Project Main Area : Technological Design

Project Thematic Area : Algorithm Design and Use

Project Name (Title) : Solving public transport routing problems with a metaheuristic algorithm inspired by the behavior of sluggish fungi

Summary

The many-headed slime mushroom (*Physarum polycephalum*) is a creature that has inspired computational algorithms with its behaviors of solving mazes and optimizing food without a central command base that can be described as a "brain". Studies that mathematically model the network fan and food transport strategy and use the behavioral algorithm to solve theoretical and practical problems in engineering and robotics are widespread. However, these algorithms are not used for problems that require searching in a discrete solution space, such as the urban transit routing problem. In our study, we developed an algorithm that solves routing problems with a method inspired by the widely used Chunky Mushroom Algorithm. This program was tested both on the Mandl public transportation system, which is used in the literature to compare different methods, and on a system we created from bus lines in Istanbul. It is planned that this algorithm, which finds the optimal routes for a given map and the number of passengers who want to go between any two stops, will be developed to prevent public transportation problems in Istanbul and contribute to actions such as rerouting in case of accidents and natural disasters.

Keywords: mathematical optimization, routing problems, metaheuristic algorithms, mushroom algorithm

Objective

This research focuses on the most appropriate layout "from the plasmodium's point of view" when planning the most frequently used bus lines in Istanbul, the most densely populated city in Turkey with the most active public transportation use.

Bio-inspired algorithms have reached, and in some cases even surpassed, the problem-solving capacity of traditional computational methods thanks to their ability to produce optimal results under unfavorable conditions, reduce energy consumption and lower costs. Shortest path problems dealing with the optimization of transportation networks and public transportation systems are among the most discussed topics in computational intelligence [1]. In this context, many metaheuristic approaches such as Ant Colony Optimization, Artificial Bee Colony Algorithm, Smart Water Drops Algorithm, Fish Swarm Algorithm have been used to solve classical NP-hard problems such as Vehicle Routing Problem, Traveling Salesman Problem and Public Transport Routing Problem [2]. Therefore, in this project, an algorithm inspired by the plasmodium form of the acellular slime fungus *Physarum polycephalum* (Slime Mold Algorithm, SMA), which is one of the prominent models in this context, was studied [3].

The pathfinding mechanisms of *P.* polycephalum have been experimentally demonstrated to create networks with efficiency, fault tolerance and cost comparable to daily life infrastructure networks such as the Tokyo Subway system [4]. In addition, SMA has been widely and effectively used in engineering designs, motion modules of intelligent robots, computer-based photo and sonar analysis, classification of genetic data, signal-to-noise separation, workshop scheduling problems and other optimization studies [5]. However, **there is no study in the literature on how effective** SMA, a next generation metaheuristic algorithm, is in solving these public transportation problems. Therefore, our aim is to test how adequate an SMA-derived algorithm is in route optimization, both in "standard" public transport models found in the literature and in a model we have created for Istanbul's bus lines, and to compare obtained in route generation, energy minimization and optimal route selection with other algorithms.

Introduction

In the 1990s, computer scientists began to take inspiration from biological systems when constructing new optimization algorithms. For the next 10 years, optimization was based on *the trace-leaving algorithm*, which was inspired by the pheromone tracking behavior of ants [2, 6]. Ant colony-based optimization methods have been found to be functional in stationary environments, but most optimization problems are dynamic and require algorithms that can quickly adapt to conditions and generate solutions [6]. Accordingly, it was predicted that swarm intelligence algorithms such as ant colonies would not be able to adapt quickly to their environment. Although this prediction was later proven to be incorrect, different bio-inspired algorithms such as the slime fungus *Physarum polycephalum* have become prominent due to their adaptability to changes in the environment and fault tolerance [7, 8].

P. polycephalum (multi-headed slime fungus) is a species of the order Physarales, subclass Myxogastromycetidae and class Myxomycetes. It is commonly known as acellular or multi-headed slime fungus [4]. The studied behaviors related to adaptive networks are exhibited in the active vegetative stage of the complex life cycle of the organism, called 'plasmodium'. The plasmodium is a visible single cell with numerous diploid nuclei. It appears as a shapeless, yellowish mass with a network of protoplasmic tubes. At this stage, the organism forms a flat body that in nature can span an area of 30×30 cm^2. Moving like an amoeba, it feeds on microscopic organisms such as bacteria, spores and microparticles [9]. It can also feed on fungi as larger food items and plant matter under favorable conditions.

In search of nutrients, the plasmodium spreads towards food sources, surrounds them and aggregates, often covering the food source, secreting enzymes and digesting the food. When multiple food sources are present in the vicinity of the plasmodium, the organism aggregates masses of protoplasm together and develops a network of protoplasmic tubes to access nutrients (Figure 1). Through these tubes, protoplasm is released in regular cycles of 1.5 to 3 minutes and transported bidirectionally in a process similar to cytoplasmic streaming in plants [10]. When the plasmodium is deprived of water or nutrients, it goes into hibernation when it is unable to move to areas with a dense food source, forming a hardened mass called a sclerotium [4]**.** When a new food source is detected, it exits this mass and returns to its plasmodium form.

