Deep Learning Assisted Memetic Algorithm for Shortest Route Problems

Ayad Turky[1(✉)], Mohammad Saiedur Rahaman[1], Wei Shao[1], Flora D. Salim[1], Doug Bradbrook[2], and Andy Song[1]

[1] School of Science, RMIT University, Melbourne, VIC 3000, Australia
{ayad.turky,saiedur.rahaman,wei.shao,flora.salim,andy.song}@rmit.edu.au
[2] Mornington Peninsula Shire Council, Rosebud, VIC 3939, Australia
doug.bradbrook@mornpen.vic.gov.au

Abstract. Finding the shortest route between a pair of origin and destination is known to be a crucial and challenging task in intelligent transportation systems. Current methods assume fixed travel time between any pairs, thus the efficiency of these approaches is limited because the travel time in reality can dynamically change due to factors including the weather conditions, the traffic conditions, the time of the day and the day of the week, etc. To address this dynamic situation, we propose a novel two-stage approach to find the shortest route. Firstly deep learning is utilised to predict the travel time between a pair of origin and destination. Weather conditions are added into the input data to increase the accuracy of travel time predicition. Secondly, a customised Memetic Algorithm is developed to find shortest route using the predicted travel time. The proposed memetic algorithm uses genetic algorithm for exploration and local search for exploiting the current search space around a given solution. The effectiveness of the proposed two-stage method is evaluated based on the New York City taxi benchmark dataset. The obtained results demonstrate that the proposed method is highly effective compared with state-of-the-art methods.

Keywords: Shortest route problems · Memetic algorithm · Deep learning · Travel times

1 Introduction

Finding shortest routes is crucial in intelligent transportation systems. Shortest route information can be utilised to enable route planners to compute and provide effective routing decisions [8,11,14,16,24]. However, shortest route computation is a challenging task partially due to dynamic environments [3]. For instance, the shortest path is impacted by various spatio-temporal factors, which are dynamic in nature, including weather, the time of the day, and the day of the week. That makes the current shortest route computation techniques ineffective [3,7]. Moreover, it is a challenging problem to incorporate these dynamic factors into shortest route computation.

V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12141, pp. 109–121, 2020.
https://doi.org/10.1007/978-3-030-50426-7_9

In recent years, the proliferation of pervasive technologies has enabled the collection of spatio-temporal big data associated with user mobility and travel routes in a real-time manner [15]. Modern cars are equipped with telematics devices including in-car GPS (Global Positioning System) devices which can be used as a source of valuable information in traffic modelling [23]. The traces generated from GPS devices has been leveraged by many scenarios such as Spatio-temporal context recognition, taxi-passenger queue time prediction, study of city dynamics and transport demand estimation [3,12,13,17,23].

One important aspect of finding shortest routes in realistic environments, which are inherently dynamic, is travel time prediction [8,22]. Due to the dynamic nature of in the travel routes, traditional machine learning methods cannot be applied directly onto travel time prediction. One of the key challenge for traditional machine learning models is the unavailability of hand-crafted features which requires substantial involvement of domain experts. One relevant approach is the recent use of evolutionary algorithms in other domains to work along with deep learning models for effective feature extraction and selection [18–21]. In this study, we aim to identify relevant features for shortest route finding between an origin and destination, leveraging the auto-feature generation capability of deep learning. Thereby we propose a novel two-stage architecture for the travel time prediction and route finding task. In particular we design a customized memetic algorithm to find shortest route based on the predicted travel time from the earlier stage. The contributions of this research are summarised as follows:

- A novel two-stage architecture for the shortest route finding under dynamic environments.
- Development of a deep learning method to predict the travel time between a origin-destination pair.
- A customised memetic algorithm to find shortest route using the predicted travel time.

The rest of the paper is organized as follows. In Sect. 2, we present our proposed methodology for this study. Section 3 describes the experimental settings which is followed by the discussion of experimental results in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 Proposed Methodology

In this paper, we propose a deep learning assisted memetic algorithm to solve the shortest route problems. The proposed method has two stages which are (1) prediction stage and (2) optimisation stage. The prediction stage is responsible to predict the travel times between a pair of origin and destination along the given route by using deep learning. The second stage uses memetic algorithm to actually find the shortest path to visit all locations along the given route. In the following subsections, we discuss the main steps of the proposed method and the components of each stage in detail. Figure 1 shows our proposed approach.

