International Academic Institute for Science and Technology

International Academic Journal of Science and Engineering
Vol. 2, No. 1, 2015, pp.129-138.
ISSN 2454-3896

International Academic Journal of Science and Engineering

www.iaiest.com

OOSP (Off-On Sources Problem)

Rohollah Omidvar[a]

[a] *Islamic Azad University Of Yasouj Branch.*

Abstract

Complicated uncertain problems and their solutions that are often inspired by nature are created in various areas of basic and engineering sciences to help people. In this paper the introduced issue is a simulation of the real world to solve computational problems using optimization algorithms. In the proposed problem a person in a city considers some sources as daily visits and starts from a source and visits all the sources in the same day. Every day a number of sources are turned off and they cannot be visited in the same day. At the end of specific number of days that person should be able to minimize the number of visits and visit them as least as possible. This problem is simulated in MATLAB simulation software as offline and the theoretical solution is presented by heuristic.

Keywords: Problem, Algorithm, Source, Node, cycle

Introduction:

Searching and sorting are among the things we do every day. We use well-known algorithms and protocols for searching and sorting [1]. A set of rules for solving a problem in a finite number of steps are called algorithm. Algorithms used for searching allow us to configure our business base on our data [2]. The advent of digital computers allows the routine use of adaptive algorithm in different applications [3]. Meta-heuristic algorithms make up a great part of optimization algorithms, computational Intelligence and soft intelligence of the contemporary world [5]. Many problems of artificial intelligence taken from the swarm intelligence of the biological development of factors such as human being, birds, fish and other creatures for example optimization of particle swarm in swarming behavior is based on the birds and fish [6]. Or Firefly algorithm is based on computer model of Firefly [7], [8]. Cuckoo *search* (CS) is an optimization *algorithm* inspired by the obligate brood parasitism of some cuckoo species [9]. In the past two decades, more than two dozen of optimization algorithms, differential evolution, the search algorithms that have the potential to solve many difficult engineering optimization problems were created [7], [10], [11], [12], [13], [14], [15]. Therefore it is possible to use many life events to help to solve various engineering problems. OOS[*] benchmark problems are used in applications such as marketing, field surveys, network, some math problems, special routing and games based on artificial intelligence. In the introduced problem several sources are navigated as marked points in a cycle. Then the number of cycles is determined and it is multiplied by the number of points and the number of paths or graphs are the number of the edges. In each cycle some points are turned off randomly and they are not navigated in that cycle. Now at the end of the navigation cycle of these points these points should equal with minimum number of navigations to achieve an optimal diversification.

OOS Benchmark Proposed

We introduce the problem as follows: a person in a city considers a number of sources and visits them unlimitedly, every day a number of sources close and the person has to ignore to visit them in that day and visits the open sources. At the end of a certain number of days he tries to divide his visits among the sources and observe diversity. In fact, the cost function is the minimum number of visiting each source. In this benchmark problem a number of points is considered as nodes for navigation and we start numbering from 1 and consider a number for each source which is the name of the source. Then the numbers of cycles in which all points are navigated are determined. For example, if we have 7 nodes and consider the number of cycles as 9, it means that all these 7 points must be navigated 7 times, thus we will have 63 path or edges that each path is a navigation from one point to another. The total number of paths is obtained based on equation 1.