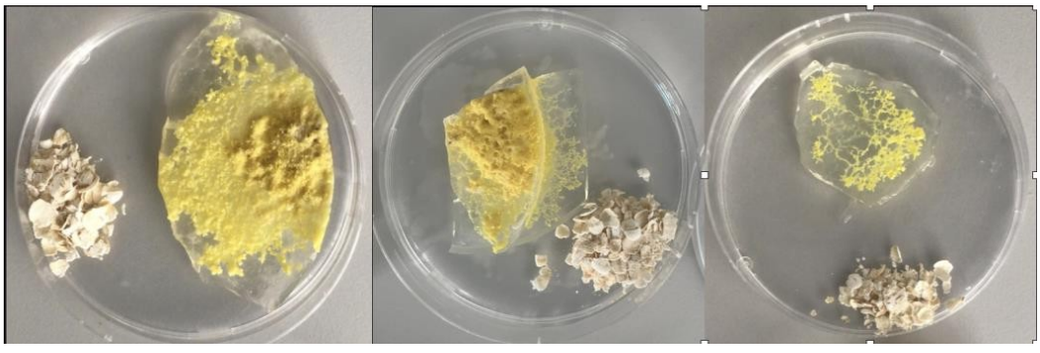

Figure 1. General morphology of *P. polycephalum* and foraging behavior by forming a tube network.

The strategy used by slime fungi in the process of food acquisition is based on two basic behaviors: expansion and contraction. Expansion has been demonstrated by models based on the shape and orientation of *P.* polycephalum, while contraction/retraction has been characterized by positive feedback dynamics [2, 6, 11, 12]. Since the total body mass of the plasmodium is limited, its expansion into a new area requires it to draw its mass from other regions. Therefore, although locally separated, the expansion and retraction movements in the organism occur in parallel. In other words, the plasmodium minimizes the distance between the mass and food sources by changing the distribution of its mass according to the distribution of nutrients.

In the 2000 paper "Maze-solving by an amoeboid organism", Nakagaki and colleagues observed that when food sources and *Physarum* plasmodium are placed in a maze, the plasmodium forms a protoplasmic tube from the center of the plasmodium to the food source, taking the shortest path and reaching the food source via an optimized path. Thus, they proved that the geometric functioning of the protoplasmic networks performs nutrient detection and transfer automatically, sensing its surroundings comprehensively and performing operations independent of other parts of the cell, without a centralized decision about the path it will follow in its subsequent progress [13].

This study and the news that plasmodium can form structures similar to the Tokyo Metro system have paved the way for Physarum-based bioinspiration studies. The behavior of plasmodium has been mathematically characterized by approaches such as cellular automata [14-17], agent-based modeling [18-20] and differential equations [7, 21, 22]. However, the most comprehensive modeling attempt was made by Mirjalili and colleagues in 2020 [23]. In this paper, which received more than 2500 citations in a short period of time, the behavior of plasmodium was mathematically modeled under three groups: "approach to food", "encirclement" and "wave motion". The approach algorithm describes the expansion and contraction actions exhibited to find food, while the encirclement and wave motion modules reflect the gathering behavior of the tube networks around the food.

Even if the designed SMA (Slime Mold Algorithm) model is generalized and "nutrients" is defined as "the variable to be optimized", the system moves the point where "the most nutrients" are available, in other words, where the most optimal solutions are found in the search space. For example, when the definition of "nutrients" is changed to "the cheapest design that meets certain conditions", the algorithm can solve engineering problems. In addition, the model has been compared with both bio-inspired algorithms such as Whale Optimization Algorithm, Firefly Algorithm, Grizzly Wolf Optimization, Moth Fire Optimization and new generation metaheuristic algorithms such as CLPSO, CBLOBA and CBA, and it has been observed that it provides more optimal results than these approaches [23]. For this reason, the use of SMA in the optimization of transportation routes in Istanbul is considered as the subject of this study.

Routing problems, which are frequently encountered in the context of transportation network analysis, are difficult to find exact solutions because they are NP-hard and have a large number of factors and variables. Therefore, especially in real-life routing problems, approximate solutions determined by metaheuristics are preferred. For example, Yoon and Chow compared three algorithms on 55 stops and 123 intermediate routes between five regions of New York, and Schmaranzer et al. designed routes based on passenger density during the day on six lines in Vienna Metro system [24, 25]. Szeto and Jiang optimized a 28-stop bus line in Hong and obtained a schedule that reduces the waiting time at a stop and consumes 5.5% less fuel than the schedule in operation. Mandl's transportation covering 21 routes between 15 cities in Switzerland is used as the gold standard in public transportation problems [26]. Although algorithms such as Artificial Bee Colony, Ant Colony Optimization, Cuckoo Search, Genetic Algorithm have been used on this network [27], the performance of SMA has not been tested. Therefore, in this study, Mandl's transportation network is optimized using an SMA-derived approach and the results are compared with the literature.

