Principles and State-of-the-Art of Engineering Optimization Techniques

Ning Xiong, Miguel León Ortiz
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
E-mail: ning.xiong@mdh.se, miguel.leonortiz@mdh.se

***Abstract*—This paper gives a survey of the principles and the state-of-the-art of engineering optimization techniques. Both the classic and emerging approaches to nonlinear optimization problems are reviewed and analyzed. All the techniques are discussed in two basic types: point-based transition and population-based transition, depending on whether a single point or multiple points are generated as new approximate solution(s) in each step. We also consider multi-objective tasks as new application trend and point out the strong potential of population-based methods to tackle multiple objectives simultaneously.**

Keywords-optimization techniques; point-based optimization; population-based optimization

I. INTRODUCTION

Nowadays, optimization has become an important issue in industrial design and product development [1]. It is necessary to enhance system performance whereas reduce product cost to meet challenges in the competitive market. From engineering perspective, optimization means adjusting or fine tuning system designs in terms of one or more performance factors. This is not a trivial task, in particular when the problem space is complex and of high-dimensionality. Application of suitable optimization techniques has shown its benefit in supporting human designers to acquire optimal or near optimal solutions within a short design time.

Generally, an engineering optimization problem can be formulated as:

$$\text{minimize} \quad f(X) = f(x_1, x_2, \cdots, x_n)$$

$$\text{subject to} \quad g_i(x_1, x_2, \cdots x_n) \le 0, \quad i = 1 \cdots m$$

where (x_1, x_2, …x_n) is the vector of design variables, $g_i(i=1 \ldots m)$ denote constraint functions that define the region of feasible solutions in the problem space, and function $f(X)$ provides objective values for vectors of variables representing alternative designs. The set of design variables x_i can take continuous or discrete values or a mixture of both depending on specific problems. Besides, the above statement is generic since maximization of a certain function is equivalent to minimization of the minus of it.

The optimization techniques can be divided into two basic categories: linear programming [2] and non-linear programming [3], [4]. The former is applied to optimization problems that have linear objective and constraint functions. Important progresses in this area include the polynomial-time ellipsoid algorithm [5] and the interior point algorithm [6], both were proposed to reduce time complexity and to allow for extremely efficient problem handling in the optimization procedure. At present, linear programming has been advanced to a soundly founded discipline and widely used technology for linear optimization problems. The second category of optimization is called nonlinear programming, which refers to the consortium of methods and approaches that are designed to deal with problems with nonlinear objective or constraint functions [7]. Nonlinearity is a very common property for many engineering optimization problems, and solving such problems often presents a challenge due to the high complexity, high dimensionality and multi-modality of the problem space. Although many techniques for nonlinear programming have been developed, there are always pros and cons for them and no single method can more competently solve all kinds of problems than others.

This paper focuses on the study of nonlinear optimization techniques. Special emphasis is made on presenting the general principle and ideas of how to reach the optimum rather than the details of computational procedures. Both the classic and emerging approaches to nonlinear optimization problems are reviewed and analyzed. We also consider multi-objective tasks as new application trend when discussing the potential capability of optimization methods.

The organization of this paper is as follows. Section II highlights the basic idea and principle for general nonlinear optimization problems. The review of concrete approaches for optimization is given in Sections III and IV, respectively. The type of approaches called **point-based transition** is discussed in Section III, and the type of approaches called **population-based transition** is addressed in Section IV. Finally, Section V provides concluding remarks and discussion.

II. GENERAL PRINCIPLE OF OPTIMIZATION

Mathematically, it is well known that an optimum of a nonlinear function $f(x_1, x_2, \ldots x_n)$ must be some point at which the partial derivatives of the function with respect to all variables are equal to zero, i.e.,

$$\frac{\partial f}{\partial x_i} = 0, \qquad i = 1, 2, \cdots, n \tag{1}$$

A solution satisfying all the equations in (1) is called a stationary point of function f. Further, the stationary point is

 ISBN: 978-1-61208-290-5

a minimum solution if the Hessian matrix of the second-order derivatives, as defined in (2), is positive definite.