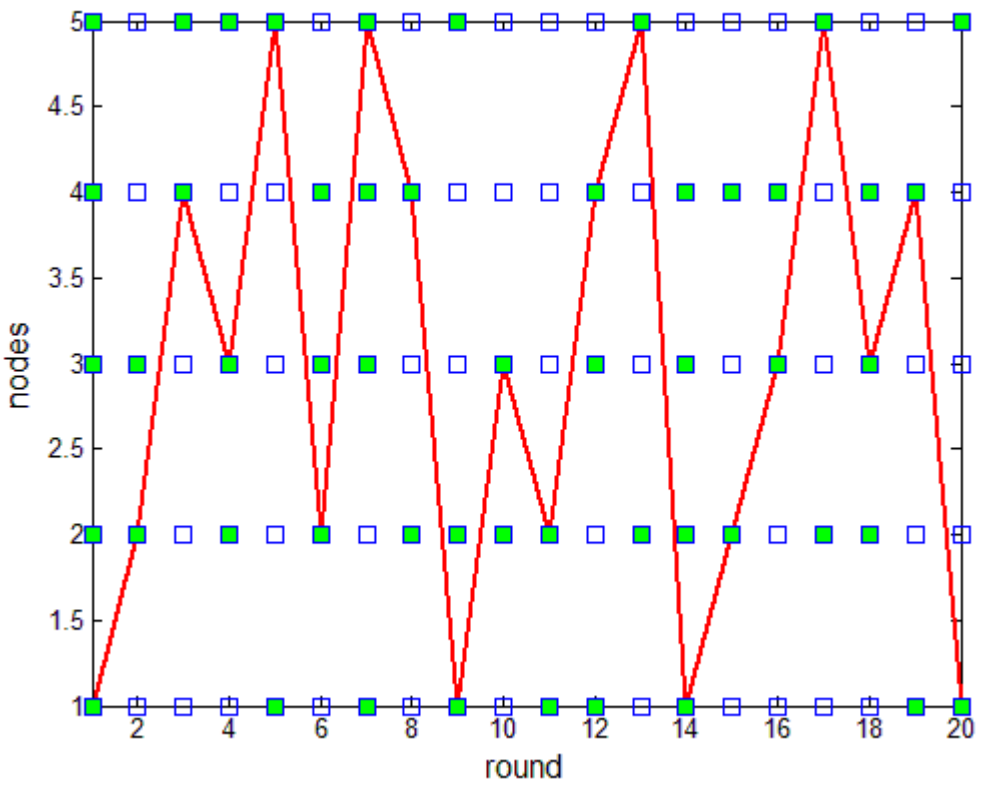
$$path = node * cycle \quad (1)$$

[*] Off-On Sources Problem

Now, at the beginning of each path one or a number of points black out and they cannot be navigated and we will have to choose among the light points to navigate which recurs continuously and in each navigation round a number of point black out. Mow at the end of navigating all paths the optimal path is the one that preserve the diversity of search and navigates each point as least as possible. For this purpose the use of standard deviation to measure and control the number of navigations of each node looks natural. Standard deviation is the most common measure of diversity and its lower value indicates that points are closer together and it is calculated based on equation 2 [4].

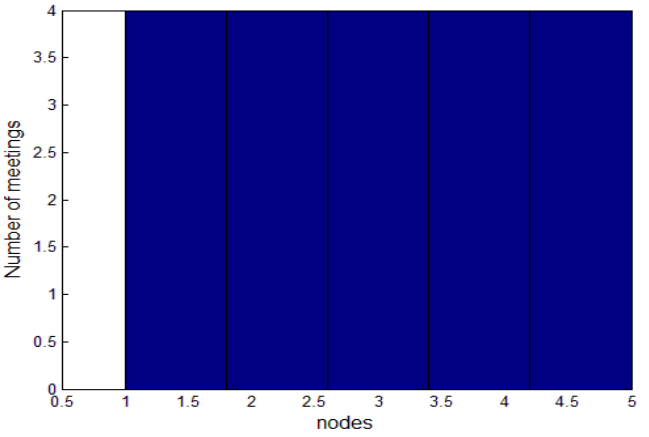
$$S = \frac{\Sigma(X-M)^2}{n-1} \quad (2)$$

