1-1-1995

Calibration of a conceptual rainfall-runoff model using simulated annealing

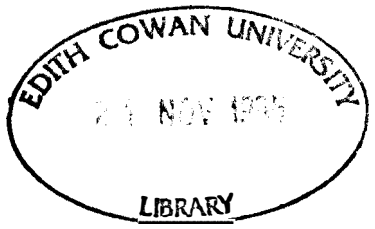
Neil R. Sumner
Edith Cowan University

Edith Cowan University

Copyright Warning

Calibration of a Conceptual Rainfall-Runoff Model using Simulated Annealing

by



Neil R. Sumner

BAppSc, GDComp, PGDCompSc

A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of

Master of Science (Mathematics and Planning)

at the Faculty of Science and Technology, Edith Cowan University

Date Submitted: June 28, 1995

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

Abstract

Simulated annealing (Kirkpatrick et al., 1983) is used to estimate the parameters of a mathematical model that predicts the water yield from a catchment. The calibration problem involves finding the global minimum of a multivariate function that has many extraneous local minima, a situation in which conventional optimisation methods are ineffective. The objective function which quantifies discrepancies between the computed and observed streamflows must be carefully selected to satisfy the least squares assumptions.

Several published simulated annealing algorithms have been implemented, tested and evaluated using standard test functions. Appropriate cooling schedules are found for each algorithm and test function investigated. The number of function evaluations required to find the minimum is compared to published results for the test functions using either simulated annealing and other global optimisation methods. A new simulated annealing algorithm based on the Hooke and Jeeves (1961) pattern search method is developed and compared with existing algorithms from the literature.

Simulated annealing is used to calibrate a conceptual rainfall-runoff model (a modification of Boughton's SFB model) by fitting modelled monthly runoff to historical data. The model estimates runoff from daily rainfall data and potential evapotranspiration estimates for a catchment. The 25 catchments used for the study cover a wide range of climatic types including humid tropical, humid subtropical, arid, semiarid, Mediterranean and marine west-coast. A parsimonious approach was adopted where only the sensitive model parameters for a catchment were fitted. The objective function was transformed to satisfy the least squares assumptions. A parameter covariance matrix and standard errors were calculated to help identify redundant parameters.

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

[redacted]

Date November 8, 1995

Acknowledgements

I am indebted to many people for their help and encouragement during the course of this study. I would like to thank two of the busiest people that I know for finding the time to supervise this thesis. Associate Professor James Cross for his guidance and uncanny knack of knowing when to let me run on my own and when to stay in touch. Dr Bryson Bates has ignited my interest in simulated annealing and given me an appreciation of surface water hydrology. Dr Bates also completed most of the programming for the final version of the optimisation program used to calibrate the rainfall-runoff model.

I would like to thank all the staff in the Mathematics Department at Edith Cowan University for their encouragement and support during a difficult stage of my life. It is greatly appreciated.

Thanks are due to the CSIRO Division of Water Resources for giving me almost unlimited time on their computing systems. The other users of the systems may now breathe a big sigh of relief.

I would like to thank Francis Chiew and Tom McMahon Department of Civil and Environmental Engineering, University of Melbourne, for collating and making the streamflow and rainfall data available for this study. Climate station records from the Australian Bureau of Meteorology were used to compute catchment rainfall and potential evapotranspiration. The streamflow data was collected by the respective Australian State and Territory water agencies. This work contributes to the CSIRO Climate Change Research Program and is partially funded through Australia's National Greenhouse Research Program.

To my family Debbie, Jessica, Ryan and Tamara for your patience and understanding, I can but offer my heartfelt thanks.

Table of Contents

1. Introduction

A conceptual rainfall-runoff model (CRRM) was required to produce streamflow estimates for benchmark catchments as part of a study on the impacts of climate change on Australia's water resources. Once calibrated, the model would be used to generate streamflow estimates using rainfall and potential evapotranspiration data from a general circulation model and a stochastic weather generator. The SFB model (Boughton, 1984) needed to be separately calibrated for each catchment.

The application of a CRRM to a watershed requires identification of the values for parameters representing model processes by fitting modelled streamflow to historical records. The calibration problem involves finding the global minimum of a multivariate function that has many extraneous local minima, a situation in which conventional local optimisation methods are ineffective (Duan et al., 1992). The existence of large numbers of local optima on the response surface causes local optimisation procedures to terminate prematurely at a local rather than global minimum.

The global optimisation problem is to find the minimum $f(\mathbf{x}^*)$ of a function of p parameters $\mathbf{x} = (x_1, x_2, \ldots, x_p)$. The function may be constrained by setting boundaries for parameter values. The point $\mathbf{x}^*$ is a local minimum of $f(\mathbf{x})$ for $\mathbf{x} \in R^p$, if a small positive number δ exists such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in R^p$ satisfying $\|\mathbf{x}^* - \mathbf{x}\| < \delta$ (Bunday and Garside, 1987; Kalivas, 1992). The point $\mathbf{x}^*$ is a global minimum of $f(\mathbf{x})$ if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in R^p$ or given domain. Function $f(\mathbf{x})$ may be either continuous or discrete. Since maximising $f(\mathbf{x})$ is equivalent to minimising $-f(\mathbf{x})$ only minimisation problems need be considered.

The problem of finding the global optimum of a multivariate function with many extraneous local minima is one of considerable complexity. Many similar discrete global optimisation problems such as the traveling salesman problem or knapsack problems are

in a class of Non-Polynomial (NP) complete problems for which no polynomial time algorithms are known. The difficulty of obtaining an accurate numerical solution grows exponentially with the number of variables. The computational complexity of multivariate integration, surface reconstruction, partial differential equations, integral equations and nonlinear optimisation is of the order of

$$(1 / \varepsilon)^{d/r} \qquad (1)$$

where all partial derivatives of the function up to order r are bounded, d is the number of dimensions and ε is the error threshold (Traub and Wozniakowski, 1994). Although the hydrologic model calibration problem can be formulated as a nonlinear optimisation problem, there is more than one local minimum from which the global minimum or a close approximation to it must be found.

Global optimisation problems occur in many disciplines including engineering, computer science, operations research, physics, image processing and biology. Efficient methods of solving these problems are required. Recent attempts at solving global optimisation problems have focused on stochastic methods which offer a close to optimal solution. Although computationally intensive these methods are more efficient than random sampling or unimodal direct search methods which stop at the first minimum encountered and cannot be readily used for finding the global minimum. Simulated Annealing (SA) (Kirkpatrick et al., 1983) is a stochastic method made feasible by the speed of modern computers. Many of the problems tackled using SA are of great scientific and industrial importance such as the design of integrated circuits, image restoration, optical design, and parameter estimation.

The parameters of a CRRM can be estimated by minimising an objective function which quantifies discrepancies between the computed and observed streamflows. The objective function must be carefully selected to avoid bias in the parameter estimates. The residuals

must be normally and independently distributed with zero mean and constant variance. Bates (1994) used SA and found it to be a reliable method of parameter estimation for the SFB model on the Scott Creek at Scotts Bottom catchment in South Australia. Bates also found that SA was more efficient and more likely to find the global minimum than multistart methods (the repetitive application of a local search method from different starting locations).

Using SA a modified version of Boughton's model was calibrated on 25 unregulated catchments (Chiew and McMahon, 1993; Chiew and McMahon, 1994). These catchments were identified as benchmark catchments by the Australian Bureau of Meteorology (1991) and represent a range of climatic and physical characteristics for Australia. The calibration identifies the important model processes for these catchments by assigning values to well determined or sensitive parameters.

2. Literature Review of Simulated Annealing

As its name suggests SA is analogous to the process of annealing in thermodynamics where metals or glass can be toughened by heating followed by slow cooling. At high temperatures the material is in a liquid state with the atoms flowing freely. Slow cooling gives the atoms time to align themselves achieving a minimal energy state for the system as the material gradually changes from a liquid to a regular crystal. SA applies this natural behaviour to the solution of both combinatorial and continuous optimisation problems.

Metropolis et al. (1953) introduced an algorithm to simulate the behavior of atoms in equilibrium at a given temperature. In this algorithm an atom is randomly displaced and the resulting change in the energy of the system ΔE is computed. If the energy of the system is lower ($\Delta E \leq 0$) the displacement is accepted and this state becomes the new starting point for the next step. When the energy of the system increases ($\Delta E > 0$) the probability of accepting the new configuration is computed from the Boltzmann probability factor

$$p(\Delta E) = \exp(-\Delta E / k_B T) \tag{2}$$

in which k_B is Boltzmann's constant and T is temperature. This allows the system to make an uphill move that may enable it to bypass a local energy minimum in the search for a more global minimum. At a high temperature a random displacement increasing the energy level of the system has a higher probability of being accepted than at a lower temperature. In general terms the algorithm normally takes a downhill step although it will sometimes take an uphill step by means of a stochastic acceptance criterion. The probability of accepting uphill moves slowly decreases as the temperature approaches zero although transitions out of a local optimum are always possible at a non zero temperature.

In its basic form SA can be described as follows:

Algorithm 1.

if $f(\mathbf{x}^k) - f(\mathbf{x}^k+\Delta\mathbf{x}) \geq 0$ **then**

$\mathbf{x}^{k+1} = \mathbf{x}^k+\Delta\mathbf{x}$

else

$\mathbf{x}^{k+1} = \mathbf{x}^k+\Delta\mathbf{x}$ with probability $\exp\{[f(\mathbf{x}^k)-f(\mathbf{x}^k+\Delta\mathbf{x})]/k_BT^k\}$

else

$\mathbf{x}^{k+1} = \mathbf{x}^k$

endif

where $\Delta\mathbf{x}$ is a random perturbation of the parameter vector $\mathbf{x}$; T^k, $k=0,1,\ldots,m$ is a sequence of positive numbers (cooling schedule) which tend to zero and $\exp\{[f(\mathbf{x}^k) - f(\mathbf{x}^k+\Delta\mathbf{x})]/k_BT^k\}$ is the Metropolis criterion from Eqn (2) where $-\Delta E = [f(\mathbf{x}^k) - f(\mathbf{x}^k+\Delta\mathbf{x})]$. In fact $\mathbf{x}^k$ is a discrete time Markov process with time dependent one step transition probabilities. At each step $\mathbf{x}^k = i$ is the current state and $\mathbf{x}^{k+1} = j$ is a neighbour selected at random to be a candidate for becoming the next state with probability

$$p\{\mathbf{x}^{k+1} = j \mid \mathbf{x}^k = i\} = q_{ij}s_{ij} \tag{3}$$

where

$$s_{i,j} = \begin{cases} 1 & \text{if } f(j) \leq f(i) \\ \exp([f(i) - f(j)] / k_BT^k) & \text{if } f(j) > f(i) \end{cases} \tag{4}$$

and q_{ij} is the transition probability from state i to state j. If the candidate is accepted a transition to the new state is made; otherwise the next state is the same as the old state.

2.1 Application of Simulated Annealing to Functions of Continuous Variables

SA was originally developed for combinatorial optimisation problems although it has been shown to be suitable for continuous optimisation problems as well (Vanderbilt and Louie, 1984; Bohachevsky et al., 1986; Corana et al., 1987; Brooks and Verdini, 1988 Press and Teukolsky, 1991; Gunel and Yazgan, 1992; Kalivas, 1992 and others). When applied to continuous functions SA behaves like a random walk with a bias in a down hill direction. The application of SA to functions of continuous variables is more challenging than for discrete problems. The random selection of a new solution point is more involved since both a direction and step size must be chosen before a transition can be evaluated (see section 2.5). Both global and local continuous optimisation methods often stagnate on flat surfaces or in long narrow valleys which are common in highly dimensional control spaces. Only continuous optimisation problems will be considered in this thesis.

SA methods are attractive because the optimum found does not depend on where the search begins, the method can bypass a local minimum in the search for a more global one and most SA algorithms do not require the computation of function derivatives.

2.2 Phases of Simulated Annealing

The SA process may pass through 1, 2 or 3 phases each designed to perform a specific task:

1) Presampling

Initial estimates of optimal step size and temperature may be obtained by presampling the function. A random starting point may be chosen or the SA algorithm may be started

from several random or remote points and a small number of independent random search paths run from each starting location.

There are several strategies for setting the initial temperature. Kirkpatrick et al. (1982) suggest using the observed acceptance rate ϕ defined as the number of accepted moves divided by the total number of moves evaluated. The procedure described by Kirkpatrick et al. (1982) entails executing a number of preliminary moves and adjusting the control variable $c = k_B T$. If ϕ is less than a specified acceptance ratio $\phi_0 = 0.8$ then the initial value c_0 is doubled. This process is repeated until $\phi > \phi_0$. If $\phi \gg \phi_0$ SA approaches a random walk, otherwise if $\phi \ll \phi_0$ the method is likely to converge to a local minimum. Kirkpatrick et al. (1982) were concerned with discrete problems and their findings may not apply to continuous optimisation problems. Johnson et al. (1987) propose determining the initial value c_0 by solving Eqn (5) where $\overline{\Delta E}$ is the mean change in the energy of the system.

$$\phi_0 = \exp(-\overline{\Delta E} / c_0) \tag{5}$$

Kalivas (1992) proposes generating samples using different temperatures until ϕ is within the desired range $0.5 < \phi < 0.9$.

2) Global search

This phase uses a heuristic algorithm, analogous to the annealing process, to find the valley containing the global minimum rather than the global minimum itself. Once the algorithm is within the region of attraction for the global minimum it may be easily located by a local search method. (A region of attraction is the region surrounding a local minimum from which application of a descent algorithm will yield the minimum.) This phase is the most expensive in terms of the number of function evaluations required.

3) Local search

The problem is now one of local rather than global optimisation and it is more efficient to use a local search technique. Provided the SA method has stopped in the neighbourhood of the global minimum a local search technique such as Nelder and Mead (1965), Hooke and Jeeves (1961) or a gradient method may be used to find the global extremum. For example, the SA method of Press and Teukolsky (1991) reduces to the Nelder and Mead (1965) downhill simplex method as $T \rightarrow 0$. If the SA method used does not have a distinct local search phase the step size should be adjusted for convergence to the required precision. This is the approach used by Corana et al (1987).

2.3 Existing Simulated Annealing Algorithms

Algorithm 1 is conceptual rather than functional since it does not contain essential information describing how the step size, starting temperature or cooling schedule are chosen. Optimum values for these variables and the optimum cooling schedule are not known. These are function dependent and a judicious choice is important since they determine the efficiency of the algorithm and probability of finding the global minimum. The step size should be selected so that the probability of exiting a local extremum is not too small causing the algorithm to slow exploration of the surface or become stuck in a local minimum. The step size should not be excessively large so that nearly all trial function evaluations are rejected or cause the algorithm not to settle in any valley including the one containing the global minimum when the cooling schedule is nearly complete. Kalivas (1992) proposes using the step size $\Delta \mathbf{x}$ as follows

$$x_j^{i+1} = x_j^i + \frac{\Delta x_j \, u_j}{(u_1^2 + u_2^2 + \ldots + u_p^2)^{0.5}} \qquad j = 1, 2, \ldots, p \tag{6}$$

where $u_1, u_2, \ldots, u_p$ are random deviates from N[0, 1] (i.e. normal density function with zero mean and unit variance). If the next transition is chosen in this manner there is no

need to choose a separate direction. Since the direction is chosen randomly, new function evaluations are likely to be uphill of the current point. A downhill bias is desirable and can be achieved by evaluating function derivatives as used by gradient methods or by using a direct search method such as Hooke and Jeeves (1961) or Nelder and Mead (1965).

When the temperature is too low the probability of accepting detrimental points is small and the algorithm becomes entrapped in local extrema. If the temperature is too high the result is a random walk. Brooks and Verdini (1988) suggest setting the temperature of the system so that about $\phi = 0.6$ of moves are accepted.

2.3.1 Generalised Simulated Annealing

Generalised Simulated Annealing (GSA) (Bohachevsky et al., 1986; Brooks & Verdini, 1988; Kalivas, 1992) was developed for optimising continuous multivariate functions and alleviates many of the problems in determining the cooling schedule for a particular optimisation problem. In GSA the Metropolis criterion given by Eqn (2) is changed to

$$p(\Delta E) = \exp\left(-\frac{1}{k_B T} \Delta E[E(\mathbf{x}) - E(\mathbf{x}^*)]^g \right) \tag{7}$$

where g is an arbitrary negative number and $E(\mathbf{x})$ the energy level at $\mathbf{x}$. If $g = -1$ and $E(\mathbf{x}^*) = 0$, then Eqn (7) becomes

$$P(\Delta E) = \exp\left(-\frac{1}{k_B T} \Delta E / E(\mathbf{x}) \right) \tag{8}$$

As the global minimum is not normally known a conservative estimate for $E(\mathbf{x}^*)$ is used. If $E(\mathbf{x}) - E(\mathbf{x}^*)$ becomes negative a new estimate for $E(\mathbf{x}^*)$ is found.

By using Eqn (7) with $g = -1$ say, GSA removes the cooling schedule and allows the probability of an uphill move to tend towards 0 as the global minimum is approached. The control parameter k_BT is fixed to an appropriate value which could be determined by presampling the function (see section 2.2). The step size $\Delta\mathbf{x}$ must also be carefully selected (see section 2.5).

2.3.2 Variable Step Size Generalised Simulated Annealing

GSA is inadequate when the exact global optimum is required. Only near global solutions can be obtained unless step size reductions were made as the global optimum is approached (Sutter and Kalivas, 1991). Variable Step Size Generalised Simulated Annealing (VSGSA) updates $\Delta\mathbf{x}$ according to the acceptance rate ϕ. Kalivas (1992) proposes maintaining ϕ such that $0.2 \leq \phi \leq 0.9$. When $\phi \geq 0.9$ the current point is presumed to reside away from the global optimum and the step sizes are increased. Otherwise if $\phi \leq 0.2$ then the current point is presumed to be close to the global optimum and the step size is reduced. Sutter and Kalivas (1992) suggest that about 20 moves are required to estimate ϕ. Initially the control parameter k_BT is fixed to an appropriate value. The interdependency of the control parameter and the step size necessitates adjusting the control parameter when $\Delta\mathbf{x}$ is updated. Here

$$k_BT^{i+1} = \frac{k_BT^i \Delta E^{i+1}}{\overline{\Delta E^i}} \tag{9}$$

where $\overline{\Delta E^i}$ represents the mean difference in the response function for the uphill moves attempted with k_BT^i and ΔE^{i+1} represents the difference for the first uphill move with the new step size. The user is required to specify the factor by which the step sizes are incremented or decremented.

2.3.3 Method of Vanderbilt and Louie (1984)

The algorithm by Vanderbilt and Louie is similar to algorithm 1 in section 2. It proceeds iteratively, starting from a given point $\mathbf{x}^0$, it generates a succession of points $\mathbf{x}^0,\mathbf{x}^1, ...,\mathbf{x}^i,...$ tending to the global minimum. New points are generated around the current point $\mathbf{x}^i$ using random moves. New candidate points are accepted or rejected according to the Metropolis criterion. The step size vector is recalculated from the step size distribution after a specified number of steps at a given temperature using

$$\Delta\mathbf{x} = \mathbf{Q} \bullet \mathbf{u} \tag{10}$$

where $u_1, u_2,...,u_n$ are uniform random deviates from the interval $[-\sqrt{3}, \sqrt{3}]$ (i.e., with mean zero and unit variance) and $\mathbf{Q}$ controls the step size distribution and is obtained from the covariance matrix of step size distributions $\mathbf{s}$ by solving

$$\mathbf{s} = \mathbf{Q} \bullet \mathbf{Q}^T \tag{11}$$

where $\mathbf{Q}$ can be obtained via Choleski decomposition. The covariance matrix automatically adapts itself to the local topography of the function and the resulting step size vector enables the method to take longer steps in the most profitable direction down the axis of long narrow valleys. The step size is calculated prior to a temperature reduction. When $\mathbf{s}$ is too small all the first steps are small and almost all are accepted and the size of the walk grows. When a large number of steps are rejected $\mathbf{s}$ begins to reflect the shape of the function. As T is reduced the walk is constrained and $\mathbf{s}$ is automatically maintained at an appropriate size. The user is required to provide the initial temperature, cooling schedule, $\mathbf{x}^0$ and an initial guess for $\mathbf{s}$. Vanderbilt and Louie applied the method to a set of seven test functions for global optimisation algorithms and a functional fitting problem.

2.3.4 Method of Corana et al. (1987)

A complete SA algorithm is given by Corana et al.. The algorithm is similar to algorithm 1 in section 2. It proceeds iteratively, starting from a given point $\mathbf{x}^0$, it generates a succession of points $\mathbf{x}^0,\mathbf{x}^1, ...,\mathbf{x}^i,...$ tending to the global minimum. New points are generated around the current point $\mathbf{x}^i$ using random moves. If the point falls outside the function domain a new point is randomly generated until a point within the function domain is found. New candidate points are accepted or rejected according to the Metropolis criterion. The algorithm includes automatic step size adjustments and the ability to set the number of function evaluations at each temperature. The step vector is periodically adjusted to try and maintain an acceptance rate of $\phi = 0.5$. A low rate means that too many moves are rejected wasting computational effort. A higher rate means that the trial configurations are too close to the starting ones and the energy of the system is not being significantly reduced. Productive configurations evolve too slowly and again computational effort is wasted. A step vector is used to record the maximum increments possible along each direction. Corana et al. recommend adjusting the step vector after every 20 moves. Corana et al. applied the method to two unimodal and three multimodal test functions.

2.3.5 Method of Press and Teukolsky (1991)

The method of Press and Teukolsky (or see Press et al., 1992) is a subtle modification of the direct search downhill simplex method of Nelder and Mead (1965). A direct search method is one that uses function values only. Press and Teukolsky's method replaces the single point $\mathbf{x}$ by a set of $p + 1$ points in p dimensional space called a simplex; thus in 2 dimensions the simplex is a triangle, in 3 dimensions it is a tetrahedron and so on. The values of the response function at the $p + 1$ vertices of the simplex are used to locate a

better point which then replaces the highest point in the simplex. This has the effect of moving the simplex in a downhill direction towards a local minimum. The simplex is moved by four basic operations: a reflection away from the highest point; an expansion away from the highest point; a one dimensional contraction towards the low point; or a multiple contraction towards the low point. The simplex method offers an efficient means for finding the downhill direction and tuning the step size based on data collected during the search. The Metropolis procedure is implemented by adding a positive logarithmically distributed random variable proportional to the notional annealing temperature T to the function value associated with every vertex of the simplex. A similar random variable is subtracted from the function value of every new candidate point being considered for entry into the simplex. This causes the algorithm to take an occasional uphill step while maintaining a downhill bias.

The SA algorithm by Press and Teukolsky uses a two-phase approach. The first (or global) phase consists of pure SA at a high temperature. The second (or local) phase occurs as $T \rightarrow 0$ and the algorithm reduces to the Nelder and Mead downhill simplex method. When $T = 0$, no uphill moves are accepted and the method converges to a local minimum. The user is required to set the starting temperature, annealing schedule and termination criteria. No examples demonstrating the use of this method were given by the authors.

2.3.6 Method of Gunel and Yazgan (1992)

The method of Gunel and Yazgan is based on the direct search method of Hooke and Jeeves (1961). The Hooke and Jeeves method consists of a series of exploration steps about the current point, and if successful, followed by an additional move in the downhill direction called a pattern move. The algorithm by Gunel and Yazgan proceeds iteratively, starting from a given point $\mathbf{x}^0$, it generates a succession of points $\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^i, \ldots$

tending to the global minimum. A new point is generated around the current point $\mathbf{x}^i$ using exploration steps with a random step size. If the point falls outside the function domain a new point is randomly generated until a point within the function domain is found. A successful series of exploration step is followed by a pattern move. New candidate points are accepted or rejected according to the Metropolis criterion. Gunel and Yazgan's method uses the direct search method to find the downhill direction and maintain a downhill bias. The user is required to set the starting temperature, annealing schedule and termination criteria.

The algorithm provided by Gunel and Yazgan has many shortcomings. The algorithm terminates if the best function value cannot be improved upon at the current temperature which usually happens well before SA is complete and before the global minimum has been found. Although the algorithm is based on a local search method it surprisingly doesn't use a local search phase to locate the bottom of the valley containing the minimum function value once SA is complete. As a result even when the valley containing the global minimum is located the algorithm cannot find the global minimum itself. The algorithm also does not have a means of adjusting the step size which prevents it from efficiently approaching the global minimum and as a result, at best, a near global solution is found. The algorithm seldom found the global minimum and was not considered further. The results given by Gunel and Yazgan using three test functions could not be reproduced. The authors reported that the algorithm found the global minimum of the Penalised Shubert function in about 140 function evaluations. The results obtained using other algorithms indicate that considerably more function evaluations are required to find the global minimum of this function with a reasonable probability of success (see section 4.4).

2.4 Cooling Schedule

There is little information in the literature on the choice of cooling schedule. An estimate of the best starting temperature T^0 and cooling rate for a function is required. Success or failure of SA is largely determined by the choice of initial temperature and cooling schedule. There is a delicate balance, temperatures that are too high waste computational effort while a temperature that is too low does not adequately sample the function and may lead to local entrapment. The following cooling schedule has been used by a number of workers (Kirkpatrick, 1983; Vanderbilt and Louie, 1984; Corana et al, 1987; Press and Teukolsky, 1991; Gunel and Yazgan, 1992)

$$T^{k+1} = rT^k, \qquad k = 0,1,\ldots,m \tag{12}$$

where T^0 is the initial temperature, $0 < r < 1$ the temperature reduction factor and m the maximum number of temperature reductions. Eglese (1990) notes that different equations for the cooling schedule produce similar results provided that their parameters are well chosen. It is extremely difficult to find an appropriate initial temperature and cooling schedule for a SA algorithm and given global optimisation problem. Usually the cooling schedule is determined by past experience with similar problems or based on trial and error. It should be noted that the initial temperature and cooling schedule giving the highest reliability (probability of successfully finding the global minimum) may not necessarily be the most efficient due to the number of function evaluations required. A robust SA algorithm will produce good results for a wide range of initial temperatures and cooling schedules.

2.5 Step Size Adjustments

The difference between the optimisation of discrete and continuous functions using SA is that the choice of random step becomes more subtle. The optimal size and direction of the step vectors are not known. Steps which are too large will invariably be rejected and may be outside the function's domain. Steps which are too small will be inefficient causing slow exploration of the function. In a valley, steps perpendicular to the valley floor are wasted and a means of determining the direction of the valley is required. As the algorithm approaches the global minimum the step size should be reduced to allow convergence. Only near global solutions can be obtained unless the algorithm is able to reduce the step size. This is the major shortcoming of the algorithm by Gunel and Yazgan (1992).

To make matters worse there is interdependency between the step size and the temperature since they both affect the probability of accepting a move. If the step size is changed it may be necessary to determine a new temperature. The variable step size generalised simulated annealing method by Kalivas (1992) computes a new value for the control parameter (temperature) each time the step size is adjusted. A simpler approach is to adjust either the step size or temperature according to the number of moves accepted and rejected. It is better to adjust the step size and let the temperature cool at a predetermined rate since increasing the temperature to obtain a specific acceptance rate can prevent cooling and leave the algorithm in an infinite loop.

The approach taken by Corana et al. (1987) is to adjust the step size so that a ratio of 1:1 between accepted and rejected configurations is maintained. This feature enables the algorithm to converge to the global minimum as the temperature is reduced. Press and Teukolsky (1991) use step size adjustments based on the expansion and contraction operations of the Nelder and Mead (1965) simplex method. The new algorithm developed in section 3 uses the same step size adjustments as the Hook and Jeeves

(1961) pattern search method to converge to the bottom of the valley containing the best function evaluation once the SA phase is complete.

2.6 Constrained Optimisation

Constrained global optimisation problems are very common. All the test functions used in section 4 are constrained optimisation problems as is the rainfall-runoff model calibration problem in section 6. The SA algorithm must be constrained to finding a solution within the specified domain for the function. One approach is to give the objective function a large value when the constraints are violated using a penalty step function. In this way the search is directed back into the feasible region. This idea has obvious intuitive appeal and is easy to program. However function evaluations requested for points outside the domain do not contribute information regarding the solution and must be handled by the function itself.

A better approach is for the SA algorithm to monitor the current location of the solution and not request function evaluations outside the domain of the function. When the current step size leads to the constraints being violated this can be achieved by temporarily reducing the step size by a factor of two, say, until a new trial point within the domain is found.

3. A New Simulated Annealing Algorithm

Combining SA with a local optimisation method can result in a SA algorithm with two desirable properties. Firstly, the local optimisation method can maintain a downhill bias during SA. Secondly, the local optimisation method can efficiently locate the minimum of the valley containing the best function evaluation when the cooling schedule is complete. Given that the local search phase is important for the efficiency of SA algorithms, it makes good sense to base a SA algorithm on a proven local search method. The nature of the response surface influences the choice of local search method to use. Given that the response surface for the catchment rainfall-runoff model contains discontinuities (Johnston and Pilgrim, 1976; Duan et al., 1992) it makes sense to use a direct search method such as that of Hooke and Jeeves (1961) or Nelder and Mead (1965) rather than a derivative based method. The SA method by Press and Teukolsky is based on the method of Nelder and Mead and has these desirable features. Although the algorithm by Gunel and Yazgan is loosely based on the method of Hooke and Jeeves it does not use a separate local search phase which is a deficiency in this algorithm. The method of Gunel and Yazgan also does not have a mechanism to reduce the step length as the algorithm approached the global minimum.

A new SA algorithm was developed based on experience with existing SA algorithms. The algorithm uses the pattern search method of Hooke and Jeeves (1961) to maintain a strong downhill bias and to locate the bottom of the valley containing the best function evaluation during the local search phase. The exploration steps and pattern moves are carried out as described by Hooke and Jeeves, the Metropolis criterion acts on the pattern moves to make the algorithm accept an occasional uphill move. The simplified algorithm is as follows (see Appendix B for program listing):

Algorithm 2.

Step 0. Initialisation.
Given initial base point $\mathbf{b}_1$, number of cycles per temperature c, initial step length h, minimum step length $h_{\min}$, initial temperature T^0 and number of temperature reduction steps m.
$f_1 = f(\mathbf{b}_1)$
$f_{\text{opt}} = f(\mathbf{b}_1)$
$i = 1$
$k = 0$
search = global

Step 1. Explore function surrounding base point $\mathbf{b}_1$.
Perform exploration to obtain new base point $\mathbf{b}_2$, with value f_2

Step 2. Try pattern move to reduce function value.
if $f_1 - f_2 \geq 0$ **then**
    Perform pattern move to obtain base point $\mathbf{b}_p$, and f_p
else if search = global **then**
    goto Step 4
else if $h < h_{\min}$ **then**
    solved
else
    $h = h / 10$
endif

Step 3. Accept downhill move or when uphill make Metropolis move.
if $f_2 - f_p \geq 0$ **then**
    $\mathbf{b}_1 = \mathbf{b}_p$
    $f_1 = f_p$
    if $f_{\text{opt}} - f_p \geq 0$ **then**
        $\mathbf{b}_{\text{opt}} = \mathbf{b}_p$
        $f_{\text{opt}} = f_p$
    endif
else if search = global **then**
    $\mathbf{b}_1 = \mathbf{b}_p$ }
    $f_1 = f_p$ } with probability $\exp\{[f_1 - f_p]/T^k\}$
else if search = local **then**
    $\mathbf{b}_1 = \mathbf{b}_2$
    $f_1 = f_2$
endif

Step 4. If i < c then goto step 1; else it is time to reduce the temperature.
if $i < c$ **then**
    $i = i + 1$
    goto Step 1
else if search = global **then**
    $\mathbf{b}_1 = \mathbf{b}_{\text{opt}}$
    $f_1 = f_{\text{opt}}$
    if $k < m$ **then**
        $i = 0$
        $k = k + 1$
        $T^k = r\ T^k$
    else
        search = local
    endif
endif
goto Step 1

The exploratory steps examine each variable in turn by adding the step length. Thus we evaluate $f(\mathbf{b}_1 + zh_1\mathbf{e}_1)$ where for global minimisation z is a uniform random deviate from the interval [0, 1] and for local minimisation $z = 1$; $\mathbf{e}_1$ is a unit vector in the direction of the x_1-axis. If the function is reduced replace $\mathbf{b}_1$ by $\mathbf{b}_1 + zh_1\mathbf{e}_1$ otherwise evaluate $f(\mathbf{b}_1 - zh_1\mathbf{e}_1)$ and likewise replace $\mathbf{b}_1$ by $\mathbf{b}_1 - zh_1\mathbf{e}_1$ if the function is reduced. If neither step results in a reduction leave $\mathbf{b}_1$ unchanged and consider changes in x_2 and so on until all n directions have been considered and we have a new base point $\mathbf{b}_2$.

Pattern moves attempt to accomplish further function minimisation using the information acquired by exploration with an additional move in the downhill direction. The pattern move attempts to move from the base point $\mathbf{b}_2$ in the direction $\mathbf{b}_2 - \mathbf{b}_1$ to $\mathbf{b}_p$ as given by

$$\mathbf{b}_p = \mathbf{b}_1 + 2(\mathbf{b}_2 - \mathbf{b}_1) \tag{13}$$

The algorithm has the following features:

1) When SA is complete the Hooke and Jeeves (1961) local search method incorporating step size reductions is used to efficiently approach the global minimum.
2) The algorithm will not terminate until the cooling schedule is complete. This ensures that the function has been sampled sufficiently.
3) The Hooke and Jeeves exploratory steps enable the algorithm to efficiently travel down long narrow valleys by determining the downhill direction.
4) The algorithm reverts to the method of Hooke and Jeeves as $T \rightarrow 0$. This is desirable since the down hill bias increases as the temperature is reduced.
5) The algorithm terminates during the local search phase when the step length has been reduced below the specified minimum step length. This provides a very simple and effective test for convergence.

6) When called with $T^0 = 0$ the Hooke and Jeeves pattern search method is performed.

7) The algorithm is automatically constrained to stay within the function domain and will not request function evaluations outside this domain as do most other methods. This problem is overcome by temporarily halving the step size until a point within the functions domain is found. This also enables the algorithm to converge on a boundary of the domain.

8) The best function evaluation is always retained. The base point is replaced by the point corresponding to the best function value every time the temperature is reduced.

4. Evaluation of Simulated Annealing Algorithms

4.1 Test Functions

Three functions previously used to test global optimisation algorithms were chosen. The functions offer different challenges and are shown in Appendix A. The three dimensional (p = 3) Hartman function has several long flat valleys designed to thwart optimisation algorithms. The global minimum was incorrectly reported by Dixon and Szego (1978) as approximately (.038, .574, .883), and $f(\mathbf{x}^*)$ = -3.2. The global minimum was correctly reported by Brooks and Verdini (1988) and Butler and Slaminka (1992); as approximately (.11, .555, .855), and $f(\mathbf{x}^*)$ = -3.86278. The function is difficult for optimisation algorithms because it is almost flat in one direction at the global minimum.

Hartman's family:

$$f(\mathbf{x}) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{p} a_{ij}(x_j - b_{ij})^2} \qquad 0 \le x_i \le 1, \quad i = 1,\ldots,p. \tag{14}$$

See function plot in Appendix A for values of a_{ij}, b_{ij} and c_i.

