there is a need for a mixture of values for f_n^* close to and far away from $\min_{y\in\Omega} s_n(y)$. Gutmann (1999) describes two different strategies for the choice of f_n^*. In this paper we study one of the strategies.

The strategy, denoted **idea 1**, is to perform a cycle of length $N + 1$ and choose f_n^* as

$$f_n^* = \min_{y\in\Omega} s_n(y) - W \cdot \left(\max_i f(x_i) - \min_{y\in\Omega} s_n(y)\right), \tag{18}$$

with

$$W = \left[\frac{(N - (n - n_{init}))\mathrm{mod(N + 1)}}{N}\right]^2, \tag{19}$$

where n_{init} is the number of initial points. Here, $N = 5$ is fixed and $\max_i f(x_i)$ is not taken over all points, except for the first step of the cycle. In each of the subsequent steps the $n - n_{max}$ points with largest function value are removed (not considered) when taking the maximum. Hence the quantity $\max_i f(x_i)$ is decreasing until the

cycle is over. Then all points are considered again and the cycle starts from the beginning. More formally, if $(n - n_{init})\text{mod}(\text{N} + 1) = 0$, $n_{max} = n$, otherwise

$$n_{max} = \max\{2, n_{max} - \text{floor}((\text{n} - \text{n}_{\text{init}})/\text{N})\}. \tag{20}$$

A check is performed when $(n - n_{init})\text{mod}(\text{N} + 1) = \text{N}$. This is the stage when a purely local search is performed, so it is important to make sure that the minimizer of s_n is not one of the interpolation points or too close to one. The test used is

$$f_{min} - \min_{y \in \Omega} s_n(y) \leq 10^{-4} \max\{1, |f_{min}|\}, \tag{21}$$

where f_{min} is the best function value found so far, i.e. $\min_i f(x_i)$, $i = 1, \ldots, n$. If (21) is true, then

$$f_n^* = \min_{y \in \Omega} s_n(y) - 10^{-2} \max\{1, |f_{min}|\}, \tag{22}$$

otherwise f_n^* is set to 0.

2.3. A Compact RBF Algorithm Description. In the previous sections the basic RBF algorithm implemented in our Matlab routine *rbfSolve* were described in detail. We now summarize the RBF algorithm in the compact description below.

- Choose n initial points $X = \{x_i, i = 1, ..., n\}$.
  Use 2^d corner points or at least $d + 1$ points.
- Compute $f_i = f(x_i), i = 1, ..., n$, set $n_{init} = n$.
- Compute Radial basis interpolation minimizing semi-norm and interpolating points

$$\begin{aligned} s_n = arg\min_s \quad & < s, s > \\ s/t \qquad\qquad & s(x_i) = f(x_i), \quad i = 1, ..., n \end{aligned} \tag{23}$$

  The optimal solution is the solution to (4).
- *While* $n < MaxFuncEval$
  Repeat Cycle $k = 0, ..., N$ (Local and global search, $N = 5$)
  1. If $k = 0$ solve the minimization problem $\min_{y \in \Omega} s_n(y)$.
  2. Compute f_n^* in (18) dependent on position k in the cycle.
  3. $x_{new} = arg\min_y -\log h_n(y)$, $h_n(y)$ defined in (14).

4. *If* new point x_{new} *acceptable* (Not too close to $x_1, \ldots, x_n$), $n = n + 1$; $x_{new} = x_n$; $f_n = f(x_{new})$; $X = [X, x_{new}]$; end
5. $f_{best} = \min f(x_i), x_i \in X$;
6. $x_{best} = arg\min f(x_i), x_i \in X$;
7. Update the matrix factorizations of Φ and P and find new interpolant s_n by solving (12).

- *End of while*

One problem is how to choose the points $x_1, \ldots, x_{n_{init}}$ to include in the initial set. We only consider box constrained problems, and choose the corners of the box as initial points, i.e. $n_{init} = 2^d$. Starting with other points is likely to lead to the corners during the iterations anyway. Having a "good" point beforehand, one can include it in the initial set.

The subproblem

$$\min_{y \in \Omega} \quad s_n(y) \ , \tag{24}$$

is itself a problem which could have more than one local minima. To solve (24) (at least approximately), we start from the interpolation point with the least function value, i.e. argminf($\mathrm{x_i}$), $i = 1, \ldots, n$, and perform a local search. In many cases this leads to the minimum of s_n. Of course, there is no guarantee that it does. We use analytical expressions for the derivatives of s_n and perform the local optimization using *ucSolve* TOMLAB running the inverse BFGS algorithm as described in Holmström and Björkman (1999). As an alternative we use the *NPSOL* solver by Gill, Murray, Saunders and Wright (1998) using the MEX-file interface that is part of TOMLAB.

