State-of-the-Art Seminar Report

on

Nature Inspired Algorithms & Their Applications

Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosphy

in

Computer Science

by

Divya Kumar

2009RCS55

Under the Guidance of

Dr. Krishn Kumar Mishra

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY

ALLAHABAD – 211004, INDIA

June, 2014

Research plan on

"Nature Inspired Algorithms and their Applications"

by
Divya Kumar
(Registration No. - 2012RCS55)
Computer Science and Engineering Department

(Approved)
Dr. K. K. Mishra
Assistant Professor,
CSED, MNNIT Allahabad, Allahabad.

❝ Guru Brahma, Guru Vishnu, Guru Devo Maheshwara. Guru Sakshath Parambrahma, Tasmai Shri Gurave Namaha❞

-Shri Saraswati Gangadhar

Contents

List of Figures

Chapter 1

Optimization and Meta-Heuristics

The desire for optimality (perfection) is inherent for humans. The search for extremes inspires mountaineers, scientists, mathematicians, and the rest of the human race. A beautiful and practical mathematical theory of optimization (i.e. search-for-optimum strategies) is developed since the sixties when computers become available. Every new generation of computers allows for attacking new types of problems and calls for new methods. The goal of the theory is the creation of reliable methods to catch the extremum of a function by an intelligent arrangement of its evaluations (measurements). This theory is vitally important for modern engineering and planning that incorporate optimization at every step of the complicated decision making process. Everyone who studies calculus knows that an extremum of a smooth function is reached at a stationary point where its gradient vanishes. Some may also remember the Weierstrass theorem [1] which proclaims that the minimum and the maximum of a function in a closed finite domain do exist. Does this mean that the problem is solved? A small thing remains: To actually find that maximum. This problem is the subject of the optimization theory that deals with algorithms for search of the extremum. More precisely, we are looking for an algorithm to approach a proximity of this maximum and we are allowed to evaluate the function (to measure it) in a finite number of points.

1.1 Introduction

In mathematics and computer science, term *optimization* refers to the study of problems in which one seeks to minimize or maximize an objective function by systematically choosing the values of real or integer variables within an allowed set. Thus an optimization problem is the problem of finding the best solution from all feasible solutions. These optimization problems are very important because these are designed to either earn profit or to minimize the loss incurred. Due to there wide applicability in business, engineering and other areas, a number of algorithms have been proposed in literature to solve these problems to get optimal solutions in minimum possible time[2].

Optimization is the process of making something better. An engineer or scientist proposes up a new idea and optimization improves on that idea. Optimization consists in trying variations on an initial concept and using the information gained to improve on the idea. Optimization problems can be seen in many areas such as manufacturing systems, economics, physical sciences, computational systems, etc.In the last years, several optimization methods have been developed based on the nature inspired analogy. However, these methods are not always able to solve some problems in the best way. Although it has been shown that these are good methods to solve complex problems, there are not yet methods to know the optimal parameters to solve problems that can be set at the beginning when using the algorithms. As an attempt to improve these methods many researchers have used fuzzy logic to adapt parameters and achieve better results than with the original methods. Therefore in this review we are including the optimization methods with parameter adaptation using fuzzy logic[3].

1.2 Global Optimization

Global optimization is a branch of applied mathematics and numerical analysis that deals with the optimization of a function or a set of functions according to some criteria. It refers to finding the extreme value of a given nonconvex function in a certain feasible region and such problems are classified in two classes; unconstrained and constrained problems. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is

known as a combinatorial optimization problem and when the variables are continious the problem is mathematical optimization. Solving global optimization problems has made great gain from the interest in the interface between computer science and operations research.

