© UNIVERSAL RESEARCH REPORTS | REFEREED | PEER REVIEWED
ISSN : 2348 - 5612 | Volume : 04 , Issue : 06 | July - September 2017

Various Heuristic techniques used for problem solving : A Review

Somnath Maji[1], Somnath Banerjee[2]

[1]Department of CSE Bankura Unnayani Institute of Engg. , Bankura

[2]BCA, Bankura ICIS College, Bankura.

ISSN : 2348-5612 © URR
9 770234 856124

Abstract : A heuristic (to find) approach to a problem is an empirical search or optimization method that often works at solving the problem, but doesn't have any of the rigorous proof that people like physicists and mathematicians expect. Nobody knows if it will always give the best answer to the problem. To put it simply, it is a short cut to solving difficult problems. Optimization algorithms can be roughly divided into two categories: exact algorithms and heuristics. Exact algorithms are designed in such a way that it is guaranteed that they will find the optimal solution in a finite amount of time. However, for very difficult optimization problems (e.g. NP-hard or global optimization) this "finite amount of time" may increase exponentially respect to the dimensions of the problem. Heuristics do not have this guarantee, and therefore generally return solutions that are worse than optimal. However, heuristic algorithms usually find "good" solutions in a "reasonable" amount of time.

Many heuristic algorithms are very specific and problem-dependent. On the other hand, a metaheuristic is a high-level problem-independent algorithmic frame-work that provides a set of guidelines or strategies to develop heuristic optimization algorithms. But a concrete definition has been elusive and in practice many researchers and practitioners interchange these terms. Thus, the term metaheuristic is also used to refer to a problem specific implementation of a heuristic optimization algorithm according to the guidelines expressed in such a framework.

Key Words : Algorithm, Heuristic, Tabu search, Swarm intelligence , Simulated annealing algorithm

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 intragroup 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. This technique is effective but its use is limited because there is no a great number of problems that can be easily partitioned and combined in a such way.

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 subproblems. The key point for using this technique is formulating the solution process as a recursion.

Greedy Techniques A popular method to construct succes sively 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 b e solved on a deterministic Turing machine in polynomial time from the size of the input. Turing machines are an abstraction that is used to formalize the notion of algorithm and computational complexity. A comprehensive description of them can be found in . Class NP consists of all those problems whose solution can be found in polynomial time on a non-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.

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 premature 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 ants can build 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. Later we discuss this group of algorithms more elaborately. 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.

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. The current paper does not pretend to be complete. It would be interesting to carry out a more profound survey of heuristics, compare implementation complexity and accuracy of the different approximate algorithms. But this task cannot be easily accomplished because of the enormous bulk of information. We even did not touch upon such a prominent area for heuristic algorithms as planning and scheduling theory. But we hope that our work makes clear the extreme importance of heuristics in modern computer science.

References :

1. Siddique, N., & Adeli, H. (2015). Nature Inspired Computing: An Overview and Some Future Directions. Cognitive Computation, 7, 706–714. http://doi.org/10.1007/s12559-015-9370-8
2. What are the differences between heuristics and metaheuristics?. Available from: https://www.researchgate.net/post/What_are_the_differences_between_heuristics_and_metaheuristics [accessed Sep 7, 2017].
3. An introduction to heuristic algorithms by Natallia Kokash
4. http://www.ida.liu.se/~zebpe83/heuristic/
5. Stojanović, I., Brajević, I., Stanimirović, P. S., Kazakovtsev, L. A., & Zdravev, Z. (2017). Application of Heuristic and Metaheuristic Algorithms in Solving Constrained Weber Problem with Feasible Region Bounded by Arcs. Mathematical Problems in Engineering, 2017.
6. Kokash, Natallia. (2017). An introduction to heuristic algorithms. .