5.2 DIRECT SEARCH OR DERIVATIVE FREE OR GRADIENT-FREE METHODS

These methods search for the minimum of an objective function without calculating derivatives analytically or numerically (they use only function evaluations). They are based upon heuristic rules which make no *a priori* assumptions about the objective function. The methods tend to have much poorer convergence rates than gradient methods and in general are considered not as efficient and robust as the indirect or gradient search methods. They are robust and reliable for *large scale optimization problems* whose objective functions exhibit many local minima.

- Direct search unconstrained minimization methods include random search, grid search (e.g. factorial statistics experimental design), pattern search, Powell's method (an extension of the basic pattern search method which is a conjugate directions method), simplex method (different than the simplex LP method) and Nelder and Mead method (a more efficient simplex method)[1].
- Direct search constrained minimization methods include, genetic algorithm, particle swarm optimization, simulated annealing, ant colony optimization, Luss-Jaakola optimization[2], Tabu search, firefly algorithm, artificial bee colony algorithm, and neural network-based optimization[3].

Random search methods employ random numbers[1]. One of their advantages is that they can find the global minimum. They are not very efficient but they may be used in a preliminary search to detect to detect a region where the global minimum may be. One then switches to a more efficient method to locate the global minimum. Among random search methods the *random walk method* finds a new or improved approximation in the i-th stage as follows

$$\mathbf{x}_{i+1}=\mathbf{x}_i+\lambda \mathbf{u}_i$$

where λ is a prescribed step length and $\mathbf{u}_i$ is a unit random vector generated in the i-th stage (see Rao for details).

[1] Rao, S.S., *Engineering Optimization*, 5th ed, Wiley, 2020; Edgar, T.F. and D.M. Himmelblau, *Optimization of Chemical Processes*, McGraw-Hill, New York, NY, 1988;

[2] Englezos, P. and N. Kalogerakis, *Applied Parameter Estimation for Chemical Engineers*, Marcel-Dekker, New York, 2001;

[3] Eren, Y. et al., Introduction to optimization in *Optimization in Renewable Energy Systems,* 2017; Rao, S.S., Engineering Optimization, 5th ed, Wiley, 2020.

One proceeds as follows

1. Start with an initial point $\mathbf{X}_1$, a sufficiently large initial step length λ, a minimum allowable step length ϵ, and a maximum permissible number of iterations N.
2. Find the function value $f_1 = f(\mathbf{X}_1)$.
3. Set the iteration number as $i = 1$.
4. Generate a set of n random numbers $r_1, r_2, \ldots, r_n$ each lying in the interval $[-1, 1]$ and formulate the unit vector $\mathbf{u}$ as

$$\mathbf{u} = \frac{1}{(r_1^2 + r_2^2 + \cdots + r_n^2)^{1/2}} \begin{Bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{Bmatrix} \tag{6.19}$$

The directions generated using Eq. (6.19) are expected to have a bias toward the diagonals of the unit hypercube [6.3]. To avoid such a bias, the length of the vector, R, is computed as

$$R = (r_1^2 + r_2^2 + \cdots + r_n^2)^{1/2}$$

and the random numbers generated $(r_1, r_2, \ldots, r_n)$ are accepted only if $R \leq 1$ but are discarded if $R > 1$. If the random numbers are accepted, the unbiased random vector $\mathbf{u}_i$ is given by Eq. (6.19).

5. Compute the new vector and the corresponding function value as $\mathbf{X} = \mathbf{X}_1 + \lambda \mathbf{u}$ and $f = f(\mathbf{X})$.
6. Compare the values of f and f_1. If $f < f_1$, set the new values as $\mathbf{X}_1 = \mathbf{X}$ and $f_1 = f$ and go to step 3. If $f \geq f_1$, go to step 7.
7. If $i \leq N$, set the new iteration number as $i = i + 1$ and go to step 4. On the other hand, if $i > N$, go to step 8.
8. Compute the new, reduced, step length as $\lambda = \lambda/2$. If the new step length is smaller than or equal to ϵ, go to step 9. Otherwise (i.e. if the new step length is greater than ϵ), go to step 4.
9. Stop the procedure by taking $\mathbf{X}_{\text{opt}} \approx \mathbf{X}_1$ and $f_{\text{opt}} \approx f_1$.

This method is illustrated with the following example.

Example 6.3 Minimize $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$ using random walk method from the point $\mathbf{X}_1 = \begin{Bmatrix} 0.0 \\ 0.0 \end{Bmatrix}$ with a starting step length of $\lambda = 1.0$. Take $\epsilon = 0.05$ and $N = 100$.

SOLUTION The results are summarized in Table 6.2, where only the trials that produced an improvement are shown.

Table 6.2 Minimization of f by random walk method.

Step length, λ	Number of trials required[a]	Components of $\mathbf{X}_1 + \lambda\mathbf{u}$ 1	2	Current objective function value, $f_1 = f(\mathbf{X}_1 + \lambda\mathbf{u})$
1.0	1	−0.936 96	0.349 43	−0.063 29
1.0	2	−1.152 71	1.325 88	−1.119 86
	Next 100 trials did not reduce the function value.			
0.5	1	−1.343 61	1.788 00	−1.128 84
0.5	3	−1.073 18	1.367 44	−1.202 32
	Next 100 trials did not reduce the function value.			
0.25	4	−0.864 19	1.230 25	−1.213 62
0.25	2	−0.869 55	1.480 19	−1.220 74
0.25	8	−1.106 61	1.559 58	−1.236 42
0.25	30	−0.942 78	1.370 74	−1.241 54
0.25	6	−1.087 29	1.574 74	−1.242 22
0.25	50	−0.926 06	1.383 68	−1.242 74
0.25	23	−1.079 12	1.581 35	−1.243 74
	Next 100 trials did not reduce the function value.			
0.125	1	−0.979 86	1.505 38	−1.248 94
	Next 100 trials did not reduce the function value.			
0.062 5	100 trials did not reduce the function value.			
0.031 25	As this step length is smaller than ϵ, the program is terminated.			

[a]Out of the directions generated that satisfy $R \leq 1$, number of trials required to find a direction that also reduces the value of f.

NOTE:

6.2.3 Random Walk Method with Direction Exploitation

In the random walk method described in Section 6.2.2, we proceed to generate a new unit random vector $\mathbf{u}_{i+1}$ as soon as we find that $\mathbf{u}_i$ is successful in reducing the function value for a fixed step length λ. However, we can expect to achieve a further decrease in the function value by taking a longer step length along the direction $\mathbf{u}_i$. Thus the random walk method can be improved if the maximum possible step is taken along each successful direction. This can be achieved by using any of the one-dimensional minimization methods discussed in Chapter 5. According to this procedure, the new vector $\mathbf{X}_{i+1}$ is found as

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \lambda_i^* \mathbf{u}_i \tag{6.20}$$

where λ_i^* is the optimal step length found along the direction $\mathbf{u}_i$ so that

$$f_{i+1} = f(\mathbf{X}_i + \lambda_i^* \mathbf{u}_i) = \min_{\lambda_i} f(\mathbf{X}_i + \lambda_i \mathbf{u}_i) \tag{6.21}$$

The search method incorporating this feature is called the *random walk method with direction exploitation*.

The ***Simplex Method*** or "Sequential Simplex" method relies on geometry to create a heuristic rule for finding the minimum of a function[4]. Note that this simplex method is not the same as the Simplex method in linear programming

Kowalik and Osborn (1968) define simplex as following: *A set of N+1 points in the N-dimensional space forms a geometric figure called simplex. When the points are equidistant the simplex is said to be regular*[5]. For a function of N variables one needs a (N+1)-dimensional geometric figure or simplex to use and select points on the vertices to evaluate the function to be minimized. Thus, for a function of two variables an equilateral triangle is used whereas for a function of three variables a regular tetrahedron. In general, for a function of N variables the method proceeds as follows:

Step 1. Form an initial simplex e.g. an equidistant triangle for a function of two variables.

Step 2. Evaluate the function at each of the vertices.

Step 3. Reject the vertex where the function has the largest value. This point is replaced by another one that is found in the direction away from the rejected vertex and through the centroid of the simplex. The distance from the rejected vertex is always constant at each search step. In the case of a function of two variables the direction is from the rejected vertex through the middle of the line of the triangle that is opposite to this point. The new point together with the previous two points define a new equilateral triangle.

Step 4. Proceed until a simplex that encloses the minimum is found. Stop when the difference between two consecutive function evaluations is less than a preset value (tolerance).

Nelder and Mead (1965)[6] presented a more efficient Simplex method. A subroutine that implements the simplex method of Nelder and Mead is available by Press et al. (1992)[7].

It is also noted that one may also employ Matlab for the Nelder and Mead Simplex Algorithm[8].

---

[4] Edgar, T.F. and D.M. Himmelblau, *Optimization of Chemical Processes*, McGraw-Hill, New York, NY, 1988

[5] Kowalik, J., and M.R. Osborne, *Methods of Unconstrained Optimization Problems*, Elsevier, New York, NY, 1968;

[6] Nelder, J.A., and R. Mead, "A Simplex Method for Function Minimization", *Comp. J.*, 7, 308-313 (1965))

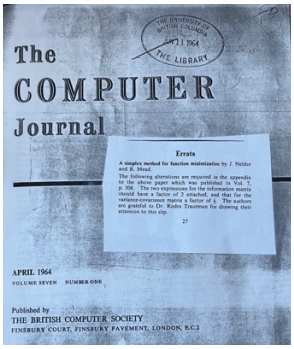
[7] Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing,* Cambridge University Press, Cambridge, UK, 1992

[8] dila (2023). Nelder and Mead Simplex Algorithm (https://www.mathworks.com/matlabcentral/fileexchange/69636-nelder-and-mead-simplex-algorithm), MATLAB Central File Exchange. Retrieved January 29, 2023.