Finally, due to the importance of public transportation in Istanbul, an SMA-based optimization study was conducted on a local transportation network. Using the open source database OpenStreetMap, a map showing all bus stops and movements in Istanbul was created, and a simplified model was developed for a sample transportation network in Üsküdar consisting of 15 stops and 21 routes. An SMA derivative was used to design the timetables on the created transportation network and time-saving schedules were created, paving the way for future optimization studies on a larger scale.

Method

Establishing initial values. Metaheuristic algorithms require a set of parameters describing public transport lines and movements to be used as a starting point for optimization. In order to ensure that the selected initial lines conform to existing public transport design principles and to avoid bias in the design, the lines were determined with the help of a software created with MATLAB programming language (Scheme 1). The software was developed according to the line design principles defined by Mumford [28]. In this approach, the second and subsequent lines are routed to the stops through which the line has already passed, aiming to combine the lines into a network. In addition, the line that passes through a stop does not return to the same stop, and the line terminates when it is necessary to return. Finally, the total number of lines and the maximum number of stops a line can pass through are restricted. Although the example code was created for the Mandl transportation network with 15 stops, the function can be applied to other systems by changing the size of the *routearray* array.

```
% MATLAB code https://users.cs.cf.ac.uk/C.L.Mumford/papers/CEC2013.pdf
It is designed based on the progression scheme (pseudo code) defined at %%.
routearray= [1:1:15];
% Maximum number of lines and total stops to be calculated.
routeno = 4;
% Maximum number of stops a single line can pass through.
routemaxlength= 10;
% An empty directory containing stop data for lines.
fullroutes= zeros(routeno,routemaxlength);
% Previously used stops to provide transit between lines
Additional parameter that provides incentive for % reuse.
chosenroutes= zeros(0:0);
% Dijkstra's algorithm was created to calculate the minimum distance with
% connection map.
connectome= graph(mandl_connectivity_binary);
% Lines consisting of interconnected but non-repeating stops
% creation.
for i=1:routeno
% Random at first stop selectionconnection to other lines at subsequent stops
Selection of stops to form %.
    if i == 1
        fullroutes(1,1)= randsample(routearray,1); arraylength
    = 1;
    else
    fullroutes(i,1)= randsample(chosenroutes,1); arraylength =
    1;
    end
% Stops with a connection are randomized under the condition of a "repeat ban"
Creation of movement lines by combining %.
    while arraylength < routemaxlength &
setdiff(neighbors(connectome,fullroutes(i,arraylength)),fullroutes(i,:))~= 0;
        pswitches= neighbors(connectome,fullroutes(i,arraylength)); pswitches
        = setdiff(pswitches,fullroutes(i,:));
        if isempty(pswitches) == 0; arraylength=
            arraylength+ 1;
            extension= pswitches(randsample(length(pswitches),1),1);; fullroutes(i,arraylength) =
            extension;
            chosenroutes(end+1)= extension; else
% Termination of the line if it cannot be extended without repeating the same
            stop. break
        end
    end
end
```

Diagram 1. Random routing function used for initial values.

Development of a fitness . Metaheuristic algorithms basically aim to minimize a function. Therefore, the throughput of the transportation network to be analyzed must first be expressed mathematically and reduced to a single function. For example, the function used in the Szeto and Jiang paper is as follows [26]:

$$z = B \sum_{i \epsilon U} \sum_{e \epsilon V} d_{ie} NR_{ie} + B_2 \sum_{i \epsilon U} \sum_{e \epsilon V} d_{ie} T_{ie}$$

Where z is the fitness function, $_{B1}$ and $_{B2}$ are the importance ratios of transfer and total travel time, i and e are the stops on the map, $_{NRie}$ is whether there is a direct route between i and e, $_{die}$ i and e is the proportion of passengers who want to go from one stop to another, and $_{Tie}$ is the travel time from i to e. Moreover, due to the characteristics of the bus route in this paper (e.g. all bus routes start from the terminals), the function can be constrained by certain conditions. For example, if the number of buses in the fleet is W and the maximum number of trips is $_{Rmax}$;

$$\sum_{n=1}^{nmax} 2 f_n t_n \, (1 - X_{00n}) \leq W$$

The constraint ensures that trips do not require more buses than are available. Here $_{X00n}$ is another constraint that resets the non-existing routes between two stations to zero, and f is the frequency of the trip. The algorithm will minimize the fitness function within the constraints, thus returning a set of data on the stops that each bus will pass through.