To minimize $-\log h_n(y)$ we use our Matlab routine *glbSolve* implementing the DIRECT algorithm (see Section 3.1). We run *glbSolve* for 500 function evaluations and choose x_{n+1} as the best point found by *glbSolve*. When $(n - n_{init})\mathrm{mod}(\mathrm{N}+1) = \mathrm{N}$ (when a purely local search is performed) and the minimizer of s_n is not too close to any of the interpolation points, i.e. (21) is not true, *glbSolve* is not used to minimize $g_n(y)$ or $f^*(y)$. Instead, we choose the minimizer of (24) as the new point x_{n+1}. The TOMLAB routine *AppRowQR* is used to update the QR decomposition.

Our experience so far with the RBF algorithm shows that the minimum is sometimes very sensitive for the scaling of the box constraints. To overcome this problem we transform the search space to the unit hypercube.

In our implementation it is possible to **restart** the optimization with the final status of all parameters from the previous run.

3. OTHER GLOBAL OPTIMIZATION ALGORITHMS

In the following sections, Section 3.1 - 3.3, short descriptions of the DIRECT, constrained DIRECT and EGO algorithms are given.

3.1. DIRECT. DIRECT is an algorithm developed by Jones, Perttunen and Stuckman (1993) for finding the global minimum of a multi-variate function subject to simple bounds, using no derivative information. The algorithm is a modification of the standard Lipschitzian approach that eliminates the need to specify a Lipschitz constant. The idea is to carry out simultaneous searches using all possible constants from zero to infinity. In Jones et al. (1993) they introduce a different way of looking at the Lipschitz constant. The Lipschitz constant is viewed as a weighting parameter that indicate how much emphasis to place on global versus local search. In standard Lipschitzian methods, this constant is usually large because it must be equal to or exceed the maximum rate of change of the objective function. As a result, these methods place a high emphasis on global search, which leads to slow convergence. In contrast, the DIRECT algorithm carries out simultaneous searches using all possible constants, and therefore operates on both the global and local level. DIRECT deals with problems of the form

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.t.} & x_L \ \leq \ x \ \leq \ x_U \ , \end{array} \tag{25}$$

where $f \in \mathbf{R}$ and $x, x_L, x_U \in \mathbf{R^d}$. The finite box defined by the bound constraints is normalized to $[0,1]^d$. and partitioned into smaller boxes. Then it is true that the side lengths of the boxes are 3^{-k} for some $k \in \mathbf{N}$. It is guaranteed to converge to the global optimal function value, if the objective function f is continuous or at least continuous in the neighborhood of a global optimum. This could be guaranteed since, as the number of iterations goes to infinity, the set of points sampled by DIRECT form a dense subset of the unit hypercube. In other words, given any point x in the unit hypercube and any $\delta > 0$, DIRECT will eventually sample a point (compute the objective function) within a distance δ of x. However, the use of the midpoint in each box leads to the disadvantage that the boundary can only be reached in the limit, and the convergence will be slow when the minimizer lies at the boundary.

We have implemented the DIRECT algorithm in Matlab, and in Björkman and Holmström (1999), we discuss the implementation details of our Matlab implementation. The efficiency of the implementation is analyzed by a comparison to the results of Jones's implementation on nine standard test problems for box-bounded global optimization. In fifteen out of eighteen runs the results were in favor of our implementation.

One version of the DIRECT code is available as the Matlab routine *gblSolve* for download at http://www.ima.mdh.se/tom, the home page of the Applied Optimization and Modeling group. It is free for academic use. A faster version, *glbSolve*, is part of the TOMLAB v3.0 optimization environment described in Holmström (2001).

3.2. Constrained DIRECT. Jones (2001) presents an extension of the DIRECT algorithm which handles nonlinear and integer constraints, a global mixed-integer nonlinear programming problem of the form

$$\begin{array}{lllllll} \min\limits_{x} & f(x) & & & & & \\ & x_L & \leq & x & \leq & x_U & \\ \text{s.t.} & c_L & \leq & c(x) & \leq & c_U & , \\ & x_i \in I & & \text{integer} & & & \end{array} \tag{26}$$

where $f \in \mathbf{R}$, $x, x_L, x_U \in \mathbf{R^d}$, $c, c_L, c_U \in \mathbf{R^m}$ and I is the index set for the integer variables. The constrained version of DIRECT does not explicitly handle equality constraints and it works best when the integer variables describe an ordered quantity. It is less effective when the integer variables are categorical. If no constraints are present, this constrained version of DIRECT reduces to the box-bounded version, with some minor differences.