The Rastrigan function (Polovinkin, 1981; Benke and Skinner, 1991) is smooth and continuous with 50 local minima in a lattice arrangement. The function has a global minimum of $f(\mathbf{x}^*)$ = -2 at (0, 0). Polovinkin reported that a Monte Carlo search required 5,917 function evaluations and multistart gradient methods 556 function evaluations to find the global minimum. Unfortunately the probability of success using these methods was not reported.

Rastrigin function :

$$f(\mathbf{x}) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2) \quad -1 \le x_i \le 1, \quad i = 1,2. \tag{15}$$

The Penalised Shubert function (Gunel and Yazgan, 1992) is a very challenging function for global optimisation algorithms because the local minima are deep, highly isolated, and their depths are not apparent except in a small volume near the core of each minimum. In the given domain the function has 760 minima. The function has a unique global minimum of $f(\mathbf{x}^*)$ = -186.731 at (-1.4251, -0.8003) and a close to optimal solution of $f(\mathbf{x})$= -186.341 at (- 0.8000, -1.4251).

Penalised Shubert function ($b = 0.5$):

$$f(\mathbf{x}) = \left\{\sum_{i=1}^{5} i\cos[(1+1)x_1 + i]\right\} \qquad \left\{\sum_{i=1}^{5} i\cos[(i+1)x_2 + i]\right\} \qquad (16)$$

$$+b\left[(x_1 + 1.4251)^2 + (x_2 + 0.8003)^2\right] \quad -10 \le x_i \le 10, \qquad i = 1,2$$

4.2 Method of Press and Teukolsky (1991) with Restarts

Press and Teukolsky (1991) suggest that an occasional restart where a vertex in the simplex is discarded in favour of the best point encountered may be beneficial increasing the efficiency of their algorithm. They recommended using a restart every time the temperature has been reduced by a factor of three. However, a restart must not be performed when the best point is currently in the simplex otherwise the simplex will degenerate. The authors reported that a restart was highly beneficial on some problems although on other problems there was no positive or a somewhat negative effect. Bates (1994) reported that the algorithm by Press and Teukolsky (1991) sometimes discarded the best solution in the simplex for a new solution with a higher function value. This can lead to the situation where the final solution in the simplex is worse than the best solution found during the SA run. Bates (1994) found that performing a restart when the solution is not currently in the simplex and T has been reduced by a factor of two or three overcame this deficiency in the algorithm. No quantitative comparison of the algorithms was made.

The restart feature was investigated by comparing the algorithm with and without restarts using the three test functions in section 4.1. A range of initial temperatures ($0 \leq T^0 \leq 40$) and cooling schedules ($0.6 \leq r \leq 1.0$; $1 \leq m \leq 51$) in Eqn (12) were used for both the original and modified algorithms.

The results obtained for the Hartman function are shown as Figure 1. The columns of points in both Figures 1a and 1b correspond to different numbers of steps in the cooling schedule (m) with the exception of the first column. The first column of points ($T^0 = 0$) corresponds to the Nelder Mead simplex method with a higher probability of success than SA with a poor choice of cooling schedule (the second column of points where $m = 1$). Each column includes a full range of reduction factors (r). The cluster of points in Figure 1a corresponds to cooling schedules ($0.6 \leq r \leq 0.88$; $21 \leq m \leq 51$) where SA is working effectively and the Nelder Mead local search method is being used to find the bottom of the valley containing the global minimum. The points where the probability of success is greater than or equal to 0.98 in Figure 1b corresponds to cooling schedules ($0.6 \leq r \leq 0.9$; $16 \leq m \leq 51$) where SA is working effectively and the Nelder Mead local search method is being used to find the bottom of the valley containing the global minimum. As can be seen by comparing Figures 1a and 1b the restart feature increased the probability of finding the global minimum of the Hartman function when a reasonable cooling schedule was used. The restart feature also increased the probability of success for cooling schedules with a small number of temperature reduction steps (m).

The results obtained for the Rastrigin function are shown as Figure 2. The restart feature greatly increased the probability of finding the global minimum of the Rastrigin function provided a reasonable cooling schedule was used. The clusters of points in Figures 2a and 2b correspond to cooling schedules where the Nelder Mead local search method is being used to find the bottom of the valley containing the global minimum.

The results obtained for the Penalised Shubert function are shown as Figure 3. The restart feature slightly increased the probability of finding the global minimum of the Penalised Shubert function provided a reasonable cooling schedule was used.

In conclusion, the modified Press and Teukolsky (1991) algorithm with restarts gave a higher probability of finding the global minimum on all three test functions. The increase in reliability gained by using the restart feature depended upon the test function used. The modified algorithm with restarts was used for all further comparisons with other algorithms.

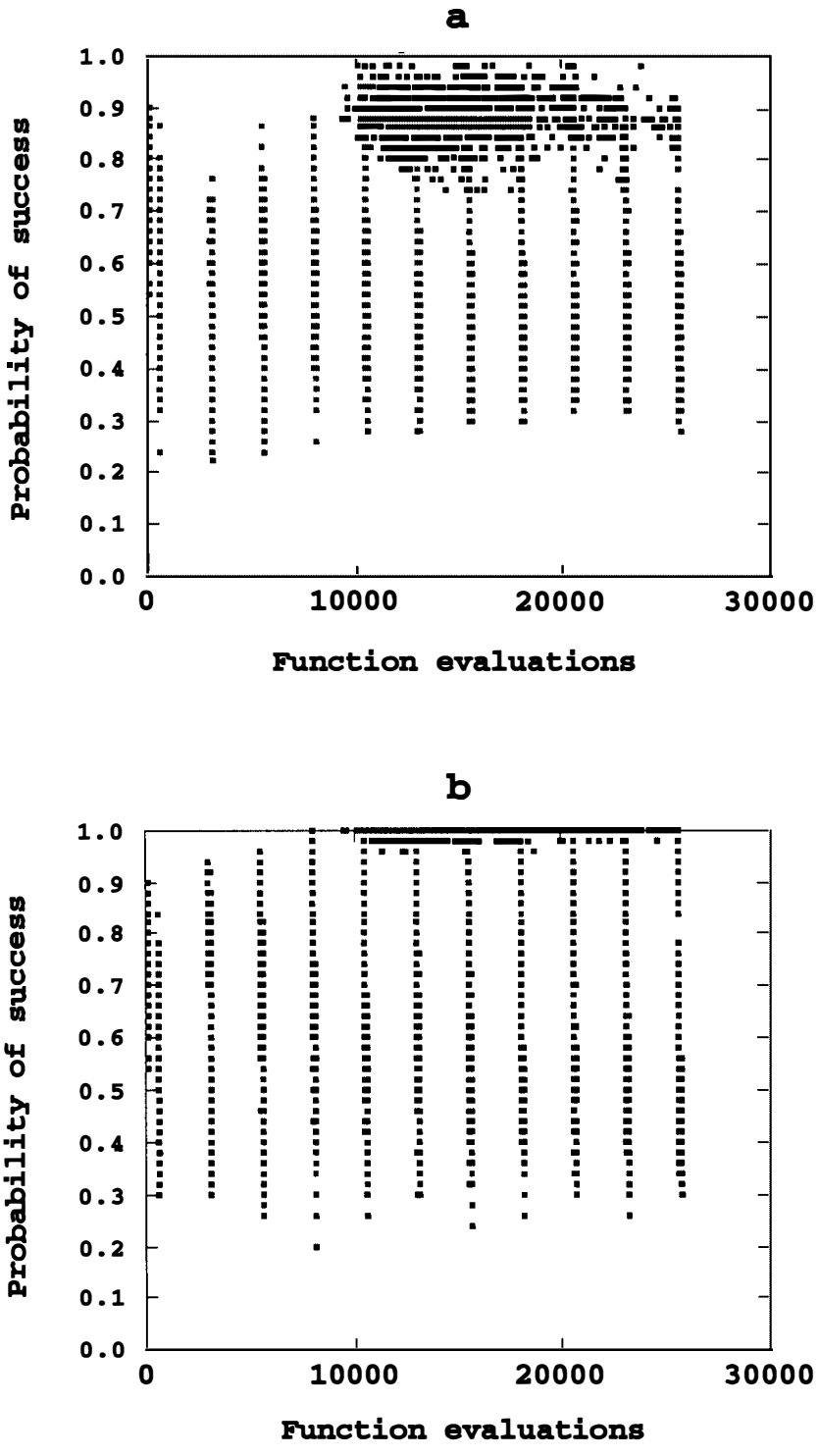


Fig. 1. Probability of successfully finding the global minimum of the Hartman function using the method of a) Press & Teukolsky, b) Press & Teukolsky with restarts.

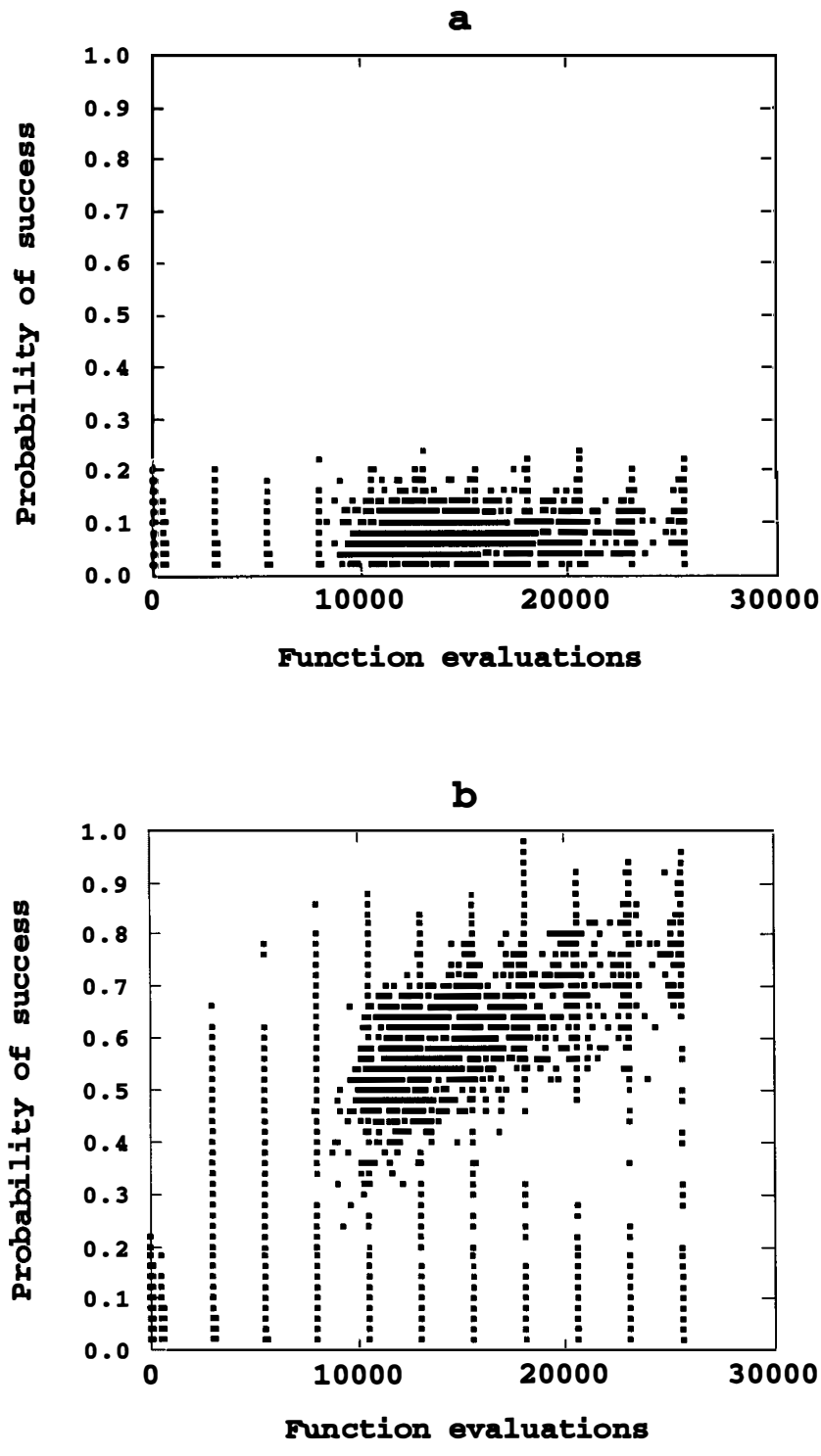


Fig. 2. Probability of successfully finding the global minimum of the Rastrigin function using the method of a) Press & Teukolsky, b) Press & Teukolsky with restarts.

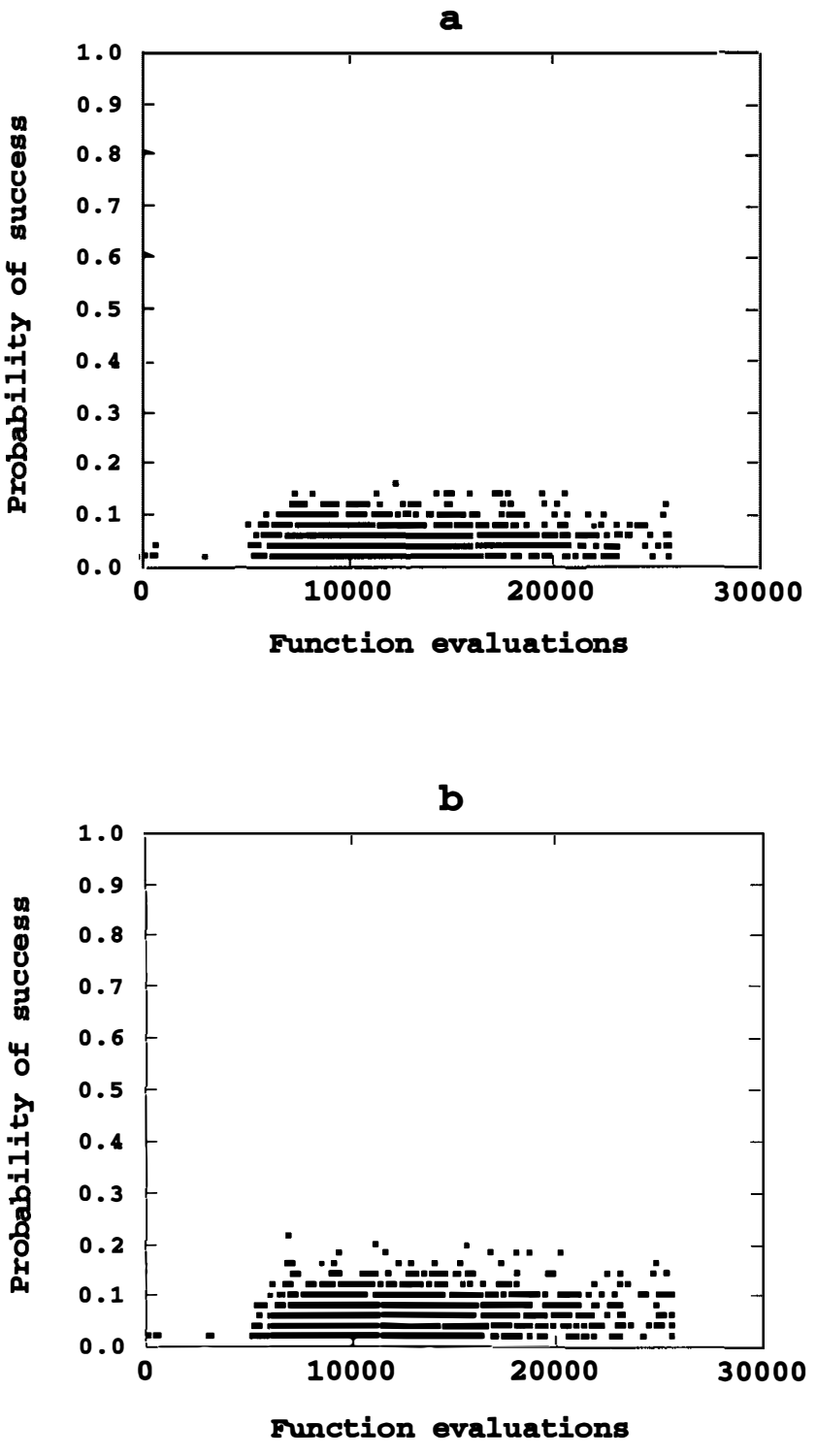


Fig. 3. Probability of successfully finding the global minimum of the Penalised Shubert function using the method of a) Press & Teukolsky, b) Press & Teukolsky with restarts.

4.3. Investigation of Initial Temperature and Cooling Schedules

This section investigates the sensitivity of four SA algorithms to the initial temperature and cooling schedule using the three test functions from section 4.1. The results are compared with recommended procedures from the literature for setting the initial temperature. Guidelines for the choice of initial temperature and cooling schedule are given where possible. The reliability and efficiency of the SA algorithms is investigated in section 4.4.

The reliability and efficiency of different cooling schedules was investigate by systematically trying different values for T^0, r and m in Eqn (12). For each initial temperature and cooling schedule 100 SA runs were initially attempted. This proved to be too great a burden on the computer system used and the sample size was later reduced to 50. The algorithms by Corana et al. (1987), Vanderbilt and Louie (1984), Press and Teukolsky (1991) with restarts and the new algorithm described in section 3 were investigated using the test functions described in section 4.1. The SA programs were considered to have found the global minimum if the following absolute convergence criteria was satisfied

$$|f(\mathbf{x}^*) - f(\mathbf{x})| < tol \tag{17}$$

where $f(\mathbf{x}^*)$ is the known global minimum, $f(\mathbf{x})$ is the estimate of the global minimum found by the program and $tol = 0.0001$. The maximum number of function evaluations for all four SA algorithms was set to 30,000.

4.3.1 Hartman Function

Algorithm by Corana et al. (1987)

The results for the algorithm by Corana et al. are shown in Figure 4. The probability of success is sensitive to T^0 and $T^0 > 0.13$ (the T^0 value where a plateau is reached or the lowest T^0 where probability of one is obtained) is required to avoid entrapment in local minima. For this function the algorithm does not appear to be sensitive to the temperature reduction factor (r). When $r = 1$ the algorithm reverts to a Monte Carlo search which also has a reasonable probability of finding the global minimum of this function provided a suitable value for T^0 is used, although a large number of function evaluations are required. The user is not able to control the number of temperature reduction steps for this algorithm since the cooling schedule proceeds until convergence is obtained. The cluster of points at $m \approx 2000$ corresponds to $r = 1$ and was caused by the maximum number of function evaluations exceeding the limit of 30,000 without satisfying the convergence criteria used by the algorithm. The increase in variance with T^0 is due to the higher final temperatures (T^f) resulting for some cooling schedules. The final temperature should be close to zero for SA. High final temperatures correspond to Monte Carlo searches when r is approaching one.

Algorithm by Vanderbilt and Louie (1984)

The results for the algorithm by Vanderbilt and Louie are shown in Figure 5. The probability of success is sensitive to T^0 and $T^0 > 0.5$ (the T^0 value where a plateau is reached or the lowest T^0 where probability of one is obtained) is required to avoid entrapment in local minima. The algorithm is sensitive to the temperature reduction factor (r) and $0.75 < r < 1$ gives the best results. Vanderbilt and Louie used $T^0 = 1.0$ and $r = 0.6$ when applying their method to the Hartman function; a higher value for r would have given better results. The user is not able to control the number of temperature

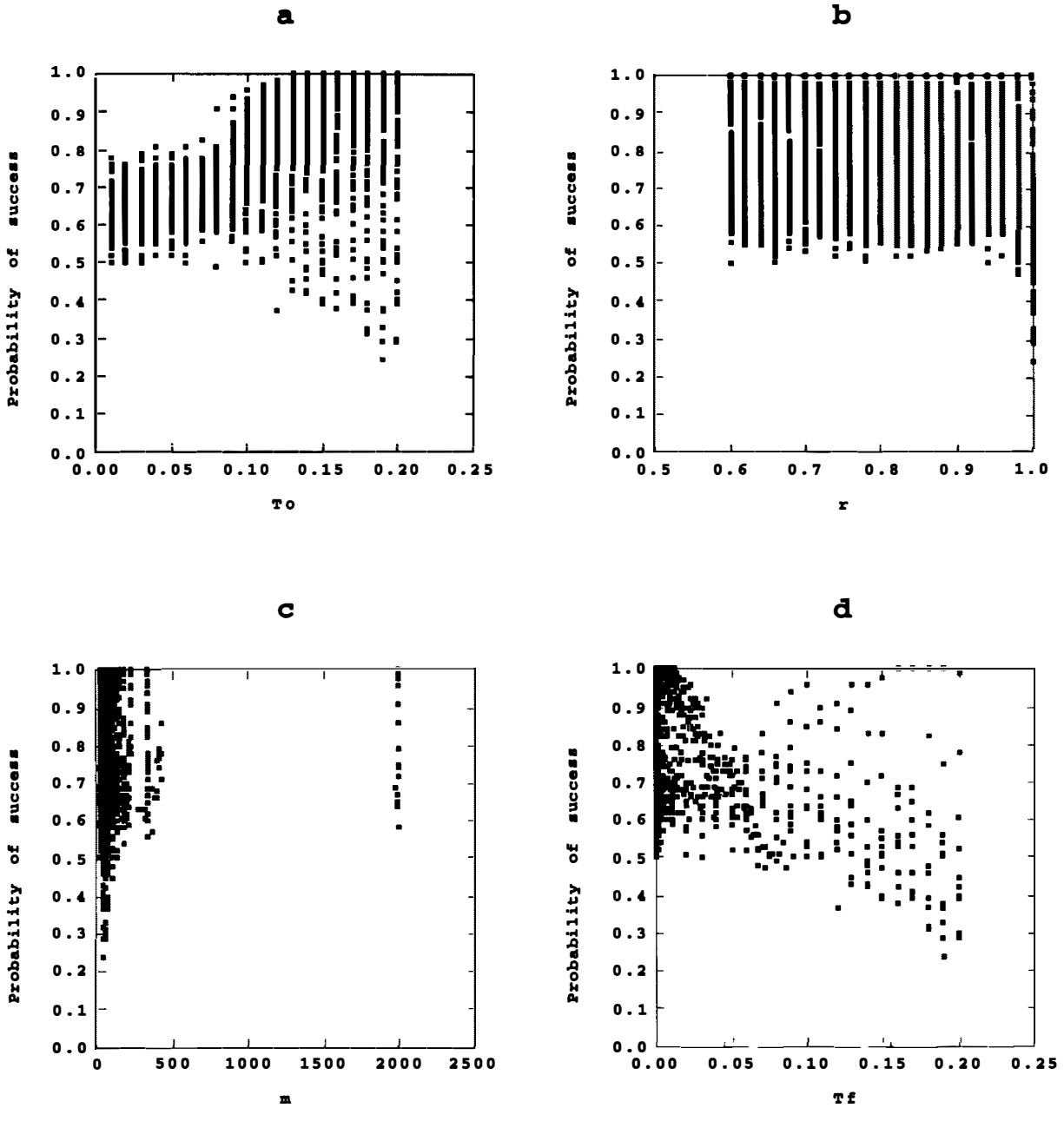


Fig. 4. Probability of successfully finding the global minimum of the Hartman function using the method of Corana et al. by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

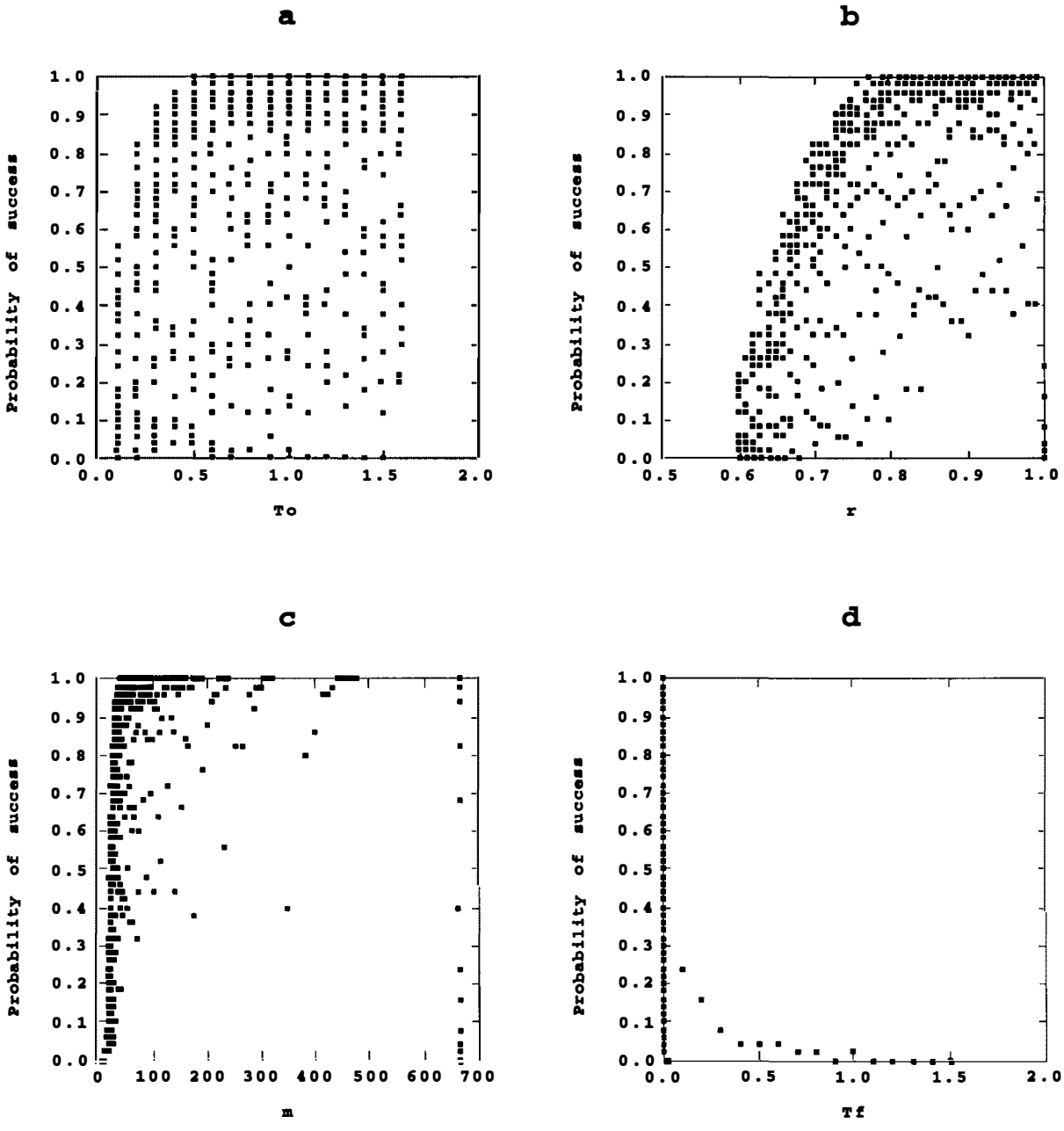


Fig. 5. Probability of successfully finding the global minimum of the Hartman function using the method of Vanderbilt & Louie by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

reduction steps for this algorithm since the cooling schedule proceeds until convergence is obtained. The cluster of points at $m \approx 680$ corresponds to $r = 1$ and were caused by the maximum number of function evaluations exceeding the limit of 30,000 without satisfying the convergence criteria used by the algorithm. The final temperature should be close to zero.

Algorithm by Press and Teukolsky (1991) with restarts

The results for the algorithm by Press and Teukolsky(1991) are shown in Figure 6. Entrapment in local minima is not a problem and the probability of success is not sensitive to T^0 provided $T^0 \geq 0.4$. $T^0 = 0$ (the first column of points) corresponds to the Nelder Mead simplex method which performed reasonably well on this function with a mean probability of success of 0.74 (probabilities between 0.56 and 0.88 occurred for different series of runs using this method). The algorithm is sensitive to the temperature reduction factor and $r < 0.90$ appears to give the best results. When $r = 1$ the algorithm reverts to a Monte Carlo search with a lower probability of finding the global minimum of this function. A large number of temperature reduction steps increases the probability of finding the global minimum with $m \approx 15$ (the m value where a plateau is reached or the lowest m where probability of one is obtained) being the most efficient. The algorithm is very sensitive to the final temperature which must be close to zero.

New Algorithm

The results for the new algorithm are shown in Figure 7. The probability of success is sensitive to T^0 and $T^0 > 4$ helps to avoid entrapment in local minima. $T^0 = 0$ (the first column of points) corresponds to the Hooke and Jeeves pattern search method with a mean probability of 0.56. Hooke and Jeeves does not perform as well as Nelder and Mead for this function because it only uses one starting point rather than four required by Nelder and Mead to form the simplex on this function. The Nelder Mead method drags

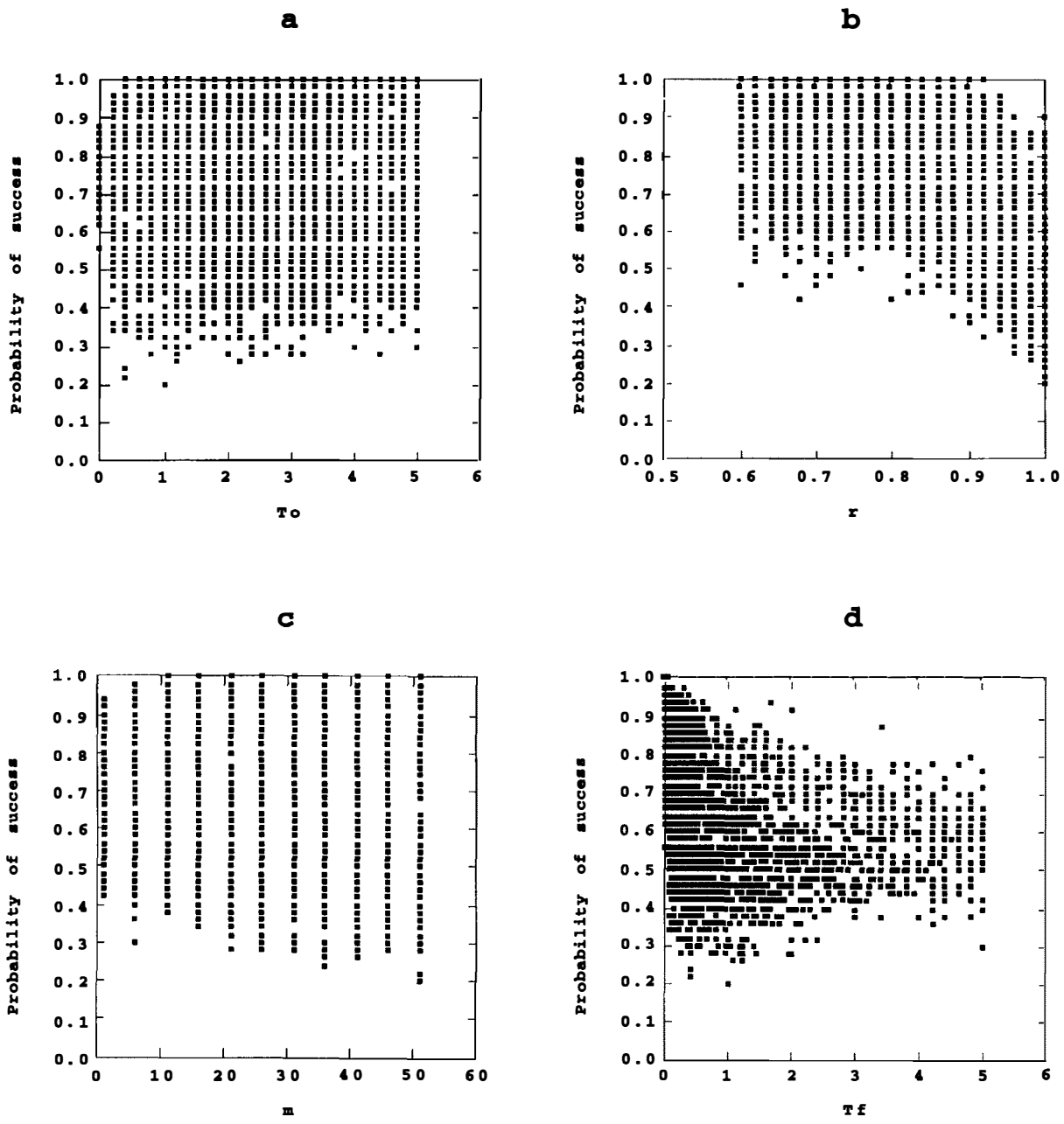


Fig. 6. Probability of successfully finding the global minimum of the Hartman function using the method of Press & Teukolsky with restarts by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

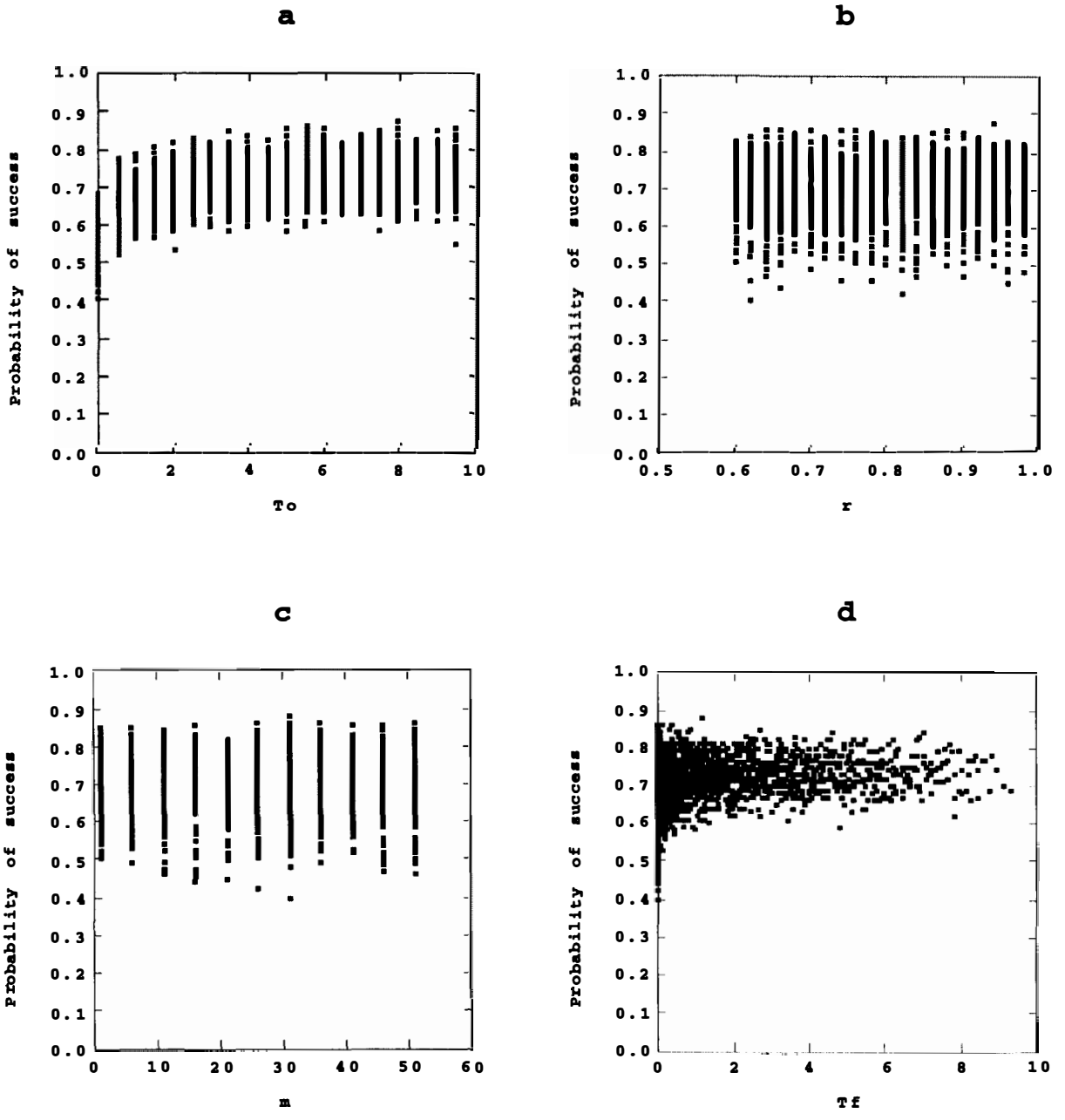


Fig. 7. Probability of successfully finding the global minimum of the Hartman function using the new method by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

all four points across the response surface as the simplex expands and contracts. The SA algorithm is insensitive to the temperature reduction factor and number of reduction steps. The algorithm is not very sensitive to the final temperature although $T^f \approx 0$ appears to be desirable. The points corresponding to $T^f = 0$ on Figure 7d with a low probability of success are due to the Hooke and Jeeves method. This SA algorithm is less sensitive to T^0, r, m and T^f than the other algorithms considered.

