Optimising itinerary generation for autonomous tour guides by integrating real time data into genetic algorithms

Cristian SANDU, Luminița DUMITRIU, Ioan ȘUȘNEA

Faculty of Automation, Computers, Electrical and Electronic Engineering, Computer Science Department, "Dunărea de Jos" University of Galați, Romania

cristian.sandu@ugal.ro, luminita.dumitriu@ugal.ro, ioan.susnea@ugal.ro

Abstract: Research in the field of autonomous vehicles has made good progress in the last years, but mass adoption is unlikely to happen very soon due to, in most part, important ethical and security concerns. The authors consider that niche uses of autonomous vehicles are more likely to be researched and developed, since they can substantially make an immediate difference in specific applications. In this paper we analyse and propose a novel genetic algorithm-based dynamic itinerary generation mechanism for use in autonomous tour guides, with the goal of achieving the highest amount of itinerary point visits in the given timeframe. It does that by taking into account the available time and the tour points that are defined as most important by the tourist, through a ranking system and by integrating real time data, such as traffic conditions. By autonomous tour guide we refer to any vehicle capable of autonomously carrying a tourist or group of tourists. The algorithm proves to be reliable and fast enough to allow for dynamic reconfiguration of the itinerary.

Keywords: Genetic Algorithms, Autonomous Tour Guides, Autonomous Vehicles, Smart Tourism, Route Planning, Artificial Intelligence.

Optimizarea generării de itinerarii pentru ghizi turistici autonomi prin integrarea datelor în timp real în algoritmi genetici

Rezumat: Cercetările în domeniul vehiculelor autonome au făcut progrese semnificative în ultimii ani, însă adoptarea în masă este puțin probabil să se întâmple foarte curând, în mare parte din cauza preocupărilor importante legate de etică și siguranță. Autorii consideră că utilizările de nișă ale vehiculelor autonome sunt mai susceptibile să fie cercetate și dezvoltate, deoarece acestea pot face o diferență substanțială imediată în aplicații specifice. În această lucrare, analizăm și propunem un mecanism inovativ de generare dinamică a itinerariilor, bazat pe algoritmi genetici, destinat ghizilor turistici autonomi, având ca scop realizarea unui număr maxim de vizite ale punctelor de itinerariu în intervalul de timp dat. Acest lucru se realizează prin luarea în considerare a timpului disponibil și a punctelor de interes stabilite ca fiind cele mai importante de către turist, printr-un sistem de clasificare, și prin integrarea datelor în timp real, cum ar fi condițiile de trafic. Prin „ghid turistic autonom" ne referim la orice vehicul capabil să transporte autonom un turist sau un grup de turiști. Algoritmul se dovedește a fi fiabil și suficient de rapid pentru a permite reconfigurarea dinamică a itinerariului.

Cuvinte cheie: algoritmi genetici, ghizi turistici autonomi, vehicule autonome, turism inteligent, planificarea rutelor, inteligență artificială.

1. Introduction

Although relevant advancements have been made in the research of autonomous vehicles, mass adoption of such vehicles is still far from being achieved, due to security, ethical and other technological and legal concerns (Bezai et al., 2021).

The present indication is that autonomous vehicles are to be used less by individuals, and mostly in other applications, such as group travel, especially due to environmental concerns, or to improve daily repetitive travel, having examples of autonomous buses (Azad et al., 2019), or autonomous waterborne vehicles for museum visits (Rong et al., 2020).