1.2.1 Single Objective Optimization Problem

A global optimization problem [4] can be defined as finding the parameter vector $\overrightarrow{x}$ that minimizes an objective function $f(\overrightarrow{x})$:

$$minimize\, f(\overrightarrow{x}),\ \overrightarrow{x} = (x_1,\, x_2, ...,\, x_{n-1},\, x_n) \in \mathbb{R}^n \tag{1}$$

which is constrained by the following inequalities and/or equalities:

$$l_i \leq x_i \leq u_i,\ i = 1, ..., n \tag{2}$$

$$subject\ to:\ g_j(\overrightarrow{x}) \leq 0,\ for\ j = 1, ..., p \tag{3}$$

$$h_j(\overrightarrow{x}) = 0,\ for\ j = p+1, ..., q \tag{4}$$

$f(\overrightarrow{x})$ is defined on a search space, S , which is a n dimensional rectngle in $\mathbb{R}^n$ ($S \subseteq \mathbb{R}^n$). The variable domains are limited by their lower and upper bounds as shown in equation (2). This problem is also known as a constrained optimization problem. If it is an unconstrained optimization problem, then both $p = 0$ and $q = 0$.

1.2.2 Multi-Objective Optimzation Problem

In most of the real world applications/problems, several criteria have to be satisfied simultaneously. These problems can be classified as multi-criteria decision making problems, which seek to optimize various performance attributes , related to the problem, simultaneously. Multi-Objective Optimization Problems (MOOPs) have a number of objectives (often conflicting) which are to be minimized or maximized[5] and these are inherent in fields, where optimal decisions are needed to be taken, like aircraft design, the oil and

gas industry, product and process design, finance, automobile design and so on. Their solution is to find, not only a single solution, but several solutions, that represent the best possible trade-offs among all the objectives that we aim to optimize[6]. A Multi-Objective Optimization Problem (MOOP) [7] can be defined as follows:

Given an n-dimensional vector of decision variables $x = \{x_1, x_2, ..., x_n\}$, in solution space X. We have to find a particular vector x^p, that minimizes/maximizes a given set of *K* objective functions $f(x^p) = \{f_1(x^p),\ f_2(x^p), ...,f_k(x^p)\}$, where f_i is i^{th}objective. A function $f : X \rightarrow Y$ is used to evaluate the quality of a specific solution by assigning it to an objective vector $\{y_1, y_2, ..., y_k\}$ in the objective space *Y*. The mapping takes place between an n-dimensional solution vector and a k-dimensional objective vector. Also the solution space is generally restricted by a series of constraints and bounds on the decision variables[8].

The Solution Trade Offs

For multi-criteria decision making problems, objectives are generally conflicting which prevents simultaneous optimization of each objective . So it is difficult to make decisions in the presence of trade-offs between two or more conflicting objectives. Optimizing a particular solution with respect to a single objective can result in unacceptable results (or at least depreciated results) with respect to other objectives. Some of the examples of this situation are:

- Quality and cost of a product.
- Performance and cost for an algorithm.
- Bandwidth, delay and cost in a network.
- Efficiency and fuel consumption of an engine.
- Weight and strength of a component.

The Pareto Optimality

The solution of MOOPs is contained in a set of trade-off solutions called Pareto optimal set that contains all the non dominated solutions. This set represents the best possible trade-offs among the various objectives. A feasible solution $x^P \in X$ is called Pareto optimal if and only if there is no other solution $x \in X$ such that x dominates x^p [6]. And a solution $s = \{s_1, s_2, ...s_n\}$ dominates a solution $z = \{z_1, z_2, ..., z_n\}$, or s has a better non-dominated rank r_s, $r_s < r_z$ where r_z is the rank of solution z, in a minimization context, iff both conditions 1 and 2 are true:

Condition 1: The solution $f(s)$ is no worse than $f(z)$ in all objectives:
$\forall i \in [1...k]\ f_i(s) \leq f_i(z)$.

Condition 2: The solution $f(s)$ is strictly better than $f(z)$ in at least one objectives:
$\exists i \in [1...k]\ f_i(s) < f_i(z)$.

The objective in solving MOOPs is to determine the Pareto-optimal front having the characteristics like:

- A close approximate of the true Pareto front.
- Solutions in this front should be well distributed.
- This set must be global for the entire search space.
- Multimodal problems

1.2.3 Multi-modal Optimization

Optimization problems are often multi-modal; that is, they possess multiple good solutions. They could all be globally good (same cost function value) or there could be a mix of globally good and locally good solutions. Obtaining all (or at least some of) the multiple solutions is the goal of a multi-modal optimizer. In a multimodal optimization task, the main purpose is to find multiple optimal (global and local) solutions associated with a single objective function. Starting with the preselection method suggested in 1970, most of the existing evolutionary algorithms based methodologies employ variants of niching

in an existing single-objective evolutionary algorithm framework so that similar solutions in a population are de-emphasized in order to focus and maintain multiple distant yet near-optimal solutions.

