

Some Contributions to Computational Mathematics: Heuristic Approaches to Intelligent Systems

O. Ibidapo-Obe¹

University of Lagos, Nigeria

Abstract

This paper discusses some contributions to computational mathematics, outlining the major concepts and applications towards developing intelligent systems. Mathematics not only expresses relationships and the laws of nature but also computes real, relative numerical values that must have meaning and some useful interpretation. It is hereby shown how multi-disciplinary approaches to problems have contributed to the emergence of the modern intelligent systems. From classical methods of Control Systems, to the more recent theory of Simulated Annealing, Fuzziness, Neural Networks, and Genetic Algorithms to the evolution of self-organizing adaptive systems such as the Modern Economic Models or the intelligent Mobile Robots.

¹ Professor of Systems Engineering and
Acting Vice-Chancellor, University of Lagos

1.0 Introduction

There had been continuing vast advances in optional solution techniques for intelligent systems in the last two decades. This has been facilitated due to the fact that many practical combinatorial problems remain too large or too complex to be solved using the well known classical solution techniques. The heuristic methods offer a very viable approach. However, the design and implementation of problem specific heuristic can be a long and expensive process and the result is often domain-dependent and not flexible enough to deal with changes, which may occur over time. Hence, considerable interest is focused on general heuristic techniques that can be applied to a variety of different combinatorial problems. Most classical or traditional heuristic techniques require substantial problem specific input and are limited in their robustness. As a result, a tremendous amount of research effort has been applied in the last two decades, to correct these anomalies, which have yielded some new generation of intelligent heuristic techniques coupled with emerging intelligent computational methods such as fuzzy sets and logic, and neural networks.

Why Computational Mathematics

Closed form or generalized analytic solutions are desired for Mathematics problems. Whenever such is not feasible, approximate methods are employed. This could lead to further analytic and numerical methods that fully describe the optimal solution to the given problem. These descriptive and number crunching search techniques suitable for computer assistance constitute Computational methods.

Heuristics

Heuristics are the knowledge used to make good judgments, or strategies, tricks, or “rules of thumb” used to simplify the solution of problems (Patterson D. W; 1990). They include “trial and error” {experience based} knowledge and intelligent guesses/procedures established for domain specific problem solving. They are particularly suitable for ill-defined or poorly posed problems, poor models such as when there is incomplete data or time to solve a ‘large’ problem. Heuristics play an important role in search strategies because of the exponential nature of most problems.

Heuristics are knowledge used to make good judgments, or strategies, tricks, or “rules of thumb” used to simplify the solution of problems.

They help to reduce the number of alternatives from an exponential number to a polynomial number and, thereby obtain a solution in a tolerable amount of time.

Intelligent Systems

Intelligence is the ability to acquire, understand and apply knowledge, or the ability to exercise thought or reason. It also embodies knowledge and feats, both conscious and unconscious, which animate beings have acquired through study and experience.

(Artificial) Intelligent systems are thus machines and coded programs aimed at mimicking such feat and knowledge. Systems have been designed to perform many types of intelligent tasks. These systems can be physical systems like robots or Mathematical/computational systems such as scheduling systems which solve diverse tasks, systems used in planning complex strategies for the military and for business, in diagnosing medical diseases and so on.

Intelligent Systems are systems, whether physical of mathematical, biological or extra terrestrial, which mimic human intelligence in terms of knowledge and feats.

Computational Schemes, which mimic intelligence inherent in human, physical or biological systems, can be referred to as intelligent Computational Methods. Using this techniques and schemes in the solution of systems confers on such systems a measure of intelligibility and is thus referred to as Intelligent Systems. This presentation discusses the links and the dividing lines between some of such intelligent heuristic techniques.

2.0 Search Algorithms

The Optimisation Problem

Given a set $S \subset U$,

Where U is a universe of potential solutions to an optimization problem and S the set of solutions that satisfies required constraints in the problem.