There are some reasons why the adoption of alternative applications is likely to happen:

https://doi.org/10.33436/v34i4y202409

- *Easier to manage in controlled environments*. Autonomous buses, for example, would drive in dedicated bus lanes, which means that some typical issues could be eliminated from the start, by installing sensors and hardware on the designated routes to help with pedestrian detection, direction control and collision avoidance. The legal framework is easier to define since other vehicles are forbidden to enter the bus lane. Even if accidents were to happen, accountability would be easy to attribute;
- *Easier legislation changes*. A new legal framework is easier to generate for a specific application, compared to broader scenarios that would have to take into account multiple unexpected unknowns;
- *Cost efficiency*. Generating higher income and profit through automation is generally a good strategy for companies to implement. Business people are more likely to invest in novel technologies than regular users. In the proposed use case, a travel agency or tour guide company could benefit from promoting travel packages with a packed itinerary that could be guaranteed through the usage of smart autonomous tour guides.

The travel trends indicate a high growth of experience-centred travel (Hosany, Sthapit & Björk, 2022), being referred to as a "mega trend". This and smart tourism, which proposes tools that can be used to automate aspects of travel, have a high potential, a reason to consider the proposal of a dynamic itinerary generation mechanism.

Itinerary generation and proposal is not something currently automated and it is not only a problem for the pre planning phase, but also one concerning the actual trip. Among the problems travel agencies usually face is finding a partner tour guide, and it can become a tedious process that doesn't guarantee the success and satisfaction of a trip.

Smart route planning is a prominent research issue in tourism (Zhou et al., 2019) and it could be a viable idea to intertwine the two domains to reach a common solution for a real problem.

Route planning for an autonomous tour guide is not only about calculating the shortest path, but also about integrating user preferences, time constraints and real time environmental changes. Traditional brute force algorithms become impractical as the number of variables increases due to their exponential runtime. Although the task resembles the Travelling Salesman Problem (TSP) in that it involves finding an optimal path through multiple locations, traditional TSP algorithms aren't well suited to handle additional variables efficiently (such as the ones mentioned earlier), and that is why genetic algorithms could be chosen, due to their flexibility and adaptability. In a context where a travel itinerary can't be rigidly fixed and must be adapted to unforeseen changes, genetic algorithms provide the best available solutions within a reasonable timeframe.

Genetic algorithms are well researched in autonomous vehicle applications, for example such algorithms have been used for the planning of autonomous UAVs in target coverage problems (Pehlivanoglu, Y.V. & Pehlivanoglu, P., 2021), route planning in AVs of different kinds (Yu et al., 2012) and car navigation systems (Kumar, Arunadevi & Mohan, 2009) and (Yan & Lu, 2019).

Section 2, begins with a short review of some related work on route planning, making a comparative analysis of two different common approaches, and ends by highlighting the limitations of these traditional methods. In section 3 the details of the proposed algorithm are explained, section 4 introduces the concept of real time data integration, and section 5 provides the results of the experiments done. Section 6 comprises the conclusions and discussion about future work.

2. Comparative analysis

The simplest and easiest attempt to try and solve a route planning problem is to use an exact algorithm. To illustrate the inefficiency of the brute-force methods, however, we performed a regression analysis on the computation times relative to the number of tour points. Using data from our empirical tests, we fitted the growth model

$$T(n) = a \times e^{b \times n} \tag{1}$$

where *T(n)* is the computation time in seconds and *n* is the number of tour points. The regressions yielded the constants $a \approx 1.5 \times 10^{-6}$, $b \approx 0.69$ with a coefficient $R^2 = 0.69$, which is a high value indicating a good fit, confirming that the computation time increases exponentially with the number of tour points.

Although 25 tour points would take approximately 16 seconds, the time required for 30 tour points is estimated at 8.6 minutes, and more than 40 points would take a few hours. Such times are impractical for real time route planning.

In Figure 1 the growth in computation time can be visualized as the number of tour point increases, as it resulted from our experiment.

