

Global Optimization by Differential Evolution and Particle Swarm Methods Evaluation on Some Benchmark Functions

SK Mishra
Dept. of Economics
North-Eastern Hill University
Shillong, Meghalaya (India)

I. A Brief History of Optimization Research: The history of optimization of real-valued non-linear functions (including linear ones), unconstrained or constrained, goes back to Gottfried Leibniz, Isaac Newton, Leonhard Euler and Joseph Lagrange. However, those mathematicians often assumed differentiability of the optimand as well as constraint functions. Moreover, they often dealt with the equality constraints. Richard Valentine (1937) and William Karush (1939), however, were perhaps the first mathematicians to study optimization of nonlinear functions under inequality constraints. Leonid Kantorovich and George Dantzig are well known for developing and popularizing linear programming, which ushered a new era of ‘operations research’, a branch of mathematical science that specializes in optimization. The development of linear programming soon prompted the study of the optimization problem of nonlinear functions (often under linear or nonlinear constraints). The joint work of Harold Kuhn and Albert Tucker (1951) – that was backed up by the work of Karush – is a landmark in the history of optimization of nonlinear functions.

Initially, optimization of nonlinear functions was methodologically based on the Leibniz-Newton principles and therefore could not easily escape local optima. Hence, its development to deal with nonconvex (multimodal) functions stagnated until the mid 1950’s. Stanislaw Ulam, John von Neumann and Nicolas Metropolis had in the late 1940’s proposed the Monte Carlo method of simulation and it was gradually realized that the simulation approach could provide an alternative methodology to mathematical investigations in optimization. George Box (1957) was perhaps the first mathematician who exploited the idea and developed his evolutionary method of nonlinear optimization. Almost a decade later, MJ Box (1965) developed his complex method, which strews random numbers over the entire domain of the decision variables and therefore has a great potentiality to escape local optima and locate the global optimum of a nonlinear function. The simplex method of John Nelder and Roger Mead (1964) also incorporated the ability to learn from its earlier search experience and adapt itself to the topography of the surface of the optimand function.

II. Global Optimization: The 1970’s evidenced a great fillip in simulation-based optimization research due to the invention of the ‘genetic algorithm’ by John Holland (1975). A genetic algorithm is a class of population-based adaptive stochastic optimization procedures, characterizing the presence of randomness in the optimization process. The randomness may be present as either noise in measurements or Monte Carlo randomness in the search procedure, or both. The basic idea behind the genetic algorithm is to mimic a simple picture of the Darwinian natural selection in order to find a good algorithm and involves the operations such as ‘mutation’, ‘selection’ and ‘evaluation of

fitness' repeatedly. The genetic algorithms may claim to have ushered the new era of global optimization.

A little later, in 1978, Aimo Törn introduced his "Clustering Algorithm" of global optimization. The method improves upon the earlier local search algorithms that needed 'multiple start' from several points distributed over the whole optimization region. Multi-start is certainly one of the earliest global procedures used. It has even been used in local optimization for increasing the confidence in the obtained solution. However, one drawback of Multi-start is that when many starting points are used, the same minimum will eventually be determined several times. In order to improve the efficiency of Multi-start this should be avoided. The clustering method of Törn avoids this repeated determination of local minima. This is realized in three steps, which may be iteratively used. The three steps are: (i) sample points in the region of interest, (ii) transform the sample to obtain points grouped around the local minima, and (iii) use a clustering technique to recognize these groups (i.e. neighbourhoods of the local minima). If the procedure employing these steps is successful, then, starting a single local optimization from each cluster would determine the local minima and, thus, also the global minimum. The advantage in using this approach is that the work spared by computing each minimum just once can be spent on computations in (i) and (ii), which will increase the probability that the global minimum will be found.