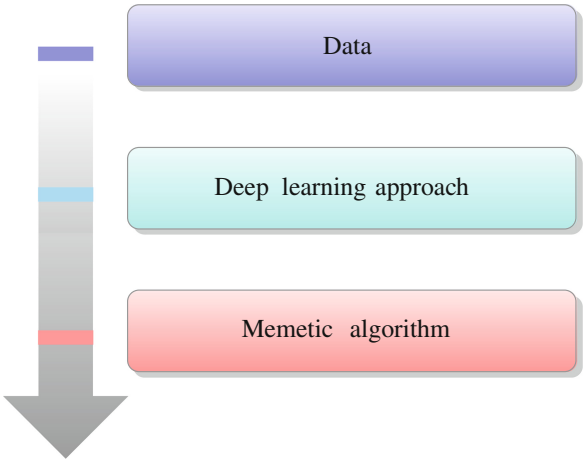

Fig. 1. Flowchart of the proposed two-stage approach

2.1 Prediction Stage

Conventional route finding methods assume fixed cost or travel time between any pairs of points. That is rarely the case in reality. One approach to the dynamic travel time issue is prediction. In this work, we incorporate the weather data along with the temporal-spatial data to develop a deep learning predictive approach. The goal of the proposed predictive approach is to predict future travel time between any points in the problem based on historical observations and weather condition. Specifically, given a group of historical travel time data, weather data and road network data, the aim is to predict travel time between source (s) and destination (d) $s_i, d_i \in R, i \in [1,2, ..., n]$, where n is the number of locations in the road network. Our predictive approach tries to predict the travel time at $t+1$ based on the given data at t. The proposed predictive approach has three parts: input data, data cleaning and aggregation, the prediction approach. Figure 2 shows the deep learning approach.

Input Data. In this work, we use data from three different sources. The data involves around 1.5 million trip records. These include the travel time data, weather data and road network data.

- **Travel time data**. The travel times between different locations were collected using 2016 NYC Yellow Cab trip record data.
- **Weather data**. We use the weather data in New York City - 2016. The data involves: date, maximum temperature, minimum temperature, average temperature, precipitation, snow fall and snow depth.
- **Road network data**. The road network data involves temporal and spatial information as follows:
  - Id - a trip identifier.

- Vendor_id - a code indicating whether the provider is involved with the trip record.
- Pickup_date-time - date and time when the meter was started.
- Drop-off_date-time - date and time when the meter was disconnected.
- Passenger_count - indicates the total number of riders in the vehicle.
- Pickup_longitude - the longitude of picked passenger.
- Pickup_latitude - the latitude of the picked passenger.
- Dropoff_longitude - the longitude of the dropped passenger.
- Dropoff_latitude - the latitude of the dropped passenger.
- Store_flag - indicates if the trip record was saved in vehicle memory before sending to the vendor where Y = store and forward; N = not a store and forward trip.
- Trip_duration - duration of the trip in seconds.

Data Preparation. This process involves removal of all error values, outliers, imputation of missing values and data aggregation. To facilitate the prediction we bound the data ranges between $(average + 2) \times standard\_deviation$ to $(average - 2) \times standard\_deviation$. Values outside of these ranges are considered as outliers and are removed. The missing values are imputed by the average values. Any overlapping pick-up and drop-off locations are also removed. In the aggregation step, we combine the travel time data, weather data and road network each time step so that it can be fed into our deep networks.

Prediction Approach. The main goal of this step is to provide high accuracy prediction of the travel times between different locations in the road network. The processed and aggregated data is provided as an input for the prediction approach. Once the prediction model is trained and retrieved, it is then ready to actually predict the travel times between given locations.