Classical optimization techniques due to their iterative approach do not perform satisfactorily when they are used to obtain multiple solutions, since it is not guaranteed that different solutions will be obtained even with different starting points in multiple runs of the algorithm. Evolutionary Algorithms[7] are however a very popular approach to obtain multiple solutions in a multi-modal optimization task [9]. Evolutionary Algorithms are discussed in detail in next chapter.

Chapter 2

Evolutionary Algorithms

Evolutionary Algorithm (EA) stands for a generic population-based class of stochastic optimization methods that simulate the procedure of natural evolution. The history of the field suggests that the origins of EAs can be traced back to the late 1960s, and since then quite a lot of evolutionary methodologies have been proposed, mainly genetic algorithms, evolutionary programming, genetic programming and evolution strategies[10]. All of these methodologies are ubiquitous now days. All these techniques work under the umbrella of a common idea: given a set of candidate solutions, the environmental pressure causes natural selection and which in turn causes a rise in the fitness of population members. Based on this fitness, some of the better candidates are chosen, as parents, to seed the next generation by applying recombination and /or mutation to them. Thus executing recombination and mutation leads to a set of fresh candidates (the offspring) that compete – based on their fitness – with the old ones (parents) for a place in the next generation. To say about termination criteria, this process is iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached (a fix number of iterations are completed). EA's are "generate and test" algorithms with the following components:

1 Representation

EA's require a method of representing the solution in a form that conveys the necessary information, required for computation. Objects forming possible solutions

within the original problem context are referred to as phenotypes and corresponding to them, their encoding, the individual within the EA's are called genotypes. So there must be a method to provide a mapping between these two representations.

2 Fitness Function

A fitness value is assigned to each solution depending on how close it actually is to the optimum solution. This function quantifies the optimality of a solution, so that a particular solution may be ranked against all other.

3 Parent Selection Method

The goal of this method is to emphasize the good solutions and eliminate the bad ones. The good solutions (solutions having high fitness value) are allowed to reproduce while bad ones die out. While comparing it with natural evolution, those individuals that are better are more likely to survive and propagate their genetic material.

4 Variation methods

The variation methods are use to create the new solutions (offspring). This is achieved through crossover and mutation operators. Cross over is mainly responsible for the search aspect of genetic algorithm. It merges the information in parent genotypes to create new and fresh offspring genotypes. Mutation operator is mainly needed to keep the diversity in the population. It is a unary operator that is applied over a genotype and produces a slight variant (mutant) genotype. Thus crossover looks for a solution near to existing solutions and mutation looks for a completely new area of search space.

5 Replacement Strategy

It is the process of making the choice that which individual will be going in the next generation. Thus individual solutions are differentiated based on their quality and the better one would be the survivors.

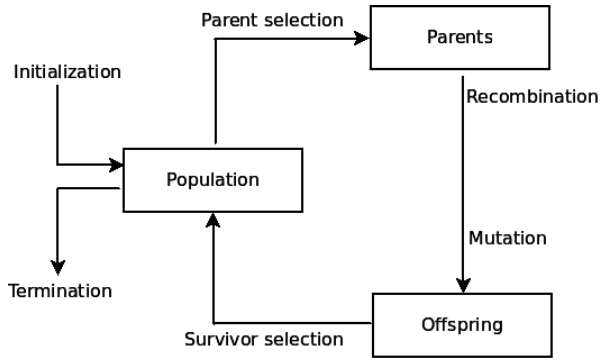

Figure 1: Structure of an Evolutionary Algorithm

Evolutionary computing techniques mostly involve metaheuristic optimization algorithms. This field has substantially developed from genetic algorithms to genetic programming, passing by evolutionary strategies and evolutionary programming [11]. A time line of various EA's is shown in fig. 2. Various other well known flavors of this field are:

- ***Evolutionary algorithms:***
  Genetic Algorithms[12]
  Genetic Programming[13]
  Evolutionary Programming[14]
  Evolution Strategies[15]