We have implemented the constrained version of the DIRECT algorithm in Matlab with a slightly more general problem formulation that explicitly handles linear constraints as

$$\begin{array}{lllllllll} \min\limits_{x} & f(x) & & & & & & & \\ & -\infty < & x_L & \leq & x & \leq & x_U & < \infty & \\ s/t & & b_L & \leq & Ax & \leq & b_U & & \\ & & c_L & \leq & c(x) & \leq & c_U, & x_j \in \mathbf{N} & \forall \mathbf{j} \in \mathrm{I}, \end{array} \tag{27}$$

where $x, x_L, x_U \in \mathbf{R^n}$, $f(x) \in \mathbf{R}$, $A \in \mathbf{R^{m_1 \times n}}$, $b_L, b_U \in \mathbf{R^{m_1}}$ and $c_L, c(x), c_U \in \mathbf{R^{m_2}}$. The variables $x \in I$, the index subset of $1, ..., n$, are restricted to be integers. Our constrained DIRECT code is available as the Matlab routine *glcSolve* in TOMLAB. Feedback from TOMLAB users and tests we have ran show that the solver works well.

3.3. EGO. The EGO (Efficient Global Optimization) algorithm by Jones, Schonlau and Welch (1998) is also an interesting algorithm, which like the RBF algorithm belongs the class of Response Surface Methods. These models first fit a model function to data collected by evaluating the objective function at a number of initial points. Then a utility function is used to determine the new point where the objective function should be evaluated. In EGO, a nonlinear stochastic process model,

the DACE (Design and Analysis of Computer Experiments) predictor, is fit by use of nonlinear regression unlike the RBF algorithm, where linear regression is used. Then, EGO balances between global and local search by choosing the new point where the objective function should be evaluated as the one which maximizes an expected improvement utility function.

We have implemented the EGO algorithm in Matlab and it is available as the Matlab routine *ego* in TOMLAB.

4. FINANCIAL APPLICATIONS

Trading strategies present an interesting and challenging application for global optimization without derivatives. The object function can be defined as the achieved profit when applying a trading system on historical data. It is clear that neither the object function nor the derivatives are available in analytical form. Furthermore, the function value is often the result of a simulated trading with many years of data, and takes in the order of minutes to compute with an ordinary desk top computer. The need for optimization routines suitable for costly object functions is therefore clear. In this section we will use the previously described DIRECT algorithm to optimize simple trading rules parameterized with three and four parameters. A big problem with such optimization is the estimation of out-of-sample performance for the obtained trading rules. In particular, it is very easy to jump into conclusions regarding trading rules that exhibit extremely profitable behavior, when tested on historical data. These misjudgments are often caused by the rules covering too few examples in the examined data. We will approach this problem with nonconvex global optimization of trading rules with a constraint added in the problem formulation. The effect is a regularization, where solutions covering too few examples are rejected. The modeling is performed with a sliding-window technique and generates different parameters for the optimized trading rules in each time window. For more details and results, refer to Hellström (2000b) where another data set is analyzed with the same approach.

4.1. Trading Rules. A general way to formulate strategies for stock trading is to define a trading rule as a time series $T(t)$ such as

$$T(t) = \begin{cases} \text{Buy} & : \text{ if } g(t) = 1 \\ \text{Sell} & : \text{ if } g(t) = -1 \\ \text{Do nothing} & : \text{ if } g(t) = 0 \end{cases} \tag{28}$$

where g is a function of the previous stock prices $Close$:

$$g : \{Close(t), Close(t-1), ..., Close(t-k)\} \rightarrow \{-1, 0, 1\}. \tag{29}$$

Trading rule (28) is designed to serve as decision support in actual stock trading, as indicated by the labels *Buy, Sell,* and *Do Nothing.* Function g determines the type of the trading rule. By extending expression (29) with the input variables *High* (highest-paid price), *Low* (lowest-paid price), *Open* (first price) and *Volume* (number of traded stocks), most standard technical indicators, such as the *Stochastic Oscillator*, the *Relative Strength Index (RSI), Moving Average Convergence/Divergence (MACD)* etc. (Kaufman (1998)), can be described in this fashion. Quite often the buy and sell decisions are controlled by separate expressions and the trading rules are then denoted *Buy rule* and *Sell rule* respectively. Hereinafter we use the notation g_s to denote a trading rule applied to one specific stock s.

Function g is normally parameterized with a few parameters that can to be determined by numerical optimization. In this paper, three trading rules for generating *Buy* signals, are used to demonstrate the techniques with constrained optimization. All three are based on standard technical indicators, well-known by the trading community. For a thorough introduction to the subject, refer to Kaufman (1998). However, the standard indicators have been augmented with a term that includes the traded volume. This too is in accordance with common practice among traders. We include the traded volume as a term in all our technical trading rules. To facilitate a uniform modeling for *all* stocks in the market, a normalized measure has to be defined.