While the clustering algorithm was optimizing on the multi-start requirements of local search algorithms and the genetic algorithm simulated the Darwinian struggle for the survival of the fittest, the simulated annealing method (Kirkpatrick et al., 1983; Cerny, 1985) proposed to mimic the annealing process in metallurgy. In an annealing process a metal in the molten state (at a very high temperature) is slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered – the liquid freezes or the metal re-crystallizes – attaining the ground state at $T=0$. This process is simulated through the Monte Carlo experiment (Metropolis et al. 1953). If the initial temperature of the melt is too low or cooling is done unduly fast the metal may become 'quenched' due to being trapped in a local minimum energy state (meta-stable state) forming defects or freezing out. The simulated annealing method of optimization makes very few assumptions regarding the function to be optimized, and therefore, it is quite robust with respect to irregular surfaces. In this method, the mathematical system describing the problem mimics the thermodynamic system. The current solution to the problem mimics the current state of the thermodynamic system, the objective function mimics the energy equation for the thermodynamic system, and the global minimum mimics the ground state. However, nothing in the numerical optimization problem directly mimics the temperature, T , in the thermodynamic system underlying the metallurgical process of annealing. Therefore, a complex abstraction mimics it. An arbitrary choice of initial value of a variable called 'temperature', how many iterations are performed at each 'temperature', the step length at which the decision variables are adjusted, and the rate of fall of 'temperature' at each step as 'cooling' proceeds, together make an 'annealing schedule'. This schedule mimics the cooling process. At a high 'temperature' the step lengths at which the decision variables are adjusted are larger than those at a lower

‘temperature’. Whether the system is trapped into local minima (quenching takes place) or it attains the global minimum (faultless crystallization) is dependent on the said annealing schedule. A wrong choice of the initial ‘temperature’, or the rate of fall in the ‘temperature’ leads to quenching or entrapment of the solution in the local minima. The method does not provide any clear guideline as to the choice of the ‘annealing schedule’ and often requires judgment or trial and error. If the schedule is properly chosen, the process attains the global minimum. It is said that using this method is an art and requires a lot of experience and judgment.

A little later, Fred Glover (1986) introduced his ‘Tabu Search’ method. This method economizes on repeated visits to the already visited points and in some sense is close to the clustering algorithms. Glover attributes its origin to about 1977. The basic concept of Tabu Search as described by Glover is "a meta-heuristic superimposed on another heuristic. The overall approach is to avoid entrainment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited (hence "tabu"). The Tabu method was partly motivated by the observation that human behavior appears to operate with a random element that leads to inconsistent behavior given similar circumstances. As Glover points out, the resulting tendency to deviate from a charted course, might be regretted as a source of error but can also prove to be source of gain. The Tabu method operates in this way with the exception that new courses are not chosen randomly. Instead the Tabu search proceeds according to the supposition that there is no point in accepting a new (poor) solution unless it is to avoid a path already investigated. This insures new regions of a problems solution space will be investigated in with the goal of avoiding local minima and ultimately finding the desired solution.

The pace of research in global optimization (GO) by stochastic process accelerated considerably in the 1990’s. Marco Dorigo in his Ph.D. thesis (1992) introduced his “Ant Colony” method of global optimization. It studies artificial systems that take inspiration from the behaviour of real ant colonies. Ants use pheromones that guide other fellow ants to identify the path that leads to a success. The chemical properties of pheromones and the ability of ants to gather information and use them are simulated in the Ant Colony method to reach at the global optimum. This method is well suited to combinatorial (discrete) optimization problems.

A couple of years later, James Kennedy and Russell Eberhart (1995) introduced their “Particle Swarm” method of global optimization. In the animal world we observe that a swarm of birds or insects or a school of fish searches for food, protection, etc. in a very typical manner. If one of the members of the swarm sees a desirable path to go, the rest of the swarm will follow quickly. The Particle Swarm method mimics this behaviour. Every individual of the swarm is considered as a particle in a multidimensional space that has a position and a velocity. These particles fly through hyperspace and remember the best position that they have seen. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. There are two main ways this communication is done: (i) “swarm best” that is known to all (ii) “local bests” are known in neighborhoods of particles. Updating of the position

and velocity are done at each iteration such that the solution often converges to the global optimum of the function. Interestingly, this method has a very sound and well-documented philosophical literature behind it (the British empiricist philosophy, the American pragmatism and others like those of Friedrich Hayek, Herbert Simon, etc.).