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (2)$$

The above principle suggests a simple procedure to obtain exact solution of an optimization problem. It is done by finding all stationary points of the objective function and then examining the property of the Hessian matrices of these points. The global optimum solution is selected from those stationary points for which the Hessian matrices are positive definite.

Unfortunately, the approach to exact solutions of optimization is rarely applicable in engineering practice. The main reason lies in the difficulty of acquiring the derivative information analytically. In many applications, only concrete objective values of individual designs are calculable through specific calculations such as simulation. But the explicit expression of the objective function is not available, not to mention the analytic formulation of the partial derivative functions. It follows that we are unable to construct the equations as formulated in (1) for determining the stationary points of the objective function.

Numerical approaches present a pragmatic alternative to solve engineering optimization problems. The main idea is to create arbitrary initial approximate(s) to the problem and then improve them progressively. The whole procedure consists of a number of iterations. In each iteration, new approximate(s) are created from the old one(s) as more promising solution(s). Depending on the number of approximates generated at a single step, two types of numerical approaches (point-based and population-based transitions) can be defined and explained as follows.

With point-based transitions, only one point as new approximate is generated and evaluated in each of the iterations. The new point is made as transition from an old one with expected better performance. The general form of such a transition can be expressed as

$$X_{i+1} = X_i + h_i S_i \quad (3)$$

where X_{i+1} and X_i denote the new and old approximates respectively, h_i decides the length of transition, and S_i is a vector determining the direction of the move from X_i. There are many different methods to determine the direction vector S_i in the literature. Some use merely values of the objective function while others require partial derivative information in addition to the objective values.

Sometimes, it is beneficial to apply point-based transition methods in combination with random sampling to increase their global search ability and thereby reducing the risk of getting stuck into local optima. For instance, the initial approximate for iteration can more favorably be decided by resorting to a random scheme [8], which generates a set of uniformly distributed points in the region of feasible solutions. We then select the sample solution that receives the best objective value as the starting point of search. The other possibility is to follow the multi-start strategy [9] when doing optimization with point-based transitions. This means that we run the optimization algorithm multiple times and every time a sample solution is selected randomly as the starting point. The best solution found from individual runs is treated as the final solution of the global optimum.

Population-based transition starts from an initial population of feasible solutions. Then it undergoes an iterative procedure in which new populations are successively created from old populations to reach progressively refined approximates to the problem. As many points in the space are explored simultaneously, population-based transition is superior to point-based transition in the global search ability; hence it has less likelihood of ending with a local minimum. Many biologically inspired optimization techniques rely on transitions of populations, such as genetic algorithms, memetic algorithms, differential evolution, particle swarm optimization, as well as ant colony optimization, which will be reviewed in Section IV.

III. OPTIMIZATION WITH POINT-BASED TRANSITION

The approaches of this type explore the problem space via transition from one feasible solution to another. The transition procedure is controlled by either deterministic or probabilistic rules. Six well known approaches in this category will be surveyed here.

A. Hill-Climbing

Hill-climbing [10] is the simplest numerical approach for optimization. It starts by creating an arbitrary solution (approximate) to the problem and then it evaluates all the neighbors of the current solution. If the best neighbor has a lower objective value than the current solution, the current one is replaced by that neighbor and the search moves on to the next iteration, otherwise the search is terminated.

Hill-climbing is a local search and can only be applied in discrete spaces as it implicitly assumes a finite number of feasible neighbors at every point. The advantages of hill-climbing lie in its simplicity and high efficiency. It has been widely used to solve many machine learning and technical optimization problems (e.g., [11], [12]). Hill-climbing is particularly recommended when there is limited time for search; for example, for real-time systems.

