J. Inst. Eng. India Ser. B
DOI 10.1007/s40031-014-0182-0

ARTICLE OF PROFESSIONAL INTERESTS

Smell Detection Agent Based Optimization Algorithm

Vinod Chandra S. S.

Received: 10 May 2014 / Accepted: 29 December 2014

Abstract In this paper, a novel nature-inspired optimization algorithm has been employed and the trained behaviour of dogs in detecting smell trails is adapted into computational agents for problem solving. The algorithm involves creation of a surface with smell trails and subsequent iteration of the agents in resolving a path. This algorithm can be applied in different computational constraints that incorporate path-based problems. Implementation of the algorithm can be treated as a shortest path problem for a variety of datasets. The simulated agents have been used to evolve the shortest path between two nodes in a graph. This algorithm is useful to solve NP-hard problems that are related to path discovery. This algorithm is also useful to solve many practical optimization problems. The extensive derivation of the algorithm can be enabled to solve shortest path problems.

Keywords Nature-inspired computing · SDA · Smell spots · Shortest path algorithm

Introduction

Nature has mystified man since time immemorial. Several natural phenomena have been adapted in computational science to create optimization algorithms. Evolutionary process of natural selection has created many emergent behavioural patterns in organisms that have optimizing overtones [1, 2]. These have been separately identified and used in nature-inspired algorithms.

Vinod Chandra S. S. (✉)
Computer Centre, University of Kerala, Senate House Campus, Palayam, Thiruvananthapuram 695034, Kerala, India
e-mail: vinodchandrass@gmail.com

Published online: 07 February 2015

The field of nature-inspired computing has been endowed with many optimizing algorithms, each distinct in the source of inspiration as well as in the methodology by which it has been mapped into the computational domain. Most of the algorithms designed till now principally deal with an agent or a multitude of agents that exhibit collective intelligent behaviour. This is most significantly manifested in the first such algorithm, ant colony optimization (ACO) algorithm [3].

Dogs are considered as one of the earliest animals to be domesticated by man. They are used as valuable aids in hunting. Their superior sense of smell enabled man in tracing the location of prey easily. Even today, dogs are employed by crime detection forces to trace the trail of criminals from the scene of crime. Such natural process can be used to make solutions to the selected optimization problems. There are many optimization problems that cannot be solved conventionally and are asymptotically NP-hard. Novel algorithms are required to solve such problems. In such contexts nature-inspired phenomena like animal behaviour can be used to make solutions to computational optimization problems [4].

An algorithm, which adapts the natural process of 'sniffing' or smell trail detection, is the theme of this paper. The problem to be solved is considered as a surface with smell trails and computational agents inspired from dogs are used to detect an optimized path, which will constitute the solution. This algorithm is extended on the top of the shortest path finding technique.

Sniffing: A Nature Sensory Mechanism

Nature has endowed organisms with various sensory mechanisms for their survival [5]. Most of the animals have

olfactory mechanisms for detecting chemical presence in air, along with other sensory organs [6]. Olfactory membrane is usually concentrated in a dedicated organ commonly called the nose or snout. Dogs, especially species like the bloodhound have exceptional concentrations of olfactory cells, providing them a much superior capability in detecting and using smells. For example, man only has a primitive olfactory sense that helps him to detect any new smell originating in his environment. He cannot distinguish the different smells generated by organisms unlike a dog, which can detect and memorise the smell signatures of all beings.

Man sheds millions of cell fragments through sweat from the skin. These fragments and bacteria constitute the scent of a man. Dogs have highly tuned receptor sites in their snouts that can lock on to this scent and thus enables them to identify the path undertaken from the origin of scent. On the trail, they also refresh their olfactory cells by taking quick sniffs and then emptying their noses to give the olfactory cells a rest. Another curious behaviour of dogs is that they urinate in different spots to mark territory. Due to the strong and distinctive smell of the urine, other dogs will be able to identify it as occupied territory.

These two peculiarities of dogs have been used to create smell detection agents (SDA) that can detect smell trails in the environment and also mark the path they undertake. This evolutionary technique can be used as a computational model for solving selected optimization problems.

