Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

Metaheuristic Recursive Graph Routing based on Collective Intelligence Algorithms Inspired by Birds Behaviors

Mohsen Taki[1], Kamal Mirzaie*[2], Mohammadreza Mollahoseini Ardakani[3]

Abstract: In the realm of graph routing, various deterministic algorithms such as Floyd and Dijkstra have been instrumental in achieving specified goals. However, as graphs grow larger in size, the torch is being passed to meta-heuristic algorithms like genetics and particle swarm optimization. These innovative approaches address the challenges posed by extensive graphs, particularly in terms of routing time and loop prevention. Distance estimation and fitness functions have become pivotal elements in contemporary algorithms, significantly contributing to the reduction of route duration. Nonetheless, the persistent challenges of graph routing remain in handling loops and optimizing routing time efficiently. A groundbreaking solution to these challenges comes in the form of a novel algorithm inspired by the legendary bird, the Phoenix. In this algorithm, the history of nodes is inherited by their offspring. During each reproductive cycle, the algorithm ensures that the developed child, while fitting on offspring and selecting the stronger one, does not reintroduce itself to its own history, effectively preventing loops.

This method outpaces existing routing techniques in terms of speed and scalability, primarily attributed to a paradigm shift in creating the adjacency list. The algorithm's unique approach, inheriting node history to offspring and employing a stringent loop avoidance mechanism, results in faster performance and extends support to larger graphs. The algorithm is anticipated to achieve a desired performance level expressed as (n/b)d, where 'b' represents the fitting effect coefficient and 'd' signifies the graph's depth. With its innovative features and efficient loop prevention mechanism, this algorithm presents a promising avenue for enhancing graph routing in large-scale networks.

key words: bird's behaviors, collective intelligence, fitness function, graph routing, meta - heuristic, optimal path, optimization, parallel routing.

1 Introduction:

In contemporary times, the need to conduct searches within graphs has become paramount for various applications, especially in road routing, social networks, and dynamic complex networks aimed at discovering relations. The prominence of new network

[1]Department of computer Engineering,Maybod Branch,Islamic Azad University,Maybod,Iran.
taki_mohsen@yahoo.com
[2]Department of computer Engineering,Maybod Branch,Islamic Azad University,Maybod,Iran.
kamal.mirzaie@iau.ac.ir*
[3]Department of computer Engineering,Maybod Branch,Islamic Azad University,Maybod,Iran
Mr.mollahoseini@iau.ac.ir

Power System Technology
ISSN:1000-3673
Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

routing and the significant support for the Internet of Things (IoT) are increasingly apparent[5]. With the daily expansion of information and the imperative to reduce response time, there is a growing demand for optimal utilization of existing hardware to enhance system efficiency [11].

In response to these challenges and the quest for optimal results, there is a pressing need to invent methodologies that increase efficiency and lead to optimal answers in shorter time frames. This article aims to introduce an innovative method for graph routing, leveraging hardware power while simultaneously providing solutions that are as close to optimal as possible.

The method employed in this research to address the challenges of loops and deadlocks draws inspiration from bird behavior algorithms. It involves carrying the history of DNA to offspring, where the "DNA" here signifies the history of visited nodes. The process includes converting each node into multiple processes, assigning history, and managing the execution of each route through processes or their recursive execution. This approach substantially enhances parallelism within the graph.

Unlike SMA*, which relies on an initial memory range[13], this method begins without a default starting point, offering greater flexibility. The article commences with a comprehensive review of the research background in the field, providing a comparative analysis of existing general methods in a table format. The subsequent section details the algorithm, including its pseudo-code, flowchart, and an example to enhance understanding.

To evaluate the algorithm's performance, the article presents tests conducted to measure time, determine algorithm efficiency, and compare it with established algorithms such as genetics and particle swarm optimization, with these differences illustrated through charts. Finally, the article concludes with the presentation of results and suggestions for future exploration.

The background section categorizes existing graph routing algorithms into two main types: deterministic and heuristic (non-deterministic). Deterministic algorithms, including Dijkstra, Bellman-Ford, A*, Floyd-Warshall, and Johnson's algorithm, are briefly described, highlighting their ability to find optimal solutions accurately [5].

In the subsequent sections, the article delves into the specifics of each deterministic algorithm, detailing their applications and complexities. The discussion encompasses critical parameters such as the number of vertices (V), number of edges (E), branching coefficient (b), and depth of optimal solution (d). The distinctive features of algorithms like Dijkstra, Bellman-Ford, A*, Floyd-Warshall, and Johnson's algorithm are elucidated [5].

The article then explores various heuristic algorithms, their classifications, and the challenges they present, such as getting stuck in local optimal points and premature convergence. Meta-heuristic algorithms, including genetic algorithms, ant colony algorithms, and bee algorithms, are introduced as solutions to overcome these challenges.

