ISSN Print: 2394-7500
ISSN Online: 2394-5869
Impact Factor: 5.2
IJAR 2017; 3(7): 538-541
www.allresearchjournal.com
Received: 13-11-2016
Accepted: 14-12-2016

Sushila
Research Scholar, FCI, Sonipat, Haryana, India

Correspondence
Sushila
Research Scholar, FCI, Sonipat, Haryana, India

A review on heuristic algorithms

Sushila

Abstract
An algorithm is a precise, step-by-step set of instructions for solving a task. An algorithm doesn't solve a task; it gives you a series of steps that, if executed correctly, will result in a solution to a task. You use algorithms every day but you often do not explicitly think about the individual steps of the algorithm. Nowadays computers are used to solve incredibly complex problems. Butin order to manage with a problem we should develop an algorithm. Sometimes the human brain is not able to accomplish this task. Moreover, exact algorithms might need centuries to manage with formidable challenges. In such cases heuristic algorithms that find approximate solutions but have acceptable time and space complexity play indispensable role. In this paper heuristics, their areas of application and the basic underlying ideas are surveyed. We also describe in more detail some modern heuristic techniques, namely Evolutionary Algorithms, Genetic algorithm and neural network.

Keywords: SVM, tabu search, ACO, PSO

1. Introduction
The most important among a variety of topics that relate to computation are algorithm validation, complexity estimation and optimization. Wide part of theoretical computer science deals with these tasks. Complexity of tasks in general is examined studying the most relevant computational resources like execution time and space. The ranging of problems that are solvable with a given limited amount of time and space into well-defined classes is a very intricate task, but it can help incredibly to save time and money spent on the algorithms design.
Modern problems tend to be very intricate and relate to analysis of large data sets. Even if an exact algorithm can be developed its time or space complexity may turn out unacceptable. But in reality it is often sufficient to find an approximate or partial solution. Such admission extends the set of techniques to cope with the problem. We discuss heuristic algorithms which suggest some approximations to the solution of optimization problems. In such problems the objective is to find the optimal of all possible solutions that is one that minimizes or maximizes an objective function. The objective function is a function used to evaluate a quality of the generated solution. Many real-world issues are easily stated as optimization problems. The collection of all possible solutions for a given problem can be regarded as a search space, and optimization algorithms, in their turn, are often referred to as search algorithms.
Approximate algorithms entail the interesting issue of quality estimation of the solutions they find. Taking into account that normally the optimal solution is unknown, this problem can be a real challenge involving strong mathematical analysis. In connection with the quality issue the goal of the heuristic algorithm is to find as good solution as possible for all instances of the problem. There are general heuristic strategies that are successfully applied to manifold problems.

2. Need of Algorithms
Algorithms are important for many reasons:
- An algorithm documents the "how to" for accomplishing a particular task.
- If an algorithm is written well, it can be used to accomplish not only a single task but a whole group of related tasks.
- The existence of an algorithm means that the task can potentially be automated (i.e. performed by a computer).

- The automation of redundant, tedious, or dangerous tasks frees people from having to perform these boring, time-consuming, or potentially lethal tasks.
- The automation of some tasks makes new things possible (e.g., accessing web pages from all over the entire world in the blink of an eye).

Consider telephones for example. When telephones were invented, the problem of how to connect one phone caller to another caller arose. For many years this problem was solved manually by phone operators who pulled wires from a console and plugged them into the correct connecting wire. It has been estimated that if this manual method was still used today to handle the current daily phone calls made around the world, every person on the planet would have to be a phone operator! Thankfully, given the invention of fast computers and algorithms to instruct those computers, almost all phone calls made today are automatically connected without human intervention. What would your life be like without a phone? Can you imagine it without an algorithm that is the way life would be!

If we can think and reason precisely and solve algorithmic problems in one domain (e.g. computer programming), then your ability to analyze and solve problems in other areas will improve.

3. Algorithms and Complexity

It is difficult to imagine the variety of existing computational tasks and the number of algorithms developed to solve them. Algorithms that either give nearly the right answer or provide a solution not for all instances of the problem are called heuristic algorithms. This group includes a plentiful spectrum of methods based on traditional techniques as well as specific ones. For the beginning we sum up the main principles of traditional search algorithms.