To implement the problem offline the number of paths with a certain value is compared and standard deviation is compared and the optimal standard deviation is the smallest one close to zero (or zero under the best condition). Then that path is selected as the optimal path within the navigation period. For example, 5 sources and 4 cycles are considered. In fact the person has 5 sources and wants to visit them within 4 days. Naturally he has 20 paths and he faces with the closure of some sources each day. Figure 1 presents the sources and their on-off condition within these 20 moves of 20 cycles. Green sources are the light ones in that cycle that the person can visit them. As can be seen in Figure 1, there are 5 nodes on the vertical axis marked with the numbers 1 to 5 and 4 cycles that multiplied by the number of nodes we will have 20 moves and in each round a number of nodes are randomly off and they cannot be navigated. For example, the nodes (2) in the rounds 3, 5, 7, 12, 16 and 20 is off and it is not navigable, Therefore it is visited in movements 2, 6, 11 and 15. In this problem due to the small number of resources and cycles, all resources have been visited equally but as the number of resources and cycles grow, the problem becomes complicated and uncertain. Figure 2 presents the histogram diagram of the number of nodes' navigation that due to the small number of nodes and navigation rounds, naturally all nodes have been navigated almost equally and standard deviation is close to zero. There are 5 sources in figure 2 each one marked with numbers from 1 to 5. A person starts navigation from source 1 and then as he wants to go to the next path, he faces the nodes 1, 4 and



Figure(1) Path optimization in example1

5 off and in order to go to the next source ha has to forget these sources in that day and choose the right source to continue and he has selected the node 2.



Figure(2) Histogram Diagram in example1

When he is in node 2 and he wants to choose the next source he is faced with the sources 1, 2 and 3 off and he is forced to choose between the nodes 4 and 5 to continue navigation. The process of selecting these choices is discussed below. This continues until the end of the path 20 which is the last path in which the person wants to have conducted the lowest visits possible.

```
OOSP Problem(cycles , Num_Nodes)
{
    For(i=0 to cycles*Num_Nodes)
{
  Rand(Node_Avalibility)
    ind(i)=rand (Num_Nodes)
  k=hist(ind,Num_Nodes);
     obt_goal=std(k);
    if obt_goal<obt_goal_old
       obt_goal
      best_Node_Avalibility=Node_Avalibility;
      best_ind=ind;
      obt_goal_old=obt_goal;
        }
 }
```

Figure(3) Pseudo code with standard deviation

OOSP Problem
1.start
2.for i=1 to n
If $x_i = on_{current\ round} \rightarrow ON_{list}$
If $x_i = on_{next\ round} \rightarrow H_{list}$
3.for i=1 to node Available
Select($x_i \in ON_{list} | x_i \in H_{list}$ **)**
Select($x_i \in ON_{list} | x_i \in Previous_{H_{list}}$**)**
4.end

