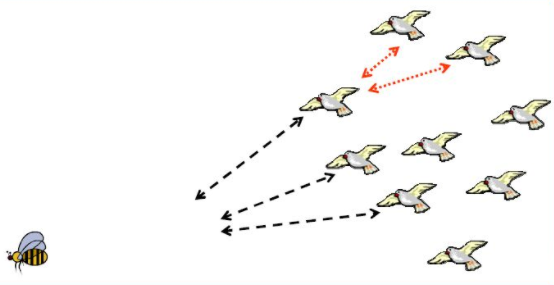
An Optimal Trust Path Selection Algorithm Based on Particle Swarm Optimization in Small World Network

Report Paper

Under the guidance of Munmun Das



Submitted by

Manisha Kumari and Saurav Kumar

Department of Information Technology

Jadavpur University,Saltlake Campus

Abstract

The optimal social trust path selection problem is known to be a challenging research problem. Trust plays a crucial role in computer mediated transactions and processes.

However, it is often hard to assess the trustworthiness of remote entities, because computerized communication media are increasingly removing us from familiar styles of interaction.

So, here we provide a Trust Path Selection Algorithm based on Particle Swarm Optimization (PSO). This paper focuses on the Algorithm based on cost based decoding encoding method which is used to encode or decode a particle. In this Paper Performance analysis is also done for the algorithm and the algorithm is tested with 4-5 datasets.

Acknowledgement

We are very grateful to my project guides Prof. Munmun Bhattacharya for giving her valuable time and constructive guidance in preparing the Project. It would have not been possible for me to complete this project in this short period of time without their encouragement and valuable guidance.

Date:

Signature :

Name of the Students : Manisha Kumari Roll no - 001411001043

Saurav Kumar Roll no - 001411001049

CERTIFICATE

This is to certify that Manisha Kumari and Saurav Kumar of INFORMATION TECHNOLOGY has successfully completed the project work titled An Optimal Trust Path Selection Algorithm Based on Particle Swarm Optimization in Small World Network. This project report is the record of authentic work carried out by them during the period from July 2017 to April 2018. They have worked under my guidance.

Signature :

(Internal) Date:

Name Project Guide

Munmun Bhattacharya

Contents

Abstract

1) Introduction

1.1) Trust path selection in Small World Network

1.2) Need of trust path in small world network

2) Background

3) The advantages of particle swarm optimization algorithm

4) Particle Swarm Optimization

4.1) Basic steps of PSO algorithm.

5) Algorithm description

5.1) Representation and meaning of the particle swarm: Existing path encoding techniques

5.2) Cost-priority-based particle encoding/decoding.

5.3) Trust value Mechanism

5.4) Main idea of the Algorithm

5.5) The design of algorithm

5.6) Coding strategy

6) Function dataset

7) Simulation Results and Performance Evaluation

7.1) Efficiency Analysis

7.2) Influence of hops

7.3) Pso vs genetic algorithm

8) Conclusion

9) Limitation

10) Future scope  
References

1. Introduction

In is a distributed network, the network participants share part of their resources, which provide the network services and content directly or indirectly. Without going through an intermediate entity, each note can directly access other peer-to-peer node. Participant in P2P network is both a resources provider and a resources receiver. Compared with traditional distributed systems, P2P technology has unparalleled advantages and broad application prospects. P2P network has good scalability, decentralization, robustness, and load balancing features. However it still lacks the trust mechanism between nodes, it is necessary to establish a distributed trust mechanism, i.e., ﬁnd a optimal trust path between trustor and trustee. This can greatly promote the exchange of resources in the network, thus eﬀectively improve the overall system availability.

* 1. Trust path selection in Small World Network

Small world network is a type of mathematical graph, in which most nodes are not neighbors of one another, but can be reached from every other by a small number of hops. P2P network is a typical small world network, it has a large scale and high risks, so trust model building and Trust Path Selecting (TPS) become big challenges.

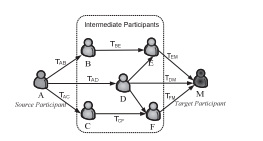
In 1967, Stanley Milgram launched a chain letter experiment, after that, he put forward the famous "six degrees of separation hypothesis", namely "small world phenomenon". Through the experiment, he proved that it is an average six middlemen for any two people on the Earth to be linked.

* 1. Need of trust path in small world network

Online service provision commonly take place between parties who have never transacted with each other before in an environment where the service customer often has insufficient information about the service provider and the goods and services offered. The consumer generally has no opportunity to see and try the product or service  i.e. to “squeeze the oranges” before he buys. The idea is that even if the consumer can’t try the product or service in advance, he can be confident that it will be what he expects as long as he trust the seller .A trusted seller therefore has a significant advantage in case the product quality can’t be verified in advance.