The simplest of search algorithms is exhaustive search that tries all possible solutions from a predetermined set and subsequently picks the best one. Local search is a version of exhaustive search that only focuses on a limited area of the search space. Local search can be organized in different ways. Popular hill-climbing techniques belong to this class. Such algorithms consistently replace the current solution with the best of its neighbors if it is better than the current. For example, heuristics for the problem of inter group replication for multimedia distribution service based on Peer-to-Peer network is based on hill climbing strategy.

Divide and conquer algorithms try to split a problem into smaller problems that are easier to solve. Solutions of the small problems must be combinable to a solution for the original one. Branch-and-bound technique is a critical enumeration of the search space. It enumerates, but constantly tries to rule out parts of the search space that cannot contain the best solution. Dynamic programming is an exhaustive search that avoids re-computation by storing the solutions of sub problems. The key point for using this technique is formulating the solution process as a recursion.

A popular method to construct successively space of solutions is greedy technique that is based on the evident principle of taking the (local) best choice at each stage of the algorithm in order to find the global optimum of some objective function.

Usually heuristic algorithms are used for problems that cannot be easily solved. Classes of time complexity are defined to distinguish problems according to their "hardness". Class P consists of all those problems that can be solved on a deterministic Turing machine in polynomial time from the size of the in- put. Turing machines are an abstraction that is used to formalize the notion of algorithm and computational complexity.

Class NP consists of all those problems whose solution can be found in polynomial time on anon-deterministic Turing machine. Since such a machine does not exist, practically it means that an exponential algorithm can be written for an NP-problem, nothing is asserted whether a polynomial algorithm exists or not. A subclass of NP, class NP-complete includes problems such that a polynomial algorithm for one of them could be transformed to polynomial algorithms for solving all other NP problems. Finally, the class NP-hard can be understood as the class of problems that are NP-complete or harder. NP-hard problems have the same trait as NP-complete problems but they do not necessary belong to class NP, that is class NP-hard includes also problems for which no algorithms at all can be provided.

In order to justify application of some heuristic algorithm we prove that the problem belongs to the classes NP-complete or NP-hard. Most likely there are no polynomial algorithms to solve such problems; therefore, for sufficiently great inputs heuristics are developed.

4. Heuristic Techniques

Branch-and-bound technique and dynamic programming are quite effective but their time complexity often is too high and unacceptable for NP-complete tasks. Hill-climbing algorithm is effective, but it has a significant drawback called pre- mature convergence. Since it is "greedy", it always finds the nearest local optima of low quality. The goal of modern heuristics is to overcome this disadvantage.

Simulated annealing algorithm, invented in 1983, uses an approach similar to hill-climbing, but occasionally accepts solutions that are worse than the current. The probability of such acceptance is decreasing with time.

Tabu search extends the idea to avoid local optima by using memory structures. The problem of simulated annealing is that after "jump" the algorithm can simply repeat its own track. Tabu search prohibits the repetition of moves that have been made recently.

Swarm intelligence was introduced in 1989. It is an artificial intelligence technique, based on the study of collective behavior in decentralized, self-organized, systems. Two of the most successful types of this approach are Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). In ACO artificial ants build solutions by moving on the problem graph and changing it in such a way that future ant scan builds better solutions. PSO deals with problems in which a best solution can be represented as a point or surface in an n-dimensional space. The main advantage of swarm intelligence techniques is that they are impressively resistant to the local optima problem.

Evolutionary Algorithms succeed in tackling premature convergence by considering a number of solutions simultaneously.

Neural Networks are inspired by biological neuron systems. They consist of units, called neurons, and interconnections between them. After special training on some given data set Neural Networks can make predictions for cases that are not in the training set. In practice Neural Networks do not always work well because they suffer greatly from problems of under fitting and over fitting. These problems correlate with the accuracy of prediction. If a network is not complex enough it may simplify the laws which the data obey. From the other point of view, if a network is too complex it can take into account the noise that usually assists at the training data set while inferring the laws. The quality of prediction after training is deteriorated in both cases. The problem of premature convergence is also critical for Neural Networks.

