Peter the Great St. Petersburg Polytechnic University

Computer Science

Cognitive Multi-agent Systems

ANY-TIME ALGORITHM
TO SOLVE COGNITIVE TASKS

Hyper Heuristic Any-Time Algorithm Implementation

Post Man Problem

Samet KAYA

1. Introduction

Incoming computer science, an algorithm that is a time algorithm can return a valid solution for one even if the problem ends up interfering. The algorithm is expected to continue to work so long to find good and better solutions.

Most algorithms run completion: they give a single answer after performing some fixed amount of calculation. In some cases, however, the user may want to end the algorithm before it is completed. The amount of calculation that may be important, for example, and the calculation resources may need to be allocated. Most algorithms provide completion or no useful solution information. Anytime algorithms are able to return a partial answer, however it depends on the amount of calculation who managed to make quality. The answer produced by time algorithms is an estimate of the correct answer.

A time algorithm can also be called an "interruptible algorithm". They are different from the pre-announced contract algorithms of their watch; In a time algorithm, a process can declare only termination.

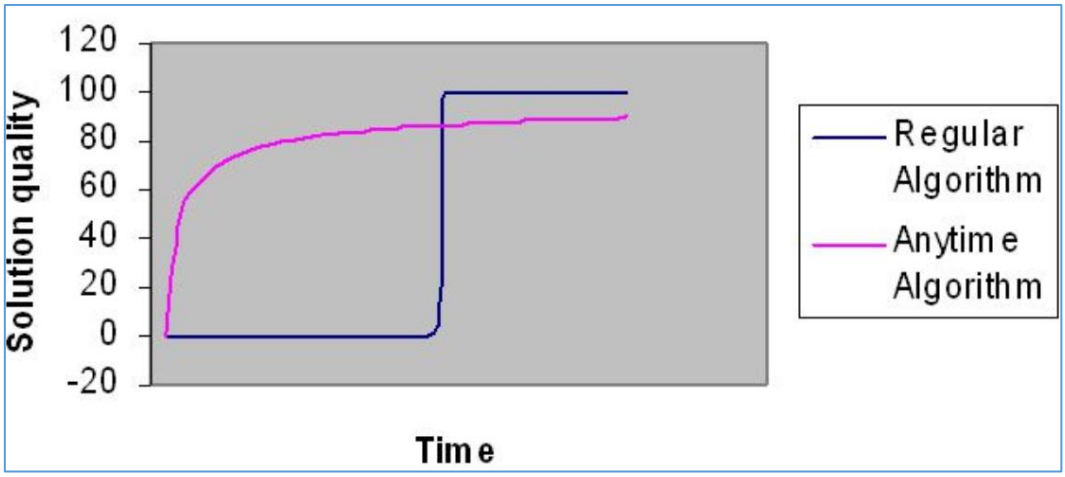


Figure 1: Difference Between Normal Algorithm and Any-Time Algorithm

The ability to make better quality results in return with the time of reversing their smart systems, which is the goal of time algorithms. They also had to be flexible in time and resources. Because it is important that artificial intelligence or AI algorithms can take a long time to complete the results. This algorithm is designed to complete a short amount of time. In addition, this system is designed to better understand agents and how collaboration is dependent and constrained. An example is Newton-Raphson applied iteration to find the square root of a number. Another example that uses time algorithms is to set a target where there are trajectory problems; The object is moving in space while waiting for the algorithm to finish and give an apparent accuracy about one answer, even if given early.

What makes algorithms unique is their ability to return many possible results for any input. It uses many well-defined quality measures to track its progress within a time algorithm for problem solving and distributed computing resources. It keeps searching for the best possible answer with the amount of time it is given. This may not work until it is complete and may increase the response if allowed to run longer. This is often used for large set of decision

problems. This would not usually provide useful information if it is not allowed to finish. Although it may look similar to this, the dynamic programming is that the difference is fine tuning thanks to a rather sequential more random fit.