- ***Swarm intelligence:***
  Ant Colony Optimization[16]
  Particle Swarm Optimization[17]
  Artificial Bee Colony[18]
  Bat Algorithm[19]
  Gravitational Search Algorithm[20]

- ***Others:***
  Artificial Immune Systems[21]
  Scatter Search[22]
  Simplex Search[23][24]
  Simulated Annealing[25]

Dynamic Programming[26]

Differential Evolution[27]

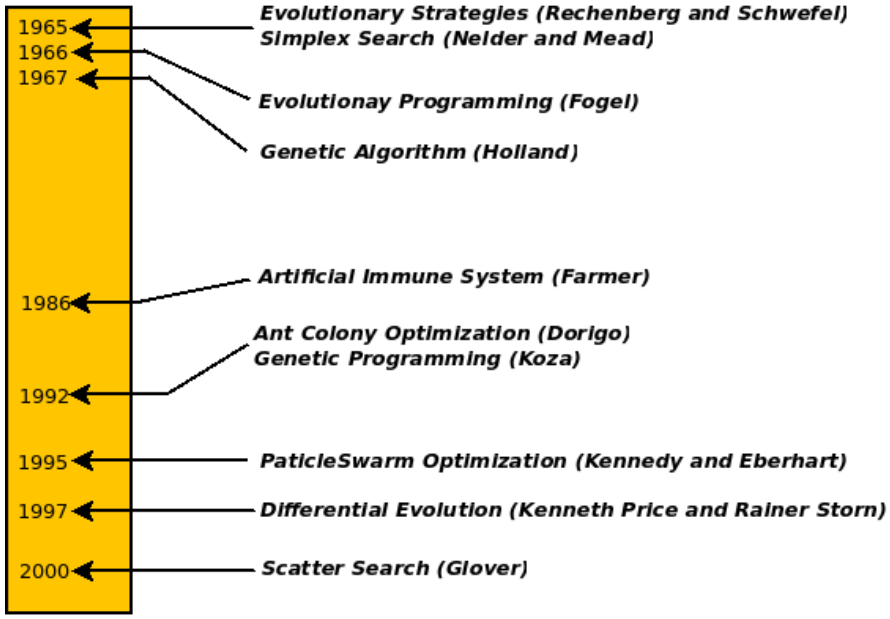

Figure 2: Growth of Evolutionary Algorithms

2.1 Genetic Algorithm (GA)

This is the most popular type of EA. One seeks the solution of a problem in the form of strings of numbers (traditionally binary), by applying operators such as selection, recombination and mutation [28]. GAs are well suited for optimizing combinatorial problems.

Solution to a problem solved by genetic algorithms is evolved. A typical GA starts with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population via cross-over (**reproduction**) operators. This is motivated by a hope, that the new population will be better than the old one (**evolution**). Solutions which are selected to form new solutions (**offspring**) are selected according to their **fitness** - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations

or improvement of the best solution) is satisfied. A mapping between real-life and Genetic Algorithm is given as follows:

- Population ⇒Set of solutions.
- Individual ⇒Candidate solution to a problem.
- Fitness ⇒Quality of a solution.
- Chromosome ⇒Encoding for a Solution.
- Gene ⇒Part of the encoding of a solution.
- Reproduction ⇒Crossover or Recombination

Most of the real world applications/problems can be classified as multi-criteria decision making problems, which seek to optimize various performance attributes, related to the problem. These problems are known as Multi-Objective Optimization Problems (MOOPs) and they have multiple objectives which are to be minimized or maximized [5]. These type of problems are inherent in fields, where optimal decisions are needed to be taken, like aircraft design, the oil and gas industry, product and process design, finance, automobile design and so on.

Evolutionary computation refers to a class of algorithms whose search strategies model the natural, biological evolution trends. This phenomenon includes natural selection, reproduction, genetic inheritance and the Darwinian strife for survival. To be clearer I must say that I'm pointing towards the concept of "survival of fittest". Today, evolutionary computation has turn out to be a collective name for a range of problem-solving techniques based on principles of biological evolution. These techniques are being increasingly broadly applied to a numerous problems from different domains, ranging from practical applications in industry and commerce to leading-edge scientific research.