The method of Differential Evolution (DE) grew out of Kenneth Price's attempts to solve the Chebychev Polynomial fitting Problem that had been posed to him by Rainer Storn. A breakthrough happened (1996), when Price came up with the idea of using vector differences for perturbing the vector population. The crucial idea behind DE is a scheme for generating trial parameter vectors. Initially, a population of points (p in d -dimensional space) is generated and evaluated (i.e. $f(p)$ is obtained) for their fitness. Then for each point (p_i) three different points (p_a , p_b and p_c) are randomly chosen from the population. A new point (p_z) is constructed from those three points by adding the weighted difference between two points ($w(p_b-p_c)$) to the third point (p_a). Then this new point (p_z) is subjected to a crossover with the current point (p_i) with a probability of crossover (c_r), yielding a candidate point, say p_u . This point, p_u , is evaluated and if found better than p_i then it replaces p_i else p_i remains. Thus we obtain a new vector in which all points are either better than or as good as the current points. This new vector is used for the next iteration. This process makes the differential evaluation scheme completely self-organizing.

III. The Characteristic Features of Population-Based GO Methods: All population-based methods of global optimization partake of the probabilistic nature inherent to them. As a result, one cannot obtain certainty in their results, unless they are permitted to go in for indefinitely large search attempts. Larger is the number of attempts, greater is the probability that they would find out the global optimum, but even then it would not reach at the certainty. Secondly, all of them adapt themselves to the surface on which they find the global optimum. The scheme of adaptation is largely based on some guesswork since nobody knows as to the true nature of the problem (environment or surface) and the most suitable scheme of adaptation to fit the given environment. Surfaces may be varied and different for different functions. A particular type of surface may be suited to a particular method while a search in another type of surface may be a difficult proposition for it. Further, each of these methods operates with a number of parameters that may be changed at choice to make it more effective. This choice is often problem oriented and for obvious reasons. A particular choice may be extremely effective in a few cases, but it might be ineffective (or counterproductive) in certain other cases. Additionally, there is a relation of trade-off among those parameters. These features make all these methods a subject of trial and error exercises.

IV. The Objectives: Our objective in this paper is to compare the performance of the Differential Evolution (DE) and the Repulsive Particle Swarm (RPS) methods of global optimization. To this end, 70 test functions have been chosen (for their algebraic specifications, see Mishra, 2006-c, d and e). Among these test functions, some are new while others are well known in the literature; some are unimodal, the others multi-modal; some are small in dimension (no. of variables, x in $f(x)$), while the others are large in

dimension; some are algebraic polynomial equations, while the other are transcendental, etc. In all, they together represent a wide variety of surfaces.

V. Some Details of the Particle Swarm Methods Used Here: In this exercise we have used (modified) Repulsive Particle Swarm method. The Repulsive Particle Swarm method of optimization is a variant of the classical Particle Swarm method (see Wikipedia, <http://en.wikipedia.org/wiki/RPSO>). It is particularly effective in finding out the global optimum in very complex search spaces (although it may be slower on certain types of optimization problems).

In the traditional RPS the future velocity, v_{i+1} of a particle at position with a recent velocity, v_i , and the position of the particle are calculated by:

$$v_{i+1} = \omega v_i + \alpha r_1 (\hat{x}_i - x_i) + \omega \beta r_2 (\hat{x}_{hi} - x_i) + \omega \gamma r_3 z$$
$$x_{i+1} = x_i + v_{i+1}$$

where,

- x is the position and v is the velocity of the individual particle. The subscripts i and $i+1$ stand for the recent and the next (future) iterations, respectively.
- r_1, r_2, r_3 are random numbers, $\in [0,1]$
- ω is inertia weight, $\in [0.01, 0.7]$
- \hat{x} is the best position of a particle
- x_h is best position of a randomly chosen other particle from within the swarm
- z is a random velocity vector
- α, β, γ are constants

Occasionally, when the process is caught in a local optimum, some *chaotic* perturbation in position as well as velocity of some particle(s) may be needed.

The traditional RPS gives little scope of local search to the particles. They are guided by their past experience and the communication received from the others in the swarm. We have modified the traditional RPS method by endowing stronger (wider) local search ability to each particle. Each particle flies in its local surrounding and searches for a better solution. The domain of its search is controlled by a new parameter (*nstep*). This local search has no preference to gradients in any direction and resembles closely to tunneling. This added exploration capability of the particles brings the RPS method closer to what we observe in real life. However, in some cases moderately wide search (*nstep*=9, say; see program) works better.