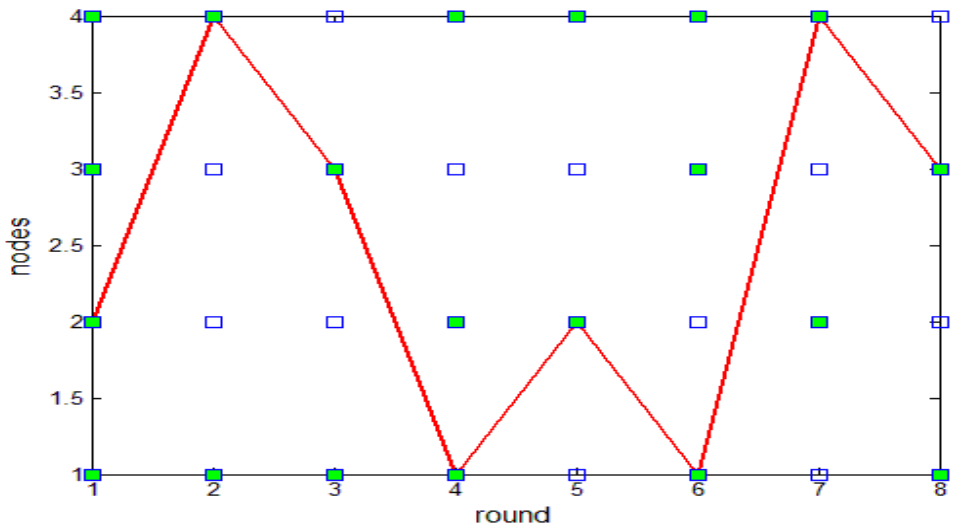
Figure(4) Pseudo code with heuristic

Based on the results within a comparable range of 2000, the standard deviation of the nodes in various paths has been compared 2000 times and each time the optimized lowest standard deviation is placed instead of the optimized target so that we achieve the lowest standard deviation in a comparable range which indicated the best path within that comparable range. Based on the observed results the lower number of nodes and cycles the standard deviation tends to zero more and with higher number of nodes and cycles, the complications to achieve a more complicated path becomes higher. Figures 3 and 4 present the pseudo-code of benchmark problem OOS based on two methods. For the offline method of solving the proposed problem first for the number of moves obtained by multiplication of the number of sources by the number of cycles using the random function, off modes are created for the sources and standard deviation is measured between the sources and each time the previous best standard deviation is replaced by the new best standard deviation and optimize the alternative path. For theoretical solution of the problem a problem is mentioned below:

The theoretical solution that navigates all search algorithms is considered for this algorithm as well. Sometimes the optimal paths in this algorithm may differ from standard deviation and theoretical approach but both of them are optimal. Now we introduce the theoretical solution of the problem:

- First we consider one node as the initiator of the navigation.

- After navigating the first node, we need a list of neighboring nodes that can be navigated at that moment and they are on and a heuristic list. Assuming the knowledge about the sources that should be navigated next (sources that can be navigated tomorrow) and a list of the nodes on at that moment,
- We analyze the nodes on and navigable at that moment and the heuristic list (nodes that can be navigated next) and if a node existed on two lists at the time, it would be removed from navigation at that moment.
- If all nodes were of equal priority we randomly select the node among the remaining ones and navigate it.
- Then we analyze the heuristic nodes and their on mode at that moment. If a node existed on two lists at the time, we ignore it and navigate the rest of nodes.
- Now a priority id added to the priorities i.e. if there was a node that was ignored due to be in two lists, it is a priority in navigation at this stage under the condition that it is absent in the next heuristic list.
- We follow these steps until each node is navigated equal to the cycle or less than it and the round are ended.

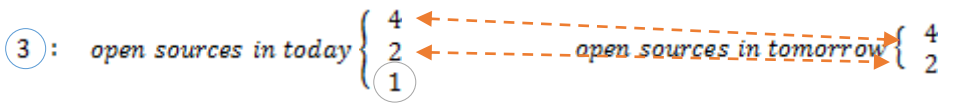


Fig(5) Histogram Diagram in example1

We present the solution with and example. If we assume that we have 4 nodes and the number of cycles is 2, the nodes should be navigated 8 rounds. We proceed step by step. We consider the first node as node 4. In this example, the left box shows the list of nodes that are on at that moment and the right box indicates the list of nodes that are on at the next moment, so with the knowledge of these two lists we start to choose the node for navigation based on what have been discussed.

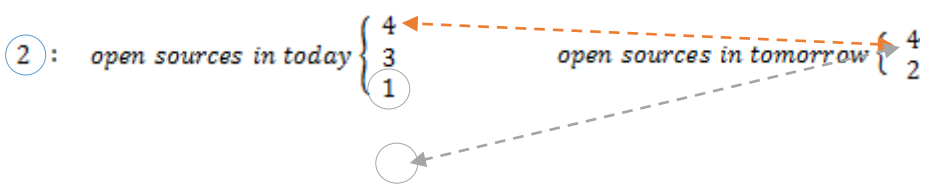
First we are in node 2 and the on nodes for navigation are 1 and 4 and the numbers in parentheses indicate the number of visits of each node. Nodes 1 and 3 are in the heuristic list or the list that will be on the next round. In fact, the person knows that today the sources 1 and 4 are on and tomorrow the sources 1 and 3, thus according to this information he decides to choose the next source. However, because the node 1 is in both lists it is ignored at this stage and node (4) is navigated.