2.2 Genetic Programming

In artificial intelligence, genetic programming (GP) is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a

user-defined task. Essentially GP is a set of instructions and a fitness function to measure how well a computer has performed a task. It is a specialization of genetic algorithms (GA) where each individual is a computer program. It is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task.

2.3 Evolutionary Programming

Evolutionary programming is one of the four major evolutionary algorithm paradigms. It is similar to genetic programming, but the structure of the program to be optimized is fixed, while its numerical parameters are allowed to evolve.

It was first used by Lawrence J. Fogel in the US in 1960 in order to use simulated evolution as a learning process aiming to generate artificial intelligence. Fogel used finite state machines as predictors and evolved them. Currently evolutionary programming is a wide evolutionary computing dialect with no fixed structure or (representation), in contrast with some of the other dialects.

2.4 Evolution Strategies

Evolution Etrategy (ES) is an optimization technique based on ideas of adaptation and evolution. Evolution strategies use natural problem-dependent representations, and primarily mutation and selection, as search operators. In common with evolutionary algorithms, the operators are applied in a loop. An iteration of the loop is called a generation. The sequence of generations is continued until a termination criterion is met.

As far as real-valued search spaces are concerned, mutation is normally performed by adding a normally distributed random value to each vector component. The step size or mutation strength (i.e. the standard deviation of the normal distribution) is often governed by self-adaptation (see evolution window). Individual step sizes for each coordinate or correlations between coordinates are either governed by self-adaptation or by covariance matrix adaptation (CMA-ES).

The (environmental) selection in evolution strategies is deterministic and only based

on the fitness rankings, not on the actual fitness values. The resulting algorithm is therefore invariant with respect to monotonic transformations of the objective function. The simplest evolution strategy operates on a population of size two: the current point (parent) and the result of its mutation.

2.5 Ant Colony Optimization

In computer science and operations research, the ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food (see Ant communication).

Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants' following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.

2.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of

quality. PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

2.7 Artificial Bee Colony

The Artificial Bee Colony Algorithm (ABC) is an optimization algorithm based on the intelligent foraging behaviour of honey bee swarm. In the ABC model, the colony consists of three groups of bees: employed bees, onlookers and scouts. It is assumed that there is only one artificial employed bee for each food source. In other words, the number of employed bees in the colony is equal to the number of food sources around the hive. Employed bees go to their food source and come back to hive and dance on this area. The employed bee whose food source has been abandoned becomes a scout and starts to search for finding a new food source. Onlookers watch the dances of employed bees and choose food sources depending on dances.

2.8 Bat Algorithm

Bat Algorithm is based on the echolocation behaviour of microbats with varying pulse rates of emission and loudness. The idealization of the echolocation of microbats can be summarized as follows: Each virtual bat flies randomly with a velocity v_i at position (solution) x_i with a varying frequency or wavelength and loudness A_i. As it searches and finds its prey, it changes frequency, loudness and pulse emission rate$r$. Search is intensified by a local random walk. Selection of the best continues until certain stop criteria are met. This essentially uses a frequency-tuning technique to control the dynamic behaviour of a swarm of bats, and the balance between exploration and exploitation can be controlled by tuning algorithm-dependent parameters in bat algorithm.

2.9 Gravitational Search Algorithm

Gravitational search algorithm (GSA) based on the law of gravity and the notion of mass interactions. The GSA algorithm uses the theory of Newtonian physics and its searcher agents are the collection of masses. In GSA, there is an isolated system of masses. Using the gravitational force, every mass in the system can see the situation of other masses. The gravitational force is therefore a way of transferring information between different masses. In GSA, agents are considered as objects and their performance is measured by their masses. All these objects attract each other by a gravity force, and this force causes a movement of all objects globally towards the objects with heavier masses. The heavy masses correspond to good solutions of the problem. The position of the agent corresponds to a solution of the problem, and its mass is determined using a fitness function. By lapse of time, masses are attracted by the heaviest mass. We hope that this mass would present an optimum solution in the search space.

2.10 Artificial Immune Systems

Artificial Immune Systems (AIS) are a class of computationally intelligent systems inspired by the principles and processes of the vertebrate immune system. The field of Artificial Immune Systems (AIS) is concerned with abstracting the structure and function of the immune system to computational systems, and investigating the application of these systems towards solving computational problems from mathematics, engineering, and information technology. AIS is a sub-field of Biologically-inspired computing, and Natural computation, with interests in Machine Learning and belonging to the broader field of Artificial Intelligence.