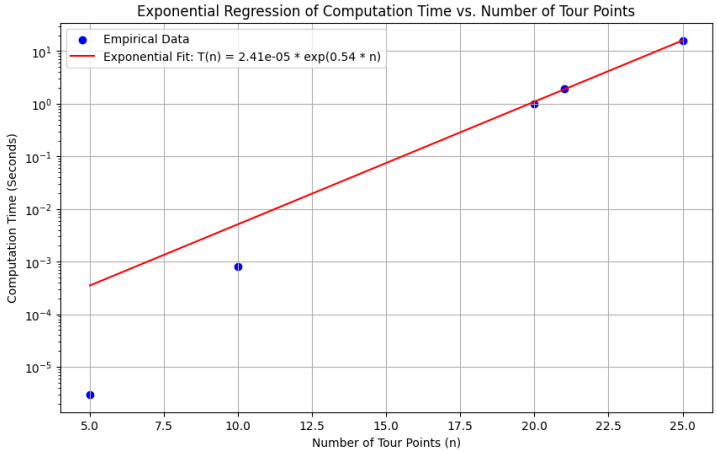


Figure 1. Exponential regression of computation time versus number of tour points

The next observation that can be made is that this problem closely resembles the TSP problem, where the shortest possible route must be achieved by visiting a set of locations only once. Some of the algorithms commonly used to solve the TSP are the nearest neighbour heuristic, branch and bound algorithm, or dynamic programming approaches, but they assume a static environment that does not fit with the real-time data integration.

There are various TSP solvers designed to find the optimal solution . It is still a matter of research. The most common method to solve TSP is the Concorde Solver developed by Applegate et al. (2006). It is widely used in academic research due to its efficiency and accuracy. Trying to find an optimal path using this method, by representing a distance matrix between 14 pairs of attractions was considered for a start. The solver requires a problem definition in the TSPLIB format, so the file describing the TSP instance as such was created:

NAME: tour_points
TYPE: TSP COMMENT: TSP instance for tour points
DIMENSION: 14
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: FULL_MATRIX
EDGE_WEIGHT_SECTION
0 5 10 15 4 6 9 8 12 20 7 5 14 16 5 0 7 12 6 4 8 5 10 18 9 3 12 14 10 7 0 5 9 11 4 6 8 15 12 9 7 10 15 12 5 0 14 16 9 11 6 10 17 14 12 8 4 6 9 14 0 3 7 6 13 21 5 4 15 17 6 4 11 16 3 0 10 7 14 22 6 2 16 18 9 8 4 9 7 10 0 3 9 17 11 8 8 11 8 5 6 11 6 7 3 0 7 15 10 5 9 12 12 10 8 6 13 14 9 7 0 8 14 11 10 4 20 18 15 10 21 22 17 15 8 0 22 19 12 5 7 9 12 17 5 6 11 10 14 22 0 7 16 19 5 3 9 14 4 2 8 5 11 19 7 0 13 15 14 12 7 12 15 16 8 9 10 12 16 13 0 9 16 14 10 8 17 18 11 12 4 5 19 15 9 0
EOF

The EDGE_WEIGHT_SECTION contains the distance matrix that is defined in Table 1:

Table 1. The distance matrix used for the TSP Concorde Solver

From/To	1	2	3	…	14
1	0	5	10	…	16
2	5	0	7	…	14
3	10	7	0	…	10
…					
14	16	14	10	…	0

By running the program an optimal tour length of 85 km was obtained, as well as the following tour order: 1 5 6 12 14 13 11 10 9 7 8 2 4 3 1. Adding up the duration spent on the travel times and the tour times, a total tour duration of approximately 14.7 hours resulted. That exceeds the time limit we set for our experiments in the following sections. Besides, this approach does not meet all constraints set, nor could it integrate real time data to improve the outcome, which is a key aspect to be solved.

Heuristics like the Ant Colony Optimization (ACO) have been effectively applied to the TSP (Li et al., 2016), but such algorithms require high computation resources and may not adapt quickly to changes, which makes it unfit.

Simulated annealing (SA) can handle large search spaces, but it typically requires careful tuning of parameters and may converge slowly for multi objective problems as found by Kirkpatrick, Gelatt & Vecchi, (2012).