A simplex method for function minimization

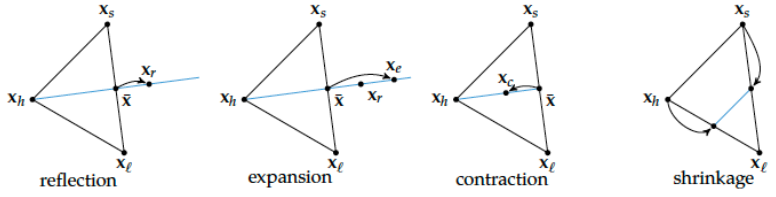
By J. A. Nelder and R. Mead†

A method is described for the minimization of a function of n variables, which depends on the comparison of function values at the $(n + 1)$ vertices of a general simplex, followed by the replacement of the vertex with the highest value by another point. The simplex adapts itself to the local landscape, and contracts on to the final minimum. The method is shown to be effective and computationally compact. A procedure is given for the estimation of the Hessian matrix in the neighbourhood of the minimum, needed in statistical estimation problems.



- Kochenderfer and Wheeler[9] present an improved algorithm based on contribution by Lagarias et al., 1998[10] . The movement of the simplex is achieved by using basic operations as follows[9].

Figure 7.10 shows the four simplex operations. Figure 7.11 shows several iterations of the algorithm.



The downhill simplex updates are also shown in the figure below[11]. Let $\vec{x}_i$ be the location of the i[th] vertex so that $f(\vec{x}_1) > f(\vec{x}_2) > \ldots > f(\vec{x}_{D+1})$. The first step is to calculate the center of the face of the simplex defined by all of the vertices other than the one we are trying to improve:

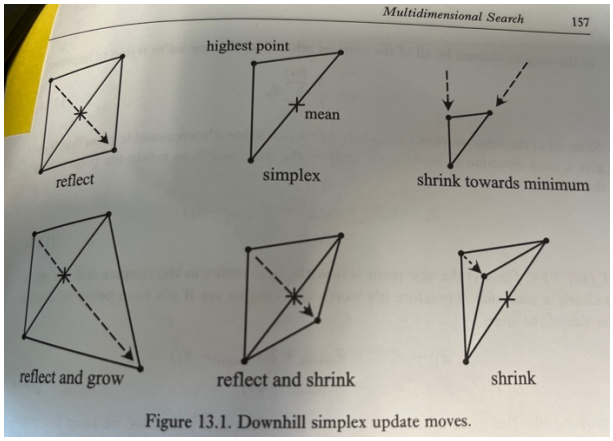


Figure 13.1. Downhill simplex update moves.

[9] Kochenderfer M.J. and T.A. Wheeler, Algorithms for Optimization, MIT Press, 2019.

[10] Lagarias, J.C., J.A. Reeds, M.H. Wright and P.E. Wright, ''Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions,'' *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[11] Gershenfeld, N., The Nature of Mathematical Modeling, CUP, 1999.

Direct Methods that can handle constraints.

The ***simulated annealing*** (SA) technique introduced by Kirkpatrick et al. (1983)[12] and independently by Cerny (1985)[13] is probably the most popular direct search method. *Annealing* refers to the cooling process of a liquid or solid[14] and SA is an optimization method based on the cooling process.

The atoms of a metal in the molten state move freely with respect to each other. With cooling and as the temperature reduces, the atoms are less mobile and get ordered to form crystals having the minimum internal energy. When the temperature of the molten metal is reduced at a very fast rate, the system may attain a polycrystalline state with higher energy than that of the crystalline state. In engineering applications, rapid cooling may introduce defects inside the material. Thus, the temperature of the heated solid (molten metal) needs to be reduced at a slow and controlled rate to eliminate defects[15]. This process of cooling at a slow rate is known as *annealing*.

The SA method tries to mimic the physical process of *annealing*. *Simulated annealing* is an algorithmic implementation of the cooling process to find the minimum of an objective function[16]. The method updates the search vector in a manner that accepts new values that decrease the objective function (assuming a minimization problem) and also new values that increase objective function values subject to a probability based on the Boltzmann–Gibbs. The minimum of an objective function represents the minimum energy of the system.

Simulated annealing (SA) uses a random search strategy, which not only accepts new positions that decrease the objective function (assuming a minimization problem), but also accepts positions that increase objective function values.

In practice, simulated annealing is implemented using the Metropolis et al. (1953) algorithm[17].

Matlab can be employed for the *simulated annealing method*.

---

[12] Kirkpatrick S., C.D. Gelatt Jr. and M.P. Vecchi, "Optimization by Simulated Annnealing", *Science*, 220, 671-680 (1983).
[13] Cerny, V., "Thermodynamic Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm", *J. Opt. Theory Applic.*, 45, 41-51 (1985).
[14] Rao, S.S., *Engineering Optimization*, 5th ed, Wiley, 2020

[15] Gershenfeld, N., The Nature of Mathematical Modeling, CUP, 1999.

[16] A. P. Engelbrecht, Computational Intelligence: An Introduction, Second Edition, 2007 John Wiley & Sons
[17] Metropolis, N., A.W. Rosenbluthh, M.N. Rosenbluth, A.H. Teller and E. Teller, "Equations of State Calculations by Fast Computing Machines", *J. Chem. Physics*, 21, 1087-1092 (1953)

Science
AAAS

Optimization by Simulated Annealing
Author(s): S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi
Source: *Science*, May 13, 1983, New Series, Vol. 220, No. 4598 (May 13, 1983), pp. 671-680
Published by: American Association for the Advancement of Science
Stable URL: https://www.jstor.org/stable/1690046

Optimization by Simulated Annealing

S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi

Summary. There is a deep and useful connection between statistical mechanics (the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature) and multivariate or combinatorial optimization (finding the minimum of a given function depending on many parameters). A detailed analogy with annealing in solids provides a framework for optimization of the properties of very large and complex systems. This connection to statistical mechanics exposes new information and provides an unfamiliar perspective on traditional optimization problems and methods.

In this article we briefly review the central constructs in combinatorial optimization and in statistical mechanics and then develop the similarities between the two fields. We show how the Metropolis algorithm for approximate numerical simulation of the behavior of a many-body system at a finite temperature provides a natural tool for bringing the techniques of statistical mechanics to bear on optimization.

We have applied this point of view to a number of problems arising in optimal design of computers. Applications to partitioning, component placement, and wiring of electronic systems are described in this article. In each context, we introduce the problem and discuss the improvements available from optimization.

Of classic optimization problems, the traveling salesman problem has received the most intensive study. To test the power of simulated annealing, we used the algorithm on traveling salesman problems with as many as several thousand cities. This work is described in a final section, followed by our conclusions.

Combinatorial Optimization

The subject of combinatorial optimization (*1*) consists of a set of problems that are central to the disciplines of computer science and engineering. Research in this area aims at developing efficient techniques for finding minimum or maximum values of a function of very many independent variables (*2*). This function, usually called the cost function or objective function, represents a quantitative measure of the "goodness" of some complex system. The cost function depends on the detailed configuration of the many parts of that system. We are most familiar with optimization problems occurring in the physical design of computers, so examples used below are drawn from that context. The number of variables involved may range up into the tens of thousands.

The classic example, because it is so simply stated, of a combinatorial optimization problem is the traveling salesman problem. Given a list of N cities and a means of calculating the cost of traveling between any two cities, one must plan the salesman's route, which will pass through each city once and return finally to the starting point, minimizing the total cost. Problems with this flavor arise in all areas of scheduling and design. Two subsidiary problems are of general interest: predicting the expected cost of the salesman's optimal route, averaged over some class of typical arrangements of cities, and estimating or obtaining bounds for the computing effort necessary to determine that route.

All exact methods known for determining an optimal route require a computing effort that increases exponentially with N, so that in practice exact solutions can be attempted only on problems involving a few hundred cities or less. The traveling salesman belongs to the large class of NP-complete (nondeterministic polynomial time complete) problems, which has received extensive study in the past 10 years (*3*). No method for exact solution with a computing effort bounded by a power of N has been found for any of these problems, but if such a solution were found, it could be mapped into a procedure for solving all members of the class. It is not known what features of the individual problems in the NP-complete class are the cause of their difficulty.

Since the NP-complete class of problems contains many situations of practical interest, heuristic methods have been developed with computational requirements proportional to small powers of N. Heuristics are rather problem-specific: there is no guarantee that a heuristic procedure for finding near-optimal solutions for one NP-complete problem will be effective for another.

There are two basic strategies for heuristics: "divide-and-conquer" and iterative improvement. In the first, one divides the problem into subproblems of manageable size, then solves the subproblems. The solutions to the subproblems must then be patched back together. For this method to produce very good solutions, the subproblems must be naturally disjoint, and the division made must be an appropriate one, so that errors made in patching do not offset the gains

S. Kirkpatrick and C. D. Gelatt, Jr., are research staff members and M. P. Vecchi was a visiting scientist at IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598. M. P. Vecchi's present address is Instituto Venezolano de Investigaciones Cientificas, Caracas 1010A, Venezuela.

$P(E) \sim e^{-E/kT}$ Boltzman

Metropolis et al J.CP 21, 1087–1092, 1953

$E_1 \to E_2$ if $E_2 < E_1$ accept more

if $E_2 > E_1$ then you accept if $R(0,1) < P_{12}$

$= e^{-\Delta E/kT}$ $\Delta E = E_2 - E_1$

$f(x)$: P_{ij} probability of moving from point X_i to X_j

$$P_{ij} = \begin{cases} 1 & \text{if } f(X_j) < f(X_i) \\ e^{-\frac{f(X_j)-f(X_i)}{kT}} & \end{cases}$$

k: B. constant

if $f(X_j) < f(X_i)$ accept more

if $f(X_j) > f(X_i)$ you still accept the more

if $R(0,1) < P_{ij}$

There are certain optimization problems such as the traveling salesman problem where simulated annealing offers an effective practical algorithm[18]. This problem is one of the most studied problems in optimization.