B. Gradient Descent

Gradient descent [13] aims to solve continuous optimization problems. It has very similar idea to that of hill-climbing. The only difference between both methods lies in the way to determine the best successor solution from a current one. Since it is not possible to evaluate every successor in the continuous space, the gradient information is utilized to identify the direction of move to reduce the objective function most quickly. Hence, the normalized direction vector of the move can be written in (4). The length

 ISBN: 978-1-61208-290-5

of move along S_i can be determined by solving a one-dimensional optimization problem. The golden section method is often used in gradient descent to find the optimal size of transition at each step.

$$S_i = \frac{\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right)}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)^2}} \quad (4)$$

Gradient descent is simple and very useful in solving many optimization problems when partial derivatives of the object function are available. However, as local search scheme, this method cannot guarantee the global optimality of the solutions returned. It should preferably be combined with the multi-start strategy to increase the chance of finding the global minimum. The other weakness with gradient descent is that, when the current solution gets close to a minimum solution, the search will become quite inefficient due to the decreasing lengths of the moves.

C. Newton's Method

Newton's method [14] attempts to improve the speed of convergence of gradient descent in the vicinity of a minimum solution. According to Taylor's expansion, the objective function near a minimum X^* can be expressed by an approximate form as:

$$f(X) = f(X^* + \Delta X) \approx f(X^*) + (\Delta X)^T g + \frac{1}{2}(\Delta X)^T H(\Delta X) \quad (5)$$

where g is the vector of the first-order partial derivatives of the objective function and H is the Hessian matrix of the second-order partial derivatives of the objective function. For all the first-order derivatives at the optimum X^* are zero, Eq.(5) is simply rewritten as:

$$f(X) \approx f(X^*) + \frac{1}{2}(\Delta X)^T H(\Delta X) \quad (6)$$

From (6), it can be seen that the objective function is approximately quadratic in the vicinity of X^*. A quadratic function has the following property:

$$g = H(\Delta X) \quad (7)$$

where g is the vector of partial derivatives evaluated at the current point, and H is the Hessian matrix which is constant for a quadratic function. Using the property in (7) enables us to obtain the minimum solution X^* from a nearby point X in terms of the following transition rule:

$$X^* \approx X - H^{-1} g \quad (8)$$

This transition rule indicates that a single move suffices to reach the minimum X^* when the current solution is nearby. This shows a substantial improvement of the late convergence speed compared with that of gradient descent.

Nevertheless, it has to be noticed that globally the objective function is not quadratic. Hence, an iterative procedure is needed to generate a sequence of moves for progressive refinement. At iteration i, we first calculate g and H at the current point X_i, and then, we use the Newton's method to create the next refined solution as

$$X_{i+1} \approx X_i - H_i^{-1} g_i \quad (9)$$

Generally, the Newton's method stated above is still a local approach and it has two drawbacks. Firstly, it requires heavy computation with the Hessian matrix of second-order derivatives and its inverse. Secondly, the method is only efficient in the neighborhood of an optimum, but away from the optimum it may progress very slowly and even diverge. So, our suggestion is not employing the Newton's method alone, but in combination with some other optimization technique and using it at the final stage.

D. Tabu Search

Tabu search [15] [16] is a metaheuristic local search algorithm to solve discrete optimization problems. It can be considered as extension of the hill-climbing search in the two aspects as follows. Firstl, there is added driving force to enforce the local minimization procedure out of a local minimum. Secondly, various memory structures are used to store historical information which is then utilized to guide the further exploration of new solutions. For instance, the tabu list is introduced as short term memory to save recently visited solutions to prevent cyclic behaviors during the search. Intermediate and long term memory is used to intensify and diversify the search to ensure adequate exploration of the problem space.

The search starts from an arbitrary point as the current solution. All the solutions in its neighborhood that are not in the tabu list or satisfy the aspiration level are successor solutions and their objective values are calculated. Then the move is made to the best successor according to the objective values, and the tabu list is updated accordingly. Both uphill and downhill moves are allowed here to give chance to escape from a local minimum. This process is repeated in a number of iterations until the termination condition is satisfied.