In the study, an approach that prioritizes the shortest time spent in public transportation was used [29]. The function created is as follows:

$$\min z = \sum_{i=1}^{rmax} \sum_{j=1}^{rmax} d_{ij} t_{ij}$$

Where z is the fitness function, i and j are the stops, $_{dij}$ is the proportion of passengers who want to go from one stop to another, and $_{tij}$ is the distance between two stops. The code expressing the function to be minimized in MATLAB is shown in Diagram 2. In addition, the matrices used in the Mandl transportation network, which show the stops with direct transit between them, the distances between the stops and the movement requests, are presented in Appendix 1. As an additional constraint, only line series with non-infinite values of *valuesum* in the fitness function, i.e. lines that can provide transportation from every stop to every stop with the existing lines, are used in the study.

The code for adapting the SMA protocol designed for continuous variables to routing problems using discrete variables is summarized in Scheme 3. This approach is similar to Kechagiopoulos and Beligiannis' work on solving routing problems using Particle Swarm Optimization (PSA) [29]. Since it is not possible to vary the values of stops and routes freely, the clustering behavior of sluggish mushroom masses at their optimal points is modeled as "route exchange" between the masses. In particular, the masses that find the most favorable environment (i.e. with the lowest *valuesum*) "guide" the less influential masses, i.e. they transfer their routes to them. In order to model the expansion-contraction behavior, the transmission, which is fast at the beginning of the simulation, gradually slows down, i.e. the masses that find the "food" stop moving. Hence, our approach can be characterized as a hybrid path that exhibits both SMA and PSA features.

Finally, a simplified transportation system using the Mandl line, which is used as a model in the study, and a simplified transportation system in Istanbul by displaying bus stops and movements with OpenStreetView is shown in Figure 2. The parameters used for the Istanbul system are presented in Appendix-2.

```
% To measure the shortest distance between established communication networks
A function written at %.
function [o]= mintransport(fullroutes);
% Total number of stops and to be minimized.
totalstops= 15;
valuesum= 0;
City distances and line demand frequency from % Mandl model.
load('mandl_distances.mat');
load('mandl_standard.mat');
% Mask used to remove lines that are not on the model.
maskmaker= zeros(totalstops,totalstops);
% Number and maximum length of lines between stops.
pathno= 4;
pathlength= 8;
% Removal of lines that are not in the system from the Mandl diagram.
for i= 1:pathno;
    for j= 2:pathlength; try
        maskmaker(fullroutes(i,j),fullroutes(i,j-1))= 1;
        maskmaker(fullroutes(i,j-1),fullroutes(i,j))= 1; end
    end
end
masked_transfers= maskmaker.*mandl_distances;
Calculate time between two stops after %Mask and stops
Algorithm that applies weights according to the movement demand between %.
masked_routepath= digraph(masked_transfers); for i
= 1:totalstops;
    for j= 1:totalstops
        [shortestid,shortestdistance]= shortestpath(masked_routepath,i,j); valuesum =
        valuesum + shortestdistance.*mandl_standard(i,j);
    end
end
o=valuesum;
end
```

Diagram 2. The fitness function designed for the study, to be minimized by SMA. The example code is written for the Mandl model, but it can be applied to other lines by changing the .mat files with the path parameters between stops and the number of stops determined by *totalstops*.