In this problem, a salesman must visit N cities while *minimizing the total mileage traveled*. Simulated annealing uses the Metropolis algorithm[19]

WITH THIS ALGORITHM, trades that do not lower the mileage are accepted based on the Boltzmann-Gibbs probability criterion

$$R(0,1) < e^{\frac{-\Delta D}{T}}$$

where ΔD is the change of distance implied by the trade (negative for a "good" trade; positive for a "bad" trade), T is a "synthetic temperature," and $R(0,1)$ is a random number in the interval [0,1]. D is called a "cost function," and corresponds to the free energy in the case of annealing a metal (in which case the temperature parameter would actually be kT, where k is Boltzmann's constant and T is the physical temperature in K. If T is large, many "bad" trades are accepted, and a large part of solution space is accessed.

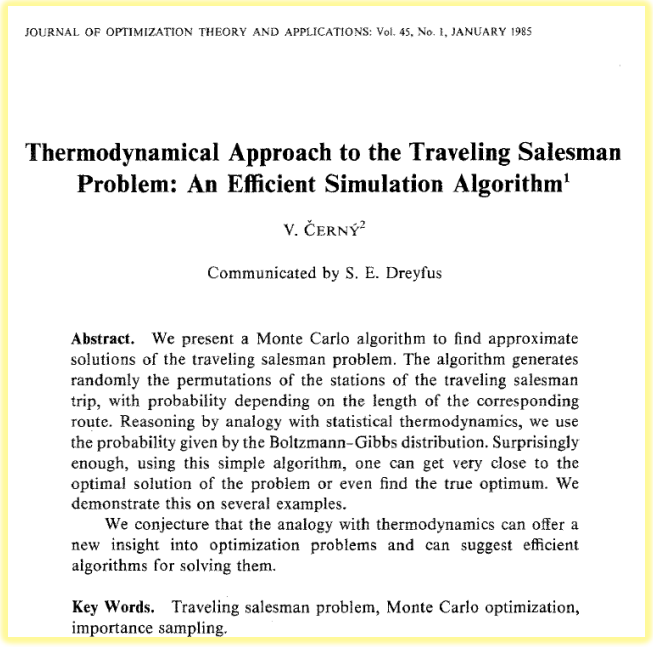
By analogy with annealing of a metal, the temperature is lowered. After making many trades and observing that the cost function declines only slowly, one lowers the temperature, and thus limits the size of allowed "bad" trades. After lowering the temperature several times to a low value, one may then "quench" the process by accepting only "good" trades in order to find the local minimum of the cost function. There are various "annealing schedules" for lowering the temperature, but the results are generally not very sensitive to the details.

P_{ij} probability $i \rightarrow j$

$P_{ij} = 1$ if $\Delta D = D_j - D_i \leq 0$

$P_{ij} = e^{-\Delta D/kT}$ $D_j - D_i > 0$

if $R(0,1) < P_{ij}$ accept more



JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 45, No. 1, JANUARY 1985

Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm[1]

V. ČERNÝ[2]

Communicated by S. E. Dreyfus

Abstract. We present a Monte Carlo algorithm to find approximate solutions of the traveling salesman problem. The algorithm generates randomly the permutations of the stations of the traveling salesman trip, with probability depending on the length of the corresponding route. Reasoning by analogy with statistical thermodynamics, we use the probability given by the Boltzmann–Gibbs distribution. Surprisingly enough, using this simple algorithm, one can get very close to the optimal solution of the problem or even find the true optimum. We demonstrate this on several examples.

We conjecture that the analogy with thermodynamics can offer a new insight into optimization problems and can suggest efficient algorithms for solving them.

Key Words. Traveling salesman problem, Monte Carlo optimization, importance sampling.

[18] Carr, Roger. "Simulated Annealing." From *MathWorld*--A Wolfram Web Resource, created by Eric W. Weisstein. https://mathworld.wolfram.com/SimulatedAnnealing.html

[19] Metropolis et al, J. *Chem. Phys*, 21, 1087-1092, 1953

LOCAL SEARCH IN COMBINATORIAL OPTIMIZATION

Edited and with a new preface by

Emile Aarts
Philips Research Laboratories, Eindhoven
Eindhoven University of Technology, Eindhoven

and

Jan Karel Lenstra
Georgia Institute of Technology, Atlanta
Eindhoven University of Technology, Eindhoven

1 INTRODUCTION

In the traveling salesman problem (TSP) we are given a set $\{c_1, c_2, \ldots, c_N\}$ of *cities* and for each pair $\{c_i, c_j\}$ of distinct cities a *distance* $d(c_i, c_j)$. Our goal is to find an ordering π of the cities that minimizes the quantity

$$\sum_{i=1}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)}).$$

This quantity is called the *tour length*, since it is the length of the tour a salesman would make when visiting the cities in the order specified by the permutation, returning at the end to the initial city. We shall concentrate in this chapter on the *symmetric* TSP, in which the distances satisfy $d(c_i, c_j) = d(c_j, c_i)$ for $1 \leqslant i, j \leqslant N$.

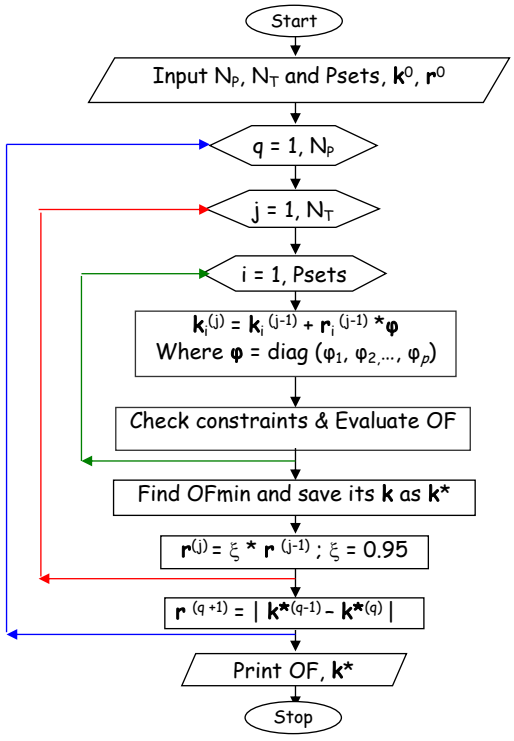
The symmetric traveling salesman problem has many applications, from very large scale integration (VLSI) chip fabrication [Korte, 1989] to X-ray crystallography [Bland & Shallcross, 1989], and a long history, for which see Lawler et al. [1985]. It is NP-hard [Garey & Johnson, 1979] so any algorithm for finding optimal tours must have a worst-case running time that grows faster than any polynomial (assuming the widely believed conjecture that P $\neq$ NP). This leaves researchers with two alternatives: either look for heuristics that merely find *near*-optimal tours, but do so quickly, or attempt to develop optimization algorithms that work well on 'real-world' rather than worst-case instances.

Because of its simplicity and applicability (or perhaps because of its intriguing name), the TSP has for decades served as an initial proving ground for new ideas related to both these alternatives. These new ideas include most of the local search variants covered in this book, which makes the TSP an ideal subject for a case study. In addition, the new ideas include many of the important advances in the related area of optimization, and to keep our discussions of local search in perspective, let us begin by noting the impressive recent progress in optimization algorithms.

The TSP is one of the major success stories for optimization. Decades of research into optimization techniques, combined with the continuing rapid growth in computer speeds and memory capacities, have led to one new record after another. Over the past 15 years, the record for the largest nontrivial TSP

The *LJ Optimization Procedure* is ione of the most reliable direct search methods[20,21,22]. It is easy to program and handles the problem of multiple optima with high reliability. An important advantage is its ability to handle multiple nonlinear constraints.

Luus – Jaakola (LJ) method



p = number of unknown variables

$\mathbf{x}$ = p-dimensional variable vector

$\mathbf{r}$ = p-dimensional region size vector

N_P = Number of passes (40)

N_T = Number of iterations (100)

Psets = number of randomly chosen points (100)

$\boldsymbol{\phi}$ = p-dimensional vector of random numbers

ξ = contraction factor (0.95)

OF = Objective Function

(Luus and Jaakola 1973; Luus, 1998)

4

NOTE : $k \equiv x$ search vector, decision variables

$r^{(0)} = |x_{max} - x_{min}|$

[20] Luus, R., and T.H.I. Jaakola, "Optimization by Direct Search and Systematic Reduction of the Search Region", *AIChE J*, 19, 760, (1973); Wang, B.C. and R. Luus, "Optimization of Non-Unimodal Systems", *Int. J. Numer. Meth. Eng.,* 11, 1235-1250 (1977

[21] Luus, R., "Determination of the Region Sizes for LJ Optimization", *Hung. J. Ind. Chem.*, 26, 281-286 (1998);

[22] Wang, B.C. and R. Luus, "Reliability of Optimization Procedures for Obtaining the Global Optimum", *AIChE J.*, 19, 619-626 (1978)

- The ***LJ Optimization Procedure*** is one of the most reliable direct search methods[13]. The method is easy to program and handles the problem of multiple optima with high reliability. A important advantage of the method is its ability to handle multiple nonlinear constraints. The adaptation of the original LJ optimization procedure to parameter estimation problems for algebraic equation models is given next.

(i) Choose an initial guess for the p-dimensional unknown parameter vector, $\mathbf{k}^{(0)}$; the *region contraction coefficient*, δ (typically δ=0.95 is used); the number of random evaluations of the objective function, N_R (typically N_R=100 is used) within an iteration; the maximum number of iterations, j_{max} (typically j_{max}=200 is used) and an initial search region, $\mathbf{r}^{(0)}$ (a typical choice is $\mathbf{r}^{(0)} = \mathbf{k}_{max} - \mathbf{k}_{min}$).

(ii) Set the iteration index j=1 and $\mathbf{k}^{(j-1)} = \mathbf{k}^{(0)}$ and $\mathbf{r}^{(j-1)} = \mathbf{r}^{(0)}$.