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

The section on genetic algorithms traces their inception by John Holland in 1960, emphasizing their application in solving the shortest path problem. The work of researchers like Mitsuo and Sen [29], who used genetic algorithms for routing, is also discussed, offering insights into the development of routing and coding paths within a graph.

Several intelligent shortest path algorithms are presented, including ant colony optimization, forbidden search, and collective optimization algorithms. These algorithms showcase advancements in optimizing routing efficiency and present viable alternatives for various applications.

The article concludes with discussions on recent approaches using particle swarm optimization (PSO) and modified coding techniques to reduce loop formation probability in path construction. Simulation experiments, comparisons between different algorithms, and proposals for new approaches are presented, providing a comprehensive review in graph routing.

In summary, this article not only introduces an innovative graph routing algorithm but also provides a thorough exploration of the existing landscape, offering readers a nuanced understanding of the challenges and solutions in the dynamic field of graph routing.

2- background: Existing algorithms for graph routing are classified into two categories: deterministic and heuristic (non-deterministic). Deterministic algorithms are able to find optimal solution accurately, and their execution time increases according to problem dimensions [5].

Deterministic traversals in graphs start from basic traversals such as Dijkstra to find best path between two points or other points. Algorithm of shortest paths from single origin (Dijkstra) uses an idea similar to first level search. Other methods have been invented and applied depending on various applications and what exactly problem is pursuing [11].

In following methods, V is number of vertices, E is number of edges, b is branching coefficient, and d is depth of optimal solution.

Among deterministic algorithms to solve this type of problem are:

Dijkstra Algorithm: solves problem of finding shortest path between two vertices, from a single origin to a single destination in O(|V||V|) time [5].

Bellman-Ford Algorithm: It solves problem of finding shortest path from single origin in a case where edges weight can be negative. Its execution time is O(|V||E|) [5].

A* search algorithm: with help of innovative search methods, it accelerates answer of finding shortest path between two vertices in O(bd) [11].

Floyd-Warshall algorithm: solves problem of finding shortest path between two vertices in O(|V||V||V|)[5].

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

Johnson's algorithm: solves problem of finding shortest path between two vertices and in sparse graphs, it may work faster than Floyd-Warshall. This algorithm has an execution time of O(V2log V +VE) [5].

In many researches, path is calculated by considering minimum path. first concept of conflict-free road routing used Dijkstra's shortest path algorithm to generate a matrix [28].

Dijkstra algorithm is a shortest path algorithm used to solve single-source shortest path problem when all edges have non-negative weights. Dijkstra’s symmetric algorithm was invented by Pole [23], which was derived by implementing two-way search method. process of symmetric Dijkstra algorithm was similar to original algorithm, which included forward search (from source node to destination node) and backward search (from destination node to source node). algorithm process ends when forward search and backward search meet at a certain node. According to Pole, this algorithm was invented in an attempt to reduce complexity of Dijkstra implementing algorithm. But in worst case, algorithm implementation complexity can be converted to two O(bd) searches [23].

sundari presents a new technique in Dijkstra routing algorithm. In this method, spent time by this algorithm is calculated as a combination of input intervals. accuracy of this method is in problem of shortest path and problem of finding a path in a graph between two vertices or nodes to minimize total weight of its constituent edges. In existing approach, it identifies a number of nodes by opening graph and calculates fastest path between it and most other nodes [22].

Ahrens improved a goal-oriented version of Dijkstra's algorithm to find shortest paths in large graphs. He provided distance-to-goal estimates that provided a better correlation between execution time and quality of path found than previously known methods, leading to an overall algorithm speedup [1].

A* algorithm was invented by Hart and Nilsson in 1968, where algorithm implements concept of integrating a heuristic into search process. A* algorithm works similarly to Dijkstra algorithm except for its different heuristic controls in node selection for each iteration. Instead of selecting node with shortest distance from start node, A* algorithm selects node based on its distance path from start node by heuristically estimating its proximity to destination node. The heuristic estimation is evaluated by one of two main evaluation functions which were Euclidean distance and Manhattan distance [13]. Zhang's A* algorithm has reduced search space required to reach destination node compared to Dijkstra's algorithm. This shows that A* algorithm will perform better compared to Dijkstra algorithm, unless its heuristic function is less accurate [32].

Bellman-Ford algorithm is suitable when graph has edges with negative value to solve shortest path problems. This algorithm works iteratively; number of repetitions is based on number of

Power System Technology
ISSN:1000-3673
Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

edges on path from start node to destination node. For each iteration, each of last visited nodes expands to its nearest node [6].

Floyd-Warshall algorithm [23] works by finding shortest distance path between all pairs of nodes in a graph. In addition, Floyd-Warshall algorithm is one of few algorithms that can solve problem in a graph that contains negative values of edges and without existence of a cycle of negative edges. main advantage of Floyd-Warshall algorithm is that it can obtain shortest distance between any two arbitrary nodes [31].

Shortest path search is an effective way to find shortest path in heavy traffic congested areas by pre-processing previous and current traffic data with different types of inputs and identifying shortest path through graph theory. In Shortest route*, start node focuses on finding closest target and then processes each step. main advantage of shortest path search algorithm is that it does not waste time to spend on any other unwanted nodes [4].

