Bulletin of Environment, Pharmacology and Life Sciences
Bull. Env. Pharmacol. Life Sci., Vol 3 [6] May 2014: 132-138
©2014 Academy for Environment and Life Sciences, India
Online ISSN 2277-1808
Journal's URL:http://www.bepls.com
CODEN: BEPLAD
Global Impact Factor 0.533
Universal Impact Factor 0.9804

ORIGINAL ARTICLE

Solving traveling Saleman Person by Simulating insect in Wheat Farms

Fariborz Ahmadi[1], Hamid Salehi[2], Seied veria hoseini[3]
[1,2,3]Young Researcher and Elite club, Ghorveh branch, Islamic azad university, Ghorveh, Iran
Email: sanandajstudent@gmail.com[1], hamid.salehi@gmail.com[2], hverya@gmail.com3

ABSTRACT
The idea of the traveling salesman problem (TSP) is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of this tour is minimized. Currently the only known method guaranteed to optimally solve the traveling salesman problem of any size, is by enumerating each possible tour and searching for the tour with smallest cost. Each possible tour is a permutation of 123 . . . n , where n is the number of cities, so therefore the number of tours is n !. When n gets large, it becomes impossible to find the cost of every tour in polynomial time.Almost all of the approach to solve NP-hard and NP-complete problem simulate artificial life. In this research, the behavior of eurygaster life is studied, so according to their life the new algorithm is introduced. In this research, using the behavior of eurygasters, a new algorithm has been invented and has been tested on traveling saleman person. The evaluation results show the advantage of researcher algorithm over ancient ones like genetic and PSO.

Keywords: *Evolutionary computation, genetic algorithm, particle swarm optimization, eurygaster algorithm, evolutionary programming.*

Received 10.01.2014 Revised 04.03.2014 Accepted 19.04. 2014

INTRODUCTION

Nonlinear problems are important in many real life applications because of their intrinsic difficulty; recently the area has attracted much research with the advances in nature inspired heuristics and multi agent systems for these problems. The dramatic increase in the size of the search space and the need for real time solutions motivated research idea in solving NP-problems using nature inspired heuristic techniques. In order to solve NP-problems some method like genetic algorithm and particle swarm optimization has been invented. These algorithms are population based and use intelligent to converge to the solution of problems.

To solve NP-problem some other algorithm such as ant colony [3], simulated annealing [4], tabu search [5], honey bee algorithm [6], photosynthetic algorithm [7], enzyme algorithm [8], glowworm swarm optimization [9], monkey search [10], firefly algorithm[11] were also introduced. These algorithms simulate behavior of artificial life of species.

In this work, researchers have invented new method called eurygaster algorithm based on eurygaster life. This algorithm is suitable for both continuous and discrete NP-problems and also, special for divide and conquer problems. In researcher approach, to find the solution of problem, problem must be divided in to some categories. After that, each cattle of eurygasters scatters over one of these categories to find the solution in search space of the category. In case of not finding the solution, new eurygasters are produced and distributed on the other categories and in case of finding the solution, the algorithm is finished. This method is easy to implement and can be simulated by a few line of computer code and is computationally inexpensive in term of memory and speed. Despite PSO that is used for continuous NP-problem, this algorithm is proposed for both continuous and discrete NP-problems. In other hand, in PSO algorithm [1,2], particles go through the search space of problem at each time continuously, while in research approach a group of eurygasters change their presentations and locations according to new category in which they colonize. In researcher approach, the search space of problems must be split on several groups or categories so that a great number of eurygasters disperse on each group to exhaustedly consider its space to find the best solution in that group.

In othr hand, the idea of the traveling salesman problem (TSP) is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of this tour is minimized.

The first instance of the traveling salesman problem was from Euler in 1759 whose problem was to move a knight to every position on a chess board exactly once.
The traveling salesman first gained fame in a book written by German salesman BF Voigt in 1832 on how to be a successful traveling salesman. He mentions the TSP, although not by that name, by suggesting that to cover as many locations as possible without visiting any location twice is the most important aspect of the scheduling of a tour. The origins of the TSP in mathematics are not really known all we know for certain is that it happened around 1931.
The standard or symmetric traveling salesman problem can be stated mathematically as follows:
Given a weighted graph G = (V , E) where the weight cijon the edge between nodes i and j is a non-negative value, find the tour of all nodes that has the minimum total cost.
Currently the only known method guaranteed to optimally solve the traveling salesman problem of any size, is by enumerating each possible tour and searching for the tour with smallest cost. Each possible tour is a permutation of 123 . . . n , where n is the number of cities, so therefore the number of tours is n !. When n gets large, it becomes impossible to find the cost of every tour in polynomial time. In this research, eurygaster life is simulated to solve traveling saleman person. The evaluation results show that researcher's algorithm is more efficient than other algorithms.
The reminder of this paper is divided into 5 sections. At first in section 2, the characteristics of previous work are detailed. After that, in section 3 behavior of eurygaster is described. Then, in section 4, our method or eurygaster algorithm is elaborated. Finally, section 5, 6 shows the evolution results of our method on TSP and draws some conclusion, respectively.