Computational Model

The concept of dog's path tracing behaviour is used to design suitable environments to solve optimization problems. This nature-inspired technique can be plotted in coordinate geometry. The basic environment is a Cartesian rectangular plot with specified extremities defining the area. These values may be varied based on the specific computational constraints of the problem. All points in the plot are not traversable, but selected random points (sampling of points) that can be visited by the SDAs. These points are called smell spots that helps in focusing the problem solving to a subset of points. Each of these smell spots are stored by two parameters. One is smell trail or value from a target or destination. Other is signature value of any SDA that has visited or marked a smell spot. Figure 1 illustrates this idea that consists of source, destination, smell spots and smell radius for a SDA to travel.

Each SDA holds two parameters. One is their assigned signature value that can be used to mark smell spots. Other is the radius value that indicates their olfactory capability. Different SDAs can have different smell detection capability. This value varies in radius of the circular area that

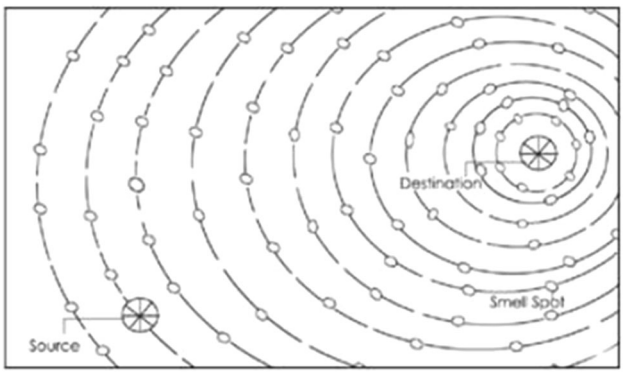

Fig. 1 Graphical representation of smell surface with smell spots

can be surveyed or sniffed by a particular SDA at a point. That is, each SDA can have different smell detection capability values. The smell spots are distributed over the plotted area with different values. This may lead a SDA to sniff and reach to a particular point. The above idea can be implemented as a data structure that is useful in algorithm design.

Data structure of the SDA is

$$D_{SDA} = (s, F, A)$$

where s is the data type used, either integer or real values. F is a set of functions—'search', 'split an SDA'. A is a set of flags or constants—'local maxima', 'start/stop'. Data structure for smell spot is

$$D_{ss} = (v, F, A)$$

where v is a data type (real/integer); F is a set of functions (nearest spots, distance measure); and A is a set of axioms (visited/not visited, radius identification, smell distribution).

Two parameter values can be used; the number of agents and the number of smell spots that are varied according to the computational resources available to solve an optimization problem. These parameter values are measured in numbers (either positive integer or real numbers) based on the optimization problem to be solved. For example, in shortest path problem the values are real numbers. The data structure is a triplet that can vary with respect to the problem.

SDA Algorithm

The considered problem has a source and destination with multiple paths traversal. When number of paths increase, the complexity of problem solving also increases. The problem solved by this algorithm is represented as a path found by multiple SDAs starting from a source traversing through the set of smell spots and reaching the target. A smell spot is marked by an SDA as its territory when it is

visited and hence the other SDAs cannot visit the marked spots. This ensures that distinct paths are obtained from the SDAs. Multiple paths can be obtained by replication of the SDA when it is struck with the choice of multiple equally optimum paths. The replication of the SDA can be a solution for this. This would entail obtaining paths that may overlap and hence ensure that any possible 'local minima' are avoided.

The environment is initialized by randomly selected smell spots and assigning smell values inversely proportional to the Cartesian distance between the source and the destination. Two data structures are used for SDAs and smell spots. All agents are initialized with signature values and radius values that are placed at the starting location. Every agent is iteratively allowed to survey the area within its radius to select an unmarked smell spot with a high smell value to visit. The SDAs are moved to chosen spots and the step is repeated until the SDAs reach the destination. It is assumed that the SDAs have different smell detection capacity, which makes an effective search from one smell point to another. Also, let us assume that a smell spot is 'marked' when an SDA is visited, in order to optimize this algorithm. The best path from the paths obtained by a SDA can be computed. The basic steps of the proposed technique are given in Algorithm 1.