Lewis investigates problem of finding shortest paths in graphs with additional penalties (additional transmission costs) at their vertices. He proposes a shortest path algorithm that can solve problem without having to perform a graph expansion, which is a common algorithmic strategy. While his method has a lower growth rate compared to existing approaches, it shows that his method is more efficient in sparse graphs [17]. Lakshna presents a new deterministic route planning, collision risk monitoring and collision avoidance method for a ship. He suggests sailing. This method is based on a modified version of Dijkstra's algorithm. It combines conventional weather routing with collision risk monitoring and collision avoidance features where edge weights change over time, and some edges may be temporarily blocked due to weather conditions [4].

Another general category for graph traversal algorithms is heuristic or uncertain algorithms. Heuristic algorithms are able to find good (near optimal) solutions in a short time for hard optimization problems. Heuristic algorithms are also classified into three categories: heuristic, meta-heuristic, and hyper-heuristic. In heuristic algorithms, a trick is tried to be used so that entire space of states is not searched completely. In other words, by using a specific heuristic, search scope is limited to reach answer [20]. A heuristic is a problem estimate that tells us how close each existing state can be to solution, and more accurate this estimate can increase algorithm's performance. Hill climbing and *A algorithms are of this type. two main problems of heuristic algorithms are their getting stuck in local optimal points and premature convergence to these points. Designing for special issues is also one of their characteristics. Meta-heuristic algorithms are presented to solve such heuristic algorithm problems. In fact, meta-heuristic algorithms are one types of heuristic optimization algorithms that have solutions for exiting from local optimal points [20]. Meta-heuristic algorithms significantly increase ability to find high-quality solutions for hard optimization problems. common feature of these algorithms is use of local optimization exit mechanisms. Meta-heuristic algorithms are divided into two general groups of break-answer and population-based methods. Algorithms based on

Power System Technology
ISSN:1000-3673
Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

break answers return one answer during search process, while population-based algorithms consider a population of answers during search. Algorithms based on solution break focus on local search areas, on other hand, algorithms based on population can perform search simultaneously in different solution areas space [31]. Algorithms such as genetic algorithm, ant colony algorithm, bee algorithm are of this type.

Genetic Algorithm is a meta-heuristic algorithm invented by John Holland in 1960 [14]. It was then developed by him and his students and colleagues at University of Michigan in 1960s and 1970s[24]. This intelligent algorithm was invented to solve shortest path problem in a flexible situation that has a very large search space and a constantly changing environment [15]. Mitsuo used genetic algorithm to solve shortest path problem, during which, main work for genetic algorithm was how to develop routing and coding a path in a graph in form of a chromosome [19].

Sen examines shortest path algorithms, focusing on graphs of roughly similar length and distance. Samaher's research also introduces class of transportation network algorithms and includes algorithms for general graphs as well as planar and complex graphs [29]. In contrast to shortest path optimization algorithms, Fu explores algorithms that target shortest path heuristics to quickly identify shortest path. goal of heuristic algorithms is to minimize computation time. This review suggests main distinctive heuristic algorithm features as well as their computational costs [10].

Xu made some simple modifications to original Dijkstra shortest path algorithm to obtain an improved algorithm for finding shortest path for a sparse network [30]. Wang Shu introduced Dijkstra label algorithm and stated that algorithm should be improved, then proposed an improved algorithm that is more effective than label algorithm and can solve shortcomings of label algorithm [25]. Garg presented a dynamic optimization for Dijkstra algorithm, which helps to effectively solve dynamic source shortest path problem [33].

Today, there are also several intelligent shortest path algorithms that have been used in research papers. In an article, Attirata analyzed continuation time of ant colony optimization algorithm to find shortest path and presented his results and achievements in this field [3]. Ghoseiri also used ant colony optimization algorithm to solve shortest path problem with two objectives. Compared with label correction algorithm, this algorithm was able to provide answers with higher quality and in a faster time [12]. In another study, kuri used meta-heuristic algorithm of forbidden search for routing [16]. Mohemmed, in his article, solved problem of finding shortest path in network by using collective optimization algorithm of particles [20].

There are many deterministic algorithms for solving shortest path problems in static topologies. However, in dynamic topologies, these deterministic algorithms are not efficient due to need for restarts. Amar presents research on application of particle swarm optimization (PSO) to solve shortest path (SP) routing problems. A modified priority-based coding including a

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

heuristic operator is proposed to reduce probability of loop formation in path construction process for particle representation in PSO.

Simulation experiments have been performed on different network topologies for networks consisting of 15 to 70 nodes. It is pointed out that proposed PSO-based approach can find optimal path with a good success rate and can also find closer sub-optimal paths with high confidence for all tested networks. He also observed that performance of proposed algorithm surpasses approaches based on genetic algorithm for this problem [2].