Then given a totally ordered set W (often of Reals. \mathfrak{R}) and an objective function $f : U \rightarrow W$ an optimization problem is of the form.

$$\max f(p)$$

such that

$$p \in S \subset U$$

Often, the constraints are “priced out”, by defining $U \ni p \in U \mid S$ for every p ; is such that the function value $f(p)$ is so small as not to guarantee the maximum value.

Where

$$W = \mathfrak{R}, p \in U \setminus S \text{ is } \ni$$

$f(p)$ is a large negative number.

The optimization problem can then be written as:

$$\max f(p)$$

$$\ni p \in U$$

Directed or Random Search Techniques

Various techniques of optimization are in vogue. Among these, the directed or random search technique has generated tremendous interests in the last decade. Methods included in this category include: Random Search, Neighborhood Search, Tabu Search, Simulated Annealing Genetic Algorithms and a range of hybrid algorithms.

The General Search Paradigm

The General Search Algorithms is of the form:

```
General Search

{objective is to maximize  $f(p) \ni p \in U$ }

P,Q,R: multiset of solutions  $\subset U$ ,

initialize ( $P$ );

while not finish ( $P$ ) do
  begin

    Q := select ( $P$ )

    R := create ( $Q$ )

    P := merge { $P,Q,R$ }

  End

End {General Search}
```

Where

P = the initial pool of one or more potential solutions to the problem. Since P may contain multiple copies of some solutions, it is more appropriately called a multiset.

Q = is a *selection* from **P**

R = is *created* from **Q**

When a new solution is *created* either initially or by using the operator “*create*”, the function value, $f(p)$ is applied to determine the value of the solution

P is reconstructed from the penultimate pool of **P**, **Q** and **R** by the operator *merge*.

The process is repeated until the pool, **P** is deemed satisfactory.

Traditional Search Algorithms

i. The Random Search

The Random Search is a typical approach to optimization. In it **P** is a random set of solutions.

$$|P| \leq k \text{ for some } k \geq 1$$

The function *select* is an identity function; the function *create* randomly generates a number of solutions from U .

ii. Neighbourhood/Local Search

A neighbourhood of solution $u \in U$ is any solution lying in the neighbourhood (u) where,

$$\text{Neighbourhood: } U \rightarrow 2^U$$

Defines a set of solutions “near” to u

Generally these solutions differ from u by some small perturbation. There are other ways of defining neighbourhood.

Any Search Algorithm in which “Create” generates a subset of neighbours of Q is known as Local Search or a Neighbourhood Search Algorithm.

iii. Random Local Search

In a random local search, the function *create* generates a number of solutions from the neighbours of $Q (= P)$. The function *merge* merely delivers a set containing the solutions of greatest value from $P \cup Q$.

In a simple local search, *initialize* selects some starting set of solutions, $P_0 \subset U$ and assigns it to P . *Select* is often the identity function but, some times Q is selected according to some criterion, e.g. to comprise a limited number of solutions of greatest value. Then *create* generates a subset, **R**, of

$$\text{Neighbourhood } (Q) = \cup \{ \text{neighbourhood}(q) \mid q \in Q \}$$

The function *merge* selects a subset of the solutions in $P \cup R$ preferring those of greater value. In the simplest form *merge* selects the subset of the solutions in $P \cup R$ which are of maximum value subject to some limitation on cardinality. The *finish* function can

ensure termination after some given number of iterations or when the value of the best solution in \mathbf{P} is not being (significantly) improved.

Examples of Neighbourhood Search

1. Greedy Algorithms or Hill Climbing Algorithms

Here, $|P| = 1$ and moves from a single solution value $v \in W$, to a neighbourhood solution value, $w \in W$ only when $w > v$. The algorithm generally terminates when there is no such neighbour with a greater value.

2. Steepest Ascent Algorithms

This is a neighbourhood search algorithm, which always selects the neighbour of v of greatest value.

3.0 Tabu Search

Tabu Search is one successful variant of the Neighbourhood search paradigm designed to avoid the problem of becoming trapped in a local optimum.