(iii) Generate or read from a file, $N_R \times p$ random numbers (R_{ni}) uniformly distributed in [-0.5, 0.5]

(iv) For n=1,2,…,N_R, generate the corresponding random trial parameter vectors from

$$\mathbf{k}_n = \mathbf{k}^{(j-1)} + \boldsymbol{R}_n \mathbf{r}^{(j-1)} \qquad (5.24)$$

where $\boldsymbol{R}_n = diag(R_{n1}, R_{n2}, \ldots, R_{np})$.

(v) Find the parameter vector among the N_R trial ones that minimizes the LS Objective function

$$S_{LS}(\mathbf{k}) = \sum_{i=1}^{N} [\hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i, \mathbf{k})]^T \mathbf{Q}_i [\hat{\mathbf{y}}_i - \mathbf{f}(\mathbf{x}_i, \mathbf{k})] \qquad (5.25)$$

(vi) Keep the best trial parameter vector, $\mathbf{k}^*$, up to now and the corresponding minimum value of the objective function, S*.

(vii) Set $\mathbf{k}^{(j)} = \mathbf{k}^*$ and compute the search region for the next iteration as

$$\mathbf{r}^{(j)} = \delta \times \mathbf{r}^{(j-1)} \qquad (5.26)$$

(viii) If $j < j_{max}$, increment j by 1 go to Step (iii); else STOP.

The algorithm can be more *efficient* if we choose a value for N_R which is a function of the number of unknown parameters e.g.

$$N_R = 50 + 10 \times p \qquad (5.27)$$

We may also use a slower contraction of the search region as p increases e.g.

$$\delta = (0.90)^{1/p} \qquad (5.28)$$

Since we have a minimization problem, significant computational savings can be realized by noting in the implementation of the LJ optimization procedure that for each trial parameter vector, we do not need to complete the summation in Equation 5.23 once the LS Objective function exceeds the smallest value found up to that point (S*), a new trial parameter vector can be selected.

Optimization by Direct Search and Systematic Reduction of the Size of Search Region

A direct search procedure utilizing pseudo random numbers over a region is presented to solve nonlinear programming problems. After each iteration the size of the region is reduced so that the optimum can be found as accurately as desired. The ease of programming, the speed of convergence, and the reliability of results make the procedure very attractive for solving nonlinear programming problems.

REIN LUUS and T. H. I. JAAKOLA

Department of Chemical Engineering and Applied Chemistry
University of Toronto
Toronto, Canada

SCOPE

The objective of this study was to develop a simple optimization procedure which would be applicable to solve nonlinear programming problems. We wanted to have a procedure which would be sufficiently simple to use so that anyone interested in system optimization could use it. Also, we wanted to develop a procedure which would be general enough so that it could be applied to any region of interest and to any type of functions.

The procedure presented in this paper meets our main objectives since it is extremely simple to use and is very effective in solving rather complex problems. In fact, our experience with several problems shows that the procedure could very well become as standard as linear programming. The method is straightforward and even simpler to use than linear programming since no complicated subroutine is required. A typical problem presented in this paper can be set up within a couple of hours and solved in a few seconds of computer time.

CONCLUSIONS AND SIGNIFICANCE

The optimization procedure based on direct search and systematic search region reduction is formulated and is found effective in solving various problems in the field of nonlinear programming. In this paper 6 sample problems are used to illustrate the procedure and to test its effectiveness. In every case the optimum could be reached in less than 10 seconds of computation time on an IBM 370/165 computer.

The procedure handles either inequality or equality constraints and the feasible region does not have to be convex. No approximations or auxiliary variables are required.

The most attractive features of the procedure are the ease of setting up the problem on the computer, speed in obtaining the optimum, and the reliability of the results.

In engineering design and operation one is frequently confronted with the problem of static nonlinear optimization. In design one wants to find the bottlenecks and redesign certain aspects of the system to maximize some function, for example return on investment. In operation one wants to know what flow rates, recycle, reactor temperature, catalyst activity, etc. to use to maximize some criterion of operation, such as the daily profit. Since the relationships between many variables could be nonlinear and the objective function may also be nonlinear, one confronts a nonlinear programming problem.

We may formulate the nonlinear programming problem as follows:

Maximize (or minimize)

$$P = f(x_1, x_2, \ldots, x_n) \tag{1}$$

subject to the constraints

$$g_i\ (x_1, x_2, \ldots, x_n) \leq 0 \quad i = 1, 2, \ldots, m \tag{2}$$

$$h_j\ (x_1, x_2, \ldots, x_n) \geq 0 \quad j = 1, 2, \ldots, r \tag{3}$$

$$q_k\ (x_1, x_2, \ldots, x_n) = 0 \quad k = 1, 2, \ldots, s \tag{4}$$

In order to have a meaningful problem, we must restrict the number of equality constraints s to be less than the number of variables n; there is, however, no limit to the number of inequalities. Thus $s < n$, and no restrictions are placed on m or r.

Various authors suggest sectional linearization of nonlinear functions, thus converting the nonlinear programming problem into a linear programming problem. For example, such an approach was used by Hovanessian and Stout (1963) and Hovanessian and Pipes (1969) in optimizing the fuel allocation in power plants, and by Di Bella and Stevens (1965) in optimizing a complex nonlinear reactor system. However, such approximations are not at all necessary if the following approach is considered.

DIRECT SEARCH PROCEDURE

In most of the problems we have encountered, each equality constraint can be readily removed by solving each equation in terms of a different variable seriatim. Thereby the number of variables to be determined is reduced by the number of equalities. For example, suppose we take the problem of determining x_1, x_2, and x_3 which will maximize the function P given by

$$P = f(x_1, x_2, x_3) \tag{5}$$

subject to the constraints

$$g_1(x_1, x_2, x_3) \leq 0 \tag{6}$$

$$g_2(x_1, x_2, x_3) \geq 0 \tag{7}$$

$$g_3(x_1, x_2, x_3) \geq 0 \tag{8}$$

$$g_4(x_1, x_2, x_3) = 0 \tag{9}$$

We assume that some operation can be performed on the equality constraint so that

$$\alpha g_4(x_1, x_2, x_3) = x_2 + h(x_1, x_3) = 0 \tag{10}$$

where α is some operator.

Thus

$$x_2 = -h(x_1, x_3) \tag{11}$$

and the problem becomes one of choosing x_1 and x_3 to maximize

$$P = f(x_1, -h(x_1, x_3), x_3) \tag{12}$$

subject to the constraints

$$g_1(x_1, -h(x_1, x_3), x_3) \leq 0 \tag{13}$$

$$g_2(x_1, -h(x_1, x_3), x_3) \geq 0 \tag{14}$$

$$g_3(x_1, -h(x_1, x_3), x_3) \geq 0 \tag{15}$$

If the set of equalities cannot be solved for the variables seriatim, then a numerical procedure could be used. We do not recommend the replacement of equalities by inequalities as was done by Bracken and McCormick (1968). This will introduce errors into the solution, as will be shown in our example. The examples also show that usually the equality constraints present no problem.

Thus without any loss of generality, we consider inequality constraints only and formulate the general problem as follows:

Maximize

$$P = f(x_1, x_2, \ldots, x_n) \tag{16}$$

subject to the constraints

$$g_i(x_1, x_2, \ldots, x_n) \leq 0 \quad i = 1, 2, \ldots, m \tag{17}$$

$$h_j(x_1, x_2, \ldots, x_n) \geq 0 \quad j = 1, 2, \ldots, r \tag{18}$$

Suppose we take 100 random values for the set $\{x_1, \ldots, x_n\}$ and test each set with respect to inequalities (17) and (18). Two possible outcomes could result: none of the sets satify all the inequality constraints, or some sets (let us say 37) satisfy all the inequality constraints. If sufficient care is taken to restrict the range over which the variables $x_1, \ldots, x_n$ are chosen it may be quite reasonable to assume that the second outcome results. Then substituting these 37 sets of values into Equation (5) will give 37 values of P out of which the largest may be picked out. Let us denote the largest of these P by $P^{*(1)}$ and label the corresponding values of the variables $(x_1^{*(1)}, x_2^{*(1)}, \ldots, x_n^{*(1)})$. Next let us choose another 100 sets of $\{x_1, \ldots, x_n\}$, but restrict that the variables be chosen at random around the point $(x_1^{*(1)}, x_2^{*(1)}, \ldots, x_n^{*(1)})$ so that the probability that x_i is greater than $x_i^{*(1)}$ is the same as the probability that x_i is less than $x_i^{*(1)}$. Now if we test each of these sets with respect to Equation (17) and Equation (18), we may find many which satisfy the constraints, let us say 51. We may calculate 51 values for P, pick the maximum and denote it by $P^{*(2)}$. $P^{*(2)}$ may be greater than $P^{*(1)}$, but not necessarily. If we were to continue in this fashion no assurance could be given that the optimum will be found in a reasonable number of steps.

We may, however, increase the probability that $P^{*(2)}$ be greater than $P^{*(1)}$ if we reduce the range over which the values of x_i may be chosen after the first step. Suppose that initially the range is $x_i \pm 0.5$ and we decrease the range by 5% after each step. Then after 200 steps the range is $(.95)^{200} = 3.5 \times 10^{-5}$; that is, $x_i \pm 1.8 \times 10^{-5}$. This large effect should not be surprising to one who thinks of investing \$1 at 5% cumulative interest for 200 years. For each of the variables the initial range can be chosen according to the understanding one has for the problem, but one has the assurance that at 5% reduction the range is reduced to 3.5×10^{-5} of its initial value. Since the values for the variables are always chosen around the best point determined in the previous iteration, there is a good likelihood of convergence to the optimum. We therefore propose the following algorithm.

DIRECT SEARCH USING RANDOM NUMBERS COMBINED WITH INTERVAL REDUCTION ALGORITHM

1. Take initial values for $x_1, x_2, \ldots, x_n$ and an initial range for each variable; denote these by $x_1^{*(0)}, x_2^{*(0)}, \ldots, x_n^{*(0)}$ and by $r_1^{(0)}, r_2^{(0)}, \ldots, r_n^{(0)}$. Set the iteration index j to 1.