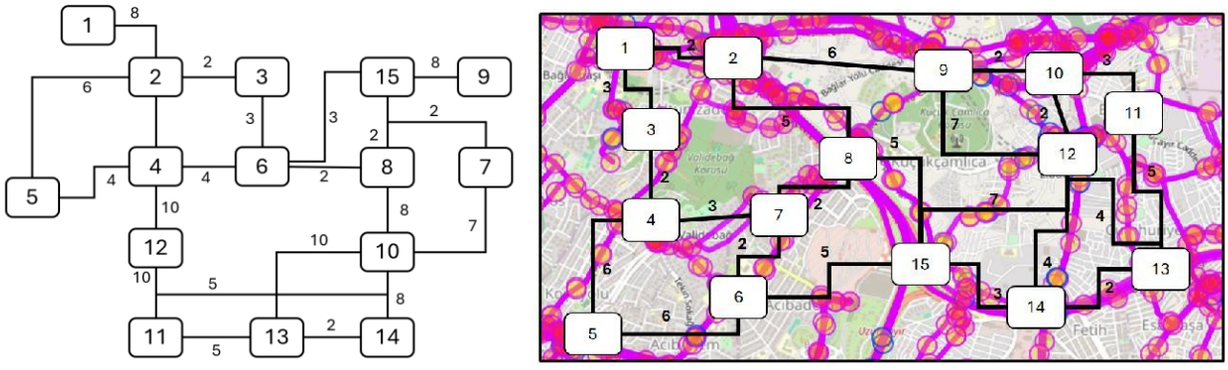

Figure 2. Graphical representation of the model transportation lines used in the study. (a) Mandl line depicted in the literature, including the matrix of stops and the matrix of roads between them (b) OpenStreetMap based transportation density map used for the an Istanbul based transportation network within the scope of the project and the local line created. Both maps were created for comparability, covering a total of 21 roads between 15 stops. The differences are the connections between stops and the length of the roads. The numbers on the road are distance values between nodes, reflecting the distance or time between two stops.

```
load('mandl_distances.mat');
load('mandl_standard.mat');
load('mandl_connectivity_binary.mat');
% Determination of variables to be used in optimization.
pathno= 4;
pathlength= 8;
totalstops= 15;
number_of_slimes= 5;
repeats= 5;
% Dispersal of sluggish fungal masses into the environment.
array_of_slimes= zeros(pathno,pathlength,number_of_slimes);
% Transfer the best scoring audiences to a separate series.
best_in_class= zeros(number_of_slimes,repeats);
% Defining the lines that audiences will follow.
for i = 1:1:number_of_slimes
    initial_paths= fullpaths();
    array_of_slimes(:,:,i)= initial_paths; best_in_class(i,1)=
    mintransport(array_of_slimes(:,:,i));
end
% The more favorable conditions for the growth of the masses of sluggish fungus
Soggy mushrooms that trigger % movement and undergo the best optimization process
Detection of % masses.
array_of_perfection= array_of_slimes; for
j = 1:1:(repeats - 1)
    iteration_parameter= j/repeats; for
    k = 1:number_of_slimes
        if rand()< (1-iteration_parameter/2)
           [routepick_k,minim]= min(best_in_class(k:number_of_slimes)); for l
               = 1:number_of_slimes
                   for m= 1:10
                   duplicate= array_of_slimes;
                   duplicate(randi(4),:,k)= array_of_perfection(randi(4),:,minim); fullroutes
                   = duplicate(:,:,k);
                   if length(unique(duplicate(:,:,k)))== totalstops & mintransport(fullroutes) <
inf
                        array_of_slimes= duplicate; break
                   else
                   end
                 end
               end
        else
                for m= 1:10
                    duplicate= array_of_slimes; duplicate(randi(4),:,k) =
array_of_perfection(randi(4),:,randi(number_of_slimes));
                    fullroutes= duplicate(:,::,k);
                            if length(unique(duplicate(:,:,k)))== totalstops &
mintransport(fullroutes) < mean(nonzeros(best_in_class(l,:)));
                        array_of_slimes= duplicate; break
                            else
                        end
                end
        end
    best_in_class(k,j+1)= mintransport(array_of_slimes(:,:,k)); end
end
```

Scheme 3. Design of a simplified algorithm for pilot studies, which transfers the motion of sluggish mushrooms to a search space with discrete values.

Project Work-Time Schedule

Activities carried out	Time Interval
Literature on the behavior of slime fungus scanning, observation of cork motion phenomena on 3D printed surfaces	March 2024 - June 2024
We provide theoretical and practical information about the slime mushroom algorithm and other bio-inspired metaheuristic algorithms. mathematical studies	May 2024 - September 2024
Designing and debugging MATLAB code	July 2024 - September 2024
Data analysis shows that the designed algorithm to be used to improve transportation	September 2024 - December 2024
Writing the project report, transforming the obtained data into a scientific article and publishing it preliminary study	December 2024 - January 2025

Findings

Table 1 shows examples of road lines generated by the initial value identification algorithm (Scheme 1) on the Mandl and Istanbul systems. As can be seen in the table, in both test platforms, the lines do not return to the same stop, go outside the specified length limits or try to move to points with no road between them. However, some lines stay at the same stop for a long time (i.e. do not move from the stop). When the obtained routes are sorted by the minimization function (*mintransport*), it is observed that the most efficient (i.e. the ones with the lowest *valuesum*) routes pass through as many stops as possible, especially nodes 6 on the Mandl line and node 12 on the Istanbul line. Moreover, due to the "line merging" principle described in Diagram 1, each line starts with a stop that has been used before. The main result of the study, the optimal routes obtained as a result of 20 mushroom masses and 20 movement functions determined using the *Minimizer* algorithm, are shown on the maps in Figure 3.

Line 1	3	2	5	4	6	8	15	9
Line 2	15	8	6	3	2	4	12	11
Line 3	11	10	8	6	4	2	1	
Line 4	12	11	13	14	10	8	15	7
(Mandl, 4 lines, max 8 stops per line)								

Line 1	14	12	10	11	13	
Line 2	13	14	15	6	5	4
Line 3	5	4	7	8	2	9
Line 4	7	4	3	1	2	8
Line 5	11	13	12	9	10	
(Istanbul5 linesmaximum 6 stops per line)						

Table 1. Example routes obtained with the *fullpaths* function. The movements on the lines can be traced on the models in Figure 2. The lines do not need to have a maximum number of stops, for example line 3 in the Mandl model has 7 stops.