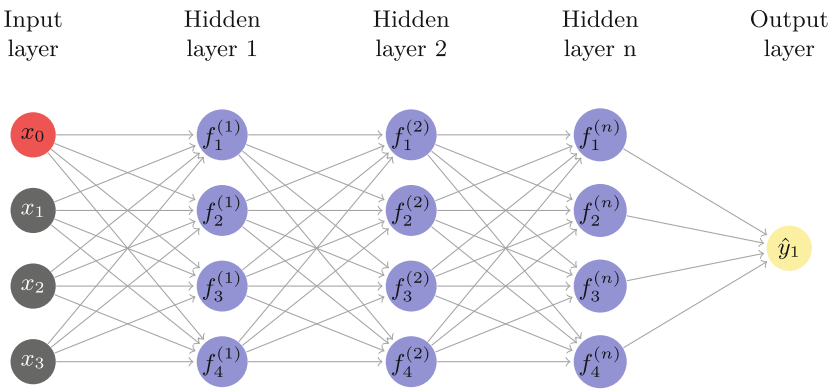

Fig. 2. Illustration of the deep network based prediction model

In this work, we propose a deep learning technique based on feedforward neural network to build our prediction approach. The deep neural network consists of one input layer, multiple hidden layers and one output layer. Each layer (input, hidden and output) involves a set of neurons. The total number of neurons in the input layer is same as the number of input variables in our input data. The output layer has one single neuron which represents the predicted value. In deep neural network, we have m number of hidden layers and each one has k number of neurons. The input layer takes the input data and then feed them into the hidden layers. The output of the hidden layers are used as an input for the output layer. Given the input data X ($X = x_1, .. x_n$) and the output value Y, the prediction approach aims to find the estimated value Y_{est} using a simple approach is as follows:

$$Y_{est} = x_1 w_1 + x_2 w_2 + x_3 w_3 + b \tag{1}$$

Where w is the weight and b is the bias. Using a four-layer (one input, two hidden and one output) neural network as example, the Y_{est} can be calculated as follows:

$$Y_{est} = x_1^{(4)} = f(w_{11}^{(2)} x_1^{(2)} + w_{12}^{(2)} x_2^{(2)} + w_{13}^{(2)} x_3^{(2)} + b_1^{(2)}) + f(w_{21}^{(3)} x_1^{(3)} + w_{22}^{(3)} x_2^{(3)} + w_{23}^{(3)} x_3^{(3)} + b_1^{(3)}) \tag{2}$$

Where is the $x_1^{(4)}$ is the output of the network and f is the activation function. In this work, Keras [1] based on TensorFlow [2] is used to develop our predication model.

2.2 Optimisation Stage

This subsection presents the proposed memetic algorithm (MA) for shortest route problems. MA is a population-based metaheuristic that combines the strengths of local search algorithm with population-based metaheuristic to improve the convergence process [9,10]. In this paper, we used genetic algorithm (GA) and local search (LS) algorithm to form our proposed MA. GA is responsible for exploring new areas in the search space of solutions. LS is used to accelerate the search convergence. The pseudocode of the proposed MA is presented in is shown in (1). The overview of the process is given below followed by detailed description of these steps.

Our proposed algorithm starts from setting parameters, creating a population of solutions, calculating the quality of each solution and identifying the best solution in the current population. Next, the main steps of MA will iterate over a number of generations until the stopping criterion is met. At each generation, good solutions are selected from the population by the selection procedure. Then the crossover operator is applied on the selected solutions to generate new solutions. After that the mutation operator is applied on the new solutions by randomly changing them. A repair procedure is applied to check the feasibility of the generated solutions and fix the infeasible solutions as some solutions are no longer feasible. Afterwards a local search algorithm is invoked to iteratively

Algorithm 1: The proposed memetic algorithm

Input: Population size, PS, crossover rate, CR, mutation rate, MR, the maximum number of generations, Max_G and consecutive non-improvement iterations;
Set P_Sol=Randomly generate a population of solutions (PS) ;
Evaluate the population of solutions;
Set $iter$=0;
while $iter < Max\_G$ **do**
    /*Selection procedure*/ ;
    FirstParent= Select_one_individual (P_Sol);
    SecondParent= Select_one_individual (P_Sol);
    /*Check the crossover probability*/;
    if $Rand[0,1] < CR$ **then**
        /*Apply the crossover operator*/;
        $Offsprings_{cx}$= Crossover(FirstParent, SecondParent);
    end
    /*Check the mutation probability*/;
    if $Rand[0,1] < MR$ **then**
        /*Apply the mutation operator*/;
        $Offsprings_{mutation}$=Mutate($Offsprings_{cx}$);
    end
    /*Apply local search to Offsprings */;
    Offsprings=LS($Offsprings_{mutation}$);
    Update the population (P_Sol);
    $iter = iter + 1$;
end
Output Best solution found ;

improve the current solutions. If one of the stopping criteria is satisfied, then the whole MA procedure will stop and the current best solution will be returned as the output. Otherwise, the fitness of the current pool of solutions will be calculated. Then the population is updated since new solutions have been generated by crossover, mutation, repair procedure and local search. After that a new iteration starts from the selection procedure again.