It is useful to apply tabu search in many practical scenarios [17] [18], mainly in combinatorial problems. A main limitation with this technique is that it requires considerable memory resources to store historical information. Besides, domain specific knowledge is needed to design suitable aspiration criteria.

E. Simulated Annealing

Simulated annealing [19] [20] is a stochastic and meta-heuristic algorithm for solving global optimization problems. It is inspired by the physical principle of annealing used in material engineering. In an annealing process, the solid is first heated to a high temperature, causing atoms to move away from their initial positions. When the material cools down slowly, the atoms adjust themselves into a new thermal equilibrium that corresponds to a minimum energy state.

 ISBN: 978-1-61208-290-5

The algorithm is iterative and the main idea is to randomly select a new solution in the neighborhood of the current solution at every step. The difference of objective values, ΔE, between the new and current solutions is calculated as analogy to the change of energy. If the new solution is better than the current one (ΔE<0), the current solution is replaced by the new one. In case when the new solution is worse (ΔE>0), there is still a chance to move to it. The probability of this move is given by the Boltzmann probability function:

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{T}\right) \quad (10)$$

The parameter T in (10) is the temperature used during the search. In the early iterations, the temperature is high, which results in high probability of moving into inferior points and thereby avoiding local minima. Contrarily, during late stages, the temperature is reduced to such a level that gives little chance for accepting worse solutions and consequently the search will finally converge.

The merit with simulated annealing is that it does not require the objective function to be continuous and differentiable, and it can handle both continuous and discrete optimization problems. But, proper parameter settings with this method are not difficult.

F. Simplex Method

A simplex is a geometric object consisting of n+1 points (vertices) in the n-dimensional spaces. Every vertex of the simplex corresponds to a feasible solution to the problem. The initial simplex can be generated randomly. The main idea is to move the simplex iteratively and every time replacing the worst vertex of it with a new better point. The search terminates when the standard deviation of the objective values of the n+1 vertices of the simplex is lower than a specified value.

According to the simplex method by Nelder and Mead [21], the new better points for replacement are generated through the operations such as reflection, expansion, and contraction. The worst vertex is first reflected through the centroid of the remaining points of the simplex. If the reflection produces a better point, expansion is done to see whether the objective function can be reduced further via moving in the same direction. Otherwise, if the reflected point is not satisfactory, contraction is performed via generation of a point between the worst vertex and the centroid for possible replacement.

The main advantage of the simplex method is that it is a global optimization technique and it does not require any derivative information of the objective function. The method is robust and efficient with a small number of design variables but it does not scale well up to high-dimensional problems. As noted in [7], the efficiency of simplex diminishes when the design variables are more than five.

IV. OPTIMIZATION WITH POPULATION-BASED TRANSITION

The approaches of this type explore the problem space via transition from one population of feasible solutions to another. They are biologically inspired techniques and probabilistic rules are used to create new solutions from old ones. Five well known approaches in this category will be reviewed here.

A. Genetic Algorithms

Genetic algorithms (GAs) are stochastic optimization algorithms that emulate the mechanics of natural evolution [22]. They are attractive to be applied in engineering optimization tasks due to the two following reasons. First, a GA evaluates many points in the search space simultaneously, as opposed to a single point, thus reducing the chance of converging to the local optimum. Second, a GA uses only values of objective functions; therefore they do not require the search space to be differentiable or continuous.

Essentially, a GA is an iterative procedure maintaining a constant population size. An individual in the population encodes a possible solution to the problem with a string analogous to a chromosome in nature. At each step of iteration, new individuals are created via applying genetic operators on selected parents, and subsequently some of the old, weak individuals are replaced by new strong ones. In this manner, the performance of the population will be gradually improved in the evolutionary process.

A classical GA works with binary code, i.e., individuals in the population are represented by binary strings. However, binary coding would not be the most appropriate choice in applications to optimization problems with continuous spaces. One reason lies in the matter of resolution, i.e., a binary string is inherently related to some loss of precision for representing the continuous value of a variable. The other reason is the extra job of decoding that is needed when doing fitness evaluation for a binary string in the population.