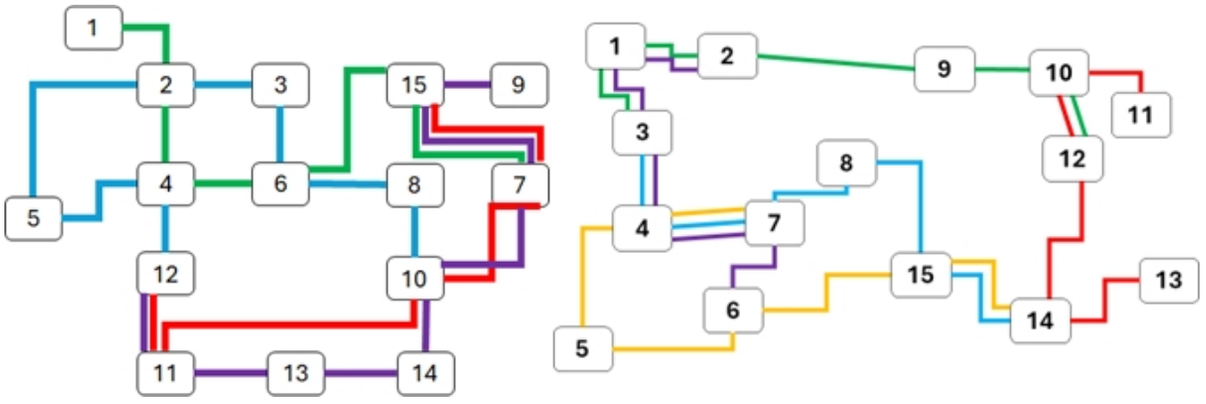

Figure 3. Optimal public transportation routes in (a) Mandl and (b) Istanbul determined by an SMA-derived approach. Each color indicates a different route. For the Mandl route, we also used a demand matrix (Annex 1), which shows which stops passengers want to move between; in the Istanbul map, the distribution of passengers is assumed equal and their demand for each stop is the same.

Conclusion and Discussion

1. The three algorithms obtained from the study were found to be capable of efficiently a) generating routes, b) ranking routes according to their ability to meet the needs of travelers, and c) determining the optimal routes in an iterative manner.

2. The optimal route chosen for the Mandl model consistent with the routes in other papers working with this system. For example, the fact that stop 10, which has the highest passenger demand, and stop 15, which offers access to many other stops, act as hubs for the lines is also seen in the work of Chakroborty and Wivedi (in this paper the first node is called 0, in our work it is called 1, hence the 9th and 14th nodes in the paper) [30].

3. However, the behavior of sluggish mushrooms in this study is reflected in the algorithm as only one parameter (the decrease in system motion over time). Therefore, the algorithm can be made more effective by using other elements of SMA. In this regard, our project is a pilot study showing that the SMA algorithm designed for continuous variables can be optimized for discrete variables and used in routing problems.

4. Considering the hundreds of bus lines in Istanbul, the 15-stop model considered in this study is too small. In addition, our study only considers distance, but other parameters such as fare and travel time have been optimized in the literature, and not only routes but also timetables have been created according to the number of fleets and passenger density during the day. However, in principle, there is no problem in transferring SMA to larger systems or using it to optimize multiple parameters.

4. One assumption of our study is that the paths are symmetric, i.e. there is no difference between going from A to B and versa. This is not true especially in Istanbul so asymmetric optimization can also be done by changing the stop distance matrix.

5. Therefore, our future goal is to design an advanced algorithm that covers all of the main stops in Istanbultaking into account road conditions and passenger demands, time spent on the road and faresto publish the results of the study as an article in a scientific journal.

Recommendations

to prepare the ground for future studies on the subject, the codes we have written will be published on the GitHub website. The MATLAB we are currently working on, coding the distance and passenger information on Istanbul lines collected from public databases, will be made available to national and researchers in a similar way.