Each particle learns from its 'chosen' inmates in the swarm. At the one extreme is to learn from the best performer in the entire swarm. This is how the particles in the original PS method learn. However, such learning is not natural. How can we expect the individuals to know as to the best performer and interact with all others in the swarm? We believe in limited interaction and limited

knowledge that any individual can possess and acquire. So, our particles do not know the ‘best’ in the swarm. Nevertheless, they interact with some chosen inmates that belong to the swarm. Now, the issue is: how does the particle choose its inmates? One of the possibilities is that it chooses the inmates closer (at lesser distance) to it. But, since our particle explores the locality by itself, it is likely that it would not benefit much from the inmates closer to it. Other relevant topologies are : (the celebrated) *ring topology*, ring topology hybridized with random topology, star topology, von Neumann topology, etc.

Now, let us visualize the possibilities of choosing (a predetermined number of) inmates randomly from among the members of the swarm. This is much closer to reality in the human world. When we are exposed to the mass media, we experience this. Alternatively, we may visualize our particles visiting a public place (e.g. railway platform, church, etc) where it (he) meets people coming from different places. Here, geographical distance of an individual from the others is not important. Important is how the experiences of others are communicated to us. There are large many sources of such information, each one being selective in what it broadcasts and each of us selective in what we attend to and, therefore, receive. This selectiveness at both ends transcends the geographical boundaries and each one of us is practically exposed to randomized information. Of course, two individuals may have a few common sources of information. We have used these arguments in the scheme of dissemination of others’ experiences to each individual particle. Presently, we have assumed that each particle chooses a pre-assigned number of inmates (randomly) from among the members of the swarm. However, this number may be randomized to lie between two pre-assigned limits.

VI. Some Details of the Differential Evolution Methods Used Here: The differential Evolution method consists of three basic steps: (i) generation of (large enough) population with N individuals [$x = (x_1, x_2, \dots, x_m)$] in the m -dimensional space, randomly distributed over the entire domain of the function in question and evaluation of the individuals of the so generated by finding $f(x)$; (ii) replacement of this current population by a better fit new population, and (iii) repetition of this replacement until satisfactory results are obtained or certain criteria of termination are met.

The crux of the problem lays in replacement of the current population by a new population that is better fit. Here the meaning of ‘better’ is in the Pareto improvement sense. A set S_a is better than another set S_b *iff* : (i) **no** $x_i \in S_a$ is inferior to the corresponding member of $x_i \in S_b$; **and** (ii) **at least one** member $x_k \in S_a$ is better than the corresponding member $x_k \in S_b$. Thus, every new population is an improvement over the earlier one. To accomplish this, the DE method generates a candidate individual to replace each current individual in the population. The candidate individual is obtained by a crossover of the current individual and three other randomly selected individuals from the current population. The crossover itself is probabilistic in nature. Further, if the candidate individual is better fit than the current individual, it takes the place of the current individual, else the current individual stays and passes into the next iteration. The

crossover scheme (called exponential crossover, as suggested by Kenneth Price in his personal letter to the author) is given below. This is coded for $ncross \Rightarrow 1$ in the program.

The mutant vector is $\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F^*(\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g})$ and the target vector is $\mathbf{x}_{i,g}$ and the trial vector is $\mathbf{u}_{i,g}$. The indices $r1$, $r2$ and $r3$ are randomly but different from each other. $U_j(0,1)$ is a uniformly distributed random number between 0 and 1 that is chosen anew for each parameter as needed.

Step 1: Randomly pick a parameter index $j = jrand$.

Step 2: The trial vector inherits the j th parameter (initially = $jrand$) from the mutant vector, i.e., $u_{j,i,g} = v_{j,i,g}$.

Step 3: Increment j ; if $j = D$ then reset $j = 0$.

Step 4: If $j = jrand$ end crossover; else goto Step 5.

Step 5: If $Cr \leq U_j(0,1)$, then goto Step 2; else goto Step 6

Step 6: The trial vector inherits the j th parameter from the target vector, i.e., $u_{j,i,g} = x_{j,i,g}$.

Step 7: Increment j ; if $j = D$ then reset $j = 0$.

Step 8: If $j = jrand$ end crossover; else goto Step 6.

There could be other schemes (as many as 10 in number) of crossover, including no crossover (only probabilistic replacement, $NCROSS \leq 0$ that works better in case of a few functions).