PREVIOUS WORKS

Local optimization tour

Among the simple local search algorithms, the most famous are two way local optimization tour and three way local optimization tours. Both of these algorithms are optimization algorithms. They don't find TSP tours, but improve upon existing tours. Therefore, both of these algorithms require other algorithms to build up a tour. The most common start-up algorithms are nearest neighbor and greedy. Nearest neighbors and greedy algorithms are fast but yield poor solutions. The result tours from these start-up tours are then passed on to two way local optimization tours or three way local optimization tour algorithms for further optimization. In two way local optimization tour algorithm, one move deletes two edges, thus breaking the tour into two paths, and then reconnects those paths in the other possible way. In three way local optimization tour algorithms, one move reconnects three edges instead of two. The popularity of these two algorithms comes from their simplicity and fairly good optimization results. However, straight forward implementations of two and three way local optimization tour are subquadratic in run time. This is due to the fact that for every move, each edge has to compare itself with all other edges to find the optimal edge swapping. To reduce the run time, an optimization based on the finding of Lin and Kernighan tells the current edge to only look at neighbor nodes that are nearby, instead of the whole space. This effectively reduces the number of edge comparisons these algorithms have to make. However, this does requires each node in the graph to keep a list of its nearest neighbor, increasing the overall memory requirements. The memory requirement grows quadratic with the number of nodes in the graph, making the algorithms unfeasible for large TSP tour graphs. To alleviate the memory constraint, many currently implementations of two and three way local optimization tour algorithms set the size of neighbor-list constant. Another optimization breaks up the TSP tour graph into many sub-regions. The neighbor-list is only kept for neighbors in the same sub-region. Both of these optimizations are implemented for the two and three way local optimization tour travling saleman person programs that we investigated [13].

Lin-Kernighan

Lin-Kernighan and its derivative algorithms are widely considered as the best symmetric TSP heuristic algorithms. The algorithm itself is a form of tabu search. Tabu search is motivated by the observation that not all locally optimal solutions are good solutions. Thus it might be desirable to modify a pure local optimization algorithm by providing some mechanism that helps us escape local optima and continue the search further. LinKernighan combines tabu search with a generalization of two and three way local optimization tour algorithm. At each step, the algorithm only considers two way local optimization tour moves that is better than the best tour seen so far. If the new tour length is shorter than the old tour length, but it is not shorter than the best tour length seen so far, the algorithm does not make the swap. The tabu condition of the algorithm is kept by maintaining two additional lists – added edges list and deleted edges list. A move is tabu if it attempts either to add an edge on the deleted tabu list or to delete an edge on the added tabu list. To optimize for memory usage, some implementations of this algorithm omit the deleted tabu list. Lin-Kernighan algorithms also maintain the neighbor lists similar to two and three way local optimization tour algorithms [13].

Simulated Annealing

Simulated annealing is a Monte Carlo approach for minimizing such multivariate functions. The term simulated annealing derives from the roughly analogous physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure. In simulation, a minimum of the cost function corresponds to this ground state of the substance. The simulated annealing process lowers the temperature by slow stages until the system ``freezes'' and no further changes occur. At each temperature the simulation must proceed long enough for the system to reach a steady state or equilibrium. This is known as thermalization. The time required for thermalization is the de-correlation time; correlated microstates are eliminated. The sequence of temperatures and the number of iterations applied to thermalize the system at each temperature comprise an annealing schedule. To apply simulated annealing, the system is initialized with a particular configuration. A new configuration is constructed by imposing a random displacement. If the energy of this new state is lower than that of the previous one, the change is accepted unconditionally and the system is updated. If the energy is greater, the new configuration is accepted probabilistically. This is the Metropolis step, the fundamental procedure of simulated annealing. This procedure allows the system to move consistently towards lower energy states, yet still `jump' out of local minima due to the probabilistic acceptance of some upward moves. If the temperature is decreased logarithmically, simulated annealing guarantees an optimal solution [13].

Genetic Algorithm

Genetic algorithms are a part of evolutionary computing, inspired by Darwin's theory of evolution. Solution to a problem solved by genetic algorithms uses an evolutionary process. The algorithm begins with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are then selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied [13].