Sources

1. Aridhi, S., Lacomme, P., Ren, L., & Vincent, B. (2015). A MapReduce-based approach for shortest path problem in large-scale networks. *Engineering Applications of Artificial Intelligence*, *41*. https://doi.org/10.1016/j.engappai.2015.02.008
2. Reid, C. R., Sumpter, D. J. T., & Beekman, M. (2011). Optimisation in a natural system: Argentine ants solve the Towers of Hanoi. *Journal of Experimental Biology*, *214*(1). https://doi.org/10.1242/jeb.048173
3. Tsompanas, M. A. I., Sirakoulis, G. C., & Adamatzky, A. I. (2015). Evolving transport networks with cellular automata models inspired by slime mould. *IEEE Transactions on Cybernetics*, *45*(9). https://doi.org/10.1109/TCYB.2014.2361731
4. Adamatzky, A. (2010). Physarum machines: Computers from slime mould. In *Physarum Machines: Computers from Slime Mould.* https://doi.org/10.1142/7968
5. Wei, Y., Othman, Z., Daud, K. M., Luo, Q., & Zhou, Y. (2024). Advances in slime molding algorithm: A comprehensive survey. Biomimetics, 9(1), Article 31. https://doi.org/10.3390/biomimetics9010031 MDPI
6. Latty, T., & Beekman, M. (2013). Keeping track of changes: The performance of ant colonies in dynamic environments. *Animal Behavior*, *85*(3). https://doi.org/10.1016/j.anbehav.2012.12.027
7. Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, *33*. https://doi.org/10.1016/j.swevo.2016.12.005
8. Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebber, D. P., Fricker, M. D., Yumiki, K., Kobayashi, R., & Nakagaki, T. (2010). Rules for biologically inspired adaptive network design. *Science*, *327*(5964). https://doi.org/10.1126/science.1177894
9. Binitha, S., & Sathya, S. S. (2012). A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSCE)*, *2*(2).
10. Adamatzky, A. (2010). Physarum machines: Computers from slime mould. In *Physarum Machines: Computers from Slime Mould.* https://doi.org/10.1142/7968
11. Nakagaki, T. (2001). Smart behavior of true slime mold in a labyrinth. In *Research in Microbiology* (Vol. 152, Issue 9). https://doi.org/10.1016/S0923-2508(01)01259-1
12. Haskins, E. F., Aldrich, H. C., & Daniel, J. W. (1984). Cell Biology of Physarum and Didymium. *Mycologia*, *76*(2). https://doi.org/10.2307/3793122
13. Nakagaki, T., Yamada, H., & Tóth,. (2000). Maze-solving by an amoeboid organism. Nature, 407(6803), 470-470. https://doi.org/10.1038/35035159
14. Bonabeau, E., Dorigo, M., & Theraulaz, G. (2000). Inspiration for optimization from social insect behaviour. In *Nature* (Vol. 406, Issue 6791). https://doi.org/10.1038/35017500
15. Gao, C., Liu, C., Schenz, D., Li, X., Zhang, Z., Jusup, M., Wang, Z., Beekman, M., & Nakagaki, T. (2019). Does being multi-headed make you better at solving problems? A survey of Physarum-based models and computations. In *Physics of Life Reviews* (Vol. 29). https://doi.org/10.1016/j.plrev.2018.05.002
16. Liu, J. (2008). Autonomy-Oriented Computing (AOC): The nature and implications of a paradigm for self-organized computing. *Proceedings - 4th International Conference on Natural Computation, ICNC 2008*, *1*. https://doi.org/10.1109/ICNC.2008.872
17. Gunji, Y. P., Shirakawa, T., Niizato, T., Yamachiyo, M., & Tani, I. (2011). An adaptive and robust biological network based on the vacant-particle transportation model. *Journal of Theoretical Biology*, *272*(1). https://doi.org/10.1016/j.jtbi.2010.12.013

18. Liu, Y., Zhang, Z., Gao, C., Wu, Y., & Qian, T. (2013). A physarum network evolution model based on IBTM. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7929 LNCS*(PART 2). https://doi.org/10.1007/978-3-642-38715-9_3
19. Tsompanas, M. A. I., & Sirakoulis, G. C. (2012). Modeling and hardware implementation of an amoeba-like cellular automaton. *Bioinspiration and Biomimetics*, *7*(3). https://doi.org/10.1088/1748-3182/7/3/036013
20. Wu, Y., Zhang, Z., Deng, Y., Zhou, H., & Qian, T. (2015). A new model to imitate the foraging behavior of Physarum polycephalum on a nutrient-poor substrate. *Neurocomputing*, *148*. https://doi.org/10.1016/j.neucom.2012.10.044
21. Jones, J. (2010). The emergence and dynamical evolution of complex transport networks from simple low-level behaviours. *International Journal of Unconventional Computing*, *6*(2).
22. Wu, Y., Zhang, Z., Deng, Y., Zhou, H., & Qian, T. (2012). An enhanced multi-agent system with evolution mechanism to approximate Physarum transport networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7691 LNAI*. https://doi.org/10.1007/978-3-642-35101-3_3
23. Heidari, A. A., Gandomi, A. H., Faris, H., Alavi, A. H., & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems, 111*, 300-323. https://doi.org/10.1016/j.future.2020.04.010
24. Tero, A., Kobayashi, R., & Nakagaki, T. (2006). Physarum solver: A biologically inspired method of road-network navigation. *Physica A: Statistical Mechanics and Its Applications*, *363*(1). https://doi.org/10.1016/j.physa.2006.01.053
25. Adamatzky, A. (2009). If BZ medium did spanning trees these would be the same trees as Physarum built. *Physics Letters, Section A: General, Atomic and Solid State Physics*, *373*(10). https://doi.org/10.1016/j.physleta.2008.12.070
26. Mandl, C. E. (1980). Evaluation and optimization of urban public transportation networks. European Journal of Operational Research, 5(6), 396-404. https://doi.org/10.1016/0377-2217(80)90056-0
27. Li, Q., & Guo, L. (2023). Nature-inspired metaheuristic optimization algorithms for urban transit routing problem. *Engineering Research Express, 5*(2), 023001. https://doi.org/10.1088/2631-8695/acb8a0
28. Mumford, C. L. (2013). New heuristic and evolutionary operators for the multi-objective urban transit routing problem. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2013)*, 1-8. https://doi.org/10.1109/CEC.2013.6557810
29. Kechagiopoulos, P. N., & Beligiannis, G. N. (2014). Solving the urban transit routing problem using a particle swarm optimization based algorithm. *Applied Soft Computing, 21*, 654-676. https://doi.org/10.1016/j.asoc.2014.03.029
30. Chakroborty, P., & Wivedi, T. (2002). Optimal route network design for transit systems using genetic algorithms. Engineering Optimization, 34(1), 83-100. https://doi.org/10.1080/03052150210909