This example shows that trust plays a crucial role in computer mediated transactions and processes. However, it is often hard to assess the trustworthiness of remote entities, because computerized communication media are increasingly removing us from familiar styles of interaction.

Also in recent years, online social networks have been attracting a large number of participants. In such social networks, each node represents a participant and each link corresponds to real world interactions or online interactions between participants. One participant can give a trust value to another based on their past interactions As each participant usually interacts with many other participants, multiple social trust paths may exist between two participants who have no direct links with each other, such as the trust path A → B → E →M  and A → C → F → M in Figure, each of which is called a social trust path . Along a social trust path linking two nonadjacent participants, such as A (termed as the source participant) and M (termed as the target participant) in Figure, the source participant can evaluate the trustworthiness of the target one based on the trust information between the intermediate participants along the path. This process is called trust propagation.



In large-scale social networks, there could be tens of thousands of social trust paths between a source participant and the target one. Evaluating the trustworthiness of the target participant based on all these social trust paths can incur huge computation time. Alternatively, we can search the optimal path yielding the most trustworthy trust propagation result from multiple paths. We call this the optimal social trust path selection problem which is known to be a challenging research problem.

So, here we provide a Trust Path Selection Algorithm based on Particle Swarm Optimization (PSO). In the algorithm, after initializing the particle swarm, each particle can update the speed and location according to its information, and then produce a new particle with better value. Repeating that process continually to implement the global search of the space, we can get the better trust path in the networks. The experimental results show that this algorithm is effective and efficient in finding the sub optimal solution of trust path, hence it can be applied in trust path searching in such small-world networks.

1. BACKGROUND

Artiﬁcial neural networks (ANNs) have been examined to solve this problem using their parallel and distributed architectures to provide a fast solution. However, this approach has several limitations. These include the complexity of the hardware with increasing number of network nodes at the same time, the reliability of the solution decreases. Secondly, they are less adaptable to topological changes in the network graph, including the cost of the arcs. Thirdly, the ANNs do not consider suboptimal paths.

Thus, the evolutionary and heuristics algorithms are the most attractive alternative ways to go for. The powerful evolutionary programming techniques have considerable potential to be investigated in the pursuit for more eﬃcient algorithms. In this direction, genetic algorithm (GA) has shown promising results. The most recent notable results have been reported in .Their algorithm shows better performance compared to those of ANN approach and overcome the limitations mentioned above. Among the notable evolutionary algorithms for path ﬁnding optimization problems in network graphs, successful use of GA and Tabu search (TS) has been reported.

1. The advantages of particle swarm optimization algorithm

The success of these evolutionary programming approaches promptly inspires investigative studies on the use of other similar (and possibly more powerful) evolutionary algorithms for this SP problem. It has been reported that the PSO performs better than other evolutionary optimization algorithms in terms of success arm optimization is one such evolutionary optimization technique , which can solve most of the problems solved by GA with less computational cost . It is to be noted that GA and TS demand expensive computational cost. Some more comparative studies of the performances of GA and PSO have also been reported. All these studies have ﬁrmly established similar eﬀectiveness of PSO compared to GA. Even for some rate and solution quality. The most attractive feature of PSO is that it requires less computational bookkeeping and, generally, a few lines of implementation codes. The basic philosophy and science behind PSO is based on the social behavior of a bird ﬂock and a ﬁsh school, and so forth. Because of the speciﬁc algorithmic structure of PSO (updating of position and velocity of particles in a continuous manner), PSO has been mainly applied to many continuous optimization problems with few attempts for combinatorial optimization problems.

Particle swarm optimization (PSO) algorithm is robust, simple in concept, and easy to implement. There is a lot of fraud in the network. Fraud is nothing more than reverse right and wrong, regard “good” nodes as “bad” and “bad” node as “good”. This paper uses the particle swarm optimization algorithm, which means that we ﬁnd out the optimal path step by step. When the algorithm is running, it will look for the non-inferior solution solutions and eliminate inferior solutions, and therefore it can eﬀectively prevent joint fraud.

In the trust network, the main diﬀiculty in the optimal path selection is the selection of the utility function. In particle swarm optimization algorithm, each particle corresponds to a point set which is composed by a path, so the greatest trust value for the point set of possible paths can be a natural choice for utility function. Considering the characteristics of small-world networks (the average shortest path is 6), you can take the limit of maximum number of steps to greatly simplify the process. In this way, the disadvantages of the PSO can be weakened as far as possible, and its advantages can be made use of fully.