4.3.2 Rastrigin Function

Algorithm by Corana et al. (1987)

The results for the algorithm by Corana et al. are shown in Figure 8. The probability of success is sensitive to T^0 and $T^0 > 1.8$ is required to avoid entrapment in local minima. Surprisingly the points with the highest probability were obtained from a Monte Carlo search ($r \approx 1$) and approximately 30,000 function evaluations. This extensive sampling required considerable computational effort to obtain a higher probability of finding the global minimum than the SA algorithm. For this function the SA algorithm is sensitive to the temperature reduction factor, $r > 0.95$ giving the best results. The cluster of points in Figure 8c corresponding to $m \approx 2000$ with $r = 1$ were caused by the maximum number of function evaluations exceeding the limit of 30,000. The final temperature should be close to zero for SA. The high final temperatures correspond to Monte Carlo searches where r was approaching one.

Algorithm by Vanderbilt and Louie (1984)

The results for the algorithm by Vanderbilt and Louie are shown in Figure 9. The probability of success is sensitive to T^0 and $T^0 > 4$ is required to avoid entrapment in local minima. The algorithm is sensitive to the temperature reduction factor (r) and

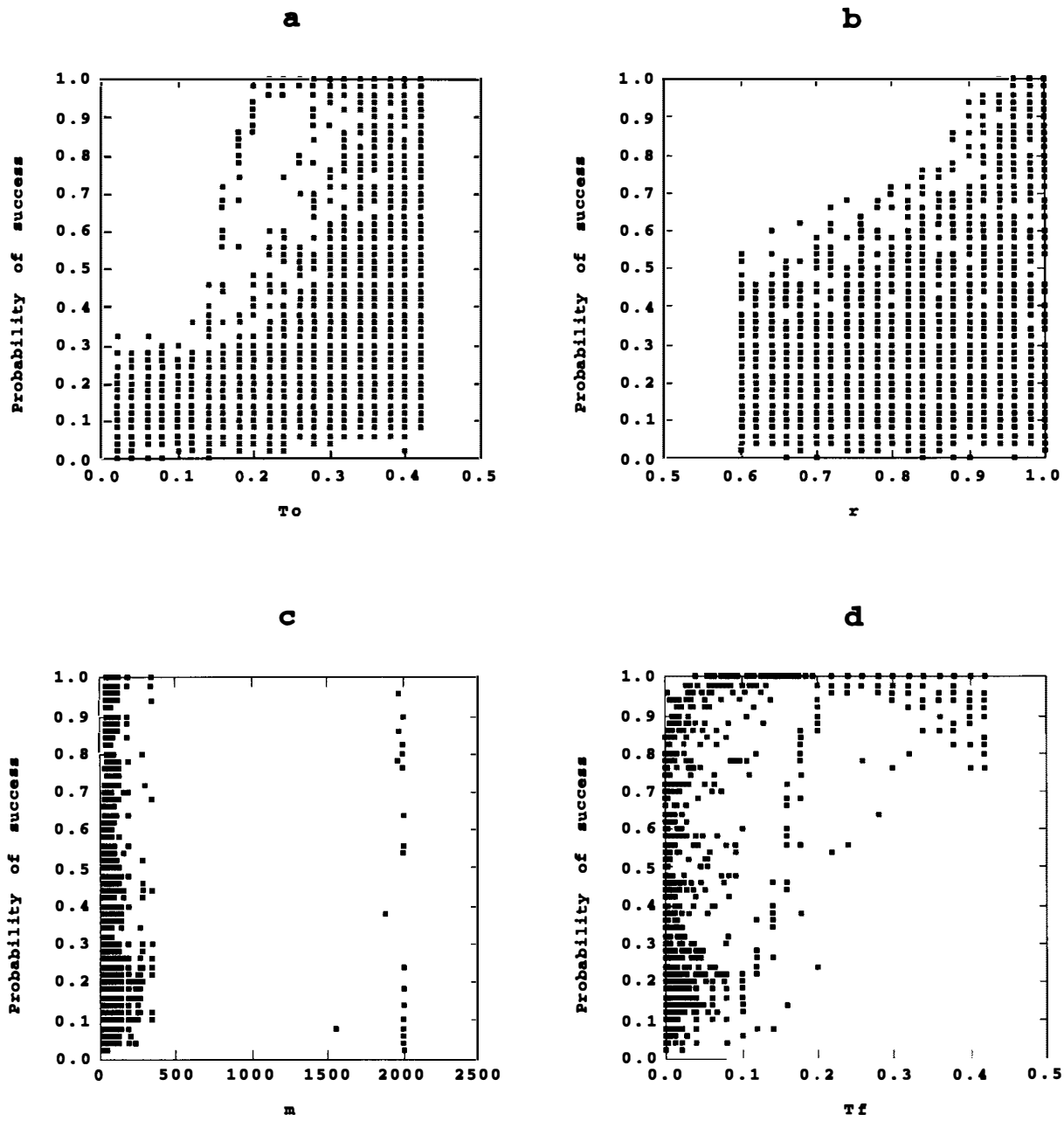


Fig. 8. Probability of successfully finding the global minimum of the Rastrigin function using the method of Corana et al. by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

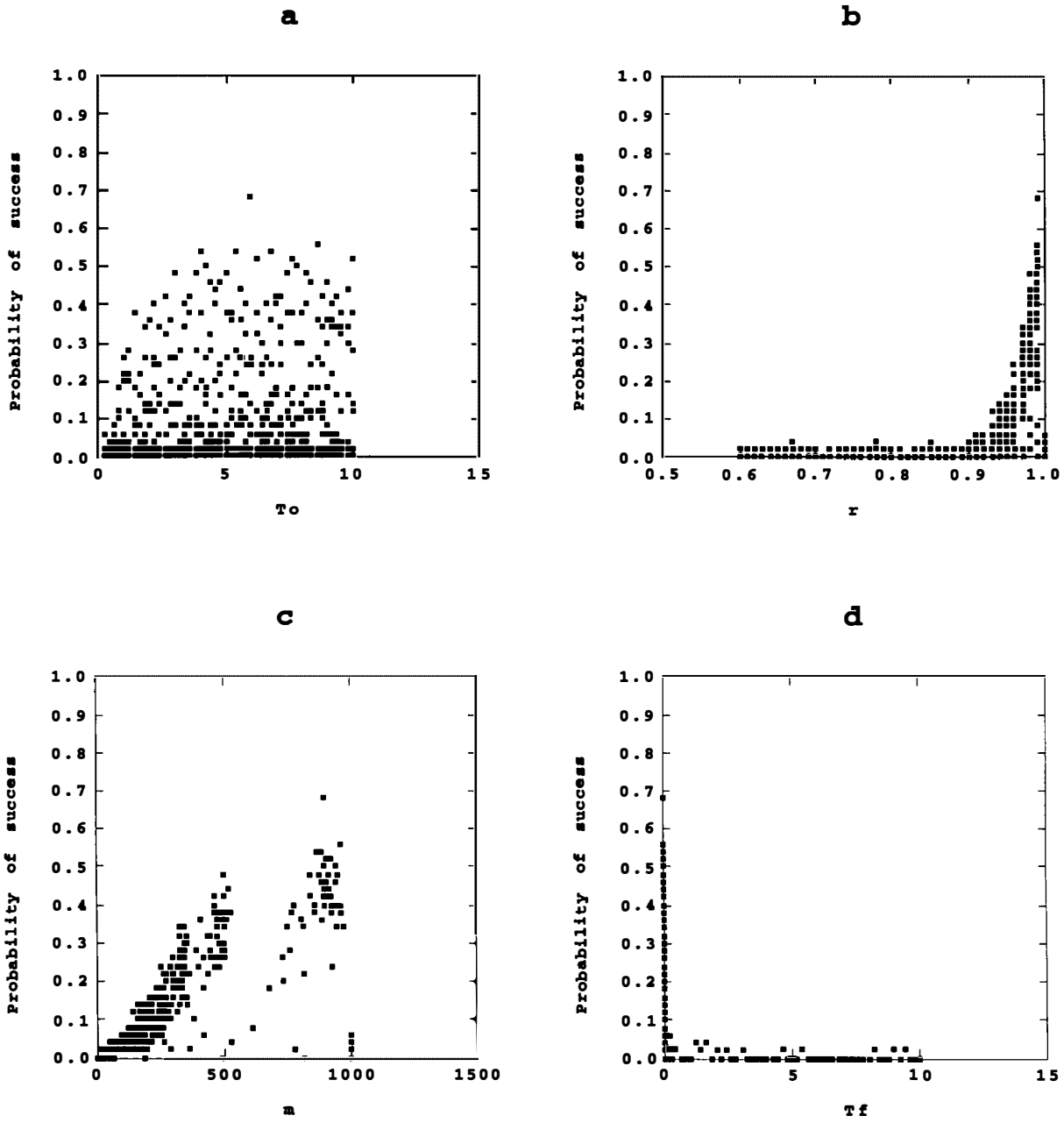


Fig. 9. Probability of successfully finding the global minimum of the Rastrigin function using the method of Vanderbilt & Louie by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

$r = 0.99$ gives the best results. The cluster of points at $m \approx 850$ corresponds to a slow cooling schedule with $r = 0.99$. The final temperature should be close to zero.

Algorithm by Press and Teukolsky (1991) with restarts

The results for the algorithm by Press and Teukolsky are shown in Figure 10. Entrapment in local minima is not a problem and the probability of success is not sensitive to T^0 provided $T^0 \geq 2$. $T^0 = 0$ (the first column of points) corresponds to the Nelder Mead simplex method with a mean probability of 0.09 which is considerably lower than that obtained by SA. The algorithm is sensitive to the temperature reduction factor and $0.82 \leq r \leq 0.94$ appears to give the best results. A large number of temperature reduction steps increases the probability of finding the global minimum with $m \approx 35$ being the most efficient. The algorithm is very sensitive to the final temperature which must be close to zero.

New Algorithm

The results for the new algorithm are shown in Figure 11. $T^0 = 0$ (the first column of points) corresponds to the Hooke and Jeeves pattern search method with a mean probability of 0.11 which is considerably lower than that obtained by SA. The performance of Hooke and Jeeves is slightly better than that obtained using Nelder and Mead for this function. The bands on the plots are due to the distinctive behavior of the algorithm under different conditions. The band with a probability of about 0.5 corresponds to a poor choice of annealing schedule with $m = 1$. The band with a probability close to one corresponds to reasonable cooling schedules where SA is working properly ($m > 1$). The probability of success is not sensitive to the initial temperature provided $T^0 \geq 0.5$. The algorithm is insensitive to the temperature reduction factor and number of reduction steps. The algorithm is not very sensitive to the final temperature although $T^f \approx 0$ appears to be desirable. Since it has a higher

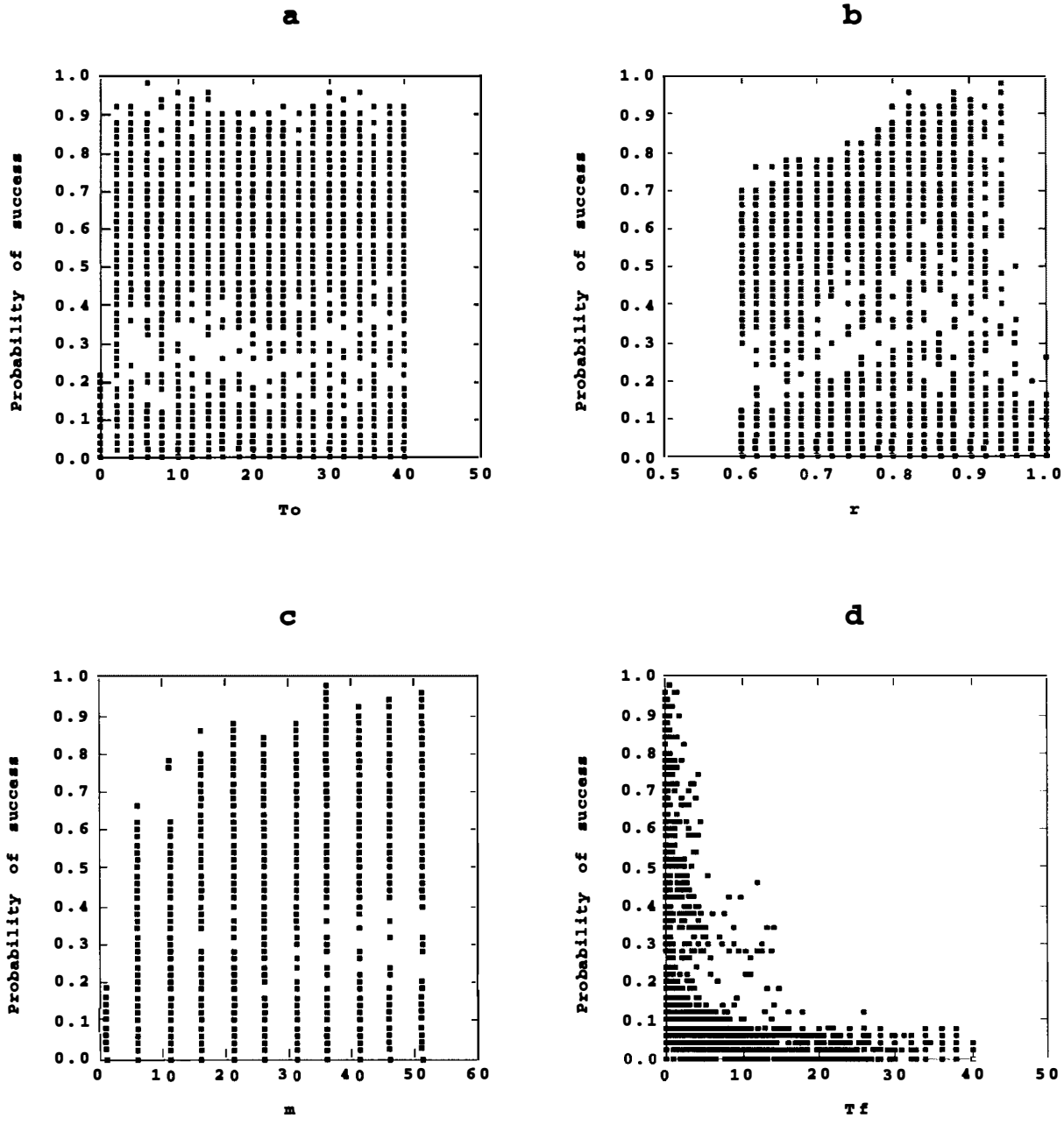


Fig. 10. Probability of successfully finding the global minimum of the Rastrigin function using the method of Press & Teukolsky with restarts by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

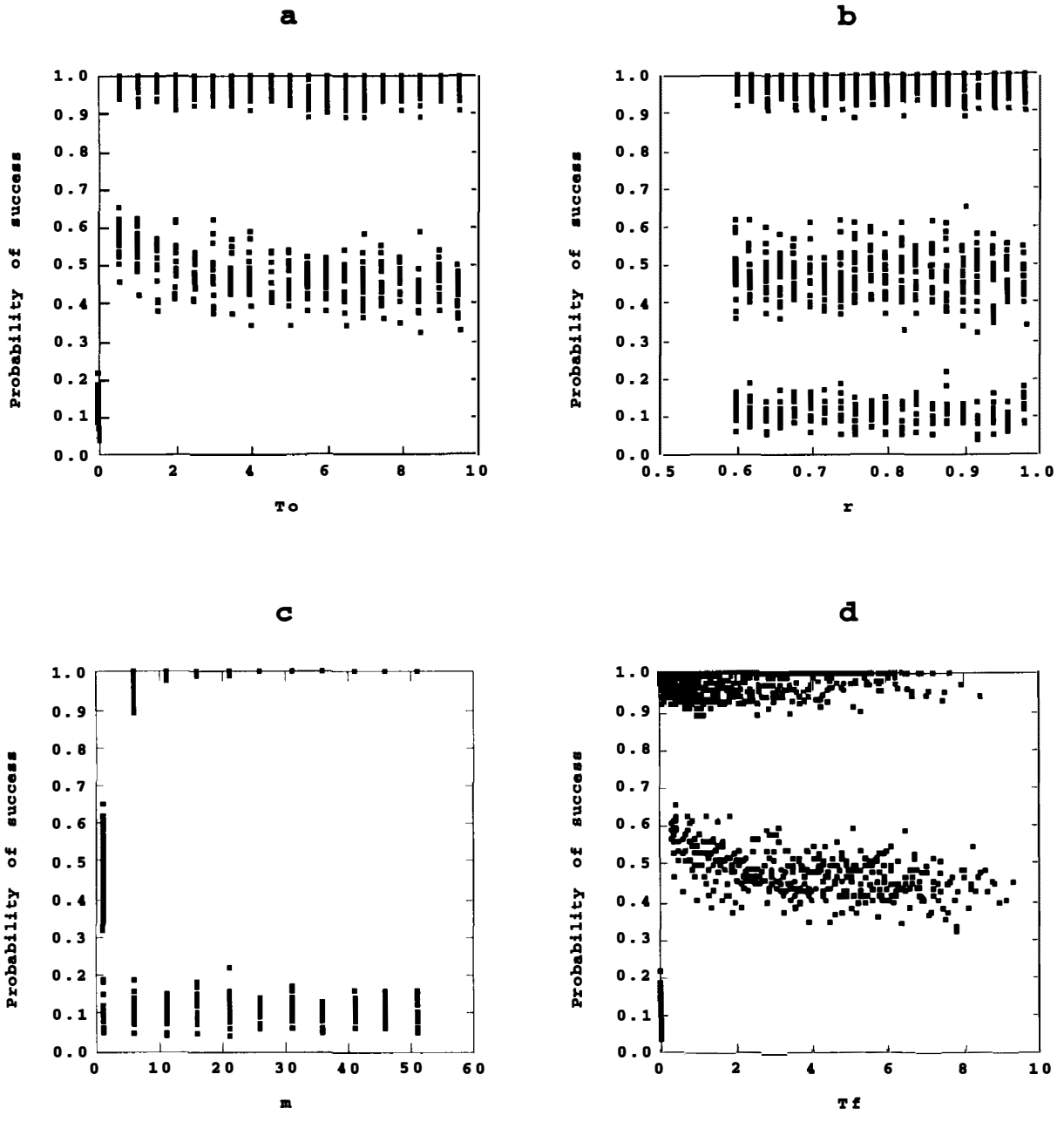


Fig. 11. Probability of successfully finding the global minimum of the Rastrigin function using the new method by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

probability of success and is less sensitive to T^0, r, m and T^f this algorithm appears to be more reliable and robust than the other algorithms considered.

4.3.3 Penalised Shubert Function

Algorithm by Corana et al. (1987)

The results for the algorithm by Corana et al. are shown in Figure 12. The probability of success increases as T^0 is increased due to the more extensive sampling performed before the algorithm converges to a minimum. Monte Carlo sampling would also give a similar increase in probability of success as the number of function evaluations is increased. It is difficult to choose an initial temperature for this algorithm because of the relationship between T^0 and the number of function evaluations required. For this function the algorithm is sensitive to the temperature reduction factor and $0.75 < r < 0.95$ gives the best results. The final temperature should be close to zero.

Algorithm by Vanderbilt and Louie (1984)

The results for the algorithm by Vanderbilt and Louie are shown in Figure 13. The probability of success is sensitive to the initial temperature and a high value for T^0 gives a slightly higher probability of success. The algorithm is sensitive to the temperature reduction factor and $r \approx 0.99$ gives the best results. The final temperature should be close to zero.

Algorithm by Press and Teukolsky (1991) with restarts

The results for the algorithm by Press and Teukolsky are shown in Figure 14. The probability of success does not appear to be sensitive to T^0. $T^0 = 0$ (the first column of

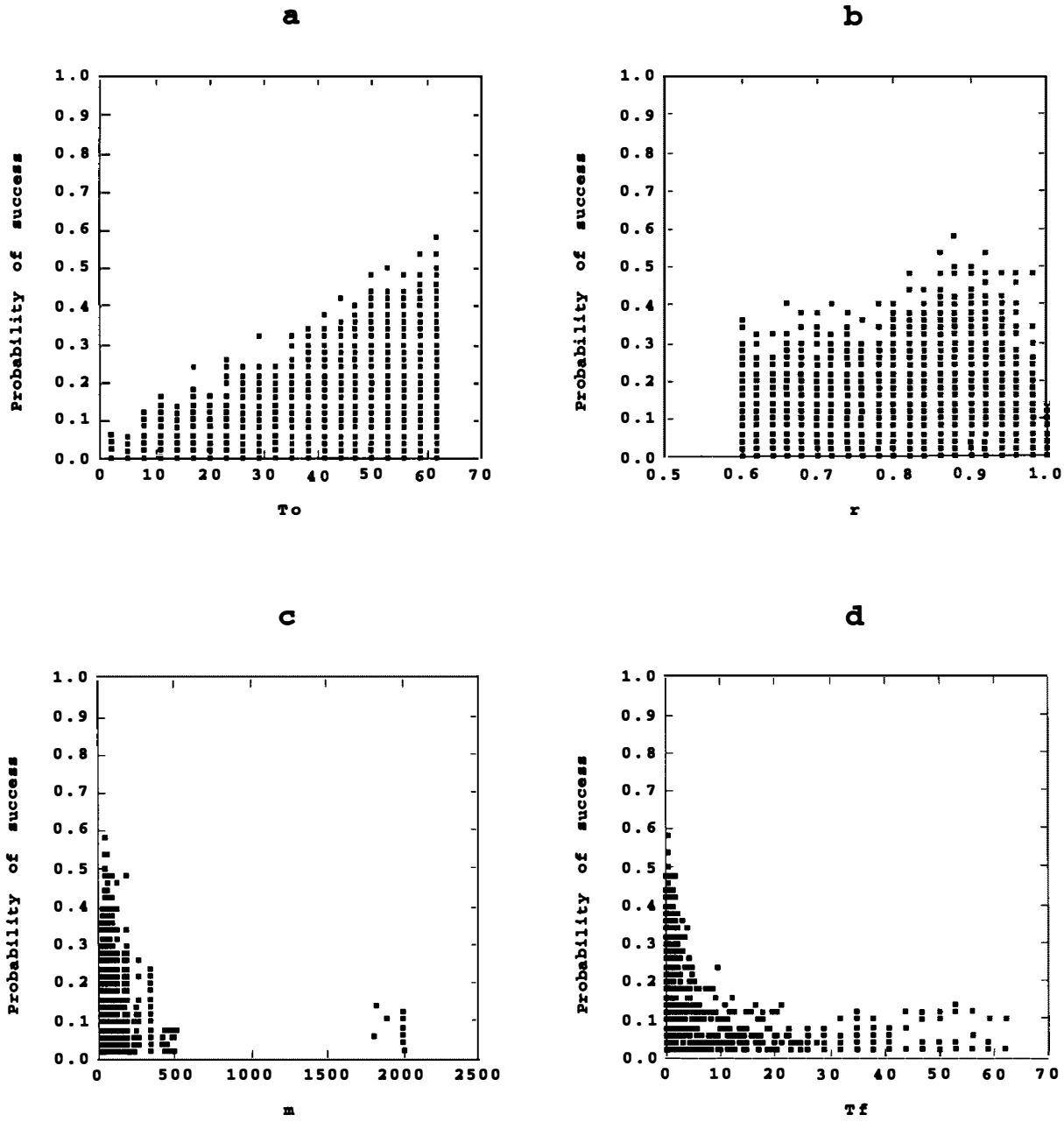


Fig. 12. Probability of successfully finding the global minimum of the Penalised Shubert function using the method of Corana et al. by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

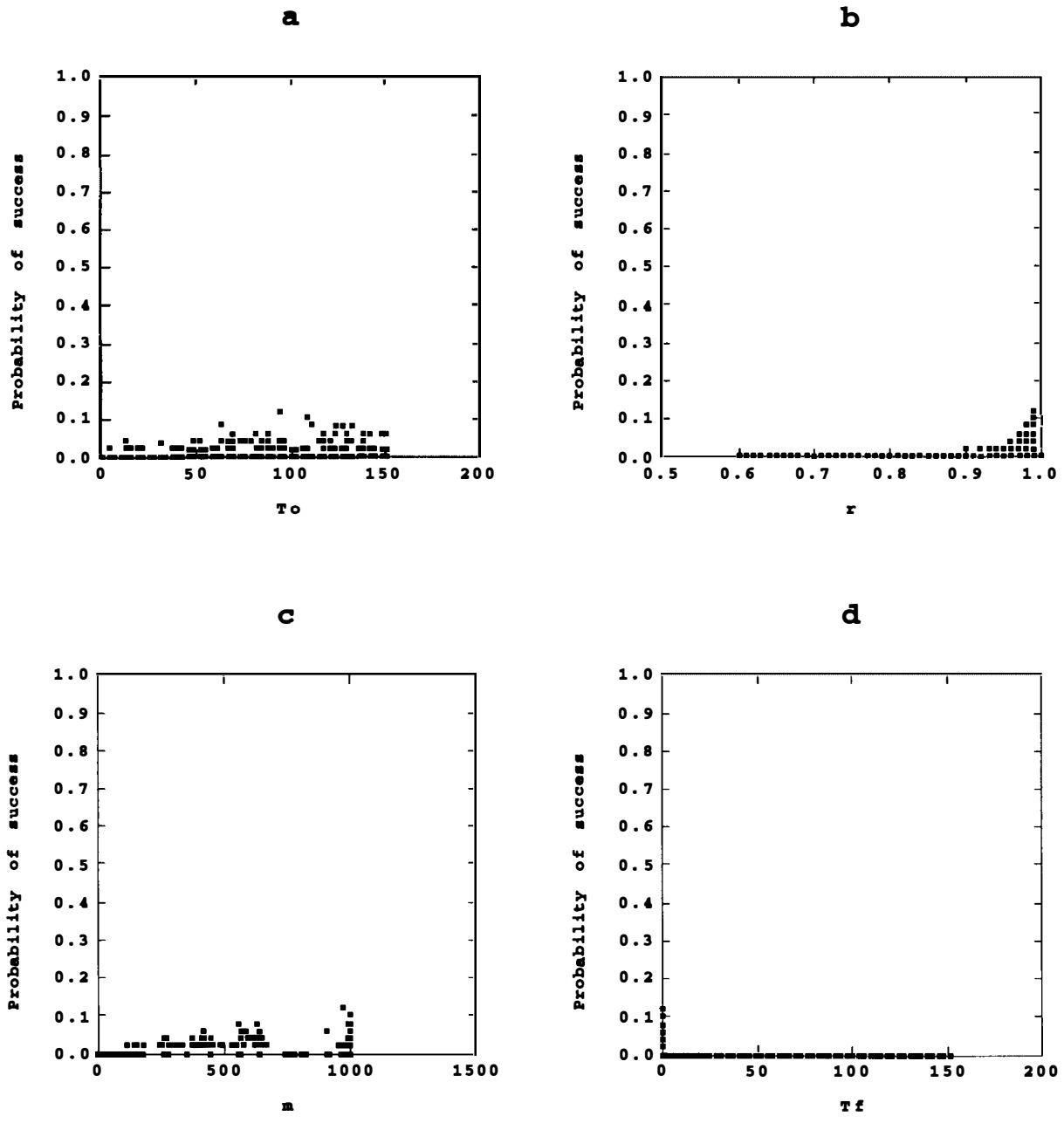


Fig. 13. Probability of successfully finding the global minimum of the Penalised Shubert function using the method of Vanderbilt & Louie by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

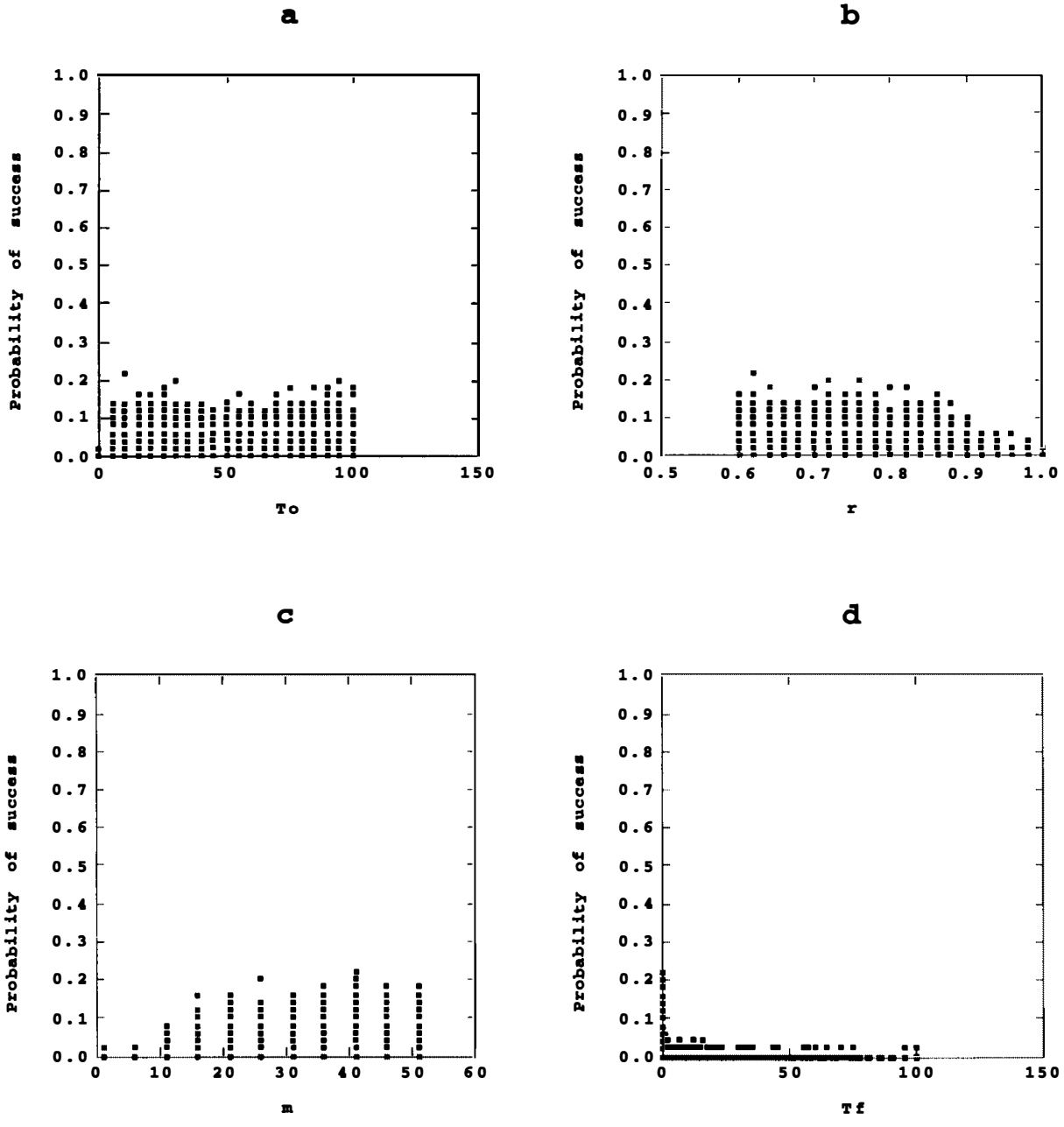


Fig. 14. Probability of successfully finding the global minimum of the Penalised Shubert function using the method of Press & Teukolsky with restarts by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

points) corresponds to the Nelder Mead simplex method with a mean probability of success of 0.0002 which is considerably lower than that obtained by SA. The algorithm is sensitive to the temperature reduction factor and $r < 0.85$ gives the best results for this function. Greater than 30 temperature reduction steps should be used to improve the probability of finding the global minimum. The algorithm is very sensitive to the final temperature which must be close to zero.

New Algorithm

The results for the new algorithm are shown in Figure 15. The probability of success is not sensitive to T^0. $T^0 = 0$ (the first column of points) corresponds to the Hooke and Jeeves pattern search method with a mean probability of success of 0.02 which is considerably lower than that obtained by SA. The performance of Hooke and Jeeves is considerably better than that obtained using Nelder and Mead for this function. The algorithm is insensitive to the temperature reduction factor and number of reduction steps. The algorithm is not very sensitive to the final temperature although $T^f \approx 0$ appears to be desirable. The points corresponding to $T^f = 0$ on Figure 15d with a low probability of success are due to the Hooke and Jeeves method. Since it is less sensitive to T^0, r, m and T^f this algorithm appears to be more robust than the other algorithms considered.

4.3.4 Discussion

The algorithms all performed reasonably on the Hartman function with flat valleys. The algorithm by Vanderblit and Louie (1984) performed poorly compared to the others on the Rastrigin function with 50 local minima in a lattice arrangement. All four algorithms performed poorly on the Penalised Shubert function where the local minima are deep, highly isolated, and their depths are not apparent except in a small volume near the core of each minimum.