In contrast, genetic algorithms have already been successfully applied to route planning, for example Yu & Lu, (2012) developed an improved GA for multi-modal route planning that effectively handles complex transportation networks and varying constraints.

Zhang, Tang & Zheng, (2019) proposed an adaptive GA that incorporates traffic factors into the dynamic TSP. They considered real-time traffic information and their algorithm adjusts routes dynamically to optimize travel times. This approach is similar to the present one.

In recent papers, such as the one by Tang, Pan & Wang, (2017), a hybrid GA for the shortest path problem is proposed, with their algorithm efficiently recalculating optimal paths in response to network changes.

The choice of a genetic algorithm approach can be justified by highlighting three important advantages. First of all, GAs can find high quality solutions faster than exhaustive methods, thus being more efficient for real time applications. Second, GAs are flexible because they can incorporate multiple objectives and constraints. Then, adaptability is another important advantage, because GAs can adjust to changes in the environment without restarting the entire computation process (Wang & Chen, 2015).

3. Methodology

To begin with, a population of possible solutions are generated from a list of predefined tour points, which are given a duration and a rating, that means a dynamic variable that can have a different value for each user of the system, and can be changed with each individual run.

Every set of all possible solutions is called a generation. The first generation is a random choice of possible solutions.

To calculate how good a solution is a fitness function considering the duration and the rating is used. It returns the cumulated time to visit for each attraction when it is lower than the limit, otherwise it doesn’t.

Then a solution with a higher fitness score is selected for moving on to the next generation, followed by the tournament selection, and the crossover function.

Although a single point crossover function was taken into consideration, it was found that using it, one of the most common types of crossover functions, was not satisfactory for the use case,

to ensure the uniqueness of the tour point as a tour point cannot be visited twice.

The final two steps were the mutation, which randomly swaps two tour points between generations, and the replace function at the end, that replaced the previous population.

The algorithm generated a result as soon as the maximum number of generations had been reached.

The novelty of this approach is represented by the introduction of various sources of real time data, to account for the changing traffic conditions and the changing weather conditions, to start with. Figure 2 we provided a graphical representation of the algorithm.

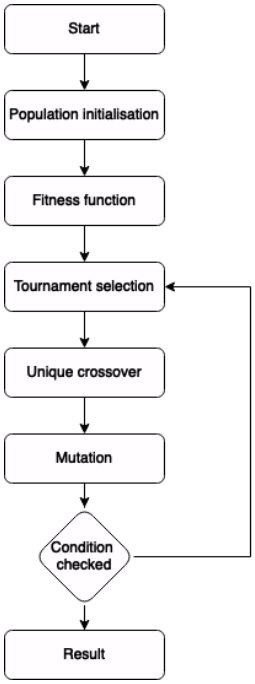


Figure 2. GA operations

3.1. Population initialisation

The first step is to define the "Initialize Population" function, which will be responsible for the generation of the initial population of potential solutions, each individual within the population representing a unique itinerary, randomly constructed by sampling a number of points from the total set of available tour points:

```
Function InitializePopulation(populationSize, tourPoints, individualSize)
  population ← empty list
  For i ← 1 to populationSize
            individual ← SampleWithoutReplacement(tourPoints.keys(), individualSize)
    Add individual to population
  End For
  Return population
End Function.
```

3.2. Fitness function

The fitness function will evaluate the suitability of a given individual itinerary based on the total rating of the selected tour points and the total duration of the tour.
It penalises itineraries that exceed a predefined limit, so that the tourists preferences are taken into consideration together with the time constraint:

```
Function Fitness(individual, tourPoints, durationLimit, penaltyRate)
  totalRating ← 0
  totalDuration ← 0
  For each pointId in individual
    point ← tourPoints[pointId]
    totalRating += point.rating
    totalDuration += point.duration
  End For
  If totalDuration > durationLimit Then
    penalty ← (totalDuration - durationLimit) * penaltyRate
    totalRating -= penalty
  End If
  Return totalRating
End Function.
```