In addition to the above advantages, the particle swarm optimization has another distinct advantage – Memory. In PSO, the global optimal solution transfer information to other particles and the entire search update process follow the current optimal solution

1. Particle swarm optimization

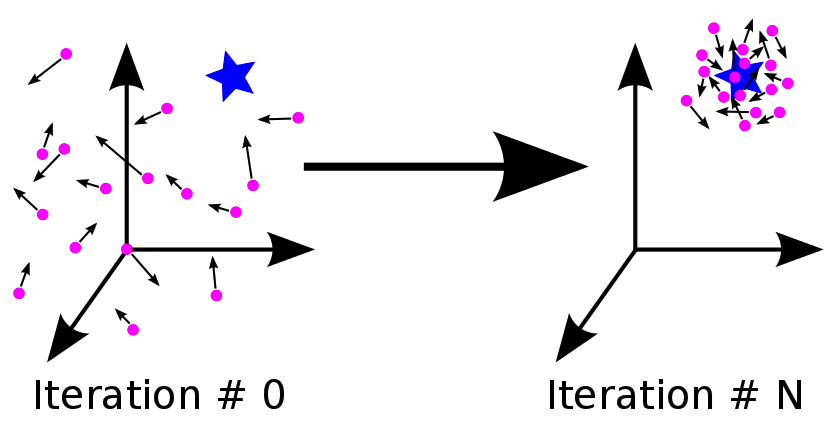
Particle swarm optimization is a population-based stochastic optimization tool inspired by social behavior of bird ﬂock (and ﬁsh school, etc.), as developed by Kennedy and Eberhart. It has been applied successfully to a wide variety of search and optimization problems (Neural Networks, Structural opt., Shape topology opt.)

PSO is a simple but powerful search technique. This new evolutionary paradigm has grown in the past decade.



4.1) Basic steps of PSO algorithm.

The PSO algorithm conducts search using a population of particles called swarm which correspond to individuals in a genetic algorithm. A population of particles is initially randomly generated. Each particle has a ﬁtness value determined by the optimization function and also has a speed to determine the direction and distance they ﬂy. Then the particles will search following the current optimal particles in the solution space. In each iteration, particles update themselves by tracking the two “extreme value”. The ﬁrst is the optimal solution found by the particle itself, this solution called individual extreme pBest. The other extreme value is the optimal solution found by the entire population, this solution called global extreme gBest. In addition to this global version, another local version of PSO keeps track of the best position among all the topological neighbors of a particle.



Assume that the search space is D-dimensional, and

Location of the particles in the particle swarm X = <xi1,xi2,…,xid>

The speed of the ith particle V = <vi1,vi2,…,vid>

The best location searched by the ith particle P = <pi1,pi2,…,pid>

The best location searched by the entire particles Pg = <pg1,pg2,…,pgd>

For each particle, the value of the ith dimension changes according to the following equation:

vid = \*vid + j1\*rnd()\*(pid-xid) + j2\*rnd()\*(pgd-xid);

xid = xid + vid;

**i**              = ith particle,

j1,j2 =   learning rates governing the **cognition** and **social** components

**g**             = index of the particle with the best p-fitness, and

**d**             = dth dimension.

Here rnd()  is a random number between [0, 1] and  j1,j2 is the acceleration coeﬀicient, generally take  j1 = j2 = 2. C is a constant set by the user, which is used to limit the maximum speed. Particles continuously search in the solution space by updating the individual extreme and the global extreme, until reach the required number of iterations or the speciﬁed error standard. At each position, the ﬂight speed of the particles cannot exceed the maximum speed.

pBest(i,t) = arg min [f (Pi( k ) )]

i={1,2,3,...........Np}

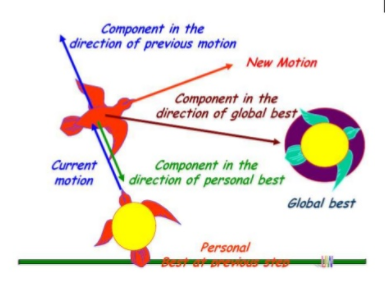
                        k=1,…,.,.t

gBest(t)   = arg min [f (Pi( k ) )]

               i=1,...Np

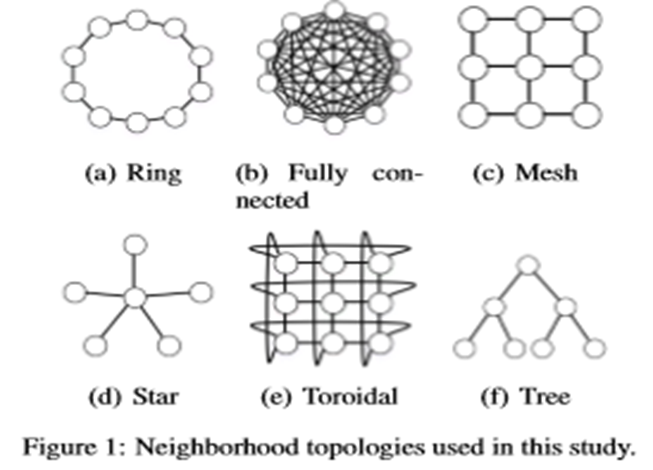
               k=1,…,.,.t

where 𝑖 denotes the particle index, 𝑁𝑃 the total number of particles, 𝑡 the current iteration number, 𝑓 the fitness function, and 𝑃 the position.