2.11 Scatter Search

Scatter search (SS) is a heuristic for integer programming. In the original proposal, solutions are purposely (i.e., non-randomly) generated to take account of characteristics in various parts of the solution space. Scatter search orients its explorations systematically relative to a set of reference points that typically consist of good solutions obtained by

prior problem solving efforts, where the criteria for "good" are not restricted to objective function values, and may apply to subcollections of solutions rather than to a single solution, as in the case of solutions that differ from each other according to certain specifications. Weighted linear combinations provide the main mechanism to generate new trial points within the space containing the reference points, accompanied by a generalized (successively iterated) rounding mechanism to assure these trial points satisfy integer feasibility conditions in the case where some variables are required to receive integer values. These mechanisms are oriented toward the goal of creating weighted centers of selected sub-regions, including regions external to the convex hull of the reference points.

2.12 Simplex Search

Simplex Algorithm (or simplex method) is a popular algorithm for linear programming. A system of linear inequalities defines a polytope as a feasible region. The simplex algorithm begins at a starting vertex and moves along the edges of the polytope, until it reaches the vertex of the optimum solution. Thus Simplex Algorithm systematically explores the extreme points of the feasible region and moves from an extreme point to its improving neighbor. It can be shown that for a linear program in standard form, if the objective function has a minimum value on the feasible region then it has this value on (at least) one of the extreme points. This in itself reduces the problem to a finite computation since there is a finite number of extreme points, but the number of extreme points is unmanageably large for all but the smallest linear programs.

2.13 Simulated Annealing

Simulated annealing (SA) is a generic probabilistic metaheuristic for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For certain problems, simulated annealing may be more efficient than exhaustive enumeration — provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.

The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Both are attributes of the material that depend on its thermodynamic free energy. Heating and cooling the material affects both the temperature and the thermodynamic free energy. While the same amount of cooling brings the same amount of decrease in temperature it will bring a bigger or smaller decrease in the thermodynamic free energy depending on the rate that it occurs, with a slower rate producing a bigger decrease.

2.14 Dynamic Programming

Dynamic Programming (DP) is a method for solving complex problems by breaking them down into simpler subproblems. It is applicable to problems exhibiting the properties of overlapping subproblems[1] and optimal substructure (described below). When applicable, the method takes far less time than naive methods that don't take advantage of the subproblem overlap (like depth-first search).

The idea behind dynamic programming is quite simple. In general, to solve a given problem, we need to solve different parts of the problem (subproblems), then combine the solutions of the subproblems to reach an overall solution. Often when using a more naive method, many of the subproblems are generated and solved many times. The dynamic programming approach seeks to solve each subproblem only once, thus reducing the number of computations: once the solution to a given subproblem has been computed, it is stored or "memo-ized": the next time the same solution is needed, it is simply looked up. This approach is especially useful when the number of repeating subproblems grows exponentially as a function of the size of the input.

2.15 Differential Evolution

Differential Evolution (DE) is a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However,

metaheuristics such as DE do not guarantee an optimal solution is ever found.

DE is used for multidimensional real-valued functions but does not use the gradient of the problem being optimized, which means DE does not require for the optimization problem to be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. DE can therefore also be used on optimization problems that are not even continuous, are noisy, change over time, etc. DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand. In this way the optimization problem is treated as a black box that merely provides a measure of quality given a candidate solution and the gradient is therefore not needed.

2.16 Multi-Objective Optimization Through Evolutionary Algorithms

Recently there has been increasing research interest in the field of evolutionary multi-objective optimization algorithms. These algorithms combine two major disciplines first evolutionary computation and second the theoretical frameworks of multi-criteria decision making. Evolutionary algorithms has become the most popular heuristic for multi-objective design and optimization problems. This is because of the matching characteristics of industrial problems and evolutionary algorithms. Various proposed MOEA's are: Multi-Objective Genetic Algorithm (MOGA) [29], Strength Pareto Evolutionary Algorithm (SPEA) [30], Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [31] and Multi-objective Evolutionary Algorithm (MEA)[32]. These are well- evaluated algorithms, being used in many industries.