VII. Specification of Adjustable Parameters: As it has been mentioned before, these methods need certain parameters to be defined before they work. In case of the DE, we have fixed: max number of iterations allowed, $Iter = 10000$, population size, $N = 10$ times of the dimension of the function or 100 whichever maximum; scheme of crossover, $ncross = 1$ (defined in the program); crossover probability, $pcros = 0.9$; scale factor, $fact = 0.5$, random number seed, $iu = 1111$ and all random numbers are uniformly distributed between -1000 and 1000; accuracy needed, $eps = 1.0e-08$. If x in $f(x)$ violates the boundary then it is forcibly brought within the specified limits through replacing it by a random number lying in the given limits of the function concerned. In case of the RPS: population size, $N=100$; neighbour population, $NN=50$; steps for local search, $NSTEP=11$; Max no. of iterations permitted, $ITRN=10000$; chaotic perturbation allowed, $NSIGMA=1$; selection of neighbour : random, $ITOP=3$; $A1=A2=0.5$; $A3=5.e-04$; $W=.5$; $SIGMA=1.e-03$; $EPSI=1.d-04$. Meanings of these parameters are explained in the programs (appended).

VIII. Results and Discussion: Among 70 functions, a few have been run for small as well as large dimensions. In total, 73 optimization exercises have been done. DE has succeeded in 65 cases while RPS has succeeded in 55 cases. In almost all cases, DE has converged faster and given much more accurate results. The convergence of RPS is much slower even for lesser stringency on accuracy. The summary of results is presented in

table-1. Some test functions have been hard for both the methods. These are: Zero-Sum (30D), Perm#1, Perm#2, Power and Bukin-6 functions.

From what we see in table 1, one cannot yet reach at the definite conclusion that the DE performs better (or worse) than the RPS. None could assure a supremacy over the other. Each one faltered in some case; each one succeeded in some others. However, *DE is unquestionably faster, more accurate and more frequently successful than the RPS*. It may be argued, nevertheless, that alternative choice of adjustable parameters could have yielded better results in either method's case. The protagonists of either method could suggest that. Our purpose is not to join with the one or the other. We simply want to highlight that in certain cases they both succeed, in certain other case they both fail and each one has some selective preference over some particular type of surfaces. What is needed is to identify such structures and surfaces that suit a particular method most.

It is needed that we find out some criteria to classify the problems that suit (or does not suit) a particular method. This classification will highlight the comparative advantages of using a particular method for dealing with a particular class of problems.

Table 1: A Summary of Success or Failure of DE and RPS Methods in Optimization of Test Functions									
Sl No	Test Function	Dim (M)	Success		Sl No	Test Function	Dim (M)	Success	
			DE	RPS				DE	RPS
1	New Fn #1	2	yes	yes	38	Linear Programming	3	yes	yes
2	New Fn #2	2	yes	yes	39	Hougen Fn	5	no	no
3	New Fn #3	2	yes	yes	40	Giunta Fn	2	yes	yes
4	New Fn #4	2	yes	yes	41	Egg-Holder Fn	2	yes	yes
5	New Fn #8	2	yes	yes	42	Trid Fn	30	yes	yes
6	Quintic Fn	4	yes	no	43	Greiwank Fn	30	yes	yes
7	Quintic Fn	30	yes	no	44	Weierstrass Fn	20	yes	no
8	Needle-eye Fn	10	yes	no	45	Levy#3 Fn	2	yes	yes
9	Needle-eye Fn	30	yes	no	46	Levy#5 Fn	2	yes	yes
10	Zero-sum Fn	10	yes	no	47	Levy#8 Fn	3	yes	yes
11	Zero-sum Fn	30	no	no	48	Rastrigin Fn	30	yes	no
12	Corana Fn	4	yes	yes	49	Ackley Fn	30	yes	yes
13	Mod RCos Fn	2	yes	yes	50	Michalewicz Fn	20	yes	no
14	Freud-Roth Fn	2	yes	yes	51	Schwefel Fn	30	yes	no
15	Anns XOR Fn	9	yes	no	52	Shubert Fn	2	yes	yes
16	Perm #1 Fn	4	no	no	53	Dixon-Price Fn	10	yes	no
17	Perm #2 Fn	5	no	no	54	Shekel Fn	4	yes	yes
18	Power-Sum Fn	4	no	no	55	Paviani Fn	10	yes	yes
19	Goldstein-Price Fn	2	yes	yes	56	Branin#1 Fn	2	yes	yes
20	Bukin-6 Fn	2	no	no	57	Branin#2 Fn	2	yes	yes
21	DCS Fn	4	yes	yes	58	Bohachevsky#1 Fn	2	yes	yes
22	New Factorial Fn	4	yes	yes	59	Bohachevsky#2 Fn	2	yes	yes
23	New Decanomial Fn	2	yes	yes	60	Bohachevsky#3 Fn	2	yes	yes
24	Judge Fn	2	yes	yes	61	Easom Fn	2	yes	yes
25	New Dodecal Fn	3	yes	yes	62	Rosenbrock Fn	10	yes	yes
26	New sum=prod Fn	2	yes	yes	63	Crosslegged Table Fn	2	yes	no
27	New AM=GM Fn	10	yes	yes	64	Cross Fn	2	yes	yes
28	Yao-Liu#2 Fn	30	yes	yes	65	Cross-in-Tray Fn	2	yes	yes
29	Yao-Liu#3 Fn	30	yes	yes	66	Crowned Cross Fn	2	yes	yes
30	Yao-Liu#4 Fn	30	yes	yes	67	TT-Holder Fn	2	yes	yes