  The commonly used PSOs are either global version of PSO or local version of PSO. In global version, all other particles inﬂuence the velocity of a particle, while in the local version of PSO, a selected number of neighbor particles aﬀect the particle’s velocity. PSO is tested with regular-shaped neighborhoods, such as global version, local version, pyramid structure, ring structure. The neighborhood topology of the particle swarm has a signiﬁcant eﬀect on its ability to ﬁnd optima .In ring topology, parts of the population that are distant from one another are also independent of one another. Inﬂuence spreads from neighbor to neighbor in this topology, until an optimum is found by any part of the population and then, this optimum will eventually pull all the particles into it. In the global version, every particle is connected to all other particles and inﬂuences all other particles immediately. The global populations tend to converge more rapidly than the ring populations, when they converge; but they are more susceptible to convergence towards local optima.

Different topology for Pso



1. Algorithm description

Here we have to find optimal trust path in small world networks, so we can consider a small world network as a graph and there is a source node and destination node and between them we have to find the optimal trust path but in pso there is a concept of particle not a path so the main issue in applying PSO (GA)  is the encoding of a network path into a particle in PSO (chromosome in GA). This encoding in turn aﬀects the eﬀectiveness of a solution/search process. A brief discussion on some of the existing path encoding techniques is presented followed by a detailed description of the proposed encoding algorithm.

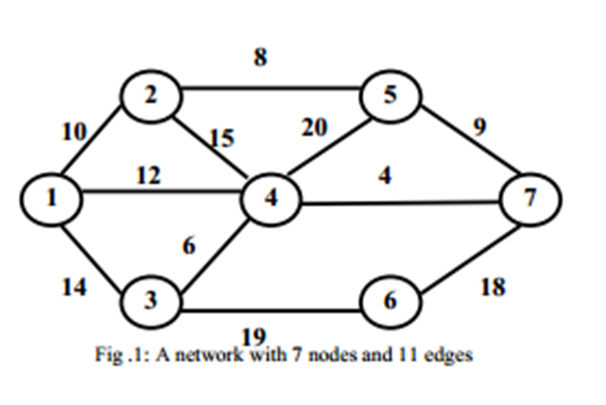
5.1) Representation and meaning of the particle swarm:

Existing path encoding techniques

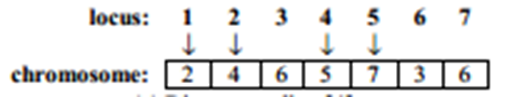
Two typical encoding techniques have been used for path representations .They are direct and indirect representations. In the direct representation scheme, the chromosome in the GA is coded as a sequence of node identiﬁcation numbers (node IDs) appearing in a path from a source node to a destination node. A variable-length chromosome of length equal to the number of nodes for encoding the problem has been used to list up node IDs from a source node to a destination based on a topological database of a network. Another similar (but slightly diﬀerent) ﬁxed-length chromosome representation has been used, that is, each gene in a chromosome represents a node ID that is selected randomly from the set of nodes connected with the node corresponding to its locus number. The disadvantage with these direct approaches is that a random sequence of node IDs may not correspond to a valid path (that terminates on destination node without any loop), increasing the number

of invalid paths returned.

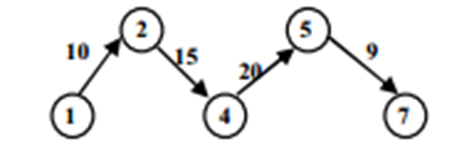
Let the graph be



After encoding a particle:



After decoding the particle to path:



An indirect scheme for chromosome representation scheme has been proposed by Genetal where instead of node IDs directly appearing on the path representation, some guiding information about the nodes that constitute the path is used to represent the path. During GA initialization, these priorities are assigned randomly. The path is generated by sequential node appending procedure beginning with the source node and terminating at the destination node, the procedure is referred as to path growth strategy. At each step of path construction from a chromosome, there are usually several nodes available for consideration and the one with the highest priority is added into path and the process is repeated until the destination node is reached. For eﬀective decoding, a dynamic node adjacency matrix is maintained in the computer implementation and is updated after every node selection so that a selected node is not a candidate for future selection. One main advantage of this encoding is that the size of the chromosome is ﬁxed rather than being variable (as in direct encoding) making it easier to apply various operators like mutation and crossover. One disadvantage is that the chromosome is “indirectly” encoded; it does not have important information about the network’s characteristics like its edges’ costs. Actually this coding is quite similar to random number encoding used for graph tree representation in genetic algorithms.