The other alternative is to directly adopt arrays of real numbers as population individuals. Real-coded GAs have been studied by many researchers and nowadays become a popular, extended version of GAs for solving real-valued optimization problems. The interesting features of real-coded GAs together with their used mechanisms and genetic operators haven been carefully discussed in [23] and [24]. In [25], real-coded GA was used to optimize similarity models for case-based reasoning.

B. Memetic Algorithms

Memetic algorithms (MAs) [26] are population-based metaheuristic search methods inspired by the principle of natural evolution and Dawkin's notion of memes capable of local adaptation. MAs can be considered as enhancement of GAs by embedding local search to allow for self-refinements of individuals. According to the idea of

 ISBN: 978-1-61208-290-5

Lamarckian learning [27], local search can be done on all or part of the population to reach a local optimum or improve the current solution.

As hybridization of GAs and local search, MAs aim to exploit the best search regions gathered by global sampling with GA. Hence, an important demand for MAs is the synergy between the exploration abilities of the GA and the exploitation abilities of the local search. In [28], a local search scheme was combined with the global search capability of evolutionary algorithms for rule extraction. The simplex method by Nelder and Mead was adopted as the local search mechanism in the memetic algorithm [29] for optimal fuzzy controller design. The hierarchical memetic algorithm was proposed in [30] for combined feature selection and similarity modeling in case-based reasoning.

C. Differential Evolution

Differential evolution (DE) [31] is a stochastic and meta-heuristic technique that has been developed for solving optimization problems with real parameters. It provides a powerful tool for searching for optimal solutions in high-dimensional spaces that are nonlinear, non-differentiable, non-continuous, and containing multiple local optima. DE is similar to GAs in the sense that both are evolutionary, population-based algorithms. But DE algorithms differ from GAs in the way evolutionary operators are manipulated to produce new child solutions. The main loop of DE algorithms is briefly explained in the following.

A DE algorithm maintains a population of real-valued parameter vectors and works iteratively. Each iteration starts with mutation, in which three distinct parameter vectors are randomly selected for every population member. The weighted differences between two parameter vectors are added to the third parameter vector to get the perturbed vector. Then, crossover is done to combine the population member and the perturbed vector to yield a new trial vector. Every parameter in the perturbed vector has a certain probability to enter the trial vector. Finally, the trial vector replaces the old population member if it has a lower objective value. A comprehensive review of various DE algorithms together with associated operators is given in [32].

DE attains increasing popularity in engineering applications due to its attractive features such as fewer running parameters to specify, ease in programming, high efficiency, as well as strong global search ability. In [31] and [33], it was indicated that DE algorithms were more efficient and more accurate than several other optimization methods, including controlled random search, simulated annealing and genetic algorithms. A weakness for DE is that there is no theoretic proof for its convergence.

D. Particle Swarm Optimization

Particle swarm optimization (PSO) algorithms [34] mimic the flocking behaviors of animals in their movement. Similar to GAs, PSO algorithms work with a population of particles which represent feasible solutions to the problem. The particles move around in the search space to improve their fitness (objective values), iteratively. The movement of each particle is determined in terms of both its best position in the history and also the best position known so far from all particles. In view of this, the speed of a particle at iteration k+1 is updated as:

$$v_{k+1} = w \cdot v_k + c_1 \cdot r_1 \cdot (P_{pb}^k - X_k) + c_2 \cdot r_2 \cdot (P_{gb}^k - X_k) \quad (11)$$

In (11), P_{pb} , and P_{gb} , denote, respectively, the best position of the particle and the best known position from all particles, w is the momentum, X_k is the position of the particle at iteration k, r_1 and r_2 are two randomly generated positive numbers, and c_1 and c_2 are the parameters used to balance the individual and social influences.