3.3. Tournament selection

Tournament selection is a method to select individuals to participate in the creation of the next generation.The function randomly selects a small subset of the population and chooses the individual with the highest fitness score within the subset. The process is repeated until a sufficient number of individuals have been selected:

```
Function TournamentSelection(population, fitnesses, tournamentSize)
  selectedParents ← empty list
  While Length(selectedParents) < Length(population) / 2
    tournament ← SampleWithoutReplacement(Combine(population, fitnesses), tournamentSize)
    winner ← MaxBySecondElement(tournament)
    Add winner[0] to selectedParents
  End While
  Return selectedParents
End Function.
```

3.4. Unique crossover

The main difference between a single point crossover function and a unique crossover function is that the unique crossover ensures the uniqueness of the individuals in the population. This function creates new generations from parent itineraries by combining their tour points. The function concatenates the tour points from both parents, removes duplicates and randomly selects tour points until the child itinerary is complete. This ensures genetic diversity while making sure the points are unique:

```
Function UniqueCrossover(parent1, parent2)
  child ← empty list
  potentialPoints ← Unique(Concatenate(parent1, parent2))
  While Length(child) < Length(parent1)
    point ← RandomChoice(potentialPoints)
    Add point to child
    Remove point from potentialPoints
  End While
  Return child
End Function.
```

3.5. Mutation without repeats

The function randomly selects two positions in the itinerary and swaps corresponding tour points. The mutation process is vital for covering as much as possible from the search space and

preventing the premature convergence to suboptimal solutions. By design, the function does not introduce duplicate tour points:

```
Function MutationWithoutRepeats(individual)
  idx1, idx2 ← RandomSample(Range(Length(individual)), 2)
  Swap(individual[idx1], individual[idx2])
End Function.
```

4. Integrating real time data

In the paper "A Novel Urban Tourism Path Planning Approach Based on a Multiobjective Genetic Algorithm" by Damos et al., 2021, the authors use a multi-objective genetic algorithm to optimize urban tourism routes based on various static factors such as entertainment value, scientific value, and quality of service. The approach simultaneously optimizes multiple objectives, ensuring a balance between various aspects of the tourism experience.

While multi-objective optimization is powerful, integrating real-time data provides several key advantages, such as dynamic adaptability, where real time data integration allows the algorithm to adapt to current live conditions and ensure the planned route remains optimal even when the conditions change.

One important advantage is the seamless operation, which doesn't require manual intervention, allowing the user to focus more on the experience instead of manually adjusting parameters.

APIs are used to bring in real time data, for the weather and traffic conditions (updated distances), and the Fitness function is extended:

```
Function Fitness(individual, tourPoints, durationLimit, penaltyRate)
  totalRating ← 0
  totalDuration ← 0

  For i ← 0 to Length(individual) - 2
    start ← individual[i]
    end ← individual[i + 1]

    point ← tourPoints[start]
    (real_time_duration, real_time_distance) ← get_real_time_traffic_data(start, end)
    weather_penalty ← get_real_time_weather_penalty(end)
    totalRating += point.rating
    totalDuration += real_time_duration + weather_penalty
  End For
  If totalDuration > durationLimit Then
    penalty ← (totalDuration - durationLimit) * penaltyRate
    totalRating -= penalty
  End If

  Return totalRating
End Function.
```

5. Experimental analysis

With a reasonable population size of 14-20 tour points, the algorithm manages to generate solutions with a very high fitness score, but random enough to satisfy the constraint of having a dynamic experience for each individual tourist. The number of generations can be increased beyond 1000 with little performance impact, the only added value being the gain in the stability of the solutions, when specific tour points are mentioned more often in the later generations, which translates into popular attractions that are usually highly rated being suggested more often.

A very similar level of fitness is achieved with less than 5000 generations and even with 5000 generations. Even up to 10000 generations, considering an 8 hour time limit and only a single day itinerary, the time in which the solution is found is negligible.