4.1.1. Gaussian Volume. The Gaussian volume $V_n(t)$ is a transformation of the traded volume (number of stocks) $V(t)$ defined as

$$V_n(t) = (V(t) - m_V(t))/\sigma_V(t), \tag{30}$$

where the mean $m_V(t)$ and the standard deviation $\sigma_V(t)$ for the volume are computed in an n days long window up to time t. V_n expresses the number of standard deviations, by which the volume differs from its running mean. The normalization makes it possible to compare values of V_n for different stocks and also for different times. In this paper the Gaussian volume V_{10} is used and is denoted by $gvol10$, since this is the name of the ASTA (Hellström (2000a)) implementation of the function.

4.1.2. Crossing Moving-Average. This is an implementation of a common trading rule based on two moving averages of different length. The trading rule signals *Buy*, if a short moving-average mav_{x_1} crosses a long moving-average mav_{x_2} from below. A *Sell* signal is issued when mav_{x_1} crosses the mav_{x_2} from above. In this paper we define the *Buy* rule mav as

$$mav(x_1, x_2, x_3) = Mavx(x_1, x_2) \wedge gvol10 > x_3, \tag{31}$$

where

$$Mavx(x_1, x_2) = mav_{x_1}(t) > mav_{x_2}(t) \wedge mav_{x_1}(t-1) \leq mav_{x_2}(t-1) \tag{32}$$

and mav_{x_1} is a x_1-day moving average of the stock prices up to time t.

4.1.3. Trading Channel Breakout. The main part of this trading rule is what is popularly known as *Bollinger Bands* (see e.g. page 91 in Kaufman (1998)). The complete trading rule is defined as

$$break(x_1, x_2, x_3) = breakout(x_1, x_2) \wedge gvol10 > x_3, \tag{33}$$

where the *breakout* function is defined as

$$\begin{aligned} breakout(x_1, x_2) = \quad & Close(t) > (mav_{x_1}(t) + x_2 \cdot \sigma_{x_1}(t)) \quad \wedge \\ & Close(t-1) \leq (mav_{x_1}(t) + x_2 \cdot \sigma_{x_1}(t)) \end{aligned} \tag{34}$$

and $mav_{x_1}(t)$ is is a x_1-day long moving average of the stock prices up to time t. Function $\sigma_{x_1}(t)$ computes the standard deviation of the $Close$ up to time t. The idea is to define an upper boundary for a trading channel and generate a *Buy* signal when the $Close$ penetrates this boundary from below. This upper boundary is defined as the sum of a moving average mav_{x_1} and x_2 times an estimate of the standard deviation σ_{x_1}.

4.1.4. Level of Resistance. The trading rule *Level of Resistance*, in this paper denoted *resist*, is based on a technique commonly executed by manual inspection of the stock charts. The general idea is to identify *peaks* in a window backwards, where the *Close* price is roughly the same. When such peaks are found, a *Buy* signal is generated if the *Close* price crosses from below the level for the found peaks. We define the trading rule *resist* as

$$resist(x_1, x_2, x_3, x_4) = xresist(x_1, x_2, x_3) \wedge gvol10 > x_4 \tag{35}$$

where

$$xresist(x_1, x_2, x_3) = Close(t) > plevel \wedge Close(t-1) \leq plevel \tag{36}$$

and

$$plevel = \begin{cases} l: & \text{if at least } x_2 \text{ peaks in } Close \text{ that differs by less than } x_3\% \text{ can} \\ & \text{be identified at level } l \text{ in an } x_1\text{-day long window backwards.} \\ 0: & \text{otherwise} \end{cases}$$

4.2. Performance Evaluation. Performance evaluation for a trading rule is needed in two stages of the process. First, in the optimization phase, when parameters for the trading rule have to be determined. The second stage is when the final trading rule is evaluated on the test data set previously unseen. For more information about performance evaluation of trading algorithm refer to Hellström (1999b) or Refenes (1995). Trading-rule-based methods are normally evaluated by trading simulation, where the trading rule controls the buying and selling of one or several stocks over a period of time. Examples of this approach in conjunction with optimization can be found in Hellström and Holmström (1999). However, it is also possible to evaluate a trading rule with a fixed prediction horizon, of which the advantage is that all situations where the trading rules fire (i.e.: $T(t) \neq$ *Do Nothing* in (28)) are evaluated. When performing a trading simulation, this is normally not the case, since the simulated trader is bounded by the real-world constraint of a limited amount of money. This prevents the trader from executing some of the *Buy* signals that the trading rules produce. Since the fraction of left-out trades can be as high as 80-90%, a scheme with randomization and repeated simulations is normally required to produce reliable performance measures for the trading rules. Therefore in this study we evaluate trading rules at fixed prediction horizons. The measure of interest is the correctness of the sign of the price change from the time of the prediction to 5 days ahead. This way of evaluating predictions has gained increased interest in recent years as an alternative to the more conventional way of minimizing the error of the level prediction. A comparative study of sign and level methods can be found in Leung, Daouk and Chen (2000) where the presented experiments suggest that methods predicting the sign provide higher profits than methods predicting the level for a number of investigated stock indexes.