2. Read in a sufficient number of random numbers (let us say 2000) between -0.5 and $+0.5$. Denote these by y_{ki}.

3. Take pn random numbers from 2 and assign these to $x_1, x_2, \ldots, x_n$ so that we may have p sets of values, each calculated by

$$x_i^{(j)} = x_i^{*(j-1)} + y_{ki}\, r_i^{(j-1)} \quad i = 1, \ldots, n \qquad k = 1, \ldots, p$$

4. Test Equation (17) and Equation (18) and calculate a value of P with each admissible set.

5. Find the set which maximizes P given by Equation (16). Write out the maximum value of P and the corresponding $x_i^{*(j)}$, $i = 1, \ldots, n$. Increment j by 1 to $j + 1$.

6. If the number of iterations has reached the maximum allowed, end the problem. For example, we may choose 200 to be the maximum number of iterations.

7. Reduce the range by an amount ϵ

$$r_i^{(j)} = (1 - \epsilon) r_i^{(j-1)}, \quad \epsilon > 0$$

For example we may choose $\epsilon = 0.05$.

8. Go to step 2 and continue.

To illustrate the procedure and to test its effectiveness we choose 6 examples. In every example at each iteration we choose 100 sets of pseudo-random numbers in an interval -0.5 to 0.5, and the reduction rate of interval, ϵ, equal to 0.05, and continue the search until 200 iterations have been reached. We followed the standard modulus method to generate 2000 pseudo-random numbers in the range 0 to 999 and had them punched out by the computer in 2I4 format on 100 cards. Then by reading these data cards in format 20F4.3 we have numbers in range 0.000 to 0.999. Then by subtracting -0.5 from each number we have the numbers in range -0.5 to 0.5. This part of calculation is included in each problem so that the computation times which are reported may be

somewhat high. The computation time is reported for IBM 370/165 digital computer and is the total time required to execute the problem including the writing out of results.

Example 1: Optimum Fuel Allocation in Power Plants

Consider the problem of minimizing the purchase of fuel oil when it is desired to produce an output of 50 MW from a two-boiler-turbine-generator combination which can use fuel oil or blast furnace gas (BFG) or any combination of these. The maximum BFG that is available is specified. This problem corresponds to case 3 of Hovenessian and Stout (1963) and to case 2 of Hovanessian and Pipes (1969).

From Figure 6 of Hovanessian and Stout (1963) by applying nonlinear curve-fitting we obtained the fuel requirements for the two generators explicitly in terms of MW produced. For generator 1 we have the fuel requirements for fuel oil in tons per hour

$$f_1 = 1.4609 + .15186x_1 + .00145x_1^2 \qquad (19)$$

and for BFG in fuel units per hour

$$f_2 = 1.5742 + .1631x_1 + .001358x_1^2 \qquad (20)$$

where x_1 is the output in MW of generator 1. The range of operation of the generator is

$$18 \leq x_1 \leq 30 \qquad (21)$$

Similarly for generator 2 the requirement for fuel oil is

$$g_1 = .8008 + .2031x_2 + .000916x_2^2 \qquad (22)$$

and for BFG,

$$g_2 = .7266 + .2256x_2 + .000778x_2^2 \qquad (23)$$

where x_2 is the output in MW of generator 2. The range of operation of the second generator is

$$14 \leq x_2 \leq 25 \qquad (24)$$

It is assumed that only 10.0 fuel units of BFG are available each hour and that each generator may use any combination of fuel oil or BFG. It is further assumed that when a combination of fuel oil and BFG is used the effects are additive. That is, if in generator 1 we use fuel oil and BFG in 1/3 ratio to produce x_1 MW, then the total fuel consumption consists of 0.25 f_1 tons of fuel oil per hour and 0.75 f_2 fuel units of BFG per hour.

The problem is to produce 50 MW from the two generators in such a way that the amount of fuel oil consumed is minimum. Mathematically the formulation of the problem is as follows:

Minimize

$$C = x_3f_1 + x_4g_1 \qquad (25)$$

where f_1 and g_2 are given by Equations (19) and (22) subject to the following constraints:

(a) Operating range for the generator 1

$$18 \leq x_1 \leq 30 \qquad (26)$$

(b) Requirement of 50 MW of power

$$x_2 = 50 - x_1 \qquad (27)$$

(c) Operating range of generator 2

$$14 \leq x_2 \leq 25 \qquad (28)$$

(d) Fraction of fuel oil used in generator 1

$$0 \leq x_3 \leq 1 \qquad (29)$$

(e) Fraction of fuel oil used in generator 2

$$0 \leq x_4 \leq 1 \qquad (30)$$

(f) Availability of blast furnace gas (BFG)

$$\text{BFG} = (1 - x_3)f_2 + (1 - x_4)g_2 \leq 10.0 \qquad (31)$$

where f_2 and g_2 are given by Equations (20) and (23).

By using Equation (27), x_2 is eliminated and the problem is to choose the variables x_1, x_3, and x_4 so that C as given by Equation (25) is minimized. There are 9 inequality constraints embodied in Equation (26), and Equation (28) to (31). Note that there is no lower restriction on Equation (31) since computationally BFG cannot become negative.

The initial values for x_1, x_3, and x_4 were taken as 20, 0.5, and 0.5 and the initial regions, as 20, 1.0 and 1.0 respectively.

It took 1 second of computation time to perform 200 iterations after which the value of C was 3.05 with $x_1 = 30.00$, $x_2 = 20.00$, $x_3 = 0.00$, and $x_4 = 0.58$. In fact, these values were reached already at the 55th iteration.

Thus we see that the minimum fuel oil consumption is 3.05 tons/hr which is very close to the answer, 3.17 tons/hr, obtained by Hovanessian et al (1963, 1969) by means of separable programming where the nonlinearities were approximated by linear sections and the problem was solved by the standard linear programming procedure.

The most attractive feature of the direct search method is the simplicity of programming. Only the very basics of computer programming are required and it is not unlikely that the program will work perfectly the first time. Another advantage of the method is that the starting condition is not crucial—any reasonable value will do.

We now turn to an example where the answer is known exactly so that the procedure could be further checked.

Example 2: A Mathematical Problem with Known Solution

Consider the test problem introduced by Rosen and Suzuki (1965) and used by Gould (1971) to test his steepest descent procedure. The problem is a mathematical one which does not have any physical interpretation but has a unique optimum. The problem is to minimize

$$C = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 \qquad (32)$$

subject to

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \leq 0 \qquad (33)$$

$$x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \leq 0 \qquad (34)$$

$$2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5 \leq 0 \qquad (35)$$

The minimum value of C is known to be

$$C = -44 \text{ at } x_1 = 0, x_2 = 1, x_3 = 2, x_4 = -1$$

Gould (1971) by using a gradient approach was able to obtain $x_1 = .0018$, $x_2 = 1.0073$, $x_3 = 2.0014$, $x_4 = -0.9967$ in 5 seconds of computation time on an IBM 360/75 computer.

By starting with $x_i = 0.0$ and $r_i = 0.5$ for $i = 1, 2, 3, 4$, we were able to obtain in 200 iterations requiring 2 seconds of computer time the following results: $C = -44.0000$, $x_1 = 0.0009$, $x_2 = 1.0004$, $x_3 = 1.9992$ and $x_4 = -1.0008$. We are thus able to reach much closer to the optimum than Gould.

To examine the effect of the choice of initial range, we also ran the program with $x_i = 0.0$ but $r_i = 5.0$ for $i = 1, 2, 3, 4$. After 200 iterations requiring again 2 sec-

onds of computer time, we obtained $C = -43.9999$, $x_1 = 0.0008$, $x_2 = 1.0014$, $x_3 = 1.9990$, and $x_4 = -1.0009$. The results are not as good as before but still considerably better than those of Gould (1971).

Example 3: Chemical Equilibrium in Complex Mixtures

We consider the chemical equilibrium problem formulated and solved by White et al. (1958) and by Dantzig et al. (1958) by means of piecewise linear approximations followed by linear programming. They also solved the problem by means of steepest descent. Here we solve the nonlinear programming problem by the proposed method.

It is known that at equilibrium the free energy is at a minimum. Therefore to determine the relative amounts of the species present in an equilibrium situation one can set up the problem of minimization of the free energy. By considering the N, H, O system of White et al. (1958) where there are 10 species present the problem is to minimize

$$C = \sum_{j=1}^{10} x_j \left[c_j + \ln\left(\frac{x_j}{x_T} \right) \right] \tag{36}$$

where x_T is the total moles and c_j are given by White et al. (1958). The constraints are

$$x_1 + 2x_2 + 2x_3 + x_6 + x_{10} = 2 \tag{37}$$

$$x_4 + 2x_5 + x_6 + x_7 = 1 \tag{38}$$

$$x_3 + x_7 + x_8 + 2x_9 + x_{10} = 1 \tag{39}$$

$$x_j \geq 0 \quad j = 1, 2, \ldots, 10$$

By using Equations (37) to (39) we may write

$$x_4 = 1 - 2x_5 - x_6 - x_7 \tag{40}$$

$$x_3 = 1 - x_7 - x_8 - 2x_9 - x_{10} \tag{41}$$

$$x_1 = 2 - 2x_2 - 2x_3 - x_6 - x_{10} \tag{42}$$

so that x_4, x_3, and x_1 can be eliminated and search be performed over the 7 remaining variables.

By starting with $x_i = 0.2$ and $r_i = 0.2$ for $i = 2, 5, 6, \ldots, 10$ (x_1, x_3, and x_4 are obtained from the equality constraints) we obtained after 200 iterations, which required 8 seconds of computer time, the minimum value for the free energy, $C = -47.7611$ and $x_1 = 0.04067$, $x_2 = 0.14774$, $x_3 = 0.78315$, $x_4 = 0.00141$, $x_5 = 0.48525$, $x_6 = 0.00069$, $x_7 = 0.02740$, $x_8 = 0.01795$, $x_9 = 0.03732$, and $x_{10} = 0.09686$. White et al. (1958), by using linear programming and steepest descent also obtained $C = -47.7611$. All the x_i we obtained agree identically to five decimal places with those of White et al. with the exception of three for which they obtained $x_2 = 0.14773$, $x_9 = 0.03731$ and $x_{10} = 0.09687$. Even here, the deviations are only in the last figure.