Set Parameters. The main parameters of the proposed MA are initialised in this step. The proposed MA has several parameters. These are: population size, the number of generations, crossover rate, mutation rate and the number of non improvement iterations for the local search.

Initial Population. The initial population is randomly generated. Each solution is represented as one chromosome, e.g. one-dimensional array. Each cell of the array contains an integer number which represent the location.

Fitness Function. In this step, the fitness value of each solution based on the objective function is calculated. The better the fitness value is, the higher chance the solution will be selected to reproduce the next generation of solutions. For shortest route problems, the fitness is the total travel time between the origin and destination locations. Therefore, solution with shortest travel time is the better.

Selection Procedure. This step is responsible for selecting two solutions for producing the next generation. In this paper, we adopted the traditional tournament selection mechanism [4–6]. The tournament size is set to 2, indicating that each tournament has two solutions competing with each other. At each call, two solutions are randomly selected from the current population and the one with highest fitness value will be added to the reproduction pool.

Crossover. This step is responsible to generate new solutions by taking the selected solutions and mixes their genetic materials to produce new offsprings. In this paper, single-point crossover method is used which only swap genetic materials at one point [5,6]. It first finds a common point between source node and destination node and then all points behind the common point are exchanged between the two solutions, thus resulting in two offspring's.

Mutation. Mutation operator helps explore a large search space by producing some random changes in various solutions. In this paper, we used a one-point mutation operator [5]. Crossover point is randomly selected and then all points behind the selected mutation point are changed with a random sequence.

Repair Procedure. The aim of this step is to turn infeasible solutions into feasible ones. After crossover and mutation operations, the resulting solutions may become infeasible [5,6]. In this paper, The MA in our experiments has repair procedure that ensure all infeasible solutions are repaired.

Local Search Algorithm. The main role of this step is to improve the convergence process of the search process in order to attain higher quality solutions [9,10]. In this paper, the utilised local search algorithm is the steepest descent algorithm. Steepest descent algorithm is a simple variation of the gradient descent algorithm. It starts with a given solution as an input and uses a neighbourhood structure to move the search process to other possibly better solutions. It uses an "accept only" improving acceptance criterion whereby only a better solution will be used as a new starting point. Given s_i, It applies a neighbourhood structure to create s_n. Replace s_n with s_i if s_n is better. The pseudocode of the steepest descent algorithm is shown in (2).

Algorithm 2: Steepest descent algorithm

```
Set MaxIter; Iter = 0;
s_i ← GenerateInitialSolution;
while Iter <MaxIter do
    s_n ← apply neighbourhood structure to s_i;
    if f(s_n) <= f(s_i) then
        s_i ← s_n;
    end
end
Return the best solution;
```

Stopping Condition. If the stopping condition is met, terminate the search process and return the best found solution. For our proposed memetic algorithm, it will stop if the maximum number of generations is reached. Otherwise, go to step 24.

Table 1. The parameter settings of the deep learning approach

Parameter	Value
Number of input parameters	12
Number of output parameters	1
Number of hidden layers	2
Hidden units in each layer	45, 35
Activation function	ReLU

Table 2. The parameter settings of the memetic algorithm

Parameter	Tested range	Suggested value
Number of generations	5−100	40 fitness evaluations
Population size	5−30	20
Crossover rate	0.1−0.9	0.4
Mutation rate	0.1−0.9	0.2
Consecutive non-improvement iterations	5−20	10

3 Experimental Settings

In this section, the parameter settings of the deep learning and the proposed algorithm are provided. The values of parameters were selected empirically based

on our preliminary experiments, where we tested the deep learning model and the proposed algorithm with different parameter combination using different values for each parameter. The values of these parameters are determined one by one through manually changing the value of one parameter, while fixing the others. Then, the best values for all parameters are recorded. The final parameter values of the deep learning and the proposed algorithm are presented in Tables 1 and 2.

4 Experimental Results

This section is divided into two subsections. The first examines the performance comparison between the deep learning approach and other machine learning models (Sect. 4.1). The second assesses the benefit of incorporating the proposed components on search performance (Sect. 4.2).

4.1 Deep Learning and Machine Learning Results

In this paper, we have implemented a number of machine learning models and the results of these models are compared with the deep learning model proposed in this work. We have tested the followings methods: XGBoost, Random forest, Artificial neural network, Multivariate regression.