Anytime algorithms have been designed so that it can be said to stop at any time and return the best result it has ever found. This is why this is called an interruptible algorithm. If they are given more time, they always maintain the final result, so that they can continue where they left off to get a better result.

2. Problem

Problem has three main location post office, home and delivery address. A postman must start from post office and after visiting all delivery addresses can go it's home. The problem is similar to the traveling salesman problem. It just doesn't return to the starting point again. As the number of delivery addresses increases, the problem becomes difficult to solve.

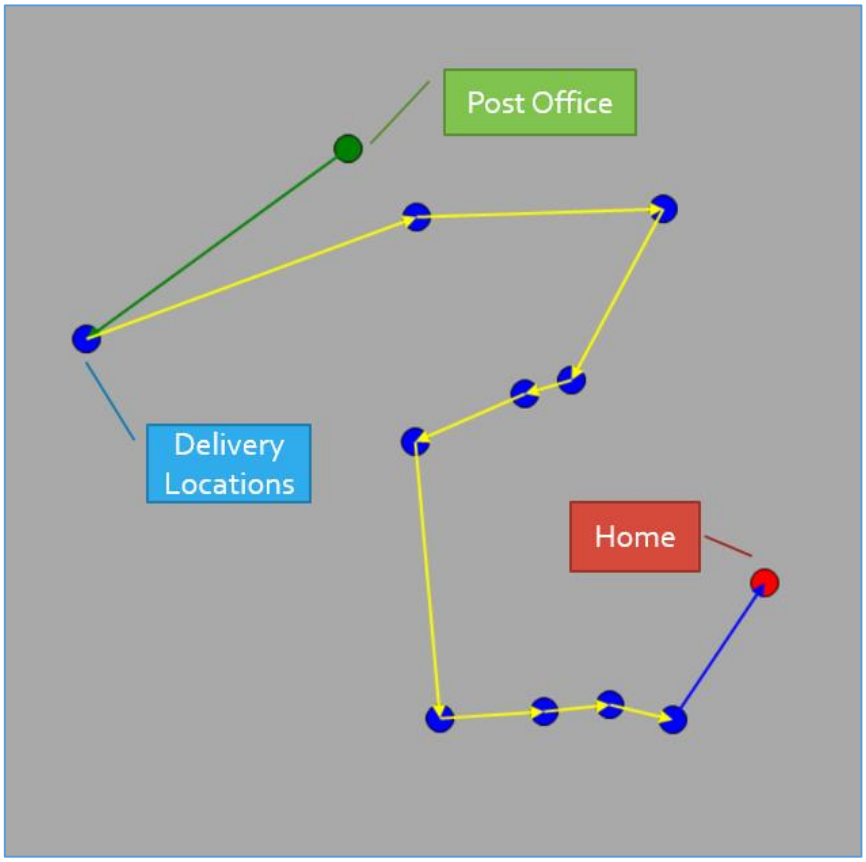


Figure 2: Postman Problem

The solution to the problem can be long with classical algorithms. As the number of delivery addresses increases, the number of permutations increases exponentially. This extends the search time for solutions with traditional algorithms.

It is an heuristic problem solving technique in computer science. He does not care whether the accuracy of the result can be proved, but generally he gets close to good solutions. Intuitive algorithms, on the other hand, are algorithms that decrease the solution time by giving up searching for the best solution in order to become more efficient during the transition period.

Heuristic algorithms do not guarantee that they will find the best result, but they guarantee that they will get a solution within a reasonable time. Generally, they reach the solution which is close to the best, quickly and easily.

As an example of heuristic search algorithms;

- A * search (A star)
- Beam search
- Hill climbing algorithm
- Best first search
- Greedy best first search
- Simulated Annealing algorithm
- Backtracking

In other words, heuristic; known as functions that calculate the cost of the shortest path from one node to another node.