Dijkstra algorithm

Dijkstra algorithm finds the shortest path between two points. It is not a TSP algorithm. Furthermore, it is not a NP algorithm. We included this algorithm in our investigation because we want to see if there is a great difference in program behavior between NP algorithms and P algorithms. Plus Dijkstra algorithm is also a graph related algorithm. We also want to see if there is any common behavior between all graph related algorithms.

Eurygaster behaviors

Eurygaster integrates is an insect pest that predominantly attacks grains, feeding on the leaves, stems and grains, reducing yield and injecting a toxin into the grains which adds a foul smell to the resulting flour, and substantially reduces the baking quality of the dough.

In winters eurygasters live under the plants and bushes in hillside, in several numbers and make a group. At the end of winter and at the beginning of spring when it gets warmer, these insects end their winter sleeps and get ready to move and fly to grain fields by moving over the high mountains and leaving the nests in groups. The first group by the use of its instinct finds the best and the nearest grain fields and stays there. Getting there, this group of insect sends signals to the air to show the other groups their being there. Based on the number of eurygasters in a place, the strength of signals will be different. If the number of eurygasters in a grain field is not great, the rate of diffused signals will be little and if the number of eurygasters in a grain fields is greater, the rate of diffused signals will be increased. They diffuse these signals to show the others that reside there. So that the other groups of eurygasters understand that they should not close to the grain field which contains the first group. Of course the other groups based on diffused signals by the first group and the strength of these signals they decide if they can land and stay there or not. If the power of diffused signals is low, it means that some of the other groups of eurygasters can land and stay by the other groups which are resident there and began to eat. While the strength of the signals in the sky is high, it means that the other groups cannot land on the field(s) containing eurygasters, and they must fly to other fields in which there are no eurygasters, to live and eat.

According to the passage mentioned above, the next group of eurygasters while flying from their nests to other fields to find the best grain fields searches the best and closest ones to land and eat based on the broadcasted signals by landed group(s). This process will continue until they will find a suitable and useful grain field to eat.

We conclude that all the grain fields in a wide area will be attacked by eurygasters, because they do not gather in a one place.so, when there is not enough food in a grain field in which the eurygasters have stayed for a time, they will fly to a new field with no eurygasters according to the process mentioned above.

EURYGASTER ALGORITHMS

In this section, proposed approach or eurygaster algorithm is described. Solving non-linear functions are so necessary in real life today and recently researchers interested in inventing methods to solve them. Thus, our approach contributes to solve NP-class problems. The great advantage of this algorithm is that it's so easy to implement and is also inexpensive in term of memory and speed. The second advantage of this algorithm is its convergence speed compared to other methods like GA and PSO.

In spite of GA that traps in local optimum, the evaluation results of this approach show this method doesn't have this drawback. The major disadvantage of this method is that it can be used only for dividable problems. So the problems must be divided into some groups to be solved by researchers approach. Despite this problem, the algorithm advantage clearly outweighs over its disadvantage.

As it was mentioned, eurygasters attack wheat fields in groups. When the new generation is produced, they attack the fields which haven't been attacked before. After some period of times, it is revealed that all the fields in a region have been attacked totally by eurygasters. The main point inspired by eurygasters behaviors is that if we can divide the problems into some partitions by producing element named eurygasters in each partition all the space of the problem can be searched. The main point in solving problems by this algorithm is the way in which the problem can be divided into some partitions or how to the problem can be partitioned. That is, the better the partitioning is done, the more accurate the solutions to the problem can be concluded.

Now suppose that the problem has been divided into n partitions. First several eurygasters based on the size of the problems partitions are generated and distributed in the space of the first part to find the solution of the problem. It should be mentioned that if the number of eurygasters is not enough to exhaustedly cover all the space of the related partition, we can search all the related space by redistributing them and changing their position so that they can cover all the space mentioned. This process in genetic algorithms is done through mutation [24]. Using mutation in the proposed approach prevents local optimum. After searching for the first partition and in case of not finding the optimal solution to the problem, new kinds of eurygaster relevant to the second part are produced in order to be used to search in the space of the second partition. This operation is continued up to the n-part and in case of finding the expected solution in each part, the related algorithm is terminated and the solution of the problem is reported. The related semi-code of the proposed algorithm is as follows:

```
I← the number of clusters

   While I <> 0 do

  1.  Initialization:  produce euragasters or particles according to characteristic of one partition
  2.  Distribution: distribute eurygasters on the regions of the partition
  3.  Evaluation: evaluate suitability of each eurygaster or particle depend on the problem
    3.1  If the suitable result of the partition is not obtained
         3.1.1.  Change the position of Eurygasters in the partition
         3.1.2.  goto 3
    3.2  If the result of the problem is not obtained
         3.2.1.  I--
         3.2.2.  goto 1
        Else
         3.2.3.  Stop algorithm or break
   End while
  4.  Report the solution of the problem
```