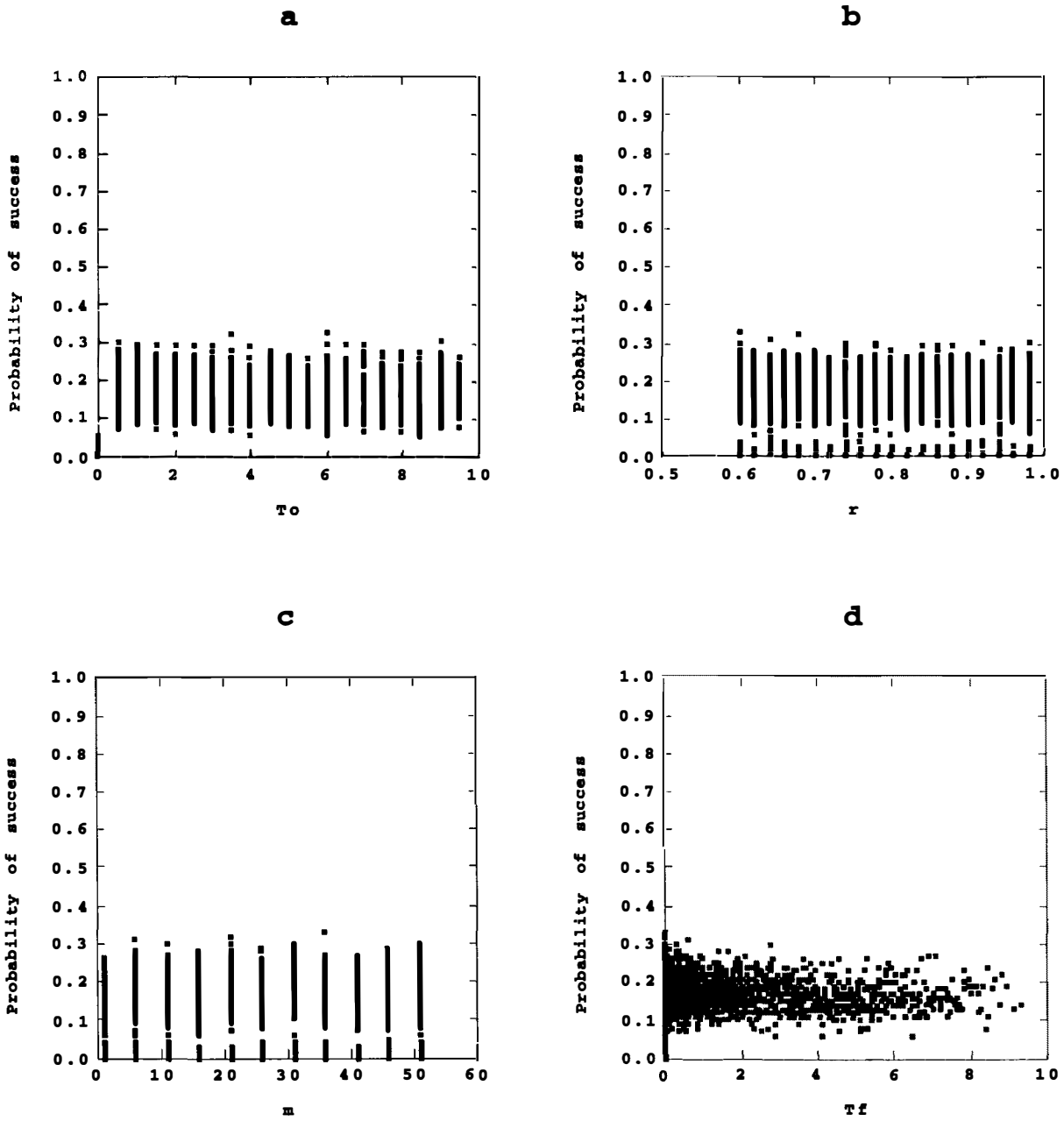


Fig. 15. Probability of successfully finding the global minimum of the Penalised Shubert function using the new method by a) Initial temperature, b) Reduction factor, c) Number of temperature reductions, d) Final temperature.

The findings using four different SA algorithms and three different test functions appears to offer some hope for practitioners working on real world problems. The choice of initial temperature and cooling schedule has been shown to determine the probability of success for the algorithms. The most appropriate initial temperature and cooling schedule is specific to both the problem and the SA algorithm used. It would be impossible to systematically attempt different initial temperatures and cooling schedules on a real world problem (the approach used was expensive in terms of computer time using simple test functions) and practitioners must rely on past experience or trial and error.

Based on these results it is possible to give the following general guidelines:

Initial Temperature

The choice of initial temperature is algorithm specific. The temperature must be sufficiently high to allow the algorithm to sample the function without becoming trapped in a local minimum. An initial temperature that is too high will waste computational effort.

Corana et al. suggest using a initial temperature the same order of magnitude as the standard deviation of the cost function. Corana et al. admit that this suggestion is based on experience with combinatorial SA problems and is likely to lead to temperatures that are too high wasting computational effort. For the Hartman function a sample of 200 function evaluations gave a standard deviation of 0.809. This initial temperature is higher than necessary for the algorithm by Corana et al. and would cause an excessively long cooling schedule wasting considerable computational effort. This initial temperature would be more suitable for the other three SA algorithms used for this study. The standard deviation of the cost function for the Rastrigin function is 0.849. This value is also higher than necessary for the algorithm by Corana et al. although the other three

algorithms considered would benefit from a initial temperature higher than the standard deviation. The standard deviation of the cost function for the Penalised Shubert function is 62. An initial temperature of 60 was found to give good results for this function using the method of Corana et al. This is also a reasonable choice for the algorithms by Vanderbilt and Louie and Press and Teukolsky although it is excessively high for the new method.

Dekkers and Aarts (1991) maintain that the initial temperature should be sufficiently large, such that approximately all transitions are accepted at this value. They suggest generating a number of trials m_0, and requiring that the initial acceptance ratio χ_0 defined as the ratio between the number of accepted transitions and the number of proposed transitions is close to 1. The initial value for T^0 is then obtained from the following expression

$$T^0 = \overline{\Delta f^+} \left(\ln \frac{m_2}{m_2 \chi_0 + (1-\chi_0) m_1} \right)^{-1} \tag{18}$$

where m_1 and m_2 denote the number of trials ($m_1 + m_2 = m_0$) with $(f(\mathbf{x}^k) - f(\mathbf{x}^k+\Delta\mathbf{x})) \geq 0$ and $(f(\mathbf{x}^k) - f(\mathbf{x}^k+\Delta\mathbf{x})) < 0$ respectively and $\overline{\Delta f^+}$ the average value of those Δf values for which $\Delta f > 0$ ($\Delta f = f(\mathbf{x}^k+\Delta\mathbf{x}) - f(\mathbf{x}^k)$).

Table 1. Calculated values for T^0 for the algorithm by Corana et al.

Test function	m_1	m_2	χ_0	$\overline{\Delta f^+}$	T^0
Hartman	55	245	0.323	1.59	2.14
Rastrigin	36	164	0.350	1.21	1.70
Penalised Shubert	8	192	0.045	136.92	55.49

Initial guesses for T^0 of 0.15, 0.4 and 0.5 were used for the Hartman, Rastrigin and Penalised Shubert functions respectively. The values shown in Table 1 were obtained

using the algorithm by Corana et al. The step size vector was set to [1, 1, 1] and the automatic step size adjustments were disabled so they would not change while the function was sampled. The calculation was then repeated using the new calculated values of T^0 shown in Table 1 as the initial guess. If the method was working correctly similar values for T^0 would be expected. The new estimates for T^0 were 22.8, 10.7 and 526.5 for the Hartman, Rastrigin and Penalised Shubert functions respectively. The estimates clearly depend on the initial guess used for T^0. The results obtained using the three test functions show that the method of Dekkers and Aarts is sensitive to the initial guess used for T^0 and may not give a reasonable estimate for T^0.

Temperature Reduction Factor

The algorithms by Corana et al., Vanderbilt and Louie and Press and Teukolsky were sensitive to the reduction factor and the appropriate value depended on the algorithm and test function being investigated. The new algorithm was not sensitive to the temperature reduction factor on the test functions used.

Final Temperature

The final temperature should be sufficiently low to allow the algorithm to settle in a valley containing hopefully the global minimum or at least a close to optimal solution. A final temperature close to zero was required for all four SA algorithms using the test functions. The performance of the algorithm by Press and Teukolsky deteriorated rapidly as the final temperature increased. This was also noted by Bates (1994).

Number of Temperature Reduction Steps

The number of temperature reduction steps used determines the number of function evaluations made. Increasing the number of temperature reductions increases the

probability of finding the global minimum until the point of diminishing returns is reached where increasing the number of temperature reduction steps (and function evaluations) does not significantly increase the probability of finding the global minimum. It was only possible to control the number of temperature reductions on the algorithm by Press and Teukolsky and the new algorithm.

Since the new algorithm is less sensitive to the initial temperature and choice of cooling schedule it appears to be more robust than the other algorithms considered.

4.4 Comparison of Simulated Annealing Algorithms

This section investigates the reliability and efficiency of four SA algorithms using the three test functions from section 4.1. The suitability of the SA algorithms for finding the global minimum of the test functions is discussed.

In order to have confidence in the results of an optimisation procedure we require that it has a small probability of failure p_f (out of 100 independent tests we expect that $100 \times p_f$ of them will fail). If we then re-run the procedure q times from q independent random locations the overall probability of failure will be

$$p_f(q) = p_f(1)^q \tag{19}$$

and tend to zero as q becomes large. Taking natural logarithms gives

$$\ln p_f(q) = q \ln p_f(1) \tag{20}$$

This simple strategy of repeating the search can increase the probability of finding the global minimum and is the basis for multistart methods. The number of restarts required to achieve an overall failure probability of $p_f(q)$ as given by (Duan et al., 1992) is

$$q = \frac{\ln p_f(q)}{\ln p_f(1)} \tag{21}$$

For low single start failure probabilities it does not take many restarts to rapidly decrease the overall failure probability. However, if the single start failure probability is high the number or restarts required rapidly increases.

The comparison of algorithms by computer simulation is a complicated task. A set of test problems must be selected which relate to the type of problems that are likely to be encountered in the real world. A metric that quantifies the algorithms must be decided on. The most suitable measures are the mean number of function evaluations required to find the global optimum within a specified precision, or the number of function evaluations required to find the global optimum with a given probability of success (see Box, 1965).

Comparisons of global optimisation methods in the literature have not been rigorous because of the effort involved. In most cases the authors have compared SA methods with other SA or optimisation methods without any attempt to find an appropriate initial temperature and cooling schedule. Interesting results on the sensitivity of different SA methods to the starting temperature, cooling schedule or step size have often been overlooked. Most authors (Vanderbilt and Louie, 1984; Brooks and Verdini, 1988; Benke and Skinner 1991; Deckers and Arts 1991; Butler and Slaminka, 1992) have given the number of function evaluations required by a SA algorithm to find the global minimum as well as the probability of finding the global minimum. Other authors (Corana, 1987; Mockus, 1989; Ratschek and Voller, 1990; Gunel and Yazgan, 1992) have only given the number of function evaluations and comparisons of their results with

those of others is generally not possible since the reliability of the method is not known. Often the test for convergence (stopping criteria) and tolerance used is not given.

The methods of Corana et al. (1987), Vanderbilt and Louie (1984), Press and Teukolsky (1991) and the new method described in section 3 were compared using the three standard test functions from section 4.1. The SA methods use different approaches and have different levels of sophistication. The maximum number of function evaluations for all four SA algorithms was set to 30,000.

4.4.1 Hartman Function

Algorithm by Corana et al. (1987)

The results for the algorithm by Corana et al. are shown in Figure 16a. The probability of success increases until about 10,000 function evaluations is reached; continuing beyond this point wastes computational effort since additional function evaluations do not improve the results. Here a probability close to one can be achieved providing the annealing schedule is well chosen.

The number of function evaluations required to achieve $p_f \leq 0.01$ is shown in Table 2 where f_{evals} is the number of function evaluations required for $p_f(1)$. For a real number z, $[z]$ (read "ceiling of z") is the smallest integer $\geq z$. Since the number of SA runs must be an integer value the number of function evaluations required for $p_f \leq 0.01$ is $[q] \bullet f_{\text{evals}}$.

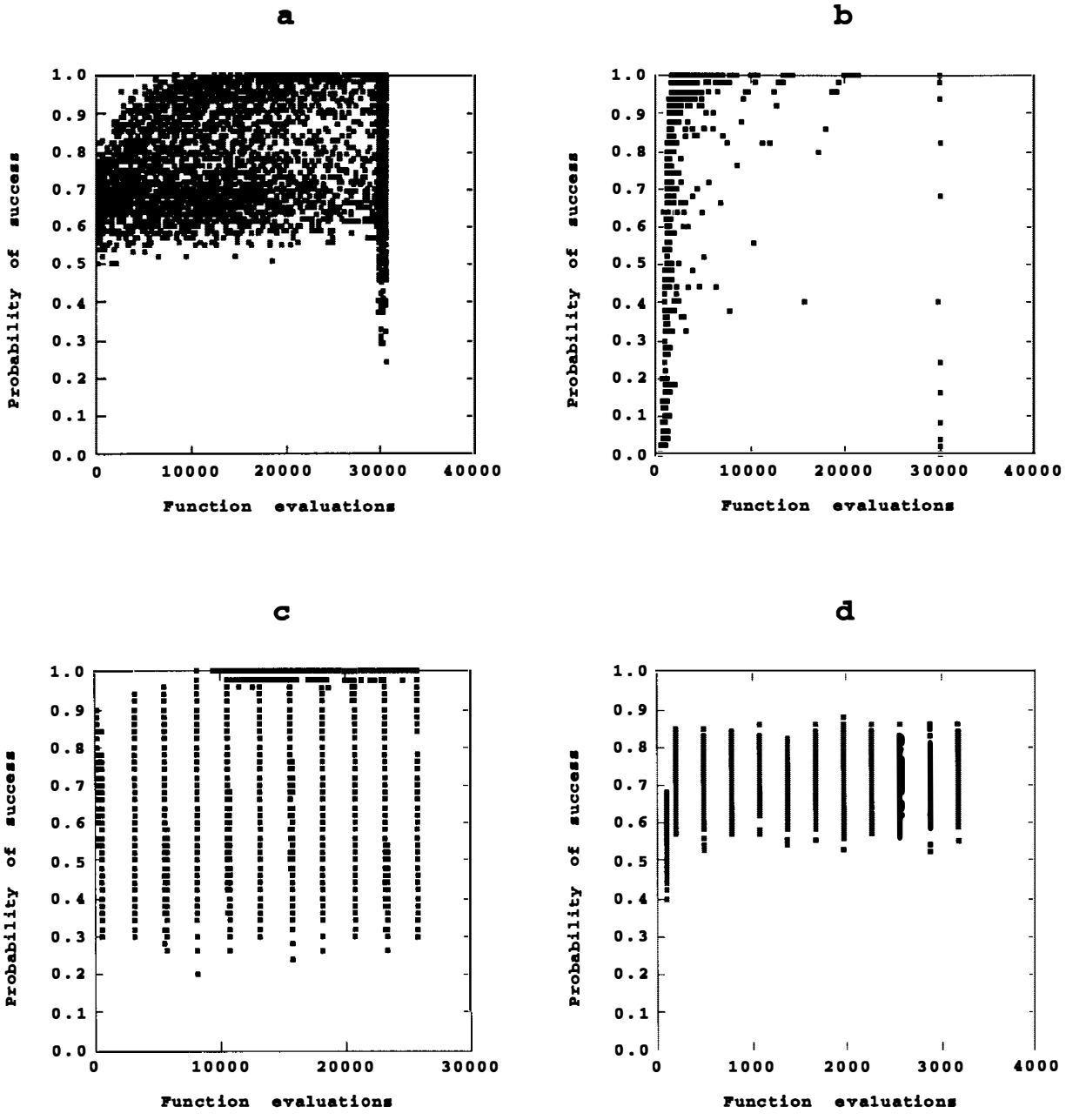


Fig. 16. Probability of successfully finding the global minimum of the Hartman function using the method of a) Corana et al., b) Vanderbilt & Louie, c) Press & Teukolsky with restarts, d) New algorithm.

Table 2. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
1,000	0.15	2.43	3,000
2,000	0.12	2.17	6,000
3,000	0.10	2.00	6,000
4,000	0.07	1.73	8,000
5,000	0.04	1.43	10,000
6,000	0.02	1.18	12,000
7,000	0.01	1.00	7,000
8,431	0.00		

As can be seen from Table 2 most efficient way to achieve $p_f \leq 0.01$ is to use three SA runs of 1,000 function evaluations.

Algorithm by Vanderbilt and Louie (1984)

The results for the algorithm by Vanderbilt and Louie are shown in Figure 16b. A probability of success of one was achieved with about 2000 function evaluations provided a reasonable cooling schedule was chosen. Vanderbilt and Louie reported that their algorithm required 1224 function evaluations to find the global minimum of the Hartman function with $p_f = 0.00$ although the tolerance they used was much coarser than that used for this study. This result can be compared to a probability of $p_f = 0.20$ obtained with 1250 function evaluations. The different tolerance used could explain the different results obtained. The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 3.

Table 3. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
1000	0.55	7.70	8000
1250	0.20	2.86	3750
1500	0.10	2.00	3000
1750	0.02	1.18	3500
2000	0.00		

Algorithm by Press and Teukolsky (1991) with restarts

The results for the algorithm by Press and Teukolsky are shown in Figure 16c. The probability of success increases until about 8,000 function evaluations is reached; continuing beyond this point wastes computational effort since additional function evaluations do not improve the results. Here a probability close to one can be achieved providing the annealing schedule is well chosen. The columns of points correspond to different numbers of steps in the cooling schedule (m). Each column includes a full range of reduction factors (r). The cluster of points with probability close to one correspond to cooling schedules ($r < 0.84$; $m > 21$) where SA is working effectively and the Nelder Mead local search method is being used to find the bottom of the valley containing the global minimum. The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 4.

Table 4. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
588	0.16	2.51	1,764
3,088	0.06	1.64	6,176
5,735	0.04	1.43	11,470
8,088	0.00		

New Algorithm

The results for the new algorithm are shown in Figure 16d (note the different scale used on the x-axis for this method). The columns of points correspond to different numbers of steps in the cooling schedule (m). Each column includes a full range of reduction factors (r). The first column of points ($T^0 = 0$) corresponds to the Hooke and Jeeves simplex method with a lower probability of success than SA. A probability of success of 0.85 can be achieved with about 200 function evaluations provided a reasonable initial temperature and cooling schedule is chosen. The probability of success does not appear to increase as more function evaluations are made. This indicates a better strategy for increasing the probability of finding the global minimum is to use several runs of about 200 function evaluations. The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 5.

Table 5. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
200	0.15	2.43	600

Discussion

The new algorithm is more efficient than the other algorithms investigated and is able to find the global minimum of the Hartman function with $p_f \leq 0.01$ using only 600 function evaluations if three separate SA runs are used. Brooks and Verdini (1988) found the global minimum in 455 (310 global evaluations plus 145 local evaluations) function evaluations with a probability of 0.78 using a generalised SA method. Using Eqn (21) it would take ($q = 3.04$) four runs of 455 function evaluations or 1820 function evaluations to achieve $p_f \leq 0.01$. The SA algorithm of Vanderbilt and Louie (1984) required 1224 function evaluations to find the global minimum with $p_f = 0.00$ although the convergence criteria they used was much coarser than that used for this study. Butler

and Slaminka (1992) used the Sniffer global optimisation algorithm by Rogers and Donnelly (1989) and reported that it required 534 function evaluations to find the global minimum with $p_f = 0.01$. Mockus (1989) reported a one-step Bayesian method took 513 function evaluations to find the global minimum although the probability was not given. The new algorithm compares favourably with these results.

4.4.2 Rastrigin Function

Algorithm by Corana et al. (1987)

The results for the algorithm by Corana et al. are shown in Figure 17a. The probability of success increases until about 25,000 function evaluations is reached; continuing beyond this point wastes computational effort since additional function evaluations do not improve the results. Here a probability of success of about 0.72 can be achieved providing the annealing schedule is well chosen. The points corresponding to about 30,000 function evaluations with a probability of greater than 0.7 were obtained from a Monte Carlo search ($T^0 > 0.18$; $r = 1$). Although the global minimum was found the SA algorithm could not converge because the temperature was too high. The results show that a Monte Carlo search can find the global minimum with a high probability of success in 30,000 function evaluations and possibly less although this was not investigated. The number of function evaluations required to achieve $p_f \leq 0.01$ is shown in Table 6.

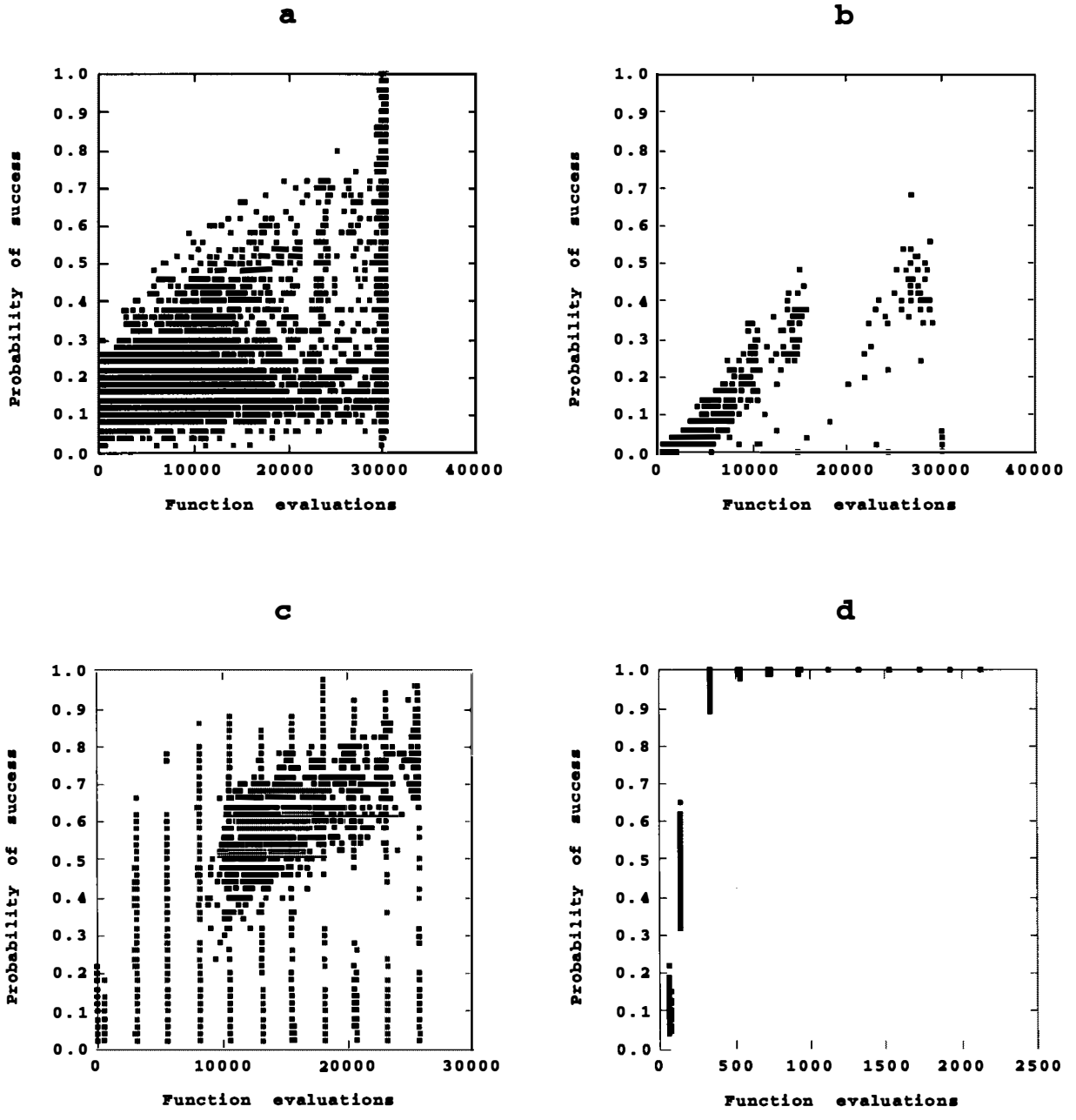


Fig. 17. Probability of successfully finding the global minimum of the Rastrigin function using the method of a) Corana et al., b) Vanderbilt & Louie, c) Press & Teukolsky with restarts, d) New algorithm.

Table 6. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
2,500	0.65	19.69	50,000
5,000	0.58	8.45	45,000
10,000	0.46	5.93	60,000
15,000	0.38	4.76	75,000
20,000	0.32	4.04	100,000
25,000	0.28	3.62	100,000
30,000	0.26	3.42	120,000

Algorithm by Vanderbilt and Louie (1984)

The results for the algorithm by Vanderbilt and Louie are shown in Figure 17b. A probability of success of over 0.5 could be achieved provided a reasonable cooling schedule was chosen. The cluster of points on the right of the plot correspond to $r = 0.99$ with the highest probability of success although a large number of function evaluations are required The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 7.

Table 7. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
5,000	0.86	30.53	155,000
7,500	0.76	16.78	127,500
10,000	0.66	11.08	120,000
20,000	0.52	7.04	160,000
30,000	0.44	5.61	210,000

Algorithm by Press and Teukolsky (1991) with restarts

The results for the algorithm by Press and Teukolsky are shown in Figure 17c. The probability of success increases as more function evaluations are used although about 11,000 function evaluations appears to be the point of diminishing returns. The columns of points correspond to different numbers of steps in the cooling schedule (m). Each column includes a full range of reduction factors (r). $T^0 = 0$ (the first column of points) corresponds to the Nelder Mead simplex method. The cluster of points corresponds to cooling schedules ($r < 0.84$; $m > 21$) where SA is working effectively and the Nelder Mead local search method is being used to locate the bottom of the valley containing the global minimum. The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 8.

Table 8. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
588	0.82	23.21	14,112
3,088	0.34	4.27	15,440
5,735	0.22	3.04	22,940
8,088	0.20	2.86	24,264
10,735	0.12	2.17	32,205

New Algorithm

The results for the new algorithm are shown in Figure 17d. The columns of points correspond to different numbers of steps in the cooling schedule (m). Each column includes a full range of reduction factors (r). The first column of points ($T^0 = 0$) corresponds to the Hooke and Jeeves pattern search method with a lower probability of success than SA. A probability of success of close to one can be achieved with about

329 function evaluations regardless of the choice of cooling schedule. The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 9.

Table 9. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
138	0.36	4.70	690
329	0.00		

Discussion

The new algorithm is more efficient than the other algorithms investigated and is able to find the global minimum of the Rastrigin function with $p_f \leq 0.01$ using only 329 function evaluations in one run of the SA program. This compares favourably with the centroid algorithm used by Benke and Skinner (1991) which took advantage of the symmetry of the function to find the global minimum in about 380 function evaluations. Polovinkin (1981) reported that a Monte Carlo search required 5,917 function evaluations and multistart gradient methods 556 function evaluations to find the global minimum. Unfortunately probability of finding the global minimum for these results was not reported.

4.4.3 Penalised Shubert Function

Algorithm by Corana et al. (1987)

The results for the algorithm by Corana et al. are shown in Figure 18a. The probability of success increases until about 15,000 function evaluations is reached; continuing beyond this point wastes computational effort since additional function evaluations do not improve the results. The points corresponding to about 30,000 function evaluations with

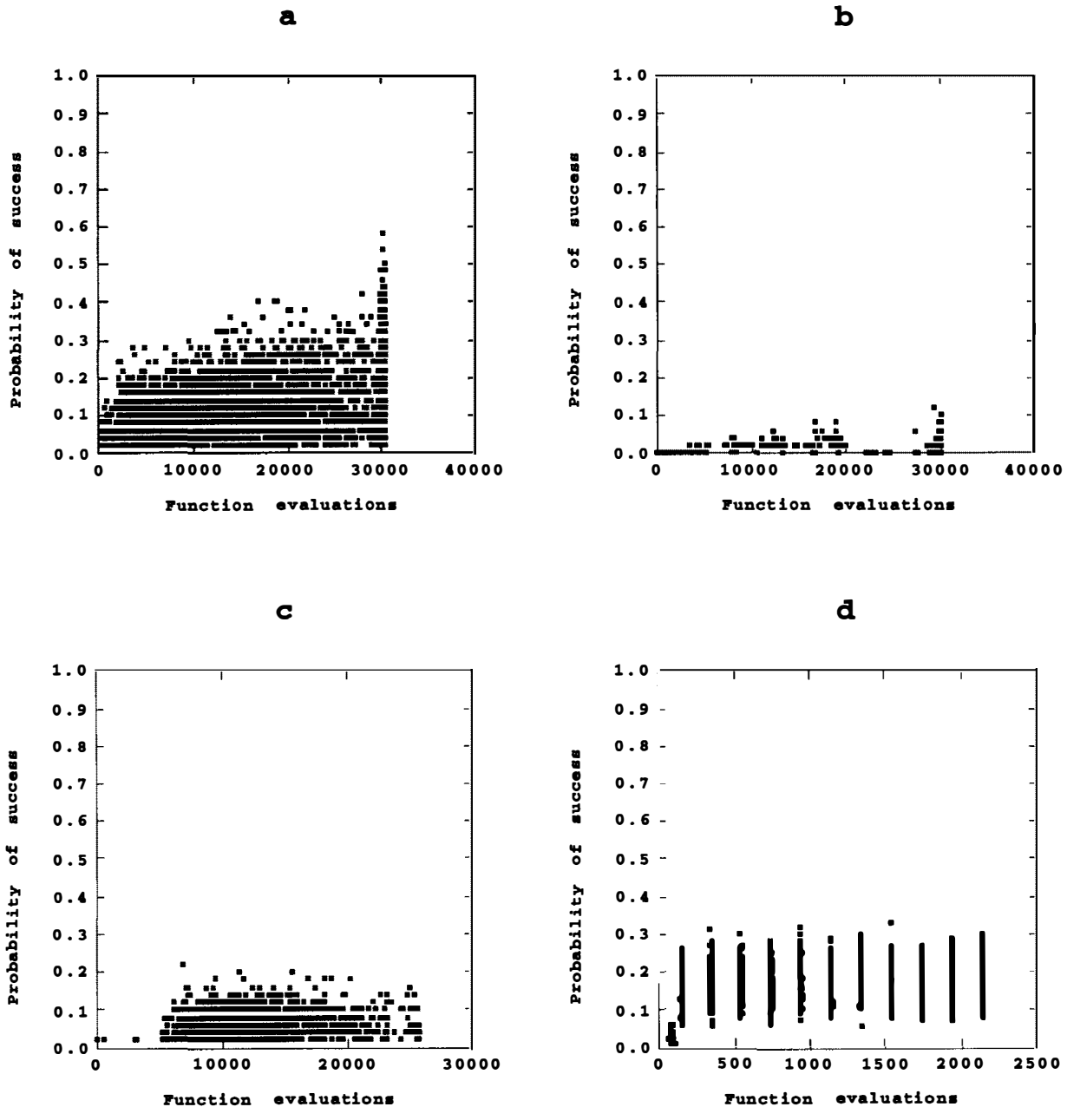


Fig. 18. Probability of successfully finding the global minimum of the Penalised Shubert function using the method of a) Corana et al., b) Vanderbilt & Louie, c) Press & Teukolsky with restarts, d) New algorithm.

a probability of greater than 0.43 were obtained from a cooling schedule where the algorithm was approaching a Monte Carlo search ($T^0 > 50$; $0.82 \leq r \leq 0.98$). Although the global minimum was found the SA algorithm could not converge because the temperature was too high. The number of function evaluations required to achieve $p_f \leq 0.01$ is shown in Table 10.

Table 10. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
2,500	0.78	18.53	47,500
5,000	0.72	14.02	75,000
10,000	0.70	12.91	130,000
15,000	0.64	10.32	165,000
20,000	0.62	9.63	200,000
25,000	0.64	10.32	275,000
30,000	0.58	8.45	270,000

Algorithm by Vanderbilt and Louie (1984)

The results for the algorithm by Vanderbilt and Louie are shown in Figure 18b. This algorithm was not suited to this function. The strategy used to set the step size vector to enable the method to take longer steps in the most profitable direction down the axis of long narrow valleys was not appropriate for this function where the minima are deep, highly isolated, with their depths are not apparent except in a small volume near the core of each minimum. The cluster of points on the right of the plot correspond to $0.99 \leq r \leq 1.00$ with the highest probability of success although a large number of function evaluations were required The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 11.

Table 11. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
5,000	0.98	227.95	1,140,000
10,000	0.96	112.81	1,130,000
20,000	0.92	55.23	1,120,000
30,000	0.88	36.02	1,110,000

Algorithm by Press and Teukolsky (1991) with restarts

The results for the algorithm by Press and Teukolsky are shown in Figure 18c. The probability of success increases as more function evaluations are used although about 11,000 function evaluations appears to be the point of diminishing returns. The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 12.

Table 12. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
2,500	0.98	227.95	557,500
5,000	0.98	227.95	1,140,000
7,500	0.84	26.41	202,500
10,000	0.82	23.21	240,000
15,000	0.80	20.64	315,000

New Algorithm

The results for the new algorithm are shown in Figure 18d. The columns of points correspond to different numbers of steps in the cooling schedule (m). Each column includes a full range of reduction factors (r). The first column of points ($T^0 = 0$) corresponds to the Hooke and Jeeves pattern search method with a lower probability of

success than SA. A probability of success of about 0.3 can be achieved with about 363 function evaluations with a good choice of initial temperature and cooling schedule. The number of function evaluations required to achieve $p_f \leq 0.01$ for this method is given in Table 13.

Table 13. Number of function evaluations required for $p_f \leq 0.01$

f_{evals}	$p_f(1)$	q	$[q] \bullet f_{evals}$
163	0.74	15.29	2,608
363	0.69	12.41	4,719
563	0.70	12.91	7,319
763	0.72	14.02	11,445
963	0.68	11.94	11,556

Discussion

The new algorithm is more efficient than the other algorithms investigated and is able to find the global minimum of the Penalised Shubert function with $p_f \leq 0.01$ using only 2,608 function evaluations in sixteen runs of the SA program. The methods of Corana et al. and Press and Teukolsky require 47,500 and 202,500 function evaluations to obtain $p_f \leq 0.01$ respectively. The method by Vanderbilt and Louie is not suited to this function. Gunel and Yazgan (1992) report that their SA algorithm could find the global minimum of this function in 140 function evaluations although the probability was not given and the result could not be repeated despite many attempts.

4.5 Summary

The addition of a restart feature improved the performance of the SA algorithm by Press and Teukolsky on all of the three test functions used. Further use of this algorithm should include the restart feature.

The choice of initial temperature and cooling schedule has been shown to determine the efficiency and probability of success for the four different SA algorithms. The most appropriate initial temperature and cooling schedule is specific to both the SA algorithm and function used. Guidelines for the choice of initial temperature and cooling schedule have been given.

The choice of SA algorithm should depend on the nature of the function being investigated. The method by Corana et al. was able to find the global minimum of all three test functions with a reasonable probability of success although it was expensive in terms of the number of function evaluations required. The method of Vanderbilt and Louie performed well on the Hartman function with long flat valleys although it was not suited to Rastrigin function or the Penalised Shubert function with highly isolated minima. The method of Press and Teukolsky with restarts performed reasonably well on the Hartman and Rastrigin function, although was not well suited to the Penalised Shubert function. The method by Press and Teukolsky and the new method used a local search phase to find the bottom of the valley containing the global minimum. This feature improved the efficiency of these algorithms.