Example 4: Alkylation Process Optimization

Alkylation process is important in upgrading gasoline. We thus consider the problem of determining the best operating conditions in the alkylation process described by Payne (1958) and used for optimization by Sauer et al. (1964) who reduced the nonlinear problem to a series of linear programming problems. Since Bracken and McCormick (1968) formulated this problem as a direct nonlinear programming model but failed to obtain a correct solution to the problem, we shall use the same notation as Bracken and McCormick, but solve the problem by the proposed procedure.

Define

x_1 = olefin feed (barrels/day)
x_2 = isobutane recycle (barrels/day)
x_3 = acid addition rate (thousands of pounds/day)
x_4 = alkylate yield (barrels/day)
x_5 = isobutane makeup (barrels/day)
x_6 = acid strength (weight percent)
x_7 = motor octane number
x_8 = external isobutane-to-olefin ratio
x_9 = acid dilution factor
x_{10} = F-4 performance number.

When we use the data from Sauer et al. (1964), the problem may be formulated as follows:

Maximize

$$P = .063x_4x_7 - 5.04x_1 - .035x_2 - 10x_3 - 3.36x_5 \tag{36}$$

subject to the inequality constraints:

$$0.010 \leq x_1 \leq 2000 \tag{37}$$

$$0.010 \leq x_2 \leq 16000 \tag{38}$$

$$0.010 \leq x_3 \leq 120 \tag{39}$$

$$0.010 \leq x_4 \leq 5000 \tag{40}$$

$$0.010 \leq x_5 \leq 2000 \tag{41}$$

$$85 \leq x_6 \leq 93 \tag{42}$$

$$90 \leq x_7 \leq 95 \tag{43}$$

$$3 \leq x_8 \leq 12 \tag{44}$$

$$1.2 \leq x_9 \leq 4 \tag{45}$$

$$145 \leq x_{10} \leq 162 \tag{46}$$

and the equality constraints

$$x_4 = x_1(1.12 + .13167x_8 - .006667x_8^2) \tag{47}$$

$$x_5 = 1.22x_4 - x_1 \tag{48}$$

$$x_2 = x_1x_8 - x_5 \tag{49}$$

$$x_6 = 89 + [x_7 - (86.35 + 1.098x_8 - .038x_8^2)]/.325 \tag{50}$$

$$x_{10} = -133 + 3x_7 \tag{51}$$

$$x_9 = 35.82 - .222x_{10} \tag{52}$$

$$x_3 = .001x_4x_6x_9/(98 - x_6) \tag{53}$$

In this problem there are 7 equality constraints. Therefore, there are only 3 independent variables. We chose these to be x_1, x_7, and x_8. From these x_4, x_5, x_2, x_6, x_{10}, x_9, and x_3 can be calculated. The problem is therefore quite easy.

By taking $p = 100$ sets of random numbers per iteration and the initial range for x_1, x_7, and x_8 to be 500, 6 and 8 respectively, in double precision we required only 2 seconds of computer time to carry out 200 iterations. We arrive at the maximum P of 1162.0 with $x_1 = 1728.4$, $x_2 = 16000$, $x_3 = 98.4$, $x_4 = 3056.0$, $x_5 = 2000.0$, $x_6 = 90.6$, $x_7 = 94.2$, $x_8 = 10.41$, $x_9 = 2.61$, $x_{10} = 149.60$.

It will be noted that these values are very close to the results of Sauer et al. (1964). The slight deviation is present because the equalities used by Sauer et al. are only approximately satisfied. The reader might want to verify this point, but in any case our value of 1162.009 is very close to the value 1162.94 obtained by Sauer et al.

Example 5: Optimization of Drying Process for a Through-Circulation Dryer

The problem is to find the air flow rate and the bed thickness which will maximize the production rate. We

use the data and the corrected equations in the work of Chung (1972).

It is necessary to find the mass flow rate x_1 and the bed thickness x_2 such that the drying production rate is maximized; that is, maximize

$$P = 0.033x_1 \left[\frac{0.036}{1 - e^{-107.9x_2/x_1^{0.41}}} + 0.095 - \frac{9.27 \times 10^{-4}x_1^{0.41}}{x_2} \ln\left(\frac{1 - e^{-5.39x_2/x_1^{0.41}}}{1 - e^{-107.9x_2/x_1^{0.41}}} \right) \right]^{-1} \quad (54)$$

subject to the constraints

$$0.2 - 4.62 \times 10^{-10}x_1^{2.85}x_2 - 1.055 \times 10^{-4}x_1 \geq 0 \quad (55)$$

$$4/12 - 8.20 \times 10^{-7}x_1^{1.85}x_2 - 2.25/12 \geq 0 \quad (56)$$

$$2 - 109.6\frac{x_2}{x_1^{0.41}} \left[\frac{0.036}{1 - e^{-107.9x_2/x_1^{0.41}}} + 0.095 - \frac{9.27 \times 10^{-4}x_1^{0.41}}{x_2} \ln\left(\frac{1 - e^{-5.39x_2/x_1^{0.41}}}{1 - e^{-107.9x_2/x_1^{0.41}}} \right) \right] \geq 0 \quad (57)$$

With a starting point of $x_1 = 800$, $x_2 = 0.5$ and $r_1 = 200$, $r_2 = 0.25$ we arrive at the maximum P of 172.49 and $x_1 = 976.76$, $x_2 = 0.5235$. The computation time was 5 seconds. These values agree quite well with the corrected results of Chung (1973).

Example 6: Complex Reactor System Optimization

A very realistic chemical engineering problem has been introduced by Di Bella and Stevens (1965) and Adelman and Stevens (1972) who considered the optimization of a system consisting of a continuous stirred tank reactor, a heat exchanger, a decanter, and a distillation column. The design problem was earlier proposed by Williams and Otto (1960). To describe the system required 9 equations in 13 variables. The objective function to be maximized was per cent return on investment. Also an upper and lower constraint was placed on the reactor temperature.

Di Bella and Stevens (1965) formulated the optimization problem as a combination of steepest descent and linear programming but were unable to reach close to the optimum even after 600 passes through the routine. Adelman and Stevens (1972) used a constrained simplex method to arrive very close to the optimum. Luus and Jaakola (1973) showed that all the equations used for system description are not independent and there are 5 independent variables instead of 4. Therefore the system is greatly simplified.

The problem is to maximize the percent return given by

$$P = (84F_A - 201.96F_D - 336F_G + 1955.52F_P - 2.22F_R - 60V\rho)/(6V\rho) \quad (58)$$

subject to the constraints

$$F_G = 1.5k_3F_{RC}F_{RP}\beta \quad (59)$$

$$F_{RA} = \frac{F_{RC}}{10k_1F_{RB}}(10k_2F_{RB} + 5k_3F_{RP} + k_2F_{RB}F_{RC}/b) \quad (60)$$

$$F_P = F_{RP} - b \quad (61)$$

$$F_D = 0.2\alpha\beta k_2F_{RB}F_{RC}/b \quad (62)$$

$$F_A = k_1F_{RA}F_{RB}\beta + F_DF_{RA}/\alpha \quad (63)$$

Table 1. Computational Requirements to Reach within 0.1% of the Optimum

Example no.	Performance index	No. of iterations	No. of function evaluations
1	3.053	55	1952
2	−43.962	28	1759
3	−47.714	17	394
4	1161.50	24	919
5	172.46	7	336
6	121.45	20	1244

Table 2. Computational Requirements to Reach within 0.01% of the Optimum

Example no.	Performance index	No. of iterations	No. of function evaluations
1	3.0526	86	2989
2	−43.996	61	3311
3	−47.757	48	1011
4	1161.90	53	1863
5	172.47	32	1323
6	121.53	41	2853

$$F_B = F_G + F_P + F_D - F_A \quad (64)$$

$$F_R = \alpha + F_G + F_P \quad (65)$$

$$F_{RE} = 10(F_{RP} - F_P) \quad (66)$$

where

$$\beta = V\rho/F_R^2 \quad (67)$$

$$b = F_{RP} - 0.2(4\beta k_2F_{RB}F_{RC} - 2.5k_3\beta F_{RC}F_{RP}) \quad (68)$$

$$\alpha = F_{RA} + F_{RB} + F_{RC} + 11b \quad (69)$$

For the purposes of optimization we consider as independent variables T, F_{RC}, F_{RP}, β, and F_{RB}.

By starting with $T = 650$, $F_{RC} = 7500$, $F_{RP} = 16000$, $\beta = 7.693 \times 10^{-9}$, $F_{RB} = 100000$, and the initial regions to be respectively 100, 3000, 9000, 6.154×10^{-9}, 50000, we obtained an optimum return of 121.534 within 200 iterations and 4 seconds of computer time. The variables are not uniquely determined at this value of the performance index as is shown by Luus and Jaakola (1973).

EVALUATION OF THE PROPOSED METHOD IN TERMS OF NUMBER OF FUNCTION EVALUATIONS

From the examples that have been provided it is clear that the optimum was always obtained with no more than 20000 function evaluations. Since the number of function evaluations cannot be determined *a priori*, we inserted a counting procedure into the programs to count the number of times the performance criterion function was evaluated. The cumulative number of function evaluations was printed out after every iteration.

In Tables 1 and 2 it can be seen that to reach within 0.01% of the optimum requires roughly three times as many function evaluations as to get within 0.1% of the optimum. Also, in every case we reached within 0.01% of the optimum in fewer than 100 iterations and fewer than 3000 function evaluations. It must be emphasized that no auxiliary functions have been introduced so it is difficult to compare the number of function evaluations to the number required for methods where auxiliary func-

tions such as gradients are required.