Node 1 is in both lists therefore it is ignored thus node 4 is navigated.

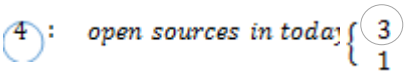


Nodes 4 and 2 are in both lists therefore node 1 is remained for navigation. We select it.

Node 4 is in both lists therefore it is ignored thus node 2 is navigated.



Here since node 4 is in both lists it is removed from being visited and among nodes 1 and 3, node 1 is navigated. Since node 4 was removed because it was in both lists and it does not exist in both lists here, it has priority.



Finally in the list of on nodes 2, 3 and 4 the node with fewer visits is selected.

Conclusions

The issue presented is a complex search benchmark problem that can be used to simulate complex benchmark environments. Based on what has been discussed we saw that the introduced OOSP problem since it is faced with of nodes and the lack of possibility of navigating multiple nodes, it is implemented in a dynamic environment and the algorithm can be implemented in various sciences. The presented algorithm can also be implemented based on the optimization algorithms followed by interesting results. This article attempted to introduce this innovative benchmark problem that is presented by coding and theoretical solution of the problem.

References:

A. H. Gandomi, X. S. Yang, and A. H. Alavi,(2011) Cuckoo search algorithm: a metaheuristic approach to solve structural optimisation problems, Engineering with Computers, 27, article DOI 10.1007/s00366-011-0241-y.

AGA institute

C. A. Floudas and P. M. Pardolos,(2009) Encyclopedia of Optimisation, 2nd Edition, Springer.

Csizmar Dalal, Amy, (2004),“ Searching and Sorting Algorithms,” Supplementary Lecture Notes, 1.

IBM Initiate ,(2012),” Introduction to algorithms” , training presentation,3.

J. C. Bansal and K. Deep,(2008) Optimisation of directional overcurrent relay times by particle swarm optimisation, in: Swarm Intelligence Symposium (SIS 2008), IEEE Publication, pp. 1-7.

J. Kennedy and R. Eberhart,(1995) Particle swarm optimisation, in: Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942-1948

R. S. Parpinelli and H. S. Lopes,(2011) New inspirations in swarm intelligence: a survey, Int. J. Bio-Inspired Computation, 3, 1-16 .

Sheik, S. S., Aggarwal, Sumit K. Poddar, Anindya, N. Balakrishnan, and K. Sekar,2004," A FAST Pattern Matching Algorithm", Bioinformatics Centre and Supercomputer Education and Research Centre, Indian Institute of Science,1.

X. S. Yang and S. Deb, (2009)Cuckoo search via L´evy flights, Proceeings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009, India), IEEE Publications, USA, pp. 210-214

X. S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooper- ative Strategies for Optimisation (NICSO 2010) (Eds. J. R. Gonzalez et al.), Springer, SCI Vol. 284, 65-74 (2010).

X. S. Yang, Firefly,(2009) algorithms for multimodal optimisation, Proc. 5th Symposium on Stochastic Algorithms, Foundations and Applications, (Eds. O. Watanabe and T. Zeugmann), Lecture Notes in Computer Science, 5792: 169-178

X. S. Yang,(2008), Nature-Inspired Metaheuristic Algorithms, Luniver Press, UK

X. S. Yang,(2010) Engineering Optimisation: An Introduction with Metaheuristic Applica- tions, John Wiley and Sons, USA .

Yang, Xin-She, He, Xingshi, (2103), Firefly Algorithm: Recent Advances and Applications, School of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, UK., School of Science, Xi'an Polytechnic University, No. 19 Jinhua South Road, Xi'an 710048, P. R. China.1.