The reliability and efficiency of the new SA algorithm compares favourably with the three SA algorithms investigated and other global optimisation methods from the literature using three test functions that offer different challenges. The new algorithm is also extremely robust and not sensitive to the choice of initial temperature or cooling schedule. The exploration steps used by the new method effectively maintained a downhill bias. The new algorithm was able to both investigate the valley containing the base point using exploration steps and investigate distant points using pattern moves. The Metropolis criterion only acted on the pattern moves capable of moving the current base point from one valley to another. The local search phase incorporating step size reductions also contributed to the efficiency of this algorithm.

SA methods are superior to local search methods such as Nelder and Mead and Hooke and Jeeves which are not suited to global optimisation problems. The results show that it is more efficient to use several smaller runs of a SA program than one large one. The results also demonstrate the need to use multiple runs of a SA program since there is no guarantee that the global minimum will be found.

Due to deadlines imposed by the CSIRO Division of Water Resources with respect to the Climate Change Research Program the model calibration work described in chapter 6 was completed prior to the development of the new SA algorithm in chapter 3 and evaluation of current SA algorithms in chapter 4. For this reason the existing SA algorithm by Press and Teukolsky (1991) with restarts was used on the CRRM calibration problem rather than the new algorithm which is shown to be more reliable and efficient on the three test functions.

5. SFB Model

Boughton's (1984) SFB model (see also Nathan and McMahon, 1990; Chiew et al., 1993; Bates et al., 1994) estimates monthly streamflow using daily rainfall and potential evapotranspiration data as input. The model (Figure 19), named after the first three parameters, has a physical foundation, and Boughton provides recommended parameter values based on his experience applying the model to catchments on the east coast of Australia. The first three parameters are the surface storage capacity in millimeters (S), the daily infiltration capacity in millimeters per day (F) controlling the movement of water from the surface store to the lower store and the baseflow factor ($0 \le B \le 1$) which determines the proportion of the daily depletion of water in the lower store that appears as baseflow. Recommended values for S, F and B are related to physical catchment characteristics. The other parameters and their default values are as follows: the fraction of the surface storage capacity that does not drain to the lower store (NDC = 0.5); the maximum limiting rate of evapotranspiration (E_{max} = 8.9 mm d^{-1}); the lower store depletion factor (DPF = 0.005); and a baseflow threshold for the lower store (SDR_{max} = 25 mm) defining the depth of water in the lower store at which baseflow will cease. Nathan and McMahon reported that fitting the NDC and DPF parameters in addition to the original three parameters only gave a slight improvement in model fit on 33 catchments in the south east of Australia and that the model should be used as proposed by Boughton. I included all parameters in the fitting process to determine which processes were significant and to see if the fitted values differed from their default values.

The model operates on a daily time step. Incident rainfall begins to fill the surface store which is emptied by evapotranspiration at the potential rate (E_{pot}) when the non-drainage component is full. Otherwise, the actual evapotranspiration rate is given by

$$E_a = \min\left\{E_{\max} \cdot s / (NDC \cdot S); E_{pot}\right\} \qquad (22)$$

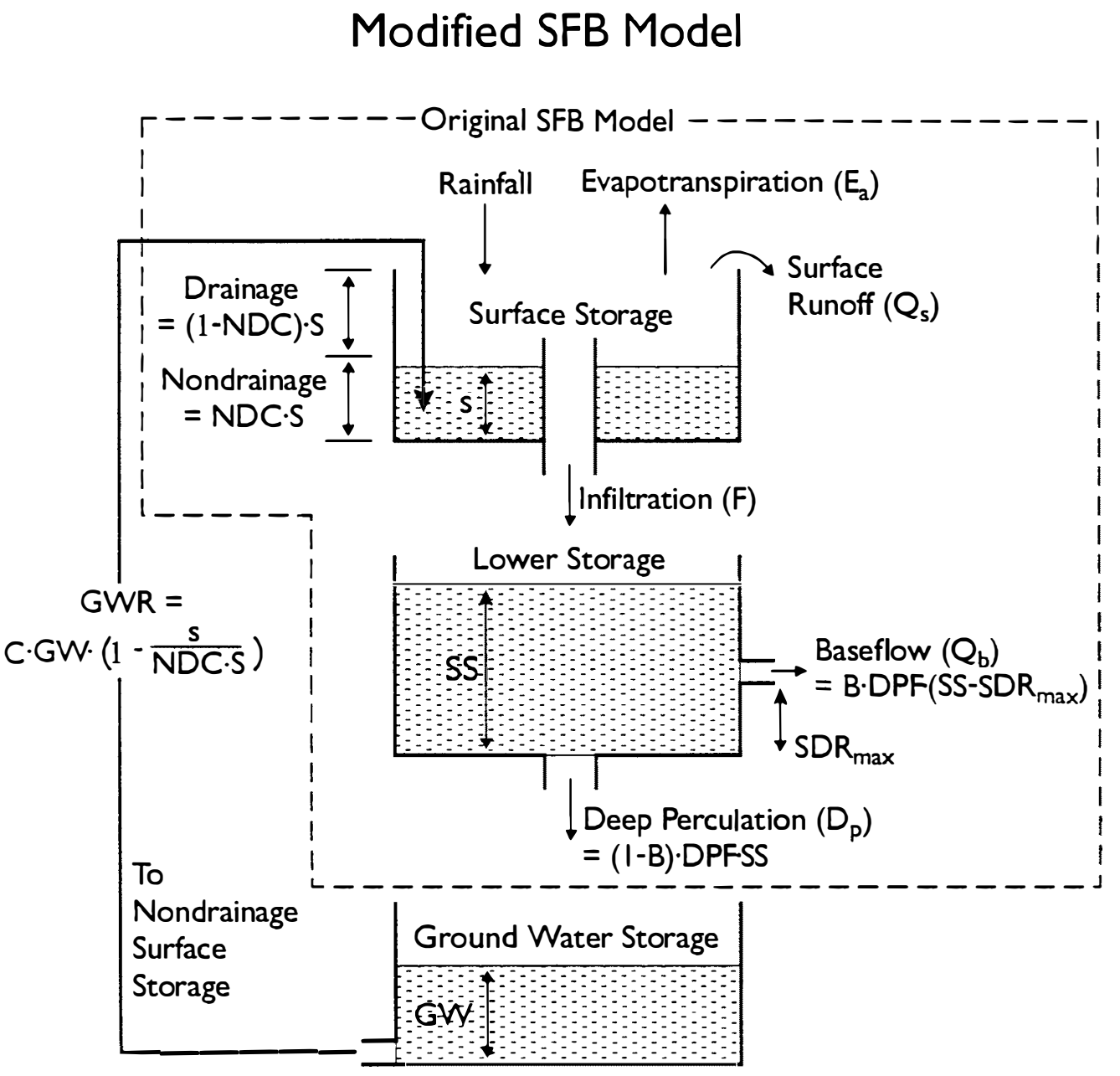


Figure 19. Structure of Boughton's (SFB) model showing groundwater replenishment modification.

where $s \geq 0$ is the depth of water in the non-drainage component of the surface store. Any water in excess of the non-drainage component of that store is subject to an infiltration rate of F mm d^{-1} until the drainage component of the store empties. Surface runoff (Q_s) occurs when the surface store is full and is calculated as

$$Q_s = P - F\tanh(P / F) \tag{23}$$

in which P is the rainfall excess remaining after the surface store is filled. The lower store is depleted by deep percolation (D_p) and baseflow (Q_b) which are defined by

$$D_p = (1 - B) \cdot DPF \cdot SS \tag{24}$$

$$Q_b = B \cdot DPF \cdot \left(SS - SDR_{\max}\right) \tag{25}$$

where $SS \geq 0$ is the depth of water in the lower store.

Since water is effectively lost from the catchment following deep percolation, the model, as proposed by Boughton does not always provide a closed water balance. The use of groundwater for evapotranspiration was considered to be an important process for some catchments particularly in the south west of Western Australia, where the regional groundwater supplies downslope and riparian vegetation during dry periods. The model was modified to enable this water to contribute towards evapotranspiration by adding a ground water store to the model and allowing a return from this store to the surface storage ($GWR \geq 0$) where

$$GWR = C \cdot GW \cdot \left(1 - \frac{s}{NDC \cdot S}\right) \tag{26}$$

and $GW \geq 0$ is the depth of water in the groundwater store. The process is regulated by the drainage coefficient and reverts to the original model when $C = 0$. Thus $C > 0$ when

$B \neq 1$ has the desirable effect of closing the modelled water balance so no water is lost from the system. The modification assumes that GWR occurs only during dry periods when $s < NDC \bullet S$ to avoid groundwater recycling to the lower storage via infiltration.

6. Calibration of the SFB Model using Simulated Annealing

Using SA I calibrated the modified version of Boughton's model on 25 unregulated catchments (Chiew and McMahon, 1993; Chiew and McMahon, 1994). These catchments were identified as benchmark catchments by the Australian Bureau of Meteorology (1991) and represent a range of climatic and physical characteristics for Australia. The calibration identifies the important model processes for these catchments by assigning values to well determined or sensitive parameters.

6.1 Parameter Estimation

6.1.1 Data

Historical streamflow, rainfall, and potential evapotranspiration records for Australian catchments provided as a supplement to Chiew and McMahon (1993) were used. The catchments are listed in Table 14. Potential evapotranspiration in the Chiew and McMahon data set was estimated by the method of Morton (1983). The model was, however, to be associated with stochastically generated data for climate change simulation, where the potential evaporation was to be estimated by the method of Priestley and Taylor (1972). Additional climate data were obtained from the Australian Bureau of Meteorology (1988) and used to estimate daily Priestley and Taylor potential evapotranspiration for model calibration. At least 8 years of record were available for each catchment.

Table 14 Rainfall and Streamflow Data

Catchment	Period of record	Catchment area (km^2)	Mean Annual Rainfall (mm)	Rainfall scale	Streamflow data	Mean Annual Streamflow (mm)	Global optimum adopted	λ_1	AR(1) parameter used	Number of parameters fitted	R^2	E
QUEENSLAND												
111105 Babinda Creek at The Boulders	1974 - 1989	39	5400	1.0	incomplete	4700	y	0.5	n	6	0.916	0.902
113004 Cochable Creek at Powerline	1974 - 1986	93	2400	1.2	complete	2100	y	0.5	y	6	0.828	0.811
118106 Alligator Creek at Allendale	1975 - 1989	69	1100	1.0	complete	480	y	0.4	y	7	0.764	0.746
120204 Broken River at Crediton	1965 - 1979	41	2100	1.0	complete	1000	y	0.5	y	6	0.951	0.950
145103 Cainbable Creek at Good Dam Site	1975 - 1987	41	900	-	complete	100	-	-	-	-	-	-
915001 Mitchell Grass at Richmond	1976 - 1988	3	450	1.0	complete	15	y	0.5	n	6	0.895	0.892
927001 Jardine River at Telegraph Line	1974 - 1989	2500	1700	1.0	incomplete	900	y	0.5	n	4	0.712	0.642
NEW SOUTH WALES												
206001 Styx River at Jeogla	1979 - 1986	163	1300	1.0	complete	450	y	0.5	n	7	0.945	0.939
210022 Allyn River at Halton	1977 - 1984	205	1200	1.2	incomplete	350	y	1.0	n	7	0.953	0.951
215004 Corang River at Hockeys	1980 - 1989	166	800	1.25	incomplete	330	y	0.5	n	7	0.692	0.495
401554 Tooma River above Tooma Reservior	1971 - 1979	114	1700	-	complete	1400	-	-	-	-	-	-
412093 Naradhan Creek at Naradhan	1978 - 1988	44	450	1.0	incomplete	2	y	0.8	n	6	0.925	0.888
420003 Belar Creek at Warkton	1973 - 1984	133	1100	0.8	incomplete	110	y	0.5	n	7	0.785	0.747
VICTORIA												
222213 Suggan Buggan River at Suggan Buggan	1972 - 1985	357	800	-	complete	150	-	-	-	-	-	-
227219 Bass River at Loch	1974 - 1985	52	1100	1.0	complete	330	y	0.5	n	7	0.927	0.925
238208 Jimmy Creek at Jimmy Creek	1970 - 1989	23	650	1.0	complete	160	n	2.0	n	7	0.608	0.593
401212 Nariel Creek at Upper Nariel	1977 - 1987	252	1200	1.15	complete	480	y	1.0	n	6	0.856	0.856
403218 Dandongadale River at Matong North	1974 - 1984	182	1300	1.0	complete	380	y	0.5	y	6	0.893	0.891
TASMANIA												
307001 Davey River D/S Crossing River	1974 - 1990	686	2100	1.3	incomplete	2000	y	0.5	n	6	0.918	0.902
315006 Forth River U/S Lemonthyme	1974 - 1985	311	2000	1.0	complete	1500	y	0.5	y	6	0.922	0.919
SOUTH AUSTRALIA												
503502 Scott Creek at Scotts Bottom	1970 - 1985	27	950	1.0	complete	130	n	0.5	y	6	0.692	0.686
505517 North Para River at Penrice	1978 - 1989	118	550	1.0	incomplete	50	y	0.6	n	5	0.880	0.875
509503 Kanyaka Creek at Old Kanyaka	1978 - 1989	180	300	-	complete	2	-	-	-	-	-	-
WESTERN AUSTRALIA												
612005 Stones Brook at Mast View	1974 - 1984	15	1000	1.0	complete	120	y	0.4	n	6	0.841	0.804
616065 Canning River at Glen Eagle	1977 - 1987	544	800	1.0	incomplete	20	y	0.5	n	7	0.782	0.774
701003 Nokanena Brook at Woottachooka	1977 - 1986	229	400	1.0	incomplete	20	n	0.7	n	6	0.809	0.807
708009 Kanjenjie Creek tributory at Fish Pool	1974 - 1986	41	400	-	complete	100	-	-	-	-	-	-
809312 Fletcher Creek at Frog Hollow	1970 - 1980	30	650	-	incomplete	20	-	-	-	-	-	-

6.1.2 Optimisation Procedure

Optimal model parameter estimates were found by fitting modelled monthly streamflow to observed streamflow using an objective function. A residual sum of squares error model was used when residuals were assumed to be independent

$$S(\hat{\theta}) = \min_{\theta} \sum_{i=1}^{n} (q_i - \hat{q}_i)^2 \tag{27}$$

where $\hat{\theta} = \left[\hat{\theta}_1, ..., \hat{\theta}_p\right]^T$ is a vector containing the p parameters to be estimated and n is the number of monthly streamflow observations in the calibration period. A two parameter transformation (Box and Cox, 1964) of observed (Q) and modelled ($\hat{Q}$) runoff was used to stabilise the variance

$$q_i = \frac{(Q_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} \qquad i = 1,2,...,n \tag{28}$$

$$\hat{q}_i = \frac{(\hat{Q}_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} \qquad i = 1,2,...,n \tag{29}$$

where λ_1 and λ_2 are the Box-Cox transformation constants. An autoregressive error model was used when the residuals were assumed to follow an AR(l) scheme

$$S(\hat{\theta}) = \min_{\theta} \sum_{i=2}^{n} \left((q_i - \hat{q}_i) - \rho(q_{i-1} - \hat{q}_{i-1})\right)^2 \tag{30}$$

where the lag 1 autocorrelation coefficient (ρ) is

$$\rho = \frac{\sum_{i=2}^{n} (q_i - \hat{q}_i)(q_{i-1} - \hat{q}_{i-1})}{\sum_{i=2}^{n} (q_i - \hat{q}_i)^2} \tag{31}$$

Assuming the transformed observations satisfy the normal theory assumptions (i.e. are independently distributed with constant variance) for some unknown λ_1, the likelihood in relation to the original observations (**Q**) is given by the product of the probability density for the normal distribution and the Jacobian of the transformation

$$\frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}}\exp\left\{-\frac{S(\theta)}{2\sigma^2}\right\}J(\lambda_1;\mathbf{Q}) \tag{32}$$

where σ^2 is the variance and $J(\lambda_1; \mathbf{Q})$ the Jacobian of the transformation

$$J(\lambda_1;\mathbf{Q})=\prod_{i=1}^{n}\frac{\partial\hat{Q}_i}{\partial Q_i}=\prod_{i=1}^{n}Q_i^{\lambda_1-1}\text{, for all }\lambda_1 \tag{33}$$

Since the value of λ_1 that will maximise Equation (32) will also maximise the natural logarithm of the function Equation (33) may be written as follows where the parameter values and variance must be estimated

$$L_{\max}(\lambda_1,\hat{\theta})=-\frac{n}{2}\ln(2\pi\hat{\sigma}^2)-\frac{S(\hat{\theta})}{2\hat{\sigma}^2}+\ln(J(\lambda_1;\mathbf{Q})) \tag{34}$$

Removing constant terms, substituting Equation (33), including λ_2 and re-arranging gives the maximum likelihood function $L_{max}(\lambda_1, \lambda_2, \hat{\theta})$ with respect to λ_1, λ_2 and the model parameters

$$L_{\max}(\lambda_1,\lambda_2,\hat{\theta})=(\lambda_1-1)\sum_{i=1}^{n}\ln(Q_i+\lambda_2)-\frac{n}{2}\ln\left(\frac{S(\hat{\theta})}{n}\right) \tag{35}$$

An initialisation period of three months was used to remove any effects of the initial store contents on the parameter estimates.

6.1.3 Simulated Annealing

The SA algorithm by Press and Teukolsky (1991) (or see Press et al., 1992) was used. The SA algorithm could bypass a local minimum in the search for a more global one by always accepting steps corresponding to a decrease in $S(\hat{\theta})$ and occasionally accepting a step corresponding to an increase in $S(\hat{\theta})$ by means of a stochastic acceptance criterion. The probability of accepting moves that increase $S(\hat{\theta})$ slowly decreases to zero as the method progresses.

The algorithm was constrained to remain in the prespecified parameter domain of plausible parameter values using a penalty step function that returned a large value to the annealing program when a model evaluation with parameter values outside the domain was requested. A restart when the temperature (T) has been reduced by a factor of three was found to be highly beneficial. The restart replaced a vertex of the simplex with the best point encountered when that point was not currently in the simplex. Without this modification the program tended to converge to a local rather than the global minimum.

The search terminates when the best function evaluation found is a vertex of the simplex and both the following criteria are satisfied

$$\frac{2\left|S(\theta)^H - S(\theta)^L\right|}{\left|S(\theta)^L\right| + \left|S(\theta)^H\right|} < \sqrt{\varepsilon} \tag{36}$$

$$\frac{\left|\theta_j^{(k)} - \theta_j^L\right|}{\max\left(\left|\theta_j^L\right|, \theta_j^0\right)} < \sqrt[3]{\varepsilon} \qquad j = 1, \ldots, p; \quad k = 1, \ldots, p+1 \tag{37}$$

where the superscripts H and L denote the vertices of the simplex with the highest and lowest function values, θ_j^0 is the initial estimate for the j^{th} parameter, $\theta_j^{(k)}$ is the j^{th}

parameter for the k^{th} vertex and ε is machine epsilon (the smallest positive number such that $1 + \varepsilon > 1$).

Cooling Schedule

The cooling schedule given as Equation (12) was used. A pragmatic approach was taken. For reasonably rapid cooling $m = 70$ with $r = 0.95$ were used, the slowest cooling schedule used $m = 500$ with $r = 0.994$. Beyond this, additional function evaluations (SFB model runs) did not appear to increase the probability of finding the global minimum. At each temperature 100 function evaluations were made.

The initial temperature was set to give acceptable perturbations while still maintaining a downhill bias. The SA algorithm used was not sensitive to the initial temperature and a change by a factor of two in the initial temperature made little difference to the probability of finding the global optimum.

6.1.4 Computation of Standard Errors and Correlation Matrix

Parameter standard errors were used to determine the significance of model parameter estimates and the precision to which they have been estimated. The correlation matrix provided a useful indicator of model parsimony and high correlation between two or more model parameters produce elongated valleys on the response surface where a change in the value of one parameter may be compensated by changes in one or more other parameters. It is difficult for optimisation methods to make any progress in these valleys which are likely to be the cause of convergence problems (Johnston and Pilgrim, 1976; Kuczera, 1988).

The construction of exact confidence regions and intervals for nonlinear least squares is much more difficult than for linear least squares problems. Approximate techniques are generally used. Linearization methods (Donaldson and Schnabel, 1987) assume that the nonlinear function can be approximated by a linear function at the solution. The linear theory covariance matrix is estimated as

$$\hat{\mathbf{V}} = \sigma^2\left(\mathbf{J}(\hat{\theta})^T\Omega^{-1}\mathbf{J}(\hat{\theta})\right)^{-1} \tag{38}$$

where $\mathbf{J}(\hat{\theta})$ is the $n \times p$ Jacobian matrix of the n fitted values evaluated at $\hat{\theta}$ with p fitted parameters:

$$\mathbf{J}_{i,j}(\theta) = \frac{\partial \hat{q}_i}{\partial \theta_j} \tag{39}$$

The estimated residual variance σ^2 is computed as

$$\sigma^2 = \frac{(q-\hat{q})^T\Omega^{-1}(q-\hat{q})}{n-p} \tag{40}$$

The covariance matrix of the errors Ω is computed using the lag 1 autocorrelation coefficient from Equation (31)

$$\Omega = \frac{\sigma^2}{1-\rho^2}\begin{bmatrix} 1 & \rho & \rho^2 & . & . & . & \rho^{n-1} \\ \rho & 1 & \rho & . & . & . & \rho^{n-2} \\ \rho^2 & \rho & 1 & . & . & . & \rho^{n-3} \\ . & . & . & & & & . \\ . & . & . & & & & . \\ . & . & . & & & & . \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & . & . & . & 1 \end{bmatrix} \tag{41}$$

The $100(1-\alpha)\%$ linear theory confidence region for θ is the set of values which satisfy

$$(\theta - \hat{\theta})^T \hat{\mathbf{V}}^{-1} (\theta - \hat{\theta}) \leq p F_{p,n-p,\alpha} \tag{42}$$

where $F_{p,n-p,\alpha}$ denotes the upper α quartile for the F distribution with p and $n-p$ degrees of freedom. Similarly, the $100(1-\alpha)\%$ linear approximation for θ_j ($j=1,\ldots,p$) consists of those values θ_j that satisfy

$$\left|\theta_j - \hat{\theta}_j\right| \leq \hat{\mathbf{V}}_{jj}^{1/2} t_{n-p,\alpha/2} \tag{43}$$

where $t_{n-p,\alpha/2}$ denotes the upper $\alpha/2$ quartile for the t distribution with $n-p$ degrees of freedom.

These confidence regions and intervals are readily evaluated. However, linearization methods assume that both the intrinsic curvature and parameter effects curvature at $\hat{\theta}$ are small compared to the critical value $(F_{p,\,n-p,\,.05})^{-1/2}$ (Bates and Watts, 1980). The similarity to the true confidence regions and intervals depends on how closely this assumption is attained. Donaldson and Schnabel (1987) discuss other methods of constructing confidence regions and intervals. They also note that the linear theory covariance matrix in Equation (38) is simpler, less expensive and more numerically stable than other methods of calculating $\hat{\mathbf{V}}$ based on the Hessian matrix with second-order terms.

6.1.5 Statistical Assumptions

Quantile plots were used to check the assumption of normality in the residuals. Autocorrelation and partial autocorrelation plots were used to detect violation of the assumption of independence. A first order autoregressive error model (Equation (30)) was used to satisfy this assumption where required.

If the assumptions that the residuals are normally and independently distributed with zero mean and constant variance are satisfied, the least squares and maximum likelihood parameter estimates would be the same. Often it was not possible to fully satisfy these assumptions. I found that setting λ_1 to maximise the maximum likelihood function in Equation (35) often did not provide a good fit to the data. The maximum likelihood fit placed more emphasis on the low-flow values with a smaller variance at the expense of the larger flows. A more acceptable fit was obtained by setting λ_1 so that modelled runoff matched observed runoff as closely as possible, and the residual variance was as constant as could be achieved.

Large departures from a constant error variance may lead to parameter estimates that are statistically inefficient and physically unrealistic. The assumption of constant residual variance proved to be the most difficult to satisfy. This is possibly because a larger error variance is associated with the higher flows, which is a characteristic of most rating curves used to convert depth observations to discharge (see Sorooshian and Dracup, 1980).

The residual variance was investigated using robust locally weighted regression (Cleveland, 1979). Fitted values were obtained using a local polynomial fit to the absolute value of the residuals $\left|Q_i - \hat{Q}_i\right|$ using weighted least squares where the weight is large for points close to the current point and small or zero otherwise. The robust fitting procedure prevents outlying points from distorting the smoothed points. The technique smoothes the residuals and provides a useful diagnostic for identifying trends in the data. I found that for some catchments the residual variance showed both increasing and decreasing tendencies, a problem which could not be resolved using the transformations in Equations (28) and (29).

6.1.6 Parameter Estimation Procedure

Starting with the conventional 3-parameter model, estimates for parameters S, F and B were obtained. The remaining model parameters were initially set to values recommended by Boughton (1984) and the drainage coefficient set to zero. These parameters were then fitted one at a time in the order NDC, DPF, SDR_{max}, E_{max}, and C. A parameter was included in the current and subsequent fitting processes if it reduced the residual variance (σ^2). If the inclusion of a parameter did not reduce the residual variance the value used by Boughton was retained. This enabled a parsimonious representation of the hydrological processes occurring on a catchment to be found since only the sensitive parameters were fitted. λ_1 was initially set to 0.5 and λ_2 was set to machine epsilon.

The estimate of the global minimum was accepted when at least four SA runs converged to the same minimum point in the parameter space. A useful but not conclusive check that the fitting process terminated at a global rather than local minimum was to ensure that the value of the objective function was reduced as more parameters were included.

6.2. Calibration Results

6.2.1 Catchment Selection

From the initial 28 catchments available in the Chiew and McMahon (1994) data set the Tooma River and Suggan Buggan catchments were not considered because of significant snow pack causing delayed runoff that Boughton's model was not capable of modelling. Cainbable Creek had an excellent streamflow record although unfortunately there were no rain gauges in the catchment or even in the same valley. The principle

rain gauge was two ridges to the east and major streamflow events occurred without corresponding rainfall events and major rainfall events occurred without matching hydrograph peaks. Simple scaling of the rainfall did not help, so model fitting was abandoned. Some of the arid-zone catchments had relatively few flow events and no base flow although the model was calibrated. Calibration was attempted on twenty-five catchments as shown in Table 15.

The model fit was unacceptable on the Kanyaka Creek, Kanjenjie Creek and Fletcher Creek catchments where the streamflow is ephemeral. There were not sufficient streamflow events on the Kanjenjie Creek, and Fletcher Creek catchments (14 and 13 respectively) to substantiate model calibration. Surprisingly, an acceptable fit was obtained for Naradhan Creek, which is also ephemeral with only 6 streamflow events. The rainfall data appeared to be inadequate for Fletcher Creek and Kanyaka Creek. Large flows occurred on Kanyaka Creek without corresponding rainfall events. The runoff at the gauging station for these catchments was 3 and 1 percent of rainfall, respectively. These results support the claim by Sorooshian et al. (1983) that it is not the length of data that is important but rather the information that the data contain and the quality of the data. Two measures of model adequacy are given in Table 14, the coefficient of determination (R^2) and the coefficient of efficiency (E) (Chiew and McMahon, 1994) defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{Q}_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(\hat{Q}_i - \overline{\hat{Q}})^2} \tag{44}$$

$$E = 1 - \frac{\sum_{i=1}^{n}(Q_i - \hat{Q}_i)^2}{\sum_{i=1}^{n}(Q_i - \overline{Q})^2} \tag{45}$$

where $\hat{Y}_i$ is obtained from the regression line relating modelled flows to observed flows and $\overline{Q}$ and $\overline{\hat{Q}}$ are the mean observed and modelled flow respectively. The coefficient of determination represents the proportion of the variation in the observed flows that is

Table 15 Parameter estimates for Boughton's model with standard errors in parenthesis

Catchment	S	F	B	E_{max}	SDR_{max}	NDC	DPF	C
QUEENSLAND								
111105 Babinda Creek at The Boulders	112(14)	20.0	1.0	8.9	0	0.56(0.08)	0.024(0.002)	0.0
113004 Cochable Creek at Powerline	44(10)	20.0	1.0	8.9	0	0.53(0.15)	0.021(0.003)	0.0
118106 Alligator Creek at Allendale	64(10)	12.6(3.8)	0.88(0.18)	7.9(4.5)	0	0.59(0.10)	0.020(0.003)	0.0
120204 Broken River at Crediton	102(8)	16.9(1.5)	0.56(0.03)	8.9	0	0.5	0.024(0.002)	1.55(2.51)
915001 Mitchell Grass at Richmond	241(34)	17.2(12.3)	0.31(0.06)	8.9	0	0.69(0.10)	0.465(0.168)	0.0
927001 Jardine River at Telegraph Line	82(12)	20.0	0.96(0.03)	8.9	25	0.5	0.012(0.001)	0.0
NEW SOUTH WALES								
206001 Styx River at Jeogla	94(8)	8.4(1.4)	0.56(0.03)	8.9	0	0.28(0.04)	0.044(0.005)	0.41(0.65)
210022 Allyn River at Halton	208(8)	8.9	0.38(0.02)	8.9	0	0.2	0.363(0.137)	1.63(3.70)
215004 Corang River at Hockeys	85(8)	1.9(0.7)	1.0	11.0(5.2)	0	0.83(0.05)	0.007(0.002)	0.0
412093 Naradhan Creek at Naradhan	65(1)	5.1(0.1)	0.0	8.9	0	0.5	0.141(1.109)	0.01(0.01)
420003 Belar Creek at Warkton	114(4)	6.7(1.2)	0.15(0.02)	8.9	0	0.54(0.03)	0.059(0.013)	0.37(0.30)
VICTORIA								
227219 Bass River at Loch	122(5)	1.8(0.1)	0.89(0.06)	8.9	88(3)	0.62(0.02)	0.064(0.015)	1.58(11.22)
238208 Jimmy Creek at Jimmy Creek	98(11)	4.0(0.6)	0.66(0.05)	8.9	0	0.63(0.07)	0.031(0.008)	1.79(3.67)
401212 Nariel Creek at Upper Nariel	259(16)	8.9(0.6)	0.65(0.03)	8.9	0	0.5	0.022(0.002)	0.08(0.09)
403218 Dandongadale River at Matong North	253(10)	3.6(0.1)	0.44(0.03)	8.9	0	0.5	0.024(0.003)	0.05(0.02)
TASMANIA								
307001 Davey River D/S Crossing River	26(8)	19.9(4.8)	0.85(0.01)	8.9	0	0.5	0.315(0.048)	2.65(0.27)
315006 Forth River U/S Lemonthyme	23(3)	2.7(0.4)	1.0	8.9	0	0.56(0.12)	0.018(0.004)	0.0
SOUTH AUSTRALIA								
503502 Scott Creek at Scotts Bottom	90(7)	6.2(0.6)	0.22(0.02)	8.9	5(2)	0.5	0.080(0.016)	1.00(0.87)
505517 North Para River at Penrice	164(4)	0.9(0.1)	0.41(0.03)	8.9	0	0.5	0.120(0.025)	0.0
WESTERN AUSTRALIA								
612005 Stones Brook at Mast View	222(7)	4.3(0.1)	0.24(0.02)	8.9	38(7)	0.5	0.028(0.002)	0.80(1.22)
616065 Canning River at Glen Eagle	345(8)	3.8(0.1)	0.06(0.004)	8.9	0	0.54(0.01)	0.082(0.013)	0.84(0.39)
701003 Nokanena Brook at Woottachooka	110(7)	6.8(1.7)	0.22(0.01)	8.9	25	0.26(0.04)	0.037(0.003)	0.53(0.80)

accounted for by the line of best fit. The coefficient of efficiency expresses the proportion of the variation in the observed flow that can be accounted for directly by the model.

6.2.2 Calibration

Simulated Annealing Algorithm

The SA algorithm found the same estimate of the global minimum using the maximum number of parameters calibrated for a catchment on 77 percent of SA runs. The SA algorithm required more function evaluations (about 7,000 for most catchments) than that of the Shuffled Complex Evolution (SCE-UA) method of Duan et al. (1992) to calibrate the 6-parameter model SIXPAR. However, this is not a reasonable comparison since the model, data sets, termination criteria and time step used in the objective function differ. The results of Bates (1994) and Duan et al. (1992) indicate that both SA and SCE-UA achieve about three times the probability of finding the global minimum than a multistart method using the same number of function evaluations. The SA method is simple to apply and would benefit from the application of several runs using a smaller number of function evaluations. The results also demonstrate the need to use multiple SA runs during the estimation process.

Exchangeable optima with similar objective function values were found for the Canning River, Kanyaka Creek and Allyn River catchments. For these catchments the SA algorithm had difficulty choosing between optima that were very similar. The catchments required a cautious approach since a slight change in the objective function such as changing λ_1 in Equations (28) and (29) or the introduction of an AR(1) model are likely to change the global optimum from one local minimum to another (see case study in section 6.3.2).

Model Parameters

A preliminary assessment of rainfall and runoff accumulations revealed a number of catchments where evaporation was negligible or excessive, pointing to errors in rainfall data in particular. These catchments often had strong rainfall gradients and so a scaling factor was introduced into six catchments as indicated in Table 14. All catchments with high rainfall and high runoff except the Forth River in Tasmania show high values for the infiltration parameter F and B equal or close to 1.0 in Table 15. Here, the model acts as a simple linear reservoir between net rainfall and streamflow, as it also does on the Forth River. The special case of the Mitchell Grass catchment in Western Queensland has high values for S, NDC and F. This is consistent with the deep cracking clays which only provide runoff after a succession of very large daily rainfalls. Surface runoff is rarely achieved and groundwater never emerges.

A closed water balance was obtained for all catchments except Alligator Creek, Mitchell Grass, Jardine River, and North Para River where the groundwater recirculation modification did not improve the model fit. The closed water balance allows the modelled groundwater levels in the catchment to carry over between seasons and years. Open water balances discharge all of the deep groundwater outside the catchment.

Boughton (1984) gives recommended values for three parameters of the model S, F and B based on catchment characteristics. The SA estimates for these parameters at the global minimum were not consistent with the catchment characteristics for five catchments. The original estimate of S for Scott Creek (S = 166mm) was considered excessively high for a catchment where the vegetation is predominantly grass. For Jimmy Creek the original estimate of S (S = 51mm) was considered low for a eucalypt woodland and the estimate of F (F = 8.2mm) excessively high for thick sandy surface

soils with highly impermeable clay subsoils. The infiltration rate for Nokanena Brook (F = 13.8mm) was considered excessively high for the duplex soils on this catchment. For these catchments the model did not appear to be behaving as intended and a near-to-optimal solution consistent with physical characteristics was found. The estimates of S for the two Tasmanian catchments (Davey River and Forth River) were not consistent with the physical characteristics of the catchments. Larger values for S were expected given the predominant tussocky grasses and graminoids on the Davey River catchment and forest vegetation on the Forth River catchment. The model fit for both catchments at the global minimum was excellent and these parameters were adopted since no local minimum that was consistent with catchment characteristics could be found. Since the conceptual and physical basis of the model does not apply to these catchments the parameter estimates should be used with caution. For all other catchments the model global minimum was consistent with the physical catchment characteristics.