31	Yao-Liu#6 Fn	30	yes	yes	68	Holder Table Fn	2	yes	yes
32	Yao-Liu#7 Fn	30	no	yes	69	Carrom Table Fn	2	yes	yes
33	Yao-Liu#12 Fn	30	yes	yes	70	Pen-Holder Fn	2	yes	yes
34	Yao-Liu#13 Fn	30	yes	yes	71	Bird Fn	2	yes	yes
35	Yao-Liu#14 Fn	2	yes	yes	72	Chichinadze Fn	2	yes	yes
36	Yao-Liu#15 Fn	4	no	yes	73	McCormick Fn	2	yes	yes
37	Linear Programming	2	yes	yes	Failed in no. of cases (in 73 trials)			8	18
Note: For differently set adjustable parameters, either method may perform better or worse than reported here.									

Bibliography

- Bauer, J.M.: "Harnessing the Swarm: Communication Policy in an Era of Ubiquitous Networks and Disruptive Technologies", *Communications and Strategies*, 45, 2002.
- Box, M.J.: "A new method of constrained optimization and a comparison with other methods". *Comp. J.* 8, pp. 42-52, 1965.
- Bukin, A. D.: *New Minimization Strategy For Non-Smooth Functions*, Budker Institute of Nuclear Physics preprint BUDKER-INP-1997-79, Novosibirsk 1997.
- Cerny, V.: "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *J. Opt. Theory Appl.*, 45, 1, 41-51, 1985.
- Eberhart R.C. and Kennedy J.: "A New Optimizer using Particle Swarm Theory", *Proceedings Sixth Symposium on Micro Machine and Human Science*, pp. 39-43. IEEE Service Center, Piscataway, NJ, 1995.
- Fleischer, M.: "Foundations of Swarm Intelligence: From Principles to Practice", Swarming Network Enabled C4ISR, arXiv:nlin.AO/0502003 v1 2 Feb 2005.
- G.E.P. Box, "Evolutionary operation: A method for increasing industrial productivity", *Applied Statistics*, 6 , pp. 81-101, 1957.
- Glover F., " Future paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, 5:533-549, 1986.
- Hayek, F.A.: *The Road to Serfdom*, Univ. of Chicago Press, Chicago, 1944.
- Holland, J.: *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, 1975.
- Karush, W. *Minima of Functions of Several Variables with Inequalities as Side Constraints*. M.Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois, 1939.
- Kirkpatrick, S., Gelatt, C.D. Jr., and Vecchi, M.P.: "Optimization by Simulated Annealing", *Science*, 220, 4598, 671-680, 1983.
- Kuhn, H.W. and Tucker, A.W.: "Nonlinear Programming", in Neymann, J. (ed) *Proceedings of Second Berkeley Symposium on Mathematical Statistics and Probability*, Univ. of California Press, Berkeley, Calif. pp. 481-492, 1951.
- Metropolis, N. [The Beginning of the Monte Carlo Method](#). *Los Alamos Science*, No. 15, Special Issue, pp. 125-130, 1987.
- Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E.: "Equation of State Calculations by Fast Computing Machines", *J. Chem. Phys.*, 21, 6, 1087-1092, 1953.
- Mishra, S.K.: "Some Experiments on Fitting of Gielis Curves by Simulated Annealing and Particle Swarm Methods of Global Optimization", *Social Science Research Network (SSRN)*: <http://ssrn.com/abstract=913667>, Working Papers Series, 2006 (a).
- Mishra, S.K.: "Least Squares Fitting of Chacón-Gielis Curves by the Particle Swarm Method of Optimization", *Social Science Research Network (SSRN)*, Working Papers Series, <http://ssrn.com/abstract=917762> , 2006 (b).
- Mishra, S.K.: "Performance of Repulsive Particle Swarm Method in Global Optimization of Some Important Test Functions: A Fortran Program" , *Social Science Research Network (SSRN)*, Working Papers Series, <http://ssrn.com/abstract=924339> , 2006 (c).
- Mishra, S.K.: "Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method", *Social Science Research Network (SSRN)* Working Papers Series, <http://ssrn.com/abstract=927134>, 2006 (d).
- Mishra, S.K.: "Repulsive Particle Swarm Method on Some Difficult Test Problems of Global Optimization" ,SSRN: <http://ssrn.com/abstract=928538> , 2006 (e).
- Nagendra, S.: *Catalogue of Test Problems for Optimization Algorithm Verification*, Technical Report 97-CRD-110, General Electric Company, 1997.
- Nelder, J.A. and Mead, R.: "A Simplex method for function minimization" *Computer Journal*, 7: pp. 308-313, 1964.
- Parsopoulos, K.E. and Vrahatis, M.N., "Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization", *Natural Computing*, 1 (2-3), pp. 235- 306, 2002.