Algorithm1. Eurygaster algorithm

In proposed algorithm by researchers, namely eurygaster algorithm, the partitions must be created before execution of the algorithm. At first, in *initialization* phase a set of the eurygasters are produced and scattered on the search space of relevant partition by *distribution* phase. After that the suitability of each eurygaster is computed according to the problem kind as you can see in evaluation phase. In case of finding the solution of the problem in any of the partition, there may be two possibilities.

1. The found solution is optimal solution for that partition but not for the main problem. In this case line 3.1 is not executed but 3.2 is executed and the algorithm executes initialization phase to create new eurygasters for searching the solution in another part because of moving control to

that phase in line 3.2.2 (goto 1).

2. The found solution is optimal solution for that partition and also for the main problem. In this case neither the sub instructions of line 3.1 nor the sub instructions of line 3.2 are executed and cause the algorithm be stopped. Therefore, this is a point that the absolute solution of the problem is reached.

In case of not finding the best solution of the partition or when it is hoped that the solution is in search space of the partition, line 3.1 is executed and changed the eurygasters location to fully cover the area of the pertition search space. Finally, if the appropriated solution to the problem is obtained, the algorithm is terminated by line 3.2.3, otherwise a new partition is created and also new eurygasters are produced to be distributed over the partition.

Now, w use researcher's algorithm to solve TSP according to following steps.

- First, divide cities into sections according to the number of cities. Namely, one section while the number of each cluster except the last one has 3 cities. Anothr section while the number of each cluster except the last one has 4 citis and so on.
- Second, produce and distribute insects (eurygasters) on one section until satisfied answer of section is obtained.
- Third, check wether the solution of previous phase is the solution of the overall problem by evaluating suitability of each insect, otherwise execute second phase on another section until the best solution is find.
- Fourth, stop the algorithm and present the city ordr.

EVALUATION RESULTS

The results of the studies show the algorithm PSO is suitable for continuous problems and its using in discrete problems is very rare and expensive, while the proposed algorithm is suitable for both continuous and discrete problems. In the proposed method the space of the problem (cities) is divided into several sections and in each step of algorithm just one section of the problem is investigated and in case of not reaching the expected solution, the other part is going to be searched. Also, in each part by changing the position of the eurygasters, we can prevent trapping in local optimum. On the one hand, our method has great advantages over genetic algorithms. In the genetic algorithms, the chromosomes of the first generation are distributed randomly all over the problem, this operation causes some problems as follows:

1. The probability of trapping in the local optimum increases.
2. Chromosomes that will be produced in the next generations can be placed in some locations in the problem space where the chromosomes of the previous generations had been placed. In other word, some regions of the problem space will be searched several times that causes execution time of the algorithm is increase and lead to decreaing in convergence speed.
3. Since in each generation all the space of the problems is searched for, the execution time of the algorithm will be increased. The proposed algorithm will not have the above-mentioned problems because of searching all the partition continuously, orderly and separately.

For comparing the efficiency of the mrthod, genetic algorithms and researchers algorithm have been encoded in C++ on a personal computer with cpu speed 3 GHz and 448 MB RAM under MS Windows XP, and for some TSPLIB instances. The experiments were performed 20 times for each instance in genetic algorithm (GA) and researcher proposed algorithm (RPM). The solution quality is measured by the sum of the differences of each run with the optimum solution reported in TSPLIB website.

Table1. Comparing GA with RPM by difference with optimum solution and runtime in second

Instance Name	Average difference of RPM with optimum solution	Average difference of GA with optimum solution	RPM average Runtime	GA average Runtime
ftv170	231.8	439.35	520.76	789.65
kro124p	195.6	287.4	501.89	540.34
ftv70	199.7	365	240.93	243.12
ft70	297.9	321.2	87.78	90.46
ftv64	179.1	275	61.92	82.23
ftv55	197.4	198.4	34.12	89.54
ft53	204.6	217.2	32.12	76.98
ry48p	159.7	164.2	12.1	27.65
ftv47	23.6	65	32.78	31.23
ftv44	53.7	67.8	12.91	35.46

p43	9.2	8.6	12.56	32.47
ftv38	8.5	12.8	3.11	3.36
ftv35	12.8	8.35	3.09	4.54
ftv33	0.4	0.4	1.67	2.98
br17	0	0	0.38	0.76
d198	212.2	389	326.67	467.87
brg180	89.1	127	342.29	432.75
lin105	42.9	72.2	108.23	320.87
eil101	31.6	78.8	217.90	322.67
kroC100	57.9	61.3	243.62	278.45
kroA100	43.7	65.2	129.6	189.43
pr76	41.1	67.7	142.12	201.89
eil76	57	76.3	197.06	213.27
berlin52	51	56.2	87.53	129.87
eil51	54.2	57.4	17.71	19.43
bayg29	12.9	14.3	1.47	3.96