PSO is simpler than GAs in its nature. Therefore, it incurs lower computational cost in creating a new population from the old one. But, the performance of PSO is heavily dependent on the parameters w, c_1, and c_2 in (11), and proper valuation of such parameters to ensure strong search ability is a crucial task.

E. Ant Colony Optimization

Ant colony optimization (ACO) [35] [36] mimics the behavior of a colony of ants in searching for food. It is a population-based metaheuristic technique used to solve hard combinatorial problems. Prior to applying ACO, the optimization problem has to be transformed into the problem of path finding on a graph. Then a group of ants work collectively to find a shortest path on the graph by pheromone communication during path formation [37].

An ant builds its path incrementally. It starts from a randomly selected vertex and then chooses an edge to go to the next vertex. The choice of an edge is stochastic yet its probability is decided by the pheromone values and heuristic information associated with the edge. The most well known rule for determining the selection probability for edge c_{ij} is given in (12)

$$P(c_{ij}) = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{c_{il} \in S_f} \tau_{il}^{\alpha} \eta_{il}^{\beta}} \quad (12)$$

where S_f denotes the set of feasible edges immediately after the current partial path, τ_{ij} and η_{ij} are the pheromone and heuristic values respectively associated with edge c_{ij}, α and β are the parameters controlling the relative importance of pheromone versus heuristic information.

Further, when the ants completed their paths, the quality of their solutions is used to update the pheromone values of the edges. These updated values are then utilized by the ants in the next iteration to build new paths. This procedure continues until the maximum iteration number is reached or all ants tend to produce a similar path.

 ISBN: 978-1-61208-290-5

F. Extension to Finding Multiple Solutions

As is seen in this section, the optimization techniques with population-based transition are actually beam search, they can handle a set of feasible solutions at the same time. It follows that it would be relatively easy to modify or extend these techniques such that a set of optimal solutions rather than a single one will be returned from a single run of the algorithm. For example, the niching genetic algorithms were developed in [38] to find multiple interesting solutions in the job shop scheduling.

Finding multiple solutions as interesting trade-offs is also very important for multi-objective optimization tasks. Traditional ways to cope with multiple objectives is to build an overall objective function as weighted combination of individual objective values. However it is very hard to assign exact weights to reflect human preference, and therefore the final solution obtained may not be most preferred by human decision makers.

Multi-objective genetic algorithms have received intensive research for more than one decade (see [39][40], as examples). A nice feature of these algorithms is that a diversity of Pareto-optimal solutions can be found and presented to human decision makers, who then choose the most preferred alternative according to their preference. The Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [41] is an improved version of the early algorithms, which incorporates a fast non-dominated sorting algorithm and the crowding-distance assignment to improve the efficiency and effectiveness of the algorithm respectively.

More recently, multi-objective PSO algorithms were developed as extension of the single-objective counterparts. The key issue here is how to select global and local best particles in terms of multiple criteria. Different selection strategies have been proposed for this purpose. In [42] the tournament niche method was used to decide the global best particle, and the local best particle was identified according to Pareto-dominance. The other interesting strategy is to stochastically choose the global best particle from the non-dominated solutions using density-based probabilities [43].

V. CONCLUSION

This paper provides a survey of the principles and the state-of-the-art of numerical techniques for solving nonlinear optimization problems. All the techniques discussed are classified into two basic types: point-based transition and population-based transition, depending on whether a single point or multiple points are generated as new approximate solution(s) in each step of the iterations. Generally, the point-based approaches are simple and effective for optimization problems with a low number of parameters. However, when the dimension of the space increases, they become less efficient and are more likely to get stuck in a local optimum. In many practical applications, a point-based search method is often combined with the random sampling or multi-start strategy to increase the chance to find a global optimum. The population-based approaches are superior to the point-based ones in global search capability; they seem to be more suitable to be applied in high-dimensional search spaces. But, larger memory requirements and more computational cost are connected with them as side effects.