- Prigogine, I. and Stengers, I.: *Order Out of Chaos: Man's New Dialogue with Nature*, Bantam Books, Inc. NY, 1984.
- Silagadge, Z.K.: "Finding Two-Dimensional Peaks", Working Paper, Budkar Institute of Nuclear Physics, Novosibirsk, Russia, arXiv:physics/0402085 V3 11 Mar 2004.
- Simon, H.A.: *Models of Bounded Rationality*, Cambridge Univ. Press, Cambridge, MA, 1982.
- Smith, A.: *The Theory of the Moral Sentiments*, The Adam Smith Institute (2001 e-version), 1759.
- Sumper, D.J.T.: "The Principles of Collective Animal Behaviour", *Phil. Trans. R. Soc. B.* 361, pp. 5-22, 2006.
- Törn, A.A and Viitanen, S.: "Topographical Global Optimization using Presampled Points", *J. of Global Optimization*, 5, pp. 267-276, 1994.
- Törn, A.A.: "A search Clustering Approach to Global Optimization" , in Dixon, LCW and Szegö, G.P. (Eds) *Towards Global Optimization – 2*, North Holland, Amsterdam, 1978.
- Tsallis, C. and Stariolo, D.A.: "Generalized Simulated Annealing", *ArXiv condmat/9501047 v1* 12 Jan, 1995.
- Valentine, R.H.: *Travel Time Curves in Oblique Structures*, Ph.D. Dissertation, MIT, Mass, 1937.
- Veblen, T.B.: "Why is Economics Not an Evolutionary Science" *The Quarterly Journal of Economics*, 12, 1898.
- Veblen, T.B.: *The Theory of the Leisure Class*, The New American library, NY. (Reprint, 1953), 1899.
- Vesterstrøm, J. and Thomsen, R.: "A comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems", *Congress on Evolutionary Computation, 2004. CEC2004*, 2, pp. 1980-1987, 2004.
- Whitley, D., Mathias, K., Rana, S. and Dzubera, J.: "Evaluating Evolutionary Algorithms", *Artificial Intelligence*, 85, pp. 245-276, 1996.
- Yao, X. and Liu, Y.: "Fast Evolutionary Programming", in Fogel, LJ, Angeline, PJ and Bäck, T (eds) *Proc. 5th Annual Conf. on Evolutionary programming*, pp. 451-460, MIT Press, Mass, 1996.