Annex 1. Inter-stop connectivity, distance and passenger demand matrices used for the Mandl model.

Annex 1a. mandl_connectivity_binary.mat

0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	1	1	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	0	1	0	0	0	1	0	1	0

Annex 1b. mandl_distances.mat

0	8	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	2	3	6	0	0	0	0	0	0	0	0	0	0
0	2	0	0	0	3	0	0	0	0	0	0	0	0	0
0	3	0	0	4	4	0	0	0	0	0	10	0	0	0
0	6	0	4	0	0	0	0	0	0	0	0	0	0	0
0	0	3	4	0	0	0	2	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	7	0	0	0	0	2
0	0	0	0	0	2	0	0	0	8	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
0	0	0	0	0	0	7	8	0	0	5	0	10	8	0
0	0	0	0	0	0	0	0	0	5	0	10	5	0	0
0	0	0	10	0	0	0	0	0	0	10	0	0	0	0
0	0	0	0	0	0	0	0	0	10	5	0	0	2	0
0	0	0	0	0	0	0	0	0	8	0	0	2	0	0
0	0	0	0	0	3	2	2	8	0	0	0	0	0	0

Annex 1c. mandl_standard.mat

0	400	200	60	80	150	75	75	30	160	30	25	35	0	0
400	0	50	120	20	180	90	90	15	130	20	10	10	5	0
200	50	0	40	60	180	90	90	15	45	20	10	10	5	0
60	120	40	0	50	100	50	50	15	240	40	25	10	5	0
80	20	60	50	0	50	25	25	10	120	20	15	5	0	0
150	180	180	100	50	0	100	100	30	880	60	15	15	10	0
75	90	90	50	25	100	0	50	15	440	35	10	10	5	0
75	90	90	50	25	100	50	0	15	440	35	10	10	5	0
30	15	15	15	10	30	15	15	0	140	20	5	0	0	0
160	130	45	240	120	880	440	440	140	0	600	250	500	200	0
30	20	20	40	20	60	35	35	20	600	0	75	95	15	0
25	10	10	25	15	15	10	10	5	250	75	0	70	0	0
35	10	10	10	5	15	10	10	0	500	95	70	0	45	0
0	5	5	5	0	10	5	5	0	200	15	0	45	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ANNEX 2. and distance matrices used for the Istanbul model.

Annex 2a. istanbul_connectivity_binary.mat

0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	1	1	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	0	1	0	0	0	1	0	1	0

Annex 2b. istanbul_distances.mat

0	2	3	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	5	6	0	0	0	0	0	0
3	0	0	2	0	0	0	0	0	0	0	0	0	0	0
0	0	2	0	6	0	3	0	0	0	0	0	0	0	0
0	0	0	6	0	6	0	0	0	0	0	0	0	0	0
0	0	0	0	6	0	2	0	0	0	0	0	0	0	5
0	0	0	3	0	2	0	2	0	0	0	0	0	0	0
0	5	0	0	0	0	2	0	0	0	0	0	0	0	5
0	6	0	0	0	0	0	0	0	2	0	7	0	0	0
0	0	0	0	0	0	0	0	2	0	3	2	0	0	0
0	0	0	0	0	0	0	0	0	3	0	0	5	0	0
0	0	0	0	0	0	0	0	7	2	0	0	4	4	7
0	0	0	0	0	0	0	0	0	0	5	4	0	2	0
0	0	0	0	0	0	0	0	0	0	0	4	2	0	3
0	0	0	0	0	5	0	5	0	0	0	7	0	3	0