Since the limitation that local optimum has created for the genetic algorithm, the percentage of successfulness by using this algorithm to get the exact solution is less than the proposed algorithm. Proposed algorithm lacks the limitations of local optimum and if the related fitness functions are defined properly, it can be claimed that almost 95 percent of cases getting the optimum solution is possible. Also, table1 show that our approach has totally more performance than genetic algorithm. The first column of table describes the instance name. The second and third columns describe the average difference of researchrs method and genetic algorithm with optimum solution in 20 runs. As table1 indicates, our method has better solution than GA except in p43. Finally columns 4 and 5 show the average running time by researcher's method and GA. As you can see in the table1, the total time of our approach is less than genetic algorithm that can be concluded researcher approach is faster and convergence speed is high.

CONCLUSIONS

In this article, a new method based on the behaviours of eurygasters has been presented to solve tsp problem. this approach unlike genetic algorithm lacks the local optimum so the probability of getting the more accurate solution in this method is much more than the genetic algorithm. Moreover, in this method every space of the problem is searched for once while in the genetic algorithm every part of the problem space can be searched several times in different generations, so the rate of convergence in this algorithm is much more than the genetic algorithm. also, this algorithm is easy to implement by computer. if takes a few lines to programming and doesn't need a huge memory or cpu speed. In the proposed method to solve tsp the space of the problem is divided to several partitions and every partition is searched for separately. the more efficient manner to reduce running time is that if the solution of a problem is found in a specific partition, the searching process be stopped in order to decrease the amount of execution time of the algorithm. since different parts of the problem have been separated, this method is very simple and efficient to be applied in parallel in solving problems like tsp. as it is seen in table1, the running time of this method is less than gntic algorithm and also the result of this algorithm is more efficient than ga. the researchers hope that the explanation in this article can satisfy its readers about the operation of the algorithm and method.

REFERENCES

1. R.C. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory". In Proceedings of the sixth international symposium on micro machine and human science, volume 43. New York, NY, USA: IEEE, 1995.
2. J. Kennedy, and R.C. Eberhart, "Particle swarm optimization". In Proceedings of IEEE international conference on neural networks, volume 4, pages 1942–1948. Perth, Australia, 1995.
3. M. Dorigo, "Optimization, Learning and Natural Algorithms", PhD thesis, Politecnico di Milano, Italie, 1992.
4. S. Kirkpatrick, C. D. JR. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", IBM Research Report RC 9355.
5. F. Glover, and M. Laguna, "Tabu Search", Kluwer Academic Publishers, Boston, 1997.
6. Pham, DT. Ghanbarzadeh, A. Koc, E. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm. Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.
7. H. Murase, "Finite element analysis using a photosynthetic algorithm", Computers and Electronics in Agriculture, 29, pp. 115-123, 2000.
8. X. S. Yang, " New enzyme algorithm", Tikhonov regulation and inverse parabolic analysis, in Advances in Computational Methods in Science and Engineering, Lecture Series on Computer and Computer Sciences, ICCMSE, Eds. T. Simons and G. Maroulis, 4, 1880-1883, 2005.
9. K.N. Krishnanand, and D.Ghose, "Detection of multiple source locations using a glowworm metaphor with applications to collective robotics," IEEE Swarm Intelligence Symposium, Pasadena, California, USA, pp. 84–91,

2005.

10. A. Mucherino, and O. Seref, “Monkey Search: A Novel Meta-Heuristic Search for Global Optimization,”AIP Conference Proceedings 953, Data Mining, System Analysis and Optimization in Biomedicine, 162–173, 2007.
11. X.S. Yang, "Firefly algorithm," (chapter8) in: Nature-inspired Met heuristic Algorithms, Luniver Press, 2008.
12. D. Doval, S. Mancoridis, and B. Mitchell,"Automatic clustering of software systems using a genetic algorithm," STEP '99, IEEE Computer Society, 1999.
13. David S. Johnson and Lyle A. McGeoch, “The Traveling Salesman Problem: A Case study in Local Optimization”,1995.