After N_3 SDAs have reached the destination, choose the best path from the obtained paths as a solution state of the problem. No two SDAs can choose the same path. When a SDA visits a smell spot, it is marked so that other SDAs need to visit the same smell spot. The cost and trace of each path can be computed by a data structure incorporated within the SDA algorithm. This idea can be used in multi source–destination problems.

Corollary

Scope of the algorithm can be extended to solve harder computational problems. In order to avoid overlapped paths the smell spot is marked as 'visited' when a SDA has reached it. This restriction is used to optimize the algorithm. If a SDA has no further movement the scope is limited to a 'local maxima'. In such cases, movement of SDAs can 'stop'. This is mentioned as a function while designing the data structure. Suppose a SDA may have more than one nearest smell spot with same value, then the SDAs can also be modified for replication or bifurcation on reaching an impasse with two equally good nodes to move to in a single step. The smell capacity of each SDA is a dynamic variable that increases the computational dynamism of each agent. This capacity can be incorporated into the SDA data structure for better optimization path detection.

The proposed algorithm is useful for many optimization problems. For example, when a database crash occurs due to some errors, we need to roll-back the database. We require an optimum path in order to restore the database to its last stable state. In such problems, this algorithm is suitable for an optimized solution. Shortest path identification is an example that shows the proof of optimization by the SDA algorithm.

Shortest Path Problem

Shortest path problem involves the computation of a path in a weighted graph between two vertices such that the chosen path has minimum combined weight [7, 8]. The

Algorithm 1 SDA Algorithm

Let

N_1: Number of SDAs

N_2: Number of smell spots

N_3: Number of SDAs that can reach the destination

1. Initialize the agents with natural numbers as signature indices and radii value in the inverse order, so that the SDA is considered in the iteration that have highest radius
2. Randomly select N_2 points within the extremities of the Cartesian plot as smell spots

Assign the smell value 's' to each smell spot,

$s = 1/(a \pm b \times d)$

Where 'cl' is the Cartesian distance between the smell spot, the destination and 'a' and 'b' are proportionality constants

3. Initialize each agent to the source point
4. For every agent from 1 to N_1

4.1 Choose the unmarked point (within the radius) from a set of smell spots that has a highest smell value

4.2 Move the SDA to the point by marking the value of SDA's signature in smell spot's index

5. Repeat step 4 until the destination is reached
6. Count the SDAs that reached the destination and assign to N_3

graph may be directed, undirected or mixed. An estimated path is the set of consecutive nodes. The algorithms consider all the paths between the predetermined nodes. From these nodes the optimised path can be computed. This problem seeks to mathematically model real word applications like logistics management and route planning.

Many algorithms are used to solve shortest path problem like Dijikstra's algorithm [9], Bellman-Ford algorithm [10] and A* algorithm [11, 12]. These algorithms follow a linear programming approach towards solving a problem. SDA algorithm employs a parallel methodology for resolving the shortest paths. That is, multiple agents simultaneously find independent probable short paths. The algorithm can also be extended for similar derivative problems like road network problem and travelling salesman problem.

The solution to shortest path problem can be achieved by the proposed SDA algorithm. The computation is performed in a parallel manner by multiple SDAs. Graph of the shortest path problem is considered as a surface of the SDAs. Each SDA seeks a path towards the destination and all such paths are disjoint, resulting in multiple unique paths. The best path is chosen from these paths. Each node in the graph is considered as a smell spot with a smell value, relative to the destination.

Initially, SDAs are placed in the source and it starts to seek adjacent nodes. When a SDA reaches the smell spot, it is marked as 'visited'. Each SDA iteratively seek the destination independently. Disjoint paths are obtained after the completion of the algorithm. The best path can be selected by comparing all the paths.

Working of the Algorithm

First, smell values with respect to the destination is assigned. The constant 'a' is the initial smell value assigned and b is the proportional decrement constant.

These values can be considered as real numbers. Adjacent nodes of the destination are assigned smell values that are inversely proportional to the weight value. The nodes adjacent to current node are assigned smell values and so on. The agents are initialized in the source node and iteratively seek the unmarked adjacent node with the highest smell of the destination and move to the node, marking the territory; until the destination is reached. Basic steps of the shortest path algorithm derived from the SDA algorithm is listed in Algorithm 2.