Most of the optimization approaches addressed here are derivative-free. This is a very useful property to promote wide applications in various situations without requiring the problem space to be continuous and differentiable. On the other side, some classical search methods such as gradient descent and Newton's method are also valuable and recommended to use, as long as the derivative information is available or achievable. The derivative-based methods are theoretically well founded and can contribute to substantial improvement of local search performance. The exploitation of derivative-based local search in a global evolutionary algorithm would be a promising direction of research for building new memetic computing frameworks.

Both the point-based and population-based approaches can be used to solve multi-objective optimization problems. For point-based approaches, it is necessary to construct an overall objective function as a weighted combination of individual objective values, and a single solution will be returned after the running of an algorithm. Since there is no clear relation between the weighting and the solution obtained, we cannot guarantee that the solution found is really the one that is most preferred by human decision makers. Comparatively, the population-based approaches appear to be more appropriate or have more potential for tackling problems with multiple objectives. As noted in Section IV, the population-based methods like GAs can easily be extended to deal with multiple objectives simultaneously and thereby returning a set of Pareto-optimal solutions rather than a single one. This enables human decision makers (designers) to choose the most preferred solution from a group of interesting trade-offs.

ACKNOWLEDGEMENT

The work is within the EMOPAC project granted by the Swedish Knowledge Foundation. We are also grateful to ABB FACTS, Prevas, and VG Power for co-financing the research.

REFERENCES

[1] L. Shi, S. Olafsson, and Q. Chen "An optimization framework for product design," Management Science, vol. 47, 2001, pp. 1681-1692 .

[2] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, Linear Programming and Network Flows, 2nd Edtion, New York: John Wiley & Sons, 1990.

[3] D. G. Luenberger, Linear and Non-linear Programming. New York: Addison-Wesley, 1990.

[4] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, Nonlinear Programming – Theory and Algorithms. New York: John

[5] L. G. Khachian, "A polynomial algorithm in linear programming," Soviet Mathematics Doklady, vol. 20, 1979, pp. 1093-1096Wiley & Sons, 1993.