Using the following input data, and expecting at least five points to be included while trying to fit the highest rating points as well, a possible solution could be seen in Table 2:

Table 2. Input data for the GA

Rating	Duration (min)	Name
8	90	Historical Museum
10	60	Royal Botanical Gardens
6	45	Artisan Market
9	120	Ancient Castle Tour
7	30	City Viewpoint
8	75	Cultural Heritage Museum
5	20	Local Brewery
9	55	Modern Art Gallery
8	80	Seaside Promenade
10	120	National Park
7	50	Aquarium Visit
8	40	Historic Church
6	60	Science Center
9	70	Botanic Garden

The total duration of the itinerary (the first five points) is 375 minutes, and there are two tour points with a rating value of 10, one with 9, one with 8 and one with 7, as registered in Table 3. By doing a simple calculation, it is obvious that higher rating tour points have not been selected, because the time limit would have been exceeded and a lot of spare time would have been left.

Table 3. Example generated itinerary

Rating	Duration (min)	Name
7	50	Aquarium Visit
10	120	National Park
10	60	Royal Botanical Gardens
9	55	Modern Art Gallery
8	90	Historical Museum

To enhance the effectiveness of the route planning algorithm, real-time data, like traffic conditions (or restrictions) and weather updates were integrated. This dynamic adjustment improves the accuracy and reliability of the generated itineraries.

To demonstrate the impact of integrating real-time data, an experiment was conducted with an initial population of 20 tour points. The algorithm was run for 1000 generations, and the fitness score was measured, the total duration and the number of high-rating points included in the itinerary. The results can be seen in Table 4.

Table 4. Results of real time data experiment

Generation	Fitness Score (Static)	Fitness Score (Real-Time Data)	Total Duration (minutes, Static)	Total Duration (minutes, Real-Time Data)	High-Rating Points (Static)	High-Rating Points (Real-Time Data)
100	45	50	380	370	3	4
100	47	52	370	360	4	4
300	48	54	365	355	4	5
400	49	55	360	350	4	5
500	50	56	355	345	4	5
600	51	57	350	340	4	5
700	52	58	345	335	5	5
800	53	59	340	330	5	5
900	54	60	335	325	5	5
1000	55	61	330	320	5	5

In the first experiment some lower ranked points were penalized and traffic delays were introduced for them, which resulted in a tour with higher ranked points and lower duration. Considering that in reality those higher rated points could have been bad weather, for example, generations for such cases were also run, because outdoor attractions are prone to bad weather, which could reflect in the travel times to the attraction. The results can be seen in Table 5.

Table 5. Results of second real time data experiment

Generation	Fitness Score (Static)	Fitness Score (Real-Time Data)	Total Duration (minutes, Static)	Total Duration (minutes, Real-Time Data)	High-Rating Points (Static)	High-Rating Points (Real-Time Data)
100	45	42	380	400	3	2
100	47	43	375	395	4	3
300	48	44	370	390	4	3
400	49	46	365	385	4	3
500	50	47	360	380	4	4
600	51	48	355	375	4	4
700	52	49	350	370	5	4
800	53	50	345	365	5	4
900	54	51	340	360	5	4
1000	55	52	335	355	5	4

Although the generated solutions do not have a higher rating compared to the generations that used only the static constraints, it is noticeable that introducing real time data lowers the number of higher rated attractions visited, but it does, in the end, generate a more suitable itinerary for the guide to follow, which will result in a more satisfactory experience for the user, because, for

example, visiting an outdoor location in the rain would not be something that a tourist would appreciate, and falling back on an indoor location with a lower rating would be more enjoyable.