Chan's main goal was to investigate and test various shortest path algorithms used in navigation system, such as Dijkstra's algorithm, Dijkstra's symmetric algorithm, A* algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm and genetic algorithm. In solving shortest path problem, results showed that Bellman-Ford algorithm performance is superior to other algorithms in most situations. Therefore, authors have concluded that Bellman-Ford algorithm is most efficient shortest path algorithm compared to other algorithms. This research also shows that performance of genetic algorithm is affected by generations number, in which higher number of generations, longer execution time and better solution. Therefore, it is very important to adjust generations number until ratio of execution time to optimal solution is obtained, so that genetic algorithm can be used in most efficient mode [7].

Omar has proposed a new approach to calculate shortest multi-faceted paths. Results showed that success rate of new approach in terms of convergence to optimal or near optimal solutions is much better than a basic genetic algorithm. Moreover, unlike traditional algorithms such as Dijkstra, proposed new approach is fast enough for practical routing applications [9].

Mensah proposes an almost efficient and accurate algorithm that is applicable to large-scale networks. Algorithm iteratively creates levels of hierarchical networks with a node-dense method to construct hierarchical graphs up to threshold. Experiments on real data show that algorithm records high efficiency and accuracy compared to other algorithms [8].In general, inconsistencies can be reduced by minimizing number of traversals for arcs or edges by changing paths to be taken between service points or by changing order of services [11] .

Technique	**Algorithm**	**power**	**weakness**
deterministic	A* Dijkstra Bellman Ford Johnso	Safe and definite conditions. Optimization	Dynamic conditions or uncertainty are not appropriate.

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

	n Floyd	includes static distances, cost and defined constraints	Several criteria and scenarios cannot be calculated effectively.
non-deterministic	Genetics Ant colony	Anticipation, reasoning or adaptability Optimization of dynamic traffic situations and events	Better performance, especially when there are different scenarios, needs to be combined with other methods.

Table 1- Comparison of several deterministic and non-deterministic algorithms in terms of strength and weakness [5].

3- Algorithm and routing method

The Phoenix generation method, inspired by the legendary Iranian bird, serves as the foundation for the generation process in this algorithm. In this approach, each node transmits its history to its offspring before being destructed, laying the groundwork for the production of the next generation. Each child follows the same path as its parents until reaching the goal. If a child revisits a node in its history, it prunes itself and the rest of its generation. Weaker children, identified by having a weight higher than the average weight of their siblings, are pruned without selection during fitting.

In this algorithm, the graph is represented as a list similar to an adjacency list but in a three-dimensional form for easy access to required information. The fitness function in this method is associated with each parent node. To develop its children, the parent node averages twice with respect to all its output edges and prune edges with a weight higher than the average. This process similar to birds removing weaker offspring or problematic eggs from the nest during difficult conditions, ensuring that paths with a lower chance of reaching the goal are pruned

Power System Technology
ISSN:1000-3673
Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

early. The fitness function is responsible for selecting generations that are more suitable and superior to other paths.

The algorithm faces a multitude of possible paths for development, leading to a substantial initial population. The fitness function prunes paths by selecting those with a higher probability of reaching a solution. Pathways progress locally and expand one generation at a time. Each generation inherits a string of history from its parents, eliminating the need to go back to the start for each generation.

The algorithm introduces three functions for reproduction: fitting, minimum inhibition (with the biggest possible weight), and repetition inhibition (loop prevention). The stopping condition for the algorithm is the impossibility of generation. Reproduction involves the fitness function selecting candidates from paths ahead. The output of the fitness function serves as input for inhibition functions, and if both functions grant reproduction permission, a new generation is generated, and history is transferred to them.

The biggest possible weight is a value approximating the actual distance from the start node to the destination, ensuring it is greater than or equal to this distance. This value acts as an upper limit, blocking paths incompatible with new conditions or identifying weaker children. A good upper limit can be predicted from the beginning by using functions to estimate the distance between the start node and destination. Still, it can also be dynamically adjusted during execution based on the success of other paths in achieving results with lower weights.

The algorithm operates in three modes depending on the initial values and method selection variable: finding the shortest path between origin and destination, finding all possible paths between origin and destination, and finding all possible paths between origin and destination with a weight less than or equal to a defined maximum value.

This algorithm accommodating directed and undirected, weighted and unweighted graphs. It supports zero weight and cannot support negative weight graphs. For unweighted graphs, the algorithm considers the distance between each level to be 1. The input is the adjacency list of the desired graph, and the output falls into one of the three mentioned modes.

In conclusion, this algorithm aims to obtain the shortest possible path or all possible paths, offering high parallelism and optimization during execution. While it may outperform its peers in speed for finding all possible ways, its strength lies in the accuracy of its answers when seeking the shortest path. If a path leads to the production of a path greater than the final estimated cost function, it is pruned, and the continuation of the path is postponed to other paths. The matrix formation in this algorithm deviates from the usual method (adjacency matrix) to accommodate changes, making it suitable for both directed and undirected graphs.