Support Vector Machines (SVMs) extend the ideas of Neural Networks. They successfully overcome premature convergence since convex objective function is used; therefore, only one optimum exists. Classical divide and conquer technique gives elegant solution for separable problems. In connection with SVMs, that provide effective classification, it becomes an extremely powerful instrument. Later we discuss SVM classification trees, which applications currently present promising object for research.

A. Evolutionary algorithms

Evolutionary algorithms are methods that exploit ideas of biological evolution, such as reproduction; mutation and recombination, for searching the solution of an optimization problem. They apply the principle of survival on a set of potential solutions to produce gradual approximations to the optimum. A new set of approximations is created by the process of selecting individuals according to their objective function, which is called fitness for evolutionary algorithms, and breeding them together using operators inspired from genetic processes. This process leads to the evolution of populations of individuals that are better suited to their environment than their ancestors.

The main loop of evolutionary algorithms includes the following steps:

- Initialize and evaluate the initial population.
- Perform competitive selection.
- Apply genetic operators to generate new solutions.
- Evaluate solutions in the population.
- Start again from point 2 and repeat until some convergence criteria is satisfied.

Sharing the common idea, evolutionary techniques can differ in the details of implementation and the problems to which they are applied. Genetic programming searches for solutions in the form of computer programs. Their fitness is determined by the ability to solve a computational problem. The only difference from evolutionary programming is that the latter fixes the structure of the program and allows their numerical parameters to evolve. Evolution strategy works with vectors of real numbers as representations of solutions, and uses self-adaptive mutation rates. The most successful among evolutionary algorithms are Genetic Algorithms (GAs). They have been investigated by John Holland in 1975 and demonstrate essential effectiveness. GAs is based on the fact that the role of mutation improves the individual quite seldom and, therefore, they rely mostly on applying recombination operators. They seek solutions of the problems in the form of strings of numbers, usually binary.

The prevalent areas for applying genetic algorithms are optimization problems requiring large scale high performance computing resources.

The main issue to consider for resource allocation is to minimize the number of Unit of Bandwidth (UB) that should be allocated. The problem has been proven to be in the class NPhard. Authors applied genetic algorithm instead of greedy search used before. Computer simulation has been performed for the idealized cellular system with one base station that can support m connections required one UB per second, constructed in a cluster of B cells. As result, it is stated that the genetic algorithm can improve the system capacity utilization. Substantial increase of the consumer demand on computers and mobile phones made network optimization problems extremely relevant. Being general technique that can be easily modified under the various conditions, evolutionary algorithms are widely used in this area.

To give an example let us mention Adaptive Mesh Problem (AMP) that has a goal to minimize the required number of base stations of cellular network to cover a region. Like the previous discussed problem, AMP is NP-hard. One of the evolutionary techniques, called Hybrid Island Evolutionary Strategy (HIES) has been applied to tackle this problem. It represents evolutionary algorithm borrowing features from two types of genetic algorithms, namely island model GA and fine-grained or cellular GA. The original problem has been transformed into a geometric meshing generation problem and specific genetic operators, micro mutation, macro mutation and crossover, have been altering an array of hexagonal cells. Initially regular honeycomb was transformed to irregular mesh that fits better to the real-life conditions. The desired result, the reduction of the number of base stations, has been achieved.

B. Genetic Algorithms

Genetic algorithms are the unorthodox search or optimization algorithms which have currently found favour with the users of optimization techniques in solution of a variety of real life problems. Genetic algorithms were suggested by John Holland in his book, Adaptation in Natural and Artificial Systems‟. As the name suggests, genetic algorithms were inspired by the processes observed in natural evaluation. They attempt to mimic these processes and utilize them for solving a wide range of optimization problems. In general, genetic algorithms perform directed random searches through a given set of alternatives with the aim of finding the best alternative with respect to some given criteria of goodness. Genetic algorithms search for the best alternative through evolution of chromosomes. Basic steps in a genetic algorithm are as follows. First, an initial population of acceptable chromosomes is randomly selected.