An empirically generated optimal tour that focuses on maximizing the total attraction rating within the time constraints was selected, too. The identified itinerary looks as follows:

- Royal Botanical Gardens (Rating: 10, Duration: 60 min);
- National Park (Rating: 10, Duration: 120 min);
- Modern Art Gallery (Rating: 9, Duration: 55 min);
- Botanic Garden (Rating: 9, Duration: 70 min);
- Historic Church (Rating: 8, Duration: 40 min);
- City Viewpoint (Rating: 7, Duration: 30 min).

This optimal tour achieves a combined rating of 53 and a duration of 375 minutes. The GA-generated tour has a total rating of 44 and a total duration of 375 minutes. Although the rating is lower than that of the optimal tour, the GA succeeded in selecting high rated attractions and it respected the time constraint. It took into consideration the traffic and weather data as well, which is not a departure from the optimal tour.

6. Discussions and conclusions

The analysis of the algorithm with the simplest constraints leads to the conviction that more realistic constraints could be introduced without highly affecting the compute time.

The efficiency of the algorithm can be its great advantage since it would need as little resources as possible while being able to dynamically readjust to unforeseen shifts in the itinerary, one of the potential use cases.

The introduction of the real time traffic and weather data proved to be useful in showing that various kinds of real time data could easily be transformed into parameters to refine the algorithm.

Other factors that are to be taken into consideration are the fixed schedule restrictions that some attractions might have, seasonal restrictions and the maximum amount of visitors that can be accommodated. This would introduce the need for a real time connection with ticketing platforms or other means of dynamically gathering these data.

A future research direction is to introduce requirements for tourists with special mobility needs. Autonomous tour guides are a good option for such tourists, because they can come packed with enhanced safety features.

The algorithm can be extended to take into account the physical state of the occupant (for example, if they become fatigued), to consider preferences for rest stops, quieter routes or specific areas that are generally easier to navigate, and reroute according to these factors.

It is very easy to integrate the algorithm in a more complex dynamic travel packaging system, since it is a very small and efficient piece of software without any specific requirements.

The simplicity of the algorithm makes it very easy to experiment with different scenarios and generate compelling results in the long term. It also leads to palpable pragmatic uses for companies and tour guides, and is yet another small component in a growing ecosystem of software solutions and pragmatic niche applications for autonomous vehicles.

REFERENCES

Applegate, D. L., Bixby, R. E., Chvátal, V. & Cook, W. (2007) *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Azad, M., Hoseinzadeh, N., Brakewood, C., Cherry, C. R. & Han, L. D. (2019) Fully Autonomous Buses: A Literature Review and Future Research Directions. *Journal of Advanced Transportation*. 2019, 4603548, 1-16. doi:10.1155/2019/4603548.

Bezai, E. N., Benachir, M., Al-Habaibeh, A., Larbi, M. C. & Fodil, F. (2021) Future Cities and Autonomous Vehicles: Analysis of the Barriers to Full Adoption. *Energy and Built Environment*. 2(1), 1-52. doi:10.1016/j.enbenv.2020.05.002.

Damos, M. A., Zhu, J., Li, W., Hassan, A., & Khalifa, E. (2021) A novel urban tourism path planning approach based on a multiobjective genetic algorithm. *ISPRS International Journal of Geo-Information*. 10(8), Article 530. https://doi.org/10.3390/ijgi10080530.

Hosany, S., Sthapit, E. & Björk, P. (2022) Memorable Tourism Experience: A Review and Research Agenda. *Psychology & Marketing*. 39(8), 1467–1486. doi:10.1002/mar.21665.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (2012) Optimization by Simulated Annealing. *Science*. 220 (4598), 671–680.

Kumar, A. J. S., Arunadevi, J. & Mohan, V. (2009). Intelligent transport route planning using genetic algorithms in path computation algorithms. *European journal of scientific research*. 25(3), 463-468.

Li, Y., Deng, W., Zheng, Y. & Zhu, Y. (2016) An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Traveling Salesman Problem. *International Journal of Computers Communications & Control*. 11(3), 401–413.