3. Hyper-Heuristic Genetic Algorithm

A hyper-heuristic is a heuristic search method that seeks to automate, often by the incorporation of machine learning techniques, the process of selecting, combining, generating or adapting several simpler heuristics (or components of such heuristics) to efficiently solve computational search problems. One of the motivations for studying hyper-heuristics is to build systems which can handle classes of problems rather than solving just one problem.

There might be multiple heuristics from which one can choose for solving a problem, and each heuristic has its own strength and weakness. The idea is to automatically devise algorithms by combining the strength and compensating for the weakness of known heuristics. In a typical hyper-heuristic framework there is a high-level methodology and a set of low-level heuristics (either constructive or perturbative heuristics). Given a problem instance, the high-level method selects which low-level heuristic should be applied at any given time, depending upon the current problem state (or search stage) determined by features.

3.1. Notion of Natural Selection

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Five phases are considered in a genetic algorithm.

- Initial population
- Fitness function
- Selection
- Crossover
- Mutation

3.2. Initial Population

The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem you want to solve.

An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution).

In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.

3.3. Fitness Function

The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

3.4. Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation.

Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

3.5. Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.

3.6. Mutation

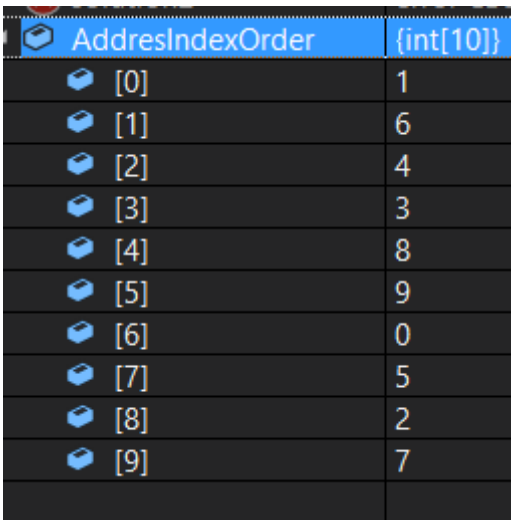
In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.

4. Project

```
int[] AddresIndexOrder;
```

```
double distance;
```

Fitness function is the total distance of the sequence calculated by the two-point distance formula.



Mutations

Inversion:

- Random two different index are selected
- Swap their contents.

Adjacent:

- Random one index is selected
- Swap contents with it's next neighbour.

Reinsertion:

- Random one index is selected.
- Pop it from list.
- Push it in random index.

Local Serachs

NextDescent:

- Select random one index
- If its second neighbor is closer than the first one, swap them.
- Continue until there is an improvement or full turn.

Hill Climbing:

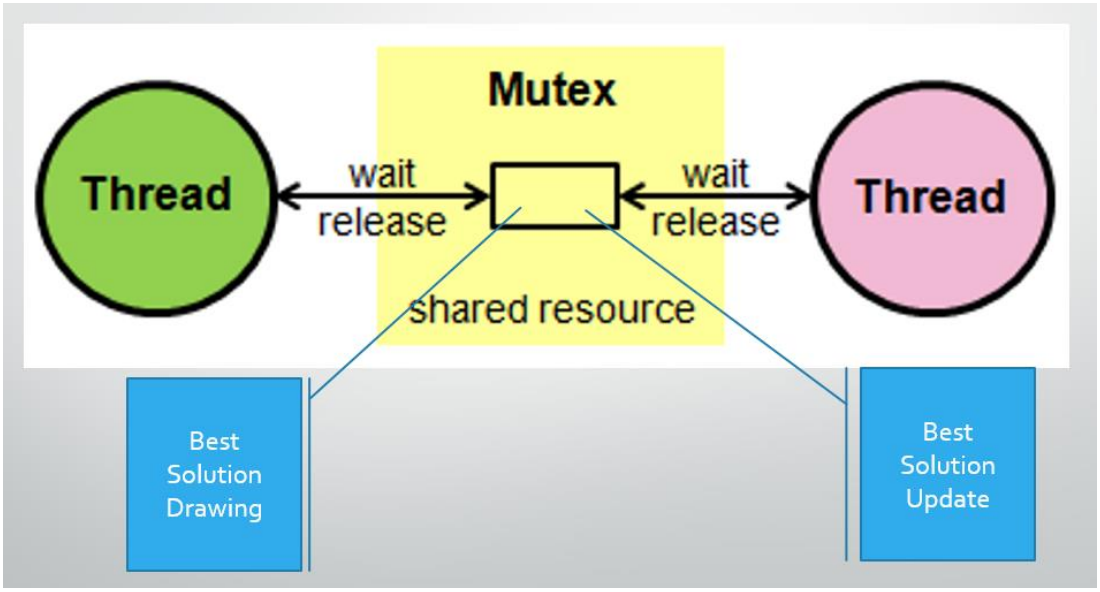
- Randomly choose two indexes.