For a time period $[1, ..., T]$ and a set of stocks S, the h-day positive hit rate for a *Buy* rule g is defined as

$$H_g^+ = \frac{card\{(t,s)|R_h^s(t+h) > 0, g_s(t) = 1, 1 \leq t \leq T-h, s \in S\}}{card\,\{(t,s)|R_h^s(t+h) \neq 0, g_s(t) = 1, 1 \leq t \leq T-h, s \in S\}} \tag{37}$$

where g_s is the function specifying the trading rule as described in (28). The return R_h^s is the relative change in price and is defined as

$$R_h^s(t) = 100 \cdot \frac{Close_s(t) - Close_s(t-h)}{Close_s(t-h)} \tag{38}$$

where $Close_s(t)$ is the price for a stock s at the end of day t. The hit rate H_g^+ for a *Buy* rule g indicates how often a *Buy* signal is followed by a true increase in the stock price. The hit rate H_g^- for a *Sell* rule is defined correspondingly but with returns $R_h < 0$.

4.3. Optimization. The function g that defines the trading rule is normally parameterized with a few parameters x that have to be determined in order to maximize the chosen performance measure on the historical data. To express this parameterization, the notation $g[x]$ will be used.

One big problem about trading rules in general and optimizing them in particular is the statistical significance of the estimated performance. The trading rule (28) normally issues *Buy* or *Sell* signals only for a minor part of the points in the time series. This results in low levels of significance for the produced performance measures. It is often easy to find a trading rule that historically outperforms any benchmark, as long as it does not have to produce more than a few signals per year. However, the performance on previously unseen data is most often very bad in these situations. We therefore formulate a constrained optimization problem for a *Buy rule* g (*Sell* rules can be treated in a similar way) as

$$\begin{aligned} & arg \max_{x} \quad H^{+}_{g[x]} \\ & \text{s.t.} \\ & card\{(s,t)|g_s[x](t) = 1, t \leq T-h, s \in S\} \geq N_0, \\ & x_L \leq x \leq x_H \end{aligned} \tag{39}$$

where x_L and x_H are lower and upper bounds for the unknown parameters and the other constraint is the total number of *Buy* signals. The hit rate $H^{+}_{g[x]}$ is given by definition (37). With the introduced notation, $g_s[x](t)$ denotes the trading rule g parameterized with parameters x and applied to stock s for time t. The optimization routine performs simulations up to time T to compute the hit rate and number of trading signals for a given $g[x]$. The purpose is to maximize the hit rate $H^{+}_{g[x]}$ by altering the variables x that parameterize the function g. The final performance measure is the out-of-sample hit rate $H^{+}_{g[x]}$, computed for time $t > T$ with the optimal estimated parameters x.

Using a 'hard' constraint in the optimization problem in (39) leads to a non-smooth problem. Because of the uncertainty in the choice of the 'most' suitable value of N_0, it is reasonable to reformulate the problem using a 'soft' constraint approach that generates a smooth problem. The approach uses a sigmoid function to smoothly model the behavior of the added constraint and is inspired by the membership-function concept used in fuzzy logic (see e.g. Klir and Yuan (1995)). The new problem formulation, in which the objective function in (39) is weighted

with the output of a sigmoid, is

$$\begin{array}{l} arg \ \max\limits_{x} \ \ H^+_{g[x]} \cdot \ \ support_{N_0}\left(card\{(s,t)|g_s[x](t)=1, t \leq T-h, s \in S\}\right) \\ \text{s.t.} \\ x_L \leq x \leq x_H \end{array} \tag{40}$$

where $support_{N_0}$ is given by the sigmoid function

$$support_{N_0}(n) = \frac{1}{1+e^{-\alpha(n-\beta)}}. \tag{41}$$

The parameters α and β are computed to fulfill the equations $support_{N_0}(N_0) = 0.99$ and $support_{N_0}(N_0 \cdot 0.5) = 0.01$. This ensures a smooth penalty for trading rules that generate less than N_0 trading signals. If more than N_0 trading signals are generated, the $support_{N_0}$ function returns essentially 1 and hence does not affect the search for an optimal function g. The constraint acts like a regularizer, since the search space for the function g is reduced by requiring a minimum number of trading signals. This improves the statistical significance of the estimated performance and the generalizability of the found solution (i.e. the achieved hit rate on previously unseen data). The choice of the cut-off value N_0 is a trade-off between the achieved hit rate on the training data and the generalizability.

The optimization problem (40) is a box-bounded nonconvex global optimization problem. It is suitable to use derivative free methods, since no analytical expressions for $g[x]$ and $H^+_{g[x]}$ are available. In our tests we are using the *DIRECT* algorithm described in Section 3.1.