In spite of the difficulty of making a direct comparison of the proposed method to widely different optimization procedures, we solved some of the problems presented by Keefer (1973) for which the only information related to computational effort is the number of function evaluations.

To solve the very trivial first problem of Keefer required only 261 function evaluations by the proposed method compared to 691 required by Keefer. For the second problem involving a simplified version of alkylation problem, Keefer obtained convergence to within 2% of the optimum in 3487 function evaluations. As reported in Table 1, we obtained convergence to within 0.1% of the optimum in 919 function evaluations. The comparison for this problem is not direct since the equations used by Keefer have been simplified and Keefer unfortunately does not give the equations which he used.

For the third problem of Keefer (1973) the xylene separation and performance index equations are given by Gottfried et al. (1970). We solved the problem with the proposed method and reached to within 0.4% of the optimum in 73 iterations and 1170 function evaluations. (For starting conditions we chose $x_1 = 1500$, $x_j = 0.5$, $j = 2, \ldots, 7$, and $r_i^{(0)} = 0.5x_i^{(0)}$.) The computation time for 200 iterations was 6 seconds, and for 73 iterations we needed 2 seconds. The number of function evaluations 3345 reported by Keefer (1973) is considerably higher and the computation time of 3 min on IBM 360/75 computer as reported by Gottfried et al. (1970) is about 18 times higher if we allow a factor of 5 for the difference in computation time of IBM 370/165 versus 360/75. This particular problem shows that equality constraints where the variables cannot be solved seriatim in terms of other variables present no difficulties.

Unfortunately inadequate information is given by Keefer (1973) on the remaining three problems. Especially problems 5 and 6 would have been interesting since Keefer reported a larger number of equality constraints than the number of variables! Also for these problems the constraints are violated to cast further doubt on the validity of Keefer's results.

Recently, overestimation of the difficulty of solving nonlinear algebraic equations has given rise to numerous optimization procedures which have rather limited use for realistic problems. Such anticipated difficulties do not usually arise as is shown in some detail by Luus (1973). We do not want to delve into this problem here except to mention that the recently proposed method of Mamen and Mayne (1972) falls into this category. Their pseudo Newton-Raphson method for function minimization requires considerable effort already with the two very simple examples that were chosen. These examples can be solved readily by our proposed method or preferably by the standard method of Lagrange multipliers followed by the procedure described by Luus (1973). Due to the difficulty of obtaining convergence with the method of Mamen and Mayne (1972) even for simple problems, we do not attempt any comparisons to that method.

For problems where more than one local optimum can occur, the proposed method is especially useful. As an illustration we may look at the problem of maximizing

$$P = x_1^2 + x_2^2 + x_3^2$$

subject to the constraints

$$x_1 + 2x_2 + 3x_3 = 1$$

$$x_1^2 + \frac{x_2^2}{2} + \frac{x_3^2}{4} = 4$$

With the proposed method the global maximum $P = 9.995$ is obtained immediately, whereas the use of standard methods may give the local maximum $P = 9.051$ or either of the two minima $P = 4.430$ or $P = 4.049$.

In terms of the number of function evaluations the proposed method is very efficient. But its greatest usefulness is in the ease of programming and the reliability of the results obtained. With this method the constraints are always satisfied and the presence of other local optima usually presents no problem.

SUMMARY

We have presented a new method to solve nonlinear programming problems. Our experience in using this direct search method with a wide variety of problems indicates that it is a very fast and straightforward procedure.

For all the examples in this paper we have arbitrarily used 100 sets of values over which maximization or minimization is done in each iteration. We continued the process until 200 iterations had been performed. Also, we have arbitrarily set the rate of region reduction after each iteration at 0.05. Although for each problem there are optimum values for these parameters, the solution procedure is so fast that the choice is not really important in solving the problems of the type of complexity presented in this paper. If the problem is much more complicated and if the optimization procedure is to be used more than once, then it is natural to investigate the most effective choice of these parameters. This best choice, however, will be problem dependent.

ACKNOWLEDGMENT

This work was performed with the assistance of a grant from the Canadian National Research Council, A-3515. Computations were carried out with the facilities of the University of Toronto Computer Centre.

NOTATION

BFG = blast furnace gas
C = function to be minimized
f = function
g = function
h = function
i = index
j = index; as superscript j is used to denote iteration number
k = index; number of equality constraints
m = number of $\{\leqq\}$ constraints
n = number of variables
p = number of sets of values taken at each iteration
P = function to be maximized
q = arbitrary function
r = number of $\{\geqq\}$ constraints
r_i = region of ith variables
x_i = variables
α = operator
ϵ = reduction factor used for region reduction (for all examples here $\epsilon = 0.05$)

$$r_i^{(j+1)} = (1 - \epsilon) r_i^{(j)}$$

LITERATURE CITED

Adelman, A., and W. F. Stevens, "Process Optimization by the 'Complex' Method," *AIChE J.*, **18**, 20 (1972).
Bracken, J., and G. P. McCormick, *Selected Applications of Nonlinear Programming*, Wiley, New York (1968).
Chung, S. F., "Mathematical Model and Optimization of

HUNGARIAN JOURNAL
OF INDUSTRIAL CHEMISTRY
VESZPRÉM
Vol. 26. pp. 281 - 286 (1998)

DETERMINATION OF THE REGION SIZES FOR LJ OPTIMIZATION PROCEDURE

R. LUUS

(Department of Chemical Engineering, University of Toronto, Toronto, Ontario, M5S 3E5, CANADA)

Received: February 9, 1998

This paper was presented at the 4th International Workshop on Chemical Engineering Mathematics, Bad Honnef, Germany, August 10-14, 1998.

To improve the rate of convergence of the direct search optimization procedure using random numbers and search region contraction as introduced by LUUS AND JAAKOLA [1], called LJ optimization procedure, a multi-pass method is suggested, where the region size for each variable at the beginning of each pass after the first one is determined from the extent of the variation of the variable during the pass. Computational experience with several typical chemical engineering systems shows that such a procedure enables the optimum to be determined more efficiently than arbitrarily choosing the region sizes through some preliminary runs, or by some intuitive approach. For illustration and evaluation of the approach, three examples are chosen. The first example involves an alkylation process consisting of ten variables and seven equality constraints. The second example involves model reduction, and the third example is a non-linear nonseparable optimization problem involving 300 variables. In each case, the method was found to be robust, and the optimum could be obtained considerably faster with this newly proposed method of region size determination.

Keywords: optimization, direct search optimization, global optimization

Problem Formulation

Let us consider the problem of determining the $(n \times 1)$ vector $\mathbf{z}$ to minimize the performance index that is a continuous function of n variables

$$I(\mathbf{z}) = f(z_1, z_2, ..., z_n) \quad (1)$$

subject to m inequality constraints that limit the range of the variables

$$g_k(z_1, z_2, ..., z_n) \le 0, k = 1, 2, ..., m \quad (2)$$

For ease of presentation here, we assume that if equality constraints are present, the equalities can be rearranged in such a way that the variables can be separated, actually simplifying the optimization problem. This is illustrated with the first example. For difficult equality constraints, where the variables can not be isolated, special procedures, such as the use of penalty functions [6,7] may be necessary.

To solve this problem by direct search optimization, we may use the procedure of LUUS and JAAKOLA [1]. However, to improve the efficiency of obtaining the optimum, we wish to use the LJ optimization procedure in a multi-pass method, where we choose the initial region sizes for the passes in a special way, based on the extent that each variable has changed during a pass.

There are inherent dangers in choosing the initial region size at the beginning of a pass in this way, since a particular variable may not change during a pass and then the region is put to zero for all subsequent passes, making it not possible for that variable to change in any subsequent iteration. Also, there is the possibility that some variable will change very rapidly during a pass, and then for the following pass the region pertaining to that variable will be excessively large, making it difficult for that variable to improve. The benefits, however, of using the extent of variation of a variable in a pass to determine the region size outweigh the disadvantages, and therefore it is important to examine the approach.

Algorithm

The procedure for optimization and choosing the region sizes in the LJ direct search optimization procedure may be summarized most conveniently in the following algorithm:

1. Take an initial estimate for $\mathbf{z}$ and select the initial region size for $\mathbf{z}$; denote these by $\mathbf{z}^{*(1)}$ and $\mathbf{r}_{in}^{(1)}$. Choose the number of iterations N to be used in each pass, the number of passes and the region contraction factor γ.
2. Set the pass index q to 1.
3. Set the iteration index j to 1, and put $\mathbf{r}^{(j)} = \mathbf{r}_{in}^{(q)}$.
4. Select R sets of values for the variables

$$\mathbf{z}^{(j)} = \mathbf{z}^{*(j)} + \theta\, \mathbf{r}^{(j)} \quad (3)$$

where θ is a diagonal matrix with random numbers between -1 and 1 as diagonal elements. Each set has a different set of random numbers.

5. Check each of the R sets for feasibility as determined by *Eq.(2)* and evaluate the performance index in *Eq.(1)* for each feasible set.
6. Compare the values of the performance index and save the set of variables $\mathbf{z}$ that yields the minimum value for the performance index I; denote this by $\mathbf{z}^{*(j)}$.
7. Reduce the size of the region by the contraction factor γ

$$\mathbf{r}^{(j+1)} = \gamma\, \mathbf{r}^{(j)} \quad (4)$$

8. Increment the iteration index j by 1 and go to step 4 and continue for the specified number of iterations.
9. Choose a new initial region size for the next pass to be equal to the change in the variables during the pass that yielded the best performance index, i.e.,

$$r_{in,i}^{(q+1)} = \left| z_i^{*(N)} - z_i^{*(1)} \right|, i = 1, 2, ..., n \quad (5)$$

10. Increment the pass number index q by 1 and go to step 3, continuing until the total number of passes has been reached, or some convergence criterion is satisfied.

Remark

To avoid region collapse for some of the variables in the first few passes, a simple region adjustment can be done at the beginning of some pass, by arbitrarily increasing each region size by a small amount. This will be illustrated by the examples. The essential feature, however, is to choose the region size for each variable to be equal to the range of active variation of the corresponding variable during a pass.