Real-life problems are challenging in a way that they contain multiple conflicting objectives and all these objectives are are to be considered equally, further more they cannot be solved by local methods because they contain several local optima. A number of sophisticated stochastic optimization techniques such as Artificial Bee Colony[18], Particle Swarm Optimization (PSO) [17], Ant Colony Optimization (ACO) [16] and Differential

Evolution (DE) [27] can also be used to find the pareto set. Also they work well for difficult problems with discontinuous search spaces and are less expensive and are easy to simulate. Artificial Bee Colony with its applications and variants is discussed in detail in the next chapter.

References

[1] N. Cotter, "The stone-weierstrass theorem and its application to neural networks," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 1, no. 4, pp. 290–295, 1990. [Online]. Available: http://europepmc.org/abstract/MED/18282849

[2] K. K. Mishra, "New bio-inspired optimization algorithm," Ph. D. Thesis, MNNIT Allahabad, India, December 2012.

[3] F. Valdez, P. Melin, and O. Castillo, "A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation," *Expert Systems with Applications*, vol. 41, no. 14, pp. 6459 – 6466, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417414002127

[4] M. Dorigo, "Ant colony optimization," vol. 2, no. 3, p. 1461, 2007, revision 90969.

[5] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[6] C. Coello Coello, C. Dhaenens, and L. Jourdan, "Multi-objective combinatorial optimization: Problematic and context," in *Advances in Multi-Objective Nature Inspired Computing*, ser. Studies in Computational Intelligence, C. Coello Coello, C. Dhaenens, and L. Jourdan, Eds. Springer Berlin Heidelberg, 2010, vol. 272, pp. 1–21.

[7] X. Yu and M. Gen, "Introduction to evolutionary algorithms. 2010."

[8] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[9] K. Deb and A. Saha, "Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '10. New York, NY, USA: ACM, 2010, pp. 447–454. [Online]. Available: http://doi.acm.org/10.1145/1830483.1830568

[10] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2003.

[11] A. Abraham, N. Nedjah, and L. de Macedo Mourelle, "Evolutionary computation: from genetic algorithms to genetic programming," in *Genetic Systems Programming*. Springer, 2006, pp. 1–20.

[12] L. Davis *et al.*, *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991, vol. 115.

[13] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[14] D. B. Fogel, L. J. Fogel, and J. W. Atmar, "Meta-evolutionary programming," in *Signals, systems and computers, 1991. 1991 Conference record of the twenty-fifth asilomar conference on*. IEEE, 1991, pp. 540–545.

[15] T. Bäck, G. Rudolph, and H.-P. Schwefel, "Evolutionary programming and evolution strategies: Similarities and differences," in *In Proceedings of the Second Annual Conference on Evolutionary Programming*. Citeseer, 1993.

[16] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 36–39.

[17] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.

[18] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[19] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[20] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[21] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1, pp. 187–204, 1986.

[22] F. Glover, M. Laguna, and R. Martí, "Scatter search," in *Advances in evolutionary computing*. Springer, 2003, pp. 519–537.

[23] G. B. Dantzig, "Expected number of steps of the simplex method for a linear program with a convexity constraint." DTIC Document, Tech. Rep., 1980.

[24] S.-K. S. Fan and E. Zahara, "A hybrid simplex search and particle swarm optimization for unconstrained optimization," *European Journal of Operational Research*, vol. 181, no. 2, pp. 527–548, 2007.

[25] P. J. Van Laarhoven and E. H. Aarts, *Simulated annealing*. Springer, 1987.

[26] R. E. Bellman and S. E. Dreyfus, "Applied dynamic programming," 1962.

[27] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[28] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimization," in *Metaheuristics for multiobjective optimisation*. Springer, 2004, pp. 3–37.

[29] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary computation*, vol. 3, no. 1, pp. 1–16, 1995.

[30] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[32] X. Hu, C. A. C. Coello, and Z. Huan, "A new multi-objective evolutionary algorithm derived from the line-up competition algorithm," *Engineering Optimization*, vol. 37, no. 4, pp. 351–379, 2005.