Another variant of indirect coding of the chromosome is called weighted encoding. Similar to the priority encoding, the chromosome is a vector of values called weights. This vector is used to modify the problem parameters, for instance the cost of the edges. First, the original problem is temporarily modiﬁed by biasing the problem parameters with the weights. Secondly, a problem-speciﬁc non evolutionary decoding heuristic is used to actually generate a solution for the modiﬁed problem. This solution is ﬁnally interpreted and evaluated for the original (unmodiﬁed) problem.

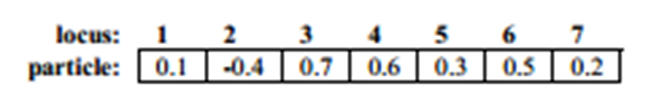
5.2) Cost-priority-based particle encoding/decoding.

Inspired by the above encoding schemes, a representation scheme, called cost-priority-based encoding/decoding, is devised to suit the swarm particles. Note that direct encoding is not appropriate for the particles as the particle updating uses arithmetic operations. In the proposed scheme, the particle encoding is based on node priorities and the decoding is based on the path growth procedure taking into account the node priorities as well as cost of the edges. The particle contains a vector of node priority values (particle length= number of nodes). To construct a path from a particle, from the initial node (node 1) to the ﬁnal node (node n), the edges are appended into the path consecutively. At each step, the next node (node j) is selected from the nodes having direct links with the current node such that the product of the (next) node priority (pj) and the corresponding edge cost is minimum, that is,

       j = min{CijPj |(i, j)∈E } , Pj  ∈ [−1.0 , 1.0].

The steps of this algorithm are summarized in Algorithm. The node priorities can take negative or positive real numbers in the range [−1.0,1.0]. The problem parameters (edge costs) are part of the decoding procedure. Unlike the priority encoding where a node is appended to the partial path based only on its priority, in the proposed procedure, a node is appended to the path based on the minimum of the product of the node (next node) priority and the edge cost that connects the current node with the next one to be selected. Experimental results show superiority of this procedure over the priority encoding when it is implemented within PSO frame. The PSO-based search is performed for optimal set of node priority values that result in shortest-path in a given network. An example of the execution steps of the cost-priority decoding for path construction for the network of Figure. It also compares the path construction from the same particle with simple priority decoding high lighting the advantage of the new approach.

After encoding a particle:



Pseudo code for decoding a particle to path

// i is the source node

// j is an adjacent node to i

// n is the destination node, 1=source node

// A(i) is the set of adjacent nodes to i

// PATH (k) is the partial path at decoding step k

// pj is the corresponding priority of node j in the particle P (position vector X)

// N∞ is a speciﬁed large number

Particle Decoding (P)

i←1,

P1 ←N∞

k←0,

PATH(k)←{1}

while ( { j∈A(i), Pj != N∞ } !=∅ )

k←k+1

j←argmin{Cij Pj  | j∈A(i), Pj  !=N∞}

i← j,

PATH (k)←PATH(k)∪{ i}

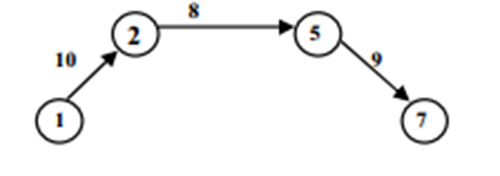
pi ←N∞

if i=n then return the path PATH (k)

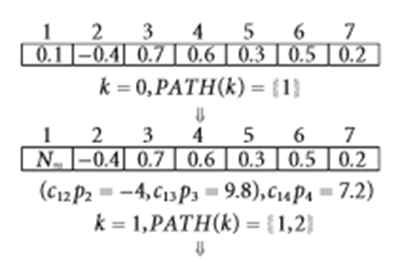
end while

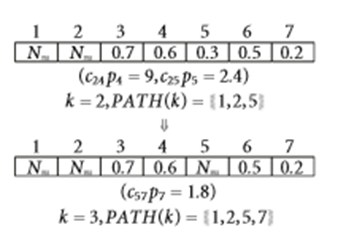
return Invalid Path

After decoding the particle to path:



Steps of path generation:





So the Path generated is path = {1,2,5,7}

5.3) Trust value Mechanism

From the perspective of multi-objective optimization, trust path-searching algorithm based on PSO search a group of multi-objective Pareto non inferior solution from the source node to the target node. The main idea is: after initializing the particle swarm, each particle can update its speed and location according to its information, and then produce a new particle with better value. Doing that process continuously, implementing the global search of the space, finally, we can get a better overall value, which is the better trust path in the networks.

In the graph, V is a set of nodes, E is a set of edge. tr is the trust value from j to k, where s V , tV , s is the source node, t is the target node, M is the max hops of this path, there are m nodes in trust path:

s t , m<=M

The trust value of that path:

) = tr

In which is a edge set of .For trust path, the bigger the value of its fitness, better the path is.