4.4. Experimental Design. Technical analysis of stocks is normally based on the premise that the market's behavior does not change much over time. While future movements in stock prices are never copies of the past, the market's way of responding to new situations is assumed to be similar to the way it has handled them in the past (Gencay, Stengos (1998)). Since this is not necessarily a valid assumption the optimization will be performed with a sliding window technique.

The hit rate H^+_g in the object function (39) is computed using the non-interactive version of the ASTA system, which performs market simulations of trading rules given in symbolic form. The ASTA system is written in Matlab and has a large number of technical indicators implemented. The system is thoroughly described in Hellström (2000a). Examples of usage is found in Hellström (1999a).

The test is utilizing a sliding-window technique with a 2-year training data period followed by a 1-year test period. The starting point of the training period is

moved between 1990 and 1995 in 1-year steps. This results in six separate modeling/test periods. The presented performance is the total for the six test periods (1992,...,1997). The purpose of using sliding windows in the optimization is twofold. First, the stability in the performance can be studied since we get six performance measures instead of one. Second, the trading rules are allowed to adapt to time-varying market conditions such as volatility, long-term trends etc. Eighty of the largest Swedish stocks are included in the test, which provides a total number of data points of around 111000 (not all stocks have data for the entire period). The trading rules select a small fraction of these points (date and stock) as suggested opportunities to buy stocks.

The results for 5-day prediction horizon are presented in Table 1 , with positive hit rate H^+ and number of points N where a trading signal is generated. Separate measures for training data and test data are presented in the columns labeled H_{tr}, N_{tr}, H_{te} and N_{te}. The rightmost column shows the lower 90% confidence limit[1] for the hit rate H_{te}. The cut-off value N_0, used for the regularization, is set to 100. Each of the eight rows represents a prediction method. The first three rows show the results for the trading rules $resist_{100}$, $break_{100}$ and mav_{100} described in Section 4.1. The parameters $x_1, x_2, ...$ are optimized for best performance on the training data, using the regularization described above ($N_0 = 100$). The following three rows show the same trading rules as above, but with no regularization to control the number of generated trading signals ($N_0 = 1$): $resist_1$, $break_1$ and mav_1. Performance for the benchmark methods Naive-5_+ and Naive-ε are also reported. The Naive-5_+ predictor of the returns for a stock s asserts today's return $R_5^s(t)$ (price increase since $t-5$) as the prediction of $R_5^s(t+5)$. The Naive-ε prediction of prices for a stock s asserts today's price $Close_s(t)$ as the best estimate of $Close_s(t+5)$. To enable comparison of hit rate predictions, the naive predictor is modified so the best estimate of today's price is assumed to be $Close_s(t+5) + \varepsilon$. This means that the predicted returns R_5^s are always positive. This naive predictor is denoted below *Naive-ϵ*.

The computed optimal parameters for a specific *Buy* rule vary for the six test periods. The ones computed for test period 1992 are presented in Table 2.

4.5. Results. As expected, the optimized trading rules perform much better for the training data than for the test data. This effect is much more emphasized for the non-regularized trading rules than for the regularized ones. The difference can be understood as over-fitting of data that can be controlled by the regularization. The out-of-sample hit rates H_{te} show no systematic difference between the two kinds of predictors. The small observed differences should be seen rather as

[1]The lower boundary for a 90% confidence interval.

Table 1: Hit rate and number of selected points for optimized trading rules. Totals from 6 1-year test periods (1992-1997) with the preceding 2 years for training. 5 days prediction horizon.

Method	H_{tr}	N_{tr}	H_{te}	N_{te}	**90%−low** H_{te}
$resist_{100}$	65.82	746	63.44	454	59.55
$break_{100}$	63.07	1075	55.64	692	52.44
mav_{100}	61.96	715	50.40	371	46.01
$resist_1$	76.84	177	59.83	117	51.82
$break_1$	64.89	786	52.04	417	47.89
mav_1	71.55	239	53.75	160	46.94
$Naive - e$	48.33	196470	50.06	102651	49.80
$Naive - 5_+$	48.83	84054	49.53	46202	49.14

Table 2: Optimized trading rules for 1992. 5 days prediction horizon.

Method	**Optimized expression**
$resist_{100}$	$xresist(84, 4, 4.06) \wedge gvol10 > 0.67$
$break_{100}$	$breakout(38, 1.5) > 0 \wedge gvol10 > 2.94$
mav_{100}	$Mavx(5, 83) \wedge gvol10 > 0.33$
$resist_1$	$xresist(42, 6, 1.83) \wedge gvol10 > 3.78$
$break_1$	$breakout(117, 2.5) > 0 \wedge gvol10 > 2.5$
mav_1	$Mavx(11, 112) \wedge gvol10 > 1.3$

stochastic fluctuations caused by the low accuracy in the estimation of the hit rates for the non-regularized trading rules. The lower 90% confidence limit reveals how uncertain the hit rates H_{te} are for these rules. This uncertainty comes from the low number of predictions generated. None of the non-regularized trading rules can be said to significantly outperform the benchmark predictors, while the regularized *resist* predictor has 63% hit rate, which is significantly higher than the benchmarks.