Pehlivanoglu, Y. V. & Pehlivanoglu, P. (2021) An Enhanced Genetic Algorithm for Path Planning of Autonomous UAV in Target Coverage Problems. *Applied Soft Computing*. 112, 107796. doi:10.1016/j.asoc.2021.107796.

Rong, H. H., Tu, W., Duarte, F. & Ratti, C. (2020). Employing waterborne autonomous vehicles for museum visits: A case study in Amsterdam. *European Transport Research Review*. 12(1), 1-12. https://doi.org/10.1186/s12544-020-00459-x

Tang, J., Pan, J.-S. & Wang, S. (2017) Hybrid Genetic Algorithm and Tabu Search for the Dynamic Shortest Path Routing Problem with Two Objectives. *Applied Soft Computing*. 53, 760–771.

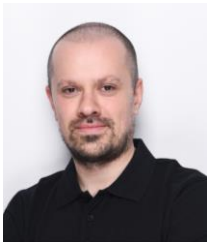
Wang, H. & Chen, Y. (2015) A Genetic Algorithm for Solving Dynamic Traveling Salesman Problem. *International Journal of Simulation Modelling*. 14(1), 138–147.

Yan, S. & Pan, F. (2019) Research on Route Planning of AUV Based on Genetic Algorithms. In *2019 IEEE International Conference on Unmanned Systems and Artificial Intelligence (ICUSAI), Xi'an, China, 2019*. IEEE. pp. 184-187. doi:10.1109/ICUSAI47366.2019.9124785.

Yu, H. & Lu, F. (2012) A Multi-Modal Route Planning Approach with an Improved Genetic Algorithm. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. 38, Part II.

Zhang, J., Tang, J. & Zheng, J. (2019) Adaptive Genetic Algorithm for the Dynamic Traveling Salesman Problem with Traffic Factors. *Applied Soft Computing*. 85, 105778.

Zhou, X., Su, M., Liu, Z. & Zhang, D. (2019) Smart Tour Route Planning Algorithm Based on Clustering Center Motive Iteration Search. *IEEE Access*. 7, 185607–185633. doi:10.1109/ACCESS.2019.2960761.



Cristian SANDU is a doctoral student and assistant professor in the Computer Science Department of the "Dunărea de Jos" University of Galati, with a strong interest in artificial intelligence and autonomous vehicles. He published numerous articles in local and international scientific journals on topics such as edge computing, autonomous vehicles and the application of their technologies in everyday assistance.

Cristian SANDU este doctorand și asistent universitar în cadrul Departamentului de Calculatoare al Universității „Dunărea de Jos” din Galați, cu un interes accentuat pentru inteligența artificială și vehiculele autonome. A publicat numeroase articole în reviste științifice locale și internaționale abordând subiecte precum edge computing, vehicule autonome și aplicarea tehnologiilor acestora în asistența cotidiană.

Luminița DUMITRIU is a professor in the Department of Computer Science of the "Dunărea de Jos" University of Galati, Romania. Author of more than 50 research articles and books, being frequently cited in scientific papers. Her research interests include Big Data, algorithms and autonomous vehicles.

Luminița DUMITRIU este profesor în cadrul Departamentului de Calculatoare al Universității „Dunărea de Jos” din Galați, România. Autor a peste 50 de articole de cercetare și cărți, citat frecvent în lucrări științifice. Domeniile sale de interes în cercetare includ Big Data, algoritmi și vehicule autonome.

Ioan ȘUȘNEA is a professor in the Computer Science Department of the "Dunărea de Jos" University of Galati, Romania. His research interests include robotics, multi-agent systems, emerging processes and contemporary education. He has published over 60 research articles and books.

Ioan ȘUȘNEA este profesor în cadrul Departamentului de Calculatoare al Universității „Dunărea de Jos” din Galați, România. Domeniile sale de interes în cercetare includ robotica, sisteme multi-agent, procese emergente și educația contemporană. A publicat peste 60 de articole de cercetare și cărți.