5.4) Main idea of the Algorithm

Here we are taking a swarm of particle where each particle is initialized randomly in the beginning i.e. its velocity and position is initialized randomly. Here the particle is nothing but a path from src to dest. So in each iteration we are calculating the fitness value of each particle (which is the trust value of the path) and updating the Personal Best of that particle .we are also updating Global Best as the best particle found till now. At last we are updating each particle velocity and position. At the end of all iteration we are declaring the global best as our optimal trust path. It is also possible to find second optimal trust path by keeping track of second global best in each iteration and so on. We can also keep a limit on number of nodes for the optimal trust path.

5.5) The design of algorithm

Pseudo code for the optimal Trust path selection

gBest ← NIL

Global\_cost ← ∞

PATH ← NIL

For each particle i

//Initialize the particle population (priority and velocity vector for each particle) randomly.

Initialize Xi randomly from [−1.0,1.0]

Initialize Vi randomly from [−1.0,1.0]

//Evaluate fitness of each particle

Evaluate fitness(Xi)

// construct the path form particle priority vector

PATH ← Particle Decoding(Xi)

fitness(Xi) ← cost (PATH)

//Calculate pBest and gBest for each particle

pBesti = Xi

Pi = fitness(Xi)

if fitness(Xi) < Global\_cost

gBest = Xi

Global\_cost = f(Xi)

Iteration\_count←0; // max iteration is the speciﬁed maximum number of iterations

While (Iteration\_count < max iteration)

For each particle i

Update Vi according to (1)

Update Xi according to (2)

//Evaluate fitness value of each particle

Evaluate fitness(Xi)

PATH ← Particle Decoding(Xi)

fitness(Xi) ←cost (PATH)

//Calculate pBest and gBest for each particle

if fitness(Xi) < Pi

pBesti = Xi , Pi = fitness(Xi)

if f(Xi) < Global\_cost

gBest = Xi , G = fitness(Xi)

Iteration\_count ← Iteration\_count+1

end While

PATH ← Particle Decoding(gBest)

return PATH

This is the algorithm for optimal trust path selection .Here we are using cost based encoding method to encode a path and decoding method to decode a particle to path. When we have to calculate the trust value of a particle then first we will decode that particle to path and then find the trust value of the path using the Fitness Function fitness().

First we are initializing the particle randomly. Then for k no of iteration we are finding the fitness value of each particle and updating the Pbest accordingly and then finding the Gbest for whole particle and at last updating the position and velocity of particle. This process is run k no of times where k is the no of iteration. At Last the Gbest is our optimal solution .so we will decode the Gbest to find the global best path which is the optimal one.

5.6) Coding strategy

We have three classes in the code

1) graph.py

2) particle.py

3) pso.py

1. graph.py

It basically implement the network part. Here network is seen as graph with n no of nodes and k no of edges with each edge having some weight which is the tust value between the nodes.

It contains 3 function

* Init()
* Add\_edge()
* Print\_graph()

Init() basically initialize the graph with n no of nodes.

Add\_edge(src,dest,weight) adds a edge from src to dest with edge weight weight

Print\_graph() just print the graph

1. particle.py

It implements a particle in the pso which is the basic of pso.A particle can have following attribute

* Position vector
* Velocity vector
* Position of Pbest
* Fitness of Pbest
* No of nodes(size of the position and velocity vector)

It contains 5 function

* Init()
* Update\_velocity()
* Update\_position()
* Reinitialize()
* Print\_particle()

Init() simply initialize the particle with random position and velocity

Update\_velocity() updates the velocity of particle according to given formula

vid = \*vid + j1\*rnd()\*(pid-xid) + j2\*rnd()\*(pgd-xid);

Update\_position() updates the position of particle according to given formula

xid = xid + vid;

Reinitialize() reinitialize the particle velocity and particle. It is called when particle get diverted from the solution

Print\_particle() print the particle position vector

1. pso.py

It implement the pso algorithm for finding optimal trust path in the network graph. The Pso class have following attributes

* swarm[] which the array of particle
* position vector of gBest
* fitness value of gBest

It contains 4 function

* Init()
* Cost\_fun()
* decode()
* Main()

Init() initialize the swarm[] array which is a array of particle. Then it runs the algorithm which is discussed in design of Algorithm .Also if a Particle is not giving any correct path for let k no of iteration then reinitialize the particle.

Cost\_fun(path) is calculating the fitness value of path according to trust mechanism in section 5.3

Decode(Position,start\_node,end\_node)

Here Position is the position of particle to decode. We are using cost priority based decoding method here to decode the particle. Cost priority based decoding and its pseudo code is discussed in section 5.2

Main()

Here we are initializing the pso algorithm and finding optimal trust path between start\_node and end\_node in a network graph.

Graph is also initialized here only.

1. Function dataset

We have tested this algorithm with 5 data sets with different no of nodes.