4.6. Stability of the Found Optima. The experimental setup with sliding windows gives a stable evaluation of the trading rules. In this section an additional test of the stability and relevance of the optimized trading rules is performed. In Table 3, the three regularized trading rules optimized with data from 1990-1991 are applied not only for 1992 but also for the following years up to 1997. This means that the optimized rules are regarded as globally valid instead of valid only for the year following the optimization period. Performance for the benchmark predictors are also presented for comparison. The results show that the average hit rate for the trading rules for the six years, is clearly lower than the one achieved by the sliding-window approach, as shown in Table 1 (the relevant value for comparison is shown in column H_{te}). Furthermore, the individual results for each year show no clear tendency and can be regarded as random variations. These observations give further credibility to the sliding-window results and show that the optimizations really are catching patterns and regularities in the data and not only spurious local optima in random and noisy object functions.

4.7. Summary of the Results. The constrained optimization that avoids too few selected points is essential, both for practical reasons (since we want to get assistance in our buy and sell decisions more than a few times per year), and for a reasonably safe estimate of the expected hit rate out-of-sample. Without safeguarding against too few points, the found optima gives excellent performance on the training data, but no significant improvement relative to pure chance on the test data. Furthermore, the results show that the high hit rate achieved with the *resist* trading rule, to a large extent is a result of the adaptive modeling with sliding windows.

5. CONCLUSIONS AND FURTHER WORK

Global optimization techniques can be used to improve the performance of trading algorithms and time series predictions. When the problems are costly to compute, the use of surrogate modeling techniques like the RBF algorithm is promising and should be further exploited.

In the RBF algorithm, work is needed to avoid too large condition number on the interpolation matrix for increasing number of sampled points. Also better choices

Table 3: Hit rate for trading rules optimized with data from 1990-1991. 5-day prediction horizon.

Method	92	93	94	95	96	97	Average	H_{te}
$resist_{100}$	55.1	66.9	59.1	52.8	57.4	56.6	57.6	63.44
$break_{100}$	54.8	65.1	48.3	46.5	55.8	52.1	54.5	55.64
mav_{100}	45.8	59.8	39.3	44.3	58.7	57.1	50.3	50.40
$Naive - e$	44.1	55.6	46.7	47.4	53.8	52.1	50.1	50.06
$Naive - 5_+$	46.7	56.2	45.1	46.0	52.2	49.3	49.5	49.53

of initial set must be investigated, when n is not small. Our goal is to implement a robust and fast RBF algorithm in both Matlab and Fortran.

We will further test the use of surrogate model techniques for the optimization of trading algorithms and time series model predictions. It is interesting to use the regularization techniques described for more advanced prediction methods, e.g. EXPAR (Exponential Autoregressive) models and for high frequency data.

BIBLIOGRAPHY

1. Björkman, M. and Holmström, K., *Global Optimization with the DIRECT Algorithm in Matlab*, Advanced Modeling and Optimization, **1**(2), (1999), 17-37.

2. Björkman, M. and Holmström, K., *Global Optimization of Costly Nonconvex Functions using Radial Basis Functions,* Optimization and Engineering, **2**, (2001), To be published.

3. Blume, L., Easley, D. and O'Hara, M., *Market statistics and technical analysis: The role of volume,* Journal of Finance, **49**, (1994), 153–181.

4. Brock, W., Lakonishok, J. and LeBaron, B., *Simple technical rules and the stochastic propertites of stock returns,* Journal of Finance, **47**, (1992), 1731–1764.

5. Campbell, J. Y., Grossman, S. J. and Wang, J., *Trading volume and serial correlation in stock returns,* Quarterly Journal of Economics, **108**, (1993), 905–940.

6. Gencay, R. and Stengos, T., *Moving average rules, volume and the predictability of security returns with feedforward networks,* Journal of Forecasting, (1998), 401–414.

7. Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H., *User's guide for NPSOL 5.0: A Fortran package for nonlinear programming,* Technical Report SOL 86-2, Revised July 30, 1998, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California 94305-4022, (1998).

8. Gutmann, H-M., *A radial basis function method for global optimization,* Technical Report DAMTP 1999/NA22, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, (1999).

9. Gutmann, H-M., *On the semi-norm of radial basis function interpolants.* , Technical Report DAMTP 2000/NA04, Department of Applied Mathematics, and Theoretical Physics, University of Cambridge, England, (2000).

10. Hellström, T., *A Random Walk through the Stock Market*, Licentiate thesis, Umeå University, Umeå Sweden, (1998).

11. Hellström, T., *ASTA - a Tool for Development of Stock Prediction Algortihms,* Theory of Stochastic Processes, **5**(21), (1999a), 22–32.