 ISBN: 978-1-61208-290-5

[6] N. Karmarkar, "A new polynomial algorithm for linear programming," Combinatorica, vol. 4, 1984, pp. 373-395.
[7] W. H. Swann, "A survey of non-linear optimization techniques," FEBS Letters, vol. 2, 1969, pp. 39-55.
[8] A. H. G, Rinnoy Kan, G. G. E. Boender, and G. T. Timmer, "A stochastic approach to global optimization," In: K. Schnittkowski (Ed.) Computational mathematical programming, 1985, pp. 281-308.
[9] J. S. Arora, O. A. Elwakeil, and A. I. Chahande, "Global optimization method for engineering applications: a review,"Structural Optimization, vol. 9, 1995, pp. 137-159.
[10] S. Russel and P. Norvig, Artificial Intelligence: A Modern Approach (2nd ed.), New Jersey: Prentice Hall, pp. 111–114, 2003.
[11] N. Xiong and P. Funk, "Construction of fuzzy knowledge bases incorporating feature selection," Soft Computing, vol. 10, 2006, pp. 796 – 804.
[12] K. A. Sullivan and S. H. Jacobson, "Ordinal hill climbing algorithms for discrete manufacturing process design optimization problems," Discrete Event Dynamic Systems, vo1. 10, 2000, pp 307-324.
[13] M. Avriel, Nonlinear Programming: Analysis and Methods, Dover Publishing, 2003.
[14] R. Battiti, "First- and second-order methods for learning: Between steepest descent and Newton's method," Neural Computation, vol. 4, 1992, pp. 141-166.
[15] F. Glover, "Tabu search – Part one," ORSA Journal Computing, vol. 1, 1989, pp. 190-206.
[16] J. A. Bland and G. P. Dawson, "Tabu search and design optimization," Computer Aided Design, vol. 23, 1991, pp. 195-201.
[17] J. A. Bland, "Structural design optimization with reliability constraints using tabu search,"Engineering Optimization, vol. 30, 1998, pp. 55-74.
[18] J. M. Emmert, S. Lodha, and D. K. Bhatia, "On using tabu search for design automation of VLSI systems," Journal of Heuristics, vol. 9, 2003, pp. 75-90.
[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, 1983, pp. 671-680.
[20] R. W. Eglese, "Simulated annealing: a tool for operational research.," European Journal of Operational Research, vol. 46, 1990, pp. 271-281.
[21] J. A. Nelder and R. A. Mead, "A simplex for function minimization," Computer Journal, vol. 7, 1965, pp. 308-313.
[22] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, New York: Addison-Wesley, 1989.
[23] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," Artificial Intelligence Review, vol. 12, 1998, pp. 265-319.
[24] F. Herrera, M. Lozano, and A. M. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study," International Journal of Intelligent Systems, vol. 18, 2003, pp. 309-338.
[25] N. Xiong, "Fuzzy rule-based similarity model enables learning from small case bases," Applied Soft Computing, vol. 13, 2013, pp. 2057-2064.
[26] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," IEEE Trans. Evolutionary Computation, vol. 9, no. 5, 2005, pp. 474-488.
[27] Y. S. Ong and A. J. Keane, "Meta-Lamarckian in memetic algorithm," IEEE Trans. Evolutionary Computation, vol. 8, 2004, pp. 99-110.
[28] J. H. Ang, K. C. Tan, and A. A. Mamun, "An evolutionary memetic algorithm for rule extraction," Expert Systems with Applications, vol. 37, 2010, pp. 1302–1315.
[29] X. Zhang, A. Erdem, H. Shi, N. Xiong, D. Isovic, and M. Bobesic, "A novel memetic algorithm incorporating Nelder-Mead method in fuzzy controller design," Proc. Int. Conf. Computational Intelligence and Software Engineering, 2012, pp. 55-59.
[30] N. Xiong and P. Funk, "Combined feature selection and similarity modeling in case-based reasoning using hierarchical memetic algorithm," Proc. of the IEEE World Congress on Computational Intelligence, 2010, pp. 1537–1542.
[31] R. Storn and K. Price, "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, 1997, pp. 341–359.
[32] S. Das, "Differential evolution: A survey of the state-of-the-art," IEEE Trans. Evolutionary Computation, vol. 15, 2011, pp. 4-31.
[33] M. M. Ali, and A. Torn, "Population set based global optimization algorithms: Some modifications and numerical studies," Computers and Operations Research, vol. 31, 2004, pp. 1703 – 1725.
[34] J. Kennedy and R. Eberhart "Particle swarm optimization," Proc. IEEE Int. Conf. Neural Networks, 1995, pp. 1942–1948.
[35] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," IEEE Trans. Systems, Man, and Cybernetics – Part B, vol. 26, 1996, pp. 29-41.
[36] M. Dorigo, "Ant colony optimization," Scholarpedia, vol. 2, 2007, pp. 1461.
[37] M. Dorigo and L.M. Gambardella, "Ant Colony System : A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evolutionary Computation, vol. 1, 1997, pp. 53-66.
[38] E. Perez, M. Posada, and F. Herrera, "Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling," Journal of Intelligent Manufacturing, 2012, vol. 23, pp. 341-256.
[39] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms, part I: A unified formulation," IEEE. Trans. Systems, Man, & Cybernetics, Part A, vol. 28, 1998, pp. 26-37.
[40] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithm," Evolutionary Computation, vol. 2, 1995, pp. 221-248.
[41] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evolutionary Computation, vol. 6, 2002, pp. 182-197.
[42] D. S. Liu, K. C. Tan, C. K. Huang, C. K. Goh, and W. K. Ho, "On solving multiobjective bin packing problems using evolutionary particle swarm optimization," European Journal of Operational Research, vol. 190, 2008, pp. 357-382.
[43] P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal, "Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients," Information Sciences, vol. 177, 2007, pp. 5033-5049.

 ISBN: 978-1-61208-290-5