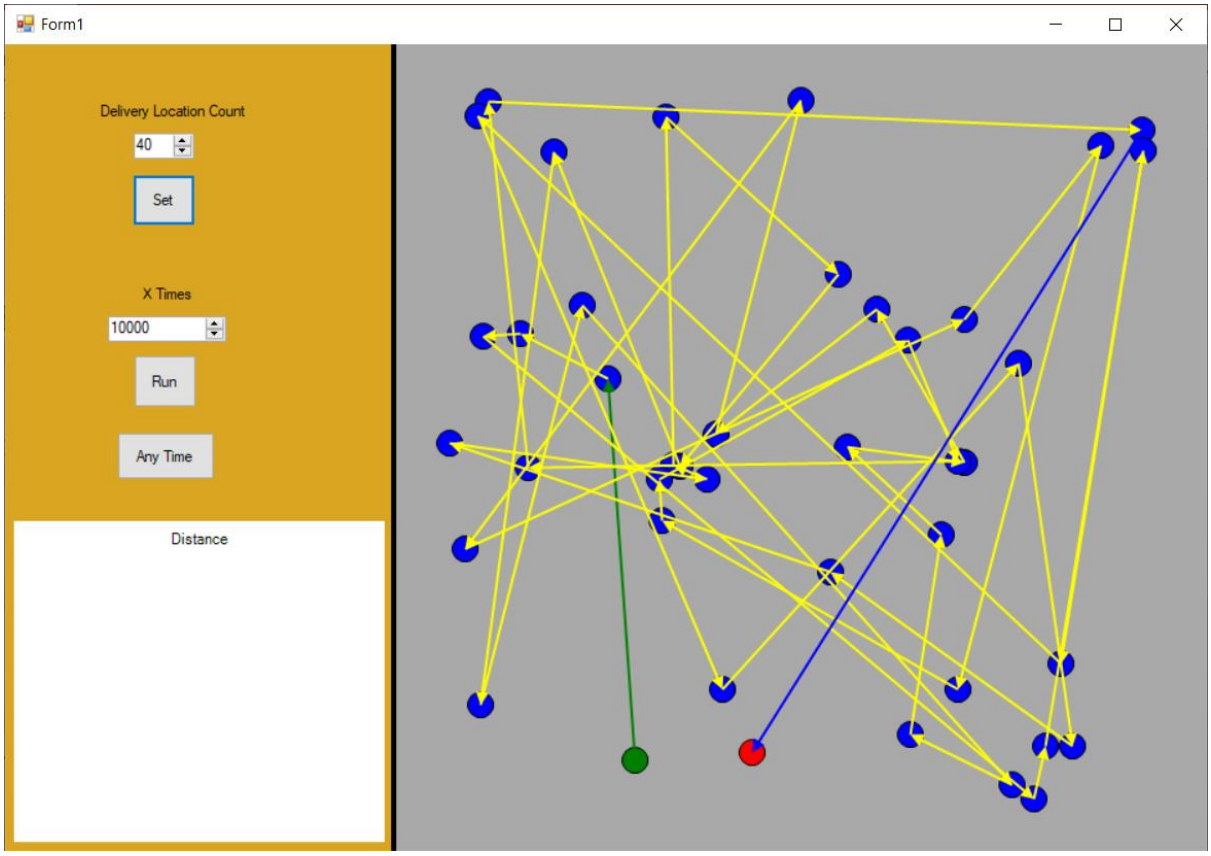
- Swap them.
- Continue until there is an improvement or all binary matches are tried