We have also computed the time taken by pso by different data sets and accordingly using this datasets for different iteration and

|  |  |  |  |
| --- | --- | --- | --- |
| Datasets | No\_of\_nodes | Average Time taken | Type of Graph |
| Nork | 500 | 50 sec | Not dense |
| Advogato | 6000 | 150 sec | Not dense |
| Freemans | 46 | 20 sec | Dense |
| Wolf | 20 | 15 sec | Dense |
| Zachart | 35 | 13 sec | Not dense |

1. SIMULATION RESULTS and Performance Evaluation

7.1) Efficiency Analysis

In order to analyze and understand the performance of the proposed algorithm the time taken by pso is recorded with

* varying no of nodes
* varying no of particle
* varying no of iteration

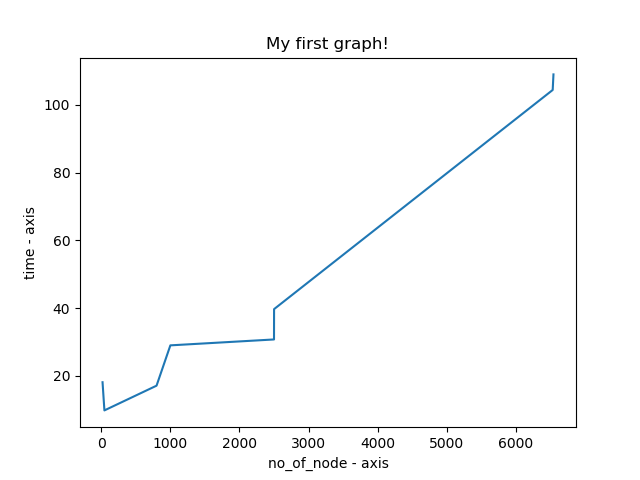
a) With Varying no of nodes

Here no of particle and no of iteration is fixed i.e.

No of particle  = 50 and

No of Iteration = 50

So we are varying the no of nodes in the graph keeping the no of iteration and no of particle constant.



b) With Varying no of particle

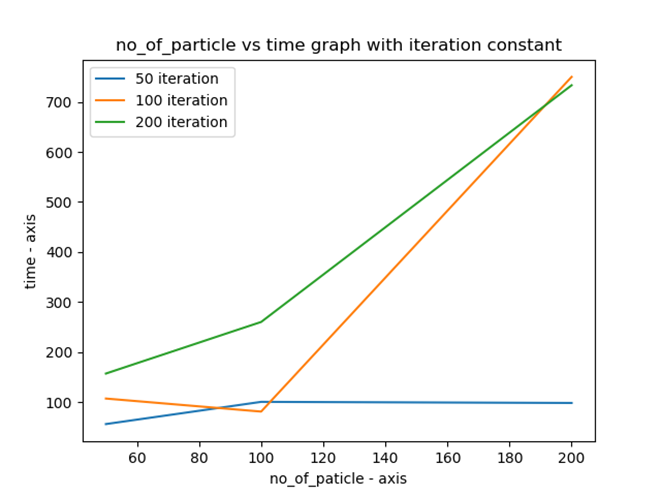
Here no of nodes and no of iteration is fixed i.e.

So we are varying the no of particle in the same graph keeping the no of iteration constant

Here 3 graph is plotted with no of iteration

* 50
* 100
* 200

For each graph we can see that time taken by pso increases with increasing no of particle. Also for each graph no of iteration is fixed and graph with 200 iteration is upper than graph with 100 iteration .so here we can conclude that if no of iteration increases than time taken by pso also increases. Also the slope of graph is high so the time taken by algorithm increases fast with no of iteration.



b) With Varying no of Iteration

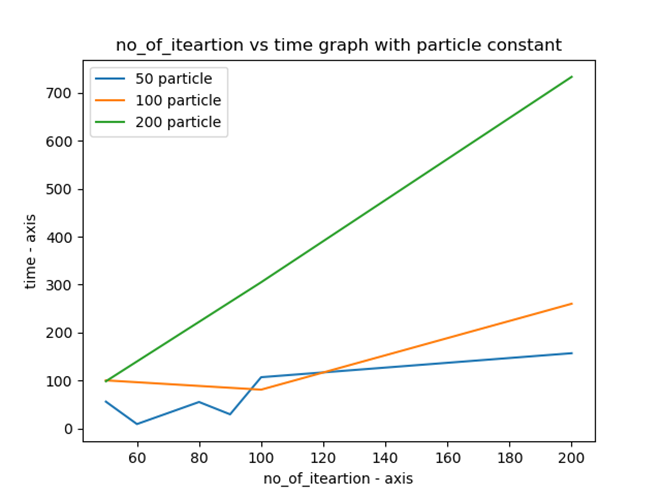
Here no of nodes and no of particle is fixed i.e.