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

Algorithm:

```
phoenix(history, node, weight)
begin
 if active node=goal and pathlenght<max then
 max= pathlenght
Way=history of visited nodes
 End if
for  i = 0; i<active node childs and next node has child  do
    if  next node isnt in history
    set parameters for develop
     phoenix(history, nextnode, tempweight)
next
end
```

Code 1-Phoenix algorithm

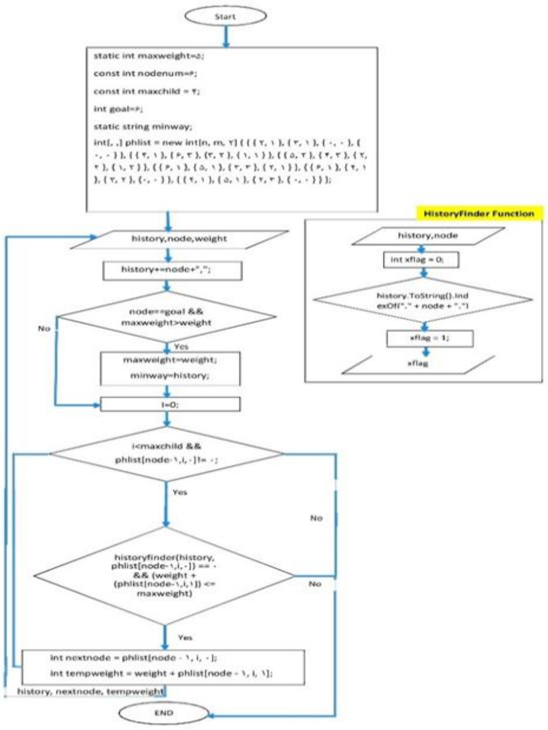
Start
static int maxweight=۵;
const int nodenum=۶;
const int maxchild = ۴;
int goal=۶;
static string minway;
int[, ,] phlist = new int[n, m, ۲] { { { ۲, ۱ }, { ۳, ۱ }, { ۰, ۰ }, { ۰, ۰ } }, { { ۲, ۱ }, { ۶, ۲ }, {۲, ۲ }, { ۱, ۱ } }, { { ۵, ۲ }, { ۴, ۲ }, { ۲, ۲ }, { ۱, ۲ } }, { { ۶, ۱ }, { ۵, ۱ }, { ۲, ۳ }, { ۲, ۱ } }, { { ۶, ۱ }, { ۴, ۱ }, { ۲, ۲ }, { ۰, ۰ } }, { { ۴, ۱ }, { ۵, ۱ }, { ۲, ۳ }, { ۰, ۰ } } };
HistoryFinder Function
history,node
int xflag = 0;
history.ToString().IndexOf("," + node + ",")
xflag = 1;
xflag
history,node,weight
history+=node+",";
node==goal &&
maxweight>weight
No
Yes
maxweight=weight;
minway=history;
I=0;
i<maxchild &&
phlist[node-۱,i,۰]!= ۰;
Yes
No
historyfinder(history, phlist[node-۱,i,۰]) == ۰ && (weight + (phlist[node-۱,i,۱]) <= maxweight)
No
Yes
int nextnode = phlist[node - ۱, i, ۰];
int tempweight = weight + phlist[node - ۱, i, ۱];
history, nextnode, tempweight
END

Figure1-Phoenix algorithm flowchart

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

4- results Comparison and calculations

To ensure confidence in the comparison results, all algorithms discussed in this section were executed on a Dell4800 laptop with the following specifications: Windows 8 operating system, i7 processor (2 physical cores, 4 logical cores), 4M cache, 512K L2 cache, 128K L1 cache, and 16GB RAM. The dataset used for comparison was selected from the standard datasets of Stanford University, comprising 800 nodes and 19176 edges. Dr. Yinyu Ye, a professor at Stanford University's School of Engineering, provided this dataset, which apparently involves interactions among university students through personal emails, reflecting direct or mediated communications between students.

Initially, the dataset was converted into an adjacency list, and the performance of Dijkstra algorithms and some other available deterministic algorithms was tested to find connections between two nodes (the first and last node). Unfortunately, due to the limited power of the hardware used and algorithm limitations, no answers were obtained.

Genetic algorithms and Particle Swarm Optimization (PSO), as two meta-heuristic algorithms, were adjusted and tested on this dataset, with results detailed below.

In different executions, the algorithm organized and searched up to 70 thousand nodes based on changes in the style of creating an adjacency list, which was significantly higher than the 7 thousand nodes achieved by conventional proximity lists and existing meta-heuristic methods.

In these datasets, every edge was assigned a fixed weight of 1. To increase pruning and improve results, a fitness function using random weights was introduced, overlaying another list on the same graph where edges had different weights. This modification did not affect the accuracy of answers or path accuracy but increased the speed of obtaining answers by 30%.