The Tabu Search Paradigm is as below:

Tabu Search

{objective is to maximize $f(p)$ such that $p \in U$ }

P,R: sets of solutions $\subset U$,

Tabu; set of rules of type, each rule of type $U \rightarrow \{\text{true, false}\}$

initialize (P);

initialize (Tabu); {very often to Φ }

while not finish (P) **do**

begin

R: = create (P , Tabu) :

Tabu: = update (Q)

P: = merge {**P**,**R**}

End

End {Tabu Search}

The simplest application of Tabu search has:

- **P** as a singleton set
- The function **create** delivers the set of all non-Tabu neighbours, N , of {**P**} i.e.

$$N = \{n \mid n \text{ is a neighbour of } p \text{ and } f(p) \text{ is false } \forall f \in \text{Tabu}\}$$

- The function **merge** merely greedily selects the solution of the greatest value from $P \cup R$

In practice, P usually contains more than one element and may include the best solutions, found to date. The sort of rules often found in Tabu list include:

1. V is Tabu if v is in **P**
2. V is Tabu if v has been in **P** recently
3. V is Tabu if v is obtained by changing a bit changed in the previous iteration.
4. V is Tabu if v is obtained by changing a bit recently changed.
5. V is Tabu if $f(v) = f(w)$ for some $w \in p$

The difficulty in Tabu search is in constructing the set of rules. Considerable expertise and experimentation is required to construct the rules and to ensure its dynamic nature is correctly controlled. If the expertise is available, the resulting search can be very efficient. Aspiration criteria are often included to help the Tabu search in not being too restrictive. These criteria are rules, which say that certain moves are to be preferred over others. Some form of expert rules may also serve as Tabu search rules. Each rule may have an associated weight, negative if Tabu and positive if an aspiration. The combine set of rules thus associates a weight to each neighbour. A large positive weight suggests it is a desirable move while a large negative weight suggests it can be discounted.

Tabu Search has found applications to real world problems such as:

- Packing and Scheduling Problems
- Travelling Salesman and Vehicle Routing
- Telecommunications
- Graph Problems
- Quadratic Assignment

4.0 Simulated Annealing

The Simulated Annealing technique is essentially a local search technique in which a move to an inferior solution is allowed with a probability that decreases as the process progresses, according to some Boltzmann-type distribution.

The inspiration for Simulated Annealing approach to optimization is the law of thermodynamics, which states that at temperature, t , and the probability of an increase in energy of magnitude, δE , is given by:

$$P(\delta E) = \exp.(-\delta E / kt)$$

Where k is the physical constant known as the Boltzmann's constant. The equation is applicable to a system that is cooling until it converges to steady "frozen" state. The system is perturbed from current state and the resulting energy change δE is calculated. If the energy has decreased, the system moves to the new state, otherwise the new state is only accepted with the probability given above. The cycle can be repeated for a number of iterations at each temperature and subsequently reduced and the number of cycles repeated for the new lower temperature. This whole process is repeated until the whole system freezes to its steady state.

We can associate the potential solutions of an optimization problem with the system states, the cost of a solution corresponds to the concept of energy and moving to any neighbour corresponds to a change of state. A simple version of the Simulated Annealing Paradigm is of the form:

```

P: =  $P_0$  where  $P_0$  is some initial solution

Temp: =  $\text{temp}P_0$  where  $\text{temp}_0$  is some initial temperature

While not finish (p) do

    For  $i$ : = 1 to  $n$  do

        Begin

        Randomly select  $p'$ , a neighbour of  $p$ .

        Improvement: =  $f(p') - f(p)$ 
        ;
        If improvement  $> 0$  then  $p := p'$ 

            Else

                Begin

                Generate  $x \in U[0,1]$ ;

                If  $x < \exp(-\text{improvement}/\text{temp})$  then  $p := p'$ 

                End {else}

            End {for};

         $t := \text{reduce}(t)$ 

    End {while}

```

It can generally be said that the Simulated Annealing paradigm varies from hill climbing in the function, *merge*. In Simulated Annealing the function *merge* selects a subset of the solutions in $P \cup R$ favouring those with highest value as in the paradigm above. Initially, there would be no strong preference for the higher valued solutions but as the temperature parameter reduces, so this favouring would become stronger.