6.3 Case Studies

Two catchments are considered in more detail. The calibration was simple for catchments such as Bass River at Lock, where the statistical assumptions were readily satisfied, and parameter estimates consistent with the catchment characteristics were obtained. Allyn River at Halton was problematic and is used to highlight some of the difficulties satisfying the statistical assumptions in section 6.1.5 and other problems that may be encountered.

6.3.1 Bass River at Loch

Bass River at Loch is located in South Gippsland Victoria and is typical of the catchments in the temperate regions of Australia. The catchment has an area of 52 km^2. The mean annual runoff in the catchment is 17300 ML (333mm) and the mean annual

rainfall is 1100 mm and has monthly means uniformly distributed over the year. The mean annual potential evapotranspiration is 886mm. The average daily maximum summer temperature is 25°C and the average daily maximum winter temperature is 13°C. The catchment is covered predominantly by grass and has duplex soils (Chiew and McMahon, 1993).

Seven parameters were fitted as shown in Table 15. The default value of 8.9 was used for E_{max} since the residual variance could not be reduced by fitting this parameter. The ground water replenishment modification reduced the residual variance although the standard error of 11.2 for parameter C indicates that it is not well determined.

Figure 20 shows plots of the monthly observed and modelled runoff for the period used to calibrate the model. There is good agreement between the modelled and observed runoff.

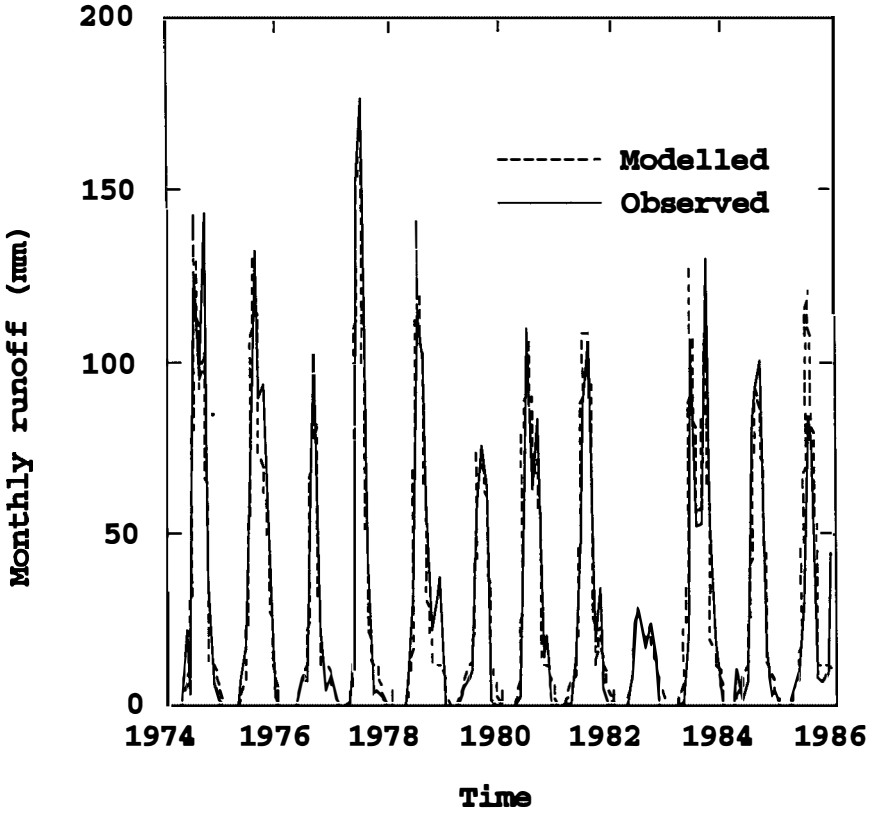


Figure 20. Plot of modelled ($\hat{Q}$) and observed (Q) monthly flows

for Bass River at Loch.

Figure 21 shows a plot of modelled runoff against observed runoff. The plot closely follows the one on one line indicating a reasonable fit to both the peaks and periods of low flow when fitted with $\lambda_1 = 0.5$. The good appearance of the fit is supported by an R^2 value of 0.927 and coefficient of efficiency (E) of 0.925.

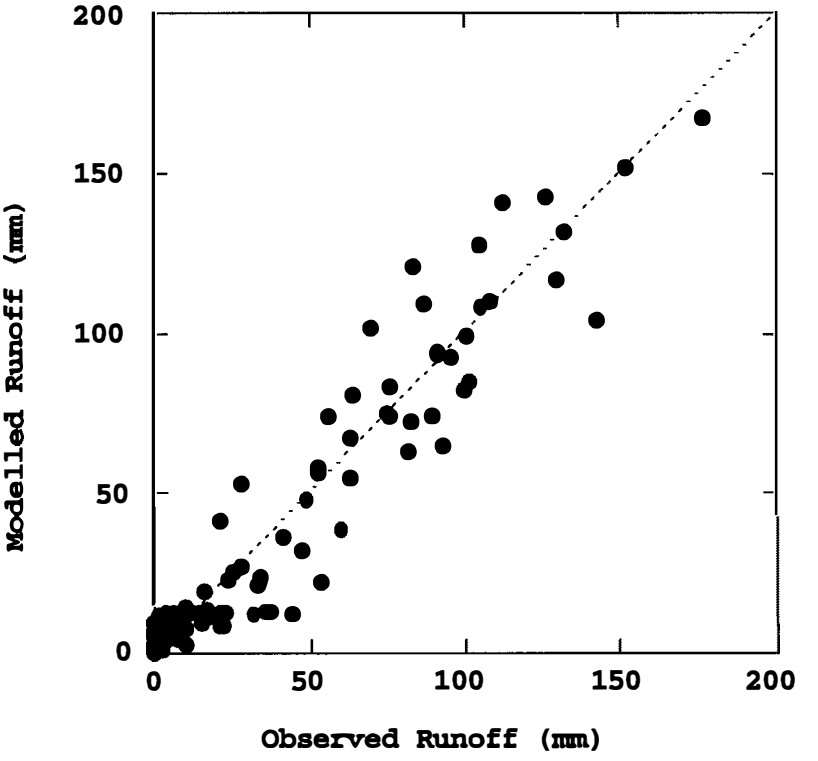


Figure 21. Plot of modelled ($\hat{Q}$) versus observed (Q) runoff values for Bass River at Loch.

Figure 22 shows a plot of the absolute value of the residuals against $\hat{q}$ (note since $\lambda_1 \neq 1$ in Equations (28) and (29) the transformed modelled runoff ($\hat{q}$) was fitted to the transformed observed runoff (q)) with a line fitted using robust locally weighted regression (Cleveland, 1979). The fitted line shows initially, a slight increase, followed by a gradual decrease in the residual variance. However, there is not sufficient evidence to indicate departure from a constant residual variance and it was assumed that the residuals are homoskedastic.

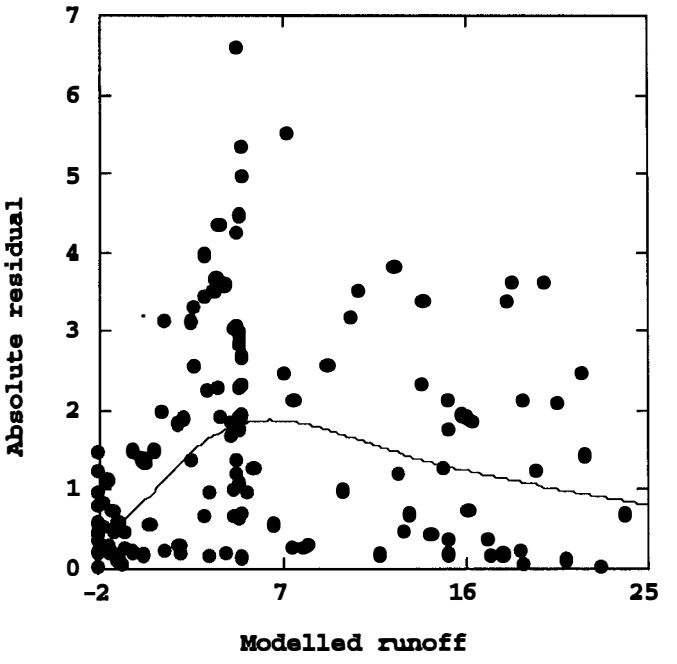


Figure 22. Plot of residuals versus transformed modelled runoff ($\hat{q}$) with line fitted using robust locally weighted regression for Bass River at Loch.

A quantile plot of the residuals (Figure 23) for this catchment showed that the points lay very nearly along a straight line. I could not reject the assumption that the residuals are normally distributed.

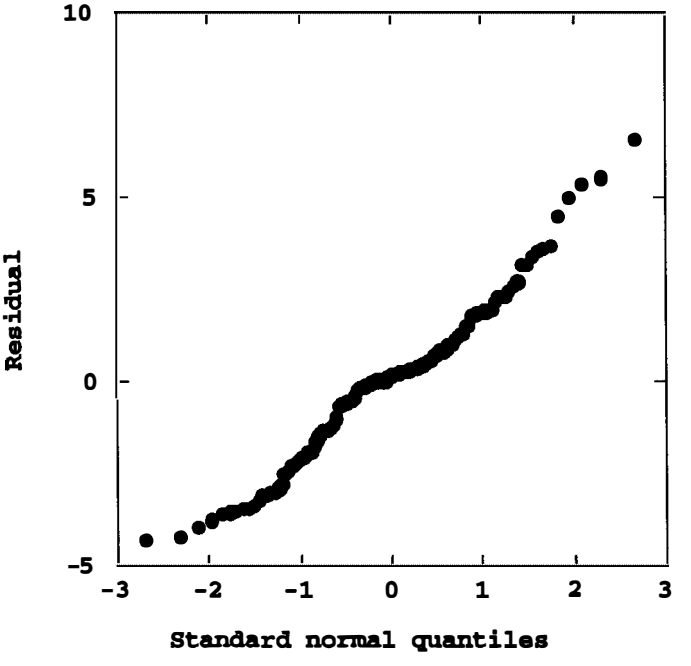


Figure 23. Quantile plot of residuals for Bass River at Loch.

Autocorrelation and partial autocorrelation plots (Figures 24 and 25) indicated that the assumption of independence could not be rejected.

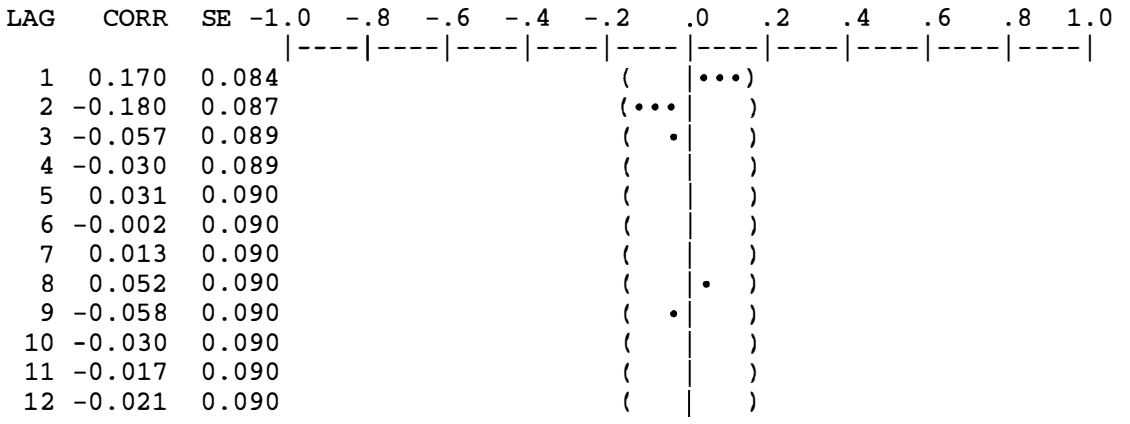


Figure 24. Autocorrelation plot of residuals for Bass River at Loch

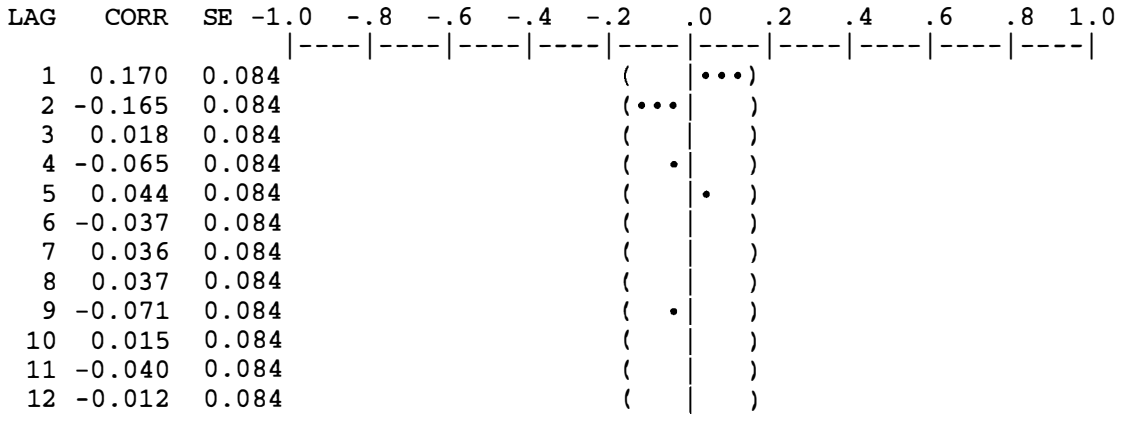


Figure 25. Partial autocorrelation plot of residuals for Bass River at Loch

6.3.2 Allyn River at Halton

Allyn River in New South Wales is a western bank tributary of the Lower Hunter River and enters above Newcastle after joining the Patterson River. The catchment above Halton has an area of 205km^2, a mean annual runoff of 71000ML (350mm) from a

mean annual rainfall of 1200 to 1400mm. Priestley-Taylor potential evaporation is about 1200mm. The upper 40 percent of the catchment is in State Forest and National Park and is heavily timbered with nothofagus rainforest and wet sclerophyll eucalypt forest. The balance is open forest, cleared grazing and some arable land close to the river. The soils are loams of varying depth. Elevation ranges from 250m to 1500m and there is a strong rainfall gradient. The average daily maximum temperature is 29°C in summer and 16°C in winter with occasional snow falls on the peaks of the Barrington Tops. As indicated in Table 14 catchment rainfall was scaled by a factor of 1.2 times the data of Chiew and McMahon (1993) to allow for the position of the raingauge and rainfall gradients. This provided more reasonable values of actual evapotranspiration.

The final parameter estimates required significant intervention to arrive at a set of meaningful values. The default value of 8.9 was used for E_{max} since the residual variance could not be reduced by fitting this parameter. SDR_{max} was set to zero, i.e. the lower bound for this parameter which minimised $S(\hat{\theta})$. NDC was set to 0.2 since I did not believe that the estimates of less than 0.2 found by global optimisation were consistent with the physical basis of the model. The transformation parameter λ_1 was then set to the successive values in Table 16, and estimates for S, F, B, DPF and C were obtained. Four SA runs were used for each value of λ_1. The run with the lowest value for $S(\hat{\theta})$ is shown.

Table 16. Estimate of global minimum for Allyn River at Halton

λ_1	S	F	B	DPF	C	$S(\hat{\theta})$	L_{max}
0.1	98	6.5	0.22	0.051	2.14	71.4	-175.6
0.2	90	6.7	0.22	0.045	2.12	92.5	-166.8
0.3	83	9.6	0.22	0.059	1.45	137.7	-164.4
0.4	91	8.9	0.25	0.068	1.18	227.1	-166.8
0.5	94	9.1	0.26	0.078	1.18	417.7	-174.2
0.5	180	13.9	0.32	0.159	1.67	423.9	-174.9
0.6	177	13.7	0.33	0.166	0.82	747.0	-181.4
0.7	179	13.8	0.35	0.170	1.36	1403.7	-188.8
0.8	178	13.8	0.36	0.182	1.04	2764.3	-199.5
0.9	179	13.8	0.37	0.179	1.08	5697.6	-212.2
1.0	181	13.8	0.37	0.173	1.20	12160.9	-226.6

Table 16 contains at least two local minima with different parameter values (note that the estimates of S, F, B and DPF consistently change at $\lambda_1 = 0.5$). When $\lambda_1 \leq 0.5$ the first local minimum is the best estimate of the global minimum and when $\lambda_1 > 0.5$ the second minimum is the best estimate of the global minimum. At $\lambda_1 = 0.5$ the two minima have similar values for $S(\hat{\theta})$ as shown in Table 16 and the optimisation program could not differentiate between them giving each minimum as the estimate of the global minimum on two occasions. For $\lambda_1 \neq 0.5$ the correct minimum was found on every occasion.

The minimum at $S \approx 90$ and $F \approx 9$ was consistent with the physical catchment characteristics as described by Boughton (1984) while the values of S and F for the second minimum were not. Setting $\lambda_1 = 0.3$ to maximise the maximum likelihood function in Equation (35) placed more emphasis on the low flow values with a smaller variance at the expense of the larger flows as shown in Figure 26.

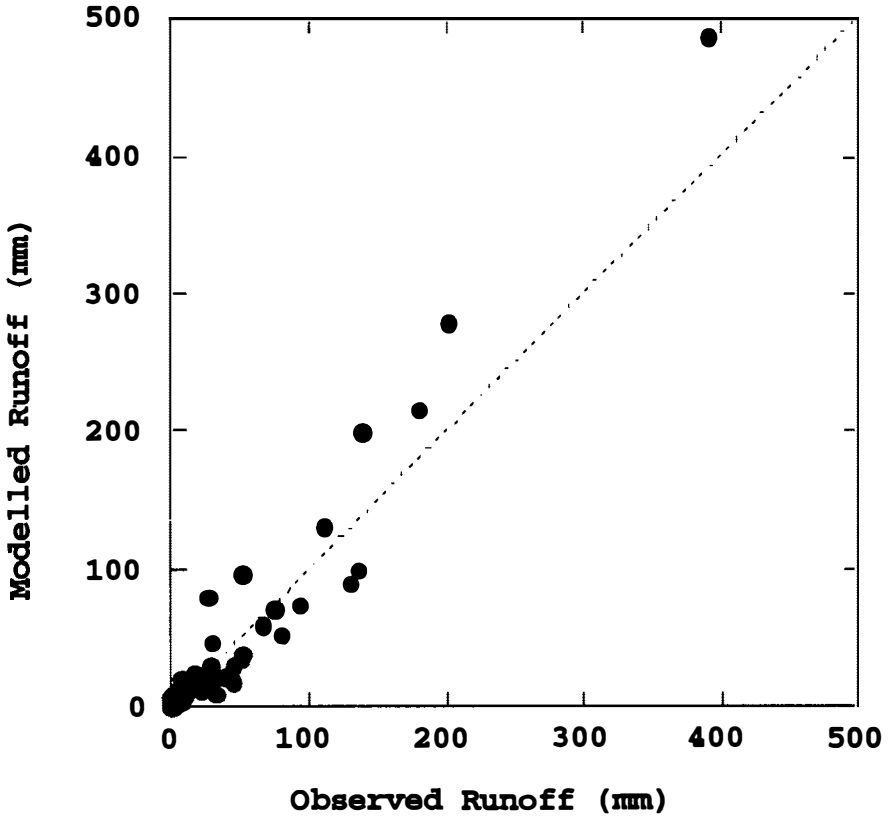


Figure 26. Plot of modelled ($\hat{Q}$) versus observed (Q) runoff values with $\lambda_1 = 0.3$ and $L_{max} = -164.4$ for Allyn River at Halton.

Increasing λ_1 to 0.4 or 0.5 gave a better fit, however λ_1 could not be increased beyond 0.5 without the program locating the other minimum. Knowledge of the catchment suggested setting the upper limit of F to 8.9 to force convergence to the first minimum, so higher values for λ_1 could be investigated. $\lambda_1 = 1.0$ was found to give the best fit and the parameter estimates are shown in Table 15. Starting at the first minimum with $\lambda_1 = 1.0$ fitting S, F, B, DPF and C, the Nelder and Mead simplex method moved towards the second minimum converging to other local minima with unacceptable parameter values. This behaviour indicated the existence of a valley containing the two extrema and other local minima. Nathan and McMahon (1990) were not able to find the global minimum fitting 5 parameters of the SFB model to this catchment using the method of Nelder and Mead. The better fit is demonstrated in Figures 27 and 28 and is supported by an R^2 value of 0.953 and a coefficient of efficiency (E) of 0.951.

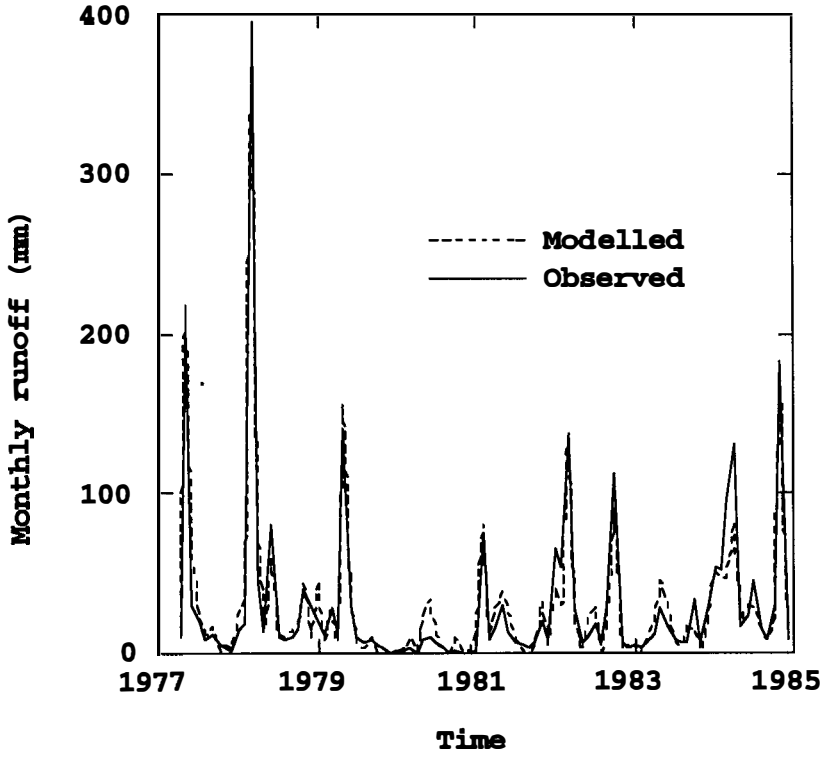


Figure 27. Plot of modelled ($\hat{Q}$) and observed (Q) monthly flows for Allyn River at Halton.

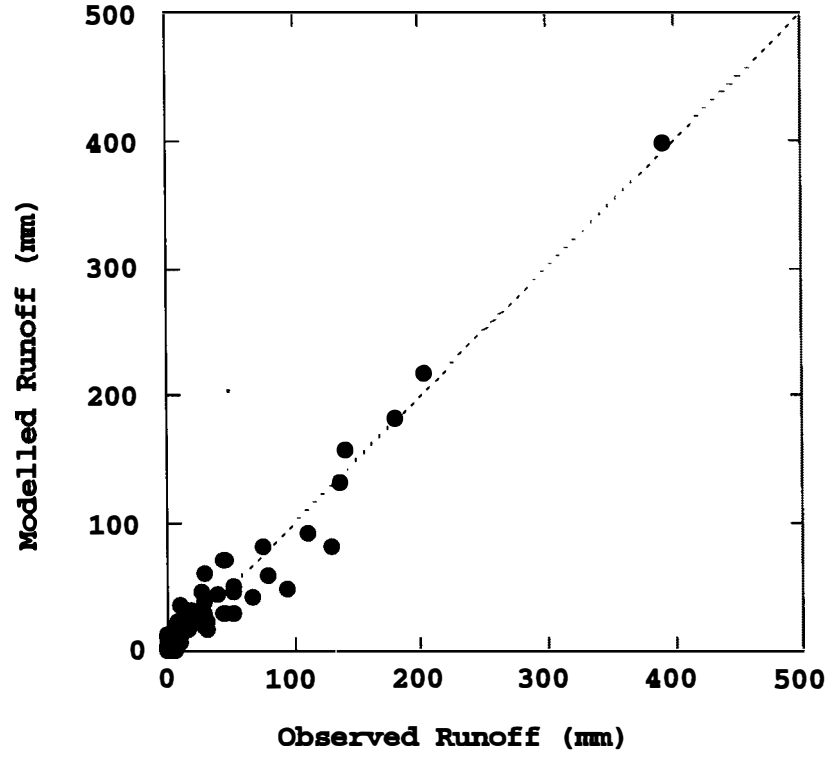


Figure 28. Plot of modelled ($\hat{Q}$) versus observed (Q) runoff values with λ_1 = 1.0 and L_{max} = -230.2 for Allyn River at Halton.

The model offers a closed water balance and a significant amount of deep drainage contributing towards sustaining evaporation from the lower part of the soil store. This behavior is consistent with that expected of the upper wetter forested part of the catchment. Figure 29 shows a plot of the absolute value of the residuals against $\hat{Q}$ (note since $\lambda_1 = 1$ in Equations (28) and (29) the modelled runoff ($\hat{Q}$) was fitted to the observed runoff (Q)) with a line fitted using robust locally weighted regression (Cleveland, 1979). The fitted line shows initially, increasing, and then decreasing residual variance. This situation cannot be corrected using the transformations in Equations (28) and (29).

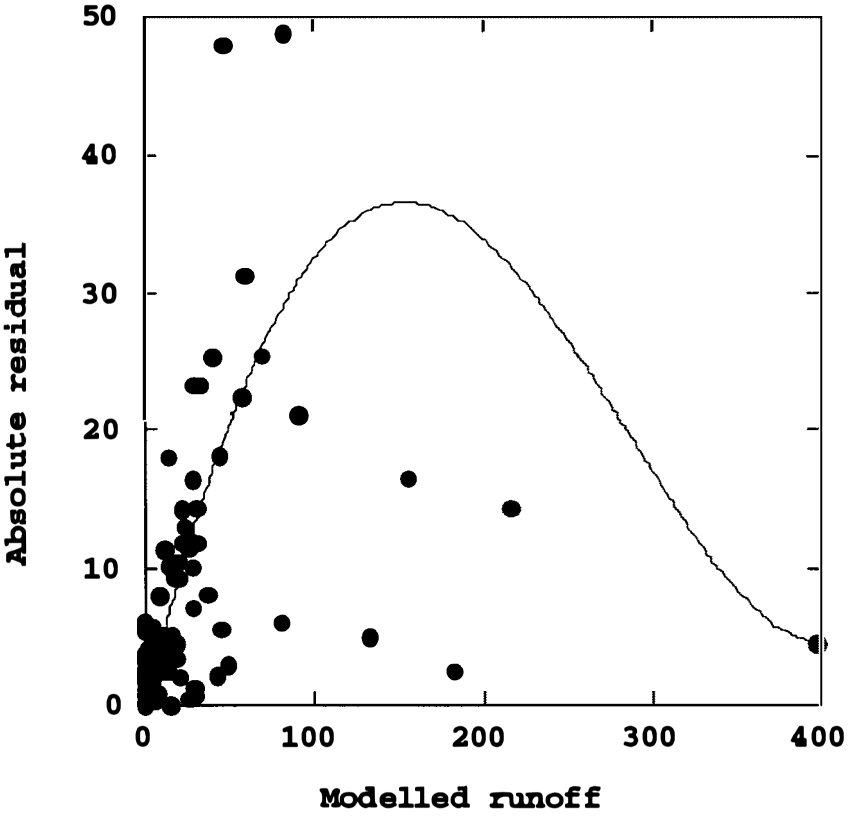


Figure 29. Plot of residuals versus modelled runoff ($\hat{Q}$) with line fitted using robust locally weighted regression for Allyn River at Halton.

A quantile plot of the residuals (Figure 30) for this catchment showed a noticeable deviation from normality in the upper tail that could not be corrected without sacrificing the fit.

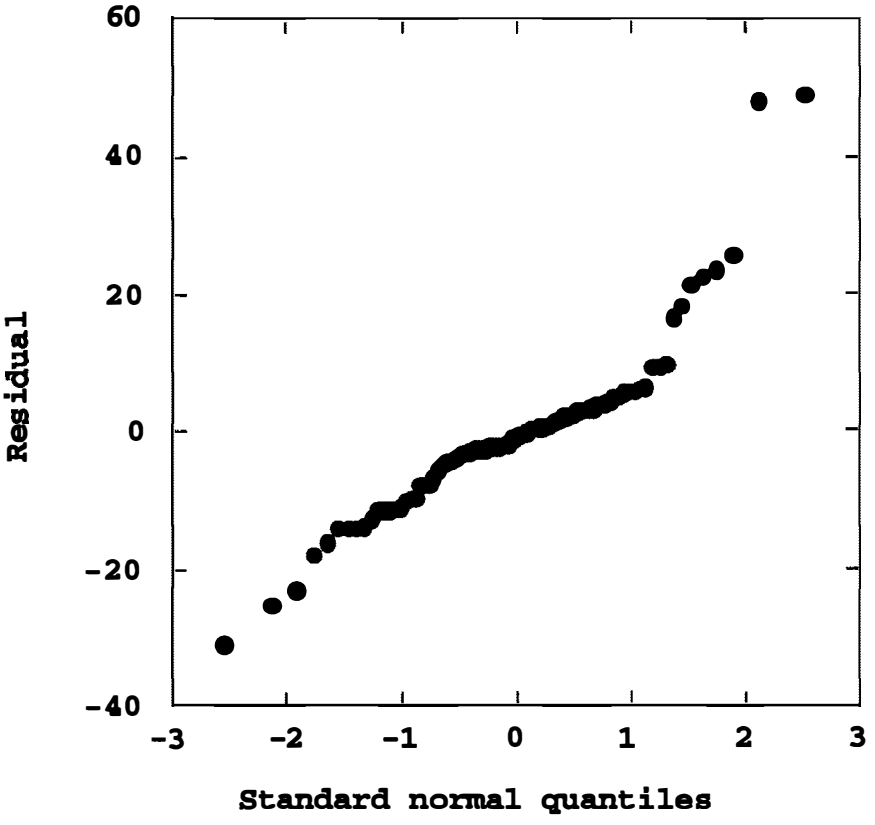


Figure 30. Quantile plot of residuals for Allyn River at Halton.

Autocorrelation and partial autocorrelation plots (Figures 31 and 32) show that the assumption of independence was not satisfied because of the presence of an AR(1) process. An autoregressive error model (Equation 30) could not be used since the streamflow data for this catchment was incomplete.

```
LAG    CORR   SE -1.0  -.8  -.6  -.4  -.2   .0   .2   .4   .6   .8  1.0
                 |----|----|----|----|----|----|----|----|----|----|
  1   0.402  0.104                    (    |••••)•••••
  2   0.043  0.119                    (    |•   )
  3   0.149  0.119                    (    |••• )
  4   0.087  0.121                   (     |••   )
  5   0.053  0.122                   (     |•    )
  6   0.053  0.122                   (     |•    )
  7   0.019  0.123                   (     |     )
  8   0.081  0.123                   (     |••   )
  9   0.028  0.123                   (     |     )
 10  -0.051  0.123                   (    •|     )
 11  -0.114  0.123                   (   ••|     )
 12  -0.031  0.125                   (     |     )
```

Figure 31. Autocorrelation plot of residuals for Allyn River at Halton

```
LAG    CORR   SE -1.0  -.8  -.6  -.4  -.2   .0   .2   .4   .6   .8  1.0
                 |----|----|----|----|----|----|----|----|----|----|
  1   0.402  0.104                    (    |••••)•••••
  2  -0.141  0.104                    ( •••|     )
  3   0.226  0.104                    (    |••••)
  4  -0.086  0.104                    (  ••|     )
  5   0.092  0.104                    (    |••  )
  6  -0.031  0.104                    (    |    )
  7   0.013  0.104                    (    |    )
  8   0.082  0.104                    (    |••  )
  9  -0.071  0.104                    (   •|     )
 10  -0.019  0.104                    (    |     )
 11  -0.142  0.104                    ( •••|     )
 12   0.098  0.104                    (    |••  )
```

Figure 32. Partial autocorrelation plot of residuals for Allyn River at Halton

6.4. Summary

SA provides an efficient solution to the CCRM calibration problem. However, the complexity of obtaining an accurate numerical solution grows exponentially with the number of parameters to be determined. As a result high dimensional problems require long slow cooling schedules with considerable computational effort. The inclusion of insensitive or poorly determined parameters in the fitting process often caused premature convergence before the global minimum was found. These parameters were identified by their inability to reduce the residual variance when included, large

standard errors, and they may have a high correlation with one or more other parameters.

The modified 8-parameter Boughton's model used for this study gave a much better fit than the original 3-parameter model. The values for DPF, SDR_{max}, and NDC recommended by Boughton(1984) and used by Nathan and McMahon(1990) and Chiew et al. (1993) were not appropriate for all catchments. Much of the improvement in model fit was due to finding an optimal value for the DPF parameter, which relates to the baseflow and deep percolation. The results support the suggestion by Irish (1991) that a higher value for this parameter should be used. The optimal values for this parameter shown in Table 15 are very different from the value of 0.005 recommended by Boughton. The optimal value for SDR_{max} was zero in 17 catchments which suggests that the recommended value of 25mm is not always appropriate. Fitting NDC reduced the residual variance for 13 catchments and the estimate was more than 2 standard deviations from Boughton's recommended value of 0.5 for 5 catchments. The value recommended by Boughton for E_{max} of 8.9mm d^{-1} is reasonable since fitting E_{max} reduced the residual variance in only 3 catchments all of which were within a standard deviation of the recommended value.

There are a number of instances of large values of the standard error relative to the value of the parameter. This seems to indicate that the modelled process is irrelevant or could be better expressed. Naradhan Creek catchment is a case in point. The hydrograph shows little interflow and no baseflow and the estimation process correctly sets B to zero so all drainage proceeds to the groundwater store for recirculation to the non-draining evaporation store. DPF is set an order of magnitude higher than most other catchments but is very poorly determined. This is not surprising since there is no baseflow for this catchment. It could be set to one or the baseflow process bypassed. Parameter C is likewise poorly determined since it does not directly influence any

streamflow mechanism but does succeed in reducing the residual variance and thus providing a better fit.

The standard errors for parameter C for all catchments are poorly estimated due to nonlinearity in this parameter; the model is considerably more sensitive to a reduction in the value of this parameter than an increase of the same size. The standard errors for C are larger than the parameter estimates for eight catchments although inclusion of this parameter reduces the residual variance thus providing a better fit.