The choice of unmarked node with highest smell in each step ensures that the most optimized paths are obtained by each of the SDAs. The function for this choice can be implemented by a simple comparison among the unmarked

Algorithm 2 Shortest path algorithm

N_1: Number of SDAs

N_2: Number of nodes in Graph

G: Graph represented as Adjacency matrix of dimension $N_2 \times N_2$

S: Index of source node

D: Index of destination node

The basic steps of the proposed algorithm are:

Let

N_1: Number of SDAs

N_2: Number of smell spots

N_3: Number of SDAs that can reach the destination

1. For $i = 1$ to N_1

$D_{SDA}.S = i$

2. For $i = 1$ to N_2

$D_{SS}.V = 0$

3. Assign smell value of spot as:

$D_{ss}.s = 1/(a \pm b \times p')$,

where "a" and "b" are proportionality constants—'a' is the initial smell value assigned and 'b' is the decrement constant. 'p′' is the effective distance between the smell spot and the destination node

4. Initialize all SDAs to S For $i = 1$ to N_1

$D_{SDA}.current = S$

5. For every SDA 1 to N_1, update each SDA to the next unvisited node with highest smell value

6. Repeat step 5 until the SDAs reaches the destination

7. Count the SDAs that reached the destination and assign to N_3

nodes itself. The SDAs will be marking each visited node with their signature. When the algorithm is completed, a labelled graph with the traversed paths of the SDAs is obtained.

Data Structures

There are two data structures in order to redesign the SDA algorithm for best path identification, one is used for storing the parameters of the smell spots and the other is used for storing SDA parameters. The parameters can be changed according to the selected optimization problem.

$$D_{SDA} = (S, \text{Total}, \text{Current})$$

where S is the signature value of the SDA; Total, the total length traversed by the SDA; and Current is the current node location of the SDA.

$$D_{ss} = (V, s)$$

where V is the signature of visited SDA; and s is the smell value of the spot with respect to the SDA.

Analysis

Basic analysis of any algorithm measures in asymptotically by time and space complexities. N_3 gives number of solution choices, if multiple path source–destination is selected for a problem. Time complexity of the SDA algorithm is time taken for 'n' agents reached from the source to the destination. The search space is visualised by more effectively in the form of adjacency list and using a search heap as a priority queue. Upper bound of the running time for searching the destination can represent in big O notation. Assume that the graph is connected and the algorithm requires $O(E + V\log V)$ time. Where, 'V' is the number of vertices and 'E' the number of edges. Time complexity is same as the Dijkstra's algorithm [9].

Comparison

Table 1 gives a comparison of derived SDA shortest path algorithm and other common shortest path algorithms in terms of complexity measures. Since these algorithms are working on the graph data structure, the space complexity is same. The proposed algorithm is same as Dijkstra's algorithm. When multiple solutions are being considered for a problem, more than one smell detection agent reaches the destination. No extra space is required for implementing this concept and its time complexity depends on the numbers of solutions. For example, if three solution paths are obtained for a problem then complexity becomes $3 \times O(|E| + |V| \log|V|)$. In multi-path Dijkstra's algorithm another inner loop is worked so that it gives the same time complexity for the solutions.

Table 1 Comparison of shortest path algorithms

Algorithm	Time complexity
Dijkstra's	$O(\|E\|+\|V\|\log\|V\|)$
Floyd–Warshall	$O(\|V\|3)$
Belman–Ford	$O(\|E\|\|V\|)$
SDA shortest path	$O(\|E\|+\|V\|\log\|V\|)$

Discussions

The ACO algorithm derives its inspiration from the unique swarm-based foraging technique of ants. In a colony of ants, one faction delves out in search of sources of food. Their random path of search is established through a secretion of pheromone, throughout the path.