5.0. Genetic Algorithms

Genetic Algorithms are search techniques, based on an abstracted model of Darwinian evolution. Fixed length strings represent solutions over some alphabet. Each such string is thought of as a “chromosome”. The value of the solution then represents the “fitness” of the chromosome. The concept of “survival of the fittest” is then used to allow better

solutions to combine to produce offspring. The Genetic Algorithm paradigm follows closely the same search concept exploited in Tabu Search and Simulated Annealing. The paradigm is:

```
GA
{objective is to maximize  $f(p)$  such that  $p \in U$ }
P,Q,R: multi-set of solutions  $\subset U$  ;
Initialise (P);
While not finish (P) do
    Begin
        Q: = select (P)
        R: = create (Q)
        P: = merge (P, Q, R);
    End
end
```

Here the operators *select*, *create* and *merge* correspond to the operators select, reproduction crossover.

The iteration loop of a basic Genetic Algorithm hence looks like,

Procedure GA

Begin

Generate initial population, $P(0)$; $l = 0$

Evaluate Chromosomes in $P(0)$;

Repeat

$l = l + 1$

Select $P(l)$ from $P(t = l)$;

Recombine chromosomes in $P(l)$ using genetic operators

Evaluate chromosomes in $P(l)$;

Until termination condition satisfied.

end

Genetic Algorithms Operators

There are three basic operators found in every genetic algorithm. These are:

- i Reproduction: This operator allows individual strings to be copied for possible inclusion in the next generation.
- ii Crossover: a genetic operator applied to two randomly selected parent solutions in the mating pool to generate two offspring solutions. (Some algorithms do not employ the crossover operator. They are rather known as evolutionary algorithms than genetic algorithms).
- iii Mutation Bit-Wise change in strings at randomly selected points.

Examples

Crossover: This is a genetic operator applied to two randomly selected parent solutions in the mating pool to generate two offspring solutions. Crossover operation is not performed on all pairs of parent solutions selected from the mating pool. Crossover is performed according to a probability, p (usually $p=0.6$). If GA decides by this probability not to perform crossover on a pair of parent solutions they are simply copied to the new population. If crossover is to take place, then one or two random splicing points are chosen in a string. The two strings are spliced and the spliced regions are mixed together to create two (potentially) new strings. These child strings are then

Fundamental Differences with Classical Optimization Methods

For a number of reasons, Gas differs from classical search and Optimization methods.

1. Gas work with a coding of decision variables a matter, which is very uncommon in classical methods. Coding discretises the search spaces and allows Gas to be applied to both discrete and discontinuous problems Gas also exploit-coding similarities to make a faster and parallel search.
2. Since no gradient information is used in Gas, they can also be applied to non-differentiable functions. This makes Gas robust in the sense that they can be applied to a wide variety of problems. (Chakroborty, p. et at.; 1995).
3. Unlike many classical optimization methods, Gas work with a population of points. This increases the possibility of obtaining the global optimal solution even in ill-behaved problems (Goldberg, D.E.; 1989).
4. Gas use probabilistic transition rules, instead of fixed rules. In early GA interactions, this randomness in GA operators makes the search unbiased toward any particular region in the search space and has an effect not making a hasty wrong decision. Use of stochastic transition rules also increases the chance of recovering from a mistake. (Chakroborty, P, et al.; 1995).
5. Gas use uses only pay-off information to guide themselves through the problem space. Many search techniques need a variety of information to guide themselves. Hill climbing methods require derivatives, for example: The only information a GA needs is some measure of fitness about a point in the space (sometimes known as objective function value). Once the GA knows the current measure of “goodness” about a pont, it can use this continue searching for the optimum.
6. Gas allows procedure-based function declaration. Most classical search methods do not permit such declarations. Thus where procedures of optimization need to be declared Gas proves a better optimization tool.