Numerical Examples

To illustrate and to test the proposed method of region determination, we choose three widely different examples. All calculations were done in double precision on Pentium/120 personal computer. It is clear that if it is required to maximize the performance index, then the procedure is the same, except we seek the maximum value of I. For efficiency of calculations we wish to take a reasonably small value for the number of random points R at each iteration. For the first two examples we take $\gamma = 0.95$ and for the third example which involves a very large number of variables we

Table 1 Number of passes required to reach the optimum P = 1162.027 by direct search optimization using 201 iterations per pass with $\gamma = 0.95$

Number of random points per iteration	Without region restoration	With region restoration after second pass
2	9	9
3	9	*
4	9	4
5	*	5
6	*	4
7	6	4
8	*	5
9	*	4
10	6	4
15	3	3
20	5	4
25	*	4

* denotes non-convergence to P = 1162.027 within 10 passes

choose $\gamma = 0.98$. In each case N = 201 iterations were used in each pass.

Example 1: Alkylation Process Optimization

The alkylation process optimization problem as first introduced by SAUER et al. [8], and used by LUUS and JAAKOLA [1], PALOSAARI et al. [9], and by LUUS and BRENEK [3] is a good test problem, since many constraint equations are non-linear. Also this example shows how equality constraints can be rearranged judiciously to reduce the dimensionality of the optimization problem.

The problem is to maximize the profit function

$$P = 0.063\, x_4 x_7 - 5.04\, x_1 - 0.035\, x_2 - 10\, x_3 - 3.36\, x_5 \quad (6)$$

subject to the inequality constraints

$$0.010 \le x_1 \le 2000 \quad (7)$$

$$0.010 \le x_2 \le 16000 \quad (8)$$

$$0.010 \le x_3 \le 120 \quad (9)$$

$$0.010 \le x_4 \le 5000 \quad (10)$$

$$0.010 \le x_5 \le 2000 \quad (11)$$

$$85 \le x_6 \le 93 \quad (12)$$

$$90 \le x_7 \le 95 \quad (13)$$

$$3 \le x_8 \le 12 \quad (14)$$

$$1.20 \le x_9 \le 4 \quad (15)$$

$$145 \le x_{10} \le 162 \quad (16)$$

Table 2 Effect of starting points and initial region sizes on convergence to P = 1162.027

x_1	x_7	x_8	Initial region size, $\mathbf{r}^{(0)\,T}$	Number of passes for convergence	Computation time for convergence, s
1500	92.7	11.5	[300, 10, 10]	4	0.17
1000	93.0	7.0	[800, 5, 10]	3	0.11
1700	93.0	10.0	[200, 5, 10]	3	0.06
1700	93.5	10.0	[500, 6, 8]	4	0.17
1600	93.5	9.0	[200, 6, 8]	8	0.28
1000	94.0	8.0	[700, 8, 8]	7	0.28
1200	92.7	6.5	[600, 10, 10]	4	0.16
1200	94.5	9.5	[600, 4, 9]	4	0.11
1200	94.5	10.5	[600, 4, 7]	5	0.22
1000	95.0	12.0	[600, 4, 9]	3	0.16
1000	95.0	12.0	[900, 5, 5]	4	0.22
1600	95.0	11.0	[200, 5, 5]	4	0.17

and the equality constraints

$$x_4 = x_1\,(1.12 + x_8\,(0.13167 - 0.006667\, x_8)) \quad (17)$$

$$x_5 = 1.22\, x_4 - x_1 \quad (18)$$

$$x_2 = x_1 x_8 - x_5 \quad (19)$$

$$x_6 = 89 + (x_7 - (86.35 + x_8(1.098 - 0.038\, x_8)))/0.325 \quad (20)$$

$$x_{10} = 3x_7 - 133 \quad (21)$$

$$x_9 = 35.82 - 0.222\, x_{10} \quad (22)$$

$$x_3 = 0.001 x_4 x_6 x_9 / (98 - x_6) \quad (23)$$

It is noted that the 7 equality constraints are arranged so that once x_1, x_7, and x_8 are available, then through the equality constraints, the rest of the variables can be calculated in sequence. Therefore, these equality constraints actually simplify the optimization problem by reducing the number of free variables to three, and we choose $\mathbf{z} = [\, x_1\, x_7\, x_8\,]^T$.

By using the systematic method of determining the region size as given in the algorithm, a very small number of randomly chosen points are necessary to obtain the optimum value of the performance index of 1162.027, as is shown in *Table 1*. By increasing the size of the region for each variable by 0.01 at the beginning of the third pass improves the success rate, as is shown in the third column of the same table. It is interesting to note that even with two randomly chosen points convergence to the optimum to 7 figures was obtained. However, to be on the safe side, for this problem it is recommended to use $R = 7$.

In fact, with R = 7, as is shown in *Table 2*, convergence to the optimum is obtained for all of the starting conditions used by LUUS and BRENEK [3], and at most 8 passes were required. The maximum computation time was 0.28 s on the Pentium/120 personal computer. It is obvious that this multi-pass method is substantially faster than the single pass where 100 randomly points are used at each iteration. In each case, the optimum value of P = 1162.027 was obtained

284

Table 3 Effect of starting points on convergence to $I = 2.49533 \cdot 10^{-3}$ with $R = 7$ random points per iteration

Starting Values for Parameters				Number of Passes to $I = 2.49533 \cdot 10^{-3}$	Computation Time on Pentium/120, s
α	β	τ	K		
4.821	2.523	0.9609	1.040	4	1.65
3.947	3.423	1.814	1.008	7	2.97
4.063	3.282	0.8392	1.008	6	2.52
2.0	0.5	1.0	1.0	2	0.82
6.0	2.0	1.0	1.0	4	1.59
5.0	3.0	2.0	1.0	9	3.79
3.0	4.0	0.5	1.25	7	2.86
2.0	5.0	0.75	0.8	10	4.22
5.75	3.25	1.25	1.25	6	2.42
2.0	3.0	1.0	1.0	6	2.41

with $x_1 = 1728.371$, $x_2 = 16000.000$, $x_3 = 98.136$, $x_4 = 3056.042$, $x_5 = 2000.000$, $x_6 = 90.619$, $x_7 = 94.190$, $x_8 = 10.414$, $x_9 = 2.616$, $x_{10} = 149.569$.

Table 4 Effect of starting points on convergence to $I = 2.49533 \cdot 10^{-3}$ with $R = 5$ random points per iteration, using three parameters

Starting Values for Parameters			Number of Passes to $I = 2.49533 \cdot 10^{-3}$	Computation Time on Pentium/120, s
α	τ	K		
4.821	0.9609	1.040	4	0.71
3.947	1.814	1.008	5	0.88
4.063	0.8392	1.008	5	0.83
2.0	1.0	1.0	2	0.33
6.0	1.0	1.0	2	0.27
5.0	2.0	1.0	3	0.49
3.0	0.5	1.25	3	0.55
2.0	0.75	0.8	4	0.72
5.75	1.25	1.25	3	0.49
2.0	1.0	1.0	2	0.39

$$I(\mathbf{z}) = \sum_{i=0}^{M} [x(t_i) - y(t_i)]^2 \qquad (29)$$

- The adaptation of the original LJ optimization procedure to parameter estimation problems for algebraic equation models will be discussed later in the course[23,24,25].

Application of Direct Search Optimization for Parameter Estimation

Rein Luus

University of Toronto, Department of Chemical Engineering
Toronto, ON M5S 3E5, Canada; e-mail: luus@ecf.utoronto.ca

Abstract. Estimation of parameters in a set of ordinary differential equations by direct search optimization procedure of Luus and Jaakola (1973)is considered. To enable the procedure to yield optimal values of parameters accurately, a multi-pass procedure is used, where the region size for each variable at the beginning of a pass is determined by the extent of the variation of the corresponding variable in the previous pass.

To test the procedure, two typical chemical engineering systems were chosen, where five parameters were to be determined from output data. The results show that accurate values for the parameters can be obtained in reasonable amount of computational effort. The most attractive feature is that the initial estimates play a very minor role, and no auxiliary variables need to be evaluated.

Keil · Mackens · Voß · Werther (Eds.)

Scientific Computing
in Chemical Engineering II

Simulation, Image Processing, Optimization,
and Control

Editors:

Prof. Dr. Frerich Keil
Technical University of Hamburg-Harburg
Chair of Chemical Reaction Engineering
Eißendorfer Straße 38
D-21071 Hamburg / Germany

Prof. Dr. Wolfgang Mackens
Technical University of Hamburg-Harburg
Section of Mathematics
Schwarzenbergstraße 95
D-21073 Hamburg / Germany

Prof. Dr. Heinrich Voß
Technical University of Hamburg-Harburg
Section of Mathematics
Schwarzenbergstraße 95
D-21073 Hamburg / Germany

Prof. Dr.-Ing. Joachim Werther
Technical University of Hamburg-Harburg
Chemical Engineering I
Denickestraße 15
D-21071 Hamburg / Germany

ISBN 3-540-65851-3 Springer-Verlag Berlin Heidelberg New York

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme
Scientific computing in chemical engineering / F. Keil ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1999
2. Simulation, image processing, optimization, and control : with tables. - 1999
ISBN 3-540-65851-3

---

[23] Englezos, P., N. E. Kalogerakis, *Applied Parameter Estimation for Chemical Engineers*, Marcel-Dekker, New York, 2001

[24] Luus, R., "Application of Direct Search Optimization for Parameter Estimation", in *Scientific Computing in Chemical Engineering II*, Keil, F., W. Mackens, H. Vob, J.Werther (Eds), Springer-Verlag, Berlin Heidelberg, 1999.

[25] Wang, B.C. and R. Luus, "Increasing the Size of the Region of Convergence for Parameter Estimation Through the Use of Shorter Data-Length", *Int. J. Control,* 31, 947-957 (1980).