The root-mean squared-error (RMSE) was used as an evaluation metric. Table 3 shows the results in term of RMSE on the NYC Taxi dataset.

Table 3. Comparing our deep prediction model with other machine learning models in term of RMSE

Model	RMSE
XGBoost	24.06
Random forest	21.34
Artificial neural network	70.21
Multivariate regression	27.19
Our deep learning model	**11.01**

In the table, the best obtained result is highlighted in bold. From Table 3, it can be seen that our deep prediction model is superior to the other machine learning models in term of RMSE. The best values with the lowest RMSE is 11.01 achieved by our approach, followed by 21.34 from random forest, 24.06 from XGBoost, 27.19 from multivariate regression and 70.21 from artificial neural network.

This good result can be attributed to the factor that deep learning consider all input features and then utilise best ones through the internal learning process. On the other hand, other machine learning methods require feature engineering step to identify the best subset of features which is a very time consuming and needs a human expert.

4.2 The Proposed Memetic Algorithm Results

This section evaluates the effectiveness of the machine learning models and the proposed memetic algorithm. To this end, genetic algorithm (GA) and memetic algorithm (MA) with different machine learning models are tested and compared against each other. These are: GA with XGBoost, GA with random forest, GA with artificial neural network, GA with multivariate regression, GA with deep prediction model, MA with XGBoost, MA with random forest, MA with artificial neural network, MA with multivariate regression and MA with deep prediction model. The main aim is to evaluate the benefit of using our deep prediction model and local search algorithm within MA.

Table 4. Results of the GA and MA with different prediction models (Part I)

Algorithm	Number of locations			
	500		1000	
	Best	std	Best	std
GA with XGBoost	5933.61	243.76	7671.84	136.72
GA with random forest	5844.78	194.14	7533.67	153.98
GA with artificial neural network	9401.86	215.61	10154.01	707.65
GA with multivariate regression	6217.77	227.02	7907.34	256.39
GA with deep prediction model	4602.03	153.69	6837.17	142.24
MA with XGBoost	5774.54	116.07	7405.12	120.02
MA with random forest	5637.81	96.16	7360.04	119.18
MA with artificial neural network	9293.44	196.01	10093.88	693.31
MA with multivariate regression	6171.63	138.09	7499.22	155.03
MA with deep prediction model	**3234.11**	**71.3**	**6411.72**	**127.69**

To ensure a fair comparison between the compared algorithms, the initial solution, number of runs, stopping condition and computer resources are the same for all instances. All algorithms were executed for 30 independent runs over all instances. We also used 4 instances with a different number of locations ranging between 500 and 2000 locations, which can be seen as small, medium, large and very large.

The computational comparisons of the above algorithms are presented in Tables 4 and 5. The comparison is in terms of the best cost (travel time) and standard deviation (std) for each number of locations, where the lower the better. The best results are highlighted in bold. A close scrutiny of Tables 4 and 5 reveals that, of all the instances, the proposed MA algorithm with deep learning approach outperforms the other algorithms in all instances. From Tables 4 and 5, we can make the following observations:

– GA with deep prediction model obtained better results when compared to GA with all other prediction models across all instances.

Table 5. Results of the GA and MA with different prediction models (Part II)

Algorithm	Number of locations			
	1500		2000	
	Best	std	Best	std
GA with XGBoost	12908.11	998.91	13634.01	501.94
GA with random forest	12833.67	1076.14	13593.47	529.61
GA with artificial neural network	15751.01	1703.07	15882.33	992.84
GA with multivariate regression	13012.41	1047.41	13809.41	591.56
GA with deep prediction model	10571.09	607.99	12946.67	388.14
MA with XGBoost	12641.08	613.12	13436.42	481.08
MA with random forest	12607.91	684.74	13309.01	503.44
MA with artificial neural network	15603.17	980.01	15206.93	755.31
MA with multivariate regression	12988.14	721.36	13498.96	564.05
MA with deep prediction model	**9561.36**	**131.29**	**12721.45**	**129.15**

- GA with deep learning obtained better results when compared to MA with all other machine learning models (apart from MA with deep learning) across all instances.
- MA with deep learning obtained better results when compared to GA and MA with all other machine learning models across all instances.

This justifies the benefit of using deep learning approach to predict the travel time and the proposed memetic algorithm to exploit the current search space around the given solution.