So we are varying the no of iteration in the same graph keeping the no of particle constant

Here 3 graph is plotted with no of particle

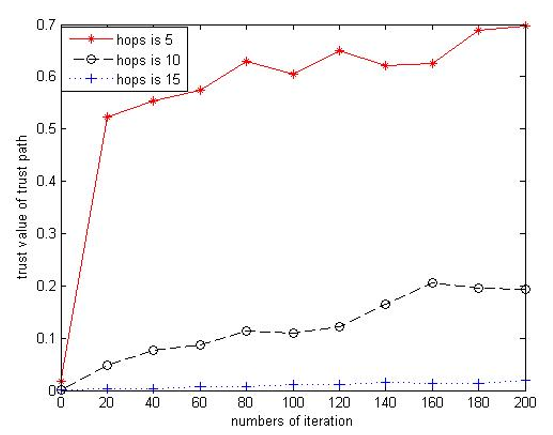
* 50
* 100
* 200

For each graph we can see that time taken by pso increases with increasing no of iteration. Also for each graph no of particle is fixed and graph with 200 particle is upper than graph with 100 particle.so here we can conclude that if no of particle increases than time taken by pso also increases.

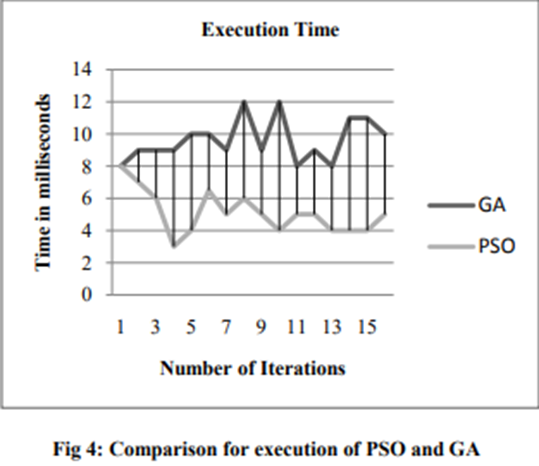


7.2) Influence of Hops

In this algorithm, we set the max hops from source node S to target node t .Below graph shows the relation of iteration and trust value of trust path, while the swarm scale is 100, hops is respectively 5, 10,and 15. As is shown in graph, the curve marked when hops is 5 is at the top, while the curve marked with 15 hops is at the bottom. That is to say, the fewer nodes the path passes, the more reliable the trust path is.



7.3) PSO Vs Genetic Algorithm



It can be seen that time taken by pso is less than genetic algorithm. So Pso is more efficient that genetic Algorithm for optimal trust path selection.

1. Conclusion

We propose an optimal trust path selection algorithm based on particle swarm optimization algorithm, and this algorithm is suitable for such small-world networks as P2P network. This algorithm uses PSO to search the global non-inferior solution through implementing lots of loops, namely the better trust path which is different from others. To some extent, it can prevent joint fraud effectively. It is also better than other algorithms in the performance and reliability, fit for the complex network environment. It can adapt to the dynamic changes of a complex network environment. It is likely to improve the experimental eﬀectiveness if improve the hardware environment, increase the number of particles and iterations, and adjust the parameters of the PSO algorithm.

Also a new cost-priority-based particle encoding/decoding scheme has also been

Devised so as to incorporate the network-speciﬁc heuristic information in the path construction process. As the iteration improves the particles reaches much better fitness value than the fitness of chromosome of GA.

It is also observed that time taken to find the shortest path is less in PSO than GA

It might take less computational time than Dijkstra’s algorithm when no of nodes is huge (10^20 nodes)

If no of particle and iteration is greater than 5% of the no of nodes, then there is more than 90% probability to get closer to optimal solution

1. Limitation

However, the algorithm makes the data fall into local optimal easily when applied to discrete data, which is less than ideal when searching the optimal path

1. Future Scope

Our future work will includes investigating more reasonable and ﬁt utility function and choosing sophisticated algorithm parameters. Another direction for us is to integrate our PSO-TPS algorithm into such trust management system as Keynote.

We can also use PSO in routing problem.

In shopping site like amazon suppose a product is given review. So the trust value of a particular review can be calculated by using pso.

Similarly trust value of a movie review can also be calculated

We can also determine if a person can be trusted or not in social networking site like Facebook.

REFERENCES

* Ammar W. Mohemmed and Nirod Chandra Sahoo "Efficient Computation of Shortest Paths in Networks Using Particle Swarm Optimization and Noising Metaheuristics "Discrete Dynamics in Nature and Society Volume 2007.
* [Kennedy, J. and Eberhart, R. (1995). “Particle Swarm Optimization”, *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942-1948, IEEE Pres
* Abhishek Toofani "Solving Routing Problem using Particle Swarm Optimization "International Journal of Computer Applications (0975 – 8887) Volume 52
* Kavitha Sooda, T. R. Gopalakrishnan Nair, A Comparative Analysis for Determining the Optimal Path using PSO and GA, International Journal of Computer Applications