The coding of decision variables n Gas also make it proficient in solving optimization problems involving equality and inequality constraints.

Fuzzy Sets and Fuzzy Logic

Fuzzy sets are due to Zadeh, L (1965) who introduced the concept of a set that admits elements with some degree of (partial) membership. Fuzzy sets aim at representing human-originated knowledge, and more specifically that part of articulated knowledge that refers to numerical universes of discourse; it proposes a natural flexible interface between symbols and numbers. Fuzzy logic is a multi-valued logic that allows intermediate values to be defined between conventional evaluations, such as yes/no, true/false, black/white etc. It has the capability of bridging the gap between articulated linguistic descriptions and numerical models of systems and generating one from the other.

Fuzzy set and fuzzy logic concepts have been used for improvement in the performance of a lot of heuristic search algorithms. Notable among these is the hybrid fuzzy genetic algorithms, which are simply implementation of genetic algorithm over fuzzy rules

Neural Networks

Neural networks model the architecture and operations of the neurons in the brain for problem solving. It is a system whereby the core features of the neural system is extracted and simulated for dealing with problems that are amenable to iterative solutions by continually feeding the output at any stage as an input for the process.

Some Applications of Heuristics to Intelligent Systems

At the University of Lagos, some of these techniques have been successfully applied to model and simulate some intelligent systems. These include Application of Genetic Algorithms to Bi-modal Transport Scheduling problem, (Ibidapo-Obe, O, and Ogunwolu, F. O). In this application, Genetic.

Algorithms is applied using a bi-level programming approach to obtain transit schedules for a bi-modal transfer station in an urban transit network with multiple dispatching stations. The arrival rates of passengers are captured as fuzzy numbers in order to confer intelligibility in the system. The model results in comparably better schedules with better levels of service (LOS) in the transit network (in terms of total waiting and transfer times for passengers) than is obtained otherwise. Another application is on An intelligent path planner for autonomous mobile robots (Ibidapo-Obe, O. & Asaolu, O. S.). For this robotic motion planner, a real-time optimal path planner was developed for autonomous mobile robots navigating within a workspace with both static and roving obstacles. With the introduction of arbitrarily moving obstacles, the dynamic obstacle avoidance problem was re-cast into a dynamic graph-search. The instantaneous graph is made of the connected edged of the rectangles boxing the ellipses swept swept out by the robot, goal and moving obstacles. We are presently also looking at the problems of machine translation and intelligent games.

Conclusions

Mathematics is about modeling and solving real life problems. Optimal solutions are often obtained via intelligent search of computational techniques. There are varieties of new solution approaches being developed from multi-disciplinary perspectives.

References

1. Chakroborty, P., Kalyanmov D., and Subrahmanyam, P. S.: “Optimal Scheduling of Urban Transit Systems Using Genetic Algorithms”. Journal of Transportation ; Vol, 121, No. 6 November/December, 1995.
2. Rayward-Smith, V. J., 1995: Applications of Modern Heuristic Methods, Alfred Walter Limited Publishers in association with UNICOM pp. 145 – 156.
3. Goldberg, D. E.; 1989: Genetic Algorithms in Search, Optimization and Machine Learning, Reading, M. A. Addison-Wesley.
4. Ibidapo-Obe, O. and Ogunwolu, F. O.; 2001: An Optimal Scheduling of a bi-modal Urban Transit System Using Genetic Algorithms An unpublished research work at the Department of Systems Engineering, University of Lagos, Lagos, Nigeria
5. Ibidapo-Obe, O., and Asaolu, O. S.; 2001: An Optimal Intelligent Path Planner for Autonomous Mobile Robot. An unpublished research work at the Department of Systems Engineering, University of Lagos, Lagos, Nigeria.