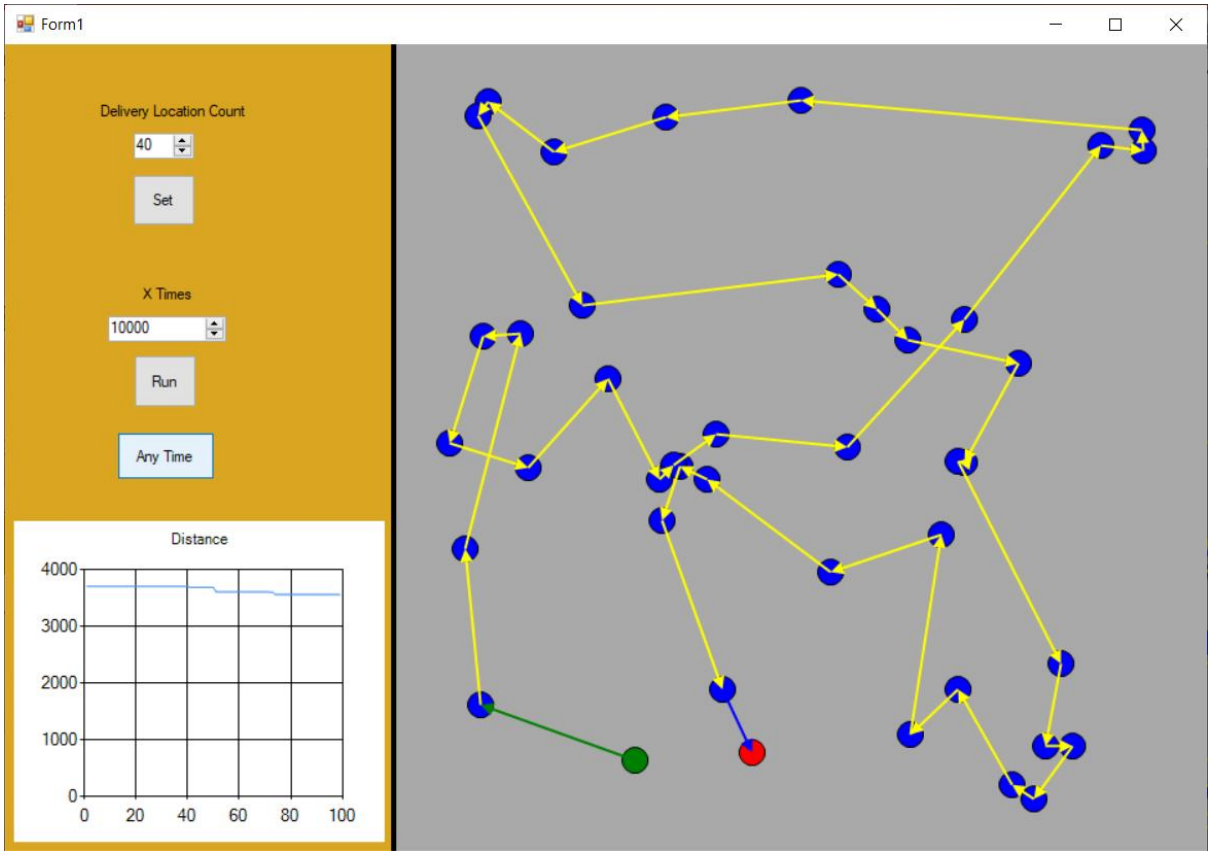
Crossovers

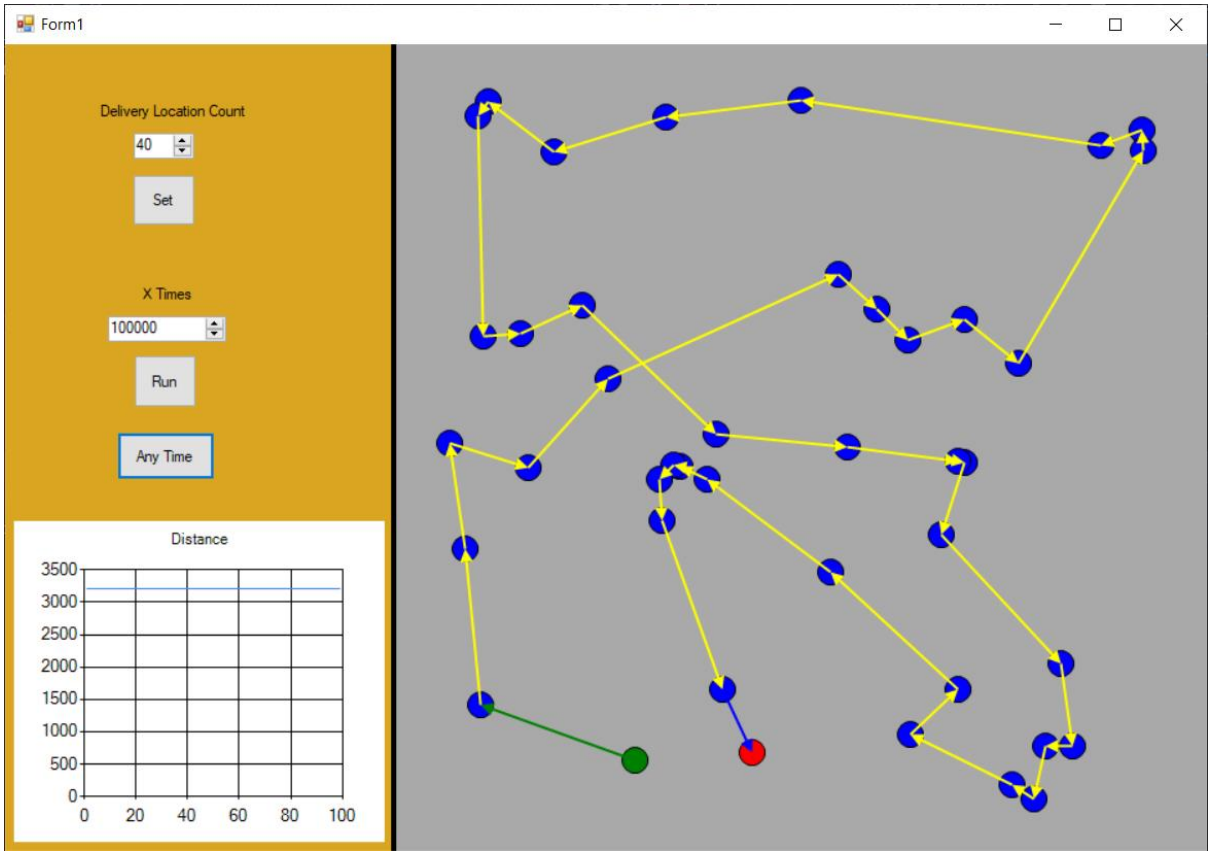
- **Ordered:**
  - Pick two cut point and paste it on new child.
  - Fill child from other parent orderly numbers which not in cut genes
- **Cycle:**
  - Pick random index from parent1
  - Get number inside parent2 index
  - Use the number as index on prarent1
  - Continue until parent 2 number index content is equal to parent 1 start content

AnyTime Implementation




Form1
Delivery Location Count
40
Set
X Times
10000
Run
Any Time
Distance


Form1
Delivery Location Count
40
Set
X Times
10000
Run
Any Time
Distance
4000
3000
2000
1000
0
0
20
40
60
80
100


Form1
Delivery Location Count
40
Set
X Times
100000
Run
Any Time
Distance
3500
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100