5 Conclusion

In this study, we proposed a novel two-stage approach for finding the shortest route under dynamic environment where travel time changes. Firstly, we developed a deep learning method to predict the travel time between the origin and destination. We also added the weather conditions into the input to demonstrate that our approach can predict the travel time more accurately. Secondly, a customised memetic algorithm is developed to find shortest route using the predicted travel time. The effectiveness of the proposed method has been evaluated on New York City taxi dataset. The obtained results lead to our conclusion that the proposed two-stage shortest route is effective, compared with conventional methods. The proposed deep prediction model and memetic algorithm are beneficial.

Acknowledgements. This work is supported by the Smarter Cities and Suburbs Grant from the Australian Government and the Mornington Peninsula Shire Council.

References

1. Chollet, F.: Keras (2015). https://keras.io. Accessed 24 Dec 2019
2. Devin, M. et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org. Accessed 24 Dec 2019
3. Fu, T., Lee, W.: DeepIST: deep image-based spatio-temporal network for travel time estimation. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 69–78. ACM (2019)
4. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Found. Genet. Algorithms **1**, 69–93 (1991)
5. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. **3**(2), 95–99 (1988)
6. Holland, J.H.: Genetic algorithms. Sci. Am. **267**(1), 66–73 (1992)
7. Lan, W., Xu, Y., Zhao, B.: Travel time estimation without road networks: an urban morphological layout representation approach. arXiv preprint arXiv:1907.03381 (2019)
8. Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., Liu, Y.: Multi-task representation learning for travel time estimation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1695–1704. ACM (2018)
9. Moscato, P., et al.: On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Comput. Program, C3P Rep. **826**, 1989 (1989)
10. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: a literature review. Swarm Evol. Comput. **2**, 1–14 (2012)
11. Qin, K.K., Shao, W., Ren, Y., Chan, J., Salim, F.D.: Solving multiple travelling officers problem with population-based optimization algorithms. Neural Comput. Appl. 1–27 (2019). https://doi.org/10.1007/s00521-019-04237-2
12. Rahaman, M.S., Hamilton, M., Salim, F.D.: Predicting imbalanced taxi and passenger queue contexts in airport. In: PACIS, p. 172 (2017)
13. Rahaman, M.S., Hamilton, M., Salim, F.D.: Queue context prediction using taxi driver knowledge. In: Proceedings of the Knowledge Capture Conference, p. 35. ACM (2017)
14. Rahaman, M.S., Hamilton, M., Salim, F.D.: Coact: a framework for context-aware trip planning using active transport. In: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 645–650. IEEE (2018)
15. Rahaman, M.S., Hamilton, M., Salim, F.D.: Using big spatial data for planning user mobility. In: Sakr, S., Zomaya, A. (eds.) Encyclopedia of Big Data Technologies, pp. 1–6. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77525-8
16. Rahaman, M.S., Mei, Y., Hamilton, M., Salim, F.D.: CAPRA: a contour-based accessible path routing algorithm. Inf. Sci. **385**, 157–173 (2017)
17. Rahaman, M.S., Ren, Y., Hamilton, M., Salim, F.D.: Wait time prediction for airport taxis using weighted nearest neighbor regression. IEEE Access **6**, 74660–74672 (2018)
18. Sabar, N.R., Turky, A., Song, A., Sattar, A.: Optimising deep belief networks by hyper-heuristic approach. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2738–2745. IEEE (2017)
19. Sabar, N.R., Turky, A., Song, A., Sattar, A.: An evolutionary hyper-heuristic to optimise deep belief networks for image reconstruction. Appl. Soft Comput. 105510 (2019). https://doi.org/10.1016/j.asoc.2019.105510

20. Song, H., Qin, A.K., Salim, F.D.: Evolutionary model construction for electricity consumption prediction. Neural Comput. Appl. 1–18. https://doi.org/10.1007/s00521-019-04310-w
21. Song, H., Qin, A.K., Salim, F.D.: Multi-resolution selective ensemble extreme learning machine for electricity consumption prediction. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.-S.M. (eds.) ICONIP 2017. LNCS, vol. 10638, pp. 600–609. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70139-4_61
22. Wang, D., Zhang, J., Cao, W., Li, J., Zheng, Y.: When will you arrive? Estimating travel time based on deep neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
23. Wang, H., Yang, H.: Ridesourcing systems: a framework and review. Transp. Res. Part B Methodol. **129**, 122–155 (2019)
24. Wang, Z., Fu, K., Ye, J.: Learning to estimate the travel time. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 858–866. ACM (2018)