7. Conclusions

I have presented a new method for global minimisation of functions of continuous variables based on simulated annealing (SA). The new method is able to find the global minimum of test functions with an extremely high number of local minima. The reliability and efficiency of the new SA algorithm presented compares favourably with three SA algorithms investigated and other global optimisation methods from the literature using three test functions that offer different challenges. The new method is also extremely robust giving good results for a wide range of initial temperatures and cooling schedules. The new method uses exploration steps from the direct search method of Hooke and Jeeves(1961) to effectively maintained a downhill bias. The algorithm was able to investigate both the current valley using exploration steps and more distant points using pattern moves.

The starting temperature and cooling schedule for SA must be carefully selected. Temperatures that are too high waste computational effort while a temperature that is too low does not adequately sample the function and may lead to local entrapment. The most appropriate initial temperature and cooling schedule is specific to both the SA algorithm used and problem to be solved. The choice of SA algorithm should depend on the nature of the test function being investigated. SA algorithms such as Press and Teukolsky (1991) and the new method that use a local search phase to find the bottom of the valley containing the global minimum are more efficient. It may also be more efficient to use several smaller runs of a SA program than one large one. Since there is no guarantee that the global minimum will be found multiple runs of a SA program should be used.

Because of the global nature of hydrologic model calibration problems, SA is a useful tool for parameter estimation. Using the maximum number of parameters that could be calibrated for all catchments, SA was able to find the same optimal point in 77 percent

of calibration attempts. The technique has been shown to calibrate Boughton's model with up to 7 parameters using data of good length and quality. However, considerable care must be taken to obtain a parsimonious set of parameter estimates for a catchment and to ensure that the least squares assumptions are satisfied. The number of parameters able to be calibrated using SA is far superior to previous calibration attempts of Boughton's model by myself and others using local optimisation methods. Because the method does not require the calculation of derivatives it works on models containing discontinuities on the response surface. The SA algorithm by Press and Teukolsky (1991) with the addition of the restart feature performed well when applied to the model calibration problem although more efficient SA algorithms may be available.

There is an inherent danger in using automatic optimisation methods because the parameter estimates obtained may not be consistent with the physical characteristics of the catchment. Global optimisation methods do not solve this problem because:

(i) the parameter estimates at the global minimum of the residual sum of squares response surface may provide a good fit although the hydrological processes are not being modelled correctly,

(ii) the global minimum may not have been found,

(iii) an inappropriate objective function may have been used, or

(iv) proper transformation of output to give normalised residuals has not been achieved.

Boughton's model appears best suited to the temperate regions of Australia. Its performance in arid areas is poor, although much of the problem is due to low and sporadic rainfall and even more sporadic streamflow events on which to substantiate model calibration. There is evidence that the values for the model parameters DPF, SDR_{max} and NDC recommended by Boughton(1984) and used by Nathan and

McMahon (1990) and Chiew et al. (1993) may not be suitable for many Australian catchments. The results obtained in this study (including 4 catchments in the same basins used by Nathan and McMahon) indicate that a better fit is obtained by fitting additional parameters, although the parameters that contribute to an improvement in model fit are specific to the catchment concerned.

References

Australian Bureau of Meteorology. (1988). Climatic averages Australia. Canberra: Australian Government Publishing Service.

Australian Bureau of Meteorology. (1991). Benchmark stations for monitoring the impact of climatic variability and change. Water Resources Assessment Section, Hydrology Branch, Melbourne, Vic., Australia, Vols. 1-7.

Bates, B.C. (1994). Calibration of the SFB model using a simulated annealing approach. Proc. Water Down Under 94 Conf., Inst. Eng., Aust., Natl. Conf. Publ. No 94/15, Vol. 3, 1-6.

Bates, B.C. Charles, S.P., Sumner , N.R. and Fleming, P.M. (1994). Climate change and its hydrological implications for South Australia. *Trans. R. Soc. S. Aust.*, 118(1), 35-43.

Bates, D.M. and Watts, D.G. (1980). Relative curvature measures of nonlinearity. *J. R. Stat. Soc. Ser. B,* 42, 1-25.

Benke, K.K. and Skinner, D.R. (1991). A direct search algorithm for global optimisation of multivariate functions. *Aust. Comp. J.* 23(2), 73-85.

Bohachevsky, I.O., Johnson, M.E. and Stein, M.L. (1986) Generalised simulated annealing for function optimisation. *Technometrics*, 28, 209-217.

Boughton, W.C. (1984). A simple model for estimating the water yield of ungauged catchments. *Civ. Eng. Trans., Inst. Eng Aust.*, CE26(2), 83-88.

Box, G.E.P. and Cox, D.R. (1964). An analysis of transformations. *J. R. Stat. Soc. Ser. B*, 26(2), 211-252.

Box, M.J. (1965). A new method of constrained optimisation and a comparison with other methods. *Computer J.*, 8, 42-52.

Brooks, D.G. and Verdini, W.A. (1988). Computational experience with generalised simulated annealing over continuous variables. *Am. J. Math. & Mgmt Sc.* 8(3&4), 425-449.

Bunday, B.D. and Garside, G.R. (1987). Optimisation methods in Pascal. London: Edward Arnold.

Butler, A.R. and Slaminka, E.E. (1992). An evaluation of the sniffer global optimization algorithm using standard test functions. *J. Comp. Physics,* 99, 28-32.

Chiew, F.H.S., Stewardson, T.A. and McMahon, T.A. (1993). Comparison of six rainfall-runoff modelling approaches. *J. Hydrol.*, 147, 1-36.

Chiew, F.H.S. and McMahon, T.A. (1993). Complete set of daily rainfall, potential evapotranspiration and streamflow data for twenty eight unregulated Australian catchments. Centre for Environmental Applied Hydrology, Univ. of Melbourne, Vic., (unpublished report).

Chiew, F.H.S. and McMahon, T.A. (1994). Application of the daily rainfall-runoff model MODHYDROLOG to 28 Australian catchments. *J. of Hydrol.*, 153, 383-416.

Cleveland, W.S. (1979). Robust locally weighted regression and smoothing scatterplots. *J. Am. Stat Assoc.*, 74(368), 829-836.

Corana, A., Marchesi, M., Martini, C. and Ridella, S. (1987). Minimizing multimodal functions of continuous variables with the "simulated annealing" Algorithm. *ACM Trans. Math Software*, 13(3), 262-280.

Deckers, A. and Aarts, E. (1991). Global optimisation and simulated annealing. *Mathematical Programming,* 50, 367-393.

Dixon L.C.W. and Szego G.P. (1978). The global optimisation problem. In: C.W.L. Dixon and G.P. Szego (ed.) Towards Global Optimisation 2. North-Holland, Amsterdam.

Donaldson, J.R. and Schnabel, R.B. (1987). Computational experience with confidence regions and confidence intervals for nonlinear least squares. *Technometrics,* 29(1), 67-82.

Duan, Q., Sorooshian, S. and Gupta, V.K. (1992). Effective and efficient global optimisation for conceptual rainfall-runoff models. *Water Resour. Res.*, 28(4), 1015-1031.

Eglese, R.W. (1990). Simulated annealing: A tool for operational research. *Eur. J. Oper. Res.*, 46, 271-281.

Gunel, T. and Yazgan, B. (1992). A new global optimization method based on simulated annealing algorithm. *Proc.Eng. Sys. Des. & Anal.*, 47(4), 199-202.

Hooke, R. and Jeeves, T.A. (1961). Direct search solution of numerical and statistical problems. *J. Assoc. Comput. Mach.*, 8(2), 212-229.

Irish, J.L. (1991). Discussion on the paper: The SFB model part 1 - Validation of fixed model parameters. *Civ. Eng. Trans., Inst. Eng Aust.*, CE33(3), 207.

Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schevon, C. (1987). Optimisation by simulated annealing: an experimental evaluation. Parts I and II, AT&T Bell Laboratories report, AT&T Bell Laboratories, Murray Hill, NJ.

Johnston, P.R. and Pilgrim, D.H. (1976). Parameter optimisation of watershed models. *Water Resour. Res.*, 12(3), 477-486.

Kalivas, J.H. (1992). Optimisation using variations of simulated annealing. *Chemom. Intell. Lab. Syst.*, 15, 1-12.

Kirkpatrick, S., Gelatt, Jr., C.D., and Vecchi, M.P. (1982). Optimization by simulated annealing. IBM Research Report RC, No. 9355, IBM, New York.

Kirkpatrick, S., Gelatt, Jr., C.D., and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220, 671-680.

Kuczera, G. (1988). On the revision of the soil dryness index streamflow yield model. *Civ. Eng. Trans., Inst. Eng Aust.*, CE30(2), 79-86.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21, 1087-1090.

Mockus, J. (1989). Bayesian approach to global optimization. Dordrecht: Kluwer.

Morton, F.I. (1983). Operational estimates of actual evapotranspiration and their significance to the science and practice of hydrology. *J. Hydrol.*, 66, 1-76.

Nathan, R.J. and McMahon, T.A. (1990). The SFB Model Part 1 - Validation of fixed model parameters. *Civ. Eng. Trans., Inst. Eng Aust.*, CE32(3), 157-161.

Nelder, J.A. and Mead, R. (1965). A simplex method for function minimisation. *The Comp. J.*, 7, 308-313.

Polovinkin, A.I. (ed) (1981). Automation of searching design. *Radio i Sviaz, Moscow.*

Press, W.H. and Teukolsky, S.A. (1991). Simulated annealing optimization over continuous spaces. *Comp. in Physics,* 5(4) 426-429.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992). Numerical Recipes in FORTRAN: The Art of Scientific Computing. 2nd ed. New York: Cambridge Press.

Priestley, C.H.B. and Taylor, R.J. (1972). On the assessment of surface heat flux and evaporation using large scale parameters. *Mon. Weather Rev.*, 100, 81-92.

Ratschek, H. and Voller, R.L. (1990). Global optimisation over unbounded domains. *SIAM J. Control & Optimisation*, 28(3), 528-539.

Rogers, J.W. and Donelly, J. (1989). A search technique for global optimisation in a global environment. *J.Optim. Theory Appl.*, 61, 111-121.

Sorooshian, S. and Dracup, J. A. (1980). Stochastic parameter estimation procedures for hydrologic rainfall-runoff models, correlated and heteroscedastic error cases. *Water Resour. Res.*, 16(2), 430-442.

Sorooshian, S., Gupta, V.K. and Fulton, J.L. (1983). Evaluation of maximum likelihood parameter estimation techniques for conceptual rainfall-runoff models: Influence of calibration, data variability and length on model credibility. *Water Resour. Res.*, 19(1), 251-259.

Sutter, J.M., and Kalivas, J.H. (1991). Convergence of generalised simulated annealing with variable step size with applications towards parameter estimations of linear and nonlinear models. *Analytical Chemistry*, 63, 2383-2386.

Traub, J.F. and Wozniakowsky, H. (1994). Breaking intractability. *Scientific Am.*, 270(3), 102-107.

Vanderbilt, D. and Louie, S.G. (1984). A Monte Carlo simulated annealing approach to optimisation over continuous variables. *J. Comput. Phys.*, 56, 259-271.

Appendix A
Test Functions

Test Function for Global Optimisation Methods
Hartman's Family
Dixon & Szego (1978); Vanderbilt & Louie (1984); Brooks & Verdini (1988); Butler & Slaminka (1992)

$m := 4 \qquad p := 3$

$i := 1..m \qquad j := 1..p$

$$a := \begin{bmatrix} 3 & 10 & 30 \\ .1 & 10 & 35 \\ 3 & 10 & 30 \\ .1 & 10 & 35 \end{bmatrix}$$

$$b := \begin{bmatrix} .36890 & .1170 & .2673 \\ .46990 & .4387 & .7470 \\ .10910 & .8732 & .5547 \\ .03815 & .5743 & .8828 \end{bmatrix}$$

$c := (1 \quad 1.2 \quad 3 \quad 3.2)^T$

$$f(x) := -\sum_i c_i \cdot \exp\left[-\sum_j a_{i,j} \cdot (x_j - b_{i,j})^2\right]$$

Function domain

0 <= xi <= 1, i=1,2,3

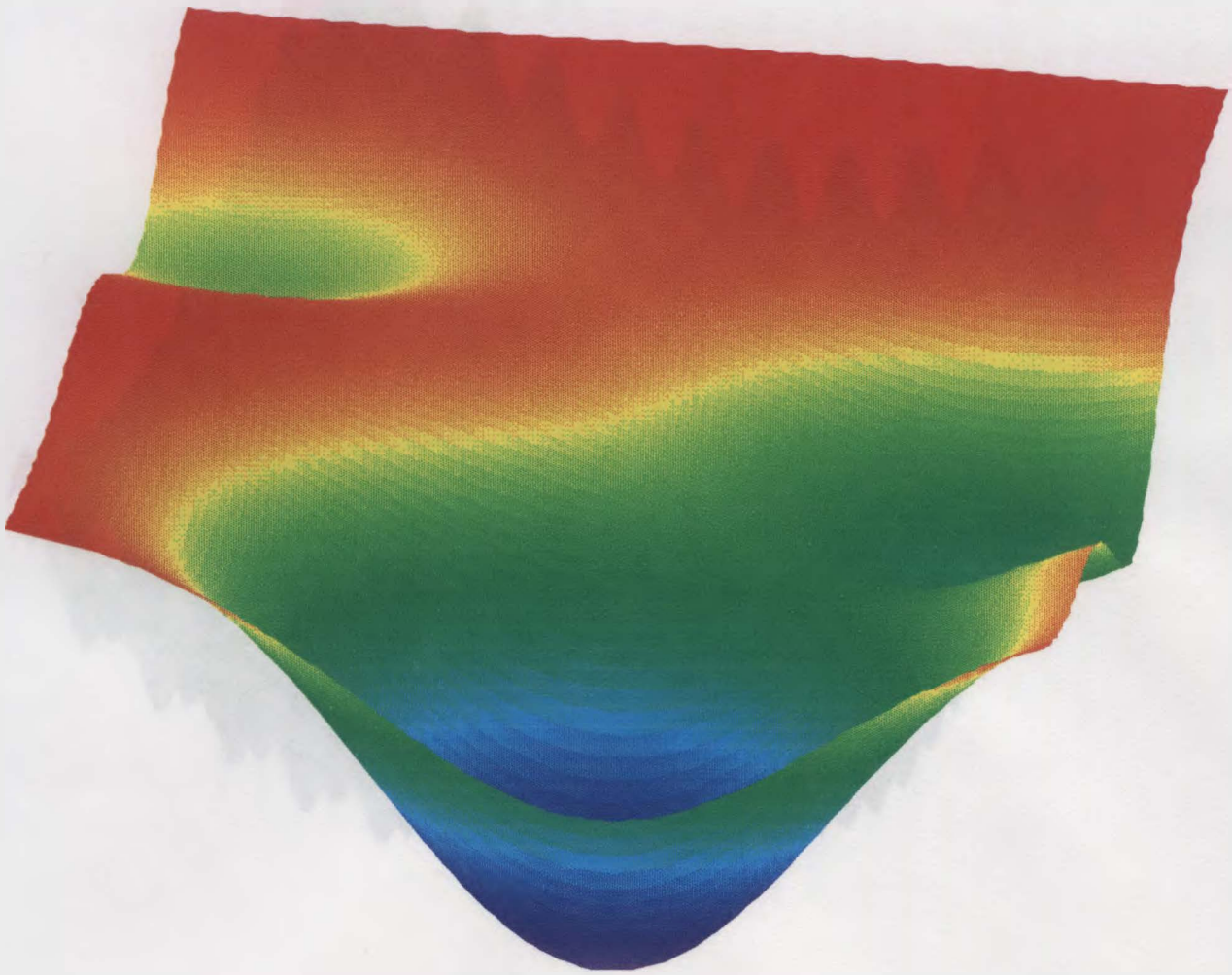
$k := 1..101 \qquad x1_k := \frac{k-1}{100}$

$l := 1..101 \qquad x2_l := \frac{l-1}{100}$

$M_{k,l} := f\left(\left(x1_k \quad x2_l \quad x2_k\right)^T\right)$

Global minimum

$f\left((0.11462 \quad 0.55565 \quad 0.85255)^T\right) = -3.86278$



Test Function for Global Optimisation Methods

Rastrigin Function

Polovinkin(1981); Benke and Skinner(1991)

$$f(x_1, x_2) := x_1^2 + x_2^2 - \cos(18 \cdot x_1) - \cos(18 \cdot x_2)$$

Function domain

-2 <= x1 <= 2

-2 <= x2 <= 2

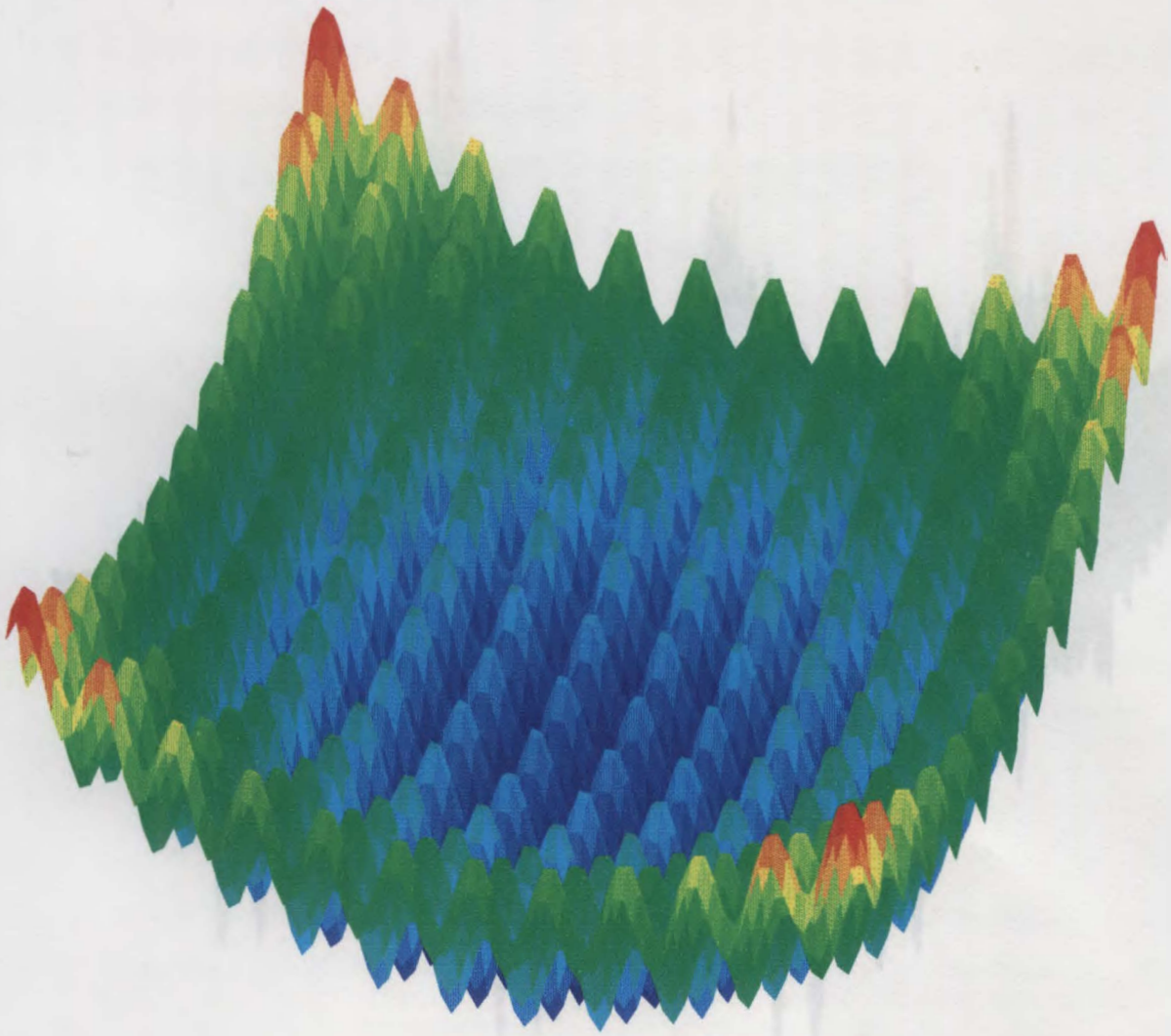
$i := 1..81$ $\qquad x_{1_i} := \frac{i - 41}{20}$

$j := 1..81$ $\qquad x_{2_j} := \frac{j - 41}{20}$

$M_{i,j} := f\left(x_{1_i}, x_{2_j}\right)$

Global minimum

$f(0, 0) = -2$



Test Function for Global Optimisation Methods

Penalised Shubert

Gunel and Yazgan (1992)

$i := 1..5 \qquad b := 0.5$

$$f(x_1, x_2) := \left[\sum_i i \cdot \cos\left[(i+1)\cdot x_1 + i\right]\right] \cdot \left[\sum_i i \cdot \cos\left[(i+1)\cdot x_2 + i\right]\right] + b \cdot \left[(x_1 + 1.4251)^2 + (x_2 + 0.8003)^2\right]$$

Function domain

-10 <= x1 <= 10

-10 <= x2 <= 10

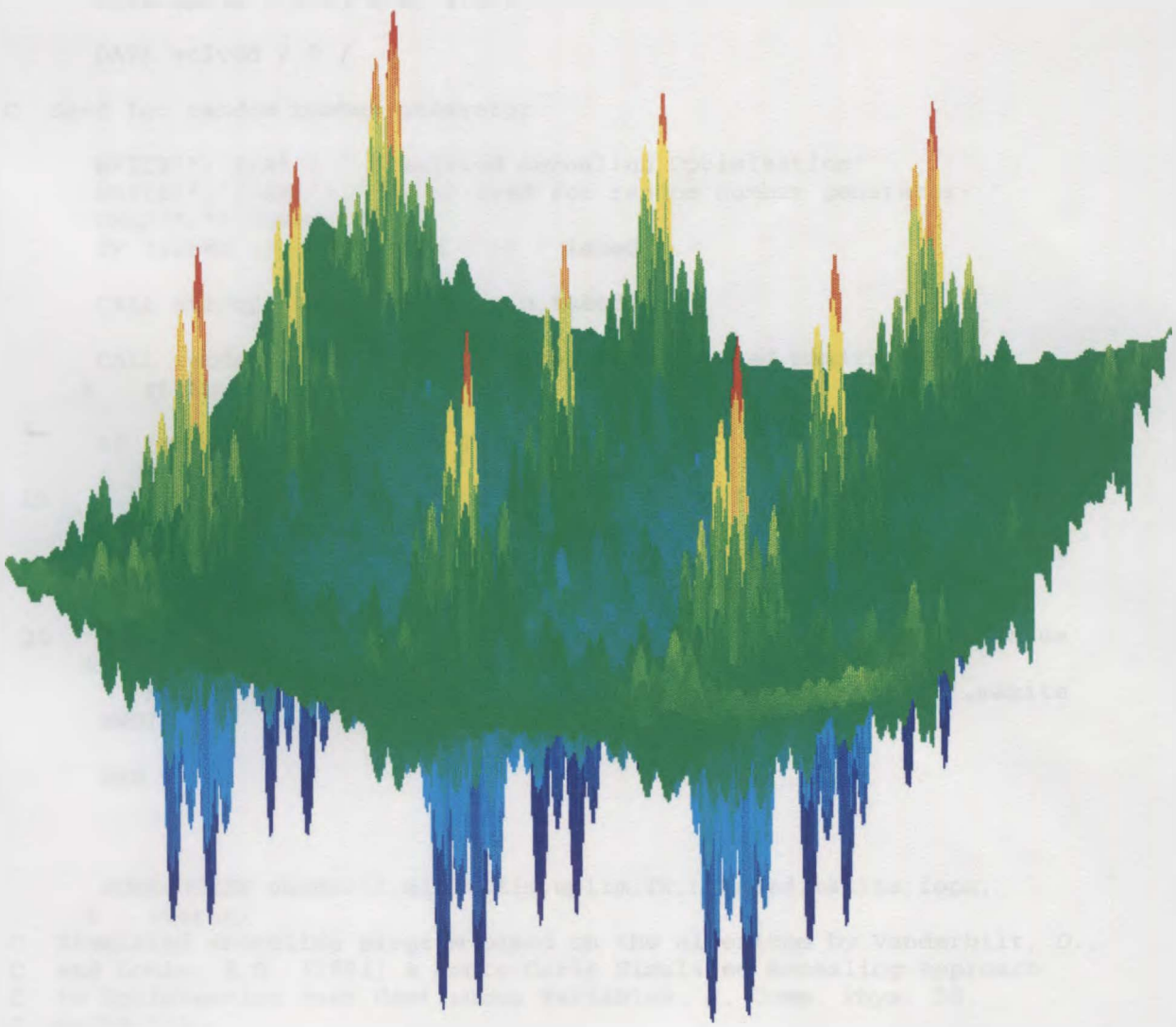
$i := 1..201 \qquad x1_i := \frac{i - 101}{10}$