12. Hellström, T., *Data Snooping in the Stock Market,* Theory of Stochastic Processes, **5**(21), (1999b), 33–50.

13. Hellström, T., *ASTA - User's Reference Guide,* Technical Report UMINF-00.16 ISSN-0348-0542, Department of Computing Science Umeå University, Umeå Sweden, (2000a).

14. Hellström, T., *Optimization of Trading Rules with a Penalty Term for Increased Risk-Adjusted Performance*, Advanced Modeling and Optimization, (2000b), **2**(3).

15. Hellström, T. and Holmström, K., *Parameter Tuning in Trading Algorithms using ASTA,* In Abu-Mostafa, Y. S. , LeBaron, B., Lo, A. W. and Weigend, A. S., (ed), Computational Finance 1999, Cambridge, MA., MIT Press, (1999).

16. Holmström, K., *The TOMLAB Optimization Environment in Matlab*, Advanced Modeling and Optimization, **1**(1), (1999a), 47–69.

17. Holmström, K., *New Optimization Algorithms and Software.,* Theory of Stochastic Processes, **5(21)**(1-2), (1999b), 55–63.

18. Holmström, K., *The TOMLAB v2.0 Optimization Environment*, In K. Holmström and E. Dotzauer, editors, Proceedings from the 6th Meeting of the Nordic Section of the Mathematical Programming Society, Västerås, 1999. Department of Mathematics and Physics, Mälardalen University, Sweden, (1999c).

19. Holmström, K., *TOMLAB v3.0 User's Guide,* Technical Report IMa-TOM-2001-01, Department of Mathematics and Physics, Mälardalen University, Sweden, (2001).

20. Holmström, K. and Björkman, M., *The TOMLAB NLPLIB Toolbox for Nonlinear Programming*, Advanced Modeling and Optimization, **1**(1), (1999), 70–86.

21. Iglehart, D. L. and Voessner, S., *Optimization of a trading system using global search techniques and local optimization,* Journal of Computational Intelligence in Finance, **6**, (1998), 36–46.

22. Jones, D. R., *DIRECT*, Encyclopedia of Optimization, (2001). To be published.

23. Jones, D. R., Perttunen, C. D. and Stuckman, B. E., *Lipschitzian optimization without the Lipschitz constant*, Journal of Optimization Theory and Applications, **79**(1), (1993), 157–181.

24. Jones, D. R., Schonlau, M. and Welch, W. J., *Efficient global optimization of expensive Black-Box functions*, Journal of Global Optimization, **13** (1998), 455–492.

25. Karpov, J. M., *The relation between price changes and traded volume,* Journal of Financial and Quantitative Analysis, **22**, (1987), 109–126.

26. Kaufman, P. J., *Trading Systems and Methods*, John Wiley and Sons, New York, (1998).

27. Klir, G.J. and Yuan, B., *Fuzzy Sets and Fuzzy Logic. Theory and Applications*, Prentice-Hall, Inc, New Jersey, USA, (1995).

28. Kuo, G. W., *Some exact results for moving-average trading rules with applications to UK indices,* In E. Acar and S. Satchell, editors, Advanced Trading Rules, Butterworth Heinemann, Oxford,(1998), 81–102.

29. Leung, M. T., Daouk, H. and Chen, A.-S., *Forecasting stock indices: a comparison of classification and level estimation methods*, International Journal of Forecasting, **16**:1,(2000), 73–190.

30. Levich, R. M. and Thomas, L. R., *The significance of technical trading-rule profits in the foreign exchange market: a bootstrap approach,* Journal of International Money and Finance, **12**, (1993), 451–474.

31. Powell, M. J. D., *The theory of radial basis function approximation in 1990,* In W.A. Light, editor, Advances in Numerical Analysis, Volume 2, Wavelets, Subdivision Algorithms and Radial Basis Functions, Oxford University Press, (1992), 105–210.

32. Powell, M. J. D., *Recent research at Cambridge on radial basis functions,* In M. D. Buhmann, M. Felten, D. Mache, and M. W. Müller, editors, New Developments in Approximation Theory, Birkhäuser, Basel, (1999), 215–232.

33. Refenes, A.-P., *Testing strategies and metrics*, In Refenes, A.-P., editor, Neural Networks in the Capital Markets, John Wiley & Sons, Chichester, England, (1995), 67–76.

34. Zar, J. H., *Biostatistical Analysis*, Prentice-Hall, Inc, New Jersey, USA, (1999).

Department of Computing Science,
Umeå University, SE-901 87 Umeå, Sweden
E-mail: thomash@cs.umu.se
http://www.cs.umu.se/ thomash

Center for Mathematical Modeling, Department of Mathematics and Physics,
Mälardalen University, P.O. Box 883, SE-721 23 Västerås, Sweden
E-mail: hkh@mdh.se
http://www.ima.mdh.se/tom