This pheromone is a chemical signal that later acts as a navigational beacon for other ants. More ants from the colony continue to tread the paths and reinforce the trail with further pheromone secretions. The optimum path is arrived at, by converging upon the path with strongest pheromone depositions. Since pheromone evaporates and loses strength over time, the shortest path will have thickest pheromone deposits, which would later on be continuously reinforced by other foraging ants. This inspirational swarm intelligence concept has been mapped into the algorithmic domain with remarkable efficiency. Another aspect of ant behaviour has also been utilized. This is in relation to its choice among multiple sources of food. An ant would prefer a smaller dollop of nutritionally better honey, rather than a bigger dollop of nutritionally inferior sugar [4]. This principle has also been used in certain problem applications like the shortest path problem solutions.

The ACO algorithm itself has been acknowledged here since it is much easier to draw distinctions between it and the SDA algorithm, considering the fundamentally different algorithmic aspects. These differences have been further traced in subsequent sections to analyze both the algorithmic and computational perspectives. ACO algorithms can be applied in combinatorial optimization problems such as routing vehicles that adapted stochastic problems, dynamic problems in real variables, multi-targets and parallel implementations. The network routing is highly connected to transportation systems through shortest path routing.

Since the proposed algorithm deviate in-principle from these algorithms regarding the design of the agent, the Ant colony algorithm itself has been chosen for a meaningful

comparative analysis. This principal difference is in the role of the agent. In ACO, the agent is active and partakes in an intelligent choice during the process of the algorithm. The agent in our algorithm just plots the path from a path-established environment in a computationally finite process. Thus, our agent is passive in nature and does not perform the active computational optimization as opposed to other contemporary nature inspired algorithms. Instead, optimizing steps in the algorithm is a background process that establishes smell trail in the environment. This step is the non-deterministic optimizing component of the algorithm.

Conclusion

The SDA algorithm is an optimization algorithm, formed from the rather curious and efficient mechanisms used by dogs; which could be used to solve computational problems. Multiple agents were used iteratively to obtain a faster and better solution. This algorithm can be applied to solve the shortest path problem in order to show the proof of this proposal. The stable path can be identified by a SDA through its settled path. Extension of the proposed algorithm would also be useful for solving NP-hard problems. The extended shortest path algorithm from the SDA algorithm gives the same asymptotic measures of the Dijkstra's algorithm. The power of the SDA algorithm is its new data structures for similar problem solving techniques. The modified SDA algorithm can be useful for problem solving in practical optimization techniques.

References

1. S. Gamier, J. Gautrais, G. Theraulaz, The biological principles of swarm intelligence. Swarm Intell. **1**(1), 3–31 (2007)
2. S. Parpinelly, H. Lopes, New inspirations in swarm intelligence: a survey. Int. J. Bio-inspired Comput **3**(1), 1–16 (2011)
3. M. Dorigoa, B. Christian, Ant colony optimization theory: a survey. Theoret. Comput. Sci. **344**(2–3), 243–278 (2007)
4. R. Saritha, C. Vinod, A novel algorithm based on honey bee foraging principle for transportation problems. ACCIS Proc. Elsevier **1**, 203–210 (2014)
5. J. Vickers, Mechanisms of animal navigation in odor plumes. Biol. Bull. **198**(1), 203–212 (2000)
6. A. Barry, Towards a common strategy for transducing olfactory information. Semin. Cell Biol. **5**(1), 55–63 (1994)
7. C. Boris, G. Andrew, R. Tomasz, Shortest paths algorithms: theory and experimental evaluation. Math. Program. **73**(2), 129–174 (1996)
8. A. Zungeru, L. Ang, P. Seng, Classical and swarm intelligence based routing protocols for wireless sensor networks: a survey and comparison. J. Netw. Comput. Appl. **35**(5), 1508–1536 (2012)
9. J. Donald, A note on Dijkstra's shortest path algorithm. J. ACM **20**(3), 385–388 (1973)
10. R. Bellman, On a routing problem. Q. Appl. Math. **16**(1), 87–90 (1958)
11. D. Rina, J. Pearl, Generalized best-first search strategies and the optimality of A*. J. ACM **32**(3), 505–536 (1985)
12. C. Vinod, H. Anand, *Artificial Intelligence and Machine Learning*, 1st edn. (Printice Hall India, New Delhi 2014)