$j := 1..201 \qquad x2_j := \frac{j - 101}{10}$

$M_{i,j} := f(x1_i, x2_j)$

Global minimum

$f(-1.4251, -0.8003) = -186.73091$



Appendix B
Program Listings

SIMULATED ANNEALING PROGRAMS

```
      PROGRAM sa
C  Simulated annealing program to find the global optimum of a
C  continuous function.

C  PROGRAMMER: Neil R. Sumner                              DATE: 28/2/95

      INTEGER n,iseed,maxits,solved,status
C     Set number of dimensions for function
      PARAMETER (n = 3)
      REAL T0,x(n),lowlim(n),uplim(n),fopt
      REAL hartman,ran1
      EXTERNAL hartman,ran1

C     Initial temperature, step size & maximum number of function
C        evaluations
      DATA T0, maxits / 1.0, 10000 /

C     Constraints (lower and upper limits for x1, x2, ...)
      DATA lowlim / 0.0, 0.0, 0.0 /
      DATA uplim / 1.0, 1.0, 1.0 /

      DATA solved / 0 /

C  Seed for random number generator

      WRITE(*,'(/A)') ' Simulated Annealing Optimisation'
      WRITE(*,'(/A$)') ' Enter seed for random number generator: '
      READ(*,*) iseed
      IF (iseed .ge. 0) iseed = -1 * iseed

      CALL startpt(lowlim,uplim,x,n,iseed)

      CALL vander(hartman,x,lowlim,uplim,T0,n,iseed,maxits,fopt,
     &   status)

      IF (status .eq. solved) THEN
         WRITE(*,10) fopt, x
 10      FORMAT(//' Optimisation Complete'/' Estimate of global optimum'
     &      ' =',F12.6,/' at x =',9F10.4,/)
         WRITE(*,'(A,I6)') ' Number of function evaluations = ',maxits
      ELSE
         WRITE(*,20) fopt, x
 20      FORMAT(//' Method has NOT converged'/' Current function value'
     &      ' =',F12.6,/' at x =',9F10.4,/)
         WRITE(*,'(A,I6)') ' Number of function evaluations = ',maxits
      ENDIF

      END



      SUBROUTINE vander(f,xi,lowlim,uplim,Tk,n,iseed,maxits,fopt,
     &   status)
C  Simulated annealing program based on the algorithm by Vanderbilt, D.,
C  and Louie, S.G. (1984) A Monte Carlo Simulated Annealing Approach
C  to Optimisation over Continuous Variables. J. Comp. Phys. 56,
C  pp259-271.
```

```
C
C  PROGRAMMER: Neil R. Sumner                                DATE: 28/2/95

C  Variables  f               : function to be minimised
C             xi              : initial guess, returns xopt
C             uplim,lowlim    : upper and lower constraints
C             Tk              : temperature
C             n               : number of variables in function f
C             iseed           : seed for random number generator
C             maxits          : maximum number of iterations
C             fopt            : best function evaluation found
C             status          : returns 0 if convergence occurs

C             xopt            : x-vector corresponding to fopt
C             r               : reduction factor for step length changes
C             count           : function evaluation counter
C             M               : number of steps in random walk
C             cov             : sample covariance matrix
C             deltax          : step vector
C             E               : average temperature for accepted moves
C             fi              : current function value

C  Parameters:
      INTEGER n,iseed,maxits,status
      REAL f,xi(n),lowlim(n),uplim(n),Tk,fopt

C  Internal variables:
      INTEGER nmax,M
      PARAMETER (nmax = 3, M = 15*nmax)
      INTEGER count,solved,limit,i,j,k,accept
      REAL ran1,r,epsilon,fi,fnew,E,p,pprime,Xs,Beta,deltax(nmax),
     &   xopt(nmax),x(nmax,M),u(nmax),xnew(nmax),Q(nmax,nmax),
     &   sum(nmax),A(M),S(nmax,nmax),const,cov(nmax,nmax),qroot
      EXTERNAL f,ran1

      DATA solved,limit / 0, 1 /
      DATA r, epsilon / 0.9, 0.00002 /
      DATA qroot, Beta, Xs / 1.732050808, 0.11, 3.0 /

C  Check array sizes

      IF (n .gt. nmax) STOP 'ERROR: Increase size of nmax in Vander'

C  Step 0.  Initialisation

      fi = f(xi,n)
      CALL vcopy(xi,xopt,n)
      fopt = fi
      count = 1
C     Initial covariance matrix
      CALL rdent(cov,nmax,n)

C  Iterate until convergence criteria satisfied

 10   CONTINUE

C  Compute step distribution matrix Q  Eqn (7)

         CALL choleski(cov,n,nmax,Q)

C  Initialise vector to store sum of step sizes

         DO 15 i=1, n
```

```
 15         sum(i) = 0.0

         E = 0.0
         accept = 0.0

C  Sample parameter space at temperature T

         DO 60 k=1, M

C  Step 1.  Compute step vector deltax = Q * u  Eqn (5)

C           Iterate through all n dimensions of parameter space
            DO 20 i=1, n
   20          u(i) = qroot * 2.0 * (ran1(iseed) - 0.5)
C           deltax = Q * u  Eqn (5)
            CALL matmul(Q,nmax,n,nmax,n, u,nmax,n,1,1, deltax,-1)
            CALL matadd(xi,nmax,n,1,1,deltax,xnew)

C  Step 2.  Check that coordinate lies inside function domain

            DO 40 i=1, n
 30             IF ((xnew(i).gt.uplim(i)).or.
     &             (xnew(i).lt.low1im(i))) THEN
                   xnew(i) = xnew(i) / 2
                   GOTO 30
               ENDIF
 40         CONTINUE

C  Save step size information

         DO 50 i=1, n
            x(i,k) = xnew(i)
 50         sum(i) = sum(i) + xnew(i)

C  Step 3.  Evaluate function

         fnew = f(xnew,n)
         IF (fnew .lt. fi) THEN
C            accept the new point
             CALL vcopy(xnew,xi,n)
             fi = fnew
             E = E + fi
             accept = accept + 1
             IF (fnew .lt. fopt) THEN
                CALL vcopy(xnew,xopt,n)
                fopt = fnew
             ENDIF
         ELSE
C           Metropolis move  Eqn (2)
            p = exp((fi - fnew) / Tk)
            pprime = ran1(iseed)
            IF (pprime .lt. p) THEN
C              accept Metropolis move
               CALL vcopy(xnew,xi,n)
               fi = fnew
               E = E + fi
               accept = accept + 1
            ENDIF
         ENDIF

  60   CONTINUE

C  Compute first (mean) & second (variance) moments Eqns (10) & (11)
```

```
      DO 65 i=1, n
 65      A(i) = sum(i) / M

      DO 80 i=1, n
         DO 80 j=1, n

         S(i,j) = 0.0
         IF (i .le. j) THEN
            DO 70 k=1, M
 70            S(i,j) = S(i,j) + (x(i,k)-A(i)) * (x(j,k)-A(j)) / M
          ELSE
             S(i,j) = S(j,i)
          ENDIF

 80   CONTINUE

C  Compute covariance matrix  Eqn (13)

      const = Xs / (Beta * M)
      CALL scalar(S,const,nmax,n,nmax,n,cov)

C  Step 6.  Reduce the temperature

      Tk = r * Tk
      count = count + M

C  Step 7.  Test for convergence

      IF (accept .gt. 0) THEN
         E = E / accept
      ELSE
         E = 1.0 / epsilon
      ENDIF
      IF (abs((E - fopt) / E) .lt. epsilon) THEN
         status = solved
         GOTO 99
      ENDIF

      CALL vcopy(xopt,xi,n)
      fi = fopt
      IF (count .lt. maxits) GOTO 10

C  Maxits reached without convergence

      status = limit

C  Return to calling program

 99   maxits = count
      CALL vcopy(xopt,xi,n)

      END
```

```
      PROGRAM sa
C  Simulated annealing program to find the global optimum of a
C  continuous function.

C  PROGRAMMER: Neil R. Sumner                                DATE: 1/9/94

      INTEGER n,iseed,maxits,solved,status
C     Set number of dimensions for function
      PARAMETER (n = 2)
      REAL T0,deltax(n),x(n),lowlim(n),uplim(n),fopt
      REAL rastrig,ran1
      EXTERNAL rastrig,ran1

C     Initial temperature, step size & maximum number of function
C        evaluations
      DATA T0, maxits / 1.0, 5000 /
      DATA deltax / 2.0, 2.0 /

C     Constraints (lower and upper limits for x1, x2, ...)
      DATA lowlim / -2.0, -2.0 /
      DATA uplim / 2.0, 2.0 /

      DATA solved / 0 /

C  Seed for random number generator

      WRITE(*,'(/A)') ' Simulated Annealing Optimisation'
      WRITE(*,'(/A$)') ' Enter seed for random number generator: '
      READ(*,*) iseed
      IF (iseed .ge. 0) iseed = -1 * iseed

      CALL startpt(lowlim,uplim,x,n,iseed)

      CALL corana(rastrig,x,lowlim,uplim,T0,deltax,n,iseed,maxits,fopt,
     &   status)

      IF (status .eq. solved) THEN
         WRITE(*,10) fopt, x
 10      FORMAT(//' Optimisation Complete'/' Estimate of global optimum'
     &      ' =',F12.6,/' at x =',9F10.4,/)
         WRITE(*,'(A,I6)') ' Number of function evaluations = ',maxits
      ELSE
         WRITE(*,20) fopt, x
 20      FORMAT(//' Method has NOT converged'/' Current function value'
     &      ' =',F12.6,/' at x =',9F10.4,/)
         WRITE(*,'(A,I6)') ' Number of function evaluations = ',maxits
      ENDIF

      END



      SUBROUTINE corana(f,xi,lowlim,uplim,Tk,v,n,iseed,maxits,fopt,
     &   status)
C  Simulated annealing program based on the algorithm by Corana, A.,
C  Marchesi, M., Martini, C. and Ridella, S. (1987) Minimizing
C  multimodal functions of continuous variables with the "simulated
C  annealing" algorithm. ACM Trans. Math Software, 13(3), pp262-280.
C
C  PROGRAMMER: Neil R. Sumner                                DATE: 1/9/94

C  Variables  f              : function to be minimised
C             xi             : initial guess, returns xopt
```

```
C             uplim,lowlim    : upper and lower constraints
C             Tk              : temperature
C             n               : number of variables in function f
C             iseed           : seed for random number generator
C             maxits          : maximum number of iterations
C             fopt            : best function evaluation found
C             status          : returns 0 if convergence occurs

C             xopt            : x-vector corresponding to fopt
C             rt              : reduction factor for step length changes
C             count           : function evaluation counter
C             v               : step size vector
C             fi              : current function value
C             xstep           : current step size

C  Parameters:
      INTEGER n,iseed,maxits,status
      REAL f,v(n),xi(n),lowlim(n),uplim(n),Tk,fopt

C  Internal variables:
      INTEGER nmax
      PARAMETER (nmax = 2)
      INTEGER Nt,Ns,num(nmax),count,solved,limit,m,j,h,u
      REAL ran1,rt,epsilon,fi,fprime,fk,fkstar,p,pprime,xstep,r,
     &   xopt(nmax),xprime(nmax),c(nmax)
      EXTERNAL f,ran1

      DATA solved,limit / 0, 1 /
      DATA Ns, rt, epsilon / 20, 0.85, 0.0001 /
      DATA c / 2.0, 2.0 /
      Nt = max(100, 5*n)

C  Check array sizes

      IF (n .gt. nmax) STOP 'ERROR: Increase size of nmax in Corana'

C  Step 0.  Initialisation

      fi = f(xi,n)
      CALL vcopy(xi,xopt,n)
      fopt = fi
      count = 1

C  Iterate until convergence criteria satisfied

 10   CONTINUE

C     Iterate Nt times before reducing temperature
      DO 40 m=1, Nt

C        Iterate Ns times before changing the step size
         DO 30 j=1, Ns

C  Step 1.  Generate a random point along the direction h

C           Iterate through all n dimensions of parameter space
            DO 30 h=1, n

               CALL vcopy(xi,xprime,n)
 20            r = 2.0 * ran1(iseed) - 1.0
                  xstep = r * v(h)
                  xprime(h) = xi(h) + xstep
```

```
C  Step 2.  If coordinate lies outside function domain go to step 1

                  IF ((xprime(h).gt.uplim(h)).or.
     &               (xprime(h).lt.lowlim(h))) GOTO 20

C  Step 3.  Evaluate function

               fprime = f(xprime,n)
               IF (fprime .lt. fi) THEN
C                 accept the new point
                  CALL vcopy(xprime,xi,n)
                  fi = fprime
                  num(h) = num(h) + 1
                  IF (fprime .lt. fopt) THEN
                     CALL vcopy(xprime,xopt,n)
                     fopt = fprime
                  ENDIF
               ELSE
C                 Metropolis move
                  p = exp((fi - fprime) / Tk)
                  pprime = ran1(iseed)
                  IF (pprime .lt. p) THEN
C                    accept Metropolis move
                     CALL vcopy(xprime,xi,n)
                     fi = fprime
                     num(h) = num(h) + 1
                  ENDIF
               ENDIF

C  Step 4.  If h <= n goto step 1

 30      CONTINUE

C  Step 5.  Update the step vector v

         DO 40 u=1, n
            IF (real(num(u)) .gt. 0.6*Ns) THEN
               v(u) = v(u) * (1.0 + c(u) * (real(num(u))/Ns -0.6) / 0.4)
            ELSE IF (real(num(u)) .lt. 0.4*Ns) THEN
               v(u) = v(u) / (1.0 + c(u) * (0.4-real(num(u))/ Ns) / 0.4)
            ENDIF
            v(u) = amin1(v(u), (uplim(u) - lowlim(u)) / 2.0 )
            num(u) = 0

 40   CONTINUE

C  Step 6.  Reduce the temperature

      Tk = rt * Tk
      fkstar = fk
      fk = fi
      count = count + Nt * Ns * n

C  Step 7.  Test for convergence (different to that used by Corana)

      IF ((abs(fk-fkstar).1e.epsilon).and.(abs(fk-fopt).1e.epsilon))THEN
         status = solved
         GOTO 99
      ENDIF

      CALL vcopy(xopt,xi,n)
      fi = fopt
      IF (count .lt. maxits) GOTO 10
```

```
C  Maxits reached without convergence

      status = limit

C  Return to calling program

 99   maxits = count
      CALL vcopy(xopt,xi,n)

      END
```

```
      SUBROUTINE press(hartman,lowlim,uplim,temptr,redn,n,
     &              ncycles,iseed,count,fopt,status)
C  Simulated annealing program to find the global optimum of a
C  continuous function.

C  PROGRAMMER: Neil R. Sumner                              DATE: 21/12/94

      INTEGER ndim,np1,iseed,ncycles,maxits,iter,count,ffail,pfail,i,j,
     &        k,solved,limit,n,status
C     Set number of dimensions for function
      PARAMETER (ndim = 3, np1 = ndim+1)
      REAL temptr,redn,p(np1,ndim),y(np1),pb(ndim),yb,x(ndim),
     &     lowlim(n),uplim(n),ftol,ptol,test,reldif
      REAL hartman,ran1
      EXTERNAL hartman,ran1
      INTEGER rstart

      DATA solved,limit / 0, 1 /

C     Fractional convergence tolerance & Parameter convergence tolerance
c      DATA ftol, ptol / 0.0001, 0.001 /
      DATA ftol, ptol / 0.00001, 0.1 /

C     Maximum number of iterations per cycle & maximum number of cycles
c      DATA maxits, ncycles / 500, 20 /
      DATA maxits / 500 /
      DATA temprn / 3.D0 /

      IF(n .ne. ndim) STOP'Error: Dimension incorrectly set in press'

C  Compute number of cycles required to reduce temperature by a factor
C  of temprn

      eps = 0.0001
      rstart = ifix(-log(temprn) / log(redn-eps)) + 1

      DO 20 i=1, np1
         CALL guess(lowlim,uplim,p,i,ndim,iseed)
         DO 10 j=1, ndim
 10         x(j) = p(i,j)
 20         y(i) = hartman(x,ndim)
      count = np1

C  Put best parameter values in pb & best y-values in yb.

      yb = 100 000.0
      DO 40 i=1, np1
         IF (y(i) .lt. yb) THEN
            DO 30 j=1, ndim
               pb(j) = p(i,j)
 30         CONTINUE
            yb = y(i)
         ENDIF
 40   CONTINUE

C  Start optimisation using annealing (round 1).

c     WRITE(*,'(//A)') ' - - - Commencing Annealing - - -'
      DO 80 i=1, ncycles
         iter = maxits
c        WRITE(*,'(/A,F7.2)') ' Annealing Temperature =', temptr
         CALL amebsa(p,y,np1,ndim,ndim,pb,yb,ftol,hartman,iter,temptr)
         count = count + maxits - iter
```

```
C        Perform restart placing yb in simplex if required
         itest = mod(i,rstart)
         IF ((itest .eq. 0) .and. (yb .lt. y(1))) THEN
C           Insert best parameter estimates into simplex
            high = yb
            DO 444 j=1, np1
               IF (y(j) .gt. high) THEN
                  ihigh = j
                  high = y(j)
               ENDIF
 444        CONTINUE
            DO 466 j=1, ndim
 466           p(ihigh,j) = pb(j)
               y(ihigh) = yb
         ENDIF

c        WRITE(*,'(/,1X,A,4X,A)') 'Contents of simplex','RSS'
c        DO 60 k=1, np1
c60         WRITE(*,70) (p(k,j),j=1,ndim), y(k)
c70         FORMAT(1X,10F10.3)
         IF (iter .gt. 0) GOTO 90
         temptr = temptr * redn
 80   CONTINUE

C  Continue optimisation with simplex method (round 2).

c     WRITE(*,'(//A)')
c    &    ' Annealing complete - continuing with simplex method'
 90   ffail = 0
      pfail = 0
      temptr = 0.0
c     DO 110 i=1, ncycles
         iter = maxits
c        WRITE(*,'(/A,I3)') ' Simplex method Cycle',i
         call amebsa(p,y,np1,ndim,ndim,pb,yb,ftol,hartman,iter,temptr)

C  Count number of function evaluations

         count = count + maxits - iter

C  Check iter and parameters for convergence.

c        IF (iter .gt. 0 .or. ffail .eq. 1) THEN
c           WRITE(*,*) 'Fractional tolerance convergence test passed'
c           ftol = 0.0000001
c           ffail = 1
c        ELSE
c           WRITE(*,*) 'Fractional tolerance convergence test failed'
c        ENDIF

c        test = 0.0
c        DO 100 j=1, ndim
c           DO 100 k=1, np1
c              reldif = abs(pb(j) - p(k,j)) / pb(j)
c              test = amax1(test,reldif)
c100     CONTINUE
c        IF (test .le. ptol .or. pfail .eq. 1) THEN
c           WRITE(*,*) 'Parameter convergence test passed'
c           pfail = 1
c        ELSE
c           WRITE(*,*) 'Parameter convergence test failed'
c        ENDIF
```

```
c        IF (ffail .eq. 1 .and. pfail .eq. 1) GOTO 120

 110  CONTINUE

c     status = limit
c     WRITE(*,115) yb, pb
c115  FORMAT(//' Method has NOT converged'/' Current function value'
c    &    ' =',F10.6,/' at x =',9F10.4,/)
c     WRITE(*,'(A,I6)') ' Number of function evaluations = ',count
c     fopt = yb
c     RETURN

c120  status = solved
c120  WRITE(*,125) yb, pb
c125  FORMAT(//' Optimisation Complete'/' Estimate of global optimum'
c    &    ' =',F12.6,/' at x =',9F10.4,/)
c     WRITE(*,'(A,I6)') ' Number of function evaluations = ',count

      fopt = yb

      END



C  (C) Copr. 1986-92 Numerical Recipes Software #p21E6W)1.1&|u2j3152.
      FUNCTION amotsa(p,y,psum,mp,np,ndim,pb,yb,funk,ihi,yhi,fac)
C  Modified by Neil Sumner to call test functions 23/8/94
      INTEGER ihi,mp,ndim,np,NMAX
      REAL amotsa,fac,yb,yhi,p(mp,np),pb(np),psum(np),y(mp),funk
      PARAMETER (NMAX=200)
      EXTERNAL funk
CU    USES funk,ran1
      INTEGER idum,j
      REAL fac1,fac2,tt,yflu,ytry,ptry(NMAX),ran1
      COMMON /ambsa/ tt,idum
      fac1=(1.-fac)/ndim
      fac2=fac1-fac
      do 11 j=1,ndim
        ptry(j)=psum(j)*fac1-p(ihi,j)*fac2
11    continue
      ytry=funk(ptry,np)
      if (ytry.le.yb) then
        do 12 j=1,ndim
          pb(j)=ptry(j)
12      continue
        yb=ytry
      endif
      yflu=ytry-tt*log(ran1(idum))
      if (yflu.lt.yhi) then
        y(ihi)=ytry
        yhi=yflu
        do 13 j=1,ndim
          psum(j)=psum(j)-p(ihi,j)+ptry(j)
          p(ihi,j)=ptry(j)
13      continue
      endif
      amotsa=yflu
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #p21E6W)1.1&|u2j3152.
```

```
C  (C) Copr. 1986-92 Numerical Recipes Software #p21E6W)1.1&|u2j3152.
      SUBROUTINE amebsa(p,y,mp,np,ndim,pb,yb,ftol,funk,iter,temptr)
C  Modified by Neil Sumner to call test functions 24/8/94
      INTEGER iter,mp,ndim,np,NMAX
      REAL ftol,temptr,yb,p(mp,np),pb(np),y(mp),funk
      PARAMETER (NMAX=200)
      EXTERNAL funk
CU    USES amotsa,funk,ran1
      INTEGER i,idum,ihi,ilo,inhi,j,m,n
      REAL rtol,sum,swap,tt,yhi,ylo,ynhi,ysave,yt,ytry,psum(NMAX),
     *amotsa,ran1
      COMMON /ambsa/ tt,idum
      tt=-temptr
1     do 12 n=1,ndim
        sum=0.
        do 11 m=1,ndim+1
          sum=sum+p(m,n)
11      continue
        psum(n)=sum
12    continue
2     ilo=1
      inhi=1
      ihi=2
      ylo=y(1)+tt*log(ran1(idum))
      ynhi=ylo
      yhi=y(2)+tt*log(ran1(idum))
      if (ylo.gt.yhi) then
        ihi=1
        inhi=2
        ilo=2
        ynhi=yhi
        yhi=ylo
        ylo=ynhi
      endif
      do 13 i=3,ndim+1
        yt=y(i)+tt*log(ran1(idum))
        if(yt.le.ylo) then
          ilo=i
          ylo=yt
        endif
        if(yt.gt.yhi) then
          inhi=ihi
          ynhi=yhi
          ihi=i
          yhi=yt
        else if(yt.gt.ynhi) then
          inhi=i
          ynhi=yt
        endif
13    continue
      rtol=2.*abs(yhi-ylo)/(abs(yhi)+abs(ylo))
      if (rtol.lt.ftol.or.iter.lt.0) then
        swap=y(1)
        y(1)=y(ilo)
        y(ilo)=swap
        do 14 n=1,ndim
          swap=p(1,n)
          p(1,n)=p(ilo,n)
          p(ilo,n)=swap
14      continue
        return
      endif
```

```
      iter=iter-2
      ytry=amotsa(p,y,psum,mp,np,ndim,pb,yb,funk,ihi,yhi,-1.0)
      if (ytry.le.ylo) then
        ytry=amotsa(p,y,psum,mp,np,ndim,pb,yb,funk,ihi,yhi,2.0)
      else if (ytry.ge.ynhi) then
        ysave=yhi
        ytry=amotsa(p,y,psum,mp,np,ndim,pb,yb,funk,ihi,yhi,0.5)
        if (ytry.ge.ysave) then
          do 16 i=1,ndim+1
            if(i.ne.ilo)then
              do 15 j=1,ndim
                psum(j)=0.5*(p(i,j)+p(ilo,j))
                p(i,j)=psum(j)
15            continue
              y(i)=funk(psum,np)
            endif
16        continue
          iter=iter-ndim
          goto 1
        endif
      else
        iter=iter+1
      endif
      goto 2
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #p21E6W)1.1&|u2j3152.
```

```
      PROGRAM sa
C  Simulated annealing program to find the global optimum of a
C  continuous function.

C  PROGRAMMER: Neil R. Sumner                                DATE: 10/11/94

      INTEGER n,m,iseed,maxits,solved,status
C     Set number of dimensions for function
      PARAMETER (n = 3)
      REAL T0,redn,h,hmin,x(n),lowlim(n),uplim(n),fopt
      REAL hartman,ran1
      EXTERNAL hartman,ran1

C     Initial temperature, step size & maximum number of function
C        evaluations
      DATA T0,m,redn,h,hmin,maxits /10000.0, 10, 0.9, 1.0, 1.E-7, 9000/

C     Constraints (lower and upper limits for x1, x2, ...)
      DATA lowlim / 0.0, 0.0, 0.0 /
      DATA uplim / 1.0, 1.0, 1.0 /

      DATA solved / 0 /

C  Seed for random number generator

      WRITE(*,'(/A)') ' Simulated Annealing Optimisation'
      WRITE(*,'(/A$)') ' Enter seed for random number generator: '
      READ(*,*) iseed
      IF (iseed .ge. 0) iseed = -1 * iseed

      CALL startpt(lowlim,uplim,x,n,iseed)

      CALL sa_opt(hartman,x,lowlim,uplim,T0,redn,h,hmin,n,m,iseed,
     &   maxits,fopt,status)

      IF (status .eq. solved) THEN
         WRITE(*,10) fopt, x
 10      FORMAT(//' Optimisation Complete'/' Estimate of global optimum'
     &      ' =',F12.6,/' at x =',9F10.4,/)
         WRITE(*,'(A,I6)') ' Number of function evaluations = ',maxits
      ELSE
         WRITE(*,20) fopt, x
 20      FORMAT(//' Method has NOT converged'/' Current function value'
     &      ' =',F12.6,/' at x =',9F10.4,/)
         WRITE(*,'(A,I6)') ' Number of function evaluations = ',maxits
      ENDIF

      END



      SUBROUTINE sa_opt(f,b1,lowlim,uplim,T0,redn,h,hmin,n,m,iseed,
     &   maxits,fopt,status)

C  Simulated Annealing program that reverts to the pattern search method
C  of Hooke and Jeeves as the temperature approaches 0.

C  PROGRAMMER: Neil R. Sumner                                DATE: 10/11/94

C  Variables  b1,b2,b3        : base points, xopt returned as b1
C             p               : pattern point
C             f1,f2,f3,fp     : function values
C             xopt,fopt       : best function evaluation found
```

```
C             uplim,lowlim    : upper and lower constraints
C             T0              : initial temperature, set to 0 for local
C                               optimisation using Hooke and Jeeves
C             h               : step length for base change calculations
C             hmin            : minimum step length
C             redn            : reduction factor for step length changes
C             count           : function evaluation counter
C             n               : number of variables in function f
C             m               : number of temperature steps k=1..m
C             iseed           : seed for random number generator
C             eps             : machine epsilon
C             f(x)            : function to be minimised
C             maxits          : maximum number of iterations
C             fopt            : best function evaluation found
C             xopt            : x-vector corresponding to fopt
C             status          : returns 0 if convergence occurs

C  Parameters:
      INTEGER n,iseed,maxits,status
      REAL f,h,b1(n),lowlim(n),uplim(n),T0,redn,fopt

C  Internal variables:
      INTEGER ncycles,count,nmax,solved,limit,m,i,k,accept,reject
      PARAMETER (nmax = 10)
      LOGICAL global
      REAL ran1,eps,f1,f2,fp,p,pprime,Tk,phi,
     &   b2(nmax),xopt(nmax),xp(nmax),improv,c,hredn
      EXTERNAL f,ran1

      DATA solved,limit / 0, 1 /
      DATA ncycles, hredn, c, eps / 10, 0.1, 1000.0, 1.0E-5 /

C  Check array sizes

      IF (n .gt. nmax) STOP 'ERROR: Increase size of nmax in global'

C  Set type of run required: local (Hooke and Jeeves) or global (SA)

      IF (T0 .gt. eps) THEN
         global = .true.
      ELSE
         global = .false.
      ENDIF

C  Step 0.  Initialise

      count = 1
      k = 0
      i = 1
      Tk = T0
      f1 = f(b1,n)
      CALL vcopy(b1,xopt,n)
      fopt = f1
      accept = 0
      reject = 0
      WRITE(*,'(2f10.4,A,9f8.4)') h, f1, '  Starting values', b1

C  Step 1.  Carry out exploration steps

 10   CALL explor(f,b1,b2,xopt,f1,f2,fopt,h,uplim,lowlim,
     &           n,iseed,count,global)
          improv = f1 - f2
```

```
C  Step 2.  Try a pattern move

          IF (improv .gt. eps) THEN
             CALL patmov(f,b1,b2,xp,fp,lowlim,uplim,n,count)
          ELSE IF (GLOBAL) THEN
              reject = reject + 1
              GOTO 20
          ELSE IF (h .lt. hmin) THEN
C             Termination criteria satisfied
              status = solved
              GOTO 99
          ELSE
              h = h * hredn
              GOTO 10
          ENDIF

C  Step 3.  Accept the new point, otherwise make a Metropolis move

          IF (fp .lt. f2) THEN
             CALL vcopy(xp,b1,n)
             f1 = fp
             IF (fp .lt. fopt) THEN
                CALL vcopy(xp,xopt,n)
                fopt = fp
             ENDIF
             accept = accept + 1

          ELSE IF (global) THEN

C            Accept or reject the move with acceptance probability p

             p = exp(c * (f2 - fp) / Tk)
             pprime = ran1(iseed)
             IF (pprime .lt. p) THEN
C               Accept the point
                CALL vcopy(xp,b1,n)
                f1 = fp
                accept = accept + 1
             ELSE
C               Reject the point (global case)
                CALL vcopy(b2,b1,n)
                f1 = f2
                WRITE(*,'(2f10.4,A,9f8.4)') h, f1,'  Base Change     ', b1
                reject = reject + 1
             ENDIF

          ELSE
C               Reject the point (local case)
                CALL vcopy(b2,b1,n)
                f1 = f2
                WRITE(*,'(2f10.4,A,9f8.4)') h, f1,'  Base Change     ', b1
          ENDIF

C  Step 4. If i < ncycles then goto step 1; else it is time to reduce Tk

 20       IF (i .lt. ncycles) THEN
             i = i + 1
             GOTO 10
          ENDIF

          IF (global) THEN
             CALL vcopy(xopt,b1,n)
             f1 = fopt
```

```
            IF (k .lt. m) THEN

C              Calculate observed acceptance ratio
               phi = real(accept) / (real(accept + reject))
               accept = 0
               reject = 0
               WRITE(*,'(A,f10.1,A,f6.3)') ' Temperature =',Tk,
     &                '   Acceptance ratio =',phi

C              Cooling schedule
               Tk = redn * Tk
               k = k + 1
               i = 1

               GOTO 10
            ENDIF
         ENDIF

C  Use method of Hooke and Jeeves to find local minimum

         global = .false.
         IF (count .lt. maxits) GOTO 10

C  Maxits reached without convergence

      status = limit

C  Return to calling program

 99   maxits = count
      CALL vcopy(xopt,b1,n)

      END



      SUBROUTINE explor(f,xold,xnew,xopt,fold,fnew,fopt,h,uplim,lowlim,
     &           n,iseed,count,global)
C  Carry out exploration steps about xold, put new base point in xnew

C  Variables  xold,xnew       : base points
C             fold,fnew       : function values
C             xopt,fopt       : best function evaluation found
C             uplim,lowlim    : upper and lower constraints
C             h               : step length for base change calculations
C             iseed           : seed for random number generator
C             count           : function evaluation counter
C             n               : number of variables in function f
C             global          : flag for global / local optimisation

C  Parameters:
      LOGICAL global
      INTEGER n,iseed,count
      REAL f,xold(n),xnew(n),fold,fnew,h,xopt(n),fopt,uplim(n),lowlim(n)
      EXTERNAL f

C  Internal variables:

      INTEGER j
      REAL ran1,r,xstep,rdiv2,fval

C  Step 1.  Explore surface in each of n directions
```

```
      r = 1.0

      CALL vcopy(xold,xnew,n)
      fnew = fold
      IF (global) r = ran1(iseed)
      xstep = r * h

C  Consider component x(j) in Rn

      DO 30 j=1, n

C  Compute next random step

         IF (xold(j) .ge. uplim(j)) GOTO 15
         xnew(j) = xold(j) + xstep
         IF (xnew(j) .gt. uplim(j)) THEN
            rdiv2 = r
 10         rdiv2 = rdiv2 / 2
               xnew(j) = xold(j) + rdiv2 * h
               IF (xnew(j) .gt. uplim(j)) GOTO 10
         ENDIF

C  Compute next function evaluation

         count = count + 1
         fval = f(xnew,n)
         IF (fval .lt. fnew) THEN
            fnew = fval
            IF (fval .lt. fopt) THEN
               CALL vcopy(xnew,xopt,n)
               fopt = fval
            ENDIF
            GOTO 30
         ENDIF

C  If fnew > fi try a step in the opposite direction

 15      IF (xold(j) .le. lowlim(j)) THEN
            xnew(j) = xold(j)
            GOTO 30
         ENDIF
         xnew(j) = xold(j) - xstep
         IF (xnew(j) .lt. lowlim(j)) THEN
            rdiv2 = r
 20         rdiv2 = rdiv2 / 2
               xnew(j) = xold(j) - rdiv2 * h
               IF (xnew(j) .lt. lowlim(j)) GOTO 20
         ENDIF

C  Evaluate function again

         count = count + 1
         fval = f(xnew,n)
         IF (fval .lt. fnew) THEN
            fnew = fval
            IF (fval .lt. fopt) THEN
               CALL vcopy(xnew,xopt,n)
               fopt = fval
            ENDIF
         ELSE
            xnew(j) = xold(j)
         ENDIF
 30    CONTINUE
```

```
      WRITE(*,'(2f10.4,A,9f8.4)') h, fnew, '  Exploration    ', xnew
      END


      SUBROUTINE patmov(f,b1,b2,xp,fp,lowlim,uplim,n,count)
C  Pattern move from b1 in the direction of b2, p is the new point

C  Variable   b1,b2          : base points
C             xp             : pattern point
C             fp             : function values
C             uplim,lowlim   : upper and lower constraints
C             n              : number of variables in function f
C             count          : function evaluation counter

      INTEGER n,count,i
      REAL f, b1(n), b2(n), xp(n), fp, lowlim(n), uplim(n)
      EXTERNAL f

      DO 10 i=1, n
         xp(i) = 2.0 * b2(i) - b1(i)
C        Constrain pattern move to stay inside function domain
         xp(i) = amin1(xp(i),uplim(i))
         xp(i) = amax1(xp(i), lowlim(i))
 10   CONTINUE
      fp = f(xp,n)
      count = count + 1
      WRITE(*,'(f20.4,A,9f8.4)')  fp, '  Pattern Move   ', xp
      END
```

TEST FUNCTIONS

```
      REAL FUNCTION hartman(x,n)
C  Hartman's Family of test functions (F4) Butler, R.A.R. and Slaminka,
C  E.E. (1992) An Evaluation of the sniffer global optimization
C  algotithm using standard test functions. J. Comp. Physics 99,
C  pp28-32.
C
      INTEGER n,i,j
      REAL x(n),a(4,3),c(4),p(4,3),temp,sum,large
      DATA a / 3.0,0.1,3.0,0.1,10.0,10.0,10.0,10.0,30.0,35.0,30.0,35.0 /
      DATA c / 1.0, 1.2, 3.0, 3.2 /
      DATA p / 0.36890,0.46990,0.10910,0.03815,
     &         0.1170,0.4387,0.8732,0.5743,
     &         0.2673,0.7470,0.5547,0.8828 /
      DATA large / 999.9 /

C     Constraints

      DO 5 i=1, 3
         IF ((x(i) .lt. 0.0) .or. (x(i) .gt. 1.0)) THEN
            hartman = large
            RETURN
         ENDIF
 5    CONTINUE

      sum = 0.0
      DO 20 i=1, 4
         temp = 0.0
         DO 10 j=1, 3
 10         temp = temp + a(i,j) * (x(j) - p(i,j)) ** 2
 20   sum = sum + c(i) * exp(-temp)

      hartman = - sum

      END



      REAL FUNCTION rastrig(x,n)
C  Rastrigin test function from Benke and Skinner (1991) A direct
C  search algorithm for global optimisation of multivariate functions.
C  Aust. Comp. J. 23(2) pp73-85.
C
      INTEGER n
      REAL x(n),large
      DATA large / 999.9 /

      DO 5 i=1, 2
         IF ((x(i) .lt. -2.0) .or. (x(i) .gt. 2.0)) THEN
            rastrig = large
            RETURN
         ENDIF
 5    CONTINUE

      rastrig = x(1)**2 + x(2)**2 - cos(18*x(1)) - cos(18*x(2))
      END
```

```
      REAL FUNCTION shubert(x,n)
C  Penalized Shubert function from Gunell, T. & C  Yazgan, B. (1992)
C  A new global optimization method based on simulated annealing
C  algorithm. Proc. Eng. Des. & Anal., 47(4), 199-202.
C
      INTEGER n,i
      REAL x(n),b,temp1,temp2,large
      DATA b / 0.5 /
      DATA large / 999.9 /

C     Constraints

      DO 5 i=1, 2
         IF ((x(i) .lt. -10) .or. (x(i) .gt. 10.0)) THEN
            shubert = large
            RETURN
         ENDIF
 5    CONTINUE

      temp1 = 0.0
      temp2 = 0.0
      DO 10 i=1, 5
         temp1 = temp1 + i * cos((i + 1) * x(1) + i)
 10      temp2 = temp2 + i * cos((i + 1) * x(2) + i)
      shubert = temp1 * temp2 + b* ((x(1)+1.4251)**2 + (x(2)+0.8003)**2)
      END
```

SUPPORT SUBROUTINES

```
      SUBROUTINE vcopy(a,b,n)
C  Copy contents of vector a to vector b

      INTEGER n,i
      REAL a(n),b(n)

      DO 10 i=1, n
 10      b(i) = a(i)
      END



      SUBROUTINE startpt(lowlim,uplim,x,n,iseed)
C  Generate a random starting point in n dimensional space

      INTEGER iseed, n, j
      REAL lowlim(n), uplim(n), x(n), ran1

      DO 10 j=1, n
 10      x(j) = ran1(iseed) * (uplim(j) - lowlim(j)) + lowlim(j)

      END



      SUBROUTINE choleski(a,n,np,l)
C  Choleski's algorithm.  Burden, R.L. and Faires, J.D. (1989)
C  Numerical Analysis. PWS-KENT

      INTEGER n,np
      REAL a(np,np),l(np,np)
      INTEGER i,j,k
      REAL sum

C  Step 1.

      l(1,1) = sqrt(a(1,1))

C  Step 2.

      DO 10 j=2, n
 10      l(j,1) = a(j,1) / l(1,1)

C  Step 3.

      DO 50 i=2, n-1

C  Step 4.

         sum = 0.0
         DO 20 k=1, i-1
 20         sum = sum + l(i,k)**2
         l(i,i) = sqrt(a(i,i) - sum)

C  Step 5.

         DO 40 j=i+1, n
            sum = 0.0
            DO 30 k=1, i-1
 30            sum = sum + l(j,k) * l(i,k)
```

```
 40         l(j,i) = (1.0 / l(i,i)) * (a(j,i) - sum)
 50    CONTINUE
C  Step 6.

      sum = 0.0
      DO 60 k=1, n-1
 60      sum = sum + l(n,k)**2
      l(n,n) = sqrt(a(n,n) - sum)
      END



      SUBROUTINE guess(lowlim,uplim,p,i,n,iseed)
C  Generate random starting points in n dimensional space.

      INTEGER iseed, n, i, j
      REAL lowlim(n), uplim(n), p(n+1,n), ran1

      DO 10 j=1, n
 10      p(i,j) = ran1(iseed) * (uplim(j) - lowlim(j)) + lowlim(j)

      END



      SUBROUTINE MATMUL(S1,NR1,NRL1,NC1,NCL1,S2,NR2,NRL2,NC2,NCL2,S3,
     #  ITY)
C
C ****************************************************************
C * PURPOSE:-    MULTIPLY TWO MATRICES
C * PARAMETERS:-          S1   INPUT MATRIX ONE
C *                  NR1,NC1   NUMBER OF ROWS AND COLUMNS IN S1
C *                 NRL1,NCL1  NUMBER OF R AND C USED IN L.H. MATRIX
C *                       S2   INPUT MATRIX TWO
C *                  NR2,NC2   NUMBER OF ROWS AND COLUMNS IN S2
C *                 NRL2,NCL2 NUMBER OF R AND C USED IN R.H. MATRIX
C *                       S3   RESULTING MATRIX
C *                      ITY   TYPE DESIGNATOR. IF = 1 THEN S3 IS
C *                            =S1.IF ITY = 2 THEN ANSWER RETURNED IN S2
C ****************************************************************
C
      DIMENSION S1(NR1,NC1),S2(NR2,NC2),S3(NR1,NC2)
C
C  CHECK MULTIPICATION IS POSSIBLE
      DOUBLE PRECISION SUM
C
      IF(NC1.NE.NR2)GOTO 100
C
      DO 30 J = 1 , NRL1
        DO 25 K = 1 , NCL2
          SUM = 0.0D0
          DO 20 L = 1 , NRL2
            SUM = S1(J,L) * S2(L,K) + SUM
 20       CONTINUE
          S3(J,K) = SUM
 25     CONTINUE
 30   CONTINUE
      IF(ITY.LT.0)GOTO 45
      DO 35 J = 1 , NRL1
         DO 40 K = 1 , NCL2
          IF(ITY.EQ.1)S1(J,K)=S3(J,K)
          IF(ITY.EQ.2)S2(J,K)=S3(J,K)
 40       S3(J,K)=0.
```

```
 35   CONTINUE
 45   RETURN
 100  WRITE(2,110)NC1,NR2
 110  FORMAT(' COLUMN NUMBER OF MATRIX 1 .NE. ROW NUMBER ',/
     1,' OF MATRIX 2 :- ',2I6)
      RETURN
      END



      SUBROUTINE MATADD(S1,NR,NRL,NC,NCL,S2,S3)
C
C ****************************************************************
C * PURPOSE:-    ADDING MATRICES
C * PARAMETERS:-          S1   INPUT MATRIX ONE
C *                    NR,NC   NUMBER OF ROWS AND COLUMNS
C *                  NRL,NCL  NUMBER OF ROWS AND COLUMNS ACTUALLY
OPERATED ON
C *                       S2   INPUT MATRIX TWO
C *                       S3   RESULT MATRIX
C ****************************************************************
C
      DIMENSION S1(NR,NC),S2(NR,NC),S3(NR,NC)
C
      DOUBLE PRECISION SUM
C
      DO 10 J = 1 , NCL
        DO 15 I = 1 , NRL
        SUM = S1(I,J) + S2(I,J)
 15     S3(I,J) = SUM
 10   CONTINUE
      RETURN
      END



      SUBROUTINE SCALAR(S1,VAL,NR,NRL,NC,NCL,S2)
C
C ****************************************************************
C * PURPOSE:-    MULTIPLE MATRIX BY A SCALAR (CONSTANT)
C * PARAMETERS:-             S1   INPUT MATRIX
C *                         VAL   SCALAR
C *                       NR,NC   NUMBER OF ROWS AND COLUMNS
C *                     NRL,NCL  NUMBER OF R AND C ACTUALLY USED
C * SPECIAL NOTE:-  THE RESULT IS RETURNED IN THE ORGINAL MATRIX S1
C ****************************************************************
C
      DIMENSION S1(NR,NC),S2(NR,NC)
C
      DO 10 J = 1 , NCL
        DO 15 I = 1 , NRL
 15     S2(I,J) = S1(I,J) * VAL
 10   CONTINUE
      RETURN
      END



      SUBROUTINE  RDENT(A,N,NL)
C ************************************************************
C * PURPOSE           CREATE AN IDENTITY MATRIX OF A GIVEN SIZE
C * ARGUMENTS         A       N,N ARRAY
C *                   N       DIMENSION OF ARRAY
```

```
C ***********************************************************
      DIMENSION A(N,N)
      DO 10 I = 1,NL
      DO 20 J = 1,NL
      A(I,J) = 0.
20    CONTINUE
      A(I,I) = 1
10    CONTINUE
      RETURN
      END



C  (C) Copr. 1986-92 Numerical Recipes Software #p21E6W)1.1&|u2j3152.
      FUNCTION ran1(idum)
      INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
      REAL ran1,AM,EPS,RNMX
      PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,
     *NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
      INTEGER j,k,iv(NTAB),iy
      SAVE iv,iy
      DATA iv /NTAB*0/, iy /0/
      if (idum.le.0.or.iy.eq.0) then
        idum=max(-idum,1)
        do 11 j=NTAB+8,1,-1
          k=idum/IQ
          idum=IA*(idum-k*IQ)-IR*k
          if (idum.lt.0) idum=idum+IM
          if (j.le.NTAB) iv(j)=idum
11      continue
        iy=iv(1)
      endif
      k=idum/IQ
      idum=IA*(idum-k*IQ)-IR*k
      if (idum.lt.0) idum=idum+IM
      j=1+iy/NDIV
      iy=iv(j)
      iv(j)=idum
      ran1=min(AM*iy,RNMX)
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software #p21E6W)1.1&|u2j3152.
```