Two effective functions in this method are fitness functions and target depth estimation. Both play a significant role in increasing pruning and reducing the speed of obtaining answers.

- fitness function

The fitness function is inspired by the behavior of birds towards weaker eggs and young. Weaker children, with a lower life expectancy, are pruned above the average limit. The operation is done once during the first visit of a node, and the averaging result is written in the first cell of the column related to the edge.

Target Depth Estimation Function:

This function has the greatest impact on increasing pruning and reducing operations in the algorithm. If the weight obtained from the traveled path exceeds the output value of this function, it leads to pruning the path. The output of this function (MaxW), when closer to the actual weight of the shortest path from the start node to the destination, significantly reduces the number of operations in the algorithm.

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

It should be noted that this number is modified by the algorithm, and if a route is found with a weight lower than MaxW, it is changed to a new value, and pruning is done based on this new value. Even routes larger than the new value are pruned in the current step. The estimation function can be obtained based on different mechanisms.

With the above explanations, the standard dataset of 800 nodes and 19176 edges was tested for Phoenix algorithms (with different modes), genetics (in two modes, normal and modified mutation), and particle swarm optimization (PSO). The output of the Phoenix algorithm was used as training input for genetic algorithm and PSO.

In addition to finding the optimal path, the Phoenix algorithm can display specific sub-maximal paths or a certain number of paths with a specific maximum path length in the output. Sixty routes with lengths below 30, below 20, and below 10 were entered as correct examples for genetic algorithm and PSO.

In the genetic algorithm, common data between two samples of the population were divided after initial tests, and resulting samples were used after matching to prevent the generation of duplicate populations. This mode dramatically increased the speed of obtaining results. Finally, the results of running the mentioned algorithms in different situations are shown in the table below. The results provided are the averages of 50 runs for each algorithm.

Table 2- Phoenix parameters setting

No	Initial Parameter	Value
1	**MaxW**(maximum allowed weight)	10 to 30
2	NN(Nodes Number)	800
3	M(Max Branch Factor)	32
4	Goal	NN-1
5	MinWay	null
6	Minw	0

Table 3- algorithms execution time Comparison

	Algorithm name	average weight	average Time(s)
A	Normal Genetic	4	13.5

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

B	Optimized Genetic	3	3.8
C	PSO	3	3.85
D	Parallel Phoenix MaxW=30	3	0.134
E	Recursive Phoenix MaxW=30	3	0.0175
F	Parallel Phoenix MW=10 and var Weights	4	0.057
G	Recursive Phoenix MW=10 and var Weights	4	0.0125