Next, a new population of chromosomes is evolved from the given population by giving greater importance to chromosomes with high degree of fitness. This is called natural selection. If the stopping criteria is not met by the evolved population, some genetic-like operations such as crossovers "mutation" etc. are performed on the existing chromosomes to produce new chromosomes called off springs.

C. Neural Networks

An artificial neural network is a computational structure inspired by observed processes in natural networks of biological neurons in the brain. It consists of simple computational units called neurons which are highly interconnected. Each interconnection has a strength which is expressed by a number called its weight. The basic capability of a neural network is to learn patterns from examples. This is achieved by adjusting the values of the weights according to some learning algorithm. Learning is guided by specifying for each training input pattern, the class to which the pattern is supposed to belong. Adjustments in the values of weights are made so as to minimize the difference between the desired and actual outputs. Once a network converges to a solution in which the patterns in the training set are recognized with high fidelity, it is capable of classifying an unknown input pattern with other patterns that are close to it in terms of the same distinguishing features. This learning process of a neural network is known as supervised learning. A neural network is also sometimes trained by a process known a sun supervised learning. In an unsupervised learning process, the network forms its own classification of the patterns. The classification is usually based on commonalities in certain features of input patterns. This requires that a neural network implementing an unsupervised learning algorithm be able to identify common features across the range of input patterns. A deviation from classical networks in any of these features requires that a properly modified learning algorithm be developed. This in some cases is not easy.

5. Conclusion

This paper has presented an overview of heuristics that are approximate techniques to solve optimization problems. Usually heuristic algorithms are developed to have low time complexity and applied to the complex problems. We briefly defined basic traditional and modern heuristic strategies. Evolutionary algorithms and Support Vector Machines were considered more comprehensively. Due to their eminent characteristics they gained a great popularity. Recently appeared research results confirm the fact that their applications can be significantly enlarged in the future.

6. References

1. Cook SA. An overview of computational complexity", in Communication of the ACM, 1983; 26(6):401-408.
2. Cormen T, Ch. Leiserson R. Rivest. Introduction to algorithms. MIT Press, 1989.
3. Garey MR, Johnson DS. Computers and Intractability. Freeman & Co, 1979.
4. Xiang Z, Zhang Q, Zhu W, Zhang Z, Zhang YQ. Peer-to-Peer Based Mul- timediaDistribution Service, in IEEE Transactions on Multimedia. 2004; 6(2):343-355.
5. Aydin ME, Fogarty TC. A Distributed Evolutionary Simulated Annealing Algorithm forCombinatorial Optimization Problems, in Journal of Heuristics, 2004; 24(10):269-292.
6. Battiti R. Reactive search: towards self-tuning heuristics, in Modern heuristic searchmethods. Wiley & Sons, 1996, 61-83.
7. Eberhart R, Shi Y, Kennedy J. Swarm intelligence. Morgan Kaufmann, 2001.
8. Kr¨ose B, Smagt P. An introduction to Neural Networks. University of Amsterdam, 1996.
9. Karaboga D. Pham. Intelligent Optimisation Techniques: Genetic Algorithms, TabuSearch, Simulated Annealing and Neural Networks. Springer Verlag, 2000.
10. Wu X, Sharif BS, Hinton OR. An Improved Resource Allocation Scheme for Plane Cover Multiple Access Using Genetic Algorithm, in IEEE Transactions on Evolutionary Computation. 2005; 9(1):74-80.
11. Crput JC, Koukam A, Lissajoux T, Caminada A. Automatic Mesh Genera- tion forMobile Network Dimensioning Using Evolutionary Approach, in IEEE Transactions on Evolutionary Computation, 2005; 9(1):18-30.
12. Divina F, Marchiori E. Handling Continuous Attributes in an Evolutionary In- ductive Learner, in IEEE Transactions on Evolutionary Computation. 2005; 9(1):31-43.
13. Yager RR. Adaptive Models for the Defuzzification Process. Proc. Second Intern.Conf. on Neural Networks, Iizuka, Japan, 1992, 65-71.