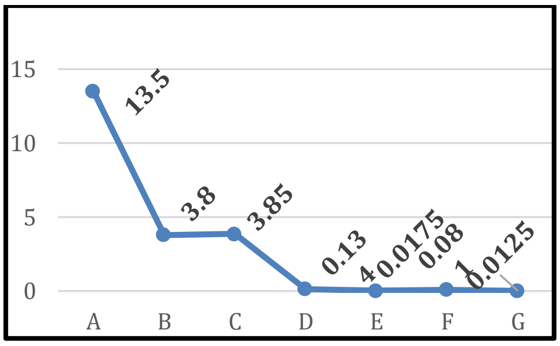

Figure2-Algorithm execution on dataset with 800 nodes

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

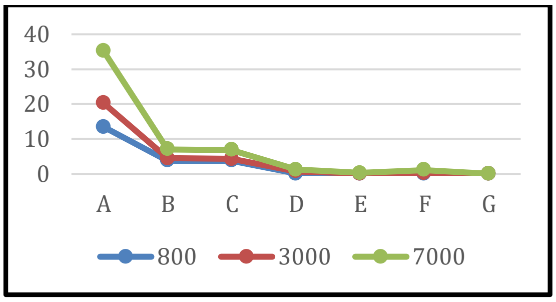

Figure3-Algorithm execution on different datasets

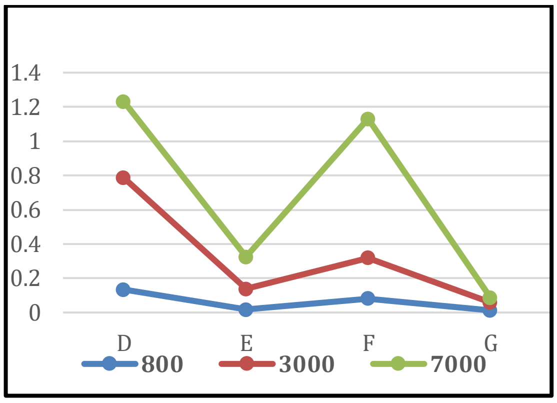

Figure4-Execution of different states of Phoenix algorithm

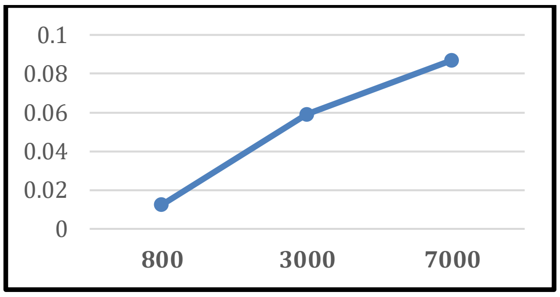

Figure5-Execution algorithm with closest Max

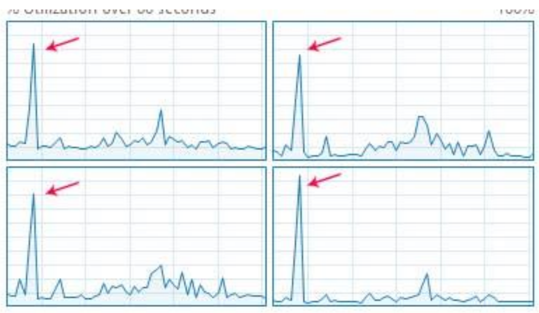

Figure6-processor activity Peak during algorithm execution

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

According to the results of tests and preliminary data, the following results can be considered as the execution time. If we consider the execution unit of an instruction as 1 for each node visit, or, in fact, we ignore side operations and consider each node visit as equivalent to executing one instruction, where the Phoenix algorithm is used without considering MaxW and fitting, it can be said that this algorithm has an order of O(n^2) in the worst case. Firstly, it is not possible to return to its history at each stage, and every path from the starting node can finally visit all nodes and cannot add a duplicate node to history.

Now, if we add MaxW as a cutting or pruning factor to the algorithm, knowing that the algorithm is limited to it and will not go further than that, and if we consider the weight of each step to be 1 in the worst case, if we imagine the best number for goal depth is equal to a small coefficient (c) multiplied by tree depth (d), which will not be more than that, the result is ndc because no return backward to history and adding history as visited path. But according to MaxW modification, after reaching the first solution, c will gradually tend to 1 and even smaller than that. With the mentioned interpretations, the algorithm in this case is an order of O(n log n).

If we add the fitness function to the mentioned algorithm and observe all previous conditions, according to all mentioned cases, n can be changed to half or even less because some of the children are not expanded. Even in the movement case among other nodes, because nodes with larger weight have development limitations, then practically (n/2)dc nodes will be met, and it tends to (n/2) log n, which in large graphs, this difference in result will be remarkable compared to the previous case. But in the worst case, it will be O (n log n).

5- Conclusion

Experiments have shown that deterministic graph routing algorithms are not capable of routing medium and large-scale graphs. Therefore, heuristic and meta-heuristic methods are used for such routing. However, in this research, the nature of the reversibility and parallelization of graphs by relying on carrying history to prevent more possible loops showed itself. It was found that if graphs have proper pruning and effective pruning functions, only by using the Phoenix Algorithm recursively, routing graphs can be much faster. Parallelization itself has a higher computational overhead than the recursive mode, and if you can work on graphs with the recursive mode, the result will naturally be more satisfactory.

In experiments with more than 70,000 nodes and 200,000 edges, both recursion and parallel recursion methods provided much faster answers than other algorithms. In the above methods, according to the presented graphs, it was shown that this algorithm is as active as possible in relation to using the processor, and complete graph segmentation has made the processor completely available to the algorithm. This movement plays a significant role in decreasing execution time. Of course, using the parallel method to visit several nodes at the same time will be more effective in routing that requires high computational efforts or complex processes to visit nodes.

Power System Technology
ISSN:1000-3673

Received: 16-05-2024 Revised: 12-06-2024 Accepted: 07-07-2024

Resourses

1) Ahrens, Markus, Henke, Dorothee, Rabenstein, Stefan, Vygen, Jens, Faster goal-oriented shortest path search for bulk and incremental detailed routing, Mathematical Programming, https://doi.org/10.1007/s10107-023-01962-4,2023.
2) Amar, D, Flandrin, E, Gancarzewicz, G, Wojda, A.P, Bipartite graphs with every matching in a cycle, Discrete Math, 2005
3) Attiratanasunthron, N, Fakcharoenphol, J, A running time analysis of an Ant Colony Optimization algorithm for shortest paths in directed acyclic graphs, Information Processing Letters, 105, 88-92, 2007.
4) Lakshna, S, Gokila, K, Ramesh, R, Smart Traffic: Traffic Congestion Reduction by Shortest Route * Search Algorithm, International Journal of Engineering Trends and Technology, Volume 71 Issue 3, 423-433, 2023.
5) Bhatia, Shaveta, Survey of shortest Path Algorithms, SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE), 2019.
6) Bellman, Richard, On a routing problem, Quarterly of Applied Mathematics, 16: 87–90, doi:10.1090/ qam/102435. MR 0102435, 1958.
7) Chan, Simon, Yew, Meng, An experiment on the performance of shortest path algorithm, 7-12, 2016.
8) Dennis, Nii, Ayeh, Mensah, Hui, Gao, Liang, Wei Yang, Approximation Algorithm for Shortest Path in Large Social Networks, Algorithms, 13, 36:10.3390/13020036, 2020.
9) Dib, Omar, Manier, Marie-Ange, Caminada, Alexandre, Memetic algorithm for computing shortest paths in multimodal transportation networks, Transportation Research Procedia 10:745-755, 2015.
10) Fu, Liping, Sun, D, Rilett, Laurence R, Heuristic shortest path algorithms for transportation applications: state of the art, Computers & Operations Research, 33.11: 3324-3343, 2006.
11) Georg, E, Frohlich, A, Gansterer, Margaretha, Doerner, Karl F, Safe and secure vehicle routing: a survey on minimization of risk exposure, International Federation of Operational Research Societies, 2023.
12) Ghoseiri, K, Nadjari, B, an ant colony optimization algorithm for the bi-objective shortest path problem, Applied Soft Computing. Vol.10, No.4, 1237-1246, 2009.
13) Hart, P, Nilsson, N, Raphael, B, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Trans. Syst. Sci. Cybern. 1968, 4, 100107, 2009.
14) Holland, John, Adaptation, H, in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
15) Kairanbay, Magzhan, Hajar, Mat Jani, A review and evaluations of shortest path algorithms, International Journal of Scientific and Technology Research, 2, 99-104, 2013.
16) Kuri, J, Puech, N, Gagnaire, M, Dotaro, E, routing foreseeable light path demands using a tabu search meta-heuristic, Proceedings of the IEEE Global Telecommunication Conference, pp.28032807, 2003.

Received: 16-05-2024 *Revised: 12-06-2024* *Accepted: 07-07-2024*

17) Lewis, R, A Shortest Path Algorithm for Graphs Featuring Transfer Costs at their Vertices, Cardi_ University, CF24,2020.
18) Merikh Bayat Farshad, meta-heuristic optimization algorithms (with applications in electrical engineering), Jihad Academic Press, 2013.
19) Mitsuo, G, Runwei, C, Dingwei, W, Genetic Algorithms for Solving Shortest Path Problems, Evolutionary Computation, IEEE International Conference on. 401-406,1997.
20) Mohammadpour, Hamidreza, Madanlu Joybari, Alireza, Review of meta-heuristic algorithms and their capabilities, the first national conference on meta-heuristic algorithms and their applications in science and engineering,iran,tehtan, 2013.
21) Mohemmed, A, Chandra, Sahoo, N, Kim Geok, T, solving shortest path problem using particle swarm optimization, Applied Soft Computing, Vol.8, No.4, pp. 1643-1653, 2008.
22) Muthusundari, s, Jothilakshmi, r, Divya, s, Kavitha, a, Umamaheswari, Computation Performance Optimization Technique of Shortest Path Routing Algorithm in Networks Using Out-degree, geintec, ISSN: 2237-0722. 11. 2, 2021.
23) Pohl, Ira, Bi-directional Search, Machine Intelligence, vol. 6, Edinburgh University Press, pp. 127–140, 1971.
24) Rigo, Michel , Advanced Graph Theory and Combinatorics,2016.
25) Robert, w. Floyd, algorithm 97: shortest path, communication of the ACM5(6):345, june 1962.
26) Samaher, Adnan, enas, Wahab, Abood, Wafaa, Abdulmuhsin, the multi-point delivery problem: Shortest Path Algorithm for Real Roads Network using Dijkstra, Journal of Physics: Conference Series 1530 (2020) 012040,2020.
27) Sunita, Deepak Garg, Dynamizing Dijkstra: A solution to dynamic shortest path problem through retroactive priority queu Journal of King Saud University – Computer and Information Sciences Article in pressmmm, 2018.
28) Vivaldini, K. T, Galdames, J. P. M, Pasqual, T. B, Sobral R. M, Araújo, R.C, Becker, M, Caurin, G. A. P, Automatic Routing System for Intelligent Warehouses, ICRA10 International workshop on Robotics and Intelligent Transportation System,2023.
29) Wang Shu-Xi, The Improved Dijkstra's Shortest Path, Algorithm and Its Application and Electronics Engineering (IWIEE) Procedia Engineering 29 (2012) 1186 – 119,0, 2012.
30) Xu, M Y, Liu, Q, Huang Y, Zhang G, an improved Dijkstra shortest path algorithm for sparse network Applied Mathematics and Computation 185 pp.247- 254, 2007.
31) Yu-Li, Chouy, Edwin, H, Robert, Romeijnz, Smithx, L, Approximating Shortest Paths in Large-scale Networks with an Application to Intelligent, Transportation Systems, September 27, 1998.
32) Zhang, H, Zhang, Z, AOA-Based Three-Dimensional Positioning and Tracking Using the Factor Graph Technique, Symmetry,2020.
33) Zyczkowski, Marcin, Szlapczynski, Rafal, Collision risk-informed weather routing for sailboats, Institute of Ocean Engineering and Ship Technology, Gdansk University of Technology, Poland,2023.